Some Nonparametric Hybrid Predictive Models: Asymptotic Properties and Applications

Tanujit Chakraborty

Statistical Quality Control and Operations Research Unit

Indian Statistical Institute, Kolkata

A thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Quality, Reliability and Operations Research

Thesis Supervisor: Prof. Ashis Kumar Chakraborty

November, 2020

This thesis is dedicated to
the memory of
Professor C. A. Murthy
(1958–2018)

Acknowledgements

I want to begin by thanking my Ph. D supervisor Prof. Ashis Kumar Chakraborty, for giving me the valuable opportunity to work with him. He has blessed me with his valuable insights during my Ph.D.

I would like to thank Late Prof. C.A. Murthy for all the support during my first two years of my Ph. D. He taught me Pattern Recognition and Neural Networks during my Ph.D coursework. His constant support and encouragement in the early days of Ph. D motivated me to do my research in the areas I have chosen.

I thank many other professors at Indian Statistical Institute (ISI), especially Prof. Arup K. Das, Prof. Utpal Garain, Prof. Saurabh Bhattacharya, Prof. Anil K. Ghosh, Prof. Mohan Delampady, Prof. Swagatam Das, and Prof. Boby John, for enriching me during my Ph. D.

At the same time, I have also been enriched by various academic discussions with my colleagues and seniors - Mr. Swarup Chattopadhyay, Dr. Sujoy Madhav Roy, Mr. Ashish Bakshi, Dr. Ajoy Mondal, Dr. Munmun Biswas, and Dr. Joydeep Chowdhury and would like to thank them for all the fruitful discussions. Thanks to my ISI friends Mr. Sourav Maji, Mr. Indrajit Ghosh, Mr. Satyabrata Jana, Mr. Gopal Maiti, Mr. Ashutosh Maurya, Mr. Sk Shahid Nadim, and Mr. Anurag Sau for their love, support and encouragement.

I am grateful to my mother Mrs. Rikta Chakraborty, my father Mr. Sekhar Chakraborty, my sister Ms. Tanushree Chakraborty, and my brother Mr. Tanumay Chakraborty for their immeasurable love and constant encouragement to help me complete my work.

Finally, I want to acknowledge with thanks the authorities of the Indian Statistical Institute for providing me funds during my research work and the Heads of the Statistical Quality Control and Operations Research Unit, Kolkata, for providing me the infrastructural support needed for my work.

Contents

List of Tables

List of Figures

List of Acronyms

ACF	Autocorrelation Function
ADF	Augmented Dickey Fuller
ADT	Air Dissolving Tube
AIC	Akaike Information Criterion
ANN	Artificial Neural Networks
ANT	Adaptive Neural Tree
ARIMA	Autoregressive Integrated Moving Average
ARNN	Autoregressive Neural Networks
AUC	Area Under the receiver operating characteristic Curve
BART	Bayesian Additive Regression Trees
BIC	Bayesian Information Criterion
BNN	Bayesian Neural Networks
BNT	Bayesian Neural Tree
CART	Classification and Regression Trees
CNN	Convolutional Neural Networks
CT	Classification Trees
CV	Cross-Validation
DAFSP	Dissolved Air Flotation cum Sedimentation Process
DFFNN	Deep Feedforward Neural Networks
DMST	Demineralized Water Storage Tank
DNDT	Deep Neural Decision Trees
DT	Decision Trees
EN	Entropy Nets
ENN	Edited Nearest Neighbors
FFNN	Feedforward Neural Networks
FFRE	Fiber-Filler Recovery Equipment
FMEA	Failure Mode and Effect Analysis
FNT	Flexible Neural Tree
GI	Gini Index
HDDT	Hellinger Distance Decision Tree
HDRF	Hellinger Distance Random Forest

IEP	Ion Exchange Process
IG	Information Gain
kNN	k Nearest Neighbors
KPI	Key Performance Indicator
LDA	Linear Discriminant Analysis
LR	Logistic Regression
MAE	Mean Absolute Error
MAPE	Mean Absolute Percent Error
MARS	Multivariate Adaptive Regression Spline
MBA	Master of Business Administration
MCMC	Markov chain Monte Carlo
MI	Misclassification Impurity
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NB	Naive Bayes
NDF	Neural Decision Forest
NN	Neural Nets
PACF	Partial Autocorrelation Function
RBFN	Radial Basis Function Networks
RBNT	Radial Basis Neural Tree
RF	Random Forest
RMSE	Root Mean Square Error
ROS	Random Oversampling
RPN	Risk Priority Number
RT	Regression Trees
SDP	Software Defect Prediction
SDT	Soft Decision Tree
SMOTE	Synthetic Minority Oversampling Technique
SMB	SMOTEBoost
SU	Symmetrical Uncertainty
SVM	Support Vector Machine
SVR	Support Vector Regression
TL	Tomek Links
VC	Vapnik-Chervonenkis
WRP	Waste Recovery Process

List of Notations

a.s.	almost surely.
i.i.d.	independent and identically distributed.
log	natural logarithm (base e).
$\mathbb{N}$	set of natural numbers.
$\mathbb{R}$	set of real numbers.
$\mathbb{R}_+$	set of positive real numbers.
$\mathbb{R}^p$	set of p-dimensional real vectors.
$\mathbb{B}$	Borel set on $\mathbb{R}^p$.
$\mathbf{x}_i$	data point (training data point).
n	number of (training) data points.
y_i	class label for training data points $\mathbf{x}_i$.
C	set of all possible class labels.
I_A	indicator of an event A.
$\|A\|$	cardinality of a finite set A.
Y^+	majority class in an imbalanced data.
Y^-	minority class in an imbalanced data.
$\|X_+\|$	number of training examples belonging to the majority class in an imbalanced data.
$\|X_-\|$	number of training examples belonging to the minority class in an imbalanced data.
$diam(A)$	diameter of a set $A \subset \mathbb{R}^p$.
$\underline{X} \in \mathbb{R}^p$	observation vector, vector-valued random variable.
$\underline{Y} \in \mathbb{R}$	response, real random variable.
$\underline{Y} \in \{0, 1\}$	class label, binary random variable.
$\\|\cdot\\|_1$	L_1-norm of a vector.
$\\|x\\|$	Euclidean norm of $x \in \mathbb{R}^p$.
$\\|f\\|$	$L_2(\mu)$ norm of $f : \mathbb{R}^p \to \mathbb{R}$.
$\\|f\\|_\infty$	supremum norm of function f.

L_n	training data, sequence of i.i.d. pairs that are independent of (X, Y).
λ	Lebesgue measure on $\mathbb{R}^p$.
Φ	partition and classification scheme.
$\mathcal{T}$	collection of partitions of the feature space.
$\Delta_n(\mathcal{T})$	growth function of $\mathcal{T}$.
$F_{n,k}$	class of neural networks having k hidden nodes.
g^*	Bayes decision function.
$\mu(A) = P\{X \in A\}$	probability measure of X.
$L^* = P(g^*(X) \neq Y)$	Bayes risk, the error probability of the Bayes decision.
$r(x) = E\{Y\|X = x\}$	regression function.
$\mathcal{A}$	class of sets.
$S_n(\mathcal{A})$	n-th shatter coefficient of the class $\mathcal{A}$ of sets.
$V_{\mathcal{A}}$	Vapnik-Chervonenkis dimension of the class $\mathcal{A}$ of sets.
$N_p(\epsilon, G, z_1^n)$	L_p ϵ-covering number of G on z_1^n.
$M_p(\epsilon, G, z_1^n)$	L_p ϵ-packing number of G on z_1^n.
$S_{x,r}$	closed Euclidean ball in $\mathbb{R}^p$ centered at $x \in \mathbb{R}^p$, with radius $r > 0$.
y_t	value of an observed time series variable at time t.
ε_t	random error at time t.
$\sigma(\cdot)$	sigmoidal activation function.
$\eta(x)$	*a posteriori probability.*
(Θ, λ)	a measurable space.
$d_H(P, Q)$	Hellinger distance between two continuous probability distributions P and Q.
$N_n(S)$	number of observations in a non-empty cell S.
$J_n(f)$	empirical risk over a suitable class of regression estimates.
H_ϵ	a family of Hellinger neighborhood.
K_γ	Kullback-Leibler neighborhood.
A^c	complement of a set A.
$pen_n(k)$	penalty term that penalizes the complexity of $F_{n,k}$.

List of Publications

T. Chakraborty and A. K. Chakraborty. Hellinger net: A hybrid imbalance learning model to improve software defect prediction. *IEEE Transactions on Reliability*, 2020a. https://doi.org/10.1109/TR.2020.3020238. 15, 77

T. Chakraborty and A. K. Chakraborty. Superensemble classifier for improving predictions in imbalanced datasets. *Communications in Statistics: Case Studies, Data Analysis and Applications*, 6(2):123–141, 2020b. 15, 72, 77, 81, 84

T. Chakraborty, S. Chattopadhyay, and A. K. Chakraborty. A novel hybridization of classification trees and artificial neural networks for selection of students in a business school. *Opsearch*, 55(2):434–446, 2018. 14, 47, 62, 101

T. Chakraborty, A. K. Chakraborty, and S. Chattopadhyay. A novel distribution-free hybrid regression model for manufacturing process efficiency improvement. *Journal of Computational and Applied Mathematics*, 362:130–142, 2019a. 15, 109, 129

T. Chakraborty, A. K. Chakraborty, and Z. Mansoor. A hybrid regression model for water quality prediction. *Opsearch*, 56(4):1167–1178, 2019b. 16, 129

T. Chakraborty, A. K. Chakraborty, and C. A. Murthy. A nonparametric ensemble binary classifier and its statistical properties. *Statistics & Probability Letters*, 149:16–23, 2019c. 14, 47, 128, 129

T. Chakraborty, G. Kamat, and A. K. Chakraborty. Bayesian neural tree models for nonparametric regression. *arXiv preprint arXiv:1909.00515*, 2019d. *Under Review*. 16, 129

T. Chakraborty, A. K. Chakraborty, M. Biswas, S. Banerjee, and S. Bhattacharya. Unemployment rate forecasting: A hybrid approach. *Computational Economics*, 2020a. https://doi.org/10.1007/s10614-020-10040-2. 16, 157

T. Chakraborty, S. Chattopadhyay, and A. K. Chakraborty. Radial basis neural tree model for improving waste recovery process in a paper industry. *Applied Stochastic Models in Business and Industry*, 36:49–61, 2020b. 15, 109, 128, 129

Chapter 1

Introduction

Summary

Prediction problems like classification, regression, and time series forecasting have always attracted both the statisticians and computer scientists worldwide to take up the challenges of data science and implementation of complicated models using modern computing facilities. But most traditional statistical and machine learning models assume the available data to be well-behaved in terms of the presence of a full set of essential features, equal size of classes, and stationary data structures in all data instances, etc. Practical data sets from the domain of business analytics, process and quality control, software reliability, and macroeconomics, to name a few, suffer from various complexities and irregularities that are often sufficient to confuse any predictive model. This can degrade the ability of the learning models to learn from the data. Motivated by this, we develop some nonparametric hybrid predictive models and study their statistical properties for theoretical robustness in this thesis. In this chapter, we provide the genesis of predictive models and the history of the hybrid and ensemble models. Subsequently, we discuss the occurrences of the different data complexities and irregularities, such as feature selection, class imbalance, regression estimation, and nonstationarity. Finally, the chapter ends with an enumeration of the contributions made herein, in an attempt to design novel solution strategies for these application-driven statistical problems.

1.1 Background

The field of *'Statistics'* is constantly challenged by the problems that science and industry bring to its door. In the early days, these problems often came from agricultural and industrial experiments and were relatively small in scope. With the advent of computers and the information age, statistical problems have exploded both in size and complexity. Challenges in the areas of data storage, organization, and searching have led to the new field of *'Data Mining'* whereas statistical and computational tools to automate this process have created the area of *'Machine Learning'*. Vast amounts of data are being generated in many fields, and the statistician's job is to make sense of it all, which includes extraction of important patterns and trends and understand "what the data says" (Hastie et al., 2009). We call this "learning from data" and this can roughly be summarized in the following steps: (a) observe a phenomenon; (b) construct a model for that phenomenon; (c) make predictions using the model.

The field of statistics and machine learning are two approaches toward the common goal of learning about a problem from data. *'Statistical Learning'* refers to a set of tools for modeling and understanding complex data sets that blends statistics with parallel developments in machine learning (Bousquet et al., 2003; Hastie et al., 2009; Vapnik and Chervonenkis, 1974). Statistical learning has emerged as a new subfield of statistics, focused on supervised and unsupervised learning and prediction (Vapnik, 2013). In supervised learning, the goal is to predict the value of an outcome measure based on some input measures whereas in unsupervised learning, there is no outcome measure, and the goal is to describe the associations and patterns among a set of input measures (James et al., 2013).

Recent technological advances have led society to capture large amounts of data in almost all fields like business, economics, quality control, software reliability, medicine, information technology, sports, etc. In many cases, the problem is either a supervised learning problem, an exploratory data analysis problem, or some combination of the above. *'Predictive modeling'* commonly refers to the process of developing statistical models using learning algorithms that approximate the relationship between a target, response, or dependent variable and various predictors or independent variables (Friedman et al., 2001). It uses multiple supervised learning techniques to predict the values of the target variables based on the given values for the explanatory variables (Siegel, 2013). The developed model is then used to predict future values of the target variable. Depending on the type of the target variable, numerical/continuous, or discrete/categorical, the problem is, respectively, called a regression or classification problem. New developments in this area can change businesses and industries by predicting future trends regularly. Over the last few decades, a significant proportion of research is devoted to the design of robust, efficient, and adaptive classification methods and regression estimation techniques.

1.1.1 A Brief History of Statistical Learning Models

Though the term statistical learning is relatively new, many of the concepts that underlie the field were developed long ago (James et al., 2013; Stigler, 1977). Already in 1632, Galileo Galilei used a procedure that can be interpreted as fitting a linear relationship to contaminated observed data (Schervish, 2012). Such fitting of a line through a cloud of points is the classical linear regression problem. A solution to this problem is provided by the famous principle of least squares method, which was discovered independently by A. M. Legendre and C. F. Gauss and published in 1805 and 1809, respectively (Stigler, 1981). One of the earliest methods developed for regression modeling is the linear regression due to F. Galton in 1889 (Galton, 1894). Linear regression is used for predicting quantitative values and was first successfully applied to the problem of astronomy. In order to predict qualitative values, such as whether a patient survives or dies, or whether the stock market increases or decreases, R. A. Fisher proposed linear discriminant analysis in 1936 (Fisher, 1938, 1940). In the 1940s, various authors put forth an alternative approach, logistic regression (Berkson, 1944; Devroye et al., 1996). In the early 1970s, Nelder and Wedderburn (1972) coined generalized linear models for an entire class of statistical learning methods that included both linear and logistic regression as special cases. In the context of time series forecasting, the early introduced parametric techniques include exponential smoothing (Holt, 1957; Winters, 1960) and autoregressive integrated moving average (ARIMA) (Box et al., 1976) models.

But most of these standard statistical techniques are parametric methods, meaning that a particular family of models, indexed by one or more parameters, is chosen for the data. The model is fitted by selecting optimal values for the parameters (or finding their posterior distribution) (James et al., 2013). Examples include linear regression (with slope and intercept parameters) and logistic regression (with the parameters being the coefficients). In these cases, it is assumed that the choice of a model family (e.g., a linear relationship with independent Gaussian error) is the correct family, and all that needs to be done is to fit the coefficients (Schervish, 2012). Recently, methodological advancement has occurred in the field of statistical learning due to the availability of massive volumes of data and the advancement of computational facilities. A preferential shift took place towards computational search-based nonparametric modeling techniques in which no prior assumptions are made about the underlying distributions of the data (Dickey, 2012). The idea behind *'Nonparametric Modeling'* is to move beyond restricting oneself to a particular family of models and utilize a much larger model space. For example, the goal of many nonparametric regression problems is to determine the continuous function that best approximates the random process without overfitting the data (Devroye et al., 1996; Györfi et al., 2002). In nonparametric setup, one is not restricted to linear functions or even differentiable functions. J. W. Tukey proposed the first nonparametric regression estimate of local averaging type in 1947, which can be regarded as a special least squares estimate (Györfi et al., 2002). Since that time, various nonparametric approaches emerged in the field of statistical learning (James et al., 2013). Among these k-nearest neighbor (Fix and Hodges Jr, 1951), classification and regression Tree (CART) (Breiman et al., 1984), artificial neural network (ANN) (Rumelhart et al., 1985), multivariate adaptive regression spline (MARS) (Friedman, 1991), Support Vector Machine (SVM) (Cortes and Vapnik, 1995), radial basis function networks (RBFN) (Krzyzak et al., 1996) and deep convolutional neural nets

(Krizhevsky et al., 2012) are most popularly used predictive models developed by statisticians (non-Bayesian) and computer scientists for a much broader community (Goodfellow et al., 2016; Murphy, 2012).

Bayesian nonparametric models are a novel class of models for Bayesian statistics and machine learning (Hjort et al., 2010). Bayesian nonparametric methods provide a Bayesian framework for model selection and adaptation using nonparametric models (Orbanz and Teh, 2010). More precisely, a Bayesian nonparametric model is a model that (1) constitutes a Bayesian model on an infinite-dimensional parameter space and (2) can be evaluated on a finite sample in a manner that uses only a finite subset of the available parameters to explain the sample (Müller and Quintana, 2004). Popular examples of Bayesian nonparametric models include Gaussian process regression, in which the correlation structure is refined with growing sample size, and Dirichlet process mixture models for clustering, which adapt the number of clusters to the complexity of the data (Orbanz and Teh, 2010). Bayesian nonparametric models have recently been applied to a variety of machine learning problems, including regression (Huang and Meng, 2020), classification (Nguyen et al., 2016), clustering (Ni et al., 2020), causal inference (Hill and Su, 2013), image segmentation (Nguyen et al., 2014), and target motion patterns (Joseph et al., 2011). Furthermore, hierarchical modeling is a fundamental concept in Bayesian statistics (Teh and Jordan, 2010). The basic idea is that parameters are endowed with distributions which may themselves introduce new parameters, and this construction recurses. In particular, nonparametric models involve large numbers of degrees of freedom, and hierarchical modeling ideas provide essential control over these degrees of freedom. Moreover, hierarchical modeling makes it possible to take the building blocks provided by simple stochastic processes such as the Dirichlet process and construct models that exhibit a richer probabilistic structure (Neal, 2000). It has a wide range of practical applications, in problems in computational biology, computer vision, and natural language processing (Ghosal and Van der Vaart, 2017).

In the context of predictive modeling, the Bayesian counterparts of some machine learning models have become very popular in modern data science, for example, Bayesian CART (Chipman et al., 1998), Bayesian additive regression trees (BART) (Chipman et al., 2010), Bayesian support vector regression (Chu et al., 2004), Bayesian neural networks (MacKay, 1992b) and Bayesian deep neural nets (Gal et al., 2017). For imbalanced classification problems where the target class distributions are not equal, some modifications to the decision tree classifier are proposed, namely the Hellinger distance decision tree (HDDT) (Cieslak et al., 2012), class confidence proportion decision tree (Liu et al., 2010), and Inter-node Hellinger distance-based decision tree (Akash et al., 2019). However, these predictive models have several limitations, which often affect the proper approximation of the relationship between the predictors and the target variables (Castillo and Melin, 2009). Firstly, real-world data sets often contain a substantial quantity of noise (e.g., errors, uninformative or highly correlated predictors, unbalanced class distributions, etc.), which can mislead the learning algorithm and produce non-optimal or wrong approximations (Kuncheva, 2004). Secondly, most statistical learning algorithms have limitations in their operations that result in the non-identification of the optimal model in the model space of the learning algorithm (Zhou, 2012). Finally, different learning algorithms vary in their interpretations of the data and noise, which may lead to varying approximations of the relationship between the target variable and its predictors.

Among various distribution-free (nonparametric) predictive models, CART and ANN are most popular in statistics and machine learning mainly for their efficiency, theoretical robustness, and ability to deal with complex data structures (Breiman et al., 1984; Hornik et al., 1989). Some key technical aspects common to these predictive modeling algorithms are the ability to generate models in the presence of noise in the data and to fabricate accurate error estimates for the generated models. These techniques provide the foundations for most modern predictive modeling methods. Also, a variety of techniques (e.g., cross-validation) are developed to handle noise and perform error estimation. Decision trees are found robust when limited data are available (Breiman et al., 1984), unlike ANN. But decision trees are high variance estimators and the variance may become large for complex problems (Loh, 2011) whereas feedforward neural networks are universal approximators (Hornik et al., 1989). Advanced neural networks are highly complex, have many free tuning parameters, and may over-fit when limited data are available (Dunson, 2018).

As a result of these limitations, the building of an optimal and efficient predictive model for a complex real-life problem is often impossible (Wozniak, 2013). The lack of universally best choice can be formalized in what is called the *'No Free Lunch theorem'* (Wolpert, 1996), which in essence says that, if there is no assumption on how the past (i.e., training data) is related to the future (i.e., test data), a prediction is often impossible. Typically, in a collection of possible models, one would look for the one that fits the data well, but at the same time is as simple as possible.

1.1.2 Developments of Hybrid Predictive Models

The diversity between statistical learning algorithms has inspired the development of hybrid and ensemble learning systems (Kuncheva, 2004; Ranawana and Palade, 2006). The relevance of hybrid and ensemble methodologies in the field of nonparametric predictive modeling is motivated by their power of being able to express knowledge contained in the data sets in multiple ways, benefiting each from the other (Kuncheva, 2002). These methods exploit the diversity of individual models and increase individual models' performance in terms of model accuracy and generalization capability. Hybrid and ensemble models introduce an intelligent combination strategy, especially while dealing with complicated classification and regression problems (Zhou, 2012). The integration of the underlying technologies into hybrid and ensemble machine learning solutions facilitates more intelligent search, enhanced optimization, reasoning and merges various domain knowledge with empirical data to solve advanced and complex problems with data irregularities. Both ensemble and hybrid methods make use of the information fusion concept but in a slightly different way.

Ensemble models combine multiple but homogeneous, weak (base) models, typically within boosting (Freund and Schapire, 1996) and bagging approaches (Breiman, 1996). Even a more general form of the ensemble is at the level of their outputs, using various fusion and combination methods (Kuncheva, 2004). This can be grouped into fixed (e.g., majority voting), and training combiners (e.g., decision templates) (Lughofer, 2012; Sannen et al., 2010). Some popular examples of nonparametric ensemble models include random forest (Breiman, 2001), gradient-boosted tree (Friedman, 2001) and Bayesian additive regression trees (Chipman et al., 2010) for pattern classification and regression estimation problems. Hellinger distance random forest (HDRF), an ensemble of HDDTs, is found useful for imbalanced pattern classification (Aler et al., 2020; Su et al., 2015). Ensemble systems have been successfully applied in many fields, for example finance (Leigh et al., 2002), bioinformatics (Tan et al., 2003), medicine (Mangiameli et al., 2004), manufacturing (Rokach, 2008; Rokach and Maimon, 2006), image retrieval (Lin et al., 2006; Tao et al., 2006) and recommender system (Schclar et al., 2009) to name a few. But, ensembles do not always improve the accuracy of the model but sometimes tend to increase the base model's error. To overcome these drawbacks, a more robust approach, namely hybridization of models, was introduced (Castillo and Melin, 2009; Wang and Lin, 2019).

Hybrid methods, in turn, combine completely different and heterogeneous statistical and machine learning models, seeking for homogeneous solutions for providing a reasonable interpretability and accuracy trade-off (Wozniak, 2013). Some popular examples of nonparametric hybrid learning models based on decision trees and neural nets include perceptron trees (Utgoff, 1989), entropy nets (Sethi, 1990), neural trees (Sirat and Nadal, 1990), soft decision tree (Frosst and Hinton, 2017), flexible neural tree (Chen et al., 2005, 2006) and recently introduced adaptive neural trees (Abpeikar et al., 2020; Tanno et al., 2019). The primary goal of these hybrid approaches was to combine decision trees with neural nets to gain the mutual benefit of both approaches. Several hybrid models are proposed by combining linear and nonlinear time series models in the context of time series forecasting. Among them, the hybrid ARIMA-ANN model (Zhang, 2003) and hybrid ARIMA-SVM model (Pai and Lin, 2005) are most popular in the literature. These hybrid approaches are applied for complex predictive modeling scenarios within the field of data-driven model-based design in which classical statistical and machine learning techniques cannot perform well.

Various extensions and implementations of the above-mentioned hybrid structures are available in the current literature of classification, regression and time series forecasting with applications in image recognition (Reinders et al., 2018; Rota Bulo and Kontschieder, 2014), medical diagnosis (Jerez-Aragonés et al., 2003; Mathan et al., 2018), fraud detection (Dong et al., 2018), knowledge acquisition (Tsujino and Nishida, 1995), river-stage predictions (Tsai et al., 2012), quality control (Sugumaran et al., 2007) and reliability modeling (Ordóñez et al., 2019) and financial time series forecasting (Adhikari and Agrawal, 2014), to name a few. Rokach (2009) offers a comprehensive review of the ensemble and hybrid literature and accommodates a wide spectrum of existing classifier ensemble and hybridization methods for pattern classification and regression estimation. Although these hybrid models are empirically shown to be useful in solving real-life problems, the theoretical results are yet to be shown for many of them.

1.1.3 Our Observation and Motivation for the Thesis

Although several taxonomies are reported in the literature, aiming to categorize hybrid systems from the system's designer point of view, there are still research gaps that need to be addressed (Rokach, 2009). On the theoretical side, the literature of hybrid predictive models is less conclusive. Regardless of their uses in practical issues of pattern classification, regression, and time series forecasting, little is known about the statistical properties of these popular hybrid models. These hybrid systems become infeasible for high-dimensional and small or medium sample-sized data sets involving both feature selection and prediction tasks. So far, the research in the field of hybrid or ensemble systems is mostly concentrated on general nonparametric regression estimation problems and relatively balanced well-structured pattern classification data sets. There is a vast scope of research to explore the beauty of hybrid models for complex situations with data irregularities. Another open problem in the hybrid literature is that the researchers only considered two or multiple frequentist methods while creating the hybridization. However, there is a scope to explore hybrid models to blend frequentist and Bayesian methods for prediction tasks. Thus, the development of novel hybrid methodologies will be required for high-dimensional, imbalanced, nonstationary, and complex real-life data sets having irregularities.

Under this scope, this thesis includes several contributions of the author dealing with real-world problems that demand new hybrid techniques to improve the robustness of the available tools. In this thesis, we are motivated to develop some novel hybrid predictive models for various supervised learning tasks. Theoretical (asymptotic properties) and practical (computational and applied) aspects of combining predictive models are studied. We consider two main goals: first is to achieve better prediction accuracy for the motivating real-life problems; and second, we study the asymptotic properties for the robustness of the methodology that makes the proposed hybrid methods theoretically well-grounded.

The primary motivation of this thesis comes from the real-world data sets, with a variety of data types, such as business, process efficiency improvement, water quality control, software defect prediction, and unemployment rate forecasting. But the emphasis is given towards the development of hybrid models that are scalable (the size of the data does not pose a problem), robust (work well in a wide variety of problems), accurate (achieve higher predictive accuracy), statistically sound (have desired asymptotic properties), and easily interpretable. Throughout the thesis, we start with the motivational applied problems followed by the development of novel hybrid predictive models. Finally, we establish asymptotic results for the proposed hybrid approaches along with relevant applications.

1.2 Problem Description

We live in the age of data, where many of the things around us are connected to various data sources, and many aspects of our daily lives are captured and stored digitally. We are surrounded by an ever-expanding sea of data fed from multitudes of sources, including social networks, video, economic data, customer transactions, stock market data, industrial process data, weather data, business data, software-based data, healthcare records, and the list goes on. We often make decisions based on the available information. As the available data or information is growing exponentially with each passing day, the opportunities to make better decisions to improve all aspects of our lives is also increasing exponentially. Given the human brain's inability to process a massive amount of data, complex data structures, and high-dimensional data, we turn to computer systems to aid in our decision-making. The process of developing these predictive models has evolved in statistics, machine learning, artificial intelligence, data mining, predictive analytics, and, more recently, data science. Although each of these fields approaches the problem from distinct perspectives using similar or different tools, they all share the same ultimate objective of making accurate predictions (Kuhn and Johnson, 2013).

Presently, though various predictive models are available in the literature, researchers are still facing the problem of choosing the best model for a particular data set (Kuncheva, 2004). Usually, there is little or no *a priori* information available about the data in hand, leaving the researchers with no other choice but a nonparametric approach. Although the predictive performance of individual models like decision trees, neural networks, and their Bayesian counterparts can sometimes be not nearly as strong on unseen data as that obtained on the training data. It is because these nonparametric methods suffer from difficulties like computational complexity, bias-variance trade-off issues, and over-fitting to the data set used for training (Kuncheva, 2004). Researchers have observed that these issues can be overcome by inducing hybrid and ensemble systems for these problems (Wozniak, 2013; Zhou, 2012). Both theoretical and empirical research affirms that hybrid and ensemble models generally perform better than individual models (Hansen and Salamon, 1990; Opitz and Maclin, 1999). Even though it is shown that diversity is an essential factor in explaining why hybrids or ensemble models perform so well, it is still an open question of how the trade-off between the accuracy of the individual models and the diversity among the models should be handled (Kuncheva and Whitaker, 2003; Tang et al., 2006). Accordingly, building a capable hybrid or ensemble system is a complex and challenging process that requires intuition about the statistical learning algorithms and in-depth knowledge about the real-life problems (Kuncheva and Whitaker, 2003). The primary concerns while developing an efficient hybrid model along with key design features are as follows:

(a) the combination of the classifiers or models to be used;

(b) the base classifier or model to be used for creating the hybridization or ensemble must be simple so that they should not overfit;

(c) to create a 'good' ensemble or hybrid model, the base learner used should be as accurate as possible;

(d) hybridization sometimes results in reduced accuracy due to difficulty in selecting the correct combination of predictive models;

(e) hybridization can help in handing data irregularities, like missing features, imbalanced data, high-dimensional low-sample-size data, etc.

(f) diversity in the methods to be used in the hybridization is considered to be a key design feature for any hybrid system.

(g) interpretability, white-box explainability, and robustness are some key desirable characteristics of the hybrid models to be developed.

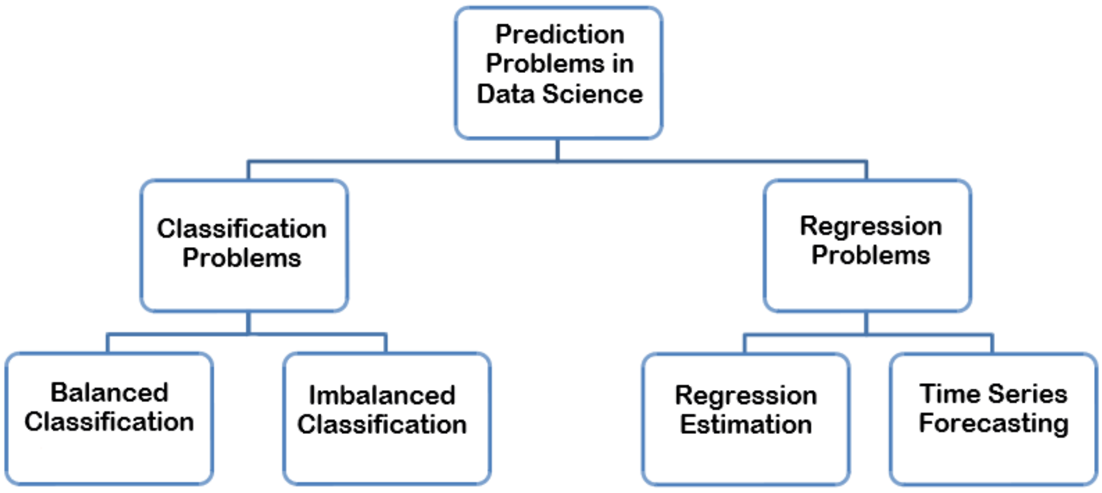


Figure 1.1: A taxonomy for different kinds of prediction problems

1.2.1 Characterization of the Problems

In this thesis, we try to develop some novel nonparametric hybrid predictive models for the data sets available from the fields of business analytics, software reliability engineering, macroeconomics, process and quality control. In general, prediction problems in data science are branched out in different types of classification and regression problems. A taxonomy of different prediction problems is presented in Figure 1.1. But real-life data pose challenges associated with supervised learning tasks under the scope of this thesis are as follows.

- **Feature selection cum classification problem.** Often, the data set comprises of several redundant information in the feature space, and the selection of essential features becomes an important job before performing the classification tasks. For example, consider a problem of the dean of a private business school in India, who would like to admit students whose placement probability is very high at the end of the Masters' program. The decision regarding the admissibility of a student has to be taken during the admission process itself. Hence, the past data that are available during the student admission process has to be the basis of the decision making process. To solve this problem, we require a model that will help the business school authorities select the important features from different academic characteristics of the students to enhance their placement probability. Finally, we would like to develop a model to predict whether a student will be placed or not based on specific characteristics (e.g., past academic records) of the student at the time of admission to the course. This business school dean's dilemma problem is addressed in Chapter 3 by developing a nonparametric hybrid predictive model based on classification tree and neural networks that can be used for both feature selection and classification tasks. Another example of a feature selection cum classification problem can be drawn from the field of medical data analysis. Consider a computerized process of myocardial perfusion diagnosis from cardiac single proton emission computed tomography (SPECT) images using a data mining and knowledge discovery approach. Kurgan et al. (2001) created a database consisting of 267 cleaned patient SPECT images (about 3000 2D images), accompanied by clinical information and physician's interpretation. The job is to develop a user-friendly model for computerizing the diagnostic process to extract a set of essential features, and then to generate explicit rules to mimic a cardiologist's diagnosis. Several other examples from the medical field are also used to show the effectiveness of the hybrid model developed in Chapter 3.

- **Imbalanced classification problem.** A common issue in many classification problems is that the classes are imbalanced. In most cases, it is the minority class that is most important to be able to predict correctly. Simultaneously, most statistical and machine learning tools perform better at predicting the majority class, making them biased towards that class. This problem occurs in software defect prediction when software engineers try to identify defects in the early phases of the software development life cycle. Imbalanced software data sets contain non-uniform class distributions, with a few instances belonging to a specific class (defective modules) as compared to that of the other class (non-defective ones). This imbalanced classification problem of the software industries is addressed in Chapter 4 by building a novel hybrid methodology, namely, the Hellinger net that outperforms state-of-the-art imbalanced classifications tools (e.g., HDDT, HDRF, etc.) to handle the 'curse of imbalanced data'. Another example of an imbalanced classification problem can be drawn from the problem of prognosis of breast cancer recurrence. The domain is characterized by 2 decision classes and 9 attributes. Data for 286 patients with known diagnostic status 5 years after the operation were available (Michalski et al., 1986). This data consists of 70% positive class examples and 30% negative class examples. Chapter 4 also attempts to deal with these types of class imbalance problems arising in different applied domains.

- **Nonparametric Regression estimation problem.** A modern paper manufacturing industry wants to improve the efficiency of the fiber-filler recovery process. The effectiveness of fiber-filler recovery equipment depends on several critical process parameters and monitoring them is a tricky proposition. The goal of improving process efficiency is to ensure an increase in the gain in percentage recovery of the fiber-filler recovery equipment, which leads the paper company to become environmentally friendly with very less ecological damage apart from being cost-effective. This problem can be viewed as a typical nonparametric regression estimation problem. One tries to establish a relationship between the response variable (recovery percentage of the equipment) with that of the significant causal variables (process parameters) without any prior assumptions of the data generating process. A novel hybrid methodology is introduced in Chapter 5 to address this problem for process efficiency improvement in the paper industry. Though we concentrate on the development of a nonparametric regression problem in Chapter 5, the model may also be useful for prediction problems arising in other domains where there is no prior knowledge available on the data generating process, for example, Auto MPG data set (Redmond and Baveja, 2002) and Wisconsin (Prognostic) breast cancer (Mangasarian et al., 1995), to name a few.

- **Combining frequentist and Bayesian methods for nonparametric regression.** Popular hybrid predictive models use different ideas to combine two or more frequentist models. Though frequentist and Bayesian methods differ in many aspects, they share some basic optimal properties. In real-life regression problems, situations exist in which a model based on one of the methods is preferable due to some subjective criterion. We try to create hybridization based on frequentist version of decision trees (neural networks) combining with the Bayesian counterparts of neural networks (decision trees) to utilize the superiority of two ideologically different paradigms in Chapter 6 of this thesis. We call this 'Bayesian neural tree' that can exploit the architecture of a tree-based method and contains a lesser number of parameters than advanced deep neural networks. The Bayesian neural tree model is further applied to solve the water quality prediction problem of boiler inlet water for the paper machine in a modern paper manufacturing company. It shows remarkable improvements compared to other conventional methods. Also, we consider data from the field of cement and concrete research (Yeh, 1998) in which concrete strength development (water-to-cement ratio) is influenced by the content of other concrete ingredients. High-performance concrete is a highly complex material, which makes modeling its behavior a very difficult task. Chapter 6 aims at developing a nonparametric regression model which can also predict the compressive strength of high-performance concrete. Several other examples have been used while comparing the performance of the model developed in Chapter 6.

- **Time series forecasting of nonstationary and nonlinear data.** This problem is motivated by the unemployment rate prediction of a country, which is a crucial factor for the country's economic and financial growth planning and a challenging job for policymakers. Traditional stochastic time series models, as well as modern nonlinear time series techniques, were employed for unemployment rate forecasting previously (Edlund and Karlsson, 1993; Katris, 2020; Vicente et al., 2015). These macroeconomic data sets are mostly nonstationary and nonlinear. Thus, it is atypical to assume that an individual time series forecasting model can generate a white noise error. Several hybrid time series models are available in the forecasting literature. We address this problem by introducing an integrated approach based on the linear ARIMA model and nonlinear autoregressive neural net model, which is an improvement over the most popular hybrid ARIMA-ANN model (Zhang, 2003) in Chapter 7. The hybrid methodology is further applied to predict the unemployment rate of various countries, namely Canada, Germany, Netherlands, New Zealand, Sweden, and Switzerland. It shows significant improvements compared to other conventional methods. Though we concentrate on the unemployment rate forecasting problem in Chapter 7, however, the developed model in this chapter can also be useful for forecasting problems arising in other domains where the data sets exhibit enough nonlinearity and nonstationarity. Some examples include exchange rate forecasting (Boothe and Glassman, 1987), electricity consumption forecasting (Cao and Wu, 2016), and forecasting of numbers of passengers in airlines (Kim and Ngo, 2001), to name a few.

1.2.2 Objectives of the Thesis

In this thesis, the main focus regarding the explicit learning strategy is on the development of specialized hybrid models for the various applied problems, drawn from the fields of business analytics, process control, quality prediction, software reliability engineering, and macroeconomic data modeling. We also employ the proposed hybrid frameworks on multiple publicly available classification and regression data sets to show the general applicability of the developed techniques in various applied domains. Furthermore, each of the chapters focuses on the implicit learning strategy, in particular, on the hybridization of tree-based methods with neural network-based models, which are introduced in Chapter 3-6 and hybridization of ARIMA with an autoregressive neural network in Chapter 7. We build some novel hybrid predictive models in a fully nonparametric setup combining both the decision tree-based models and neural network-based models in such a way that these models are capable of performing well in the prediction tasks in Chapter 3-6. In Chapter 7, a hybrid model is introduced for forecasting macroeconomic time series by combining linear and nonlinear forecasting models and its asymptotic stationarity is derived. In this thesis, the emphasis is on the distribution-free properties of the newly developed predictive models, and thus most of the consistency results presented in this thesis are valid for all the distributions of the data. The theoretically proven consistency results for the proposed models can guarantee that taking more samples essentially suffices to reconstruct the unknown distribution of the data roughly. Motivated by this, we derive the asymptotic properties of each of the developed hybrid models in the subsequent chapters of this thesis from a statistician's perspective.

Throughout the thesis, several novel hybrid predictive models are developed to solve a wide variety of applied data science problems and their statistical, computational and practical aspects are studied to address the shortcomings of the current literature. The primary goal of statistics is making inductive inferences with the developed model and emphasizes on theoretical supports for the model unlike building a 'black-box-like' model. In the thesis, all the newly introduced hybrid models have the desired statistical properties and are empirically shown to be useful in solving real-life problems from various applied fields. The main goal of this thesis is to develop novel hybrid predictive models combining two different models for studying the problem of inductive inference based on the data sets available in the field of quality, reliability, economics, and business analytics. The theoretically proven consistency results for the proposed nonparametric hybrid predictive models represent the topology in the sense of 'generalizing' the observed values of neighborhoods. Thus, this thesis will necessarily fill the gap between theory and practice that exists from the very beginning of the development of hybrid models for the last three decades. A chapter-wise enumeration of the contributions made in this thesis is presented in the following section, along with brief descriptions of each of the contributions.

1.3 Thesis layout: Chapter-wise Contributions of the Thesis

This section will look at the novel contributions made in the subsequent chapters of this thesis. Several nonparametric hybrid methodologies are developed in the following chapters to address some crucial predictive analytics problems drawn from the fields of business analytics, manufacturing process control, quality control, macroeconomic forecasting, and software defect prediction. Chapter 2 introduces the basics of statistical learning theory, some relevant statistical learning models, brief details of popular hybrid models, and describes useful performance metrics to be used in the subsequent chapters of this thesis. In Chapter 3-4, we develop some novel hybrid techniques to address the problems of the feature selection cum classification and class imbalance issues, respectively. Chapters 5-6 attempt to create hybrid methods in the context of regression estimation (frequentist) and Bayesian nonparametric regression, respectively, along with methodological developments and relevant applications. Chapter 7 presents a hybrid model for time series forecasting, which can simultaneously handle linear and nonlinear time series. Conclusions are drawn in Chapter 8 based on a critical evaluation of the contributions made in this thesis. The codes for the methods presented in this thesis can be found at `https://github.com/tanujit123`.

In Chapter 2, some statistical and machine learning techniques related to hybridization to be used in this thesis are recalled. In Section 2.2 of the chapter, underlying ideas, definitions, and some useful theorems on statistical learning theory are given. These ideas are used throughout the remaining part of the thesis. In Section 2.3, an introduction to the constituent predictive models related to hybridization is given. Introducing these models will help us in building hybrid predictive models in the later chapters. In Section 2.4, some popular hybrid methods are recalled as these methods relate closely to the context of this thesis and are widely used in predictive analytics problems. Various performance evaluation metrics to be used in the subsequent chapters for comparing the developed models with the state-of-the-art methods are briefly described in Section 2.5.

In Chapter 3, we begin by looking at a type of classification problem that requires selecting a set of essential features from the feature space prior to performing the classification task. This problem occurs in the selection of Masters' students in a private business school where the authorities of the business school want to come up with a model that can let them select the academic characteristics of students to enhance their placement probability after completion of the professional course. To solve the problem, a hybrid model based on classification trees (CT) and ANN is developed to strengthen both the feature selection and classification tasks, following Chakraborty et al. (2018, 2019c). In Section 3.2 of the chapter, we discuss the motivating example of the business school data sets in detail that motivates us to design this new hybrid approach in this chapter. In Section 3.3, we present the formulation of the proposed hybrid CT-ANN model. Several statistical properties of the model are studied in Section 3.4. In Section 3.5, our proposed hybrid model is applied to the business school data. In Section 3.6, we apply the newly developed hybrid CT-ANN model for various medical data sets to show the potential application of the methodology in other applied domains. A simulation study is also presented in Section 3.7 to make our results more convincing.

In Chapter 4, we start with another critical classification problem in which the class distributions are not equally distributed. It has been observed that there are situations where many cases belong to a larger (majority) class and fewer cases belong to a smaller (minority) yet usually more exciting class. This is called an imbalanced classification problem, where traditional classifiers tend to misclassify the minority class cases as a member of the majority class. The curse of imbalanced data sets and motivational example of imbalanced software defect prediction data are given in Section 4.2. A nonparametric hybrid model, namely Hellinger net, is developed that can solve this imbalanced classification problem arising in software reliability engineering, following Chakraborty and Chakraborty (2020a,b). Hellinger net utilizes the strength of a skew insensitive distance measure, namely Hellinger distance, in handling class imbalance problems. The detailed formulation of the Hellinger net algorithm is described in Section 4.3, followed by the asymptotic consistency of the framework, which is discussed in Section 4.4. In Section 4.5, the performance of our model is assessed over ten software defect prediction data sets and compared with the other state-of-the-art models. In Section 4.6, we apply the newly developed imbalanced classifier for standard UCI data sets to show the potential application of the methodology in other applied domains. A simulation study is also presented in Section 4.7 to make our results more convincing.

In Chapter 5, we consider another wing of the prediction problem, namely the regression estimation problem, where we try to improve the efficiency of the waste recovery process in a modern paper manufacturing company. Here, we extend the approach presented in Chapter 3 in the regression framework with conspicuous modifications in the hybrid methodology, following Chakraborty et al. (2019a, 2020b). The detailed formulation of a hybrid algorithm, namely the radial basis neural tree (RBNT) model, based on regression trees and radial basis neural net, is described in Section 5.2. The proposed model has the advantages of easy interpretability and excellent performance when applied to the process efficiency improvement problem. We study the asymptotic properties of the framework to show its theoretical robustness in Section 5.3. In Section 5.4, we describe the application of the proposed RBNT model to the waste recovery problem. A simulation study is also presented in Section 5.5 to investigate the asymptotic behavior of our proposed methodology.

In Chapter 6, we have broadened the scope of research on hybrid predictive models from a frequentist approach to building hybridization of frequentist and Bayesian methods for nonparametric regression tasks. Nonparametric regression techniques, such as regression trees and neural networks, have frequentist and Bayesian (Bayesian CART and Bayesian neural networks) counterparts to learn from data. Hence, we present a hybrid model that blends two distinct paradigms while building the hybridization. The formulation of the hybrid approach, namely Bayesian neural tree, is discussed in Section 6.2, following Chakraborty et al. (2019b,d). We study the consistency of the proposed models and derive the optimal value of a critical model parameter in Section 6.3. We discuss the application of the proposed hybrid structures to the problem of water quality prediction faced by a modern paper manufacturing company in Section 6.4. In Section 6.5, our model's performance is assessed over some standard regression data sets to show the general applicability of the newly introduced hybrid approach that blends two contrasting statistical paradigms.

In Chapter 7, we consider a different kind of regression problem involving autocorrelations in the data. This is motivated by the unemployment rate forecasting problem, a perpetual topic of research over the past three decades in economics. These macroeconomic data sets, mostly nonstationary and nonlinear, are described in Section 7.2. We present a hybrid approach based on linear and nonlinear models that can predict the unemployment rates more accurately in Section 7.3, following Chakraborty et al. (2020a). We discuss the asymptotic stationarity of the proposed hybrid approach using Markov chains and nonlinear time series analysis techniques in Section 7.4. The application of the proposed approach to various unemployment rate data sets is presented in Section 7.5. A simulation study is also presented in Section 7.6 to make our results more convincing.

Finally, in Chapter 8, relevant conclusions that result from the different chapters are summarized along with indications of some possible future directions of research.

Chapter 2

Preliminaries

Summary

This part of the thesis introduces the basic mathematical concepts needed to understand the statistical aspects of machine learning models and some relevant hybrid predictive models. We begin with general ideas from statistical learning theory, some fundamental bounds, and empirical process techniques. Next, we briefly describe some constituent models that will be useful in the development of hybrid predictive models in the later parts of this thesis. We also discuss some popular hybrid models based on decision trees and neural networks, useful for classification and regression data analysis. Finally, the performance evaluation metrics to be used for comparing different predictive models are described. In the subsequent parts of this thesis, we develop some novel hybrid predictive models within this framework.

2.1 Introduction

The main goal of statistical learning theory is to provide a framework for studying the problem of inference: gaining knowledge, constructing models from a set of data, and making predictions using the model to make appropriate decisions (Hastie et al., 2009). Indeed, the theory of statistical inference should be able to give a formal definition of words like learning, generalization, overfitting, and to characterize the performance of learning algorithms so that, ultimately, it may help design better learning algorithms (Bousquet et al., 2003). In statistical learning, the word 'learning' is considered as the process of converting experience into knowledge. The input to a statistical learning algorithm is training data, representing experience, and the output is some expertise. For a formal mathematical understanding of this concept, we recall basic ideas from statistical learning theory, predictive models, and hybrid learning systems which we use in this thesis. Statistical learning theory puts the learning process into a statistical or mathematical framework. Definitions and necessary results on empirical risk minimization, empirical processes, and useful bounds for complexity regularization are covered in Section 2.2; for a detailed treatment on these topics, one may refer to Bousquet et al. (2003); Devroye et al. (1996); Györfi (2002); Györfi et al. (2002).

In Section 2.3 of this chapter, we give a brief description of some popular statistical and machine learning models, mainly tree-based and neural network-based models, that will be further used for hybridization in subsequent chapters. We also briefly describe some popular hybrid models in Section 2.4; also refer to Kuncheva (2004); Rokach (2010); Wozniak (2013); Zhou (2012) for more details on these hybrid models. The newly developed hybrid models presented in this thesis are compared with these hybrid models already available in the literature in order to show better performance of the proposed models. The existing hybrid methods are critically reviewed from a pragmatic viewpoint, and we identify some drawbacks of the models available in the current literature, which eventually fabricates our motivation for this thesis. Finally, we recall some standard performance evaluation metrics in Section 2.5 which are used for comparison purposes.

2.2 Basics of Statistical Learning Theory

Statistical learning theory is the mathematical theory of machine learning, for which the foundations were mainly laid by Vapnik and Chervonenkis (1974). The goal of learning theory is to study in a statistical framework, the properties of learning algorithms. In particular, most results take the form of the so-called error bounds. This section introduces the techniques that are used to obtain such results in the subsequent chapters.

2.2.1 A Binary Classification Problem

In this section, we consider a supervised learning framework for pattern classification. Binary pattern classification is all about predicting the unknown class $Y \in \{0,1\}$ of an observation based on d-dimensional feature vector X after seeing the data $D_n = \{(X_1,Y_1),\cdot\cdot\cdot,(X_n,Y_n)\}$. In pattern recognition, we create a function $g(x) : \mathbb{R}^d \to \{0,1\}$ which represents one's guess of y given x. The mapping g is called a 'classifier'. To model the learning problem, let (X,Y) be an $\mathbb{R}^d \times \{0,1\}$-valued random pair which is defined by the pair (μ,η), where μ is the probability measure for X and η is the regression of Y on X. More precisely, for a Borel-measurable set $A \subseteq \mathbb{R}^d$,

$$\mu(A) = P[X \in A]$$

and for any $x \in \mathbb{R}^d$,

$$\eta(x) = P\left[Y = 1|X = x\right] = E\left[Y|X = x\right].$$

Thus, $\eta(x)$ is called the *a posteriori probability* and the distribution of (X, Y) is determined by (μ, η). An error occurs if $g(X) \neq Y$ and the probability of error for a classifier g is

$$L(g) = P\left[g(X) \neq Y\right] = E\left[1_{g(X) \neq Y}\right].$$

The Bayes classifier is given by

$$g^*(x) = \begin{cases} 1 & \text{if } \eta(x) > 1/2 \\ 0 & \text{otherwise} \end{cases}$$

that minimizes the probability of error. Also, the smallest error probability $L^* = P\left[g^*(X) \neq Y\right]$ is called the Bayes error or Bayes risk. Note that g^* depends upon the distribution of (X, Y). If this distribution is known, g^* may be computed. Mostly, the distribution of (X, Y) is unknown, so that g^* is unknown too (Györfi, 2002).

A novice might ask simple questions like this: How does one construct a good classifier? Can we estimate how good a classifier is? We call a classifier as 'good' if it is consistent (Devroye et al., 1996). A consistent rule guarantees that taking more samples essentially suffices to roughly reconstruct the unknown distribution of (X, Y). In other words, infinite amounts of information can be gleaned from finite samples. Without this guarantee, we would not be motivated to take more samples. One should be careful to not impose conditions on (X, Y) for the consistency of a rule because such conditions may not be valid. If a rule is consistent for all distributions of (X, Y), it is said to be universally consistent. Below we present formal definitions of weakly and strong universal consistency (Györfi, 2002).

Definition 1 *The classifier g is called weakly universal consistent if*

$$P\left[g(X) \neq Y\right] \to L^*$$

for all distributions of (X, Y).

Definition 2 *The classifier g is called strongly universal consistent if*

$$P\left[g(X) \neq Y \,|\, D_n\right] \to L^* \ a.s.$$

for all distributions of (X, Y).

We assume that a class $\mathbb{C}$ of classifiers $g : \mathbb{R}^d \to \{0, 1\}$ is given, and we want to choose one classifier that provides a small probability of errors. Due to having a lack of knowledge about the underlying distributions of the data, we take resort to a nonparametric set-up and we use the data to estimate the probabilities of error for the classifiers in $\mathbb{C}$. It is alluring to pick a classifier from $\mathbb{C}$ that minimizes an estimate of the probability of error over the class. The most natural way to estimate the probability of error $L(g) = P[g(X) \neq Y]$ is the error count

$$\hat{L}_n(g) = \frac{1}{n}\sum_{j=1}^{n} 1_{\{g(X_j)\neq Y_j\}},$$

where $\hat{L}_n(g)$ is called the empirical error of the classifier g. A good method should pick a classifier with a probability of error that is close to the minimal probability of error in the class. Intuitively, if we can estimate the error probability for the classifiers in $\mathbb{C}$ uniformly, then the classification function that minimizes the estimated probability of error is likely to have a probability of error that is close to the best in the class. We denote by g_n^* the classifier that minimizes the estimated probability of error over the class:

$$\hat{L}_n(g_n^*) \leq \hat{L}_n(g) \text{ for all } g \in \mathbb{C}.$$

Then for the probability of error $L(g_n^*) = P[g_n^*(X) \neq Y \mid D_n]$ of the selected rule, we have the following inequality:

Lemma 1 *(Györfi, 2002, Lemma 1.1)*

$$L(g_n^*) - \inf_{g\in\mathbb{C}} L(g) \leq 2\sup_{g\in\mathbb{C}} |\hat{L}_n(g) - L(g)|,$$

$$i.e., |\hat{L}_n(g_n^*) - L(g_n^*)| \leq \sup_{g\in\mathbb{C}} |\hat{L}_n(g) - L(g)|.$$

We see that upper bounds for $\sup_{g\in\mathbb{C}} |\hat{L}_n(g) - L(g)|$ provide us with upper bounds for two things simultaneously:

1. An upper bound for the sub-optimality of g_n^* within $\mathbb{C}$, that is, a bound for $L(g_n^*) - \inf_{g\in\mathbb{C}} L(g)$.

2. An upper bound for the error $|\hat{L}_n(g_n^*) - L(g_n^*)|$ committed when $\hat{L}_n(g_n^*)$ is used to estimate the probability of error $L(g_n^*)$ of the selected rule.

2.2.2 Regression Problem

Let Y be a real-valued random variable and X be a d-dimensional random vector. We do not assume anything about the distribution of (X, Y). In a regression problem, one wishes to estimate Y given X, i.e., one wants to find a function f defined on the range of X so that $f(X)$ is "close" to Y. Assume that the main aim of the analysis is to minimize the mean squared error:

$$\min_f E\left\{(f(X) - Y)^2\right\}.$$

The regression function $m(x)$ is defined by

$$m(x) = E[Y|X = x]$$

for $E|Y| < \infty$. For each measurable function f one has

$$\begin{aligned} E\left\{(f(X) - Y)^2\right\} &= E\left\{(m(X) - Y)^2\right\} + E\left\{(m(X) - f(X))^2\right\} \\ &= E\left\{(m(X) - Y)^2\right\} + \int_{\mathbb{R}^d} |m(X) - f(X)|^2 \mu(dx), \end{aligned} \tag{2.1}$$

where μ denotes the distribution of X. The second term of the R.H.S. of (2.1) is called L_2 error for the function f and is denoted by

$$J(f) = \int_{\mathbb{R}^d} |m(X) - f(X)|^2 \mu(dx).$$

In the regression estimation problem, we let $(X_1, Y_1), ..., (X_n, Y_n)$ to be independent and identically distributed copies of (X, Y). The regression function estimate is denoted by

$$m_n(x) = m_n\left(x, (X_1, Y_1), ..., (X_n, Y_n)\right).$$

We are interested in the $L_2(\mu)$ convergence of the regression estimate m_n to m and below we present the formal definitions of weak and strong universal consistency for regression set-up.

Definition 3 *The estimator m_n is called weakly universal consistent if*

$$E\left\{J(m_n)\right\} \to 0$$

for all distributions of (X, Y) with $E|Y|^2 < \infty$.

Definition 4 *The estimator m_n is called strongly universal consistent if*

$$\left\{J(m_n)\right\} \to 0 \ a.s.$$

for all distributions of (X, Y) with $E|Y|^2 < \infty$.

In order to get universally consistent estimates it suffices to show that $\int \left|m_n(x) - m(x)\right|^2 \mu(\mathrm{dx})$ converges to 0 for all distributions of (X, Y) with $E|Y|^2 < \infty$. This is formulated in the following lemma:

Lemma 2 *(Györfi, 2002, Lemma 10.1) Let $\mathcal{F}_n = \mathcal{F}_n(D_n)$ be a class of functions $f : \mathbb{R}^d \to \mathbb{R}$ depending on the data $D_n = \{(X_1, Y_1), ..., (X_n, Y_n)\}$. If m_n satisfies*

$$m_n(\cdot) = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{j=1}^{n} |f(X_j) - Y_j|^2, \ \textit{then}$$

$$\int \left|m_n(x) - m(x)\right|^2 \mu(dx) \leq 2 \sup_{f \in \mathcal{F}_n} \left| \frac{1}{n} \sum_{j=1}^{n} \left|Y_j - f(X_j)\right|^2 - \mathbb{E}\left|Y - f(X)\right|^2 \right| \\ + \inf_{f \in \mathcal{F}_n} \int \left|f(x) - m(x)\right|^2 \mu(dx). \quad (2.2)$$

We call the first term of the R.H.S. of (2.2) the estimation error and the second term the approximation error. The estimation error measures the distance between the L_2 risk of the estimate and the L_2 risk of the best function in $\mathcal{F}_n$. The approximation error measures how well the regression function can be approximated by the function of $\mathcal{F}_n$ in L_2. In order to get universally consistent estimates it suffices to show that both the terms converge to 0 for all distributions of (X, Y) with $E|Y|^2 < \infty$.

2.2.3 Vapnik-Chervonenkis (VC) Theory

Let $X_1, \cdots, X_n$ be i.i.d. random variables taking values in $\mathbb{R}^d$ with common distribution $\mu(A) = P[X_i \in A]$, where $A \subset \mathbb{R}^d$. Define the empirical distribution

$$\mu_n(A) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{X_i \in A\}}, \text{ where } A \subset \mathbb{R}^d.$$

Consider a class $\mathcal{A}$ of subsets of $\mathbb{R}^d$. We are concerned about the behavior of the random variable $\sup_{A \in \mathcal{A}} |\mu_n(A) - \mu(A)|$. Here we derive inequalities for the expected value, in terms of certain combinatorial inequalities related to $\mathcal{A}$. The first such quantity is the VC shatter coefficient, defined by

$$\mathcal{S}_{\mathcal{A}}(n) = \max_{x_1, \cdots, x_n \in \mathbb{R}^d} \left| \{\{x_1, \cdots, x_n\} \cap A; A \in \mathcal{A}\} \right|.$$

The $\mathcal{S}_{\mathcal{A}}(n)$ is the maximal number of different subsets of a set of n points which can be obtained by intersecting it with elements of $\mathcal{A}$. We define VC dimension as follows:

Definition 5 *(VC dimension) Let $\mathcal{A}$ be a class of subsets of $\mathbb{R}^d$ with $\mathcal{A} \neq \phi$. The VC dimension $V_{\mathcal{A}}$ of $\mathcal{A}$ is defined by*

$$V_{\mathcal{A}} = \sup \{n \in \mathcal{N} : \mathcal{S}(\mathcal{A}, n) = 2^n\},$$

i.e., the VC dimension $V_{\mathcal{A}}$ is the largest integer n such that there exists a set of n points in $\mathbb{R}^d$ which can be shattered by $\mathcal{A}$, where $\mathcal{S}(\mathcal{A}, n)$ is the n-th shatter coefficient of $\mathcal{A}$.

The theorem below is the classical version of the Vapnik-Chervonenkis inequality (Vapnik and Chervonenkis, 1974):

Theorem 1 *(Vapnik-Chervonenkis Inequality)*

$$P\left\{\sup_{A\in\mathcal{A}}|\mu_n(A)-\mu(A)|>t\right\}\leq 4\mathcal{S}_{\mathcal{A}}(2n)e^{-nt^2/8}$$

for any n and $t>0$.

The main virtue of the Vapnik-Chervonenkis inequality is that it converts the problem of uniform variations of empirical average into a combinatorial problem. The key to understand the behavior of the maximal deviations is the investigation of the behavior of $\mathcal{S}_{\mathcal{A}}(n)$. We see that if $\mathcal{A}$ is any class of sets with VC dimension V, then using VC inequality

$$E\left\{\sup_{A\in\mathcal{A}}|\mu_n(A)-\mu(A)|\right\}\leq 2\sqrt{\frac{V\log(n+1)+log2}{n}}$$

i.e., whenever $\mathcal{A}$ has a finite VC dimension, the expected largest deviation over $\mathcal{A}$ converges to zero at a rate $O\left(\sqrt{\frac{\log n}{n}}\right)$. The next result by Sauer (1972) shows the VC dimension provides a useful bound for the shatter coefficient of a class.

Theorem 2 *(Sauer's lemma) Let $\mathcal{A}$ be a class of sets with VC dimension $V<\infty$. Then for all n,*

$$\mathcal{S}_{\mathcal{A}}(n)\leq\sum_{i=0}^{V}\binom{n}{i}.$$

Now, one may immediately deduce the following using Sauer's lemma (Györfi, 2002, Corollary 1.4).

Corollary 1
1. *If $\mathcal{A}$ is the class of all linear halfspaces, viz., subsets of $\mathbb{R}^d$ of the form $\{x : a^T x\geq b\}$, where $a\in\mathbb{R}^d, b\in\mathbb{R}$ takes all possible values then $V\leq d+1$.*
2. *If $\mathcal{A}$ is the class of all closed balls in $\mathbb{R}^d$, viz., sets of the form*

$$\left\{x=\left(x^{(1)},\cdots,x^{(d)}\right):\sum_{i=1}^{d}\left|x^{(i)}-a_i\right|^2\leq b\right\},$$

where $a_1,\cdots,a_d,b\in\mathbb{R}$, then $V\leq d+2$.

Note that the above-mentioned result implies that the VC dimension of the class of all linear halfspaces actually equals $d+1$.

Remark 1 *An interpretation of VC dimension and shatter coefficient is that they measure the effective size of the class, that is the size of the projection of the class onto finite samples. In addition, this measure does not just 'count' the number of functions in the class but does not depend on the geometry of the class.*

Finally, the finiteness of the VC dimension ensures that the empirical risk will converge uniformly over the class to the true risk.

To complete our argument, we need to relate the VC dimension of a class of sets $\mathcal{A}$ to the covering numbers $N(r, \mathcal{A}(x_1^n))$ which is a monotonically decreasing function of r, defined as follows:

Definition 6 *(Covering numbers) The covering number of $\mathcal{F}$ at radius ϵ, with respect to D_n, denoted by $N(\mathcal{F}, \epsilon, n)$ is the minimum size of a cover of radius ϵ.*

The relationship between VC dimension with that of covering numbers are given in the following theorem:

Theorem 3 *Let $\mathcal{A}$ be a class of sets with VC dimension $V < \infty$. For every $x_1, \cdots, x_n \in \mathbb{R}^d$ and $0 \leq r \leq 1$,*

$$N(r, \mathcal{A}(x_1^n)) \leq \left(\frac{4e}{r^2}\right)^{\frac{V}{1-1/e}}.$$

Proof For proof, see Theorem 1.17 of Györfi (2002). □

Let $Z_1^n = (Z_1, \cdots, Z_n)$ is a sequence of i.i.d. random variables. Then $N_1(\epsilon, \mathcal{G}, Z_1^n)$ is also a random variable. Using the bounds above, we may derive some interesting inequalities (Haussler, 1992; Pollard, 1984, 1990) which will be useful in the later portions of the thesis .

Theorem 4 *(Pollard's Inequality (1984)) Let $\mathcal{G}$ be a set of functions $g : \mathbb{R}^d \to [0, B]$. For any n, constant B (> 0) and $\epsilon > 0$,*

$$P\left\{\sup_{g \in \mathcal{G}} \left|\frac{1}{n}\sum_{i=1}^{n} g(Z_i) - E\{g(Z_1)\}\right| > \epsilon\right\} \leq 8E\left\{N_1\left(\frac{\epsilon}{8}, \mathcal{G}, Z_1^n\right)\right\}.e^{-\frac{n\epsilon^2}{(128B^2)}}.$$

Proof For proof, see Theorem 9.1 of Györfi et al. (2002). □

Theorem 5 *(Haussler's Inequality (1992)) Let $\mathcal{G}$ be a set of functions $g : \mathbb{R}^d \to [0, B]$ with $V_{\mathcal{G}^+} \geq 2$, let $p \geq 1$, also let $Z_1^n \in \mathbb{R}^{d.n}$ and let $0 < \epsilon < \frac{B}{4}$. Then*

$$N_p(\epsilon, \mathcal{G}, Z_1^n) \leq 3\left(\frac{2eB^p}{\epsilon^p}\log\frac{3eB^p}{\epsilon^p}\right)^{V_{\mathcal{G}+}}.$$

Proof For proof, see Theorem 9.4 of Györfi et al. (2002). □

Theorem 6 *(Pollard's Theorem (1990)) Let $\mathcal{F}$ and $\mathcal{G}$ be classes of real functions on $\mathbb{R}^d$, bounded by M_1 and M_2, respectively (i.e., for e.g., $|f(x)| \leq M_1$ for every $x \in \mathbb{R}^d$ and $f \in \mathcal{F}$). For arbitrary fixed points $z_1^n = (z_1, \cdots, z_n)$ in $\mathbb{R}^d$ define the sets $\mathcal{F}(z_1^n)$ and $\mathcal{G}(z_1^n)$ as follows:*

$$\mathcal{F}(z_1^n) = \{(f(z_1), \cdots, f(z_n)); f \in \mathcal{F}\} \text{ and } \mathcal{G}(z_1^n) = \{(g(z_1), \cdots, g(z_n)); g \in \mathcal{G}\}.$$

Introduce $\mathcal{T}(z_1^n) = \{(h(z_1), \cdots, h(z_n)); h \in \mathcal{T}\}$ for the class of functions

$$\mathcal{T} = \{fg; f \in \mathcal{F}, g \in \mathcal{G}\}.$$

Then for every $\epsilon > 0$ and z_1^n

$$N(\epsilon, \mathcal{T}(z_1^n)) \leq N\left(\frac{\epsilon}{2M_2}, \mathcal{F}(z_1^n)\right).N\left(\frac{\epsilon}{2M_1}, \mathcal{G}(z_1^n)\right).$$

Proof For proof, see Theorem 29.7 of Devroye et al. (1996). □

2.2.4 Basic Bounds and Concentration Inequalities

In this section, we show how to obtain simple error bounds (also called generalization bounds) from empirical processes and some other interesting inequalities within this framework.

Definition 7 *Let (X, d) be a metric space and let $\epsilon > 0$. A set $A \subset X$ is an ϵ-net of X if for all $x \in X$ there exists an $y \in A$ such that $d(x, y) \leq \epsilon$.*

If X has a finite ϵ-net, then one may define the ϵ-covering number $N(X, \epsilon)$ of X as the cardinality ϵ-net.

Definition 8 *A set $A \subset X$ is an ϵ-packing if for all $x \neq y \in A$, $d(x, y) \geq \epsilon$. The ϵ-packing number $M(X, \epsilon)$ is the number of points in the ϵ-packing with largest cardinality.*

The next lemma follows immediately from the above definitions.

Lemma 3 *For all X and $\epsilon > 0$,*

$$M(X, \epsilon/2) \leq N(X, \epsilon) \leq M(X, \epsilon).$$

Hoeffding's Inequality. Let us rewrite the quantity we are interested in as follows

$$L(g) - L_n(g) = E\left[f(Z)\right] - \frac{1}{n}\sum_{i=1}^{n} f(Z_i).$$

It is easy to recognize here the difference between the expectation and the empirical average of the random variable $f(Z)$. By the laws of large numbers, we immediately obtain that

$$P\left[\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} f(Z_i) - E\left[f(Z)\right] = 0\right] = 1.$$

This indicates that with enough samples, the empirical risk of a function is a good approximation to its true risk. It turns out that there exists a quantitative version of the law of large numbers when the variables are bounded (Györfi, 2002, Theorem 1.2).

Theorem 7 *(Hoeffding's Inequality) Let $Z_1, \cdots, Z_n$ be n i.i.d. random variables with $f(Z) \in [a, b]$. Then for all $\epsilon > 0$, we have*

$$P\left[\left|\frac{1}{n}\sum_{i=1}^{n} f(Z_i) - E\left[f(Z)\right]\right| > \epsilon\right] \leq 2\exp\left[-\frac{2n\epsilon^2}{(b-a)^2}\right].$$

An important feature of the result above is that it is completely distribution-free. The actual distribution of the data does not play a role at all in the upper bound, which will be very useful to derive the consistency results of our proposed hybrid frameworks in later parts of this thesis. Below we present an important result from measure theory about sequences of events.

Lemma 4 *(Borel-Cantelli Lemma) Suppose that $\{A_n : n \geq 1\}$ is a sequence of events in a probability space. Then the event $A(i.o.) = \{A_n \text{ occurs for infinitely many } n\}$ is given by*

$$A(i.o.) = \bigcap_{k=1}^{\infty}\bigcup_{n=k}^{\infty} A_n.$$

If

$$\sum_{n=1}^{\infty} P(A_n) < \infty,$$

then $P(A(i.o.)) = 0$; only a finite number of the events occur, w.p. 1.

No Free Lunch Theorem. We now cite an important result that tells for a fixed sample size, one can construct arbitrarily bad problems for a given algorithm (Wolpert, 2002). The theorems below state that the 'bad' probability measure is constructed on a countable set (where the outputs are not related at all to the inputs so that no generalization is possible), and is such that the rate at which one gets to see new inputs is as slow as the convergence of a_n.

Theorem 8 *(No Free Lunch) For any algorithm, any n and any $\epsilon > 0$, there exists a distribution μ such that $L^* = 0$ and*

$$P\left[L(g_n) \geq \frac{1}{2} - \epsilon\right] = 1.$$

Theorem 9 *(No Free Lunch at all) For any algorithm and any sequence $\{a_n\}$ that converges to 0, there exists a distribution μ such that $L^* = 0$ and*

$$L(g_n) \geq a_n.$$

2.3 Overview of Constituent Models

The primary goal in statistics is making the scientific inferences and emphasizes on theoretical supports (like asymptotic properties) of the classifier unless building a "black-box-like" model for the task at hand (Dunson, 2018). Decision trees and neural nets are two competitive models having a strong statistical background. Below we briefly describe these constituent models to be used in the development of hybrid methods in Chapters 3-6. For Chapter 7, we discuss the component time series models (ARIMA and ARNN) to be used in the hybridization within the chapter itself.

2.3.1 Classification and Regression Tree (CART)

CART is a greedy divide-and-conquer algorithm that finds axis-parallel partitions via recursive partitioning of the feature space into homogeneous regions (Breiman et al., 1984). The binary decision tree construction starts with assigning the total training data points in one group, named as the parent node. Parental nodes are split into two child nodes using one of the feature vectors. The selection of attributes is made based on any entropy-based impurity function for classification problems. Classification tree (CT) usually uses Gini impurity and information gain to measure the splits' quality, and the best split amongst all the splits at a given branch is selected. The splitting process ends when a full tree is grown. Different tree complexity measures are usually employed to prune the branches with very few data points to avoid data over-fitting. Finally, we can assign the class labels for each terminal or leaf nodes. Regression trees or RT, for short, are designed for dependent variables that take a finite number of unordered values, with prediction error typically measured by the squared difference between the observed and predicted values (Quinlan, 1990). Given a test data point, a sequence of tests along the decision nodes starting from the root node will determine the path along the tree until it reaches a terminal node. At the terminal node, a prediction is made according to the local model associated with that node. To construct an RT using the training set, we start at the root node. We select the variable (and its split threshold) whose splitting will lead to the largest reduction in mean squared error (MSE). We continue these splits recursively until the MSE reaches an acceptable threshold. A typical practice is to perform some kind of pruning for the tree, once designed. This will eliminate ineffective nodes and to keep in check model complexity. CART models are popular for several reasons (Hastie et al., 2009): they are easy to interpret, they can easily handle mixed discrete and continuous inputs, they are insensitive to monotone transformations of the inputs (because the split points are based on ranking the data points), they perform automatic variable selection, they are relatively robust to the outlier, they scale well to large data sets, and they can be modified to handle missing inputs. However, CART models also have some disadvantages. The primary one is that they do not predict very accurately compared to other kinds of models. This is in part due to the greedy nature of the tree construction algorithm. The related problem is that trees are unstable: small changes to the input data can have significant effects on the structure of the tree. This is due to the hierarchical nature of the tree-growing process, causing errors at the top to affect the rest of the tree. In frequentists terminology, we say that trees are high variance estimators (Loh, 2011).

2.3.2 Bayesian CART Model

A CART model consecutively divides the predictor space into multiple regions. The partitioning begins at the root node, followed by splits at each internal node. A splitting rule (i.e., a chosen predictor and a split threshold) for a node is determined based on the minimization of the mean squared error (MSE) in regression settings. For each node, a stopping criterion called ‘minsplit’ is defined in terms of the minimum number of observations required in the node for further splitting. A node with less than ‘minsplit’ samples are labeled as a terminal node. At a terminal node, the predictor space is not split any further. Every data point falls into a region defined at one of the terminal nodes, and predictions are made using the local parameter. A fully grown tree is often pruned back via cross-validation or cost-complexity pruning to avoid overfitting. In contrast, in the Bayesian version of CART, we assume that a tree T has b terminal nodes. Let the set of terminal node parameters be $\Lambda = \{\lambda_1, \ldots, \lambda_b\}$. A prior is then placed on (Λ, T) as

$$P(\Lambda, T) = P(\Lambda|T)\ P(T), \tag{2.3}$$

where $P(T)$ is specified as a tree generating stochastic process comprising two functions, namely $P_{split}(m, T)$, the probability that a terminal node m in a tree T is split, and $P_{rule}(\gamma|m, T)$, the probability that a splitting rule γ is assigned if m is split (Chipman et al., 1998). A general form of $P_{split}(m, T)$ is (Chipman et al., 1998)

$$P_{split}(m, T) = \alpha(1 + D_m)^{-\beta}, \tag{2.4}$$

where D_m denotes the number of splits before the m^{th} node, and $0 < \alpha < 1$ and $\beta \geq 0$. Larger values of β make the splitting of deeper nodes less probable, since the RHS in (2.4) is a decreasing function of the depth D_m of a node. The prior $P_{rule}(\gamma|m, T)$ is specified so that at an internal node, each available predictor is equally likely to be chosen for a split, and for a chosen predictor, each of its observed values is equally likely to be chosen as a splitting threshold. $P(\Lambda|T)$ is generally specified so that the marginalization

$$P(Y|T, X) = \int P(Y|X, \Lambda, T)\ P(\Lambda|T)\ d\Lambda \tag{2.5}$$

is feasible (Chipman et al., 1998). For a continuous Y, we model the values in the m^{th} terminal node as a Gaussian with mean μ_m and variance σ_m^2, where $1 \leq m \leq b$. Thus, we have $\Lambda = \left\{\mu_m, \sigma_m^2\right\}_{m=1}^{b}$, with μ_m and σ_m^2 having conjugate Gaussian and Inverse-Gamma priors respectively, as in Chipman et al. (1998, 2002). The posterior over the possible tree models $P(T|Y, X)$ is analytically explored via a Metropolis-Hastings search algorithm. A 'good' tree is usually found as a trade-off between the number of terminal nodes b, and a high value of the marginal probability $P(Y|T, X)$.

2.3.3 Bayesian Additive Regression Trees (BART)

Bayesian additive regression trees (BART) is a fully Bayesian approach to modeling with ensembles of trees in which the overall conditional mean of a response given predictors is expressed as the sum of many trees (Chipman et al., 2010). The BART algorithm includes an effective Markov Chain Monte Carlo algorithm which explores the complex space of an ensemble of trees without pre-specifying the dimension of each tree (Pratola et al., 2014). Guided by the prior, the complexity of the model is inferred. The overall model is made up of many small contributions from ensemble models. Typically, as in BART, the models in the ensemble are binary trees (Hill et al., 2020). BART can uncover complex regression functions with high-dimensional regressors in a fairly automatic way and provide Bayesian quantification of the uncertainty through the posterior (Chipman et al., 2010). Let y denote the response and x denote the vector of predictor variables. BART consider the basic model

$$Y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2), \tag{2.6}$$

The goal is to be able to infer about the function f with minimal assumptions and high dimensional x. BART lets

$$f(x) = \sum_{j=1}^{m} g(x; T_j, M_j) \tag{2.7}$$

where each $g(x; T_j, M_j)$ represents the function captured by a single binary tree. Each binary tree is described by the tree T which encode the structure of the tree and all of the decision rules and $M = (\mu_1, \mu_2, \dots, \mu_b)$ which records the values associated with each bottom or leaf node of the tree which has b bottom nodes. Thus, the construction of the BART model comprises of two main steps: a sum-of-trees model and a regularization prior on the parameters of that model. Chipman et al. (2010) provides a very detailed description of BART prior, BART MCMC and specification of the prior on σ.

2.3.3.1 BART Prior

BART entails both a prior on the parameter $\Theta = ((T_1, M_1), \dots, (T_m, M_m), \sigma)$ and a MCMC algorithm for exploring the posterior. Note that the dimension of each T_j is not fixed. The prior has the form

$$p(\Theta) = p(\sigma) \prod_{j=1}^{m} p(T_j, M_j)$$

with $p(T, M) = p(T)p(M|T)$. Note that the dimension of M depends on T.

$$p(M|T) = \prod_{i=1}^{m} p(\mu_i)$$

with $p(\mu) \sim \mathcal{N}(0, \tau^2)$ so that conditional on all the trees all the μ's at the bottom of all three are i.i.d. $\mathcal{N}(0, \tau^2)$. The prior for σ is the standard inverted chi-squared: $\sigma^2 \sim (\nu\lambda)/(\chi^2_\nu)$. Chipman et al. (2010) describe a tree growing process to specify the prior $p(T)$ and data based default prior choices for τ and (ν, λ). The key to these prior choices is that the prior expresses of preference for small trees and shrinks all the μ towards zero in such a way that only the overall sum can capture f. These prior specifications enable BART to make each individual tree a "weak learner" in that it only makes a small contribution to the overall fit.

2.3.3.2 BART MCMC

Given the observed data y, the BART model induces a posterior distribution $p((T_1, M_1), \dots, (T_m, M_m), \sigma|y)$ on all the unknowns that determine a sum-of-trees model (2.6 and 2.7). Although the sheer size of the parameter space precludes exhaustive calculation, the following backfitting MCMC algorithm can be used to sample from this posterior. At a general level, the algorithm is a Gibbs sampler (Casella and George, 1992). For notational convenience, let $T_{(j)}$ be the the set of all trees in the sum *except* T_j, and similarly define $M_{(j)}$. Thus $T_{(j)}$ will be a set of $m - 1$ trees, and $M_{(j)}$ the associated terminal node parameters. The Gibbs sampler here entails m successive draws of (T_j, M_j) conditionally on $(T_{(j)}, M_{(j)}, \sigma)$:

$$(T_j, M_j)|T_{(j)}, M_{(j)}, \sigma, y,$$

$j = 1, \ldots, m$, followed followed by a draw of σ from the full conditional:

$$\sigma|T_1, \ldots, T_m, M_1, \ldots, M_m, y.$$

Each (T, M) draw is done by letting $p(T, M|o) = p(T|o)p(M|T, o)$ where o denotes all the other conditioning information. Given the prior choices one can analytically integrate out the μ to obtain an computationally convenient expression for $p(T|o)$ Metropolis Hastings steps are then use to propose changes to T (Chipman et al., 2010). While many useful steps are in the literature the key steps are the birth/death pair. A birth step proposes adding a decision rule to a bottom node of the current tree so that it spawns left and right child bottom nodes. A death move proposes the elimination of a left/right pair of bottom nodes. This key birth/death pair of moves allows the MCMC to explore trees of varying complexity and size, as in Chipman et al. (2010).

However, BART assumes independent and identical distributed (i.i.d) normal errors. This strong parametric assumption can lead to misleading inference and uncertainty quantification (George et al., 2019). A recent modification to the BART model using the classic Dirichlet process mixture (DPM) mechanism can adapt to non-normal errors (George et al., 2019). Also, Chakraborty (2016) provides a flexible Bayesian regression tree model when the response variable is a vector and the components of the vector are highly correlated.

2.3.4 Hellinger Distance Decision Tree (HDDT)

One way of handling an imbalanced data set is to take recourse to sampling techniques during the preparation of the data set for further analysis. However, a significant disadvantage of these techniques is that in the process of sampling, we lose a lot of information in the form of losing the real-life data. Cieslak and Chawla (2008) proposed HDDT, which uses HD as the splitting criterion to build a decision tree. HD is used as a measure of distributional divergence and has the property of skew insensitivity (Rao, 1995). Let (Θ, λ) denote a measurable space. For any binary classification problem, let us suppose that P and Q be two continuous distributions concerning the parameter λ having the densities p and q in a continuous space Ω, respectively. Define HD as follows:

$$d_H(P, Q) = \sqrt{\int_\Omega (\sqrt{p} - \sqrt{q})^2 d\lambda} = \sqrt{2\left(1 - \int_\Omega \sqrt{pq} d\lambda\right)}$$

where $\int_\Omega \sqrt{pq}d\lambda$ is the Hellinger integral. It is noted that HD doesn't depend on the choice of the parameter λ. Given a countable space Φ, HD can also be written as follows:

$$d_H(P,Q) = \sqrt{\sum_{\phi \in \Phi} \left(\sqrt{P(\phi)} - \sqrt{Q(\phi)} \right)^2}$$

The bigger the value of HD, the better is the discrimination between the features. A feature is selected that carries minimal affinity between the classes. For the application of HD as a decision tree criterion, the final formulation can be given as follows:

$$d_H(X_+, X_-) = \sqrt{\sum_{j=1}^{K} \left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}} \right)^2} \tag{2.8}$$

where $|X_+|$ indicates the number of examples that belong to the majority class in the training set and $|X_{+j}|$ is the subset of the training set with the majority class and the value j for the feature X. A similar explanation can be written for $|X_-|$ and $|X_{-j}|$ but for the minority class. Here K is the number of partitions of the feature space X. Since equation (2.8) is not influenced by prior probability, it is insensitive to the class distribution. Based on the experimental results, Cieslak and Chawla (2008) concluded that unpruned HDDT is recommended for dealing with imbalanced problems as a better alternative to sampling approaches.

2.3.5 Artificial Neural Networks (ANN)

Neural network models are inspired by biological nervous systems (Kuncheva, 2004). The connections between elements mainly determine the network functions. The training of a neural network can be done by performing a particular function by adjusting the values of the connections (weights) between elements. Neural networks are trained so that a particular input (feature vectors) leads to a specific target output (class level). The network is adjusted, based on a comparison of the output and the target, until the network output matches the observed class. Mapping functions used in ANN are very flexible. Given the right weights, this function can approximate almost any functional form to any degree of accuracy. This function approximation is mainly made by an activation function (for example, logsig, tansig, etc.). The neural networks most commonly used in engineering applications are the multilayer perceptron networks, also known as "backpropagation" or "feedforward" networks. Back-propagation is a gradient descent algorithm that compares actual outputs with desired outputs (Rumelhart et al., 1985). If an error exists, its reduction is accomplished by back-propagating the error through the network and adjusting the weights. A network can have one or more hidden layers. While training the network with any particular data set, the problem of overfitting can be avoided by training the network for a limited number of epochs (Goodfellow et al., 2016).

A feedforward neural network model with p input nodes, one hidden layer with M hidden nodes, one output node and activation functions Ψ is a model relating p explanatory variables $x = (x_1, x_2, \ldots, x_p)$ and a response variable y of the form

$$\hat{y}(x) = \sum_{j=1}^{M} \beta_j \Psi(x^{'} \gamma_j + \delta_j) \tag{2.9}$$

with $\beta_j \in \mathbb{R}, \gamma_j \in \mathbb{R}^p$. The terms δ_j are designated biases and may be assimilated to the rest of the γ_j vector if we consider an additional input with constant value one, say $x_0 = 1$. The typical setup for FFNNs is: Given data $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and fixed M, choose $\beta = (\beta_1, \beta_2, \ldots, \beta_M), \gamma = (\gamma_1, \gamma_2, \ldots, \gamma_M)$ according to a least squares criterion

$$\min_{\beta,\gamma} \sum_{i=1}^{N} (y_i - \hat{y}(x_i))^2 ,$$

via backpropagation (Rumelhart et al., 1985), an implementation of steepest descent algorithm. Hence, at least implicitly, we are assuming a normal error model and we are viewing a nonlinear parametric regression problem. It is also sometimes suggested to include a regularisation term in the objective function to avoid data overfitting (Goodfellow et al., 2016).

Deep neural networks contain multiple nonlinear hidden layers, and this makes it very useful for complex, large-scale image data sets (Nowlan and Hinton, 1992). But, deep neural net models are highly complex and over-parameterized models that lack substantial flexibility and sometimes lead to overfitting (Dunson, 2018). It requires extensive data sets (like image, audio, video data sets), which poses a significant problem in many real-life situations. Statisticians are usually not satisfied with a 'black-box-like' model for prediction; instead, they prefer classifiers with asymptotic behavior like consistency (Dunson, 2018). Because of the universal consistency of one-hidden-layer feedforward neural networks (Hornik et al., 1989), there is a little theoretical gain in considering neural networks with more than one hidden layer. However, there may be an information-theoretic gain as the number of hidden neurons needed to achieve the same performance may be substantially reduced (Devroye et al., 1996).

2.3.6 Radial basis function Networks (RBFN)

An ANN is a nonparametric model consisting of an input layer, a certain number of hidden layers, and an output layer. All inputs to the network pass through the hidden layers, after which they are mapped to the final output. Each interconnection of neurons in an ANN is associated with a weight and such weights are obtained by minimizing an error function and its gradient. RBFN, a family of ANNs, consists of only a single hidden layer and uses a nonlinear function called a radial basis function as an activation function, unlike MLP. RBFN is a three-layered feed-forward structure where the input layer distributes inputs to the hidden layer, which contains neurons with a nonlinear activation function (Györfi et al., 2002). Since Gaussian functions are most frequently used in this layer, we use the Gaussian kernel in this thesis:

$$\phi_i(x_i) = \phi\Big(\parallel x_i - c_i \parallel ; \sigma_i \Big) = exp\Big(- \frac{\parallel x_i - c_i \parallel^2}{2\sigma_i^2} \Big)$$

where x is an input vector, ϕ_i is the output of i^{th} hidden neuron in the hidden layer with centers c_i and σ_i as the standard deviations of the RBFN model. Finally, the output layer can be written as a weighted sum of hidden layer outputs:

$$f(x_i) = \sum_{j=1}^{k} \gamma_j \, \phi\Big(\parallel x_i - c_j \parallel \Big) + \gamma_0,$$

where γ_j is the weight of the link from j^{th} hidden neuron to the i^{th} output neuron. An exciting property of RBFN which distinguishes it from other types of ANNs is that the center vector can be selected as cluster centers of the input data.

2.3.7 Bayesian Neural Networks (BNN)

In BNN, we consider an ANN with parameter vector θ and σ denotes the variance function, which contains the network weights and a general offset (or bias) parameter. In the Bayesian approach to neural network learning, the objective is to find the predictive distributions for the target values in a new test case given the inputs for that case as well as inputs and target for the training cases. In the Bayesian setting, a zero-mean multivariate Gaussian prior is placed on θ (MacKay, 1992b; Neal, 1996) as

$$P(\theta) = \frac{1}{\left(\frac{2\pi}{\sigma_p}\right)^{\frac{l}{2}}} .exp\left(-\frac{\sigma_p}{2} ||\theta||^2 \right), \tag{2.10}$$

where l is the length of θ. The likelihood function is modeled as a Gaussian function which is given by

$$P(L_n|\theta) = \frac{1}{\left(\frac{2\pi}{\sigma_l}\right)^{\frac{n}{2}}} .exp\left(-\frac{\sigma_l}{2} \sum_{i=1}^{n} \left(\hat{Y}_i - Y_i \right)^2 \right). \tag{2.11}$$

Predictions are obtained from the posterior predictive distribution

$$P(Y|X, L_n) = \int_\theta P(Y|X, \theta)\ P(\theta|L_n)\ d\theta. \tag{2.12}$$

The integral in (2.12) is approximated by $P(Y|X, \tilde{\theta})$, where $\tilde{\theta}$ is obtained by locally minimizing

$$E = \frac{\sigma_l}{2} \sum_{i=1}^{n} \left(\hat{Y}_i - Y_i\right)^2 + \frac{\sigma_p}{2} ||\theta||^2. \tag{2.13}$$

The first term in the R.H.S. of (2.13) corresponds to the error function minimized in frequentist settings. The second term corresponds to a regularization term that penalizes larger values in θ and, hence, restrains overfitting. A BNN can also have a variable architecture, i.e., the number of hidden nodes can be subject to a Geometric distribution, which enables one to place a lower probability on more extensive networks, see Rios Insua and Müller (1998). Neal (1996) applied hybrid Markov chain Monte Carlo (MCMC) numerical integration techniques for the implementation of Bayes procedures. Also, Holmes and Mallick (1998) have used Bayesian neural network modeling in the regression context and Ghosh et al. (2004) considers a hierarchical Bayesian neural network approach for posterior prediction probabilities of certain features indicative of non-organ-confined prostate cancer (Chakraborty et al., 2005).

2.3.7.1 Variable Architecture in BNN and BNN Priors

An important issue that has received comparatively little attention in the literature is the choice of architecture which, for the model we are considering, following the notations of Section 2.3.5, consists mainly of choosing the number M of hidden nodes. Typically, a choice is made by trial and error, though there are several heuristics helping in that task (Rios Insua and Müller, 1998). We rewrite Eqn. 2.9 after including a linear regression term $x^{'}\lambda$ (λ denotes linear weights) as follows.

$$y_i = x_i^{'}\lambda + \sum_{j=1}^{M} \beta_j \Psi(x_i^{'}\gamma_j + \delta_j) + \epsilon_i, \quad i = 1, 2, \ldots, N, \tag{2.14}$$

where $\epsilon_i \sim N(0, \sigma^2)$. A choice of Ψ is a logistic activation function. The parameters in variable architecture-based BNN model are the linear weights $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_p)$ and $(\beta_0, \beta_1, \ldots)$, the logistic parameters $(\gamma_0, \gamma_1, \ldots)$, the number M of terms and the error variance σ^2. The prior over network parameters is:

$$\beta_j \sim N(\mu_\beta, \sigma_\beta^2), \lambda_j \sim N(\mu_\beta, \sigma_\beta^2), \gamma_j \sim N(\mu_\gamma, S_\gamma), \sigma^{-2} \sim Gamma(s/2, sS/2).$$

We provide a prior over the number M of hidden nodes. We choose a geometric prior with parameter α based on the recommendation of Rios Insua and Müller (1998):

$$M \sim Geom(\alpha).$$

Assuming geometric prior for the number M of hidden nodes rewards by putting geometrically decreasing prior probability on larger networks and have also been recommended in Lee (2004).

2.3.7.2 MCMC Posterior Simulation for BNN Model with Variable Architecture

First, the variable architecture model defines a model with changing dimensionality parameter vector: posterior inference has to mix over models with hidden layers of different size. Rios Insua and Müller (1998) introduced an algorithm which addresses the changing dimensions by using an implementation of reversible jump (Green, 1995) with moves corresponding to "birth", "death", "thinning" and "seeding" of hidden nodes since straightforward implementation of commonly used MCMC schemes is hindered by a variety of issues. Second, the high dimensional parameter vector is typically highly correlated *a posteriori*, necessitating MCMC strategies which use blocking to jointly update as many parameters as possible and marginalizing to partly avoid the random walk nature of Metropolis algorithms. Rios Insua and Müller (1998) introduced a scheme which marginalizes over the weights β_j when updating the input weights γ_j. Posterior simulation in this BNN model is done by stating the algorithm for one sweep, i.e., the transition from imputed parameter values at iteration t to those values at iteration $t+1$. The list below outlines the steps in one sweep. Details of each step are discussed in Rios Insua and Müller (1998). An item of the form $[a|b]$ indicates that parameter a is being updated using current values for parameters b. The absence of some parameter c in the conditioning set (to the right of the bar) indicates that either c is being marginalized over, or that a and c are conditionally independent given b. Let $\gamma_j = (\gamma_{j1}, \ldots, \gamma_{jp})', \gamma = (\gamma_1, \ldots, \gamma_M)', \beta = (\beta_1, \ldots, \beta_M)', \lambda = (\lambda_0, \ldots, \lambda_p)$ and $\nu = (\mu_\beta, \sigma_\beta, \mu_\gamma, \sigma_\gamma, \sigma)$. Let γ_{-jk} indicate the list of all weights $\gamma_{j'k'}$ without γ_{jk}. The following steps define one sweep of the Markov chain:

(i) $[\gamma_{jk}|\gamma_{-jk}, M, \nu, D],\ j = 1, \ldots, M,\ k = 0, \ldots, p,$

(ii) $[M|\gamma, \nu, D]$, "Birth/Death"

(iii) $[M|\gamma, \nu, D]$, "Seed/Thin"

(iv) $[\beta, \lambda|\gamma, M, \nu, D]$,

(v) $[\nu|\beta, \gamma, \lambda, D]$.

Convergence of the algorithm to the desired stationary distribution is also discussed in Rios Insua and Müller (1998).

2.4 Overview of Hybrid Predictive Models

Hybridization of two (or more) models offers many alternatives for handling complex data structures involving uncertainty, ambiguity, and high dimensionality of the data. Hybridization appears in many domains, including human biological systems, such as the central nervous system, which is a *de facto* hybrid composition of many diverse computational units (Von Neumann and Kurzweil, 2012). Hybrid approaches in predictive modeling problems try to exploit the strength of the component models, obtaining enhanced performance by their combinations. Even the *"no free lunch theorem"* suggests that there is no single computational view that can solve all problems (Wolpert, 2002). Historically, the idea of hybridization was first proposed by Chow (1965), who gave conditions for optimality of the joint decision of independent binary classifiers with appropriately defined weights. In 1979, Dasarathy and Sheela (1979) combined a linear classifier and a k-NN classifier and identified the region of the feature space where the classifiers disagree. Another work by Rastrigin and Erenstein (1981) performed a feature space partitioning first and then assigned to each partition region an individual classifier that achieves the best classification accuracy over it. Below we first describe the need for hybridization. Then we discuss some widely used hybrid predictive models that combine decision trees (DT) and neural nets (NN) to gain the mutual benefits of both approaches. Finally, we comment on the pros and cons of these popular hybrid approaches and the need for future research.

2.4.1 Need for Hybridization

The benefits from the hybridization of two learning algorithms can be formally written as follows:

- **Statistical issue:** It is often the case that the model space is too large to explore for limited training data, and that there may be several different models giving the same accuracy on the training data. The risk of choosing the wrong model can be reduced by combining two models, like DT and ANN.

- **Representation issue:** In many learning tasks, the true unknown hypothesis could not be represented by any hypothesis in the hypothesis space. By hybridization, it may be possible to expand the space of representable functions. Thus the learning algorithm may be able to form a more accurate approximation to the true unknown hypothesis.

- **Computational issue:** Many learning algorithms perform some kind of local search that may get stuck in local optima. Even if there are enough training data, it may still be challenging to find the best hypothesis. By combining two or more models, the risk of choosing a wrong local minimum can be reduced.

These three issues are among the most critical factors for which the traditional learning approaches fail. A learning algorithm that suffers from the statistical issue is generally said to have a high "variance", a learning algorithm that suffers from the computational issue can be described as having a high "computational variance", and a learning algorithm that suffers from the representational issue is generally said to have a high "bias". Therefore, through combination, the variance, as well as the bias of learning algorithms, may be reduced; this has been confirmed by many empirical studies (Bauer and Kohavi, 1999; Opitz and Maclin, 1999).

2.4.2 Hybrid Methods based on DT and NN

Substantial research has been published that combines decision trees and NNs. Several hybrid models have been developed over the last four decades, integrating the decision tree algorithm with neural networks (Sakar and Mammone, 1993; Sethi, 1990; Sirat and Nadal, 1990; Tsujino and Nishida, 1995). All of these methods have been shown to improve the decision tree and aid in interpretability. Others have used ANNs to improve the splitting in a decision tree. Here we provide an expanded review of related works based on the following categories:

1. Mapping Decision Trees to Neural Networks: The first work in the area of mapping DT into NNs was led by Paul. E. Utgoff in his seminal paper on 'Perceptron tree' (Utgoff, 1989). The hybrid model has the advantages of a hierarchical organization of DT, together with the perceptron's ability to deal with many variables. An extension of the idea, namely neural tree (NT), introduced by Sirat and Nadal (1990) for multi-class classification problems, includes a hierarchical net structure, a decision tree organization, and an efficient learning rule. Pan et al. (2003) implemented an intrusion detection system (IDS) model based on neural network and C4.5 algorithm viz. decision tree algorithm to solve the problem of network security. Sakar and Mammone (1993) introduced neural tree networks (NTN), which consists of neural networks connected in a tree structure. In NTN, the neural networks at each tree node are recursively used to partition the feature space into sub-regions, resulting in better classification performance. Later on, Foresti and Dolso (2004) introduced adaptive neural tree (ANT) and Chen et al. (2005) developed a flexible neural tree (FNT) for feature selection and intrusion detection problems. An experimental result by Chen et al. (2012) had shown that the FNT model is efficient for forecasting small-time scale traffic measurements and can produce the statistical features of real traffic measurements. A typical flexible neuron operator and a neural tree model with $+_n$ as the flexible neuron, x_i's as leaf nodes in the input layer, w_i's as weights and y as the output, are illustrated in Figure 2.1. Next, Rota Bulo and Kontschieder (2014) used an ANN to determine splits for random forest (ensemble of trees). Other seminal works include using a random forest to make predictions after designing an ANN (Kontschieder et al., 2015). Modern versions of Adaptive neural trees (ANTs) incorporates representation learning into edges, routing functions, and leaf nodes of a decision tree, along with a backpropagation-based training algorithm that adaptively grows the architecture from primitive modules (e.g., convolutional layers) (Abpeikar et al., 2020; Tanno et al., 2019). The ANT growing procedure is related to the progressive growth of NNs (Fahlman and Lebiere, 1990; Hansen and Salamon, 1990; Irsoy et al., 2012), or more broadly, the field of deep neural architecture search (Cortes et al., 2017). This allows the architectures of ANTs to adapt to the data available.

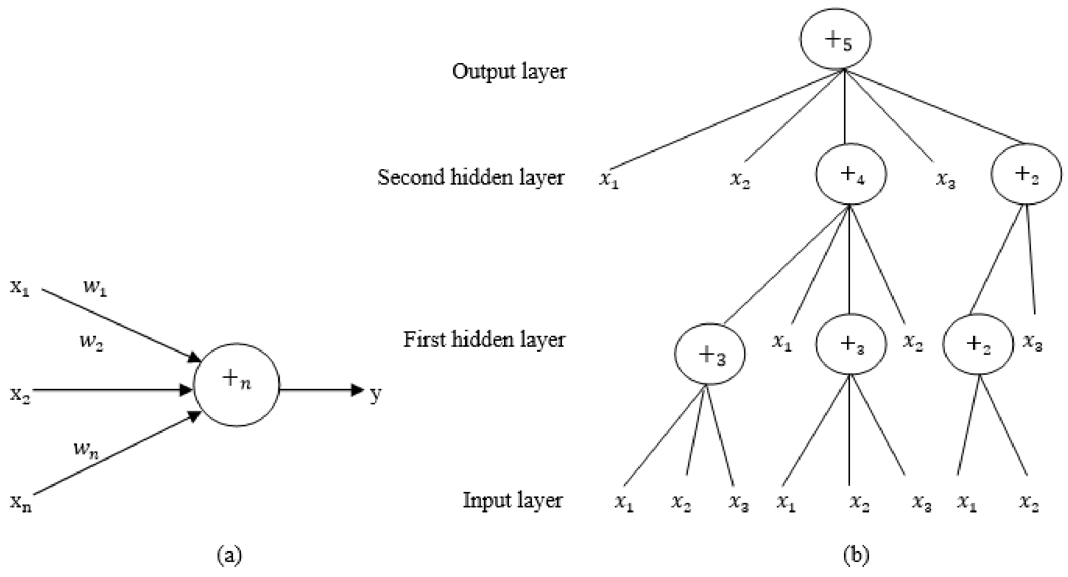


Figure 2.1: (a) A flexible neuron operator and (b) a typical representation of the FNT with function instruction set $F = \{+_2, +_3, +_4, +_5\}$, and terminal instruction set $T = \{x_1, x_2, x_3\}$.

2. Soft Decision Trees: Entropy nets (EN), a mapping of decision trees into a multilayered neural network structure, introduced by Sethi (1990), used the idea of pruning the DT using neural networks. The number of neurons in the input layer of the neural network equals the number of internal nodes of the decision tree. Then these neurons are run through the hidden layer, and finally, the number of neurons in the output layer equals the number of distinct classes. EN has the advantage of having relatively fewer neural connections and improving DT performance in a significant margin (Sethi, 1991, 1995). A real-life application of entropy nets is given by Tsai et al. (2012) to solve the complicated problem of water-stage predictions under the interaction of upstream flows and tidal effects during typhoon attacks in Taiwan. Their decision tree-based ANN model used the idea of splitting the feature space into areas by the CART and builds a locally specialized ANN model in each of the regions and is employed to river-stage predictions. Figure 2.2 represents a schematic diagram of the EN model. In a soft decision tree, the ANN is used to determine the splits. Based on the idea of Entropy nets, a soft decision tree (SDT) model was introduced by Suárez and Lutsko (1999). The hierarchical mixture of experts (HMEs) proposed by Jordan and Jacobs (1994) is a variant of SDTs whose routers are linear classifiers, and the tree structure is fixed. Léon and Denoyer (2016) recently proposed a more computationally efficient training method that can directly optimize hard-partitioning by differentiating through stochastic gradient estimators. More modern SDTs (Frosst and Hinton, 2017; Laptev and Buhmann, 2014; Rota Bulo and Kontschieder, 2014) have used multilayered perceptrons (MLP) or convolutional layers in the routers to learn more complex partitioning of the input space. However, the simplicity of identity transformers used in these methods means that input data is never transformed. Thus, each path on the tree does not perform representation learning, limiting their performance. Frosst and Hinton (2017) used a distilling neural network to build a new version of a 'soft decision tree'.

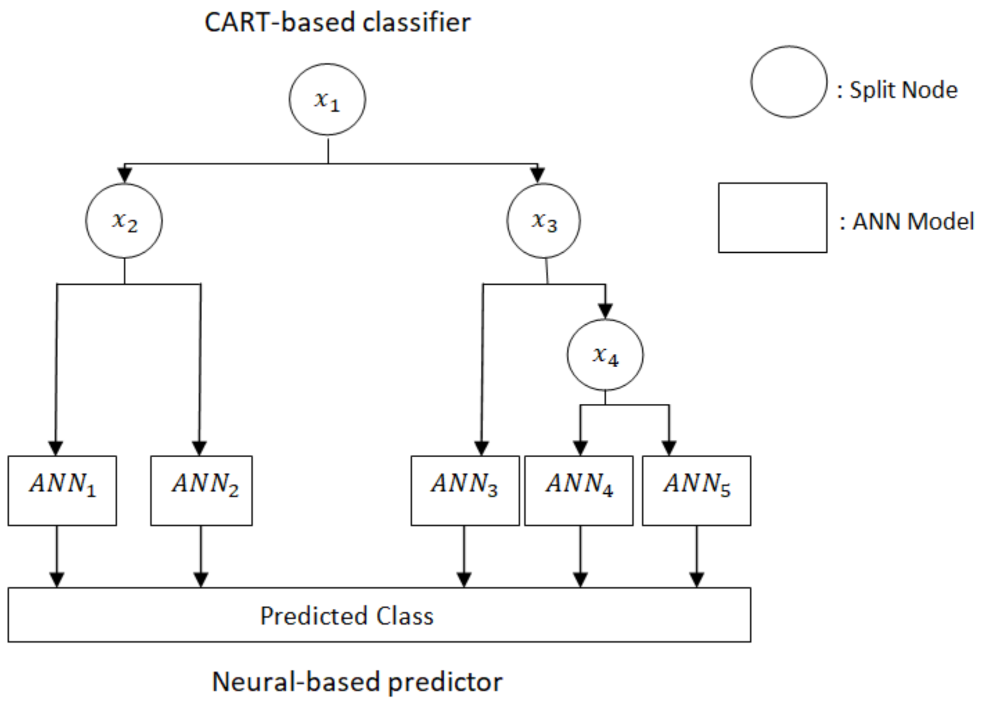


Figure 2.2: Schematic diagram of Entropy nets configuration.

3. Deep Neural architecture search using DT: Nowadays, it is possible to train NNs in an end-to-end fashion (Hinton et al., 2006). One area which still uses progressive growth is lifelong learning, in which a model needs to adapt to new tasks while retaining performance on previous ones (Xiao et al., 2014; Yoon et al., 2017). In particular, Xiao et al. (2014) introduced a method that grows a tree-shaped network to accommodate new classes. However, their method never transforms the data before passing it to the children classifiers, and hence never benefit from the parents representations. While we learn the architecture of an ANT in a greedy, layer-wise fashion, several other methods like evolutionary algorithms (Real et al., 2017; Stanley and Miikkulainen, 2002), sequential optimisation (Liu et al., 2018) and boosting (Cortes et al., 2017) are found extremely useful for complex architectures. More recent works suggested that integrating non-linear transformations of data into DTs would enhance model performance. The neural decision forest (NDF) (Kontschieder et al., 2015), which held a cutting-edge performance on ImageNet (Deng et al., 2009) in 2015, is an ensemble of DTs, each of which is also an instance of ANTs where the whole GoogleNet architecture (Szegedy et al., 2015) (except for the last linear layer) is used as the root transformer, prior to learning tree-structured classifiers with linear routers. Xiao (2017) employed a similar approach with an MLP at the root transformer, and is optimized to minimize a differentiable information gain loss. The conditional network proposed by Ioannou et al. (2016) sparsified convolutional neural network (CNN) architectures by distributing computations on hierarchical structures based on directed acyclic graphs with MLP-based routers, and design models with the same accuracy with reduced computing cost and less number of parameters. However, in all cases, the model architectures are pre-specified and fixed. These models are highly useful for high dimensional data sets with very large number of training samples (as in image/video/audio processing) (Dunson, 2018).

2.4.3 Advantages and Drawbacks of the Existing Hybrid Models and the Work Done in this Thesis

By the design, the hybrid models discussed in Subsection 2.4.2 inherit the following desirable properties from both DTs and NNs:

- **Architecture learning:** by progressively growing tree-to-network-mapped models, the architecture adapts to the availability and complexity of data. The growth procedure can be viewed as an architecture search with a hard constraint over the model class (Irsoy et al., 2012).

- **Representation learning:** as each root-to-leaf path in a soft decision tree is an NN, features can be learned end-to-end with gradient-based optimization. Combined with the tree structure, an SDT can learn such features which are hierarchically shared and separated (Frosst and Hinton, 2017).

- **Lightweight inference:** at inference time, NDFs and ANTs perform conditional computation, selecting a single root-to-leaf path on the tree on a per-sample basis, activating only a subset of the parameters of the model (Kontschieder et al., 2015; Tanno et al., 2019).

Despite the practical usage of these hybrid methods in the real-life problems of classification and regression, these methods incur several drawbacks. A few drawbacks of the existing neural trees, soft decision trees, and ANTs are listed below:

1. Accurate classification of high dimensional feature space leads to more depth trees, thus achieving less depth neural trees require more complex computations at each node (Abpeikar et al., 2020; Rani et al., 2015).

2. Regardless of the practical use of SDT and neural trees, theoretical properties like universal consistencies of these hybrid methods are unknown. Thus, one needs to analyze the data complexity for splitting, which leads to more accurate classification in the neural trees node.

3. The data partition of each neural tree node is considered as a sub-problem (Utgoff, 1989). Each sub-problem needs an eligible neural network, which leads to accurate and fewer depth trees (Foresti and Dolso, 2004). There are some studies in hybrid neural trees with different MLP, but there does not exist any neural tree with expert nodes.

4. Because of the huge amount of redundancy in high-dimensional feature spaces, the previously used hybrid models sometimes over-fit for small or moderate sample-sized data sets. In NDF, each node in their oblique decision tree involves all features rather than a single feature, which renders the model uninterpretable.

The interpretability of predictive models is important, especially in cases where ethics are involved, such as law, economics, medicine, business, and finance applications, where we wish to verify a model's reasoning manually. Deep neural networks (Goodfellow et al., 2016) have achieved excellent performance in many areas, such as computer vision, speech processing, and language modeling. However, the lack of interpretability prevents this family of black-box models from being used in applications. Moreover, in some areas like Business Intelligence, quality control, software reliability, etc., it is often more important to know how each factor contributes to the prediction rather than the conclusion itself. Thus, after generating a set of base learners, rather than trying to find the best single learner, hybrid methods resort to combinations for achieving a strong generalization ability, where the combination method plays a crucial role (Kuncheva, 2004).

This thesis develops some novel hybrid predictive models combining DTs and ANNs in Chapters 3-6 and ARIMA and ARNN in Chapter 7. The primary motivation of this thesis comes from the real-world data sets, with a variety of data types, such as business, macroeconomics, process efficiency improvement, water quality control, and software defect prediction. But as a secondary motivation, we emphasize on the development of hybrid models that are scalable (the size of the data does not pose a problem), robust (work well in a wide variety of problems), accurate (achieve higher predictive accuracy), statistically sound (have desired asymptotic properties), and easily interpretable. These hybrid methods are expected to outperform the current state-of-the-art and overcome the deficiencies of the current models available in the literature. Throughout the thesis, we discuss a motivational applied problem that triggers the development of novel hybrid predictive models. Finally, we establish asymptotic results for the proposed hybrid approaches along with relevant applications.

2.5 Performance Evaluation Metrics

We describe different performance metrics to be used in the subsequent chapters to assess the performances of the newly introduced hybrid frameworks in this thesis. To date, various performance metrics have been proposed and employed to evaluate the accuracy of the classification and regression models, but no single performance metric has been recognized as the universal standard. As a result, we need to assess the performance based on several different metrics for various problems.

2.5.1 Performance Metrics for Classification case

A confusion matrix is a matrix representation of the classification results. It contains data about real and predicted classifications done by a classification framework. Once the confusion matrix for a model is built, the Precision, Recall, F-measure, Accuracy percentage, and area under the receiver operating characteristic curve (AUC) are effectively ascertained (Bradley, 1997; Kubat and Matwin, 1997). The performance of different classifiers is evaluated based on these measures. The ratio of correctly predicted positive observations to the total predicted positive observations is called Precision, while the ratio of correctly predicted positive observations to the total observations in the actual class is called Recall. The harmonic mean of Precision and Recall is known as F-measure, while Accuracy is the ratio of correctly predicted observations to the total observations. AUC estimates the area under the ROC curve that illustrates the trade-off between detection and false alarm rates (Bradley, 1997). Higher the value of performance metrics, the better the classifier is. For symmetric data sets where the values of false positives and false negatives are almost the same, accuracy turns out to be the best measure. However, when we deal with imbalanced class distribution in a binary classification set up, F-measure and AUC are usually more useful than accuracy (Davis and Goadrich, 2006). These performance measures are used in Chapter 3 and Chapter 4 of the thesis. The expressions for different performance measures are as follows:

$\text{AUC} = \frac{\text{Recall}+\text{Specificity}}{2};$

F-measure $= 2\dfrac{(\text{Precision}\times\text{Recall})}{(\text{Precision}+\text{Recall})}$;
Accuracy $= \frac{(TP+TN)}{(TP+TN+FP+FN)}$;

Precision $= \frac{TP}{TP+FP}$; Recall $= \frac{TP}{TP+FN}$; Specificity $= \frac{TN}{FP+TN}$;
where, TP (True Positive): correct positive prediction; FP (False Positive): incorrect positive prediction; TN (True Negative): correct negative prediction; FN (False Negative): incorrect negative prediction.

2.5.2 Performance Metrics for Regression case

For evaluating the regression models, we use two absolute performance measures, viz. the mean absolute error (MAE) and the root mean squared error (RMSE); one relative measure, viz. the mean absolute percentage error (MAPE), and two "goodness of fit measures", i.e., the coefficient of determination (R^2) and adjusted R^2. These performance measures truly represent different angles to evaluate regression models. These performance measures are used in Chapter 5 to Chapter 7 of the thesis. The formulae used for the performance metrics are as follows (Hastie et al., 2009):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|,$$
$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|,$$
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2},$$
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \overline{Y})^2},$$
$$\text{Adjusted } R^2 = 1 - \frac{(1-R^2)(n-1)}{(n-d-1)},$$

where, $Y_i, \overline{Y}, \widehat{Y_i}$ denote the actual value, average value and predicted value of the dependent variable, respectively for the i^{th} instant. Here n, d denote the number of data used for performance evaluation and the number of independent variables. It is to be noted that lower values of MAE, MAPE, and RMSE, and higher values of R^2 and adjusted R^2 indicate better model performance.

Chapter 3

A Nonparametric Hybrid Model for Pattern Classification

Related Publications:

1. Chakraborty, T., Chakraborty, A. K., Murthy, C. A. (2019). A nonparametric ensemble binary classifier and its statistical properties. *Statistics & Probability Letters*, 149, 16-23.

2. Chakraborty, T., Chattopadhyay, S., Chakraborty, A. K. (2018). A novel hybridization of classification trees and artificial neural networks for selection of students in a business school. *Opsearch*, 55(2), 434-446.

Summary

Private business schools in India face a regular problem of picking quality students for their Master of Business Administration (MBA) programs to achieve the desired placement percentage. Selecting a wrong student may increase the number of unplaced students. Also, the more the number of unplaced students more is the negative impact on the institute's reputation. Business school authorities would, therefore, always want to ensure that they admit the right set of students to their MBA program. In this chapter, we propose a novel hybrid model based on classification tree (CT) and artificial neural network (ANN) to be referred to as a hybrid CT-ANN model to analyze and select the optimal academic characteristics of students to enhance their placement probability. Several statistical properties, including theoretical consistency and upper bound of an essential parameter of the proposed classifier, are derived. Our experimental findings show that the proposed hybrid CT-ANN model achieves greater accuracy in predicting students' placement than conventional supervised learning models. Our proposed hybrid classifier can also be used in a wide variety of feature selection cum classification problems that arise in other applied domains. We assess its performance using various other real-life classification data sets to show the general applicability of the proposed model.

3.1 Introduction

Out of the many reasons behind the closing down of many of the private business schools, the foremost one is the inability of the authorities to provide jobs to Master of Business Administration (MBA) students passing out of these business schools. The most challenging task for administrations is to find out the optimal set of parameters for choosing the right candidates in their MBA program, which will ensure the employability of the candidates. Attracting students in business schools are highly dependent on the business schools' placement records. If the right set of students are not selected for a few years, the number of unplaced students will certainly accumulate, resulting in the damage of reputation for the business school. One needs to develop a model so that the model ensures appropriate feature selection (selection of important students characteristics) with a decision on the optimal values or ranges of the features and higher prediction accuracy of the classifier.

Distribution-free classification models are specially used in the fields of statistics and machine learning for more than forty years now, mainly for their accuracy and ability to deal with high dimensional features and complex data structures. Two such models, CT and ANN, can model arbitrary decision boundaries. CT is found to be robust when limited data are available, unlike ANN. But decision trees are high variance estimators and greedy (Breiman et al., 1984), whereas neural networks are universal approximators (Hornik et al., 1989). Though advanced neural networks are highly complex with many free tuning parameters, and may over-fit when limited data are available (Dunson, 2018). To utilize the positive aspects of two powerful models, theoretical frameworks for combining both classifiers are often used jointly to make decisions. The ultimate goal of designing classification models is to achieve the best possible performance in terms of accuracy measures for the task at hand. This objective led researchers to create efficient hybrid models and prove their statistical properties to make their best use. Mapping tree-based models to neural networks allow exploiting the former to initialize the latter. Parameter optimization within the ANN framework will yield an intermediate model between CT and ANN, as found in the past literature (Sakar and Mammone, 1993; Sethi, 1990). Tsai neural tree model (Tsai et al., 2012) uses the idea of splitting the parameter space into areas by CT and builds in each of the regions a locally specialized ANN model (Sirat and Nadal, 1990). In deep neural tree model (Yang et al., 2018), a decision tree provides the final prediction, and it differs from conventional CT by introducing a global optimization of split and leaf node parameters using ANN. But the significant disadvantages of these algorithms are slow training, having many tuning parameters, easy sticking on local minima, and poor robustness (Tsai et al., 2012). These hybrid models are empirically shown to be useful in solving real-life problems, but the theoretical results are yet to be proved for many of them.

On the theoretical side, the literature is less conclusive, and regardless of their use in practical problems of classification, little is known about the statistical properties of these models. The most celebrated theoretical result has given the sufficient conditions for almost-sure L_1-consistency of histogram-based classification and data-driven density estimates (Lugosi and Nobel, 1996). In the case of neural networks, it is theoretically proven that if a one hidden layered (1HL) neural network is trained with an appropriately chosen number of neurons to minimize the empirical risk on the training data, it results in a universally consistent classifier (Faragó and Lugosi, 1993; Lugosi and Zeger, 1995). Devroye et al. (1996) have theoretically shown that there is some gain in considering two hidden layers (2HL), but it is not necessary to go beyond 2HL in ANN. In the case of hybrid models, the asymptotic results are less explored in the literature.

Motivated by the above discussion, we have proposed a hybrid CT-ANN model (Chakraborty et al., 2018, 2019c) that exploits the strengths of CT and ANN models to overcome their drawbacks. The approach is mainly developed for feature selection cum classification problems, which can solve the business school data set problem (refer to Section 3.5). Later different training schemes are experimentally evaluated on various small and medium-sized medical data sets having high dimensional feature spaces. The proposed model is found to be efficient for feature selection cum classification tasks. We have established the consistency results and upper bound for the model parameter of the hybrid CT-ANN model, which assures a basic theoretical guarantee of the efficiency of the proposed algorithm. In our model, we have used CT as a feature selection algorithm (Breiman et al., 1984) and have justified that CT output plays an essential role in the further model building using the ANN algorithm. The proposed ensemble CT-ANN model has the advantages of significant accuracy, a very less number of tuning parameters, and easy interpretability. The superiority of the proposed algorithm lies in its proven theoretical consistency using empirical risk minimization. Further, numerical evidence based on the business school data set shows the usefulness of the proposed model. Besides having the ability to deal with small and medium-sized data sets, our model is useful for the selection of essential features and performing classification tasks in other high-dimensional feature spaces and complex data structures also.

3.2 Motivating Example

Recently, many of the business schools are getting closed, especially in major cities in India (Ojha, 2017). There are many reasons behind the closing down of several business schools, out of which, a rise in the number of unplaced students played an important role. Administrators of the business schools thought it was due to their inability to detect the parameters that could be used to identify students who were ideal for the program as far as placement after the program is concerned (Ganatra and Dinesh Kumar, 2014). Placement is one of the major factors attracting a student to a business school. If non-placeable students are selected, then it affects the reputation of the institute, which further may increase the number of unplaced students in subsequent years. Also, there are always possibilities to reject a placeable candidate at the time of admission to the MBA program. In this chapter, we try to explore the important characteristics of students who enter a business school and relate them with the prospects of getting placed.

We want to develop a model that can help the authorities of a business school predict whether a student will be placed or not based on specific characteristics of that student available at the time of admission to the professional course. The advancement in the data mining field makes it possible to mine educational data to improve the quality of the educational processes (Asif et al., 2017). Different supervised learning algorithms were tried to predict various parameters connected with educational data (Pena Ayala, 2014; Romero and Ventura, 2010). Meedech et al. (2016) used decision trees and rule induction models to discover knowledge from the students' data in a university in Thailand to predict dropouts of students. Yadav and Pal (2012) used decision tree methodology to select students for admission in a particular course. Another critical study made by Kovacic (2010) was able to identify up to what extent the enrollment data containing characteristics of students can be used to pre-identify successful and unsuccessful students as the results of the final examination are concerned. However, the accuracy level achieved by the authors was only 60.5 percent. El Moucary (2011) used both ANN and classification and regression tree (CART) algorithm with important cross-validation and testing that enables authorities of an engineering college to decide about the selection criteria of engineering students in Masters' studies. Gupta et al. (2013) explored the socio-demographic variables that may help in pre-identifying successful and unsuccessful students. Their research suggests the use of CART to build a data warehouse for a particular university to predict future student admission. It is, however, clear that majority of the classifiers, namely, k-Nearest Neighbor (kNN), Linear Discriminant Analysis (LDA), Random Forest (RF), CART, Support Vector Machines (SVM), ANN and Logistic Regression (LR) hardly lift the accuracy level of prediction to around 60 percent.

In this chapter, classification trees and artificial neural networks are combined together to improve the accuracy of individual models. To illustrate the effectiveness of the hybrid CT-ANN approach, we performed this technique on the business school data. In addition to this, the number of hidden neurons is also varied to see its effect on the convergence rate. Our results indicate that the proposed combined approach predicts much more accurately and converges much faster than the conventional CART method or ANN approach or any other traditional supervised methodologies. Finally, we propose this hybrid CT-ANN model to select the optimal characteristics of the students looking to enter a business school, which will help them get a placement at the end of the curriculum. This is probably the first time such a hybrid model is used to solve this challenging problem. The performance of the proposed model has also been compared with conventional models to show that the proposed hybrid model works better for solving this real-life problem.

3.3 Formulation of the Proposed Model

Common theoretical frameworks for combining classifiers that use distinct pattern representations are often used jointly to make a decision. The ultimate goal of designing pattern classifiers is to achieve the best possible classification performance for the task at hand (Kim, 2016). This objective led us to the development of a new classifier for the problem of selecting optimal student-characteristics in a business school to predict the possibility of students' placement accurately at the end of the curriculum. Decision trees and neural networks are competitive techniques for modeling classification problems. Classification trees are hierarchical classifiers relatively superior to ANN in the readability of knowledge (Murthy, 1998). But, ANNs are better in implementing comprehensive inference over the inputs (Zhou et al., 2002).

CT is a nonparametric classification algorithm that has a built-in mechanism to perform feature selection (Quinlan, 1993). Unlike many other classification models, CT doesn't have any strong assumption about the normality of the data and homoscedasticity of the noise terms. In our proposed model, we first split the feature space into different areas by CT algorithm. Most important features are chosen using CT, and redundant features are eliminated. Then we build the ANN model using the important variables obtained through the CT algorithm along with prediction results made by the CT algorithm, which is used as an additional input feature in the neural networks. Then the ANN model is applied with one (hidden) layer to get the final classification results. The optimum value of the number of neurons in the hidden layer is derived in the next section. Since we have taken CT output as an input feature in the ANN model, the number of hidden layers is chosen to be one. The informal work-flow of the proposed model is as follows:

- First, apply the CT algorithm to train and build a decision tree and record important features. The prediction results of CT algorithm is also applied as an additional feature for further modeling.
- Using important input variables obtained from CT along with an additional input variable (CT output), a neural network is generated. We run one hidden-layered ANN algorithm with logistic sigmoidal activation function and record the classification results.

- The optimum number of neurons in the hidden layer of the model to be chosen as $O\left(\sqrt{\frac{n}{d_m log(n)}}\right)$, where n, d_m are the number of training samples and the number of input features in ANN model, respectively (to be discussed in Section 3.4).

The effectiveness of the hybrid CT-ANN model lies in selecting essential features using the CT model and incorporating CT predicted class levels as a feature in the ANN model. The inclusion of CT output as an input feature in the ANN model increases the dimensionality of feature space that results in better class separability as well. The fundamental work by Cover (1965) states that if the feature space is not densely populated, then in an intricate pattern classification problem (e.g., business school problem), the feature space becomes linearly separable in a high-dimensional space than in a low-dimensional space. Experimental results by Lee and Srihari (1995) have shown that the performance of the combined decision tree and ANN algorithm improves when more information is included. The theoretical set-up is presented in Section 3.4, which gives robustness and statistical aspects of the proposed model. A flowchart of the hybrid CT-ANN model is shown in Figure 3.1.

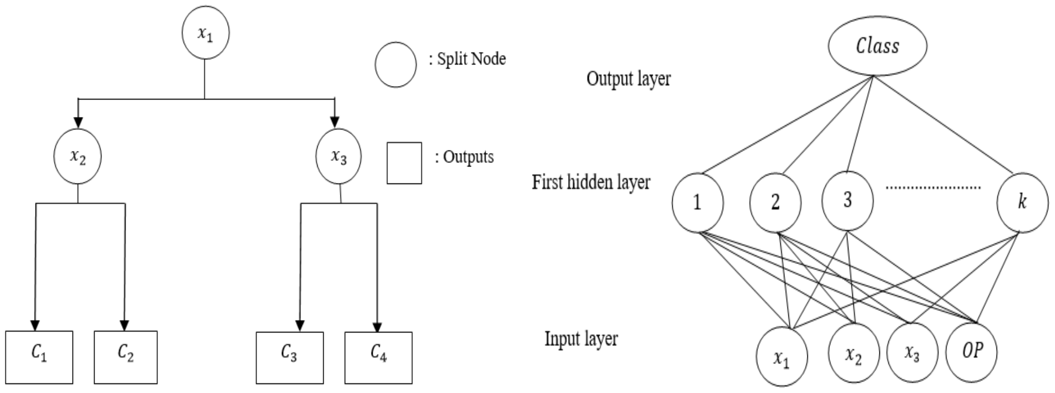


Figure 3.1: An example of Hybrid CT-ANN classifier with x_i, where $i = 1, 2, 3$, as important features obtained using CT, c_i be the leaf nodes and OP as CT output.

Our proposed model can be used for feature selection cum complex classification problems. On the theoretical side, it is necessary to show the universal consistency of the proposed model and other statistical properties for its robustness. The Hybrid CT-ANN model is a two-step problem-solving approach, such as initial feature selection followed by improving the model using an optimum ANN technique. The proposed model, when applied to the business school problem, can choose the optimal characteristics of the students that affect the placements along with accurate future predictions. We will experimentally show that the proposed model works better for the business school data than the other existing supervised models available in the literature. Our proposed methodology can be used for selecting features of items that will satisfy a specific goal and also can be employed for modeling such complex situations. The flowchart of the hybrid CT-ANN model developed is presented in Figure 3.2.

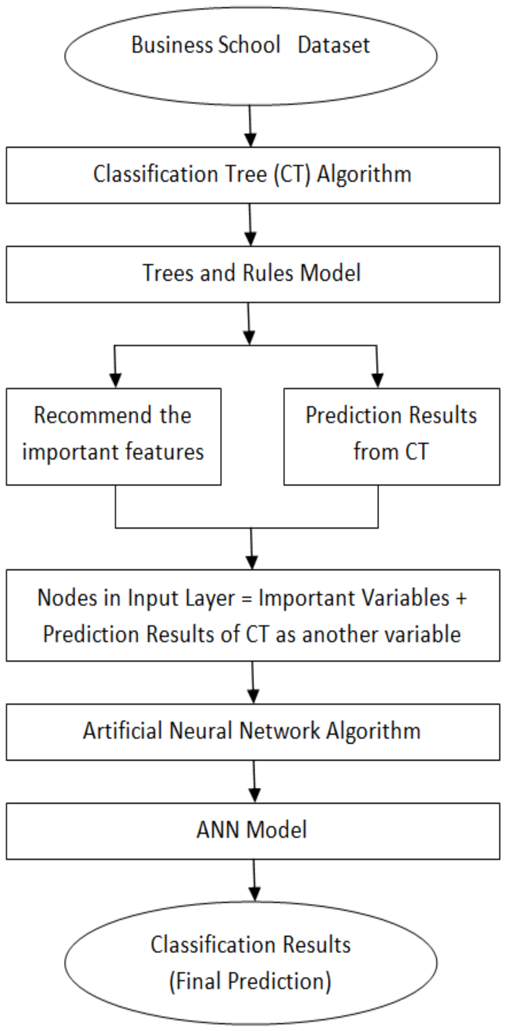

Business School Dataset
Classification Tree (CT) Algorithm
Trees and Rules Model
Recommend the important features
Prediction Results from CT
Nodes in Input Layer = Important Variables + Prediction Results of CT as another variable
Artificial Neural Network Algorithm
ANN Model
Classification Results (Final Prediction)

Figure 3.2: Flowchart of proposed hybrid CT-ANN Model

Remark 2 *One of the major challenges involved in our proposed algorithm is to correctly choose the number of neurons in the hidden nodes in the specialized ANN model. It was previously shown that too few hidden nodes limit network generalization capabilities, while too many hidden layers can result in over-training of the neural network (Devroye et al., 1996). We solved this problem by providing a theoretical bound for the number of hidden neurons in the hidden layer of the ANN model in Section 3.4 that plays a vital role in this novel method of hybridization.*

3.4 Statistical Properties of the Proposed Model

Our proposed ensemble classifier has the following nomenclature. First, it extracts essential features from the feature space using the CT algorithm. Then it builds one hidden layered ANN model with the essential features extracted using CT along with CT outputs as an additional feature. In this section, we investigate the statistical properties of the proposed ensemble CT-ANN model by introducing a set of regularity conditions for consistency of CT, followed by the importance of CT outputs for further model building. And finally, we will discuss the consistency results of ANN algorithm with an optimal value of the number of neurons in the hidden layer of the model.

Let $\underline{X}$ be the space of all possible values of p features and C be the set of all possible binary outcomes. We are given a training sample with n observations $L = \{(X_1, C_1), (X_2, C_2), ..., (X_n, C_n)\}$, where $X_i = (X_{i1}, X_{i2}, ..., X_{ip}) \in \underline{X}$ and $C_i \in C$. Also let $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ be a partition of the feature space $\underline{X}$. We denote $\widetilde{\Omega}$ as one such partition of Ω.

Define $L_{\omega_i} = \{(X_i, C_i) \in L : X_i \in \omega_i, C_i \in C\}$ as the subset of L induced by ω_i and let $L_{\widetilde{\Omega}}$ denote the partition of L induced by $\widetilde{\Omega}$. The information gain (to be introduced later) from the feature space is used to partition $\underline{X}$ into a set $\widetilde{\Omega}$ of nodes and then we can construct a classification function on $\widetilde{\Omega}$. There exists a partitioning classification function $d : \widetilde{\Omega} \to C$ such that d is constant on every node of $\widetilde{\Omega}$. Now, let us define $\widehat{L}$ to be the space of all learning samples and $\mathbb{D}$ be the space of all partitioning classification function, then $\Phi : \widehat{L} \to \mathbb{D}$ such that $\Phi(L) = (\psi \circ \phi)(L)$, where ϕ maps L to some induced partition $(L)_{\widetilde{\Omega}}$ and ψ is an assigning rule which maps $(L)_{\widetilde{\Omega}}$ to d on the partition $\widetilde{\Omega}$. The most basic reasonable assigning rule ψ is the plurality rule $\psi_{pl}(L_{\widetilde{\Omega}}) = d$ such that if $x \in \omega_i$, then

$$d(\underline{x}) = \arg\max_{c \in C} |L_{c,\omega_i}|.$$

The plurality rule is used to classify each new point in ω_i as belonging to the class (either 0 or 1 in this case) most common in L_{ω_i}. This rule is very important in proving risk consistency of the CT algorithm. Now, let us define a binary split function $s(\omega_i)$, that maps one node to a pair of nodes, i.e., $s(\omega_i) = (s_1(\omega_i), s_2(\omega_i)) = (\omega_1, \omega_2)$, then $\omega_1 \cup \omega_2 = \omega_i$, $\omega_1 \cap \omega_2 = \phi$ and $\omega_1, \omega_2 \neq \phi$. Binary split function partitions a parent node $\omega_i \subseteq \underline{X}$ into a non-empty child nodes ω_1 and ω_2, called left child and right child node respectively. A set of all potential rules that we will use to split $\underline{X}$ is a finite set $S = \{s_1, s_2, ...s_m\}$.

Define G as a goodness of split criterion which maps (ω_i, s) for all $\omega_i \in \underline{X}$ and $s \in S$ to a real number. For any parent node ω_i, the goodness of split criterion ranks the split functions. We have used the following impurity function as goodness of split criterion:

$$G(L_{\omega_i}, s) = H(L_{\omega_i}) - \frac{|L_{s_1(\omega_i)}|}{|L_{\omega_i}|} H(L_{\omega_1(t)}) - \frac{|L_{s_2(\omega_i)}|}{|L_{\omega_i}|} H(L_{\omega_2(t)})$$

where, H is taken as Gini Index and can be written as follows:

$$H_{gini}(\omega_i) = \sum_{c \neq c'} \frac{|L_{\omega_i,c}|}{|L_{\omega_i}|} . \frac{|L_{\omega_i,c'}|}{|L_{\omega_i}|}.$$

This criterion assesses the quality of a split $s(\omega_i)$ by subtracting the average impurity of the child nodes ω_1, ω_2 from the impurity of the parent node ω_i. The stopping rule in CT is decided based on the minimum number of the split in the posterior sample called 'minsplit' to be denoted by $r(\omega_i)$. If $r(\omega_i) \geq \alpha$, then ω_i will split into two child nodes and if $r(\omega_i) < \alpha$, then ω_i is a leaf node and no more split is required. Here α is determined by the user, though for practice, it is usually taken as 10% of the training sample size.

A binary tree-based classification and partitioning scheme Φ is defined as an assignment rule applied to the limit of a sequence of induced partitions $\phi^{(i)}(L)$, where $\phi^{(i)}(L)$ is the partition of the training sample L induced by the partition $(\phi_i \circ \phi_{i-1} \circ .... \circ \phi_1)(\underline{X})$. For every node ω_i in a partition $\widetilde{\Omega}$ such that $r(\omega_i) \geq \alpha$, then the function $\phi(\widetilde{\Omega})$ splits each node into two child nodes using the binary split in the question set as determined by G. For other nodes $\omega_i \in \widetilde{\Omega}$ such that $r(\omega_i) < \alpha$, then $\phi(\widetilde{\Omega})$ leaves ω_i unchanged. We write

$$\Phi(L) = (\psi \circ \lim_{i \to \infty} \phi^{(i)})(L) \tag{3.1}$$

where, $\phi^{(i)}(L) = L_{(\phi_i \circ \phi_{i-1} \circ .... \circ \phi_1)(\underline{X})}$. CT as an axis-parallel split on $\mathbb{R}^p$ splits a node by dividing it into child nodes consisting of either side of some hyperplane. We need to show that the CT scheme is well defined, which will be possible only if there exists some induced partition L' such that

$$\lim_{i \to \infty} \phi^{(i)}(L) = L'.$$

In fact, we need to show that the following lemma holds:

Lemma 5 *If L is a training sample and $\phi^{(i)}$ is defined as above, then there exists $N \in \mathbb{N}$ such that for $n \geq N$*

$$\phi^{(n)}(L) = \lim_{i \to \infty} \phi^{(i)}(L) \tag{3.2}$$

Proof Let $\{L_{\widetilde{\Omega}}\}$ denote the sequence $\{L, \phi^1(L), \phi^2(L), \dots\}$. Defining

$$\omega_i^{max} = \max\{\omega_i \in \widetilde{\Omega}_i : r(\omega_i) > \alpha\}$$

as the size of the largest non-leaf node(s) in $\widetilde{\Omega}_i$. Suppose there exists $N \in \mathbb{N}$ such that $(\omega_i)_N^{max}$ does not exist. Then every node in $\widetilde{\Omega}_N$ is a leaf. For all $n > N$, $\widetilde{\Omega}_n = \widetilde{\Omega}_N$, then (3.2) holds.

The sequence $\{|\omega_i^{max}|\}$ is strictly decreasing if it exists. Further if ω_{i+1}^{max} exists then $|\omega_{i+1}^{max}| \leq |\omega_i^{max}| - 1$ and $|\omega_i^{max}| \geq 1$ and $|\omega_1^{max}| = |L|$. This means that $(\omega_i)_{|L|}^{max}$ can not exist, so (3.2) always holds with $N \leq |L|$. □

For a wide range of partitioning schemes, the consistency of histogram classification schemes based on data-dependent partitions was shown in Lugosi and Nobel (1996). To introduce the theorem, we need to define the followings:
For any random variable X and set A, let

$$\eta_{n,X}(A) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \in A)$$

be the empirical probability that $X \in A$ based on n observations and $I(z)$ denotes the indicator of an event C. Now let $\mathfrak{T} = (\widetilde{\Omega}_1, \widetilde{\Omega}_2, ...)$ be a finite collection of partitions of a measurement space $\underline{X}$. Define maximal node count of $\mathfrak{T}$ as the maximum number of nodes in any partition $\widetilde{\Omega}$ in $\mathfrak{T}$ which can be written as $\lambda(\mathfrak{T}) = \sup_{\widetilde{\Omega}\in\mathfrak{T}} |\widetilde{\Omega}|$. Also let, $\Delta(\mathfrak{T}, L) = |\{L_{\widetilde{\Omega}} : \widetilde{\Omega} \in \mathfrak{T}\}|$ be the number of distinct partitions of a training sample of size n induced by partitions in $\mathfrak{T}$. Let $\Delta_n(\mathfrak{T})$ be the growth function of $\mathfrak{T}$ defined as

$$\Delta_n(\mathfrak{T}) = \sup_{\{L:|L|=n\}} \Delta(\mathfrak{T}, L).$$

Growth function of $\mathfrak{T}$ is the maximum number of distinct partitions of $L_{\widetilde{\Omega}}$ which partitions $\widetilde{\Omega}$ in $\mathfrak{T}$. Take any class $A \subseteq \mathbb{R}^p$, $S_n(A) = \max_{\{B\subset\mathbb{R}^p:|B|=n\}} |A \cap B : A \in A|$ is the maximum number of partitions of B induced by A, where B is some n point subset of $\mathbb{R}^p$, called shatter coefficient (Devroye et al., 1996). For a partition $\widetilde{\Omega}$ of X, let $\widetilde{\Omega}[x \in X] = \{\omega \in \widetilde{\Omega} : x \in \omega\}$ be the node ω in $\widetilde{\Omega}$ which contains x.
For a set $A \subseteq \mathbb{R}^p$, let

$$D(A) = \sup_{x,y\in A} \parallel x - y \parallel$$

be the diameter of A. Now, for the sake of completeness we are rewriting the Theorem 2 of Lugosi and Nobel (1996) in our context:

Theorem 10 *Let $(\underline{X}, \underline{Y})$ be a random vector taking values in $\mathbb{R}^p \times C$ and L be the set of first n outcomes of $(\underline{X}, \underline{Y})$. Suppose that Φ is a partition and classification* *scheme such that $\Phi(L) = (\psi_{pl} \circ \phi)(L)$, where ψ_{pl} is the plurality rule and $\phi(L) = (L)_{\tilde{\Omega}_n}$ for some $\tilde{\Omega}_n \in \mathcal{T}_n$, where $\mathcal{T}_n = \{\phi(\ell_n) : P(L = \ell_n) > 0\}$. Also suppose that all the binary split functions in the question set associated with Φ are hyperplane splits. As $n \to \infty$, if the following regularity conditions hold:*

$$\frac{\lambda(\mathcal{T}_n)}{n} \to 0 \tag{3.3}$$

$$\frac{log(\triangle_n(\mathcal{T}_n))}{n} \to 0 \tag{3.4}$$

and for every $\gamma > 0$ and $\delta \in (0, 1)$,

$$\inf_{S \subseteq \mathbb{R}^p : \eta_x(S) \geq 1-\delta} \eta_x(x : diam(\tilde{\Omega}_n[x] \cap S) > \gamma) \to 0 \tag{3.5}$$

with probability 1. then Φ is risk consistent.

Proof For the proof of Theorem 10, one may refer to Lugosi and Nobel (1996). □

Remark 3 *Now instead of considering histogram-based partitioning and classification schemes, we are going to show the risk consistency of CT as defined above. We can produce a simultaneous result with conditions 3.3 and 3.4 of Theorem 10 replaced by a simple condition. However, the shrinking cell condition 3.5 of Theorem 10 is assumed.*

Theorem 11 *Suppose $(\underline{X}, \underline{Y})$ be a random vector in $\mathbb{R}^p \times C$ and L be the training set consisting of n outcomes of $(\underline{X}, \underline{Y})$. Let Φ be a classification tree scheme such that $\Phi(L) = (\psi_{pl} \circ \lim_{i\to\infty} \phi^{(i)})(L)$ where, ψ_{pl} is the plurality rule and $\phi(L) = (L)_{\tilde{\Omega}_n}$ for some $\tilde{\Omega}_n \in \mathcal{T}_n$, where $\mathcal{T}_n = \{\lim_{i\to\infty} \phi^{(i)}(\ell_n) : P(L = \ell_n) > 0\}$. Suppose that all the split function in CT in the question set associated with Φ are axis-parallel splits. Finally if for every n and $w_i \in \tilde{\Omega}_n$, the induced subset L_{w_i} has cardinality $\geq k_n$, where $\frac{k_n}{log(n))} \to \infty$ and 3.5 holds true, then Φ is risk consistent.*

Proof Since $|w_i| \geq k_n \quad \forall \quad w_i \in \tilde{\Omega}_n$, we can write

$$|\tilde{\Omega}_n| \leq \frac{n}{k_n} \tag{3.6}$$

for every $\tilde{\Omega}_n \in \mathcal{T}_n$ and in that case, we will have

$$\frac{\lambda(\mathcal{T}_n)}{n} \leq \frac{1}{k_n}$$

.
As $n \to \infty$, we can see $\frac{1}{k_n} \to 0$ which gives $\frac{\lambda(\mathcal{T}_n)}{n} \to 0$. Hence condition 3.3 holds true.

Now for every $\tilde{\Omega}_n \in \mathcal{T}_n$, using Cover's theorem (Cover, 1965), any binary split of $\mathbb{R}^p$ can divide n points in $\mathbb{R}^p$ in at most n^p ways. Using equation (3.6), we can write $\Delta_n(\mathcal{T}_n) \leq (n^p)^{\frac{n}{k_n}}$ and consequently

$$\frac{log(\Delta_n(\mathcal{T}_n))}{n} \leq p\frac{log(n)}{k_n} \tag{3.7}$$

As $n \to \infty$, R.H.S. of equation (3.7) goes to 0. So condition 3.4 of Theorem 10 also holds and hence the theorem. $\square$

Remark 4 *Note that no assumptions are made on the distribution of the pair $(\underline{X}, \underline{Y}) \in \mathbb{R}^p \times C$. Also sub-linear growth of the number of cells (condition 3.3) and sub-exponential growth of a combinatorial complexity measure (condition 3.4) are not required, instead a more flexible restriction such as if each cell of L_{ω_i} has cardinality $\geq k_n$ and $\frac{k_n}{log(n)} \to \infty$, then CT is said to be risk consistent. So, feature selection using the CT algorithm is justified, and now we are going to check the importance of CT output in further model building. It is also noted that the choice of important features based on CT is a greedy algorithm. The optimality of local choices of the best feature for a node doesn't guarantee that the constructed tree will be globally optimal (Kuncheva, 2004).*

Using CT given features, a list of essential features are selected from p available features. It is noted that the CT output also plays an important role in further modeling. To see the importance of CT given classification results (to be denoted by OP in the rest of the chapter) as a relevant feature, we introduce a nonlinear measure of the correlation between any feature and the actual class levels, namely C-correlation (Yu and Liu, 2004), as follows:

Definition 9 *C-correlation is the correlation between any feature F_i and the actual class levels C, denoted by $SU_{F_i,C}$. Symmetrical uncertainty (SU) (Press et al., 1992) is defined as follows:*

$$SU(X,Y) = 2\left[\frac{H(X) - H(X|Y)}{H(X) + H(Y)}\right] \tag{3.8}$$

where, $H(X)$ is the entropy of a variable X and $H(X|Y)$ is the entropy of X while Y is observed.

We can heuristically decide a feature to be highly correlated with class C if $SU_{F_i,C} > \beta$, where β is a relevant threshold to be determined by users. Using Definition 9, we can conclude that OP can be taken as a non-redundant feature for further model building. This also improves the performance of the model at a significant rate, to be shown in Section 3.5.

Now, we build an ANN model with CT selected features and OP as another input feature in the ANN model. The dimension of the input layer in the ANN model, denoted by d_m ($\leq p$), is the number of essential features obtained by CT + 1. We will use one hidden layer in the ANN model due to the incorporation of OP as input information in the model. It should be noted that one-hidden layer neural networks yield strong universal consistency, and there is a little theoretical gain in considering two or more hidden layered neural networks (Devroye et al., 1996). In the hybrid CT-ANN model, we have used one hidden layer with k neurons. This makes the proposed hybrid binary classifier less complicated and less time consuming while implementing the model.

After elimination of redundant features by the CT and incorporating OP as an another input vector, let us now consider the following training sequence $\xi_n = \{(Z_1, Y_1), ..., (Z_n, Y_n)\}$ of n i.i.d copies of $(\underline{Z}, \underline{Y})$ taking values from $\mathbb{R}^{d_m} \times C$. A classification rule realized by a one-hidden layered feedforward neural network is chosen to minimize the empirical L_1-risk, where the L_1 error of a function $\psi : \mathbb{R}^{d_m} \to \{0, 1\}$ is defined by $J(\psi) = E\{|\psi(Z) - Y|\}$. Before stating the sufficient conditions for the consistency of the algorithm and optimal number of nodes (k) in hidden layer for practical use of the model, let us define the followings:

Definition 10 *A sigmoidal function $\sigma(z) = \frac{1}{1+exp(-z)}$ is called a logistic squasher if it is non-decreasing and satisfies $\lim_{z\to\infty} \sigma(z) = 0$ and $\lim_{z\to-\infty} \sigma(z) = 1$.*

The next theorem is based on the seminal work of Lugosi and Zeger (1995), where they proved the results for universal consistency of the feedforward neural networks in case of regression function estimation problem. Since this is a bit different classification framework and the results of the Theorem will be useful in finding the bounds for k, we state the Theorem and outline the idea of the proof below.

Theorem 12 *Consider a neural network with one hidden layer with bounded output weight having k hidden neurons and let σ be a logistic squasher. Let $F_{n,k}$ be the class of neural networks with logistic squasher defined as*

$$F_{n,k} = \left\{ \sum_{i=1}^{k} c_i \sigma(a_i^T z + b_i) + c_0 : k \in \mathbb{N}, a_i \in \mathbb{R}^{d_m}, b_i, c_i \in \mathbb{R}, \sum_{i=0}^{k} |c_i| \leq \beta_n \right\}$$

and let ψ_n be the function that minimizes the empirical L_1 error over $\psi_n \in F_{n,k}$. It can be shown that if k and β_n satisfy

$$k \to \infty, \quad \beta_n \to \infty, \quad \frac{k\beta_n^2 log(k\beta_n)}{n} \to 0$$

then the classification rule

$$g_n(z) = \begin{cases} 0, & \textit{if } \psi_n(z) \leq 1/2. \\ 1, & \textit{otherwise.} \end{cases} \tag{3.9}$$

is universally consistent.

Proof To show the universal consistency, it is sufficient to show that $J(\psi_n) - J^* \to 0$ in probability, where $J(\psi_n) = E\{|\psi_n(Z) - Y||\xi_n\}$ and $J^* = \inf_{\psi_n} J(\psi_n)$ (Devroye et al., 1996). We write

$$J(\psi_n) - J^* = \left(J(\psi_n) - \inf_{\psi \in F_{n,k}} J(\psi)\right) + \left(\inf_{\psi \in F_{n,k}} J(\psi) - J^*\right)$$

where, $(J(\psi_n) - \inf_{\psi \in F_{n,k}} J(\psi))$ is called estimation error and $(\inf_{\psi \in F_{n,k}} J(\psi) - J^*)$ is called approximation error, as described in Section 2.2.
Now, to handle the approximation error, let $\psi^{'} \in F_{n,k}$ be a function such that

$$E\{|\psi^{'}(Z) - g^*(Z)|\} \leq E\{|\psi(Z) - g^*(Z)|\}$$

for each $\psi \in F_{n,k}$. Clearly,

$$\begin{aligned}\inf_{\psi \in F_{n,k}} J(\psi) - J^* \leq J(\psi^{'}) - J^* & \\ &= E\{|\psi^{'}(Z) - Y|\} - E\{|g^*(Z) - Y|\} \\ &\leq E\{|\psi^{'}(Z) - g^*(Z)|\}\end{aligned}$$

which converges to zero as $n \to \infty$. To handle estimation error, let us write

$$\begin{aligned} J(\psi_n) - \inf_{\psi \in F_{n,k}} J(\psi) &\leq 2 \sup_{\psi \in F_{n,k}} |J(\psi) - J_n(\psi)| \\ &= 2 \sup_{\psi \in F_{n,k}} \left| E\{|\psi(Z) - Y|\} - \frac{1}{n}\sum_{i=1}^{n} |\psi(Z_i) - Y_i| \right| \end{aligned} \tag{3.10}$$

Define the class $M_{n,k}$ of functions on $\mathbb{R}^{d_m} \times \{0,1\}$ by

$$M_{n,k} = \left\{ m(z,y) = \left| \sum_{i=1}^{k} c_i \sigma(a_i^T z + b_i) + c_0 - y \right| : a_i \in \mathbb{R}^{d_m}, b_i \in \mathbb{R}, \sum_{i=0}^{k} |c_i| \leq \beta_n \right\}$$

Then the upper bound of equation (3.10) becomes

$$2 \sup_{m \in M_{n,k}} \left| E\{m(Z,Y)\} - \frac{1}{n}\sum_{i=1}^{n} m(Z_i, Y_i) \right|$$

Applying uniform law of large numbers we can observe for each $m \in M_{n,k}$ that

$$\begin{aligned} m(z,y) &= \left| \sum_{i=1}^{k} c_i \sigma(a_i^T z + b_i) + c_0 - y \right| \\ &\le 2max\left(\sum_{i=1}^{k} c_i \sigma(a_i^T z + b_i) + c_0, 1 \right) \\ &\le 2max\left(\sum_{i=1}^{k} |c_i|, 1 \right) \\ &\le 2\beta_n \end{aligned}$$

For large n and $\beta_n \ge 1$, using Theorem 4 (Pollard, 1984), we write

$$\begin{aligned} P\Bigg\{ \sup_{m \in M_{n,k}} \left| E\{m(Z,Y)\} - \frac{1}{n} \sum_{i=1}^{n} m(Z_i, Y_i) \right| > \epsilon \Bigg\} \\ \le 8E\{N(\epsilon/8, M_{n,k}(D_n))\} e^{-n\epsilon^2/(512\beta_n^2)} \end{aligned} \tag{3.11}$$

where $N(\epsilon, M_{n,k}(D_n))$ denotes the l_1-covering number of the random set

$$M_{n,k}(D_n) = \{(m(Z_1, Y_1), ..., (m(Z_n, Y_n) : m \in M_{n,k}\} \subset \mathbb{R}^n.$$

Observe that for $m_1, m_2 \in M_{n,k}$ with $m_1(z,y) = |\psi_1(z) - y|$ and $m_2(z,y) = |\psi_2(z) - y|$, for any probability measure ν on $\mathbb{R}^{d_m} \times \{0,1\}$ with μ as the marginal of ν on $\mathbb{R}^{d_m}$,

$$\int |m_1(z,y) - m_2(z,y)| \nu(d(z,y)) \le \int |\psi_1(z) - \psi_2(z)| \mu(dz).$$

It follows from above that

$$N(\epsilon, M_{n,k}(D_n)) \le N(\epsilon, F_{n,k}(Z^n)), \text{ where } Z^n = (Z_1, Z_2, ..., Z_n).$$

It is evident that an upper bound on the covering number of the class of neural networks $F_{n,k}$ is also the upper bound on the quantity of our interest. Now, we define three collections of functions:

$$\begin{aligned} G_1 &= \{a^T z + b; a \in \mathbb{R}^{d_m}, b \in \mathbb{R}\}, \\ G_2 &= \{\sigma(a^T z + b); a \in \mathbb{R}^{d_m}, b \in \mathbb{R}\}, \\ G_3 &= \{c\sigma(a^T z + b); a \in \mathbb{R}^{d_m}, b \in \mathbb{R}, c \in [-\beta_n, \beta n]\}. \end{aligned}$$

Using Corollary 1, we can write the Vapnik-Chervonenkis (VC) dimension of the class of sets $G_1^+ = \{(z,t) : t \le \psi(z), \psi \in G_1\}$ is $V_{G_1^+} \le d_m + 2$. This implies $V_{G_2^+} \le d_m + 2$ (Pollard, 1990).

So, for any $z^n = (z_1, z_2, ..., z_n)$, using Theorem 5 (Haussler, 1992), we write:

$$N(\epsilon, G_2(z^n)) \leq 2\left(\frac{4}{\epsilon}\right)^{2(d_m+2)}$$

where $G_2(z^n) = \{z^{'} \in \mathbb{R}^n : z^{'} = (g(z_1), g(z_2), ..., g(z_n)), g \in G_2\}$. Using Theorem 6 (Pollard, 1990), we estimate covering numbers of $G_3(z^n)$:

$$N(\epsilon, G_3(z^n)) \leq \frac{4}{\epsilon} N(\epsilon/(2\beta_n), G_2(z^n)) \leq \left(\frac{8e\beta_n}{\epsilon}\right)^{2(d_m+5)}$$

Now if $\beta_n > 2/e$, we obtain the following:

$$N(\epsilon, F_{n,k}(z^n)) \leq \frac{2\beta_n(k+1)}{\epsilon} N(\epsilon/(k+1), G_3(z^n))^k \leq \left(\frac{8e(k+1)\beta_n}{\epsilon}\right)^{k(2d_m+5)+1}.$$

Substituting this bound into equation (3.11), we get for large n,

$$\begin{aligned} P\left\{\sup_{\psi \in F_{n,k}} \left| E|\psi(Z) - Y| - \frac{1}{n}\sum_{j=1}^{n} |\psi(Z_j) - Y_j| \right| > \epsilon \right\} \\ \leq 8\left(\frac{64e(k+1)\beta_n}{\epsilon}\right)^{k(2d_m+5)+1} e^{-n\epsilon^2/(512\beta_n^2)} \end{aligned} \tag{3.12}$$

which tends to zero if

$$\frac{k\beta_n^2 log(k\beta_n)}{n} \to 0.$$

It is easy to see that if we assume the following: if there exists δ (> 0) such that

$$\frac{\beta_n^2}{n^{1-\delta}} \to 0,$$

then strong universal consistency follows by applying the Borel-Cantelli Lemma (Lemma 4) to the last probability in equation (3.12). $\square$

Next, we find the optimal choice of k using the regularity conditions of strong universal convergence and the idea of obtaining upper bounds on the rate of convergence, i.e., how fast $J(\psi_n)$ approaches to zero (Györfi et al., 2002).

To get an upper bound on the rate of convergence, we will have to impose some regularity conditions on the posteriori probabilities. In the case of the rate of convergence of estimation error, we will have a distribution-free upper bound (Faragó and Lugosi, 1993). To obtain the optimal value of k, it is enough to find upper bounds of the estimation and approximation errors. The upper bound of approximation error investigated by Barron (1993) is given in Lemma 6.

Lemma 6 *Assume that there is a compact set $E \subset \mathbb{R}^{d_m}$ such that $Pr\{Z \in E\} = 1$ and the Fourier transform $\widetilde{P_0}(w)$ of $P_0(z)$ satisfies*

$$\int_{\mathbb{R}^{d_m}} |\omega||\widetilde{P_0}(\omega)|d\omega < \infty$$

then

$$\inf_{\psi \in F_{n,k}} E\Big(f(Z,\psi) - P_0(Z)\Big)^2 \leq \frac{c}{k},$$

where c is a constant depending on the distribution.

Remark 5 *The previous condition on Fourier transformation and extensive discussion on the properties of functions satisfying the condition (including logistic squasher function) is given in Barron (1993).*

Proposition 1 *For a fixed d_m, let $\psi_n \in F_{n,k}$. For the artificial neural networks satisfying the regularity conditions of universal consistency and the conditions of Lemma 6, the optimal choice of k is $O\left(\sqrt{\frac{n}{d_m log(n)}}\right)$.*

Proof Applying Cauchy-Schwarz inequality in Lemma 6, we can write

$$\inf_{\psi \in F_{n,k}} E\,|f(Z,\psi) - P_0(Z)| = O\left(\frac{1}{\sqrt{k}}\right)$$

It is well known from Devroye et al. (1996) that for any ψ

$$J(\psi) - J^* \leq 2E\,|f(Z,\psi) - P_0(Z)|$$

So, the upper bound of approximation error is found to be $O\left(\frac{1}{\sqrt{k}}\right)$.

It is noted that the approximation error goes to zero as the number of neurons goes to infinity for a universally consistent classifier. For practical implementation, the number of neurons is often fixed (e.g., can't be increased with the size of the training sample). Now, it is enough to bound the estimation error.

Let us define $r(\psi_n) = E(J(\psi_n)) = P(\psi_n(Z) \neq Y)$ is the average error probability of ψ_n. Using Lemma 3 of Faragó and Lugosi (1993), we can write that the estimation error is always $O\left(\sqrt{\frac{kd_m log(n)}{n}}\right)$.
Bringing the above facts together, we can write

$$r(\psi_n) - J^* = O\left(\sqrt{\frac{kd_m log(n)}{n}} + \frac{1}{\sqrt{k}}\right)$$

Now, to find optimal value of k, the problem reduces to equating $\sqrt{\frac{kd_m log(n)}{n}}$ with $\frac{1}{\sqrt{k}}$, which gives $k = O\left(\sqrt{\frac{n}{d_m log(n)}}\right)$. □

Remark 6 *We can see a remarkable property of the proposed hybrid CT-ANN model from Proposition 1. Since for this class of posteriori probability function as shown in Lemma 6, the rate of convergence does not necessarily depend on the dimension, in the sense that the exponent being a constant, it can be concluded that the proposed model does not suffer from the 'curse of dimensionality'.*

The optimal value of hidden nodes is found to be $O\left(\sqrt{\frac{n}{d_m log(n)}}\right)$ for the universally consistent hybrid CT-ANN model. For practical use, if the data set is small or medium sample-sized, the recommendation is to use $\left(\sqrt{\frac{n}{d_m log(n)}}\right)$ for achieving utmost accuracy of the proposed model. The practical usefulness and competitiveness of the proposed classifier in solving real life imbalanced business school data problem is shown in Section 3.5.

3.5 Application to Business School Data

In this section, we first describe the business school data in brief that are used in this study. Subsequently, we will report the experimental results and compare our proposed hybrid CT-ANN model with other state-of-the-art classifiers.

3.5.1 Data Description

The data was provided by a private business school that receives a huge number of applications from across the country for the MBA program and admits a pre-specified number of students every year (Chakraborty et al., 2018; Ganatra and Dinesh Kumar, 2014). The data set comprises of several parameters of passed out students' profile along with their placement information. We divided the data into a training set (80 percent of the records) for building the model and test set (20 percent of the records) to check the accuracy of the model. We aim to select the optimal set of features and the corresponding optimal supervised model for selecting the right set of students who will be fit for the MBA program of the business school and, at the end of the program, will be able to get a placement. The data set contains 24 explanatory variables, out of which 7 are categorical variables and others are continuous variables. The response variable indicates whether the student got placed or not. Sample data set is given in Table 3.1. We also applied 5×2 cross-validation while evaluating classifiers on the data sets. Each data set is broken into class-stratified halves, allowing two experiments in each half, one is used as training and others as testing.

Table 3.1: Sample business school data set.

ID	Gender	SSC Percentage	HSC Percentage	DEGREE Percentage	E.Test Percentile	SSC Board	HSC Board	HSC Stream	Placement
1	Male	68.4	85.6	72	70	ICSE	ISC	Commerce	Yes
2	Male	59	62	50	79	CBSE	CBSE	Commerce	Yes
3	Male	65.9	86	72	66	Others	Others	Commerce	Yes
4	Female	56	78	62.4	50.8	ICSE	ISC	Commerce	Yes
5	Female	64	68	61	24.3	Others	Others	Commerce	No
6	Female	70	55	62	89	Others	Others	Science	Yes
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

3.5.2 Analysis of Results

Before we apply the proposed model for analyzing the data, we explored a few of the popular models used for binary classified data. Among them, LR, LDA, kNN method, and SVM were used. To increase the accuracy of the prediction of these methods, dimensionality reduction of the feature variables was carried out using the correlation matrix and information value provided by each of the independent variables. LR gave a maximum accuracy of 77.143 percent, which turns out to be the same when we applied LDA, a popular alternative to logistic regression. For the k-NN method, the value of k is chosen to minimize the cross-validation error. The k-NN method with $k = 13$ increased the accuracy level of prediction to 80 percent. However, another supervised model SVM obtained the accuracy level to 75 percent.

Decision tree algorithms like RF and CART are again quite popular application methodologies for classification problems. RF is a combination of several decision trees, and hence its accuracy level, quite naturally, be higher. In the present context, it gave a prediction accuracy of 82.857 percent on test data. The CART model uses a Gini index of diversity with 24 variables. The variable importance indicator C_p was used for selecting variables to enter and leave the CART model. Based on the results of CART, four variables viz. HSC Percentage, Degree Percentage, SSC Percentage, and Entrance test Percentile were chosen in the final model. The predictive accuracy of the CART model further improved to 83.333 percent. The optimal classification tree generated by the CART model is given in Figure 3.3.

ANN model is again quite a popular supervised learning methodology that is used in predicting data in classification problems. We have normalized our data before training a neural network because avoiding normalization may lead to a complicated training process. A min-max method is used for scaling the data in the interval $[0, 1]$. Usually, 2/3 of the input layer size is taken as the number of neurons (Zhou et al., 2002). Three hidden layers with the configuration $24 : 16 : 6 : 2 : 1$ is used. The input layer has 24 inputs, with three hidden layers having 16, 6, and 2 neurons, and the output layer has a single output. Similarly, the neural trees algorithm and entropy nets model were also used, and the accuracy levels were compared.

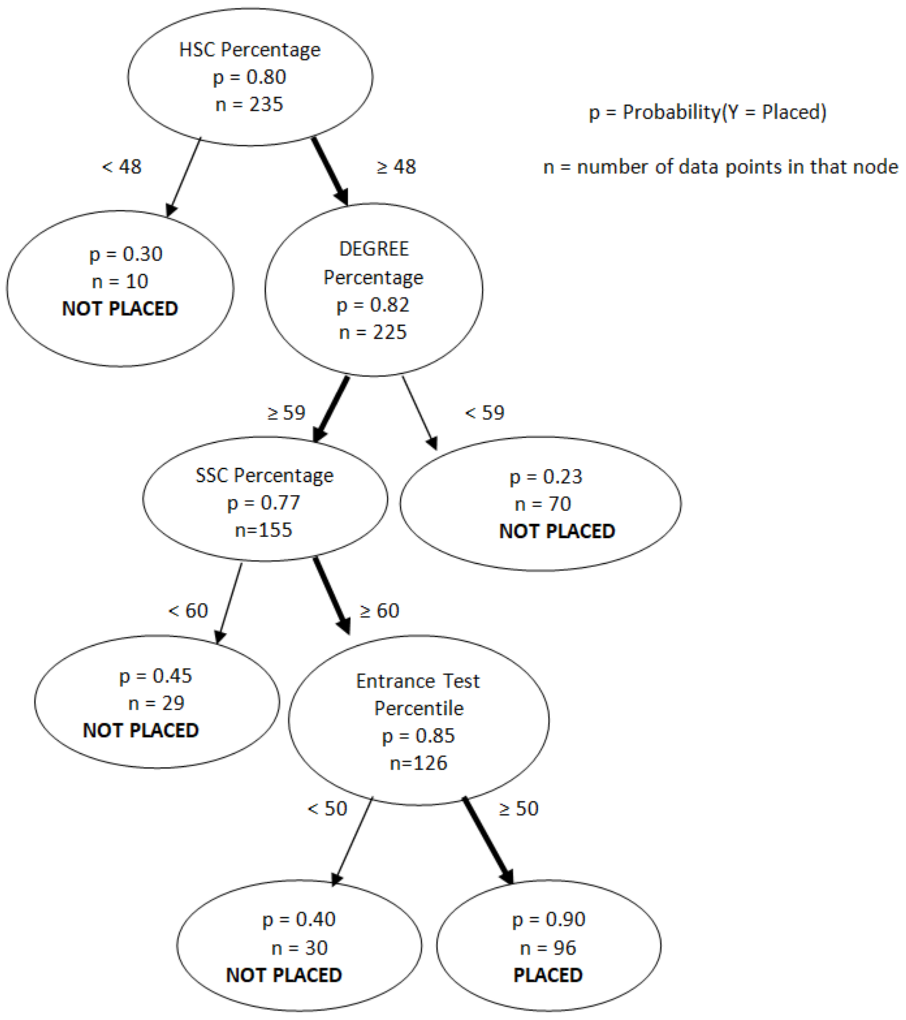


Figure 3.3: The optimal classification tree generated by CART

Finally, we apply our proposed CT-ANN model to the business school data set. There are three layers, namely, input, hidden, and output layer included in the proposed CT-ANN model. The input layer has five nodes, out of which the first four input values are based on the identification results made by CT. The prediction results produced by the CT model were used as the other input information for training the neural network. The neural network is illustrated in Figure 3.4, which contains an input layer with five features, one hidden layer with ten neurons and one single output layer. Various performance measures obtained using different algorithms (average results after cross-validation) are presented in Table 3.2. It is clear from Table 3.2 that the accuracy level of our proposed hybrid CT-ANN model is 91.667 percent, by far the best among the models used for this data. For comparisons, the predictive results using the confusion matrix for LR, LDA, kNN, RF, CART, SVM, ANN, and the hybrid models, including our proposed model, are summarized in Table 3.2.

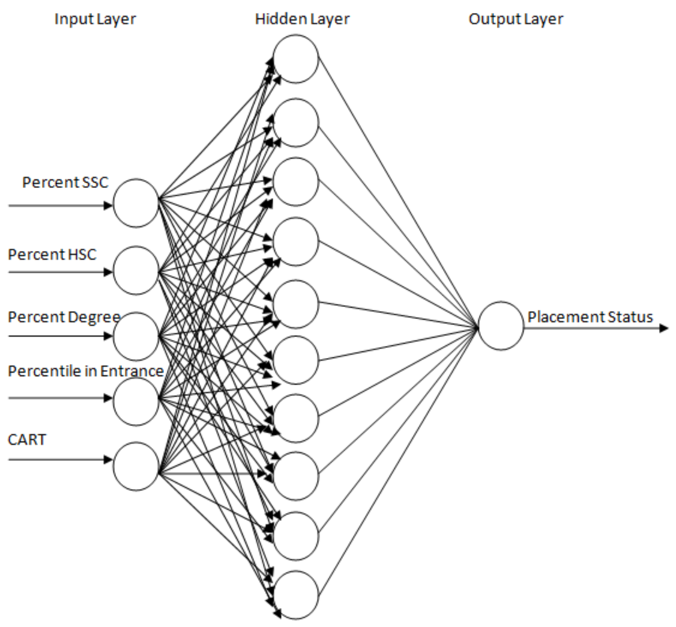


Figure 3.4: Hybrid CT-ANN topology for business school placement data

Table 3.2: Quantitative measures of performance for different classifiers.

Classifier	Precision	Recall	F-measure	Accuracy
LR	0.964	0.794	0.871	77.143
LDA	0.964	0.794	0.871	77.143
kNN	0.800	1.000	0.889	80.000
SVM	0.964	0.771	0.857	75.000
RF	0.823	1.000	0.903	82.857
CART	0.823	1.000	0.903	83.333
ANN	0.928	0.812	0.867	77.142
Neural Trees	0.918	0.894	0.906	85.169
Entropy Nets	0.839	0.928	0.881	80.555
Hybrid CT-ANN	**0.942**	**0.970**	**0.956**	**91.667**

Table 3.2 also suggests that the proposed model of CT-ANN outperforms other supervised models in terms of F-measure. Hence, our proposed hybrid CT-ANN model for the data under discussion turns out to be the 'best' model for predicting whether a student will be placed or not, after s/he passes out of MBA. The proposed model will help the authorities of the private business school in recruiting prospective students with a higher chance of placements.

3.6 Experiments with UCI Data

In this section, we apply the proposed hybrid CT-ANN model to various medical data sets available at UCI public repository. The data sets and implementation results are described, and this shows the general applicability of the proposed classifier for feature selection cum classification problems in different domains.

3.6.1 Data

The proposed model is evaluated using six publicly available medical data sets from the UCI Machine Learning repository (https://archive.ics.uci.edu/ml/datasets.html) dealing with various diseases. These binary classification data sets have a limited number of observations and high-dimensional feature spaces. The breast cancer data set has 9 discrete features, whereas the pima diabetes data set consists of 8 continuous features in the input space (Rodriguez et al., 2006). Heart disease data set contained a total of 303 examples for 13 continuous features initially, out of which 6 contained missing class values, and 27 are disputed cases that were removed from the data set. Promoter gene sequences data set has 57 sequential DNA nucleotide attributes. SPECT images data set is represented by 22 binary features that have either 0 or 1 values, but the data set is imbalanced. Wisconsin breast cancer data set consists of 699 examples carrying 9 continuous features in the input space (Kurgan et al., 2001). Table 3.3 gives a summary of these data sets.

Table 3.3: Characteristics of the data sets used in experimental evaluation

Data	Classes	Objects (n)	Number of feature (p)	Number of (+)ve instances	Number of (−)ve instances
breast cancer	2	286	9	85	201
heart disease	2	270	13	120	150
pima diabetes	2	768	8	500	268
promoter gene sequences	2	106	57	53	53
SPECT heart images	2	267	22	55	212
wisconsin breast cancer	2	699	9	458	241

3.6.2 Experimental Results

In order to show the impact of the proposed 2-step pipeline model, it is applied to the high-dimensional small or medium-sized medical data sets. These are such types of data sets in which not only classification is the task but also feature selection plays a vital role before it. We shuffled the observations in each of the six medical data sets randomly and split it into training, validation, and test data sets in a ratio of $50 : 25 : 25$. We have also repeated each of the experiments ten times with different randomly assigned training, validation and testing data sets.

Our proposed algorithm is compared with Classification Tree (CT), Random Forest (RF), Support Vector Machine (SVM), Artificial Neural Network (ANN) with 1HL and 2HL, entropy nets, Tsai's neural trees (NT) (Tsai et al., 2012), deep neural decision trees (DNDT) (Yang et al., 2018) based on the different performance metrics. All these classifiers other than DNDT are implemented in R Statistical software on a PC with a 2.1GHz processor with 8GB memory. We compared the proposed model with 1-HL ANN and 2-HL ANN without employing feature selection. Since the data sets are small and medium-sized, going beyond 2HL ANN will over-fit the data set (Devroye et al., 1996), and this is also reminiscent of universal approximation theorem (Hornik et al., 1989). For 1HL ANN, the number of hidden neurons used is $k \approx \sqrt{n}$ (Devroye et al., 1996), and for 2HL ANN, 2/3 of the input sizes are taken as the number of neurons in the 1st HL and 1/3 of the input sizes in case of 2nd HL (Zhou et al., 2002). Similarly, Tsai's NT was also built, and the accuracy levels were compared. DNDT searches tree structure and parameter with stochastic gradient descent, which was implemented in TensorFlow (Abadi et al., 2016), and it is a kind of GPU-accelerated computing (Yang et al., 2018). Breiman's random forest (Breiman, 2001) also has an in-built feature selection mechanism which was implemented using *'party'* implementation in R and results are reported in Table 3.4.

To apply our proposed model to the medical data sets, we first apply CT with 'minsplit' as 10% of the training sample size using the *'rpart'* package implementation in R statistical software. CT model uses the Gini index of diversity with the available input feature space. The variable importance indicator C_p was used to select variables to enter or leave the CT model. Based on the results of CT, important variables or features were chosen in the final model along with CT output. The number of reduced features after feature selection using CT is reported in Table 3.4. The number of hidden neurons in the hidden layer is calculated using this formula $k = \sqrt{\frac{n}{d_m log(n)}}$, where n is the number of training samples and d_m as the number of input features in neural networks. We have further normalized the data sets before training the neural network. Min-max method is used for scaling the data in an interval of $[0, 1]$. Our model recommends using the upper bound of the number of neurons in the HL of the ensemble model for small or medium data sets. The ensemble CT-ANN model is trained using *'neuralnet'* implementation in R. Training time and memory requirement for our proposed model is quite low compared with DNDT, which needs the availability of GPU. Table 3.4 gives the obtained results from different classifiers used for experimental evaluation over six medical data sets.

We can conclude from Table 3.4 that the proposed model achieves the overall highest accuracy for most of the data sets while working with reduced features as compared to other state-of-the-art and remains competitive for other few data sets as well. The proposed hybrid CT-ANN performed 'best' among the other classifiers considered in this study for four out of six data sets and remained competitive for the rest of the two data sets.

Table 3.4: Performance measures (mean values and their standard deviation) of different classification algorithms over six medical data sets

Classifiers	Data set	The number of (reduced) features after feature selection	Classification accuracy (%)	F-measure
CT	breast cancer	7	68.26 (6.40)	0.70 (0.07)
	heart disease	7	76.50 (4.50)	0.81 (0.03)
	pima diabetes	6	71.85 (4.94)	0.74 (0.03)
	promoter gene sequences	17	69.43 (2.78)	0.73 (0.01)
	SPECT heart images	9	75.70 (1.56)	0.78 (0.00)
	wisconsin breast cancer	8	94.20 (2.98)	0.89 (0.01)
RF	breast cancer	6	69.00 (7.30)	0.72 (0.07)
	heart disease	8	80.19 (4.23)	0.84 (0.01)
	pima diabetes	6	73.49 (4.12)	0.76 (0.03)
	promoter gene sequences	20	71.26 (1.97)	0.75 (0.03)
	SPECT heart images	10	79.70 (1.23)	0.82 (0.01)
	wisconsin breast cancer	8	95.75 (2.01)	0.96 (0.02)
SVM	breast cancer	9	64.62 (5.21)	0.68 (0.05)
	heart disease	13	78.95 (4.89)	0.83 (0.01)
	pima diabetes	8	70.39 (3.56)	0.72 (0.03)
	promoter gene sequences	57	59.35 (1.37)	0.63 (0.02)
	SPECT heart images	22	**83.46** (1.29)	**0.85** (0.00)
	wisconsin breast cancer	9	93.30 (2.78)	0.94 (0.01)
ANN (with 1HL)	breast cancer	9	61.58 (5.89)	0.64 (0.04)
	heart disease	13	73.56 (5.44)	0.79 (0.02)
	pima diabetes	8	66.78 (4.58)	0.69 (0.04)
	promoter gene sequences	57	61.77 (3.46)	0.65 (0.02)
	SPECT heart images	22	79.69 (0.23)	0.81 (0.01)
	wisconsin breast cancer	9	94.80 (2.01)	0.96 (0.01)
ANN (with 2HL)	breast cancer	9	62.20 (5.12)	0.64 (0.03)
	heart disease	13	78.81 (3.96)	0.82 (0.03)
	pima diabetes	8	69.78 (3.89)	0.73 (0.02)
	promoter gene sequences	57	63.46 (2.19)	0.68 (0.02)
	SPECT heart images	22	82.71 (0.78)	0.84 (0.01)
	wisconsin breast cancer	9	95.60 (2.54)	0.96 (0.10)
Entropy Nets	breast cancer	7	69.00 (6.25)	0.72 (0.05)
	heart disease	7	79.59 (4.78)	0.83 (0.01)
	pima diabetes	6	69.50 (4.05)	0.72 (0.02)
	promoter gene sequences	17	66.23 (1.98)	0.70 (0.01)
	SPECT heart images	9	76.64 (1.70)	0.78 (0.01)
	wisconsin breast cancer	8	95.96 (2.18)	0.96 (0.00)
Tsai's NT	breast cancer	7	69.45 (7.17)	0.71 (0.07)
	heart disease	7	80.25 (4.68)	0.85 (0.01)
	pima diabetes	6	71.59 (4.19)	0.74 (0.03)
	promoter gene sequences	17	70.67 (2.83)	0.74 (0.02)
	SPECT heart images	9	76.95 (1.27)	0.78 (0.01)
	wisconsin breast cancer	8	**97.40** (2.11)	**0.98** (0.01)
DNDT	breast cancer	8	66.12 (7.81)	0.68 (0.08)
	heart disease	7	81.05 (3.89)	0.86 (0.02)
	pima diabetes	6	69.21 (5.08)	0.72 (0.05)
	promoter gene sequences	17	69.06 (1.75)	0.71 (0.01)
	SPECT heart images	10	75.50 (0.89)	0.77 (0.00)
	wisconsin breast cancer	7	94.25 (2.14)	0.95 (0.00)
Hybrid CT-ANN	breast cancer	7	**72.80** (6.54)	**0.77** (0.06)
	heart disease	7	**82.78** (4.78)	**0.89** (0.02)
	pima diabetes	6	**76.10** (4.45)	**0.79** (0.04)
	promoter gene sequences	17	**75.40** (1.50)	**0.79** (0.01)
	SPECT heart images	9	81.03 (0.56)	0.82 (0.00)
	wisconsin breast cancer	8	97.30 (1.05)	0.98 (0.00)

3.7 Simulation Study

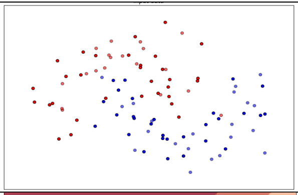
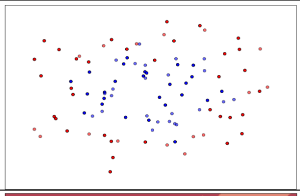
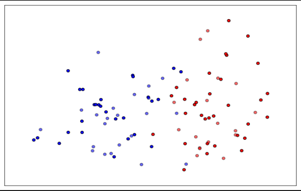
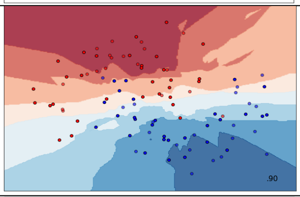
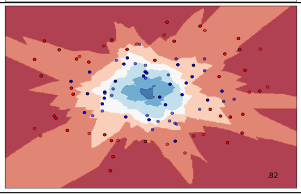
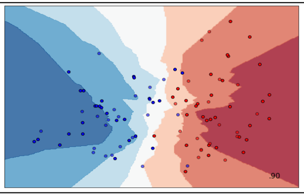
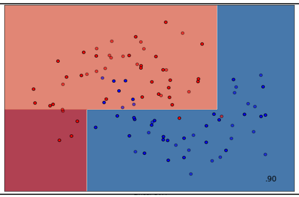
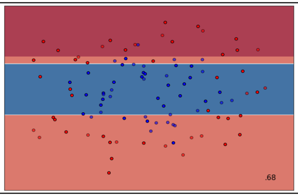
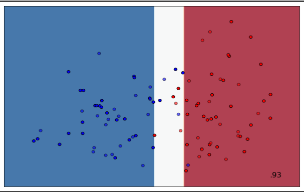
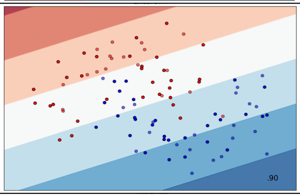
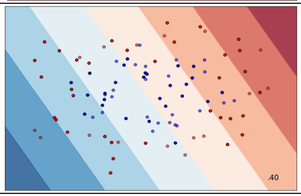
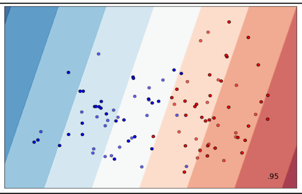
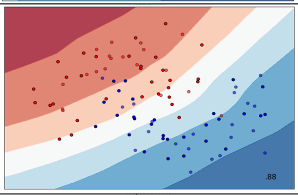
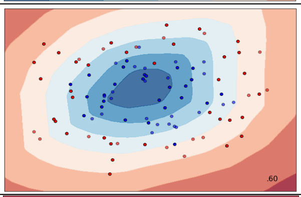
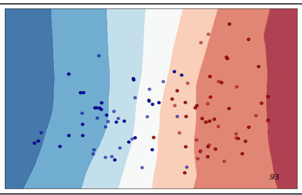
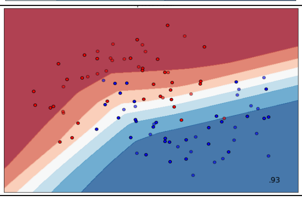
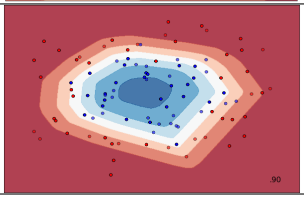
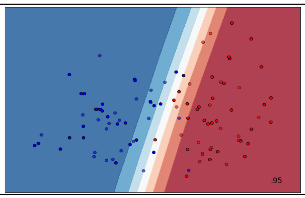
This section provides a comparison of several classifiers on synthetic data sets from SCIKIT-LEARN library in Python. The point of this example is to illustrate the nature of the decision boundaries of different classifiers. Three popularly used toy data sets are generated to visualize the decision boundaries of the classification algorithms used in this chapter. The details of the data generation process are described with codes here: `https://github.com/scikit-learn/scikit-learn/blob/0fb307bf3/sklearn/datasets/_samples_generator.py`. The 'make_moons' function in SCIKIT-LEARN library generates a two moons data set where we take the number of samples to be 100 with the moderate noise level. We added Gaussian noise to the data with the standard deviation equals to 0.3. The 'make_circles' function in SCIKIT-LEARN generates a binary classification problem with data sets that fall into concentric circles. Again, as with the moons test problem, we can control the amount of noise in the shapes. This test problem is suitable for algorithms that can learn complex nonlinear manifolds. We generate a circle data set with some noise, refer to the input data plots in Table 3.6. This creates a large circle containing a smaller circle. Lastly, we generate linearly-separable data with added noise. All these toy data sets represent binary classification problems in 2D. In all the experiments, 60% of the data samples are used for training, and the rest 40% of the data are for testing. Classification accuracy of all the models in three synthetic data sets is reported in Table 3.5.

Table 3.5: Classification accuracy percentage of different classifiers on three synthetic data sets. Best results in the Table are made **bold**.

Classifiers	Moon data	Circle data	Linearly-separable data
kNN	90	82	90
CT	90	68	93
Linear SVM	90	40	95
ANN	88	60	93
Hybrid CT-ANN	**93**	**90**	**95**

To assess our proposed hybrid CT-ANN model, we perform experiments on these data sets by employing all the algorithms and generate a graph representation of each data set. The implementation of all these models is done as in Section 3.6. The choice of tuning parameters for all the models is as follows. For the kNN model, the value of k is varied between 2 to 7 and the best results are reported for all the synthetic data sets. In the case of CT, the "Gini" index used as the tree splitting criteria for all the data sets, and the maximum depth of the tree is set to 5 for all the examples. In the case of linear SVM, we set the regularization parameter for all these toy data sets at 0.025. While implementing ANN, we used the logistic sigmoidal function with a stochastic gradient-based optimizer, namely 'adam' by Kingma and Ba (2014). Finally, our proposed model is applied to the data sets, and tuning parameters are chosen as described in Section 3.3. As shown in the plots of Table 3.6, our approach can correctly classify the majority of the points and achieves the highest classification accuracy in comparison with the single classifiers. This result seems to confirm that the proposed approach can deal with complex data structure, i.e., nearby points and points on the same structure are likely to have the same label.

Table 3.6: A comparison of several classifiers on synthetic data sets. The plots show training points in solid colors and testing points semi-transparent. The lower right in each plot shows the classification accuracy on the test set.

Synthetic data Classifiers	Moon data set	Circle data set	Linearly-separable data set
Input Data			
kNN			
CT			
Linear SVM			
ANN			
Hybrid CT-ANN			

3.8 Conclusions and Discussion

One of the aims of this chapter was to develop a model for selecting optimal student-characteristics so that the chosen characteristics would ensure placement for students. Our study presented a hybrid CT-ANN model that combines both the artificial neural network and classification tree, which gives more accuracy than other competitive models. We have found CT to be the optimal technique for feature selection and found a hybrid CT-ANN model to be the optimal supervised model for accurate prediction of a student's placement potential. Significant accuracy of 91.667 percent has been achieved through the use of our experimentally optimized model. Consequently, the CT-ANN successfully demonstrates the best performance and offers a practical solution to the problem of finding optimal criteria for selecting students in a business school. It can also be used for modeling the selection of students based on past reports of placement. The proposed CT-ANN model may be used for similar problems like choosing possible customers in a city for a particular product based on the information available about the customers who bought that specific product in other cities as well as those customers who did not buy that particular product.

From the general applicability point of view, the newly introduced hybrid classifier achieved higher accuracy in classification performance with minimal computational cost (by working with a subset of input features). Our proposed feature selection cum classification model is robust in nature. The proposed hybrid CT-ANN model is universally consistent and less time consuming during the actual implementation. We have also found the optimal value of the number of neurons in the hidden layer so that the user will have less tuning parameters to be controlled. When applied to real-life data sets, the proposed model performed better compared to other state-of-the-art models for most of the data sets and remained competitive for the few different data sets. Besides the theoretical results, the proposed model performs well on several data sets involving feature selection cum classification task in a supervised setting.

In the light of current advances in ANN, one might ask a simple question: What is the need for a two-step pipeline (like hybrid CT-ANN model) over advanced ANN models? A straight-cut answer to this question could be unwise. The primary goal of 'statistics' is to make scientific inferences from the model compared to building a "black-box-like" model that may perform well for some specific data sets, but may not be considered a general theory (Dunson, 2018). Our proposed model is robust, universally consistent, easily interpretable, and highly useful for high dimensional small or medium-sized data sets (for example, medical data sets) to perform feature selection cum classification tasks. Advanced ANN models (say, deep neural net) are highly complex, over-parameterized models and found useful when the data sets are huge (like image, audio, and video data sets) (Dunson, 2018).

Nevertheless, no model can have a dominant advantage in all aspects, and there will always be a trade-off between accuracy, interpretability, and complexity of the model (Wolpert, 1996). In recent work, we have improved the hybrid CT-ANN model, especially for imbalanced data sets, which also involves feature selection as a task before imbalanced pattern classification (Chakraborty and Chakraborty, 2020b). Situations when feature selection is not a job (e.g., software defect prediction data sets) in classification problems, our model may not perform well. But the ensemble classifier will have an edge where the data analysis requires important variable selections in the early stage followed by predictions using classifiers for limited data sets. In the next chapter, we will try to solve the problem of imbalanced pattern classification paradigm arising in software reliability engineering with another novel hybrid approach.

Chapter 4

Hellinger Net : A Hybrid Model for Imbalanced Learning

Related Publications:

1. Chakraborty, T., Chakraborty, A. K. (2020). Hellinger Net : A Hybrid Imbalance Learning Model to Improve Software Defect Prediction. *IEEE Transactions on Reliability*, https://doi.org/10.1109/TR.2020.3020238.

2. Chakraborty, T., Chakraborty, A. K. (2020). Superensemble classifier for improving predictions in imbalanced data sets. *Communications in Statistics - Case Studies and Data Analysis*, 6(2), 123-141.

Summary

Learning from an imbalanced data set presents a tricky problem in which traditional statistical learning algorithms perform poorly. Traditional classifiers usually aim to optimize the overall accuracy without considering the relative distribution of each class. This problem occurs among others in Software defect prediction (SDP) in which one tries to identify defects in the early phases of the software development life cycle that yield a cost-effective and good quality of software products. Various statistical learning models have been employed to predict defects in software modules. But the imbalanced nature of this type of SDP data sets is pivotal for the successful development of a defect prediction model. Imbalanced software data sets contain non-uniform class distributions, with a few instances belonging to a specific class as compared to that of the other class. This chapter proposes a novel hybrid methodology, namely the Hellinger net model, for imbalanced learning to improve defect prediction for software modules. Hellinger net, a tree to network mapped model, is a deep feedforward neural network with a built-in hierarchy, just like decision trees, and uses a skew insensitive distance measure, namely Hellinger distance, in handling class imbalance problems. On the theoretical side, we prove the theoretical consistency of the proposed model. A thorough experiment was conducted over the ten SDP data sets to show the superiority of the proposed method.

4.1 Introduction

Software defect prediction (SDP) is a celebrated research topic in the field of software reliability engineering, which has attracted a lot of attention from both the industrial communities and academicians since the last three decades (Fenton and Neil, 1999; Jing et al., 2016; Pelayo and Dick, 2012; Siers and Islam, 2015). It allows software engineers to allocate limited human resources, time, and other resources to defect-prone modules through early defect prediction. It also plays a vital role in reducing software development costs and maintaining the high quality of software systems. Existing works on SDP can be categorized into three broad areas as follows (Liu et al., 2014):

- Finding the estimates for the number of defects in a software system. Various statistical methods like defect detection profile methods, capture-recapture models have been applied to estimate the number of defects that exist in software based on testing, inspection, code metrics, and process quality data (Briand et al., 2000).
- Capturing software defect associations using data mining technologies (for example, association mining algorithms (Song et al., 2006)).
- Classifying between defect-prone and non-defect-prone categories in software modules. Machine learning approaches, such as classification trees (CT) (Breiman et al., 1984), random forest (RF) (Breiman, 2001), deep feedforward neural networks (DFFNN) (Hornik et al., 1989), and ensemble learning methods (Kuncheva, 2004) are used to predict the defect-proneness of new software modules (Sun et al., 2012; Wang and Yao, 2013; Zheng, 2010).

However, the studies mentioned above have not considered an important characteristic of the SDP problems, viz., the highly imbalanced nature between the defect and non-defect classes of these data sets. In most cases, the collected training defect prediction data sets contain much more non-defective modules (majority) than defective ones (minority). The imbalanced distribution is the critical factor accounting for the poor performance of traditional statistical and machine learning methods, especially on the minority class (Hall et al., 2011; Lessmann et al., 2008). Imbalanced learning is an emerging research domain in machine learning since the last two decades that aims to deal with this kind of problem in a better way (Gong et al., 2019). Several researchers noticed the negative effect of class imbalance on SDP and applied imbalance learning techniques to improve the performance of their defect predictors recently (Gong et al., 2019; Malhotra and Kamal, 2019; Sun et al., 2012; Wang and Yao, 2013). However, there is a scope to enhance class imbalance learning in the context of SDP.

4.1.1 Motivation

Previous studies in SDP developed various software defect prediction models to predict the occurrence of a defect in the unseen (future) version of a software product (Briand et al., 2000; Fenton and Neil, 1999; Jing et al., 2016; Liu et al., 2014). Defects in software modules cause failures and do not allow them to produce desirable results. Therefore an early-warning (detection) system of software defects is essential. In the initial phases of software development life cycles, a set of identified defects could be corrected appropriately (Pelayo and Dick, 2012; Siers and Islam, 2015; Song et al., 2006). Further, if such defects are prevented from propagation to the later stages, then it will be cost-saving for the producer. Thus, an accurate SDP model aids in the development of good quality software products with a lower maintenance cost, which also results in higher customer satisfaction. SDP models highly rely on past data sets to classify software modules as defective or non-defective. Recent work demonstrated that 80% of the defect occurred in very few modules (20%). This indicates that the defective class is there as a minority class in comparison to the non-defective (majority) class that results in an imbalanced scenario in the SDP data sets. It is interesting to note that the minority (defective) class, even being less in number, carries more cost when they are misclassified. Even this results in higher testing costs and escaping crucial errors lead to poor quality of the software. Therefore, it is important to address the imbalanced data problem in SDP arena to reduce future defect percentages, and for successful development of the software.

Traditional classifiers assumes that the classes to be distinguished should have a comparable number of instances. Still, this assumption does not hold in real-world classification problems (Kuncheva, 2004). Although existing machine learning techniques have shown great success in many real-world applications, the problem of learning from imbalanced data is a relatively new challenge that has attracted growing attention from both academia and industry. The imbalanced learning problem is concerned with the performance of learning algorithms in the presence of underrepresented data, and severe class distribution skews (He and Garcia, 2009). Real-world data sets are usually skewed, in that many cases belong to a larger class, and fewer cases belong to a smaller yet usually more exciting class (Fernández et al., 2018b). Here the cost of misclassifying minority examples is much higher due to the seriousness of the problem (Rastgoo et al., 2016). Due to higher weightage is given to the majority class, traditional classifiers tend to misclassify the minority class cases as a member of the majority class (Wang et al., 2019a). This inherent complex characteristics of imbalanced data sets require new understandings, principles, algorithms, and tools to efficiently transform vast amounts of raw data into information and knowledge representation (Zhu et al., 2020). For example, consider a binary classification problem with the class distribution of $90 : 10$. In this case, a straightforward method of guessing all instances to be positive class would achieve an accuracy of 90%.

There are many approaches developed in the literature to handle imbalanced data sets that are applied to SDP problems. One way to deal with the imbalanced data problems is to modify the class distributions in the training data using sampling techniques to the data set. Sampling techniques either oversamples the minority class to match the size of the majority class or undersamples the majority class to match the size of the minority class (Guo and Viktor, 2004). Hybrid sampling approaches not only balance the data but also remove noisy instances lying on the wrong side of the decision boundaries (for example, Synthetic minority oversampling technique (SMOTE) + TL (Tomek links) and SMOTE + ENN (edited nearest neighbor)) (Fernández et al., 2018b; Lemaître et al., 2017). But these approaches have apparent deficiencies. Undersampling majority instances may lose potentially useful information in the data set, whereas oversampling increases the size of the training data set that may increase computational cost (Fernández et al., 2018a). Even cost-sensitive learning methods do not effectively solve the problem since assigning cost values is a difficult proposition. To overcome these deficiencies, "imbalanced data-oriented" algorithms are popularly used in imbalanced pattern classification that can handle class imbalance without modifying the class distributions. Hellinger distance decision tree (HDDT) uses HD as the tree splitting criterion, and it is insensitive towards the skewness of the class distribution (Cieslak and Chawla, 2008; Cieslak et al., 2012). Based on the experimental results, Chawla concluded that unpruned HDDT is recommended to deal with imbalanced problems as a better alternative to sampling approaches (Cieslak et al., 2012). An ensemble version of the HDDT method is Hellinger distance random forest (HDRF) (Su et al., 2015), which can also handle imbalanced data sets. Though HDDTs are robust against class imbalance and mitigate the need for sampling, they are greedy algorithms that sometimes overfit the data set since no pruning techniques are applied (Boonchuay et al., 2017). HDDT also suffers from the drawbacks of sticking to local minima and overfitting the data set (Chaabane et al., 2019) when the tree size is enormous as compared with the number of training data. Motivated by the studies mentioned above, this chapter proposes a novel hybrid approach to deal with class imbalance problems in SDP. We propose a novel methodology, namely Hellinger net, an "imbalanced data-oriented" pattern classifier that can be used as a defect predictor to achieve better accuracy for standard imbalanced SDP data sets. It is shown to be more productive and efficient than other traditional classifiers at predicting defects and improving the overall performance.

4.1.2 Contribution

Software testing takes place at different stages of the software development process. Each module of software is tested separately whenever the module is ready. After that, an integration testing takes place, which verifies whether the software is working correctly or not. The NASA SDP data sets to be considered in this chapter comes from McCabe and Halstead feature extractors of source code

(Boetticher, 2007) on different modules. In particular, the data features various attributes of a module on the different types of complexity measures proposed in the software engineering literature (Briand et al., 2000). To understand the overall defect percentage of a module, one may take the ratio of the number of times the module did not produce expected results to the total number of trials or inputs used to test the module (Fenton and Neil, 1999). Hellinger net is a novel classification tool proposed in this chapter that uses Hellinger distance to find the essential attributes of the modules and further use them as predictors in our proposed Hellinger net model. The proposal is a hybrid learning approach for imbalanced pattern classification problems that uses both HDDT and neural networks. The proposed Hellinger net method can be thought of as a mapping from HDDT to DFFNN which have a built-in hierarchy, and the number of neurons in the hidden layers will be known *apriori*.

In Hellinger net, a pre-trained HDDT can be reformulated as two hidden layered neural networks with similar types of predictive behavior, but with improved accuracy in imbalanced SDP problems (Chakraborty and Chakraborty, 2020a,b). Extensive works are done earlier in the area of hybrid models based on DT and DFFNN; see for example, Bifet et al. (2010); Chen et al. (2006); Kubat (1998); Sethi (1990); Tanno et al. (2019); Zhou and Chen (2002). The motivation behind developing the Hellinger net is to construct an HDDT and then simulate it using neural networks to avoid the deficiencies of HDDT in the imbalanced framework. Hellinger net is composed of three necessary steps: (a) converting an HDDT into rules, (b) constructing two hidden layered DFFNN architecture from the rules, and (c) training the DFFNN using stochastic gradient descent backpropagation (Rumelhart et al., 1985). Soft pruning using DFFNN architecture in the HDDT framework will avoid the need for tree-pruning vis-a-vis the risk of overfitting for imbalanced classification problems. The proposed Hellinger net model has the advantages of significant accuracy and the ability to handle high-dimensional medium-sized defect prediction data sets. The novel hybrid formulation can prevent HDDT from over-fitting and provides a better generalization of a trained neural network than one can learn directly from the training data. We conduct computational experiments to show the performance of the proposed Hellinger net model for SDP using ten imbalanced data sets from the PROMISE repository. Experimental results show that our proposed approach achieves overall higher accuracy than the existing methods for imbalanced learning in SDP. In this chapter, we also prove the theoretical consistency of the Hellinger net using statistical learning theoretic approaches like complexity regularization and covering numbers. We use several results previously developed in the field of DT, FFNN, and hybrid DT-FFNN models (Biau et al., 2019; Brent, 1991; Lugosi and Nobel, 1996; Lugosi and Zeger, 1995) to show the theoretical consistency of the Hellinger net classifier. The approach depends on the choice of the total number of leaves of HDDT and certain restrictions imposed on neural network hyper-parameters to ensure the consistency of the proposed model.

The remainder of this chapter is organized as follows. In Section 4.2, we review related works on class imbalance learning and software defect predictions. We then describe the formulation of the proposed hybrid model for handling class imbalance problems in Section 4.3. In Section 4.4, we discuss asymptotic properties (theoretical consistency) of the proposed Hellinger net model. Section 4.5 reports the results of the experiments on real SDP benchmark data sets. In Section 4.6, we apply the newly developed imbalanced classifier for standard UCI data sets to show the potential application of the methodology in other applied domains. A simulation study is also presented in Section 4.7 to make our results more convincing. Finally, we conclude this chapter with some discussion in Section 4.8.

4.2 Related Works

In this section, we discuss the two main focal points of this chapter. At first, we describe the class imbalance problem and state-of-the-art methods for solving this problem. Further, we review the current research progress in the area of SDP.

4.2.1 Class Imbalance Learning

The basic concept of class imbalance in the binary pattern classification problem is concerned with the situation in which one class of data is highly under-represented as compared to the other class. By convention, the under-represented class is known as minority class, whereas the other class having a larger number of instances is called the majority class. From the SDP point of view, the defect cases are less likely to happen than the non-defect cases. Misclassifying an example from defect class (minority class) is more costly since the failure of finding a defect can degrade the quality of the software by a considerable amount. Finding out a pattern classifier that can provide high accuracy for the minority class instances without affecting the degree of correctness of the majority class remains the concern for the research in imbalanced learning. The problem is challenging since almost all the traditional classifiers give higher weightage to the majority class and result in misclassifying the minority class examples as the majority class (Gong et al., 2019).

Numerous methods have been developed to tackle the curse of data imbalance at both data and algorithm levels. Data-level methods concentrate on manipulating training data to balance the skewed class distributions using various resampling techniques. Algorithm-level methods modify the training mechanism to improve performance accuracy on the minority class examples using techniques like cost-sensitive learning and ensemble learning. The prevalent methodologies of solving class imbalance can be categorized into the following.

1. *Sampling Techniques:* It uses oversampling or undersampling techniques to modify the data class distributions of the imbalanced data sets. Shatnawi (2012) found random oversampling (ROS) more efficient than SMOTE in adding training instances in the defective class of the SDP data sets. López et al. (2013) have shown that using oversampling may result in overfitting as the training data set may have multiple replicate instances. To overcome this limitation, combined approaches of oversampling and undersampling like ROS+TL, SMOTE+TL have been applied to the SDP data sets and found more effective than individual methods but were worse than ensemble learning methods (Gong et al., 2019).

2. *Cost-sensitive Learning:* This method mainly attaches different costs for different class instances. Once the defective instances are misclassified, they are given a higher cost, whereas if the non-defective instances are misclassified, they are attached with a lower cost (Liu et al., 2014). Ryu et al. (2017) proposed a cost-sensitive transfer learning approach for the cross-project defect prediction problem whereas Wan et al. (2017) proposed a cost-sensitive method using dictionary learning to solve the class-imbalance problem in SDP.

3. *Ensemble Learning Techniques:* This method combines multiple classifiers and assign different weights to the component methods for dealing with imbalance classification problems. Ensemble learning can enhance overall performance by combining the strength of individual learners. Wang and Yao (2013) proposed a dynamic version of the AdaBoost.NC for the imbalanced classification problem in SDP. Kernel ensemble learning method based on AdaBoost was also experimentally shown to be effective for 12 projects from NASA SDP data sets (Wang et al., 2016). Laradji et al. (2015) combined the feature selection method with ensemble learning approaches to solve the class imbalance problem.

4.2.2 Software Defect Prediction

Software reliability is often modeled through stochastic processes, in general, Poisson processes that are capable of dealing with rare events like software defects. They are also used to forecast future defects (Xie, 1991; Yamada, 2014). The reliability of software cannot be predicted without some real information about the software system we are interested in. An important and useful type of information is the failure time data collected during the testing stage (Jelinski and Moranda, 1972). Many software reliability models arc developed for the estimation of software reliability based on failure time data. Some well-known models are the exponential model, the logarithmic model, and the s-shaped model (Musa and Okumoto, 1984; Pham, 2006). These models are commonly known as nonhomogeneous Poisson process (NHPP) models and the basic assumption is that the failure process can be described by an NHPP (Xie, 1995). Using different mean value function describing the behavior of the failure process, the reliability can be predicted. Also, Software metrics can be used for the estimation of software reliability. Ruggeri et al. (2008) discussed two models, namely a hidden Markov model and a self-exciting point process with latent variables that can incorporate the case of reliability deterioration due to potential introduction of new bugs to the software during the development phase. Since the introduction of bugs is an unobservable process, latent variables are introduced to incorporate this characteristic into the models.

SDP problems can also be formulated as binary classification problems in which software modules are either classified as defect-prone or non-defect-prone based on a set of software metrics. There are various types of software metrics available from previously developed systems by standard tools, as discussed in Nagappan and Ball (2005a). The first of its kind of software metric, popularly known as the CK metric, was introduced by Chidamber and Kemerer (1994). Various other software metrics such as code metrics (Basili et al., 1996), process metric (Nagappan and Ball, 2005b), and previous defaults (Kim et al., 2007) were subsequently introduced. In software reliability engineering, a set of static code attributes is extracted from previous software releases with the log files of defects. These values are used to build classifiers to predict defective modules for the next phase of release. This helps in locating the parts of the software that are most likely to contain defects. Thus, a 'good' defect predictor system can be used as a guide to the software engineers to focus on the testing of the defect-prone parts of the software systems. PROMISE repository (Boetticher, 2007), an open-source of defect prediction data sets from real-world projects, made rapid growth in the field of SDP by making data sets available for public use. SDP researchers have developed several defect predictors to improve the quality of software and reduce the cost of delivery of those software systems.

A variety of statistical and machine learning tools have been applied to solve SDP problems, such as DT (Breiman et al., 1984; Khoshgoftaar and Seliya, 2002), RF (Breiman, 2001; Guo et al., 2004), DFFNN (Hornik et al., 1989; Zheng, 2010), support vector machines (Gray et al., 2009), Naive Bayes (NB) (Rish, 2001; Turhan and Bener, 2009) and artificial immune systems (Catal and Diri, 2009). Some researcher have considered using ensemble learning (Guo et al., 2004; Sun et al., 2012), analogy-based methods (Khoshgoftaar and Seliya, 2003), genetic programming (Khoshgoftaar and Liu, 2007), kernel-based methods (Gray et al., 2009; Yang and Li, 2007), cost-sensitive learning (Zheng, 2010), and transfer learning (Turhan, 2012) to build SDP models. It was shown that no single method is found to be the 'best' for all the SDP data sets, but overall RF and cost-sensitive learning approaches appear to be 'good' choices for the majority of the data sets (Wang and Yao, 2013).

However, many of the previous studies had overlooked the class imbalance scenario in SDP data characteristics. Although, many researchers have considered the imbalanced data distribution between defect and non-defect classes and used ensemble algorithms and other methods of dealing with data imbalance (Gong et al., 2019). Also, some researches showed the usefulness of resampling based on tree-based learners (Pelayo and Dick, 2012; Wang and Yao, 2013). Ensemble and cost-sensitive learning approaches were also studied and found beneficial if a proper cost ratio is set (Laradji et al., 2015; Zheng, 2010). But these approaches have obvious deficiencies like modifying the actual data sets using sampling techniques or choice of appropriate weights in creating ensembles. To overcome these drawbacks, a few "imbalanced data-oriented" classifiers were introduced in the recent literature (Aler et al., 2020; Cieslak and Chawla, 2008; Su et al., 2015). These classifiers (e.g., HDDT and HDRF) can handle class imbalance without modification to the class distributions, but there is scope for improvements.

4.3 Formulation of the Hellinger Net Model

The idea of the Hellinger net algorithm is to map an HDDT into a DFFNN model. The additional training in the HDDT framework using stochastic gradient descent backpropagation can necessarily improve the performance of HDDT for imbalanced classification problems that arise in SDP. This will prevent the HDDT from overfitting, and since DFFNN is used in the hybridization, there is no need for further tree-pruning in the proposed Hellinger net model. In the following section, we first discuss the failure of decision trees in data imbalance frameworks and then describe the HDDT, which will be used in the proposed Hellinger net model. Further, we describe the proposed model in mathematical details.

4.3.1 Main Insight: Failure of Decision Trees

We first investigate the effect of class imbalance on the decision trees, following Chakraborty and Chakraborty (2020b). It is essential to see how decision boundaries created by DT get affected by imbalance ratio (the ratio between the number of minority and majority examples).

Table 4.1: An example of notions of classification rules

class and attribute	$X^{\geq}$	$X^{<}$	sum of instances
Y^{+}	a	b	$a+b$
Y^{-}	c	d	$c+d$
sum of attributes	$a+c$	$b+d$	n

Let X be an attribute and Y be the response class. Here Y^{+} denotes majority class, Y^{-} denotes minority class, and n is the total number of instances.

Also, let $X^{\geq} \longrightarrow Y^{+}$ and $X^{<} \longrightarrow Y^{-}$ be two rules generated by DT. Table 4.1 shows the number of instances based on the rules created using DT. In the case of imbalanced data set, the majority class is always much larger than the size of the minority class, and thus, we will always have $a+b >> c+d$. The generation of rules based on confidence in DT is biased towards the majority class. Various measures, like information gain (IG), Gini index (GI), and misclassification impurity (MI) expressed as a function of confidence, are used to decide which variable to split in the important feature selection stage (Flach, 2003). From Table 4.1, we can define

$$\text{Confidence}(X^{\geq} \longrightarrow Y^{+}) = \frac{a}{a+c}.$$

Let us consider a binary classification problem with the label set $\Omega = \{\omega_1, \omega_2\}$ and let $P(j/t)$ be the probability for class ω_j at a certain node t of the classification tree, where, $j = 1, 2$ for binary classification problems. These probabilities can be estimated as the proportion of points from the respective class within the data set that reached the node t. Using Table 4.1, we compute the following:

$$P(Y^{+}/X^{\geq}) = \frac{a}{a+c} = \text{Confidence}(X^{\geq} \longrightarrow Y^{+}) \tag{4.1}$$

For an imbalanced data set, Y^{+} will occur more frequently with $X^{\geq}$ & $X^{<}$ than to Y^{-}. So the concept of confidence is a fatal error in an imbalanced classification problem where minority class is of more interest and data is biased towards the majority class. In binary classification, information gain for splitting a node t is defined as:

$$\text{IG} = \text{Entropy}(t) - \sum_{i=1,2} \frac{n_i}{n} \text{Entropy}(i) \tag{4.2}$$

where i represents one of the sub-nodes after splitting (assuming we have two sub-nodes only), n_i is the number of instances in sub-node i and n is the total number of instances. Entropy at node t is defined as:

$$\text{Entropy}(t) = -\sum_{j=1,2} P(j/t) log\big(P(j/t)\big) \tag{4.3}$$

The objective of classification using DT is to maximize IG which reduces to (assuming the training set is fixed and so the first term in equation (4.2) is fixed as well):

$$\text{Maximize}\left\{ -\sum_{i=1,2} \frac{n_i}{n} \text{Entropy}(i) \right\} \tag{4.4}$$

Using Table 4.1 and equation (4.3); the maximization problem in equation (4.4) reduces to:

$$\text{Maximize}\Bigg\{\frac{n_1}{n}\Big[P(Y^+/X^{\geq})log\left(P(Y^+/X^{\geq})\right)+P(Y^-/X^{\geq})log\left(P(Y^-/X^{\geq})\right)\Big]$$
$$+\frac{n_2}{n}\Big[P(Y^+/X^{<})log\left(P(Y^+/X^{<})\right)+P(Y^-/X^{<})log\left(P(Y^-/X^{<})\right)\Big]\Bigg\} \quad (4.5)$$

The task of selecting the 'best' set of features for node i is carried out by picking up the feature with maximum IG. As $P(Y^+/X^{\geq}) >> P(Y^-/X^{\geq})$, we face a problem while maximizing (4.5). We can conclude from the above discussion that impurity-based measures for tree splitting in DT are biased towards majority class.

4.3.2 Hellinger Distance Decision Tree (HDDT)

The Hellinger distance (HD), a symmetric and non-negative measure of distributional divergence, is related to the Bhattacharyya's distance and the Kullback-Leibler's divergence (Rao, 1995). Cieslak and Chawla (2008) proposed the use of HD as a decision tree split criterion for imbalanced data classification. They were able to utilize the distance by considering two distributions P and Q to be the normalized frequencies of feature values in a binary classification scenario. Cieslak et al. (2012) used the notion of 'affinity' between P and Q as a decision tree split criterion. The aim is to split tree nodes on those features with minimal affinity, i.e., maximal HD. This approach is appealing since it enables the splitting of features based on how well they discriminate between the examples seen so far in the stream, rather than on the feature which describes the largest possible number of instances seen so far (as with information gain in case of Breiman's CART (Breiman et al., 1984)). Intuitively, HD is skew insensitive, since an abundance of examples of one class will only serve to make its sample distribution more representative of its real distribution. If a feature is a good class discriminator, then irrespective of the balance, it will remain as such. We formally define HD as follows:

Definition 11 *Let* (Θ, λ) *denote a measurable space and assume that* P *and* Q *be two continuous distributions for the parameter* λ *having the densities* p *and* q *in a continuous space* Ω*, respectively (Akash et al., 2019). We define HD as follows:*

$$d_H(P,Q)=\sqrt{\int_{\Omega}(\sqrt{p}-\sqrt{q})^2d\lambda}=\sqrt{2\left(1-\int_{\Omega}\sqrt{pq}d\lambda\right)} \quad (4.6)$$

where $\int_{\Omega}\sqrt{pq}d\lambda$ *is the Hellinger integral.*

It is noted that HD doesn't depend on the choice of the parameter λ. HD carries the following important properties:

1. $d_H(P, Q)$ is in $[0, \sqrt{2}]$;
2. HD is symmetric and non-negative namely $d_H(P, Q) = d_H(Q, P) \geq 0$;
3. The bigger the value of HD is, the better discrimination of the feature is.

HDDT uses HD as the tree-splitting criterion and builds the tree based on the idea of Breiman's CART (Breiman et al., 1984) as described in Cieslak et al. (2012). The core of their approach is a tree with k_n leaf regions defined by a partition of the space based on the n data points. When constructing the tree, the so-called CART-split criterion is applied recursively. This criterion determines which input direction should be used to split and where the cut should be made. HDDT uses the same idea of CART (other than the choice of the split criterion) to make a hierarchical axis-parallel split of the feature input spaces. Each tree node corresponds to one of the segmentation subsets in the feature space. We consider only a binary tree where a node has exactly two child nodes or zero child nodes (Cieslak et al., 2012).

Remark 7 *HD as a distance function satisfies the metric properties and it is also comparable with statistical distance (Chakraborty and Chakraborty, 2020b). Thus, HD is used as a popular choice of splitting criteria in decision trees (Akash et al., 2019) and an ensemble of trees (Aler et al., 2020) for improved performance in imbalanced problems. But HDDT suffers from the drawbacks of sticking to local minima and overfitting for the small or medium sample-sized data sets (Chaabane et al., 2019). Therefore, we propose a Hellinger net model that adds a soft pruning in the HDDT framework using neural networks based on the idea of Biau et al. (2019); Frosst and Hinton (2017); Utgoff (1989).*

4.3.3 Proposed Hellinger Net Model

Suppose we have a training sample $D_n = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$ with n observations on p input features $\underline{X} \in C^p = [0, 1]^p$ and we try to predict its corresponding class label $\underline{Y} \in \{0, 1\}$. HDDT consists of split nodes (for example, $x^{(i)} \geq \alpha$ for some $i \in \{1, 2, ..., p\}$ and some $\alpha \in C$) and leaf nodes. The feature space C^p is partitioned into axis-parallel hyper-rectangles. The tree splitting criteria that is used to create the HDDT after slightly modifying the definition of HD in (4.6) is as follows (Cieslak and Chawla, 2008; Cieslak et al., 2012):

$$HD = d_H(X_+, X_-) = \sqrt{\sum_{j=1}^{p} \left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}} \right)^2}, \tag{4.7}$$

where $|X_+|$ indicates the number of examples that belong to the majority class in training set and $|X_{+j}|$ is the subset of training set with the majority class and the value j for the feature X. Similar explanation can be written for $|X_-|$ and $|X_{-j}|$ but for the minority class. Here p is the number of partitions of the feature space $\underline{X}$. The bigger the value of HD, the better is the discrimination between the features. A feature is selected if it carries the minimal affinity between the classes. Since equation (4.7) is not influenced by prior probability, it is insensitive to the class distribution. Here, instead of using class probability, normalized frequencies aggregated over all the p partitions across classes are used. HDDT is strongly considered to be skew insensitive because of not using prior probability in the distance calculation. However, the split criterion defined in equation (4.7) only works on the binary classification problem. Even various modifications of HDDT were found useful for handling imbalanced data classification problems (Akash et al., 2019; Su et al., 2015).

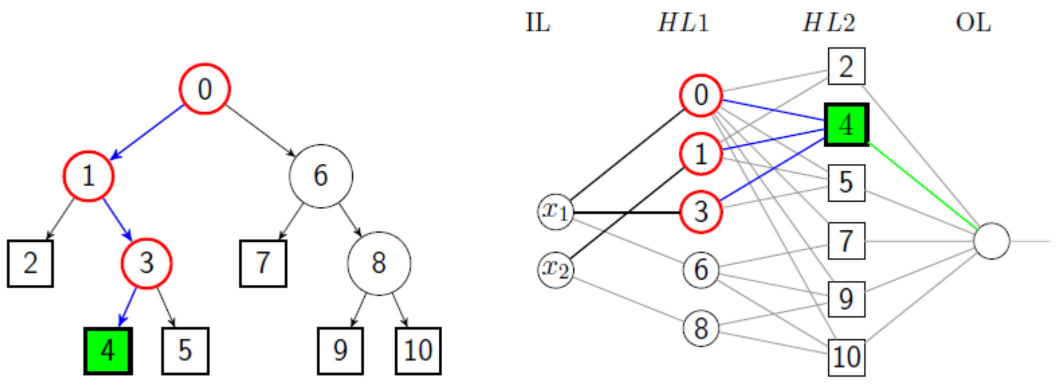


Figure 4.1: An example of Hellinger net model: An HDDT (left) and its corresponding DFFNN structure (right). The circle nodes in the tree belong to split nodes and square nodes to leaf nodes. The path to the green shaded leaf (4) consists of all red nodes (0, 1, 3). Numbers in neurons correspond to numbers in tree model nodes. The highlighted connections in the network are those relevant for the activity of the green neuron and its output value.

During prediction, the input is first passed into the tree root node and then iteratively transmitted to the leaf node. This procedure is repeated until a leaf node is reached. If a leaf node represents region S, then the tree estimate takes the following form:

$$t_n(x) = \frac{1}{N_n(S)} \left(\sum_{\{x_i \in S\}} Y_i \right), \tag{4.8}$$

where $N_n(S)$ is the number of observations in cell S. We assume, by convention, $0/0 = 0$. Suppose we are given with an HDDT estimate t_n (the construction is necessarily data-dependent and reminiscent to imbalanced classification framework), that can take constant values on each of the k (≥ 2) terminal nodes. Since the value of k depends on n, using the notation k_n is also appropriate. Finally, these estimates are reinterpreted as two hidden layered DFFNN model based on the following rules given below. This additional training to the HDDT using stochastic gradient descent backpropagation can necessarily improve the performance of HDDT for imbalanced classification problems that arise in SDP. This will prevent the HDDT from overfitting, and since DFFNN is used in the hybridization, there is no need for further tree-pruning in the proposed model.

Designing the first hidden layer: Define $HL1 = \{H_1, H_2, ..., H_{k-1}\}$ be the collection of all hyperplanes participating in the construction of t_n. We note that each $H_{k'} \in HL1$ is of the form $H_{k'} = \{x \in C^p : h_{k'}(x) \geq 0\}$, where $h_{k'}(x) = x^{(i_{k'})} - \alpha_{i_{k'}}$ for some (eventually data-dependent) $i_{k'} \in \{1, 2, \ldots, p\}$ and $\alpha_{i_{k'}} \in C$. For reaching out in the leaf of any query point x, we try to find, for each hyperplane $H_{k'}$, the side on which x falls (we accordingly assign $+1$ if it falls in the right side and -1 for left). The input layer supplies the features to $HL1$ of neurons that corresponds to $k-1$ perceptrons, with the threshold activation function $\tau(h_{k'}(x)) = \tau(x^{(i_{k'})} - \alpha_{i_{k'}})$, where $\tau(u) = 2I_{u\geq 0} - 1$ and I denotes an indicator function. Thus, for each split in the HDDT, there exists a neuron in $HL1$ whose activity encodes relative position of an input value x with respect to the respective split. The outputs of $HL1$ are ± 1 vector $(\tau(h_1(x)), \ldots, \tau(h_{k-1}(x)))$, that will delineate all the decisions of the inner tree nodes (it also includes the nodes off the tree path of x). It is worth noting that $\tau(h_{k'}(x))$ is $+1$ if x is on one side of the hyperplane $H_{k'}$, and -1 if x is on the other side of $H_{k'}$ (where, by convention, $+1$ if $x \in H_{k'}$). In Hellinger net, we use sigmoidal activation function instead of the relay-type activation function $\tau(u)$ with a hyperbolic tangent activation function $\sigma(u) = \tanh(u)$ which has a chosen range from -1 to 1. More precisely, Hellinger net uses $\sigma_1(u) = \sigma(\beta_1 u)$ at every neuron of the first hidden layer for better generalization, where β_1 is a positive hyper-parameter that determines the contrast of the hyperbolic tangent activation function.

Designing the second hidden layer: $HL1$ outputs a $(k-1)$-dimensional vector of ± 1-bits that encodes the precise position of x in the leaves of the tree. The second hidden layer has k neurons, one for each terminal nodes, and assigns a leaf cell to the value x. Define $HL2 = \{L_1, \ldots, L_k\}$ as the collection of all the terminal nodes of the tree, and let $L(x)$ be the leaf containing x. A connection from a unit k' of $HL1$ to another unit k'' of $HL2$ is set if and only if the hyperplane $H_{k'}$ is necessitated in the sequence of splits making the path from the root to the leaf nodes $L_{k''}$. The aforesaid connection will have weight $+1$ if the split by $H_{k'}$ is from a right child node in that path, and -1 otherwise. We will have $(u_1(x), \ldots, u_{k-1}(x))$ as the vector of ± 1 bits obtained at the output of $HL1$, then the output $v_{k''}(x) \in \{-1, 1\}$ of neuron k'' is $\tau(\sum_{k' \to k''} b_{k',k''} u_{k'}(x) + b^0_{k''})$.

By the notation $k' \to k''$, we mean k' is connected to k'' and $b_{k',k''} = \pm 1$ is the corresponding weight. The offset $b^0_{k''}$ is set to

$$b^0_{k''} = -l(k'') + \frac{1}{2}, \tag{4.9}$$

where $l(k'')$ is calculated as the length of the path from the root to $L_{k''}$. The rationale behind the choice of (4.9) is that there are exactly $l(k'')$ connections starting from the HL_1 and pointing to k'', thus we have

$$\begin{cases} \sum_{k' \to k''} b_{k',k''} u_{k'}(x) - l(k'') + \frac{1}{2} = \frac{1}{2} & \text{if } x \in L_{k''} \\ \sum_{k' \to k''} b_{k',k''} u_{k'}(x) - l(k'') + \frac{1}{2} \leq -\frac{1}{2} & \text{if } x \notin L_{k''} \end{cases} \tag{4.10}$$

Using (4.10), the value of $v_{k''}(x) = 1$ if and only if the terminal cell of the value x is $L_{k''}$. To summarize, $HL2$ outputs a vector $(v_1(x), ..., v_k(x))$ of ± 1 bits. In Hellinger net, we use $\sigma_2(u) = \sigma(\beta_2 u) = \tanh(\beta_2 u)$ at every neuron of the second hidden layer instead of relay-type threshold activation function for better generalization, where β_2 is a positive hyper-parameter that determines the contrast of the activation function.

Designing the output layer: The output layer calculates the average $\bar{Y}_{k''}$ of the Y_i corresponding to X_i falling in $L_{k''}$ when $v_{k''}(x) = +1$. Equivalently, we consider the following form of tree estimates:

$$t_n(x) = \sum_{k''=1}^{k} w_{k''} v_{k''}(x) + b_{\text{out}}, \tag{4.11}$$

where $w_{k''} = \bar{Y}_{k''}/2$ for all $k'' \in \{1, \dots, k\}$, and $b_{\text{out}} = \frac{1}{2} \sum_{k''=1}^{k} \bar{Y}_{k''}$.

Then the classification rule to be used by Hellinger net model can be written as

$$g_n(x) = \begin{cases} 0, & \text{if} \quad t_n(x) \leq \frac{1}{2} \\ 1, & \text{otherwise,} \end{cases} \tag{4.12}$$

where $g_n(x)$ is the predicted class of the proposed Hellinger net model.

We considered a tree estimate t_n (obtained from HDDT) and seen it as a neural network estimate. The architecture of this network (conditional on D_n) is fixed, and so are the weights and offsets of the three layers. The original idea is to keep the network structure intact and let the parameters vary in a subsequent network training procedure with backpropagation training. Once the connections between the neurons have been designed by the tree-to-network mapping, we could then learn even better network parameters by minimizing some empirical mean squared error for this network over the sample D_n. This additional training can potentially improve the predictions of the base HDDT model. We can achieve better generalization in the neural network training stage by incorporating soft nonlinearities in the neurons by choosing smooth activation functions. So, to increase the generalization capabilities of the proposed tree-to-network mapped model as described above, we replace the original relay-type activation function $\tau(u) = 2I_{u\geq 0} - 1$ in its hidden layers with a smooth hyperbolic tangent activation function,

$$\sigma(u) = \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}} = \frac{e^{2u} - 1}{e^{2u} + 1},$$

which has a range of -1 to 1. More precisely, we use $\sigma_1(u) = \sigma(\beta_1 u)$ at every neuron of the first hidden layer and $\sigma_2(u) = \sigma(\beta_2 u)$ at every neuron of the second one. Here, β_1 and β_2 are positive hyperparameters that determine the contrast of the hyperbolic tangent activation: the larger β_1 and β_2 are, the sharper is the transition from -1 to 1. Obviously, when β_1 and β_2 approach infinity, the continuous functions σ_1 and σ_2 converge to the threshold function. Besides eventually providing better generalization, the hyperbolic tangent activation functions favor smoother decision boundaries and permit a relaxation of crisp tree node membership (Brent, 1991). Using a smooth continuous activation function in the architecture makes the network loss function differentiable with respect to the parameters everywhere, and gradients can be backpropagated to train the network.

Remark 8 *In the Hellinger net, training of a neural network with sparse connectivity retains some degree of interpretability of its internal representation. It may be seen as a relaxation of original tree structures. The initial tree-type setting provides a strong inductive bias, compared to a random initialization, which contains valuable information and mimics the function of an HDDT-type tree (Cieslak et al., 2012) before the backpropagation training. Compared to a random initialization, this gives the network an effective warm start. Also, the additional training using backpropagation with hyperbolic tangent activation function will potentially improve the predictions of the DT as well as the risk of overfitting. The use of hyperbolic tangent activation functions instead of threshold activation function provide better generalization, smooth decision boundaries, and fast implementation. They also support the differentiability of the empirical loss function with respect to its parameters due to the continuous property of the tangent activation function. Thus the gradients can be backpropagated to train the Hellinger net model.*

4.3.4 Hellinger Net Algorithm

We present an informal workflow of the Hellinger net algorithm below (also see Figure 4.1).

- Build HDDT with $(k_n - 1)$ split nodes and k_n leaf nodes. HDDT is mapped into a two hidden layered DFFNN model having $(k_n - 1)$ and k_n hidden neurons in first hidden layer ($HL1$) and second hidden layer ($HL2$), respectively.

- The first hidden layer of the model is called the partitioning layer, which partitions the input feature spaces into different regions. The partitioning layer corresponds to the internal nodes of the HDDT. In $HL1$, the neurons compute all the tree split decisions and indicate the split directions for the inputs.

- Further, $HL1$ passes the information to $HL2$. The second hidden layer implements the ANDing of partitioned regions. Also, an ORing in the final layer of the two-layered DFFNN model is implemented. The neurons in the second hidden layer represent the terminal nodes of the DT.

- Train the tree-structured neural network using a stochastic gradient descent backpropagation algorithm. The additional training using backpropagation potentially improves the predictions of the HDDT and can deny tree pruning steps vis-a-vis the risk of overfitting. Hellinger net gives weight to the nodes according to their significance, as determined by the stochastic gradient backpropagation algorithm.

- The final layer is the output class label to be decided by the neural network training. In the Hellinger net, the neural network follows the built-in hierarchy of the originating tree since connections do not exist between all pairs of neurons in any two adjacent layers.

Remark 9 *Fitting a fully connected DFFNN algorithm with 2 hidden layers having $k_n - 1$ and k_n neurons in the first and second hidden layer, respectively, requires optimizing a total of $(p+1)(k_n-1) + k_n^2 + k_n + 1$ parameters, which is $O(pk_n + k_n^2)$ (Brent, 1991). If one assumes that the tree generated by HDDT algorithm is roughly balanced, then the average depth of the tree is $O(\log k_n)$. This gives, on an average, $2(k_n-1) + O(k_n \log k_n) + k_n + 1$ parameters to be fitted, which is $O(k_n \log k_n)$ for Hellinger net model. For large k_n, this quantity can be much smaller than $O(pk_n + k_n^2)$ and this gives a major computational advantage in high-dimensional settings.*

4.4 Asymptotic Results

To prove the theoretical consistency of Hellinger net, we consider the tree in the ensemble model and denote it by $G_1 \equiv G_1(D_n)$ as a bipartite graph which models the connections between the input vectors $x = (x^{(1)}, ..., x^{(p)})$ and $k_n - 1$ hidden neurons of $HL1$. Also, let $G_2 \equiv G_2(D_n)$ represent the connections between $HL1$ and k_n hidden neurons of $HL2$. Define $M(G_1)$ be the set of $p \times (k_n - 1)$ matrices $A = (a_{ij})$ such that $a_{ij} = 0$ if $(i,j) \notin G_1$. Also, define $M(G_2)$ to be the $(k_n - 1) \times k_n$ matrices $B = (b_{ij})$ such that $b_{ij} = 0$ if $(i,j) \notin G_2$. The model specifying parameters in the $HL1$ (of the size $k_n - 1$) are put in a matrix A of $M(G_1)$ identified by the weights over the edges of G_1 and a column vector of biases b_1. Similarly, for HL_2 (of the size k_n), a matrix B of $M(G_2)$ of weights over G_2 and a column vector b_2 of offsets is formed. Let us take the output weights and offset to be $W_{\text{out}} = (w_1, \ldots, w_{k_n})^\top \in R^{k_n}$ and $b_{\text{out}} \in R$, respectively. And the parameters that specify the Hellinger net model are constituted by a 'vector' as shown below:

$$\lambda = (A, b_1, B, b_2, W_{\text{out}}, b_{\text{out}}) \in M(G_1) \times R^{k_n - 1} \times M(G_2) \times R^{k_n} \times R^{k_n} \times R.$$

We further assume that there will always exist a positive constant c_1 such that

$$\|B\|_\infty + \|b_2\|_\infty + \|W_{\text{out}}\|_1 + |b_{\text{out}}| \leq c_1 k_n, \tag{4.13}$$

where $\|\cdot\|_\infty$ is the supremum norm of a matrix and $\|\cdot\|_1$ is the L_1-norm of a vector. The rationale behind this assumption (4.13) is that the weights and offsets are taken by the computation units of HL_2 and the output layer. We note that the condition is satisfied by the tree estimates as Y takes values in $\{0, 1\}$.

Thus, let $\Lambda_{n,k_n} = \big\{\lambda = (A, b_1, B, b_2, W_{\text{out}}, b_{\text{out}}) : \text{(4.13) is satisfied}\big\}$, then DFFNN implements the functions of this particular form

$$f_\lambda(x) = W_{\text{out}}^\top \sigma_2\Big(B^\top \sigma_1(A^\top x + b_1) + b_2\Big) + b_{\text{out}}, \quad x \in C^p,$$

where $\lambda \in \Lambda_{n,k_n}$. We aim to tune the parameters λ using the training data D_n such that the function realized by the acquired neural net becomes a 'good' estimate that can minimize the empirical error. Let $F_{n,k_n} = \big\{f_\lambda : \lambda \in \Lambda_{n,k_n}\big\}$, where F_{n,k_n} be the class of neural networks.

We define the L_1 error of a function $\psi : C^p \to \{0, 1\}$ by $J(\psi) = E\{|\psi(X) - Y|\}$. To show the strong consistency of the classification rule

$$g_n(x) = \begin{cases} 0, & \text{if } \psi_n(x) \leq \frac{1}{2}. \\ 1, & \text{otherwise.} \end{cases}$$

We need to show $J(\psi_n) - J^* \to 0$ in probability, where $J(\psi_n) = E\{|\psi_n(X) - Y||D_n\}$ and $J^* = \inf_{\psi_n} J(\psi_n)$. Write

$$J(\psi_n) - J^* = \left(J(\psi_n) - \inf_{\psi \in F_{n,k_n}} J(\psi)\right) + \left(\inf_{\psi \in F_{n,k_n}} J(\psi) - J^*\right) \tag{4.14}$$

where, $(J(\psi_n) - \inf_{\psi \in F_{n,k_n}} J(\psi))$ is called estimation error and $(\inf_{\psi \in F_{n,k_n}} J(\psi) - J^*)$ is called approximation error (Devroye et al., 1996). The main result for consistency of the Hellinger net classifier is formally presented in the following theorem.

Theorem 13 *Assume X is uniformly distributed in $[0,1]^p$, $Y = \{0,1\}$, and $\psi \in F_{n,k_n}$. As $n \to \infty$ and for any $k_n, \beta_1, \beta_2 \to \infty$ if the following conditions are satisfied:*

$$
\begin{aligned}
&(A1) \quad \frac{k_n^4 log(\beta_2 k_n^4)}{n} \to 0,\\
&(A2) \quad \textit{there exists} \quad \delta > 0 \quad \textit{such that} \quad \frac{k_n^2}{n^{1-\delta}} \to 0,\\
&(A3) \quad \frac{k_n^2}{e^{2\beta_2}} \to 0, \quad \textit{and}\\
&(A4) \quad \frac{k_n^3 \beta_2}{\beta_1} \to 0,
\end{aligned}
$$

then the classification rule $g_n(x)$ is strongly consistent.

Remark 10 *Theorem 13 states that with certain restrictions imposed on the number k_n of terminal nodes and the parameters β_1, β_2 being properly regulated as functions of n, the empirical L_1 risk-minimization provides strong consistency of the Hellinger net classifier. It should be noted that the larger the value of k_n, β_1 and β_2, the better is the model in practice. These results are in terms of theoretical feasibility. In practice, the learning model can be difficult due to optimization difficulties.*

Proof To prove Theorem 13, we need to show that under the conditions (A1)-(A4), the estimation error and the approximation error tend to zero. This will ensure the strong consistency of the proposed Hellinger net model. To handle estimation error, we write

$$
\begin{aligned}
&J(\psi_n) - \inf_{\psi \in F_{n,k_n}} J(\psi)\\
&\leq 2 \sup_{\psi \in F_{n,k_n}} |J(\psi) - J_n(\psi)| \\
&= 2 \sup_{\psi \in F_{n,k_n}} \left| E\{|\psi(X) - Y|\} - \frac{1}{n}\sum_{i=1}^{n} |\psi(X_i) - Y_i| \right|
\end{aligned}
\tag{4.15}
$$

We can assume that for each $\psi \in F_{n,k_n}$, satisfies $\|\psi\|_\infty \leq c_1 k_n$. Define the class M_{n,k_n} of functions on $C^p \times \{0,1\}$ by

$$
M_{n,k_n} = \left\{ m(x,y) = \big|y - \psi(x)\big| : (x,y) \in C^p \times \{0,1\} \right\}
$$

Then the upper bound of equation (4.15) becomes

$$
2 \sup_{m \in M_{n,k_n}} \left| E\{m(X,Y)\} - \frac{1}{n}\sum_{i=1}^{n} m(X_i, Y_i) \right|.
$$

Applying uniform law of large numbers we can observe that the function $m \in M_{n,k_n}$ will satisfy the following: $0 \leq m(x,y) \leq 2c_1k_n$. For large n and $c_1k_n \geq 1$, upon using Theorem 4 (Pollard, 1984), we obtain the following:

$$P\left\{ \sup_{m \in M_{n,k_n}} \left| E\{m(X,Y)\} - \frac{1}{n}\sum_{i=1}^{n} m(x_i, Y_i) \right| > \epsilon \right\} \tag{4.16}$$

$$\leq 8E\{N(\epsilon/8, M_{n,k_n}(D_n))\} \exp\{-n\epsilon^2/(512c_1^2k_n^2)\},$$

where $N(\epsilon, M_{n,k_n}(D_n))$ denotes the l_1-covering number of the random set

$$M_{n,k_n}(D_n) = \{(m(X_1,Y_1), ..., (m(X_n,Y_n) : m \in M_{n,k_n}\}.$$

Observe that for $m_1, m_2 \in M_{n,k_n}$ with $m_1(x,y) = |\psi_1(x) - y|$ and $m_2(x,y) = |\psi_2(x) - y|$, for any probability measure υ on $C^p \times \{0,1\}$,

$$\int \big|m_1(x,y) - m_2(x,y)\big|\upsilon(d(x,y)) \leq \int \big|\psi_1(x) - \psi_2(x)\big|\mu(dx),$$

where μ is the marginal of υ on C^p. It follows from above that $N(\epsilon, M_{n,k_n}(D_n)) \leq N(\epsilon, F_{n,k_n}(X^n))$, where $X^n = (X_1, X_2, ..., X_n)$.

Let the output of neurons of $HL1$ belong to the class

$$G_1 = \{\sigma_1(a^\top x + a_0) : a \in R^p, a_0 \in R\},$$

and for $0 < \varepsilon < 1/4$,

$$N(\varepsilon, G_1, X^n) \leq 2\left(\frac{4e}{\varepsilon}\right)^{2p+4}.$$

Next, letting $G_2 = \{bg : g \in G_1, b \in [-c_1k_n, c_1k_n]\}$ we get

$$\begin{aligned} N(\varepsilon, G_2, X^n) &\leq \frac{4c_1k_n}{\varepsilon} N\Big(\frac{\varepsilon}{2c_1k_n}, G_1, X^n\Big) \\ &\leq \Big(\frac{8ec_1k_n}{\varepsilon}\Big)^{2p+5}. \end{aligned}$$

The second unit will compute the functions of the collection

$$G_3 = \left\{ \sigma_2\Big(\sum_{i=1}^{k_n - 1} g_i + b_0\Big) : g_i \in G_2, b_0 \in [-c_1k_n, c_1k_n] \right\}.$$

Note that σ_2 satisfies the Lipschitz property $|\sigma_2(u) - \sigma_2(v)| \leq \beta_2|u - v|$ for all

$(u, v) \in R^2$. Therefore,

$$N(\varepsilon, G_3, X^n) \leq \frac{2c_1\beta_2 k_n^2}{\varepsilon} N\Big(\frac{\varepsilon}{2\beta_2 k_n}, G_2, X^n\Big)^{k_n-1}$$
$$\leq \Big(\frac{16ec_1\beta_2 k_n^2}{\varepsilon}\Big)^{(2p+5)k_n}.$$

Also, let $G_4 = \{wg : g \in G_3, w \in [-c_1 k_n, c_1 k_n]\}$.

Now, without loss of generality $c_1, \beta_2 \geq 1$, and we get

$$N(\varepsilon, G_4, X^n) \leq \frac{4c_1 k_n}{\varepsilon} N\Big(\frac{\varepsilon}{2c_1 k_n}, G_3, X^n\Big)$$
$$\leq \Big(\frac{32ec_1^2\beta_2 k_n^3}{\varepsilon}\Big)^{(2p+5)k_n+1}.$$

Finally, we can write

$$F_{n,k_n} = \Bigg\{\sum_{i=1}^{k_n} g_i + b_{\text{out}} : g_i \in G_4, b_{\text{out}} \in [-c_1 k_n, -c_1 k_n]\Bigg\}.$$

We conclude

$$N(\varepsilon, F_{n,k_n}, X^n) \leq \frac{2c_1 k_n(k_n+1)}{\varepsilon}\Big[N\Big(\frac{\varepsilon}{k_n+1}, G_4, X^n\Big)\Big]^{k_n}$$
$$\leq \Big(\frac{32ec_1^2\beta_2(k_n+1)^4}{\varepsilon}\Big)^{(2p+5)k_n^2+k_n+1}. \tag{4.17}$$

Combining (4.16)-(4.17) together, we obtain

$$P\Big\{\sup_{f\in F_n}\Big|\frac{1}{n}\sum_{i=1}^{n}\big|Y_i - f(X_i)\big| - E\big|Y - f(X)\big|\Big| > \varepsilon\Big\}$$

$$\leq 8\Big(\frac{256ec_1^2\beta_2(k_n+1)^4}{\varepsilon}\Big)^{(2p+5)k_n^2+k_n+1}\exp\Big(-\frac{n\varepsilon^2}{512c_1^2k_n^2}\Big).$$

Now if the conditions (A1) and (A2) of Theorem 13 holds, then

$$\sum_{n=1}^{\infty} P\Big\{ \sup_{f\in F_{n,k_n}} \Big| \frac{1}{n}\sum_{i=1}^{n} \big|Y_i - f(X_i)\big| - E\big|Y - f(X)\big| \Big| > \varepsilon \Big\}$$

$$< \sum_{n=1}^{\infty} 8\exp\left[\Big((2p+5)k_n^2 + k_n + 1\Big) \times \log\left(\frac{256ec_1^2\beta_2(k_n+1)^4}{\epsilon}\right) - \frac{n\epsilon^2}{512c_1^2k_n^2}\right]$$

$$= \sum_{n=1}^{\infty} 8\left[- n^{\delta}.\frac{n^{1-\delta}}{k_n^2}\left(\frac{\epsilon^2}{512c_1^2} - \frac{\big((2p+5)k_n^2 + k_n + 1\big)k_n^2 \log\left(\frac{256ec_1^2\beta_2(k_n+1)^4}{\epsilon}\right)}{n}\right)\right]$$

$$< \infty.$$

Applying the above together with Borel-Cantelli Lemma (Lemma 4), we have estimation error approaches to zero.

To handle the approximation error, let us consider a piece-wise constant function (also, referred as pseudo-estimate) in resemblance with the tree t_n. Define

$$(W_{\text{out}}^{\star})_{k''} = \frac{1}{2}E\left[Y|X \in L_{k''}\right] \text{ and } b_{\text{out}}^{\star} = \frac{1}{2}\sum_{k''=1}^{k_n} E\left[Y|X \in L_{k''}\right].$$

Thus, the tree-type pseudo-estimate has the form

$$t_{\lambda^{\star}}(x) = W_{\text{out}}^{\star\top}\tau\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^{\star}) + b_2^{\star}\Big) + b_{\text{out}}^{\star},\ x \in R^p,$$

for some $\lambda^{\star} = (A^{\star}, b_1^{\star}, B^{\star}, b_2^{\star}, W_{\text{out}}^{\star}, b_{\text{out}}^{\star})$ and $\|W_{\text{out}}^{\star}\| \leq \frac{k_n}{2}$.
Also by definition,

$$f_{\lambda^{\star}}(x) = W_{\text{out}}^{\star\top}\left[\sigma_2\Big(B^{\star\top}\sigma_1(A^{\star\top}x + b_1^{\star}) + b_2^{\star}\Big)\right] + b_{\text{out}}^{\star}.$$

Now, let $\psi^{'} \in F_{n,k_n}$ be a function such that

$$E\{|\psi^{'}(X) - g^{*}(X)|\} \leq E\{|\psi(X) - g^{*}(X)|\}$$

for each $\psi \in F_{n,k_n}$. The existence of such a function is justified because $E\{|\psi(X) - g^{*}(X)|\}$ is a continuous function of the parameters of the neural network ψ.

Clearly,

$$\begin{aligned}\inf_{\psi\in F_{n,k_n}} J(\psi) - J^* &\le J(\psi^{'}) - J^* \\ &= E\{|\psi^{'}(X) - Y|\} - E\{|g^*(X) - Y|\} \\ &\le E\{|\psi^{'}(X) - g^*(X)|\} \\ &\le E\{\big|f_{\lambda^\star}(x) - t_{\lambda^\star}(x)\big|\}.\end{aligned}$$

Thus, for any positive k_n and using triangle inequality,

$$\begin{aligned}&\big|f_{\lambda^\star}(x) - t_{\lambda^\star}(x)\big| \\ &\le \frac{k_n}{2} \times \Big\|\sigma_2\Big(B^{\star\top}\sigma_1(A^{\star\top}x + b_1^\star) + b_2^\star\Big) - \tau\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big\| \\ &\le \frac{k_n}{2}\Bigg[\Big\|\sigma_2\Big(B^{\star\top}\sigma_1(A^{\star\top}x + b_1^\star) + b_2^\star\Big) - \sigma_2\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big\| \\ &\qquad + \Big\|\sigma_2\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big) - \tau\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big\|\Bigg] \\ &= \frac{k_n}{2}\Bigg[I_1 + I_2\Bigg], \text{ say.} \qquad (4.18)\end{aligned}$$

Recall that σ_i is hyperbolic tangent activation function and τ is a threshold activation function, then we can write for all $u \in R$,

$$|\sigma_i(u) - \tau(u)| \le 2e^{-2\beta_i|u|} \text{ for all } i = 1, 2.$$

Now, using Lipschitz property and the definition of $B^\star$, we can find the upper-bound for I_1 as follows:

$$\begin{aligned}I_1 &\le \sum_{j=1}^{k_n}\beta_2\Big|\Big(B^{\star\top}\big(\sigma_1(A^{\star\top}x + b_1^\star) - \tau(A^{\star\top}x + b_1^\star)\big)_j\Big| \\ &\le \beta_2 k_n\big\|\sigma_1(A^{\star\top}x + b_1^\star) - \tau(A^{\star\top}x + b_1^\star)\big\| \\ &\le 2\beta_2 k_n\sum_{j=1}^{k_n-1}\exp\Big(-2\beta_1|(A^{\star\top}x + b_1^\star)_j|\Big) \\ &\le 2\beta_2 k_n^2\exp\big(-2\beta_1\varepsilon\big) \text{ (for some fixed } j \text{ and arbitrary } \varepsilon > 0).\end{aligned}$$

For all n large enough, choosing $\varepsilon = \frac{\log(\beta_1)}{2\beta_1}$, we get

$$I_1 \le \frac{2\beta_2 k_n^2}{\beta_1}.$$

Similarly,

$$I_2 \leq 2\sum_{j=1}^{k_n} \exp\Big[-2\beta_2\big|\big(B^{\star\top}\tau(A^{\star\top}x+b_1^\star)+b_2^\star\big)_j\big|\Big] \leq 2k_n e^{-\beta_2}.$$

Since for every j,

$$\big|\big(B^{\star\top}\tau(A^{\star\top}x+b_1^\star)+b_2^\star\big)_j\big| \geq \frac{1}{2} \text{ from the definition of } (A^\star, b_1^\star, B^\star, b_2^\star).$$

Putting these upper bounds of I_1 and I_2 together in (4.18) we get the following:

$$E\big|f_{\lambda^\star}(x) - t_{\lambda^\star}(x)\big| \leq \left[k_n^2 e^{-\beta_2} + \frac{\beta_2 k_n^3}{\beta_1}\right]. \tag{4.19}$$

R.H.S. of (4.19) tends to zero if the conditions (A3) and (A4) of Theorem 13 hold. □

4.5 Computational Experiments

In this section, we first describe the data sets to be used in this study. Subsequently, we are going to report the experimental results and compare our proposed model with other state-of-the-art methods.

4.5.1 Data Description

In our experiments, we take ten software defect prediction data sets from publicly available NASA data sets, which are available at the PROMISE repository (Boetticher, 2007). These data sets are clean data sets that do not have any data quality issues and have been used in current studies on SDP (Gong et al., 2019). Each data set reports the attributes of a software module along with the class label of whether this module contains defects or not. The imbalance rate of these data sets varies from 2.1 to 35.2%. Table 4.2 gives an overview of these SDP data sets.

4.5.2 Results and Comparisons

In order to show the impact of the proposed Hellinger net classifier, it is applied to the standard SDP data sets. We do experiments on ten clean PROMISE data sets for SDP and compare our proposed method with seven competitive methods from the SDP and class imbalanced learning literature. To start, we first shuffled the observations in each of the different PROMISE data sets randomly and split them into training, validation, and test data sets in a ratio of 50 : 25 : 25. We have employed 10-fold cross-validation (CV) with different randomly assigned training,

Table 4.2: Characteristics of the PROMISE SDP data sets (Defect % is the percentage of defective modules)

Data set	Classes	Objects	Attributes	Defect %
CM1	2	327	37	12.8
KC3	2	194	39	18.6
MC1	2	1988	38	2.3
MC2	2	125	39	35.2
MW1	2	253	37	10.7
PC1	2	705	37	8.7
PC2	2	745	36	2.1
PC3	2	1077	37	12.4
PC4	2	1287	37	13.8
PC5	2	1711	38	27.5

Table 4.3: Recall, AUC and F-measures (mean values and their standard deviation) for different classifiers over ten SDP data sets

data set	Measure	ISDA	STr-NN	SMB+DT	VCB-SVM	ABNC+DT	HDDT	HDRF	Hellinger net
CM1	*F-measure*	0.84 (0.02)	0.845 (0.012)	0.792 (0.01)	0.76 (0.03)	0.842 (0.02)	0.785 (0.015)	0.82 (0.054)	**0.865** (0.052)
	AUC	0.55 (0.032)	0.566 (0.02)	0.653 (0.017)	0.595 (0.04)	0.66 (0.019)	0.573 (0.04)	0.671 (0.049)	**0.706** (0.04)
	Recall	0.30 (0.01)	0.38 (0.02)	0.593 (0.001)	0.395 (0.04)	0.320 (0.005)	0.30 (0.01)	0.491 (0.049)	**0.66** (0.034)
KC3	*F-measure*	0.73 (0.01)	0.77 (0.02)	0.78 (0.06)	0.80 (0.02)	0.732 (0.006)	0.892 (0.014)	0.902 (0.014)	**0.832** (0.027)
	AUC	0.56 (0.05)	0.656 (0.04)	**0.748** (0.03)	0.626 (0.017)	0.541 (0.07)	0.532 (0.12)	0.737 (0.01)	0.744 (0.02)
	Recall	0.35 (0.02)	0.5 (0.015)	0.348 (0.024)	0.465 (0.01)	0.33 (0.07)	0.584 (0.03)	0.536 (0.008)	**0.60** (0.02)
MC1	*F-measure*	0.975 (0.001)	0.95 (0.003)	0.965 (0.002)	0.97 (0.001)	0.97 (0.005)	0.94 (0.005)	0.97 (0.004)	**0.985** (0.003)
	AUC	0.64 (0.02)	0.76 (0.03)	0.69 (0.01)	0.66 (0.025)	0.59 (0.01)	0.532 (0.06)	0.70 (0.01)	**0.80** (0.03)
	Recall	0.18 (0.00)	0.58 (0.00)	0.42 (0.001)	0.36 (0.008)	0.17 (0.00)	0.333 (0.00)	0.50 (0.01)	**0.60** (0.00)
MC2	*F-measure*	0.67 (0.02)	0.71 (0.02)	0.66 (0.01)	0.61 (0.04)	**0.75** (0.01)	0.57 (0.06)	0.68 (0.01)	0.73 (0.04)
	AUC	0.638 (0.01)	0.65 (0.01)	0.59 (0.01)	0.57 (0.02)	**0.69** (0.01)	0.43 (0.06)	0.60 (0.01)	0.673 (0.008)
	Recall	**0.60** (0.01)	0.5 (0.00)	0.33 (0.00)	0.26 (0.02)	0.4 (0.01)	0.5 (0.00)	0.55 (0.01)	0.54 (0.02)
MW1	*F-measure*	0.80 (0.04)	**0.91** (0.014)	0.88 (0.144)	0.779 (0.132)	0.832 (0.090)	0.772 (0.120)	0.795 (0.134)	0.902 (0.017)
	AUC	0.612 (0.08)	**0.816** (0.02)	0.70 (0.016)	0.643 (0.014)	0.58 (0.10)	0.535 (0.04)	0.627 (0.1)	0.790 (0.023)
	Recall	0.40 (0.01)	**0.70** (0.02)	0.50 (0.06)	0.35 (0.02)	0.620 (0.05)	0.40 (0.01)	0.55 (0.04)	0.65 (0.03)
PC1	*F-measure*	0.915 (0.02)	0.890 (0.05)	0.89 (0.032)	0.88 (0.03)	0.92 (0.002)	0.882 (0.01)	0.902 (0.04)	**0.945** (0.008)
	AUC	0.586 (0.08)	0.808 (0.027)	0.786 (0.008)	0.67 (0.010)	0.632 (0.07)	0.532 (0.05)	0.719 (0.08)	**0.854** (0.022)
	Recall	0.842 (0.108)	0.858 (0.087)	0.796 (0.118)	0.838 (0.080)	0.815 (0.107)	0.832 (0.090)	0.839 (0.108)	**0.87** (0.07)
PC2	*F-measure*	0.97 (0.00)	0.97 (0.00)	0.96 (0.01)	0.97 (0.01)	0.96 (0.00)	0.95 (0.00)	0.97 (0.00)	**0.98** (0.005)
	AUC	0.5 (0.00)	0.5 (0.00)	0.56 (0.01)	0.45 (0.005)	0.620 (0.04)	0.432 (0.00)	0.570 (0.01)	**0.66** (0.02)
	Recall	0 (0.00)	0.1 (0.00)	0 (0.00)	0 (0.00)	0.05 (0.00)	0.1 (0.00)	0.1 (0.00)	**0.2** (0.00)
PC3	*F-measure*	0.888 (0.01)	0.840 (0.05)	0.82 (0.03)	0.81 (0.02)	0.880 (0.02)	0.796 (0.03)	0.883 (0.017)	**0.907** (0.04)
	AUC	0.598 (0.018)	0.756 (0.012)	0.635 (0.03)	0.654 (0.018)	0.65 (0.02)	0.532 (0.04)	0.614 (0.02)	**0.835** (0.009)
	Recall	0.48 (0.01)	0.67 (0.02)	0.45 (0.03)	0.48 (0.02)	0.38 (0.028)	0.49 (0.01)	0.64 (0.02)	**0.70** (0.005)
PC4	*F-measure*	**0.905** (0.015)	0.86 (0.01)	0.88 (0.01)	0.79 (0.02)	0.89 (0.04)	0.82 (0.006)	0.87 (0.01)	0.90 (0.02)
	AUC	0.805 (0.01)	0.836 (0.01)	0.80 (0.03)	0.67 (0.04)	0.744 (0.03)	0.679 (0.05)	0.783 (0.01)	**0.842** (0.03)
	Recall	0.50 (0.06)	0.80 (0.03)	0.71 (0.01)	0.47 (0.07)	0.53 (0.04)	0.41 (0.09)	0.63 (0.03)	**0.82** (0.03)
PC5	*F-measure*	0.74 (0.05)	0.72 (0.10)	0.75 (0.02)	0.78 (0.02)	0.73 (0.05)	0.78 (0.02)	0.80 (0.01)	**0.825** (0.01)
	AUC	0.65 (0.03)	0.66 (0.04)	0.62 (0.06)	0.68 (0.01)	0.56 (0.05)	0.62 (0.005)	0.64 (0.06)	**0.70** (0.05)
	Recall	0.45 (0.00)	0.360 (0.00)	0.40 (0.00)	0.26 (0.00)	0.35 (0.00)	0.25 (0.00)	0.40 (0.00)	**0.50** (0.00)

Table 4.4: Statistical test results (p-values) between Hellinger net and other comparative methods for SDP data sets

Measure	ISDA	STr-NN	SMB+DT	VCB-SVM	ABNC+DT	HDDT	HDRF
AUC	0	0.01	0.005	0.001	0.02	0	0
Recall	0.01	0.01	0	0.045	0.03	0	0.001
F-measure	0.004	0.05	0.003	0.007	0.032	0.233	0.118

validation, and test sets. Further, we compare our proposed classifier mostly with "imbalanced data-oriented" classifiers as baseline comparisons. Five 'best' class imbalance learning methods are random undersampling SMOTEBoost (SMB) (Chawla et al., 2003), AdaBoost.NC (ABNC) (Wang and Yao, 2013), stratification embedded in nearest neighbor (STr-NN) (Gong et al., 2019), improved subclass discriminant analysis (ISDA) (Jing et al., 2016) and value-cognitive boosting with support vector machine (VCB-SVM) (Ryu et al., 2016) that are used for comparison study in this study. A python toolbox, namely 'imbalanced-learn', is used to tackle the curse of imbalanced data sets, provides an application of a wide range of available sampling methods that have been used in our study (Lemaître et al., 2017). We implement these two sampling techniques using *"imbalanced-learn"* package in python with the default parameters available in the toolbox. These methods provide us balanced data sets with equal class distributions. On the balanced data sets, we use the well known DT learner in the SMB and ABNC sampling methods as the base classifier for all the experiments. It is the most commonly discussed technique in SDP literature and we use the default parameter setting available in the 'scikit-learn' package in python toolbox. We only disable the tree pruning since it may remove leaves from the minority class concept. In the case of another individual learner NB, we use data pre-processed by the log filter, and another ensemble method RF is trained by using 50 unpruned trees for all the data sets used here. We implemented the HDDT algorithm based on Cieslak et al. (2012) for learning from imbalanced SDP data sets. HDDT usually achieved higher accuracy than standard DT. This indicates that "imbalanced data-oriented" classifiers perform better than the conventional supervised classifiers designed for general purposes. Further, we implemented HDRF based on Su et al. (2015) which is among other imbalanced data-oriented algorithms. Implementation of sampling technique methods requires careful verification of parameter settings so that one can control the strength of emphasizing the minority class prior to the learning algorithm. SMB proceeds with 50 classifiers constructed based on the training data, with SMOTE applied at each round of Boosting. The number of nearest neighbors is chosen to be five based on the recommendation of Wang and Yao (2013). ABNC has, as a prerequisite, random oversampling applied to the minority class first to ensure both the classes have the same size. Then 50 classifiers are constructed sequentially by ABNC. λ is a penalty strength assigned for encouraging the ensemble diversity and it is varied from 1 to 20 with the increment of 1 for all the experiments.

Then we started experimenting with our proposed model. The training procedure for the optimal hybrid model is as follows. HDDT is first built using the 'scikit-learn' package implementation (Pedregosa et al., 2011) for tree designing. From the HDDT, we extracted the set of all split directions and split positions and used them to build neural network initialization parameters. The hybrid models are then trained using the 'TensorFlow' library in python software (Abadi et al., 2016). The optimization with the network model is done by minimizing the empirical error on the training set. It is achieved by employing an iterative stochastic gradient-descent optimization technique. The architecture of the network is kept fixed, and thus the weights and offsets of the three-layered DFFNN model are also fixed. A natural idea is then to keep the structure of the network intact and let the parameters vary in a subsequent network training procedure with backpropagation neural network training. We have used the default functions available in *TensorFlow* for this. DFFNN, in the hybrid set up, was trained for 100 epochs. The default hyper-parameter values were chosen for the gradient-based optimization algorithm available in *Python* machine learning software. We have experimentally found that using a lower value for β_2 than that for β_1 is appropriate for achieving the high accuracy of the model. In this case the initial parameters of the tan-hyperbolic activation function in the two layers were chosen as: $\beta_1 = 100, \beta_2 = 1$. This is also practically very significant since for a relatively small β_2, the transition in the activation function from -1 to $+1$ is smoother and a very stronger stochastic gradient signal reaches the first hidden layer in backpropagation training. Similarly, a converse explanation can be given for β_1. The training time and memory requirements are also quite low for the hybrid model compared to advanced deep neural network models. Our proposed hybridization is faster, especially when trained on a GPU. In Table 4.3, we present the results of different classifiers on the SDP data sets and the best results are displayed in bold fonts.

To sum up, among seven competitive methods, the proposed method is the winner according to the performance metrics (Recall, AUC value, and F-measure) for the seven out of ten NASA SDP data sets. From the statistical point of view on the nature of the SDP problem, the robustness of the proposed method to class imbalance problem in SDP implies that the extracted features are appropriate for describing the software modules' attributes. Moreover, STrNN and ISDA are 'second' and 'third' best choices as an imbalanced classifier in terms of the performance metrics for the majority of the SDP data sets as compared to the other traditional methods considered in this study. From the experimental evaluation of different classifiers, it can be concluded that, on average, the Hellinger net model outperforms other individuals, ensemble, and hybrid statistical and machine learning models by a significant margin. The possible reasons for the failure of the earlier state-of-the-arts in comparison with our proposed approach for 70% of the data sets are: (a) Methods like STrNN, ABNC, VCB, SMB, and ISDA only focused on the characteristics of the source data set and worked on data-level; (b) These methods do not consider the information of target data set to select the appropriate training data; (c) These methods do not consider the class imbalance problem in algorithmic level. Thus, the proposed Hellinger net method can be a 'good' choice for imbalanced learning in SDP data sets to the software engineers working on early defect predictions and software quality improvements.

4.5.3 Significance of Improvements

In this section, we comment on the significance of the improvement and describe the potential threats to impact the results of our studies.

- **(a) Level of Imbalance:** It is clear from Table 4.2 that the proposed Hellinger net model is able to handle data sets with an imbalance rate varying from 2.1% to 35.2%. There is further validation required from other standard UCI imbalanced data sets and simulated data sets which are given in Section 4.6 and 4.7.

- **Potential Improvements:** Finally, from Table 4.3, we can see that there is a significant difference between our proposed Hellinger net model and other state-of-the-art approaches in terms of Recall, F-measure, and AUC. Since defective instances are more important than clean examples in software defect testing, the Hellinger net improves the Recall values that well relieves the class-imbalance problems for SDP. In terms of AUC, Recall, and F-measure, our proposed method performed superior to all the state-of-the-art for seven out of ten data sets and has shown significant improvement over them.

- **Statistical Significance:** To determine the statistical significance of comparative methods, we performed a Wilcoxon signed-rank test. This is a distribution-free test between two classifiers where we make a hypothesis that there exist no significant differences between our proposed Hellinger net model and each of the other approaches at a confidence level of 95%. If the p-value of the test is below 0.05, we conclude that there is a significant difference between our proposed Hellinger net and each of the other state-of-the-art approaches. These results are presented in Table 4.4.

- **Validity of Performance Measures:** We consider only F-measure, AUC, and Recall as the performance metrics in this study. There are different metrics available in the literature and it may influence the performance of classifiers. However, our experiments' performance measures are carefully selected to ensure the reliability of experimental results, including F-measure, AUC, and Recall. In future work, other performance measures will be considered.

- **Data Validity:** We choose ten cleaned data sets from publicly available data sets PROMISE and NASA which have been widely used in many studies for SDP. Our selected data sets are diverse in size, features, and percentage of defective instances, and this helps draw the generalization of our findings. Also, we consider a larger number of data sets, and this further validates our studies. However, further investigations on some commercial data sets are required which are given in Section 4.6.

- **Conclusion Validity:** Threats to conclusion validity concern the relationship between the treatment and the outcome. In these experiments, we employed 10-fold cross-validation while experimentation with SDP data sets to avoid random bias, and calculate the average results to achieve the performance of all compared approaches. F-measure, AUC, and Recall are used to measure the effectiveness of the compared methods. Furthermore, the Wilcoxon signed-rank test is used to compare the state-of-the-art methods statistically.

4.6 Experimental Analysis with UCI Data

In this section, we describe the standard UCI data sets in brief and also discuss the performance evaluation metric. Subsequently, we are going to report the experimental results and compare our proposed model with other state-of-the-art classifiers.

4.6.1 Data Description

The proposed Hellinger net model is evaluated using five publicly available data sets from a wide variety of application areas such as management, business, and medicine, available at UCI Machine Learning repository (Asuncion and Newman, 2007). The breast cancer data set consists of 9 discrete features, whereas the Pima diabetes data set has 8 continuous features in its feature space. German credit card data set (also popularly known as Statlog data set) consists of 13 qualitative features and 7 numerical features. In this data set, entries represent persons who take credit from a bank, and each person is classified as good or bad credit risks according to the collection of attributes. Page blocks database has numeric attributes, contains blocks of the page layout of a document that has been detected by a segmentation process. Indian business school data set includes 10 continuous and 7 categorical variables on the characteristics of students admitting in a business school and the response variable denotes whether the student will be placed or not at the end of the curriculum (Chakraborty et al., 2018). To measure the level of imbalance of these data sets, we compute the coefficient of variation (CV), which is the proportion of the deviation in the observed number of samples for each class versus the expected number of examples in each class

(Wu et al., 2010). We have chosen the data sets with a CV more than equal to 0.30− a class ratio of 2 : 1 on a binary data set as imbalanced data. Table 4.5 gives an overview of these data sets.

Table 4.5: Characteristics of the UCI data sets used in experimental evaluation

Data set	Classes	Objects (n)	Number of feature (p)	Number of (+)ve instances	Number of (−)ve instances	CV
breast cancer	2	286	9	201	85	0.41
german credit card	2	1000	20	700	300	0.40
indian business school	2	480	17	400	80	0.56
page blocks	2	5473	10	4913	560	0.80
pima diabetes	2	768	8	500	268	0.30

4.6.2 Results and Comparisons

In order to show the impact of the proposed classifier, it is applied to the high-dimensional small or medium-sized data sets from various applied areas. These are such types of data sets in which not only classification is the task but also feature selection plays a vital role as well. To start, we first shuffled the observations in each of the five different data sets randomly and split them into training, validation, and test data sets in a ratio of 50 : 25 : 25. We have repeated each of the experiments five times with different randomly assigned training, validation, and test sets. Further, we compare our proposed classifier mostly with "imbalanced data-oriented" classifiers as baseline comparisons. Even we apply different sampling approaches over traditional classifiers and evaluate AUC values to see the competitiveness of the proposed model.

We start the experimental analysis by implementing a CT algorithm to five publicly available imbalanced data sets. A tree-based CT model is trained using *'rpart'* package implementation in R. CT uses the Gini index, and C_p has been used for the selection of variables to enter and leave the tree structure. Further, an ensemble of trees, random forests (RF), was implemented using *'randomForest'* package in R. We report their prediction performances in Table 4.6. Another simple nonparametric algorithm, k-nearest neighbor (kNN) is applied to the data sets using *class* implementation in R. To implement neural nets, we first standardize the data sets and run the ANN model. And we used the "logsig" transfer function to bring back the original form at the end of modeling. We implemented the ANN model with different combinations of hidden layers without employing any other feature selection algorithm using *'neuralnet'* package. Since the data sets are small or medium sample-sized, thus going beyond two hidden layered (2HL) neural nets will overfit the data sets. For one hidden layer (1HL) ANN model, the number of hidden neurons is chosen as 2/3 the size of the input layer, plus the size of the output layer. But for the 2HL ANN model number of neurons in 1st HL is chosen as 2/3 the size of the input layer and the number of neurons in 2nd HL are chosen as 1/3 the size of the input layer. RBFN is a particular class of ANN that uses a radial basis kernel for nonlinear classification. Using the *'RSNNS'* package, we applied the RBFN model with the Gaussian kernel function, and the maximum number of iterations in all these NN implementations is chosen as 100. Execution time for the RBFN model is lesser than ANN but higher than tree-based models.

Table 4.6: AUC results (and their standard deviation) of classification algorithms over original imbalanced test data sets

Classifiers	breast cancer	German credit card	Indian business school	page blocks	pima diabetes
CT	0.603 (0.04)	0.665 (0.03)	0.810 (0.04)	0.950 (0.00)	0.724 (0.02)
RF	0.690 (0.06)	0.725 (0.03)	0.850 (0.04)	0.964 (0.00)	0.747 (0.04)
k-NN	0.651 (0.03)	0.727 (0.01)	0.750 (0.03)	0.902 (0.02)	0.730 (0.05)
RBFN	0.652 (0.06)	0.723 (0.04)	0.884 (0.05)	0.935 (0.01)	0.725 (0.04)
HDDT	0.625 (0.04)	0.738 (0.04)	0.933 (0.02)	0.974 (0.00)	0.760 (0.02)
HDRF	0.636 (0.04)	0.742 (0.03)	0.939 (0.02)	**0.988** (0.00)	0.760 (0.03)
ANN (with 1HL)	0.585 (0.03)	0.700 (0.03)	0.768 (0.05)	0.918 (0.02)	0.649 (0.03)
ANN (with 2HL)	0.621 (0.02)	0.715 (0.02)	0.820 (0.04)	0.925 (0.01)	0.710 (0.03)
Hellinger net	**0.730** (0.05)	**0.802** (0.03)	**0.968** (0.01)	0.980 (0.02)	**0.809** (0.03)

We now implemented the HDDT algorithm by using R Package 'CORElearn' for learning from imbalanced data sets. HDDT usually achieved higher accuracy than CT and RF. It indicates that "imbalanced data-oriented" classifiers perform better than the conventional supervised classifiers designed for general purposes. Further, we implemented HDDT and HDRF which are among other imbalanced data-oriented algorithms. Finally, we applied our proposed Hellinger net classifier which is a tree-to-network-mapped model. The implementation of the Hellinger net model is similar to the one described in Section 4.5.2. We reported the performance of various classifiers in terms of AUC value in Table 4.6. It is clear from Table 4.6 that our proposed methodology achieved better performance than the other state-of-the-art models for most of the UCI data sets used in this study except the page blocks data. It is to be noted that even in the case of page blocks data, the difference in the AUC between HDRF and Hellinger net is marginal. We have highlighted the highest AUC value in the table with bold for all the data sets. It is clear from computational experiments that our model stands as very much competitive with state-of-the-art.

4.7 Simulation Study

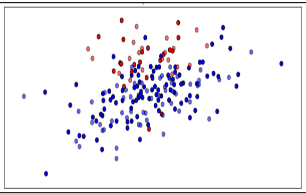
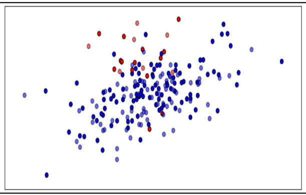
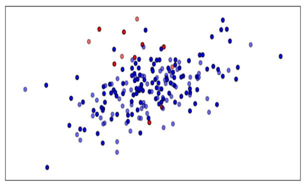
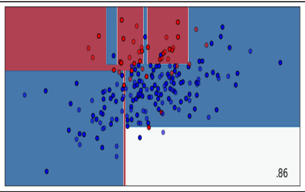
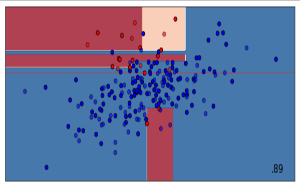
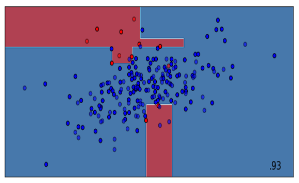
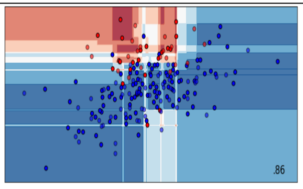
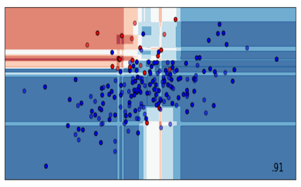
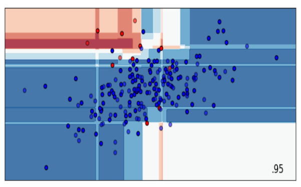
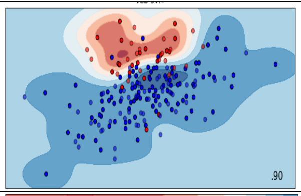
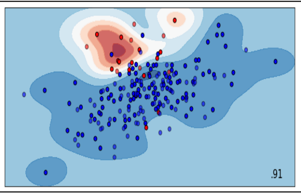
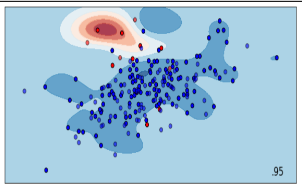
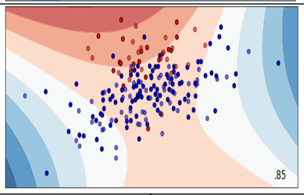
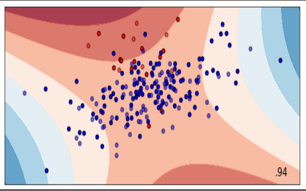
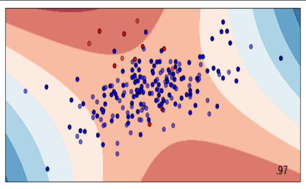
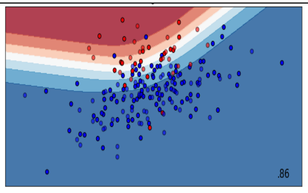
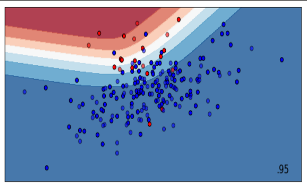
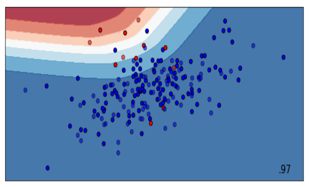
Real-world data sets commonly show the particularity to have a number of samples of a given class under-represented compared to other classes. This imbalance gives rise to the “class imbalance” problem or “curse of imbalanced data sets” (Chawla et al., 2003) which is the problem of learning a concept from the class that has a small number of samples. This section provides a comparison of our proposed Hellinger net model with several other “imbalanced data-oriented” classifiers on synthetic data sets from IMBALANCED-LEARN library in Python. IMBALANCED-LEARN is an open-source python toolbox aiming at providing various imbalanced data sets along with a wide range of methods to cope with the problem of imbalanced data frequently encountered in machine learning and pattern recognition (Lemaître et al., 2017). The point of this example is to illustrate the nature of decision boundaries of different classifiers and to understand how well they perform on minority class examples. Three toy data sets (binary) are generated with weights = [0.2, 0.8], [0.1, 0.9] and [0.05, 0.95], i.e., data sets with imbalance rates of 20%, 10% and 5%, respectively. These data sets will be useful to visualize the decision boundaries of the classifiers used in this chapter and also to understand the percentage of imbalance rate that can be handled by the proposed Hellinger net model. The ‘linearly_separable’ function in SCIKIT-LEARN library generates 100 samples with moderate noise level. We added Gaussian noise to the data with the standard deviation equals to 0.5. This test problem is suitable for algorithms that can learn data imbalance problems in complex nonlinear manifolds. The example (refer to the input data plots in Table 4.8) generates data sets with Gaussian noise. All these toy data sets represent binary imbalanced classification problems in 2D. In all the experiments, 60% of the data samples are used for training, and the rest of them are for testing. The classification accuracy of all the models in three synthetic data sets is reported in Table 4.7.

Table 4.7: AUC results of different imbalanced classifiers on three synthetic data sets.

Imbalanced Classifiers	Simulated Data with IR = 20%	Simulated Data with IR = 10%	Simulated Data with IR = 5%
HDDT	0.80	0.85	0.91
HDRF	0.82	0.88	0.91
VCB-SVM	**0.87**	0.89	0.93
ISDA	0.84	0.91	0.90
Hellinger net	0.86	**0.92**	**0.95**

In order to assess our proposed Hellinger net model, we perform experiments on these data sets by employing all the algorithms and generate a graph representation of each data set. The implementation of all these models is done as discussed in Section 4.5.2. The choice of tuning parameters for all the models is as follows. For the HDDT model, the maximum depth of the tree is set to 5 for all the examples. In the case of HDRF, we set the number of trees to be built as 30. Finally, our proposed model is applied to the data sets and the results are reported in Table 4.7. As shown in the plots of Table 4.8, our approach is able to correctly classify minority samples and achieves the highest AUC values in comparison with the other imbalanced classifiers for two data sets. This result seems to confirm that the proposed approach can deal with a highly imbalanced data structure.

Table 4.8: A comparison of several imbalanced classifiers on synthetic data sets. The plots show training points in solid colors and testing points semi-transparent. The lower right in each plots shows the classification accuracy on the test set.

Synthetic data	Imbalance = 20% rate	Imbalance = 10% rate	Imbalance = 5% rate
Input Data			
HDDT			
HDRF			
VCB-SVM			
ISDA			
Hellinger net			

4.8 Conclusions and Discussion

SDP is a widely popular research domain in software reliability engineering. The primary objective of the SDP is to find as many defective software modules as possible in the initial stages of software testing. The difficulty in building an effective prediction model with high performance in SDP data sets is due to the imbalance nature in its data characteristics. This chapter proposed a novel hybrid model, namely the Hellinger net, for improving predictions in binary imbalanced SDP problems.

Our proposed model considers data imbalance and overcomes the deficiencies of component tree-based and neural network models. We experimented with 10 NASA SDP data sets from the PROMISE repository to validate the performance of our proposed method in comparison with other state-of-the-art methods. Different training algorithms like Naive Bayes with log filter and Random Forest were implemented on the SDP data sets. Then, two sampling methods, such as SMOTEBoost and AdaBoost.NC, along with base classifier decision trees, were employed on these SDP data sets. Further, 'imbalanced data-oriented' classifiers, namely HDDT and HDRF, were tested on these data sets. But the usefulness of the proposed Hellinger net model lies in its theoretical robustness, overall excellent performance, and easy interpretability compared to complex "black-box-like" models. Finally, the proposed hybrid classifier, namely the Hellinger net, based on HDDT and neural networks, was implemented to improve HDDT and overcome its drawbacks.

The proposed Hellinger net classifier offers the advantage of the highest accuracy in terms of two popular performance evaluation metrics (AUC and F-measure) as compared to the traditional models discussed in this chapter. The experimental results depicted that the proposed Hellinger net is adequate to solve the class-imbalance problems. Hellinger net model also has the desired statistical properties like theoretical consistency, easy interpretability, and achieves higher accuracy. The work in this chapter is mostly focused on the development of an imbalanced classifier for the improvements of software defect predictions on NASA data sets. But we also provide results with standard UCI and simulated data sets to show the general applicability of the proposed Hellinger net model. An immediate extension of this chapter is to extend this work for imbalanced classification problems in the presence of a conceptual shift that happens to be a key research area in industrial statistics. It is also essential to look for other SDP scenarios like learning from data with minimal defective modules and many unlabelled modules (e.g., semi-supervised setting), which can also be a possible extension of this work. Lastly, one can also try to use the Hellinger net model in data imbalance problems from other domains to find the extensive usage of the proposed methodology. In the next chapter, we will discuss another type of prediction problem, namely, regression estimation.

Chapter 5

A Distribution-free Hybrid Method for Regression Modeling

Related Publications:

1. Chakraborty T, Chattopadhyay S, Chakraborty A.K. (2020) Radial basis neural tree model for improving waste recovery process in a paper industry. *Applied Stochastic Models in Business and Industry*, 36, 49-61.

2. Chakraborty, T., Chakraborty, A. K., Chattopadhyay, S. (2019). A novel distribution-free hybrid regression model for manufacturing process efficiency improvement. *Journal of Computational and Applied Mathematics*, 362, 130-142.

Summary

This work is motivated by a particular problem of a modern paper manufacturing industry, in which the maximum efficiency of the waste recovery process is desired. As a by-product of the paper manufacturing process, a lot of waste along with valuable fibers and fillers come out from the paper machine. The waste recovery process (WRP) involves separating the unwanted materials from the valuable ones so that the recovered fibers and fillers can be further reused in the production process. This job is done by fiber-filler recovery equipment (FFRE). The efficiency of FFRE depends on several crucial process parameters and monitoring them is a difficult proposition. To solve this problem, we propose a novel hybrid methodology, namely, radial basis neural tree (RBNT) model, for waste recovery process improvement in a paper industry. The proposed model can be useful to find the essential parameters from the set of available data and perform prediction tasks to improve WRP efficiency. An idea of parameter optimization along with regularity conditions for the universal consistency of the proposed model are given. The proposed model performs superior when applied to the FFRE efficiency improvement problem. This work will help the paper company to become environmentally friendly with less ecological damage apart from being cost-effective.

5.1 Introduction

Regression problems arise in many practical situations where a specific response variable can be expressed through a relationship with the so-called causal variables. In practical applications, it becomes quite challenging to identify the right set of causal variables. This chapter is motivated by a specific problem in a modern paper industry that produces papers for multiple uses. Paper machines usually produce papers by using pulp, fibers, fillers, chemical lubricants, and substantial water resources. As a by-product of the paper manufacturing process, many waste materials come out from the paper machine. These waste materials contain lean water, garbage, and valuable fibers, fillers. To save the expensive and reusable materials from the waste, an equipment called FFRE (also popularly known as Krofta supercell) is used by many paper manufacturing industries (Krofta, December 2, 1986,D). The dissolved air flotation cum sedimentation process (DAFSP) is used in FFRE to collect valuable materials from waste. If FFRE becomes efficient, it will help the company to cut costs by reusing the relevant materials in the production process. However, the efficiency of FFRE is not always satisfactory, and that causes a monetary loss for the company. A lot of preliminary analysis was done to find out a set of possible causal variables that affect FFRE efficiency, which, in other words, means high recovery percentage of valuable materials. The objective of this work is to correctly find out the most important process parameters from the set of all available causal variables. Also, we will try to develop a prediction model that can help the company to estimate future losses. This work aims to help the company in improving the quality and productivity of the paper manufacturing process. Furthermore, process improvement through waste management and valuable material recovery can make the manufacturing process environmentally friendly with very less ecological damage (Jiang et al., 2018).

The problem can be viewed as a typical nonparametric regression problem where one can establish a relationship between the response variable (recovery percentage of FFRE) and the major causal variables (process parameters of DAFSP) without having any prior information about the data. Regression trees (RT), support vector regression (SVR), and artificial neural networks (ANN) have been applied for various prediction tasks in water quality improvement, water planning, and many other related problems (Bhattacharya and Solomatine, 2005; Gmar et al., 2017; Mahuli et al., 1993; Shrimali and Singh, 2001). Even various hybrid regression models have been developed for performing regression tasks in many real-life problems like water demand forecasting and others (Brentan et al., 2017; Cancho et al., 2016; Lee and Chen, 2005; Sebri, 2016). To develop a prediction model for the waste recovery improvement that can also determine critical parameters among the set of possible parameters, we take recourse to nonparametric regression methodology. The tree-structured model, RT, is famous for its easy interpretability and built-in mechanism of important variable selection (Breiman et al., 1984), but they are sometimes unstable. On the other hand, neural networks were developed to mathematically model human intellectual abilities by biologically plausible engineering designs (Kuncheva, 2004), but it may fail when limited data are available (Pektas and Cigizoglu, 2017). Among the family of ANNs, radial basis function networks (RBFN) have the advantages of having only one hidden layer, less time complexity, and easy interpretability. RBFN models are found to be more effective than ANN in practical problems like daily trip flow modeling (Celikoglu and Cigizoglu, 2007) and travel time measure specification (Celikoglu, 2011). RBFN theory has been applied to various problems, like estimation, prediction, and/or classification problems in previous literature (Silgu and Celikoglu, 2015). To harness the advantages of single models, several previous works have concentrated on the hybridization of RT and ANN models, viz. entropy nets (Sethi, 1990) and neural tree (Sirat and Nadal, 1990), and various others (Frosst and Hinton, 2017; Humbird et al., 2018; Kim, 2016; Pektaş and Cigizoglu, 2013; Tsai et al., 2012).

Motivated by the above discussion, we have proposed a radial basis neural tree (RBNT) model that utilizes both the tree-based models and ANN to solve the production process efficiency problem (Chakraborty et al., 2019a, 2020b). In the hybridization, we have used RT as a feature selection algorithm and utilized RT given features along with RT predicted values as input features in the RBFN model. Since RBFN doesn't need pre-specifying the number of hidden nodes unlike other ANNs, thus, we have used RBFN in the proposed RBNT model. We also try to fill the gap between theory and practice by showing the consistency of the proposed hybrid regression model. Our model combines RBFN with RT to enhance the predicting accuracy for this typical regression problem. In the FFRE data set, parameter optimization within the proposed RBNT framework yields the proposed model that is intermediate between RT and RBFN and outperforms RT, RBFN, and other commonly used nonparametric regression models. The proposed model has the advantages of higher accuracy, converges much faster than many other complex hybrid models, and easy interpretability compared to many advanced "black-box-like" models. Unlike parametric regression models, the proposed model has no assumptions on the distribution of the input and output variables. When applied to solve the process efficiency problem, we found a set of most important process parameters from the available data that controls process recovery percentage. Our proposed model is found useful for forecasting future process efficiency in terms of recovery percentage until the company can make suggested changes to the process for its improvement. To summarize, the contributions are:

- We propose a two-step pipeline model, namely, the RBNT model for improving the waste recovery process based on decision tree and neural networks.
- Optimization of model parameters and the regularity conditions for the universal consistency of the proposed model are discussed.

- The robustness of the proposed algorithm is shown through experimental evaluation on the FFRE data set.

5.2 Radial Basis Neural Tree (RBNT) Model

One of the ultimate goals of designing a regression model is selecting the best possible regressors that can predict the response variable accurately. RT is a nonparametric regression technique that has a built-in mechanism for feature selection. RBFN is a particular type of non-linear neural network which is more intuitive than the multilayered perceptron (MLP). Generally, RT uses a hierarchical segmentation of the input feature spaces, whereas RBFN uses a radial basis function as an activation function in its network structure. Both models do not assume normality of the data, nor do they assume homoscedasticity of noise terms. In the proposed RBNT model, we first split the input feature space into areas by RT algorithm. Based on feature rankings provided by RT, a set of important features are chosen and extracted from the training data set. We then build the RBFN model using the important variables obtained through the RT algorithm along with the prediction results obtained from RT as another input information in the input layer of the network. The effectiveness of the proposed classifier lies in the selection of important features and the use of prediction results of RT followed by the application of the RBFN model. The inclusion of RT output as an additional input feature improves the model accuracy in a significant margin. The proposed RBNT regression model can handle high-dimensional data sets through the implementation of RT in selecting features as well as the incorporation of its predicted outputs tied up with one hidden layered RBFN model. This hybridization improves the performances of RT, RBFN, and it also reduces the biases and variances of these individual models. A simple workflow of our proposed model is as follows:

- Apply RT algorithm to train and build a decision tree. Use the tree to extract the important features and find the splits between different adjacent values of the features.
- Choose the features that have minimum mean squared error as important input variables and record RT predicted outputs.
- Export important input variables along with an additional feature (prediction values of RT algorithm) to the RBFN model and a neural network is generated.
- RBFN model uses a Gaussian kernel as an activation function, and parameter optimization is done using a gradient descent algorithm. Finally, we obtain the final outputs.

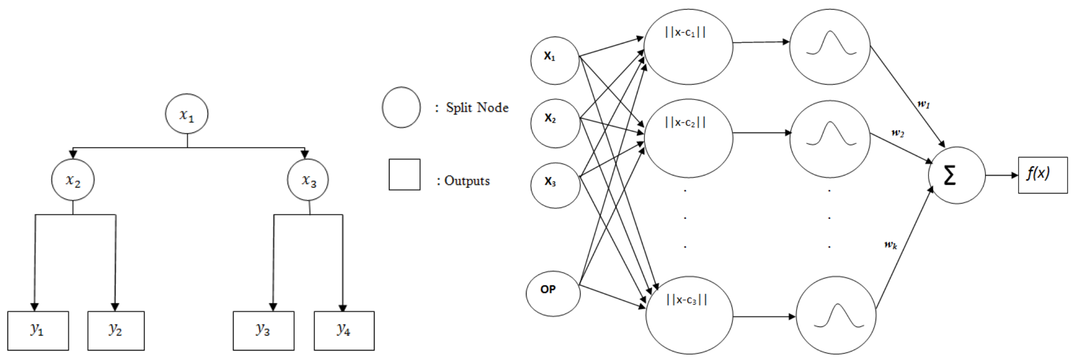


Figure 5.1: An example of RBNT with $x_i; i = 1, 2, 3$ as important features obtained by RT, $y_j; j = 1, 2, 3, 4$ as leaf nodes and OP as RT output. An RT (Left) and one hidden layered RBFN model (Right).

RBNT model is a two-step pipeline approach such as choosing the most important features among the set of available features and finally getting improved prediction results. Since RT is robust in handling the curse of high-dimensional data sets, incorporating its predicted results and important features obtained by RT as input features in RBFN will necessarily improve the model's performance. The proposed RBNT model can be used for feature selection cum prediction task in regression problems. On the theoretical side, it is necessary to prove the universal consistency of the model for its robustness (to be discussed in the next section). A flowchart of the RBNT model is presented in Figure 5.1.

5.3 Statistical Properties of the RBNT Model

In this section, we investigate the theoretical consistency of the proposed model by introducing a set of regularity conditions on the RBNT model. Further, an idea about parameter optimization in the RBNT model is also proposed.

5.3.1 Regularity Conditions for Universal Consistency

To explore the statistical properties, we are going to investigate the sufficient condition(s) for the consistency of RT. For a wide range of data-dependent partitioning schemes, the statistical consistency of histogram-based regression estimates was shown in the literature (Nobel, 1996). It requires a set of regularity conditions to be satisfied to show the consistency of histogram regression estimates. Also, it is assumed to have regression variables to be bounded throughout. But in our case, we will represent regression trees where the partitions are chosen to have rectangular cells, i.e., regression trees employing axis-parallel splits and response variable can take values within a specific range. There is no assumption made on the distributions of predictor and response variables.

Consider a nonparametric regression framework where $\underline{X}$ is the space of all possible values of p features and $\underline{Y}$ is the set of all possible outcomes. We also assume that the response variable can take values in $[-K, K]$, where $K \in \mathbb{R}$. We try to predict a regression function $r(x) = E(Y|X = x) \in [-K, K]$ based on the given n training samples, $L_n = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$. Also let $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ be a partition of the feature space $\underline{X}$ based on RT algorithm. We denote $\widetilde{\Omega}$ as one such partition of Ω. Define $(L_n)_{\omega_i} = \{(X_i, Y_i) \in L_n : X_i \in \omega_i, Y_i \in [-K, K]\}$ to be the subset of L_n induced by ω_i and let $(L_n)_{\widetilde{\Omega}}$ denote the partition of L_n induced by $\widetilde{\Omega}$.

The criterion used to identify the best features is mean squared error (MSE) for RT model. We don't make any assumption on the distribution of the pair $(\underline{X}, \underline{Y}) \in \mathbb{R}^p \times [-K, K]$. MSE is used to partition feature space into a set $\widetilde{\Omega}$ of nodes. Thus, there exists a partitioning regression function $d : \widetilde{\Omega} \to Y$ such that d is constant on every node of $\widetilde{\Omega}$. Now let us define $\widehat{L}_n$ to be the space of all learning samples and $\mathbb{D}$ be the space of all partitioning regression function. Then, we define a binary partitioning and regression tree based rule $\Phi : \widehat{L}_n \to \mathbb{D}$ such that $\Phi(L_n) = (\psi \circ \phi)(L_n)$, where ϕ maps L_n to some induced partition $(L_n)_{\widetilde{\Omega}}$ and ψ is an assigning rule which maps $(L_n)_{\widetilde{\Omega}}$ to a partitioning regression function d on the partition $\widetilde{\Omega}$. The most basic reasonable assigning rule ψ is the plurality rule $\psi_{pl}((L_n)_{\widetilde{\Omega}}) = d$ such that if $x \in \omega_i$, then

$$d(\underline{x}) = \arg \min_{Y_i \in [-K,K]} \left|\frac{1}{n} \sum_{i=1}^{n} (\Phi(L_n) - Y_i)^2\right|$$

The stopping rule in RT is decided based on the minimum number of split in the posterior sample called "minsplit". If "minsplit" $\geq \alpha$ then ω_i will split into two child nodes and if "minsplit" $< \alpha$ then ω_i is a leaf node and no more split is required. Here α is determined by the user, usually it is taken as 10% of the training sample size.

Now let $\mathcal{T} = (\widetilde{\Omega}_1, \widetilde{\Omega}_2, ...)$ be a finite collection of partitions of a measurement space $\underline{X}$. Let us define maximal node count of $\mathcal{T}$ as the maximum number of nodes in any partition $\widetilde{\Omega}$ in $\mathcal{T}$ which can be written as

$$\lambda(\mathcal{T}) = \sup_{\widetilde{\Omega}_i \in \mathcal{T}} |\widetilde{\Omega}_i|$$

Also let, $\Delta(\mathcal{T}, L_n) = |\{(L_n)_{\widetilde{\Omega}} : \widetilde{\Omega} \in \mathcal{T}\}|$ be the number of distinct partitions of a training sample of size n induced by partitions in $\mathcal{T}$. Let $\Delta_n(\mathcal{T})$ be the growth function of $\mathcal{T}$ defined as

$$\Delta_n(\mathcal{T}) = \sup_{\{L_n : |L_n| = n\}} \Delta(\mathcal{T}, L_n).$$

Growth function of $\mathcal{T}$ is the maximum number of distinct partitions of $(L_n)_{\widetilde{\Omega}}$ that partitions $\widetilde{\Omega}$ in $\mathcal{T}$ and can be induced in any training samples with n observations. For a partition $\widetilde{\Omega}$ of $\underline{X}$, $\widetilde{\Omega}[x \in \underline{X}] = \{\omega_i \in \widetilde{\Omega} : x \in \omega\}$ be the node ω_i in $\widetilde{\Omega}$ which contains x.

For consistency of any histogram based regression estimates, the sub-linear growth of restricted cell counts (see in equation 5.1), sub-exponential growth of a combinatorial complexity measure (see in equation 5.2) and shrinking cell (see in equation 5.3) conditions are to be satisfied (Nobel, 1996):

$$\frac{\lambda(\mathcal{T}_n)}{n} \to 0 \quad \text{as} \quad n \to \infty \tag{5.1}$$

$$\frac{log(\triangle_n(\mathcal{T}_n))}{n} \to 0 \quad \text{as} \quad n \to \infty \tag{5.2}$$

and for every $\gamma > 0$ and $\delta \in (0,1)$,

$$\inf_{S \subseteq \mathbb{R}^p :\ P(S) \geq 1-\delta} P(x : diam(\tilde{\Omega}_n[x] \cap S) > \gamma) \to 0 \quad \text{with probability 1.} \tag{5.3}$$

Our objective is to provide a single sufficient regularity condition for the binary partitioning and regression tree-based rule to be universally consistent. Optimal binary regression trees are shown to be consistent when the size of the tree grows as $o(\frac{n}{log(n)})$, where n is the number of training samples. Now we are going to show that instead of satisfying regularity conditions of equations (5.1), (5.2) and (5.3), as in Nobel (1996), if a regression tree estimate Φ satisfies the condition of Theorem 14, then RT will be consistent.

Theorem 14 *Suppose $(\underline{X}, \underline{Y})$ be a random vector in $\mathbb{R}^p \times [-K, K]$ and L_n be the training set of n outcomes of $(\underline{X}, \underline{Y})$. Finally if for every n and $w_i \in \tilde{\Omega}_n$, the induced subset $(L_n)_{w_i}$ contains at least k_n of the vectors of $X_1, X_2, ..., X_n$, then empirically optimal regression trees employing axis parallel splits are consistent when the size k_n of the tree grows as $o(\frac{n}{log(n)})$.*

Proof Since $\mathcal{T}(k_n)$ contains all the binary RT partitions having k_n leaves,

$$\lambda(\mathcal{T}(k_n)) = k_n.$$

Therefore, $\frac{\lambda(\mathcal{T}(k_n))}{n} = \frac{k_n}{n}$ tends to zero as $n \to \infty$. Hence,

$$\frac{\lambda(\mathcal{T}(k_n))}{n} \to 0.$$

Thus, the condition (5.1) holds.

Now, regression trees having k_n leaves has $k_n - 1$ internal nodes and therefore each partition is based on at most $k_n - 1$ intersecting half-spaces. Any binary split of $\mathbb{R}^p$ can divide n points in at most n^p ways based on the Cover's theorem (Cover, 1965). So, their intersection will partition n points in at most $n^{(k_n-1)p}$ ways. Thus we write the following:

$$\triangle_n(\mathcal{T}(k_n)) \leq n^{(k_n-1)p},$$

and consequently,

$$\frac{log(\triangle_n(\mathcal{T}(k_n)))}{n} \leq \frac{p(k_n-1)}{n} log(n). \tag{5.4}$$

As, $n \to \infty$, R.H.S. of equation (5.4) goes to zero. So condition (5.2) holds.

Now, if $k_n = o\left(\frac{n}{log(n)}\right)$, then for every n and $\omega \in \tilde{\Omega}_n$, the induced subset $(L_n)_\omega \in \mathcal{T}(k_n)$ such that for every compact set $V \subseteq \mathbb{R}^d$ we can write

$$max_{A\in(L_n)_{\omega_i}} diam(A \cap V) \to 0.$$

This implies condition (5.3) is satisfied and hence the theorem. $\square$

Remark 11 *Regression trees are consistent when the size of the tree grows as $o(\frac{n}{log(n)})$, where n is the number of training samples. It should be noted that the choice of important features based on RT is a greedy algorithm and the optimality of local choices of the best feature for a node doesn't guarantee that the constructed tree will be globally optimal (Kuncheva, 2004). It is also noted that with sufficiently large n, the optimal regression tree will not divide the regions of the feature space on which the regression function is constant. We further conclude that feature selection using the RT algorithm is justified, and the RT output will also play an important role in designing the regression model for increasing the predictive accuracy of the model. It should also be mentioned that incorporating RT output as an input feature in RBFN, the dimensionality gets increased. Thus the performance of the ANN model will be improved at a significant rate (Kohonen et al., 1988).*

The proposed hybrid model has two parts: extracting essential features from the feature space using RT algorithm and building one hidden layered ANN model with the important features extracted using RT along with RT output as another input vector in the RBFN model. In the second stage of the pipeline, the RBFN model is constructed with RT extracted features and OP (RT outputs) as another input feature in our model. The dimension of the input layer in the RBFN model, denoted by d_m $(< p)$ equals the number of important features obtained by RT + 1. Since the RBFN model consists of strictly one hidden layer, thus the proposed model is easily interpretable and fast in implementation. Our next objective is to discuss the sufficient condition for the universal consistency of the RBFN model. After incorporating RT output in the feature space, we have n training sequence $\xi_n = \{(Z_1, Y_1), ..., (Z_n, Y_n)\}$ of n i.i.d copies of $(\underline{Z}, \underline{Y})$ taking values from $\mathbb{R}^{d_m} \times [-K, K]$. A regression estimate realized by a one-hidden layered neural network is chosen to minimize the

$$\text{empirical } L_2\text{-risk } = \frac{1}{n}\sum_{j=1}^{n} \mid f(z_j) - Y_j \mid^2 .$$

RBFN is a family of ANNs, consisting of only a single hidden layer and uses a nonlinear function called radial basis function as an activation function, unlike feed forward neural network (Györfi et al., 2002). Gaussian functions, most frequently used in this layer, can be defined as follows:

$$\phi_i(z_i) = \phi\big(\parallel z_i - c_i \parallel; \sigma_i\big) = exp\bigg(- \frac{\parallel z_i - c_i \parallel^2}{2\sigma_i^2}\bigg)$$

where z_i is an input vector, ϕ_i is the output of i^{th} hidden neuron in the hidden layer with centers c_i and σ_i^2 as the variance. Here we can select the center vector as cluster centers of the input data. For practical use, the number of clusters is generally chosen to be much smaller than the number of data points resulting in RBFN of less complexity than other types of ANNs. Let us now consider a RBF network with one hidden layer having k nodes for a fixed Gaussian function given by the equation:

$$f(z_i) = \sum_{j=1}^{k} w_j \ \phi\big(\parallel z_i - c_j \parallel; \sigma_i\big) + w_0,$$

where $\sum_{j=0}^{k} |w_j| \leq b \ (> 0)$ and $c_1, c_2, ..., c_k \in \mathbb{R}^{d_m}$. The weights w_j and c_j are parameters of the RBFN and ϕ is the Gaussian radial basis function with $\phi(z) \to 0 \ as \ z \to \infty$. The next theorem gives regularity conditions for the universal consistency of RBFN model.

Theorem 15 *Consider a RBF network with Gaussian radial basis kernel having one hidden layer with k (> 1) nodes. If*

$$k \to \infty, \ b \to \infty \ \ and \ \ \frac{kb^4 log(kb^2)}{n} \to 0 \ \ as \ \ n \to \infty,$$

then RBFN model is said to be universally consistent for all distribution of $(\underline{Z}, \underline{Y})$.

Remark 12 *The idea of the proof is based on Krzyzak et al. (1996). It is worthwhile to note that Theorem 14 and 15 together give the regularity conditions for consistency of the proposed RBNT model.*

Proof Given the training set $\xi_n = \{(Z_1, Y_1), (Z_2, Y_2), ..., (Z_n, Y_n)\}$, our estimate of the regression function $m(z) = E[Y|Z = z]$ is an RBF network m_n which minimizes L_2 risk. Using the above formulation of RBFN model, for each n we fix Θ_n as the set of parameters, defined by

$$\Theta_n = \left\{ \theta = (w_0, w_1, ..., w_k, c_1, c_2, ..., c_k, \sigma_1, \sigma_2, ..., \sigma_k) : \sum_{j=0}^{k} |w_j| \leq b \right\}.$$

Our objective is to choose a regression estimator m_n from the class

$$F_{n,k} = \{f_\theta : \theta \in \Theta_n\} = \left\{ \sum_{j=1}^{k} w_j \phi(\parallel z_i - c_j \parallel) + w_0 : \sum_{j=0}^{k} |w_j| \leq b \right\}$$

with m_n satisfying

$$\frac{1}{n} \sum_{j=1}^{n} |m_n(Z_j) - Y_j|^2 = \min_{f \in F_{n,k}} \frac{1}{n} \sum_{j=1}^{n} |f(Z_j) - Y_j|^2.$$

Since we considered the Gaussian kernel function, it satisfies the properties of regular radial kernels, like non-negativity, monotonically decreasing, and left continuity. For universal consistency under the condition of the theorem,

$$E \int (m_n(z) - m(z))^2 \mu(dz) \to 0 \quad (n \to \infty)$$

for all distribution of (Z, Y). Here Y is bounded and μ be the probability measure of Z. Using Lemma 2 (Györfi et al., 2002), we write

$$E \int |m_n(z) - m(z)|^2 \mu(dz) \leq 2E \sup_{f \in F_{n,k}} \left| \frac{1}{n} \sum_{j=1}^{n} |f(Z_j) - Y_j|^2 - E|f(Z) - Y|^2 \right| + E \inf_{f \in F_{n,k}} \int |f(z) - m(z)|^2 \mu(dz).$$

The first term in the R.H.S. of the above inequality (the estimation error) can be handled by using non asymptotic uniform deviation inequalities and covering numbers. The second term of the above inequality (the approximation error) converges to zero when $k \to \infty$. Thus, we only need to show

$$\lim_{n \to \infty} E \left\{ \sup_{f \in F_{n,k}} \left| \frac{1}{n} \sum_{j=1}^{n} |f(Z_j) - Y_j|^2 - E|f(Z) - Y|^2 \right| \right\} = 0$$

for universal consistency of the model when the conditions of Theorem 15 are satisfied.

To prove this, we consider the functions $h(z,y) = (f(z)-y)^2, f \in F_{n,k}$ to be bounded by $h(z,y) \leq 4\max\{|f(z)|^2, |y|^2\} \leq 4\max\{b^2\phi^{*2}, K^2\} \leq 4b^2\phi^{*2}$, where ϕ is assumed to be bounded by ϕ^* and $|Y| \leq K$. Using Theorem 4 (Pollard, 1984), we obtain for arbitrary $\epsilon > 0$,

$$P\left\{ \sup_{f \in F_{n,k}} \left| \frac{1}{n} \sum_{j=1}^{n} |f(Z_j) - Y_j|^2 - E|f(Z) - Y|^2 \right| > \epsilon \right\} \leq 8E\left\{ N\left(\frac{\epsilon}{32\phi^* b}, F_{n,k}, \xi_n \right) \right\} e^{-\frac{n\epsilon^2}{128(4\phi^{*2}b^2)^2}} \tag{5.5}$$

where, N is the L_1 ϵ-covering number of F with respect to ξ_n.

Define the class of functions: $G = \left\{ \phi\left(\| z - c \|; \sigma\right) : c \in \mathbb{R}^{d_m} \right\}$. To bound the L_1 ϵ-covering number of G with respect to ξ_n, we use the VC dimension of graph sets of functions in G. We write

$$\begin{aligned}
N\left(\frac{\epsilon}{32\phi^* b}, F_{n,k}, \xi_n \right) &\leq \prod_{j=1}^{k} N\left(\frac{\epsilon}{32\phi^* b(k+1)}, \{w.g : g \in G, |w| \leq b\}, \xi_n \right) \\
&\quad \times N\left(\frac{\epsilon}{32\phi^* b(k+1)}, \{w : |w| \leq b\}, \xi_n \right) \\
&\leq \prod_{j=1}^{k} N\left(\frac{\epsilon}{64\phi^* b^2(k+1)}, G, \xi_n \right) N\left(\frac{\epsilon}{64\phi^{*2} b(k+1)}, \{w : |w| \leq b\}, \xi_n \right) \\
&\quad \times N\left(\frac{\epsilon}{32\phi^* b(k+1)}, \{w : |w| \leq b\}, \xi_n \right) \\
&\leq \prod_{j=1}^{k} \left(\frac{2b}{\frac{\epsilon}{64\phi^{*2} b(k+1)}} N\left(\frac{\epsilon}{64\phi^* b^2(k+1)}, G, \xi_n \right) \right) \left(\frac{2b}{\frac{\epsilon}{32\phi^* b(k+1)}} \right) \\
&\leq \left(\frac{128\phi^{*2} b^2(k+1)}{\epsilon} \right)^{k+1} \left(N\left(\frac{\epsilon}{64\phi^* b^2(k+1)}, G, \xi_n \right) \right)^{k} \\
&\leq \left(\frac{128\phi^{*2} b^2(k+1)}{\epsilon} \right)^{k+1} \left[3\left(\frac{6e\phi^*}{\frac{\epsilon}{64\phi^* b^2(k+1)}} \right)^{2\left(d_m^2 + d_m + 2\right)} \right]^{k} \\
&\leq \left(\frac{384e\phi^{*2} b^2(k+1)}{\epsilon} \right)^{2\left(d_m^2 + d_m + 2\right)k+1}
\end{aligned} \tag{5.6}$$

Collecting (5.5)-(5.6) together we get,

$$P\left\{\sup_{f\in F_{n,k}}\left|\frac{1}{n}\sum_{j=1}^{n}|f(Z_j)-Y_j|^2-E|f(Z)-Y|^2\right|>\epsilon\right\}$$
$$\leq 8\left(\frac{384e\phi^{*2}b^2(k+1)}{\epsilon}\right)^{2\left(d_m^2+d_m+2\right)k+1}\exp\left(-\frac{n\epsilon^2}{128\left(4\phi^{*2}b^2\right)^2}\right).$$

Let $\widetilde{Z}$ be a non-negative random variable and $\epsilon>0$, then

$$E\{\widetilde{Z}\}=\int_0^\infty P\{\widetilde{Z}>t\}dt\leq\epsilon+\int_0^\infty P\{\widetilde{Z}>t\}dt$$

Therefore, for any $\epsilon\in[0,\frac{1}{4})$,

$$E\Big\{\sup_{f\in F_{n,k}}\Big|\frac{1}{n}\sum_{j=1}^{n}|f(Z_j)-Y_j|^2-E|f(Z)-Y|^2\Big|\Big\}$$
$$\leq\epsilon+8\int_\epsilon^\infty\Big(\frac{384e\phi^{*2}b^2(k+1)}{t}\Big)^{2\left(d_m^2+d_m+2\right)k+1}\exp\Big(-\frac{nt^2}{2048\phi^{*4}b^4}\Big)dt$$
$$\leq\epsilon+8\Big(\frac{384e\Phi^{*2}b^2(k+1)}{\epsilon}\Big)^{2\left(d_m^2+d_m+2\right)k+1}\left(\frac{2048\phi^{*4}b^4}{n\epsilon}\exp\Big(-\frac{n\epsilon^2}{2048\phi^{*4}b^4}\Big)\right)$$
$$\leq\epsilon+\frac{8\times2048\phi^{*4}b^4(k+1)}{n\epsilon}\exp\left[\left(2\left(d_m^2+d_m+2\right)k+1\right)\log\Big(\frac{384e\phi^{*2}b^2(k+1)}{\epsilon}\Big)-\frac{n\epsilon^2}{2048\phi^{*4}b^4}\right]$$

Therefore,

$$E\Big\{\sup_{f\in F_{n,k}}\Big|\frac{1}{n}\sum_{j=1}^{n}|f(Z_j)-Y_j|^2-E|f(Z)-Y|^2\Big|\Big\}\longrightarrow\epsilon$$

if $\frac{kb^4\log(kb^2)}{n}\to0$ as $n\to\infty$ and $k,b\to\infty$. Since ϵ was arbitrary, the proof is complete. $\square$

Remark 13 *We can conclude that if the RBNT model satisfies the regularity conditions as stated in Theorem 14 and Theorem 15, then the proposed algorithm will be universally consistent. This is a fundamental property of any model for its robustness and general use. But computationally choosing parameters of the RBFN by minimizing empirical L_2 risk will be very costly. In practice, parameters of the RBFN model are learned by a gradient descent algorithm as mentioned below.*

5.3.2 Optimization of Model Parameters

Here we are going to discuss about the tuning parameters of the RBNT model. In the first stage of the pipeline model, "minsplit" function is chosen as 10% of the training data set which is recommended as the stopping rule in the RT algorithm. Further, we use RT suggested features along with RT output as an additional feature in the input space of the RBFN. RBFN is trained using a linear combination of Gaussian basis functions. Therefore we need to use an optimization algorithm for empirical error (to be denoted by E_e in the rest of the paper) minimization on ξ_n (Poggio and Girosi, 1990). Three important parameters to be optimized while training the RBF network are: centers (c_i), standard deviation (σ_i) and weights (w_j) of each neuron. We use a gradient descent algorithm over E to perform the optimization task in the RBFN model (Karayiannis, 1999), as follows:

$$\Delta c_i = -\rho_c \nabla_{c_i} E_e,$$

$$\Delta \sigma_i = -\rho_\sigma \frac{\partial E_e}{\partial \sigma_i},$$

$$\Delta w_j = -\rho_w \frac{\partial E_e}{\partial w_j},$$

where ρ_c, ρ_σ and ρ_w are small positive constants. Using this the parameters of the Gaussian basis function will be optimized. Generalization error is usually estimated by cross-validation method, and the optimum value of k would be found by trial and error.

5.4 Application to Waste Recovery Problem

A modern paper manufacturing company is facing a problem of fiber and filler losses which account for a substantial monetary loss for the company. The objective of this work is to improve the waste recovery process by improving the percentage of recovery of the FFRE which will reduce the fiber and filler losses. The inlet to FFRE is the waste (lean backwater combined with some valuable materials) coming out of the paper machine which undergoes froth flotation when treated with chemical and air (Aldrich et al., 2010; Marques and Tenório, 2000). The chemical helps to form the flocks from the fines present in the lean backwater and air helps to form the cake over the FFRE for recovery. The cake is then scooped out, collected and sent back to the paper machine process stream. The remaining water is sent to a water level tank for further purification.

FFRE removes solids using air flotation and sedimentation processes. Turbulence caused by water movement is an important factor in flotation and greatly reduces the efficiency of the other types of flotation units. Conventionally there must always be water movement for the water to flow from inlet to outlet. In FFRE, the inlet and outlet of the process are not stationary, and it rotates about the center. The rotation is synchronized so that the water in the water level tank achieves zero velocity during flotation. In practical terms, this allows better clarification in smaller surface areas and a much shallower tank. The processing time of water from the inlet to the outlet takes around 2-3 minutes. Air is dissolved into the water using air dissolving tube (ADT) of FFRE (Krofta, December 8, 1998), and unclarified water is released through a valve. The water flows in at the exact center, through a rotary joint. Coarse air is discharged through a vent pipe in the duct. The flow is directed to eliminate turbulence. Since the inlet distribution is moving forward at the same speed with that of the water is flowing out, the water stays in one spot in the tank without any movement during flotation. The floated material is recovered from the top surface through the FFRE spiral scoop. The scoop is designed to remove the floated material at the highest possible consistency, with a minimum surface disturbance. The level of water determines the consistency of the floated material removed. The flotation system in FFRE removes the solid content in the water by floating them to the surface for removal. Chemicals are used to increase the clarifier's efficiency by flocking out small and colloidal particles, that otherwise would not float or settle in the clarifier. The schematic diagram of the FFRE process is shown in Figure 5.2.

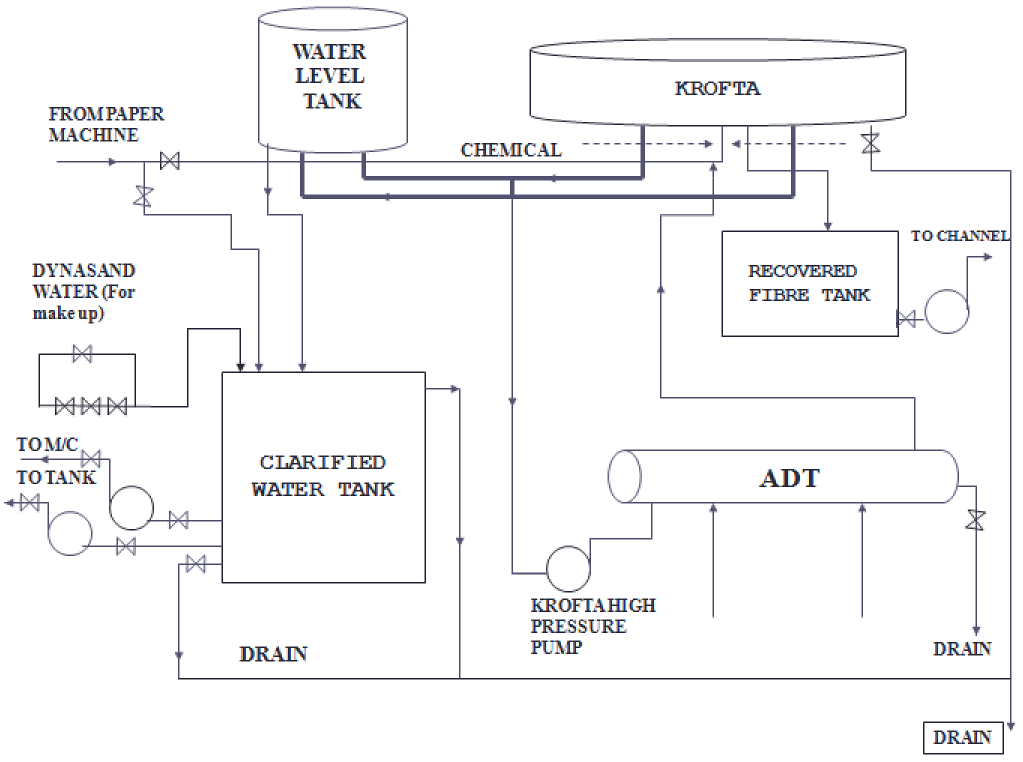


Figure 5.2: Schematic diagram of fiber and filler recovery process

5.4.1 Data Collection Plan

Initially, the efficiency of Krofta performance was measured and found to be not satisfactory. The efficiency of Krofta performance is quantified in terms of Inlet parts per million (Inlet PPM) and Outlet PPM, viz.,

$$\text{Efficiency} = \frac{(\text{Inlet PPM} - \text{Outlet PPM}) * 100}{\text{Inlet PPM}}.$$

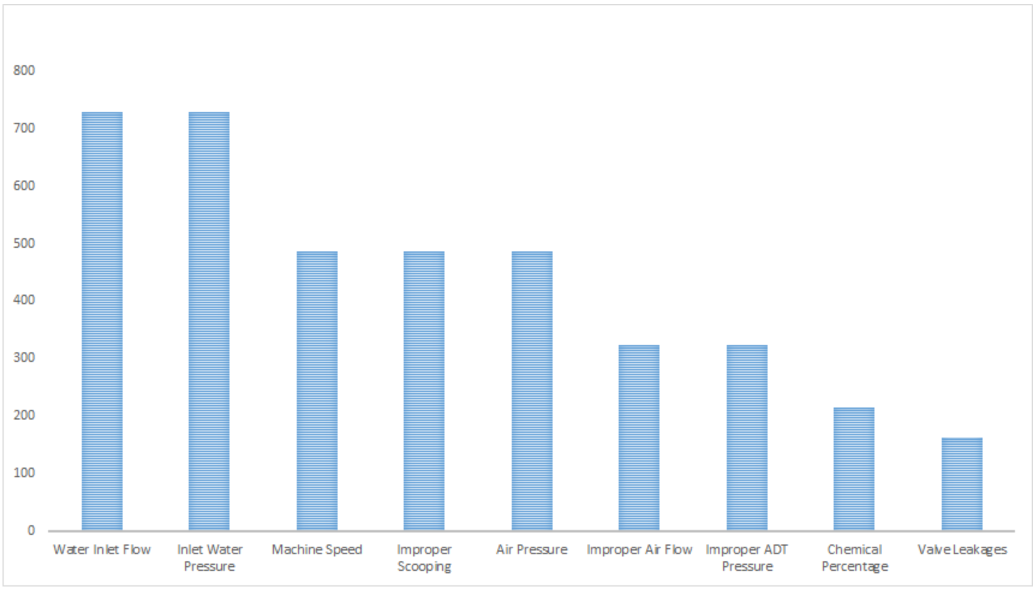


Figure 5.3: Summary of the FMEA

To identify the important parameters affecting the Krofta efficiency, a failure mode and effect analysis (FMEA) was performed with the help of process experts. FMEA is a useful analytical tool used in many process industries for understanding the potential areas of failures, so that appropriate measures can be taken *apriori* in order to avoid the occurrence of failure (Arvanitoyannis and Varzakas, 2009). While carrying out FMEA, potential failure modes are identified and potential effects of those failures on the system are also noted. Severity (S) of these potential failures is subjectively estimated by the experts on a 1-10 scale, 10 being given for the highest possible severity and 1 for the lowest possible severity. The potential causes for such failures are identified next along with each of its estimated occurrence (O) rating, which is also on a 1-10 scale. The higher possible occurrence will fetch a higher rating. Later detectability (D) rating for such potential failure causes are estimated on a 1-10 scale where the least possible detectability fetches a higher rating like 9 or 10 and higher detectability brings a lower value for the rating. The combined impact of all such ratings is represented by risk priority number (RPN), where $RPN = S \times O \times D$. Higher the value of RPN, more is the concern for that particular failure mode.

In the present case of the Krofta efficiency improvement problem, Figure 5.3 depicts some critical process parameters in the X-axis and their corresponding RPN value on the Y-axis. Only a few of these parameters are controllable. Thus, the controllable essential parameters were taken up for further investigation after a brainstorming session with the process experts. The data were collected for the paper tissues, taking Krofta efficiency percentage as the response and other parameters as the potential causal variables. The details of these parameters are given in Table 5.1.

5.4.2 Data

The data set comprises of 300 observations that are collected for a year from the process on the following causal variables: Inlet Flow, Water Pressure (water inlet pressure to ADT), Air Pressure, Pressure of Air-Left, Pressure of Air-Right, Pressure of ADT-D Left, Pressure of ADT-D Right and Amount of chemical lubricants. The response variable (FFRE recovery percentage) lies between 20 to 100. The sample data set for the paper tissue is presented in Table 5.1. This data set will be used for finding crucial process parameters and also finding a prediction model that can help the company to forecast future recovery percentage of FFRE.

Table 5.1: Sample process data set

Sl. No.	Inlet Flow	Water Pressure	Air Pressure	Air-Left	Air-Right	ADT-D Left	ADT-D Right	Amount of chemical	Recovery Percentage
1	1448	6.4	5.8	1.0	2.1	3.2	4.0	2.0	96.80
2	1794	5.2	5.6	2.4	1.6	3.6	4.0	3.0	97.47
3	2995	6.0	6.0	1.5	4.5	4.0	4.8	4.0	28.87
4	1139	6.5	6.0	1.2	1.7	3.0	4.6	2.0	33.05
5	2899	6.2	5.7	2.0	1.2	3.1	4.0	2.0	97.91
6	1472	6.6	6.8	3.7	3.1	5.2	4.8	4.0	57.77
7	1703	6.2	6.0	2.9	1.0	3.0	4.2	2.0	26.94
8	1514	5.5	5.0	2.0	2.1	3.8	4.7	2.0	67.01
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

5.4.3 Analysis of Results

Here we elaborate on the experimental evaluation of the proposed RBNT model while applied to the waste recovery process (WRP) data set. We first shuffled the observations of the WRP data set randomly and split them into training and testing data in a ratio of 60 : 40. Each experiment is repeated ten times with different randomly assigned training and test sets (10-fold cross-validation). We have considered here six well-known machine learning models for comparison purposes. These algorithms have been executed with their default parameters. The performance of each model was evaluated using several statistical measures.

The FFRE efficiency corresponding to a tissue depends on several process features such as Water Pressure, Air Pressure, Inlet Flow, AIR-Left, AIR-Right, amount of chemical, ADT-D Left and ADT-D Right. Our task is to select the relevant features out of the above and apply the model to improve the recovery percentage. We first apply the RT method to choose the important FFRE features and then compute the RBFN model with the selected important features along with RT output, as shown in Figure 5.1. RT is trained using *rpart* package implementation in R that uses the Gini index, and C_p for selection of variables to enter and leave the tree structure. To implement neural nets, we first normalize the data sets and run the ANN model. A transfer function ("logsig") is applied to bring back the original form at the end of modeling. We implemented the ANN model with different combinations of hidden layers without employing any other feature selection algorithm using *neuralnet* package. Due to the fact that the data set is medium-sized, thus going beyond two hidden layered (2HL) neural net will over-fit the data set. The number of neurons in the 1st hidden layer is chosen as 2/3 the size of the input layer and the number of neurons in the 2nd hidden layer is chosen as 1/3 the size of the input layer. RBFN with Gaussian radial basis kernel was trained using *RSNNS* package and the maximum number of iterations in all these NN implementations is chosen as 100. Execution time for the RBFN model is much less than ANN but higher than RT. Further, we compare our proposed model with different regressions models, viz. SVR, Bayesian additive regression tree (BART), and Tsai's neural tree. BART was implemented on the original data set using *BART* package implementation in R statistical software with the number of trees in the sum model chosen to be 50. And SVR was implemented with the built-in package *e1071* in R with the Gaussian kernel function. Tsai's neural tree is a hybrid model based on RT and RBFN used for water stage forecasting and has been used for comparison purposes (Tsai et al., 2012). It was found that Tsai's neural tree performs better than single regression models used in this study.

To apply the proposed RBNT model to the waste recovery data set, we first use RT with "minsplit" as 10% of the training sample size using *rpart* package in R. The important features are extracted and RT outputs are recorded. Further, we build an RBFN model using *RSNNS* package with the Gaussian kernel activation function and gradient descent algorithm for parameter optimization. Each time before applying neural networks, we have normalized the data sets. The essential features for the FFRE recovery data set using our proposed RBNT model are: Water Pressure, Air Pressure, Inlet Flow, ADT-D Left, and ADT-D Right. Table 5.2 exhibits the quantitative measures corresponding to MAE, RMSE, MAPE, R^2 and Adj(R^2), respectively, for the data set and for all the competing methods. A method is said to perform better than the other comparative algorithms if it minimizes the MAE, RMSE, MAPE values, and maximize the R^2 and Adj(R^2) values between the predictions relative to the observations. Table 5.2 exhibits the results based on 10 fold cross-validations and we reported the mean values of respective measures along with standard deviations. It can be observed from Table 5.2 that the MAE, RMSE and MAPE values obtained through the proposed RBNT model are always smaller than the values obtained from all other competing methods over the WRP data set. Tsai Neural tree provides the second best performance and performs better than the remaining single models in terms of having low RMSE, MAE, and MAPE values. The proposed model has smaller standard deviations with respect to different performance measures under 10 fold cross-validation. The model can be used for forecasting the future values of FFRE efficiency for the paper manufacturing company.

Table 5.2: Quantitative measures of performance for different regression models. Results are based on 10-fold cross validations. Mean values of the respective measures are reported with standard deviation within the bracket.

Regression Models	MAE	RMSE	MAPE	R^2	Adj(R^2)
RT	11.691 (0.45)	16.927 (0.89)	29.010 (1.02)	59.028 (3.25)	55.304 (1.95)
ANN	12.334 (0.25)	17.073 (0.56)	27.564 (1.85)	58.310 (2.98)	54.529 (2.08)
SVR	12.460 (0.28)	20.362 (1.23)	40.010 (1.81)	40.174 (2.05)	35.325 (2.64)
BART	12.892 (0.59)	16.010 (1.25)	30.038 (1.95)	59.380 (2.50)	56.458 (1.75)
RBFN	13.926 (2.50)	18.757 (3.25)	32.48 (3.45)	49.689 (5.45)	46.335 (3.95)
Tsai Neural tree	10.895 (0.78)	16.012 (0.50)	24.021 (1.85)	65.120 (2.89)	62.946 (1.78)
RBNT	**9.226** (0.35)	**14.331** (0.82)	**20.187** (1.45)	**70.632** (2.00)	**68.675** (2.13)

Based on the model, we further created an experimental design to obtain the optimal level of the tuning parameters. Final recommendations based on the results of the design of experiments were implemented in the process to monitor the Krofta efficiency. However, we have discussed only the proposed model and its accuracy level compared to other relevant state-of-the-art models. Our model helped the manufacturing process industry to achieve an efficiency level of about 80% from the current level of approximately 60% to improve the Krofta supracell recovery percentage.

5.5 Application to Simulated Data

We investigate the asymptotic behavior of the proposed RBNT model on an artificial data set created by sampling inputs $\underline{x}$ uniformly from the p-dimensional hypercube $[0,1]^p$ and computing outputs y as

$$y(x) = \sum_{j=1}^{p} \sin\left(20x^{(j)} - 10\right) + \varepsilon,$$

where ε is a zero mean Gaussian noise with variance σ^2, which corrupts the deterministic signal. We choose $p = 2$ and $\sigma = 0.01$, and investigate the asymptotic behavior as the number of training samples increases. Figure 5.4 illustrates the

RMSE for an increasing number of training samples and shows that the RBNT model error decreases much faster than other competitive model errors as sample size increases.

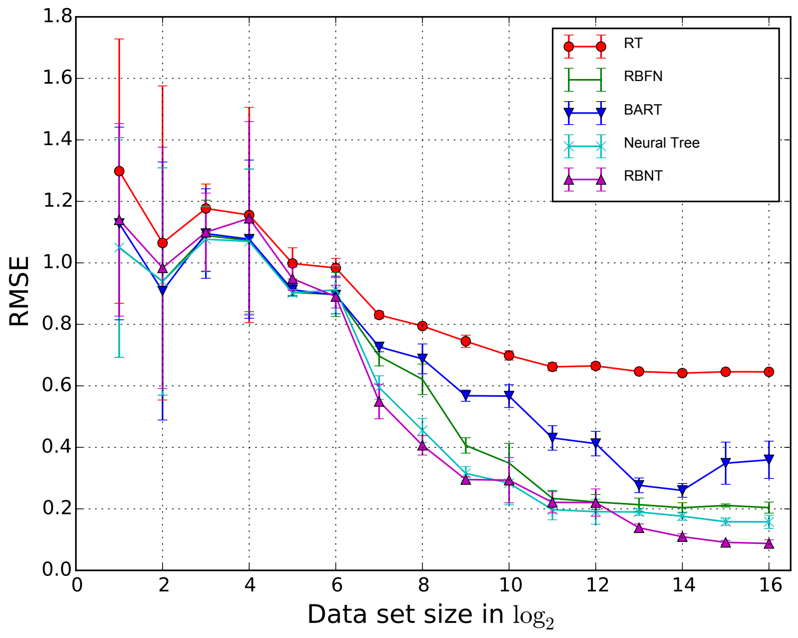


Figure 5.4: This figure shows the test RMSE for synthetic data with exponentially increasing training set size (x-axis). Solid lines connect the mean RMSE values obtained across 3 randomly drawn data sets for each data set size, whereas error bars show the empirical standard deviation.

5.6 Conclusions and Discussion

In this chapter, we proposed a novel distribution-free hybrid regression model which is a hybridization of RT and RBFN model for improving the waste recovery process in a paper company. The proposed model demonstrated the best performance and offered a practical solution to the WRP problem of finding crucial process parameters to improve the recovery percentage of fiber filler recovery equipment. The model can also be useful for future recovery prediction of the process that may give the company an indication of the stability of the process. Using the recommendation of the model, we further designed a set of experiments to find the optimal level of essential process parameters causing the FFRE efficiency. This resulted in improved waste recovery from an existing average recovery percentage of 60% to 80% which added financial benefit to the paper company. Our proposed model takes into account the curse of high-dimensional data and performs well on small or medium-sized data sets. We have shown our proposed methodology performed quite well through computational experiments compared to the other state-of-the-arts in the tissue paper data set. The proposed RBNT model has the most desired statistical property, viz. universal consistency. The usefulness and effectiveness of the model lie in its robustness and easy interpretability as compared to complex "black-box-like" models. Though the hybrid RBNT model was primarily developed for modeling the FFRE efficiency improvement problem, the model can also be used for other regression problems. The asymptotic behavior of the proposed RBNT model is also evaluated using an artificial data set.

Our proposed model is robust, universally consistent, easily interpretable, and highly useful for high dimensional small or medium-sized data sets to perform feature selection cum regression estimation tasks. Advanced neural net models (say, deep neural net) are highly complex, over-parameterized models, and found useful when the data sets are enormous (like image, audio, and video data sets). Nevertheless, no model can have a dominant advantage and one may also refer to no free lunch theorems (Wolpert and Macready, 1997). Typically, there will always be a trade-off between accuracy, interpretability, and complexity of the model for every new finding. An immediate future work of this chapter is to investigate the applicability of the proposed model on spatio-temporal and survival data sets involving regression tasks. In the next chapter, we look at the regression estimation problem from a Bayesian point of view. Chapter 6 attempts to blend frequentist and Bayesian paradigm in a hybridization framework.

Chapter 6

Bayesian Neural Tree Models for Nonparametric Regression

Related Publications:

1. Chakraborty, T., Chakraborty, A. K., Mansoor, Z. (2019). A hybrid regression model for water quality prediction. *Opsearch*, 56(4), 1167-1178.

2. Chakraborty, T., Kamat, G., Chakraborty, A. K. (2019). Bayesian Neural Tree Models for Nonparametric Regression. *ArXiv preprint arXiv:1909.00515*, (Under Review).

Summary

Frequentist and Bayesian methods differ in many aspects but share some basic optimal properties. In real-life prediction problems, situations exist in which a model based on one of the above paradigms is preferable depending on some subjective criteria. Nonparametric classification and regression techniques, such as decision trees and neural networks, have both frequentist (classification and regression trees (CART) and artificial neural networks) as well as Bayesian (Bayesian CART and Bayesian neural networks) approach to learning from data. In this chapter, we present two hybrid models combining the Bayesian and frequentist versions of CART and neural networks, which we call the Bayesian neural tree (BNT) models. BNT model can simultaneously perform feature selection and prediction, are highly flexible, and generalize well in settings with limited training observations. We study the statistical consistency of the proposed approach and derive the optimal value of a vital model parameter. The newly proposed BNT models are applied to solve a practical problem of water quality prediction in a paper manufacturing company. We also provide some illustrative examples using a wide variety of standard regression data sets from UCI repository to show the superiority of the proposed models in comparison to popularly used Bayesian CART and Bayesian neural network models.

6.1 Introduction

Methodologies in nonparametric regression employ either a frequentist or a Bayesian approach to learning from data. The choice between these two paradigms is often philosophical and based on subjective judgments. Two models, namely decision trees and neural networks, have primarily been used in the frequentist setting, but have robust Bayesian counterparts. Classification and regression trees (CART) were introduced by Breiman et al. (1984) for flexibly modeling the conditional distribution of an outcome variable given the predictors. For a data set, a tree is grown by sequentially splitting its internal nodes, and then pruning the grown tree back to avoid overfitting (Loh, 2011). The splitting rule for each node is based on minimizing the mean squared error (MSE) in regression and Gini index in classification. The Bayesian approach for finding a 'good' tree model entails specification of a prior distribution and stochastic search (Chipman et al., 1998, 2002). The fundamental idea behind Bayesian CART (BCART) is to have the prior induce a posterior distribution that can guide a (posterior) stochastic search towards a promising tree model (Chipman et al., 2002).

On the other hand, an artificial neural network (ANN) is an interconnected gathering of artificial neurons organized in layers (Hornik et al., 1989). A standard ANN model has three layers of nodes, namely input, hidden, and output layers, where nodes are neurons that use a nonlinear activation function (except for the input nodes). A backpropagation gradient descent algorithm is used to compare the network outputs with the actual outputs (Rumelhardt et al., 1986). If an error exists, it is backpropagated through the network, and the weights in the network architecture are adjusted accordingly (LeCun et al., 2015). An ANN, however, is often prone to overfitting when the data comprise a limited number of observations. A Bayesian treatment to an ANN offers a practical solution to this problem by naturally allowing for regularization (MacKay, 1992a; Neal, 1996). A Bayesian neural network (BNN) can also deal with the issue of model complexity, e.g., by selecting the number of hidden neurons in the model. In particular, a BNN treats the network weights to be random and obtains a posterior distribution over them (Barber and Bishop, 1998; Kendall and Gal, 2017). Neal (1996) introduced advanced Bayesian simulation methods, specially the hybrid Monte Carlo method, into the analysis of neural networks.

Although CART, BCART, ANN, and BNN individually perform well, they exhibit certain drawbacks. Tree-based models may overfit the training data, or stick to a local minimum in the decision boundaries. Additionally, the training of neural networks suffers considerably in a limited-data setup. Thus, a hybrid (or ensemble) formulation of trees and neural networks can be used to leverage their strengths and overcome their limitations. Several such hybrid models blending CART and ANNs have been discussed in the literature (Chakraborty et al., 2019c, 2020b; Micheloni et al., 2012; Sethi, 1990; Sirat and Nadal, 1990; Utgoff, 1989;

Vanli et al., 2019), and have been useful for improving the prediction accuracy of the individual models. These hybrid models, however, only consider frequentist implementations of their components. Some other works have explored hybrid frequentist-Bayesian models in the context of parametric inference, hypothesis testing, and other inferential problems (Bayarri and Berger, 2004; Bickel, 2015; Yuan, 2009). However, we are not aware of any hybrid algorithms blending frequentist and Bayesian methods for nonparametric regression.

Motivated by this, we propose a hybrid approach, called the Bayesian neural tree (BNT) model, for feature-selection-cum-prediction purposes (Chakraborty et al., 2019b,d). BNT model utilizes the built-in feature selection mechanisms of tree-based models (CART and BCART), along with the accuracy and flexibility of neural net (ANN and BNN), particularly in limited-data-size settings. The proposal can overcome the deficiencies of the component models, have a lesser number of tuning parameters, and are easily interpretable. On the theoretical side, we prove the statistical consistency of the models, which gives a theoretical guarantee of their robustness. We apply the proposed hybrid frameworks to solve a specific problem faced by a modern paper manufacturing company. Boiler inlet water quality is a significant concern for the paper machine. If the water treatment plant can not produce water of desired quality, it results in poor health of the boiler water tube and, consequently, affects the paper's quality. This variation is due to several crucial process parameters. We use the newly developed BNT models for boiler water quality prediction and show its excellent performance compared to other state-of-the-art. Finally, we also explore the performance of the BNT models using various standard regression data sets.

6.2 Formulation of the BNT Models

We begin by establishing notations. We assume that models are trained on n observations, and that there are d predictor variables. For data point i, where $1 \leq i \leq n$, let Y_i denote the response variable, $\overline{Y}_i$ denote its mean value, and $\hat{Y}_i$ denote the final prediction obtained from a model. Let $X_i = (X_{i1}, \ldots, X_{id})'$ denote the input vector for the i^{th} data point, where $1 \leq i \leq n$. We denote the training data as $L_n = \{Y_i, X_i\}_{i=1}^{n}$. In what follows, we omit the subscript i for simplicity of notation.

We now describe the working principles of the proposed BNT model. We present two variants of BNT models where each consists of a Bayesian (frequentist) implementation of a tree-based component for feature selection purposes, and a frequentist (Bayesian) implementation of a neural network component for prediction purposes (see Figure 6.1). Such hybridization or blending trees and neural networks in entirely frequentist settings were first proposed and theoretically justified in Chakraborty et al. (2019a,c, 2020b). In this chapter, we extend our approaches proposed in Chapter 5 but consider frequentist as well as Bayesian versions of the component models. In theory, both BNT models are asymptotically consistent, as we prove in Section 6.3.

6.2.1 BNT-1 model

The BNT-1 model comprises of two stages. In the first stage, a classical CART model is fit to the data, taking all d predictors. The CART model implicitly selects a feature at each internal split (based on maximum reduction in the MSE). Thus, the features used to construct the CART model can be considered important features in the data. We record these features, as well as the predictions obtained from the CART model. In the second stage, we construct a BNN with one hidden layer, where the input variables are the selected features from CART plus the prediction results from stage one. We use a Gaussian prior for the network weights and also model the data likelihood to be Gaussian. The prior for the number of hidden neurons (k) is taken to be a Geometric distribution with probability of success p. This is made possible by introducing a Markov Chain Monte Carlo (MCMC) posterior simulation scheme using reversible jump (Green, 1995) steps to move between different size architectures as in Rios Insua and Müller (1998). Since the BNN model can naturally be regularized through its implementation, it is less likely to overfit the data. The final set of predictions are obtained after fitting the BNN model to the data. A detailed discussion about the priors to be used and the MCMC algorithm for training the Bayesian model is described in Section 2.3.7 of Chapter 2.

Algorithm 6.1: BNT-1

```
  Input: $L_n = \{Y; X_1, \dots, X_d\}$
  Output: $\hat{Y}$
1 Fit a CART model to $L_n$ with a specified ‘minsplit’ value.
      • Record $S \subseteq \{X_1, \dots, X_d\}$, the set of selected features from CART.
      • Record $\hat{Y}_{cart}$, the predictions from CART.
      • Construct $S^{'} = \{S, \hat{Y}_{cart}\}$, the complete set of features for the BNN model.
2 Fit a BNN model with $k$ hidden neurons, where $k \sim$ Geometric $(p)$, and
    with input feature set $S^{'}$.
      • Record $\hat{Y}$, the final set of predictions from the BNN.
```

Thus, the proposed BNT-1 model utilizes the intrinsic feature selection ability of CART in the first stage, and trains a BNN model in the second stage using the selected features and predicted values from CART. This improves the accuracy of the individual models, as utilizing the CART output as a feature in the BNN adds non-redundant information. We present a formal workflow of the BNT-1 model below.

6.2.2 BNT-2 model

The BNT-2 model also follows a two-step pipeline. A BCART model is fitted to the data in the first stage, with the best fitting tree found via posterior stochastic search. For feature selection in the context of BCART, Bleich et al. (2014) illustrates three different schemes based on variable inclusion proportions or the proportion of times a predictor variable is used for a split within each posterior sample. The three schemes differ in thresholding the inclusion proportions: 'local', 'global max', and 'global SE' procedures. Any of the procedures can be utilized for feature selection based on the data and prediction problem at hand. In this chapter, we use the local thresholding procedure. A detailed discussion about the priors to be used and the MCMC algorithm for training the Bayesian model is described in Section 2.3.3 of Chapter 2.

Algorithm 6.2: BNT-2

```
Input: $L_n = \{Y; X_1, \dots, X_d\}$
Output: $\hat{Y}$
1 Fit a BCART model to the data via a posterior stochastic search over the
   possible tree models.
      • Record $S \subseteq \{X_1, \dots, X_d\}$, the set of selected features obtained using a
        thresholding procedure.
      • Record $\hat{Y}_{bcart}$, the prediction from BCART.
      • Construct $S^{'} = \{S, \hat{Y}_{bcart}\}$, the complete set of features for the ANN
        model. Denote the dimension of $S^{'}$ as $d_m$.
2 Fit a one-hidden-layer ANN model with input feature set $S^{'}$, and with
   number of hidden neurons $k = \sqrt{\frac{n}{d_m log(n)}}$.
      • Record $\hat{Y}$, the final set of predictions from the ANN.
```

Thus, we record the important features and predictions from BCART and use these as inputs to a one-hidden-layer ANN in stage two. One hidden layer in the ANN sufficed, due to incorporating the selected features and predicted outputs from BCART. Using a single hidden layer also reduces the model's overall complexity and the risk of overfitting in small and medium-sized data sets (Devroye et al., 1996). The optimal choice for the number of hidden neurons (k) for the ANN is derived under Proposition 2 in Section 6.3.2, and is given as $\sqrt{\frac{n}{d_m log(n)}}$, where d_m is the dimension of the input feature space of the ANN, and n is the

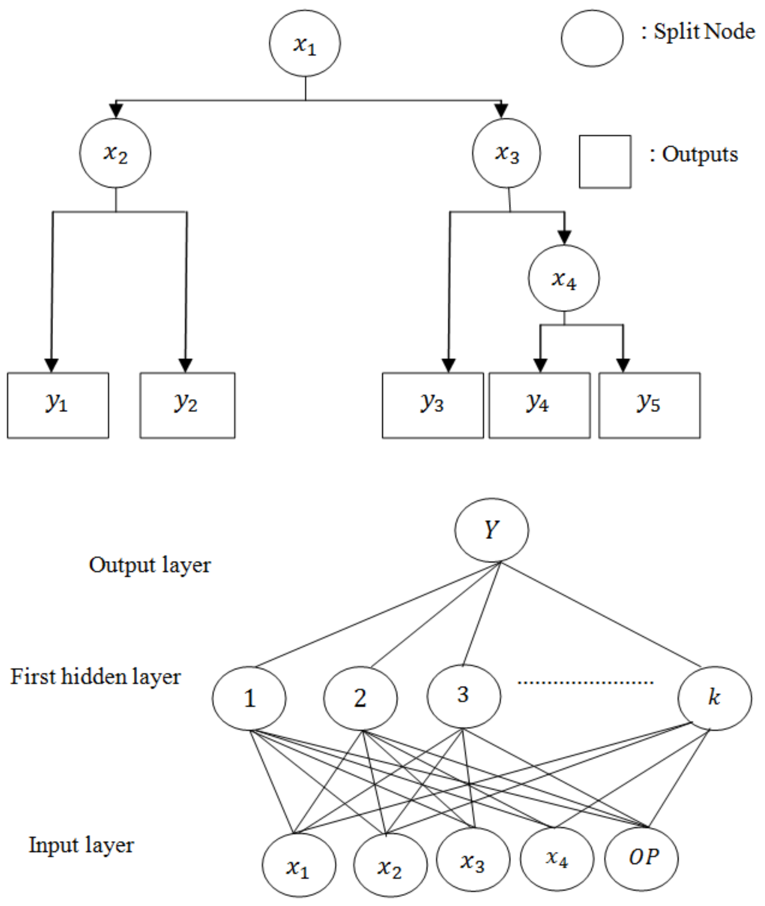


Figure 6.1: An overview of Bayesian neural tree models. A CART (BCART) model is at the top and its corresponding BNN (ANN) model at the bottom. *OP* denotes the tree (CART/BCART) output.

training sample size. The final set of predictions is obtained after fitting the ANN model to the data. The precise algorithm is as follows.

6.3 Statistical Properties of BNT Models

The results on the consistency of multivariate histogram-based regression estimates on data-dependent partitions (Lugosi and Nobel, 1996; Nobel, 1996) and that of regression estimates realized by an ANN (Devroye et al., 1996; Lugosi and Zeger, 1995), we know that under certain conditions, both the nonparametric models converge to the true density functions. In Bayesian settings, posterior concentration of the BCART model (Rocková and van der Pas, 2020), and posterior consistency of the BNN model (Lee, 2000, 2004) have been previously explored. We use these results to prove the theoretical consistency of the BNT models under certain conditions. We also find the optimal value of the number of hidden nodes in the BNT-2 model in Section 6.3.2.

6.3.1 Asymptotic Properties of the BNT-1 Model

Let $\mathbb{X} = (X_1, X_2, ..., X_d)$ be the space of all possible values of d features, and let $\mathbb{Y} = (Y_1, \ldots, Y_n)'$ be the response vector, where each Y_i takes values in $[-K, K]$, and $K \in \mathbb{R}$. A regression tree (RT) $f : \mathbb{R}^d \to \mathbb{R}$ is defined by assigning a number to each cell of a tree-structured partition. We seek to estimate a regression function $r(x) = E(Y|X = x) \in [-K, K]$ based on n training samples $L_n = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$. The regression function $r(x)$ minimizes the predictive risk $J(f) = E\big|f(X) - Y\big|^2$ over all functions $f : \mathbb{R}^d \to \mathbb{R}$. Practically, given the training data L_n, we can find an estimate $\hat{f}$ of f that minimizes the empirical risk

$$J_n(f) = \frac{1}{n}\sum_{i=1}^{n}\Big(f(X_i) - Y_i\Big)^2$$

over a suitable class of regression estimates, since the distribution of $(\mathbb{X}, \mathbb{Y})$ is not known *apriori*. We let $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ be a partition of the feature space $\mathbb{X}$ and denote $\widetilde{\Omega}$ as one such partition of Ω. Define $(L_n)_{\omega_i} = \{(X_i, Y_i) \in L_n : X_i \in \omega_i, Y_i \in [-K, K]\}$ to be the subset of L_n induced by ω_i and let $(L_n)_{\widetilde{\Omega}}$ denote the partition of L_n induced by $\widetilde{\Omega}$. Now define $\widehat{L}_n$ to be the space of all learning samples and $\mathbb{D}$ be the space of all partitioning regression functions. Then a binary partitioning rule $f : \widehat{L}_n \to \mathbb{D}$ is such that $f \in (\psi \circ \phi)(L_n)$, where ϕ maps L_n to some induced partition $(L_n)_{\widetilde{\Omega}}$ and ψ is an assigning rule which maps $(L_n)_{\widetilde{\Omega}}$ to a partitioning regression function $\hat{f}$ on the partition $\widetilde{\Omega}$. Consistent estimates of $r(\cdot)$ can be achieved using an empirically optimal regression tree if the size of the tree grows with n at a controlled rate.

Theorem 16 *Suppose* $(\mathbb{X}, \mathbb{Y})$ *is a random vector in* $\mathbb{R}^p \times [-K, K]$ *and* L_n *is the training set of* n *outcomes. Finally, for every* n *and* $w_i \in \widetilde{\Omega}_n$*, the induced subset* $(L_n)_{w_i}$ *contains at least* k_n *of the vectors of* $X_1, X_2, ..., X_n$*. Let* $\hat{f}$ *minimizes the empirical risk* $J_n(f)$ *over all* k_n *nodes of RT,* $f \in (\psi \circ \phi)(k_n)$*. If* $k_n \to \infty$ *and* $k_n = o\Big(\frac{n}{log(n)}\Big)$*, then* $P\big|\hat{f} - r\big|^2 \to 0$ *with probability 1.*

Proof For proof, refer to Theorem 14 of Chapter 5. □

The BNT-1 model essentially uses the feature selection mechanism of RT, and RT output also plays an important role in designing the ensemble model. We further build a one hidden layered BNN model using RT given features and RT output as another input feature in the BNN model. We denote the dimension of the input feature space of the BNN model in the ensemble as d_m $(\leq d)$. We further assume that these covariates are fixed and have been rescaled to $[0, 1]^{d_m} = \mathbb{C}^{d_m}$.Now, let the random variables $\mathbb{Z}$ and $\mathbb{Y}$ take their values from

$\mathbb{C}^{d_m}$ and $[-K,K]$ respectively. Denote the measure of $\mathbb{Z}$ over $\mathbb{C}^{d_m}$ by μ and $m : \mathbb{C}^{d_m} \to [-K,K]$ be a measurable function such that $m(Z)$ approximates Y. Given the training sequence $\xi_n = \{(Z_1,Y_1),(Z_2,Y_2),...,(Z_n,Y_n)\}$ of n i.i.d. copies of $(\mathbb{Z},\mathbb{Y})$, the parameters of the neural network regression function estimators are chosen such that it minimizes the

$$\text{empirical } L_2\text{-risk} = \frac{1}{n}\sum_{j=1}^{n} |f(Z_j) - Y_j|^2.$$

We have used logistic squasher as a sigmoidal function in BNN and treat the number of hidden nodes (k) as a parameter in the proposed Bayesian ensemble formulation. In usual Bayesian nonparametrics, the number of hidden nodes grows with the sample size, and thus we can use an arbitrarily large number of hidden nodes asymptotically. But we use the formulation by Rios Insua and Müller (1998) and treat the number of hidden nodes in the ensemble model as a parameter and show that the joint posterior becomes consistent under certain regularity conditions. We consider geometric prior for k, following Rios Insua and Müller (1998). This will give better uncertainty quantification by allowing the unconstrained size of the hidden nodes. The major advantage of using a Bayesian setting over the frequentist approach is that it allows one to use background knowledge to select a prior probability distribution for the model parameters. Also, the predictions of future observations are made by integrating the model's prediction for the posterior parameter distributions obtained by updating the prior by taking into account the data. We address this by properly defining the class of prior distribution for neural network parameters that reach sensible limits when the size of the networks goes to infinity and further implementing the Markov Chain Monte Carlo (MCMC) algorithm in the network structure (MacKay, 1992b). We define

$$E\big[Y_i|Z_i = z_i\big] = \beta_0 + \sum_{j=1}^{k} \beta_j \sigma(a_j^T z_i) + \epsilon_i, \tag{6.1}$$

where k is the number of hidden nodes, β_j's are the weights of these hidden nodes, a_j's are vectors of location and scale parameters, and $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0,\sigma^2)$. Expanding (6.1) in vector notation yields the following equation:

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j \sigma\left(a_{j0} + \sum_{h=1}^{d_m} a_{jh} z_h\right) + \epsilon_i, \tag{6.2}$$

where d_m is the number of input features. We consider the asymptotic properties of the neural network in the Bayesian setting. We show the consistency of the posterior for neural networks in Bayesian setting which along with Theorem 16 ensures the consistency of the proposed BNT-1 model.

Let $\lambda_i = P(k = i)$ be the prior probability that the number of hidden nodes is i, and of course $\sum_i \lambda_i = 1$. Also, Π_i be the prior for the parameters of the regression equation, given that $k = i$. We can then write the joint prior for all the parameters as $\sum \lambda_i \Pi_i$. Here we consider $\Pi_i \overset{ind}{\sim} \mathcal{N}(0, \tau^2)$ and the prior for k be geometric distribution. In the sequel, we also assume that

$$Y|Z = z \sim \mathcal{N}\left(\beta_0 + \sum_{j=1}^{k} \frac{\beta_j}{1 + \exp\left(-a_{j0} - \sum_{h=1}^{d_m} a_{jh} z_h\right)}, 1\right).$$

Let $f_0(z, y)$ be the true density. We can define a family of Hellinger neighborhoods as

$$H_\epsilon = \{f\ ;\ D_H(f, f_0) \leq \epsilon\},$$

with $D_H(f, f_0)$ as defined below:

$$D_H(f, f_0) = \sqrt{\int\int \left(\sqrt{f(z, y)} - \sqrt{f_0(z, y)}\right)^2 dzdy}.$$

Let F_n be the set of all neural networks with parameters $|a_{jh}| \leq C_n$ and $|\beta_j| \leq C_n$, where $j = 1, \ldots, k$ and $h = 1, \ldots, d_m$, and C_n grows with n such that $C_n \leq \exp\ (n^{(b-a)})$ for any constant b such that $0 < a < b < 1$ when $k \leq n^a$. The Kullback-Leibler divergence (not a distance metric) is defined as

$$D_K(f_0, f) = E_{f_0}\left[\log \frac{f_0(z, y)}{f(z, y)}\right].$$

For any $\gamma > 0$, we define Kullback-Leibler neighborhood by

$$K_\gamma = \{f\ : D_K(f_0, f) \leq \gamma\}.$$

We denote the prior for f by $\Pi_n(\cdot)$ and the posterior by $P\big(\cdot\ |(Z_1, Y_1), ..., (Z_n, Y_n)\big)$. We will present results on the asymptotic properties of the posterior distribution for the neural network model present in the ensemble BNT-1 model over Hellinger neighborhoods.

Theorem 17 *Assume that $\mathbb{Z}$ is uniformly distributed in $[0, 1]^{d_m}$, $\Pi_i \overset{ind}{\sim} \mathcal{N}(0, \tau^2)$, $k \sim$ Geometric, and the following conditions hold:*
(A1) For all i, we have $\lambda_i > 0$;
(A2) $B_n \uparrow n$, for all $r > 0$, there exists $q > 1$ and N such that $\sum_{i=B_n+1}^{\infty} \lambda_i < exp\big(-n^q r\big)$ for $n \geq N$;
(A3) There exists $r_i > 0, N_i$ such that $\Pi_n(F_n^c) < exp(-nr_i)$ for all $n \geq N_i$;
(A4) For all $\gamma, v > 0$, there exists I and M_i such that for any $i \geq I$, $\Pi_i(K_\gamma) \geq$

$exp(-nv)$ for all $n \geq M_i$.
Then for all $\epsilon > 0$, the posterior is asymptotically consistent for f_0 over Hellinger neighborhoods and $P\big(H_\epsilon \,|(Z_1, Y_1), ..., (Z_n, Y_n)\big) \to 1$ in probability.
In other words, the posterior probability of any Hellinger neighborhood of f_0 converges to 1 in probability.

Proof To prove the theorem, we first show that the regularity conditions hold when we assume a Geometric prior for k. And finally, show the posterior consistency by using conditions (A1)-(A4). Since we take geometric prior for k, it is obvious that $\lambda_i > 0$. Now,

$$\begin{aligned}\sum_{i=B_n+1}^{\infty} \lambda_i = P(k > B_n) \\ &= \sum_{i=B_n+1}^{\infty} p(1-p)^i = (1-p)^{B_n+1} \\ &= exp\big[(B_n+1)log(1-p)\big] \\ &= exp\left[-n^q\big(-log(1-p)\big)\right] \text{ (Using } B_n = O(n^q) \text{ for } q > 1) \\ &\leq exp\big(-n^q.r\big) \quad \text{for } r > 0 \text{ and sufficiently large } n. \qquad (6.3)\end{aligned}$$

We consider a geometric prior with parameter p. Also let, $B_n = O(n^q)$ for any $q > 1$. For any i, we write $i < n^a$ for $a > 0$ and sufficiently large n, where θ be the vector of all parameters (other than k):

$$\begin{aligned}\Pi_i\big(F_n^c\big) &= \int_{F_n^c} \Pi_i(\theta)d\theta \\ &\leq \sum_{i=1}^{d_n} 2\int_{C_n}^{\infty} \phi\left(\frac{\theta_i}{\tau_i}\right)d\theta_i \\ &\leq d_n\left[\frac{2\tau\phi\big(\frac{C_n}{\tau_i}\big)}{\frac{C_n}{\tau_i}}\right] \quad \text{by Mill's ratio} \\ &= d_n\left[\frac{2\tau_i^2}{C_n}.\frac{1}{\sqrt{2\Pi}}.exp\left(-\frac{C_n^2}{2\tau_i^2}\right)\right] \\ &\leq d_n\left(\tau_i^2\sqrt{\frac{2}{\Pi}}\right).exp\left(-n^{b-a} - \frac{1}{2\tau_i^2}e^{2n^{b-a}}\right) \; \big[\text{Taking } C_n = e^{n^{b-a}}, 0 < a < b < 1\big] \\ &= exp\left[-n^{b-a} + \log\left(d_n\tau_i^2\sqrt{\frac{2}{\Pi}}\right)\right].exp\left(-\frac{1}{2\tau_i^2}e^{2n^{b-a}}\right) \\ &\leq exp\left(-\frac{1}{2\tau_i^2}e^{2n^{b-a}}\right) \big[\text{Using } d_n = (p+2)n^a + 1 \leq (p+3)n^a \text{ for large } n\big] \\ &\leq e^{-nr_i}, \text{ where } e^{2n^{b-a}} > n \text{ for large } n \text{ and taking } r = \frac{1}{2\tau_i^2}. \qquad (6.4)\end{aligned}$$

We can write

$$G_n^c = \bigcup_{i=0}^{\infty} F_i^c,$$

where F_i is the set of all neural networks with i nodes and with all the parameters less than C_n in absolute value, $C_n \leq exp(n^b), 0 < b < 1$.

$$\Pi(G_n^c) = \sum_{i=0}^{\infty} \lambda_i \Pi_i(F_i^c) \leq \sum_{i=0}^{B_n} \lambda_i \Pi_i(F_i^c) + \sum_{i=B_n+1}^{\infty} \lambda_i = I_1 + I_2, \text{ say.}$$

To handle I_1 and I_2, we use (6.4) and (6.3):

$$\begin{aligned} I_1 &\leq \sum_{i=0}^{B_n} \lambda_i exp(-nr_i) \\ &\leq exp(-nr^*) \sum_{i=0}^{B_n} \lambda_i \ \big(\text{By letting } r^* = min\{r_0, r_1, ..., r_{B_n}\}\big) \\ &\leq exp(-nr^*). \end{aligned}$$

And, $I_2 \leq exp(-n^q r^*)$ for sufficiently large n. For large n, $q > 1$ and $r = r^*/2$, we have

$$\Pi(G_n^c) < exp(-nr).$$

Now, to show (A4) holds true, we consider the following:

$$\begin{aligned} \Pi_i(M_\delta) &= \prod_{i=1}^{\hat{d_n}} \int_{\theta_i-\delta}^{\theta_i+\delta} \frac{1}{\sqrt{2\Pi\tau^2}}.exp\left(-\frac{u^2}{2\tau^2}\right)du \\ &\geq \prod_{i=1}^{\hat{d_n}} 2\delta \inf_{u\in[\theta_i-1,\theta_i+1]} \frac{1}{\sqrt{2\Pi\tau^2}}.exp\left(-\frac{u^2}{2\tau^2}\right) \\ &= \prod_{i=1}^{\hat{d_n}} \delta\sqrt{\frac{2}{\Pi\tau^2}}.exp\left(-\frac{\xi_i}{2\tau^2}\right) \quad \big[\xi_i = max\{(\theta_i-1)^2, (\theta_i+1)^2\}\big] \\ &\geq \left(\delta\sqrt{\frac{2}{\Pi\tau^2}}\right)^{\hat{d_n}}.exp\left(-\frac{\hat{\xi}\hat{d_n}}{2\tau^2}\right) \quad \big[\hat{\xi} = \max_i\{\xi_1, \xi_2, \dots, \xi_{\hat{d_n}}\}\big] \\ &> e^{-nv} \ \big[\text{Using } \hat{d_n} \leq (p+3)n^a \text{ and for large } n \text{ and for any } v\big]. \end{aligned} \tag{6.5}$$

For any $\delta > 0$, let l be the number of hidden nodes required by the theorem for making g_0 continuous and square differentiable. Using (6.5) we write

$$\Pi(M_\delta) = \sum_{i=0}^{\infty} \lambda_i \Pi_i(M_\delta) \geq \lambda_l \Pi_l(M_\delta) \geq \lambda_l exp(-nv^*).$$

For sufficiently large n and for any v^*, l is a constant, thus λ_l does not depend on n and is positive for geometric prior. Thus, $\Pi(M_\delta) \geq \exp(-nv)$ for any sufficiently large n.
We can now use conditions (A1)-(A4) to show that

$$P\big(H_\epsilon\, |(Z_1, Y_1), ..., (Z_n, Y_n)\big) \to 1 \quad \text{in probability.}$$

Alternatively, $P\big(H_\epsilon^c\, |(Z_1, Y_1), ..., (Z_n, Y_n)\big) \to 0$ in probability. Now,

$$\begin{aligned} P\big(H_\epsilon^c\, |(Z_1, Y_1), ..., (Z_n, Y_n)\big) &= \frac{\int_{H_\epsilon^c} \prod_{i=1}^n f(z_i, y_i) d\Pi_n(f)}{\int \prod_{i=1}^n f(z_i, y_i) d\Pi_n(f)} \\ &= \frac{\int_{H_\epsilon^c} R_n(f) d\Pi_n(f)}{\int R_n(f) d\Pi_n(f)}, \text{ where } R_n(f) = \frac{\prod_{i=1}^n f(z_i, y_i)}{\prod_{i=1}^n f_0(z_i, y_i)} \\ &= \frac{D_1}{D_2}, \text{ say.} \end{aligned}$$

Using Wong and Shen (1995) and (A1-A4), we can find the supremum of the likelihood ratios $R_n(f)$. Thus, we have

$$D_1 < e^{\frac{-nt}{2}} + e^{-2c_2\epsilon^2}, \text{ where } t, c_2 > 0.$$

Using Lee (2000, Lemma 5), we have $D_2 > e^{-n\delta}$ for large n, except on a set with probability approaching to 0. Finally, we have

$$\begin{aligned} P\big(H_\epsilon^c\, |(Z_1, Y_1), ..., (Z_n, Y_n)\big) &< \frac{e^{\frac{-nt}{2}} + e^{-2c_2\epsilon^2}}{e^{-n\delta}} \\ &\leq e^{-n\epsilon^{'}} + e^{n\epsilon^2\epsilon^{'}}, \text{ where } \epsilon^{'} > 0 \\ &\longrightarrow 0 \text{ for sufficiently large } n. \end{aligned}$$

□

Remark 14 *Theorem 17 shows that the posterior is consistent when the number of hidden neurons of the neural network (with Bayesian setting) is a parameter that can be estimated from the data. Thus, we can let the data derive the number of hidden nodes in the model and emphasize on model selection during practical implementation.*

6.3.2 Asymptotic Properties of the BNT-2 model

We consider the nonparametric regression model

$$Y_i = f_0(X_i) + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} \mathcal{N}(0, 1),$$

where the output variable $\mathbb{Y} = (Y_1, Y_2, ..., Y_n)'$ is dependent on a set of d potential covariates $\mathbb{X} = (X_{i1}, X_{i2}, ..., X_{id})'$, $1 \leq i \leq n$. We further assume that these covariates are fixed and have been rescaled such that every $X_{ij} \in [0,1]^d = \mathbb{C}^d$, $1 \leq i \leq n$ and $1 \leq j \leq d$. The true unknown response surface $f_0(X_i)$ is assumed to be smooth.

Recent work by Rocková and van der Pas (2020) had shown that the BCART model achieves a near-minimax-rate optimal performance when approximating a single smooth function. Thus, the optimal behavior of a BCART model is guaranteed, and even under a suitably complex prior on the number of terminal nodes, a BCART model is reluctant to overfit. In the BNT-2 model, we build a BCART model in the first stage and perform variable (feature) selection as in Bleich et al. (2014), which ensures that we can obtain a consistent BCART model under the assumptions of Theorem 4.1 of Rocková and van der Pas (2020).

The selected important features, along with the BCART outputs, are trained using an ANN model with one hidden layer. We denote the dimension of the input feature space of this ANN model as d_m $(\leq d)$. The rescaled feature space is denoted by $\mathbb{C}^{d_m} = [0,1]^{d_m}$. Using one hidden layer in the ANN makes the BNT-2 model less complex and fastens its actual implementation. Moreover, there is no theoretical gain in considering more than one hidden layer in an ANN (Devroye et al., 1996). Below, we establish sufficient conditions for consistency of the BNT-2 model along with the optimal value of the number of hidden nodes k.

Let the rescaled set of features of the ANN be $\mathbb{Z}$. $\mathbb{Z}$ and $\mathbb{Y}$ take values from $\mathbb{C}^{d_m}$ and $[-K, K]$, respectively. We denote the measure of $\mathbb{Z}$ over $\mathbb{C}^{d_m}$ by μ and $m : \mathbb{C}^{d_m} \rightarrow [-K, K]$ be a measurable function that approximates $\mathbb{Y}$. Given the training sequence $(\mathbb{Z}, \mathbb{Y})$ of n i.i.d copies, the neural network hyperparameters are chosen by empirical risk minimization. We consider the class of neural networks having a logistic sigmoidal activation function in the hidden layer and k hidden neurons, with bounded output weights.

$$F_{n,k} = \left\{ \sum_{i=1}^{k} c_i \sigma(a_i^T z + b_i) + c_0 : k \in \mathbb{N}, a_i \in \mathbb{R}^{d_m}, b_i, c_i \in \mathbb{R}, \sum_{i=0}^{k} |c_i| \leq \beta_n \right\},$$

and obtain $m_n \in F_{n,k}$ satisfying

$$\frac{1}{n} \sum_{i=1}^{n} |m_n(Z_i) - Y_i|^2 \leq \frac{1}{n} \sum_{i=1}^{n} |f(Z_i) - Y_i|^2, \text{ if } f \in F_{n,k},$$

where, m_n is a function that minimizes the empirical L_2-risk in $F_{n,k}$. The theorem below, due to Lugosi and Zeger (1995, Theorem 3), states the sufficient conditions for the consistency of the neural network.

Theorem 18 *Consider an ANN with a logistic sigmoidal activation function having one hidden layer with k (> 1) hidden nodes. If k and β_n are chosen to satisfy*

$$k \to \infty, \ \beta_n \to \infty, \ \frac{k\beta_n^4 log(k\beta_n^2)}{n} \to 0$$

as $n \to \infty$, then the model is said to be consistent for all distributions of $(\mathbb{Z}, \mathbb{Y})$ with $\mathbb{E}|\mathbb{Y}|^2 < \infty$.

Proof For the proof, one may refer to Györfi et al. (2002, Chapter 16). □

Now, we obtain an upper bound on k using the rate of convergence of a neural network with bounded output weights. In what follows, we have assumed that m is Lipschitz (δ, C)-smooth according to the following definition:

Definition 12 *A function $m : C^{d_m} \to [-K, K]$ is called Lipschitz (δ, C)-smooth if it satisfies the following inequality:*

$$|m(z^{'}) - m(z)| \leq C\|z^{'} - z\|^{\delta}$$

for all $\delta \in [0, 1]$, $z^{'}, z \in \mathbb{C}^{d_m}$, and $C \in \mathbb{R}_+$.

Proposition 2 *Assume that $\mathbb{Z}$ is uniformly distributed in $\mathbb{C}^{d_m}$ and $\mathbb{Y}$ is bounded a.s. and m is Lipschitz (δ, c)-smooth. Under the assumptions of Theorem 18 with fixed d_m, and $m, f \in F_{n,k}$, also f satisfying $\int_{C^{d_m}} f^2(z)\mu(dz) < \infty$, we have $k = O\left(\sqrt{\frac{n}{d_m log(n)}}\right)$.*

Proof To prove Proposition 2, we use results from statistical learning theory of neural networks (Györfi et al., 2002, Chapter 12). We use the complexity regularization principle to choose the parameter k in a data-dependent manner (Hamers and Kohler, 2003; Kohler, 2006; Kohler and Krzyżak, 2005). Consistency results presented in Theorem 18 state that

$$\mathbb{E}\int_{C^{d_m}} (m_n(Z) - m(Z))^2 \mu(dz) \to 0 \quad \text{as} \quad n \to \infty.$$

We can write using Lemma 2 (Györfi et al., 2002) that

$$\mathbb{E}\left[\int_{C^{d_m}} \left|m_n(Z) - m(Z)\right|^2 \mu(\mathrm{dz})\right] \leq 2\mathbb{E}\left[\sup_{f\in F_{n,k}} \left|\frac{1}{n}\sum_{i=1}^{n} \left|Y_i - f(Z_i)\right|^2 - \mathbb{E}\left|Y - f(Z)\right|^2\right|\right] + \mathbb{E}\left[\inf_{f\in F_{n,k}} \int_{C^{d_m}} \left|f(Z) - m(Z)\right|^2 \mu(\mathrm{dz})\right] \quad (6.6)$$

where μ denotes the distribution of $\mathbb{Z}$. For the consistency of the neural network model, the *estimation error* (first term in the R.H.S. of 6.6) and the *approximation* *error* (second term in the R.H.S. of 6.6) should tend to 0. To find the bound for k, we apply non-asymptotic uniform deviation inequalities and covering numbers corresponding to $F_{n,k}$. Assuming Y is bounded as in Theorem 18, we write (6.6) as

$$\mathbb{E}\int_{C^{d_m}} \big|m_n(Z)-m(Z)\big|^2\mu(\mathrm{d}z) \le 2\min_{k\ge 1}\Big\{pen_n(k) \\ + \inf_{f\in F_{n,k}}\int_{C^{d_m}}\big|f(z)-m(z)\big|^2\mu(\mathrm{d}z)\Big\} + O\Big(\frac{1}{n}\Big). \quad (6.7)$$

We have assumed that for each $f \in F_n$, Y is bounded. Let $w_1^n = (w_1, w_2, \ldots, w_n)$ be a vector of n fixed points in $\mathbb{R}^{d_m}$ and let H be a set of functions from $\mathbb{R}^{d_m} \to [-K, K]$. For every $\varepsilon > 0$, we let $N(\varepsilon, H, w_1^n)$ be the L_1 ε-covering number of H with respect to $w_1, w_2, \ldots, w_n$. $N(\varepsilon, H, w_1^n)$ is defined as the smallest integer N such that there exist functions $h_1, \ldots, h_N : \mathbb{R}^{d_m} \to [-K, K]$ with the property that for every $h \in H$, there is a $j \in \{1, \ldots, N\}$ such that

$$\frac{1}{n}\sum_{i=1}^{n}\big|h(w_i)-h_j(w_i)\big| < \varepsilon.$$

Note that if $W_1^n = (W_1, W_2, \ldots, W_n)$ is a sequence of i.i.d. random variables, then $N(\varepsilon, H, W_1^n)$ is also a random variable. Now, let $W = (Z, Y)$, $W_1 = (Z_1, Y_1), \ldots, W_n = (Z_n, Y_n)$, and $C^{d_m} = [0, 1]^{d_m}$, we write

$$H_n = \Big\{h(z,y) := |y-f(z)|^2 : (z,y)\in C^{d_m}\times[-K,K] \text{ and } f\in F_n\Big\}.$$

The functions in H_n will satisfy the following:

$$0 \le h(z,y) \le 2\beta_n^2 + 2K^2 \le 4\beta_n^2.$$

Using Theorem 4 (Pollard, 1984), we have, for arbitrary $\varepsilon > 0$,

$$P\Big\{\sup_{f\in F_{n,k}}\Big|\frac{1}{n}\sum_{i=1}^{n}\big|Y_i-f(Z_i)\big|^2 - E\big|Y-f(Z)\big|^2\Big| > \varepsilon\Big\} \\ \le P\Big\{\sup_{h\in H_n}\Big|\frac{1}{n}\sum_{i=1}^{n}h(W_i)-E(h(W))\Big| > \varepsilon\Big\} \\ \le 8\mathbb{E}\Big[N\Big(\frac{\varepsilon}{8}, H_n, W_1^n\Big)\Big]\exp\Big(-\frac{n\varepsilon^2}{128(4\beta_n^2)^2}\Big). \quad (6.8)$$

Next, we try to bound the covering number $N\big(\frac{\varepsilon}{8}, H_n, W_1^n\big)$. Let us consider two functions $h_i(z,y) = |y - f_i(z)|^2$ of H_n for some $f_i \in F_n$ and $i = 1, 2$. We get

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n} \big|h_1(W_i) - h_2(W_i)\big| \\
&= \frac{1}{n}\sum_{i=1}^{n} \Big|\big|Y_i - f_1(Z_i)\big|^2 - \big|Y_i - f_2(Z_i)\big|^2\Big| \\
&= \frac{1}{n}\sum_{i=1}^{n} \big|f_1(Z_i) - f_2(Z_i)\big| \times \big|f_1(Z_i) - Y_i + f_2(Z_i) - Y_i\big| \\
&\le \frac{4\beta_n}{n}\sum_{i=1}^{n} \big|f_1(Z_i) - f_2(Z_i)\big|.
\end{aligned}$$

Thus, if $\{h_1, h_2, ..., h_l\}$ is an $\frac{\varepsilon}{8}$ packing of H_n on W_1^n, then $\{f_1, f_2, ..., f_l\}$ is an $\frac{\varepsilon}{32\beta_n}$ packing of F_n.

$$\text{Thus, } N\Big(\frac{\varepsilon}{8}, H_n, W_1^n\Big) \le N\Big(\frac{\varepsilon}{32\beta_n}, F_n, Z_1^n\Big). \tag{6.9}$$

The covering number $N\left(\frac{\varepsilon}{32\beta_n}, F_n, Z_1^n\right)$ can be upper bounded independently of Z_1^n by extending the arguments of Lemma 1 (Györfi et al., 2002). We now define the following classes of functions:

$$G_1 = \{\sigma(a^\top z + b) : a \in \mathbb{R}^{d_m}, b \in \mathbb{R}\},$$

$$G_2 = \{c\sigma(a^\top z + b) : a \in \mathbb{R}^{d_m}, b \in \mathbb{R}, c \in [-\beta_n, \beta_n]\}.$$

For any $\varepsilon > 0$,

$$N(\varepsilon, G_1, Z_1^n) \le 3\Big(\frac{2e}{\varepsilon}\log\frac{3e}{\varepsilon}\Big)^{d_m+2} = 3\Big(\frac{3e}{\varepsilon}\Big)^{2d_m+4}.$$

Also, we get

$$\begin{aligned}
N(\varepsilon, G_2, Z_1^n) &\le \frac{4\beta_n}{\varepsilon} N\Big(\frac{\varepsilon}{2\beta_n}, G_1, Z_1^n\Big) \\
&\le \Big(\frac{12e\beta_n}{\varepsilon}\Big)^{2d_m+5}.
\end{aligned}$$

We obtain the bound on the covering number of F_n,

$$\begin{aligned}
N(\varepsilon, F_n, Z_1^n) &\le \frac{2\beta_n}{\varepsilon} N\Big(\frac{\varepsilon}{k+1}, G_2, Z_1^n\Big)^k \\
&\le \Big(\frac{12e\beta_n(k+1)}{\varepsilon}\Big)^{(2d_m+5)k+1}.
\end{aligned} \tag{6.10}$$

According to (6.10), and for any $Z_1^n \in \mathbb{R}^{d_m}$, we have

$$N\Big(\frac{1}{n}, F_{n,k}, Z_1^n\Big) \leq \Big(12en\beta_n(k+1)\Big)^{(2d_m+5)k+1}. \tag{6.11}$$

Using the complexity regularization principle we have

$$\sup_{Z_1^n} N\left(\frac{1}{n}, F_{k,n}, Z_1^n\right) \leq N\left(\frac{1}{n}, F_{k,n}\right)$$

to be the upper bound on the covering number of $F_{k,n}$, and define for $w_k \geq 0$,

$$pen_n(k) = \frac{\text{constant} \times K^2 \times logN\big(\frac{1}{n}, F_{k,n}\big) + w_k}{n}$$

as a penalty term penalizing the complexity of $F_{k,n}$ (Kohler and Krzyżak, 2005). Thus, equation (6.11) implies that $pen_n(k)$ is of the following form with $w_k = 1$ and $\beta_n = \text{constant} < \infty$,

$$pen_n(k) = \frac{\text{constant} \times K^2 \times (2d_m+6)klog\big(12en\beta_n\big) + 1}{n} = O\left(\frac{kd_m log(n)}{n}\right).$$

The approximation error $\inf_{f \in F_{k,n}} \int_{C^{d_m}} \big|f(z) - m(z)\big|^2 \mu(\mathrm{dz})$ depends on the smoothness of the regression function. According to Theorem 3.4 of Mhaskar (1993), for any feedforward neural network with one hidden layer satisfying the assumptions of Proposition 2, we have

$$\big|f(z) - m(z)\big| \leq \left(\frac{1}{\sqrt{k}}\right)^{\frac{\delta}{d_m}}$$

for all $z \in [0,1]^{d_m}$. Thus, we have,

$$\inf_{f \in F_{n,k}} \int_{C^{d_m}} \big|f(z) - m(z)\big|^2 \mu(\mathrm{dz}) = O\Big(\frac{1}{k}\Big).$$

Using (6.7), we have

$$\mathbb{E}\int_{C^{d_m}} \big|m_n(Z) - m(Z)\big|^2 \mu(\mathrm{dz}) \leq O\left(\frac{kd_m log(n)}{n}\right) + O\left(\frac{1}{k}\right)$$

for sufficiently large n.

Now we can balance the approximation error with the bound on the covering number to obtain the optimal choice of k from which the assertion follows. □

Remark 15 *For practical purposes, we choose the number of hidden neurons in* *the BNT-2 model to be* $k = \sqrt{\frac{n}{d_m log(n)}}$.

6.4 Benchmark Comparison Experiments

We now present applications of the two BNT models to real-life data sets, and evaluate them against their component regression models, namely a simple CART model, a simple BCART model, a one-hidden-layer ANN, and a one-hidden-layer BNN.

6.4.1 Data

Table 6.1: Data set characteristics: number of samples and number of features, after removing observations with missing information or nonnumerical input features

Data set	Number of observations	Number of features
AutoMPG	398	7
Housing	506	13
Power	9568	4
Crime	1994	101
Concrete	1030	8

We use regression data sets available on the UCI machine learning repository (`https://archive.ics.uci.edu/ml/datasets.html`). These data sets have a limited number of observations and high-dimensional feature spaces (Lichman et al., 2013). Concretely, we use the Auto MPG, Housing, Power, Communities and Crime (Redmond and Baveja, 2002), and Concrete Compressive Strength (Yeh, 1998) data sets as testing ground for the models. To showcase the abilities of the BNT, we picked diverse, but mostly small regression data sets with few samples. It is on these small data sets where the benefits of neural optimization can usually not be exploited, but with the specific inductive bias of the BNT it becomes possible. As a part of the data cleaning process, we systematically eliminate all nonnumerical features and observations with missing values. Table 6.1 summarizes the characteristics of the data sets.

6.4.2 Implementation and Results

We shuffled the observations in each data set randomly and split it into training, validation, and test set in a ratio of 50/25/25. Each experiment is repeated 10 times with different randomly assigned training, validation, and test set. Nonnumerical features and samples with missing entries were systematically removed. Table 16.1 summarizes the characteristics of the resulting data sets. In preliminary experiments, data normalization or rescaling to the unit cube did not have a big impact on the models, so we kept the original values given in each of the sets. Experiments are carried out using R (version 3.6.1).

We fitted a CART model using the rpart package, with the stopping parameter 'minsplit' set to 10% of the training sample size. Random forest was implemented using the randomForest package in R. To fit a simple MacKay's BNN (MacKay, 1992a), we simply call it BNN, we use the brnn package with the number of hidden layers set to one and the number of hidden neurons set to the default value (i.e., 2). The brnn package implements a BNN with a Gaussian prior and likelihood. To fit a simple, one-hidden-layer ANN, we make use of the neuralnet package and set the number of hidden neurons to the default value (2). A Bayesian CART model is fit using the bartMachine package (Kapelner and Bleich, 2016), with the number of trees set to one. For feature selection under BCART, we use local thresholding of the variable inclusion proportions, although empirical explorations show that results are not very sensitive to other thresholding methods. As seen in Table 6.2, the component models of the BNTs exhibit consistent results, and neural networks perform better than the tree-based models for a majority of the data sets. For comparison, we also evaluate Bayesian Additive Regression Trees (BART) (Chipman et al., 2010). BART is trained also with 30 trees, 1000 MCMC steps (after burn-in of 100 steps), and otherwise default hyperparameters from the BayesTree R implementation. Also, Neal's Bayesian neural network, (Neal, 1996), we call it Neal's BNN, we used 5 hidden units for all the data sets and priors are assigned based on the recommendation given in (Chakraborty et al., 2005). We assigned a $N(0, 100^2)$ prior to the output bias and independent Gaussian prior to the rest of the parameters. The precision parameter has a gamma distribution with a mean of 0.05 and a variance of 0.01. More details about the prior specifications and the software can be obtained from Neal's website (`http://www.cs.utoronto.ca/œradford/fbm.software.html`). Results using Neal's neural network are reported under the column name Neal's BNN. It is clear from Table 6.2 that Neal's BNN many times improves the predictions of Mackay's BNN for the UCI data sets used in the experimentation.

We now turn to the implementation of the two BNT models. To implement BNT-1, we first record the selected features and predictions from the CART model, forming the set of features for the subsequent BNN model. Again, a CART model is trained with the stopping parameter 'minsplit' set to 10% of the training sample size. A one-hidden-layer BNN is then fit with the number of hidden neurons k drawn from Geometric distributions with success probabilities $p \in \{0.3, 0.6, 0.9\}$. The reversible jump MCMC to fit the Bayesian model is described in Section 2.3.7 of Chapter 2. We followed the implementation scheme

Table 6.2: Test set results (average) for each of the model across different data.

Data Set	Model		Performance Metrics				
		No. of features used	MAE	MAPE	RMSE	R^2	adjusted R^2
AutoMPG	CART	3	2.640	0.120	3.419	83.45	83.03
	BCART	3	2.796	0.117	3.693	80.65	80.32
	RF	5	2.541	0.109	3.440	84.96	82.67
	ANN	7	2.241	0.096	3.164	85.80	85.04
	BNN	7	2.253	0.097	3.123	86.12	85.44
	Neal's BNN	7	2.249	0.094	3.058	86.55	84.91
	BART	5	**2.002**	**0.090**	**2.933**	**88.50**	**87.50**
	BNT-1 (p=0.3)	4	2.111	0.091	3.016	87.15	86.77
	BNT-1 (p=0.6)	4	2.110	0.092	3.013	87.12	87.00
	BNT-1 (p=0.9)	4	2.119	0.092	3.018	87.30	87.00
	BNT-2	4	2.081	**0.090**	3.033	86.90	86.82
Housing	CART	3	3.161	0.163	5.068	69.62	69.00
	BCART	4	3.683	0.194	5.057	69.79	68.91
	RF	7	2.954	0.165	4.980	72.50	70.98
	ANN	13	2.736	0.132	4.782	72.95	70.62
	BNN	13	2.742	0.132	4.793	70.44	70.27
	BART	7	2.750	0.130	4.655	72.64	71.97
	BNT-1 (p=0.3)	4	2.643	0.129	4.731	73.58	73.00
	BNT-1 (p=0.6)	4	**2.641**	**0.128**	4.730	73.58	73.00
	BNT-1 (p=0.9)	4	**2.641**	**0.128**	4.730	73.58	73.00
	BNT-2	5	2.751	0.134	**4.597**	**75.04**	**74.85**
Power	CART	2	4.157	0.009	5.389	90.18	90.10
	BCART	2	5.502	0.008	4.561	92.97	92.91
	RF	4	4.515	0.008	4.950	93.05	92.98
	ANN	4	3.558	0.008	4.501	93.79	93.70
	BNN	4	3.563	0.007	4.510	**94.05**	**94.05**
	BART	4	3.522	0.008	4.495	93.69	92.45
	BNT-1 (p=0.3)	3	3.444	0.008	4.460	93.20	93.20
	BNT-1 (p=0.6)	3	3.443	0.008	4.463	93.20	93.20
	BNT-1 (p=0.9)	3	3.442	0.008	4.461	93.20	93.20
	BNT-2	3	**3.408**	**0.007**	**4.410**	93.40	93.40
Crime	CART	12	0.166	0.435	0.230	39.96	33.50
	BCART	15	0.186	0.580	0.231	39.44	25.07
	RF	15	0.161	0.505	0.208	54.62	52.07
	ANN	101	0.164	0.442	0.222	46.33	44.81
	BNN	101	0.167	0.567	0.290	58.00	58.00
	BART	15	**0.142**	**0.305**	**0.150**	**60.08**	**58.75**
	BNT-1 (p=0.3)	13	0.158	0.395	0.218	46.30	40.66
	BNT-1 (p=0.6)	13	0.154	0.395	0.218	46.30	40.66
	BNT-1 (p=0.9)	13	0.158	0.395	0.218	46.30	40.66
	BNT-2	16	0.143	0.367	0.193	57.88	57.40
Concrete	CART	5	7.462	0.286	9.414	69.42	68.95
	BCART	3	7.909	0.304	10.064	65.14	64.98
	RF	5	5.955	0.246	8.390	75.97	72.88
	ANN	8	6.987	0.235	9.194	70.92	70.14
	BNN	8	6.043	0.268	7.676	74.60	74.23
	BART	5	5.495	0.190	7.052	82.74	80.90
	BNT-1 (p=0.3)	6	5.493	0.194	6.961	83.33	83.00
	BNT-1 (p=0.6)	6	5.492	0.194	6.950	84.00	83.00
	BNT-1 (p=0.9)	6	5.493	0.194	6.961	83.33	83.00
	BNT-2	4	**5.473**	**0.178**	**6.636**	**87.95**	**87.80**

given by Rios Insua and Müller (1998) to implement BNN with variable architecture in the BNT-1 model. To implement BNT-2, we record important features and predictions from the BCART model using the bartMachine package in R. Then we use these as inputs to the ANN model with one hidden layer. The number of neurons in the ANN is taken to be $\sqrt{\frac{n}{d_m log(n)}}$, which is the optimal number derived in Section 6.3.2. Additionally, all data sets are min-max scaled to be in the $[0, 1]$ range before training the neural network models. From Table 6.2, we observe that across all data sets, the proposed BNT models greatly improve the performance of their component models. We note that the BNT-2 model outperforms all others on the 60% of the data sets (best results are made bold in Table 6.2). Consequently, we can expect the BNT predictions to be at least better than the individual model predictions, since cases, where further optimization is likely to have led to overfitting, are directly filtered out. BART (Chipman et al., 2010) is ranked second among all the regression models used in this study and outperformed BNT models for two out of five UCI data sets. Overall, the experimental results suggest that BNT models can be thought of as a new competitor to the popular BART (Chipman et al., 2010) model in the context of nonparametric regression.

6.5 Application to Water Quality Prediction

We consider a particular problem in a modern paper industry that produces papers for multiple uses. Paper machines produce papers using pulp, fiber, filler, chemical lubricant, and a considerable amount of water. The boiler produces steam for power generation purposes and also helps to make pulp for paper production. The steam produced in the boiler is used for cooking wood chips (along with the cooking chemicals). Steam is also sent to dryer cans to remove water from the sheet produced by the paper machine. The boiler stipulates the desired level of water quality to be received from the water treatment plant. In the plant, the process of demineralization (DM process) is applied for the removal of dissolved solids by the ion exchange process (IEP) that involves two stages of demineralization (Batchelder, 1965). In the first stage, it removes cations from water by the cation exchange process and then removes anions from water by the anion exchange process in the second stage (Lhassani et al., 2001).

DM process outlet pH happens to be the key performance indicator (KPI) of the water treatment plant. It was found that the plant can not produce water of desired quality specified by the boiler to be supplied to the paper machine. Finding a prediction model for the water quality will help the company to address the problem of variation in DM outlet water pH as well as an indication for the health of the boiler water tube. Controlling water pH will necessarily improve the water quality of the boiler, thus resulting in a monetary benefit for the company (Wang et al., 2019b; Zhang et al., 2019a). An extensive preliminary data analysis was conducted to determine a set of possible causal variables that happen to be the key to water pH level variations. Several statistical and machine learning tools like statistical quality control techniques, regression trees (RT), support vector regression (SVR) and artificial neural networks (ANN), etc. have previously been applied to solve the problems of water quality of river (Antanasijević et al., 2020; Mahuli et al., 1993; Ouyang et al., 2006; Singh et al., 2009) and surface water planning (Avila et al., 2018; Bhattacharya and Solomatine, 2005; Gmar et al., 2017). We investigate our proposed methodology on the water quality data set collected from a modern paper manufacturing company.

6.5.1 Data Collection Plan

In this section, we describe the motivating business problem and the data collection plan for our study. The data on pH is taken regularly from the boiler lab of the paper manufacturing company. However, when the study was taken up, we considered only the most recent 12 months of data for analysis. Measurement of water pH level at boiler lab is done through a digital meter which is calibrated once a month using a standard solution. DM process outlet pH happens to be the KPI of the water treatment plant, which gives the water of desired quality specified by the boiler.

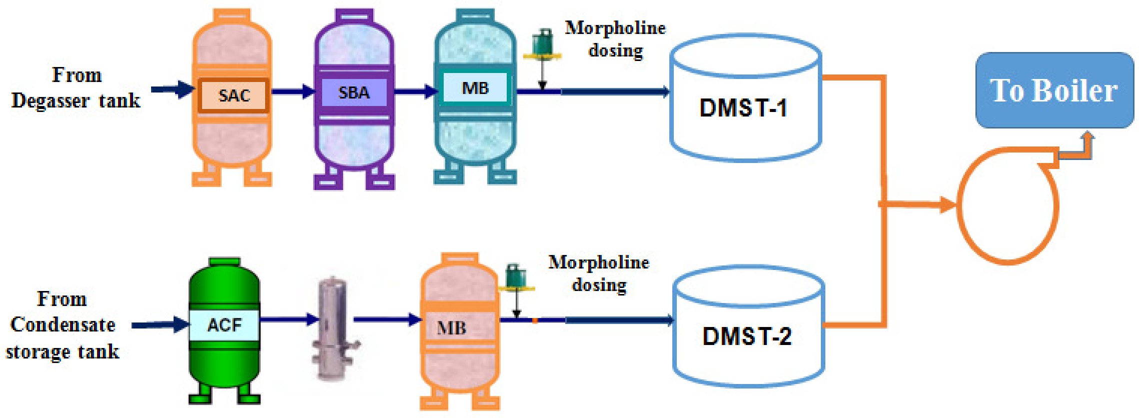


Figure 6.2: Process flow diagram of DM plant process

Water treatment plant comprises of two units: fresh-water treatment and condensate water treatment. Fresh-water treatment plant involves filtration of fresh-water obtained from the tube well followed by treatment of water with chemicals, making the water suitable for boiler and turbine. Condensate treatment involves filtration and treatment of condensate water obtained from paper machines. The quality of condensate coming from paper machines is the primary constraint for the current problem. Once the filtration of water or condensation is done, they are sent to the DM plant, where water is made suitable for the use of boiler (Vedelago and Millar, 2018). Brief details of the types of equipment used in DM plant processes are given below (see Figure 6.2).

- Strong acid cation (SAC) exchanger extracts the cations from the water by the cation resin and converts it into mineral acids. A vessel is also provided for controlling the flow rates of the inlet and outlet water.
- Strong base anion (SBA) exchanger is an anion exchanger in which a vessel is internally lined with rubber to prevent corrosion. An external vessel is also provided with manually operated valves to control the inlet and outlet water flow rates.
- The mixed bed (MB) unit comprises of a mild steel rubber-lined pressure vessel. Externally, the unit is provided with piping and valves to control the flow of water during service.
- Activated carbon filters (ACF) are used to remove chlorine and organic matter from water. Usually, ACFs are placed downstream of multi-grade filters and need to be regularly back-washed by reversal of flow to keep the surface of the carbon particles clean.

Once the water is treated, it is stored in a tank called the DM water storage tank (DMST). The water is pumped from DMST to the boiler via the DM transfer pump. Chemicals like HCl and caustic soda are used for regeneration of the resins in SAC, SBA, and MB Exchanger. Morpholine is used for scaling the pH of DM water. Condensate water is also sent through an MB exchanger for the treatment of water. It is essential to maintain DM outlet water of pH in the range of 8.5-9.2 for the excellent health of the water tubes and paper machine. It was found that the variation in pH of DM water is significant, and thus maintaining boiler water pH is an essential task for the company. A set of causal variables affecting the boiler water quality was found with the help of process experts and after some preliminary data analysis, which is discussed in the next subsection.

6.5.2 Data Description

Several brainstorming sessions were held with the process experts to identify the causal parameters causing the variations in the DM water outlet pH. Since not all the parameters are controllable by the users of the processes, a list of controllable parameters was prepared. Accordingly, data were collected for the DMST-1 and DMST-2 processes. The data set collected from the process for a year consists of the observations from the following causal variables: Inlet Flow, Water Pressure (water inlet pressure to the Exchanger), Air Pressure, MB stroke, and Amount of Morpholine or Chemical dosing (Liter per hr). The values of the response variable (DM water pH) varies between 7 to 10 (refer to Figure 6.3 for a graphical summary). Sample data sets for DMST-1 and DMST-2 are given in Tables 6.3 and 6.4. These data sets will be used for further model building that can help the company to forecast future water pH levels and take necessary actions for the reduction in water pH variation.

Table 6.3: Sample data set for DMST-1

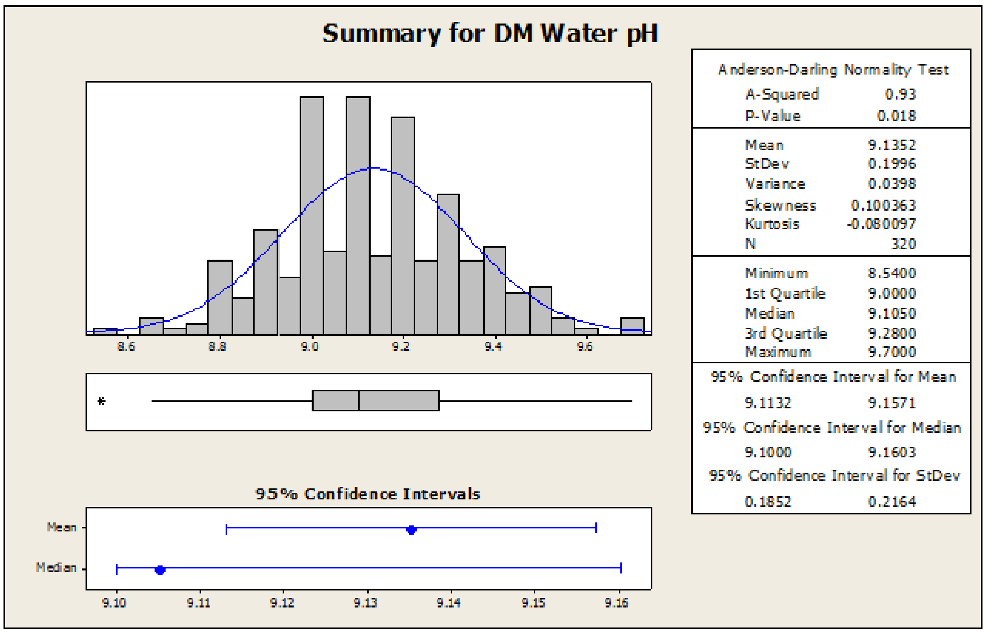
Sl. No.	Inlet Flow	Water Pressure	Air Pressure	MB stroke	Amount of chemical	DM water outlet pH
1	1980	5.8	5.0	70	9.96	9.276
2	2150	7.0	7.0	60	8.69	9.094
3	1780	6.0	5.0	45	7.73	8.594
4	2808	5.2	6.4	50	6.54	8.738
5	1590	6.2	5.7	40	5.56	8.592
6	2995	6.0	6.0	50	7.23	9.099
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Table 6.4: Sample data set for DMST-2

Sl. No.	Inlet Flow	Water Pressure	Air Pressure	MB stroke	Amount of chemical	DM water outlet pH
1	1489	5.4	6.0	80	10.04	9.246
2	1139	5.8	5.0	65	10.66	9.033
3	1448	5.4	5.0	45	7.69	8.483
4	1703	6.2	5.8	40	7.54	8.594
5	1258	6.2	5.7	35	5.99	7.589
6	1139	6.0	5.2	50	6.97	9.021
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

6.5.3 Analysis of Results

We started analyzing the data with the Anderson-Darling normality test on DM water outlet pH, and it confirms that the dependent variable doesn't follow a normal distribution (see Figure 6.3). An absence of normality in the DM water outlet pH data removed the possibility of applying conventional parametric regression methods. This leads us to think about nonparametric regression approaches. The stability of the process was checked using a $\overline{X} - R$ control chart (see Figure 6.3). From the $\overline{X} - R$ chart, it can be easily concluded that DM water outlet pH has high variation, and it also indicates the presence of some assignable causes in the process. The data set contains only numerical features with no missing entries. Thus no data cleaning task was performed.



Anderson-Darling Normality Test	
A-Squared	0.93
P-Value	0.018
Mean	9.1352
StDev	0.1996
Variance	0.0398
Skewness	0.100363
Kurtosis	-0.080097
N	320
Minimum	8.5400
1st Quartile	9.0000
Median	9.1050
3rd Quartile	9.2800
Maximum	9.7000
95% Confidence Interval for Mean	
9.1132	9.1571
95% Confidence Interval for Median	
9.1000	9.1603
95% Confidence Interval for StDev	
0.1852	0.2164

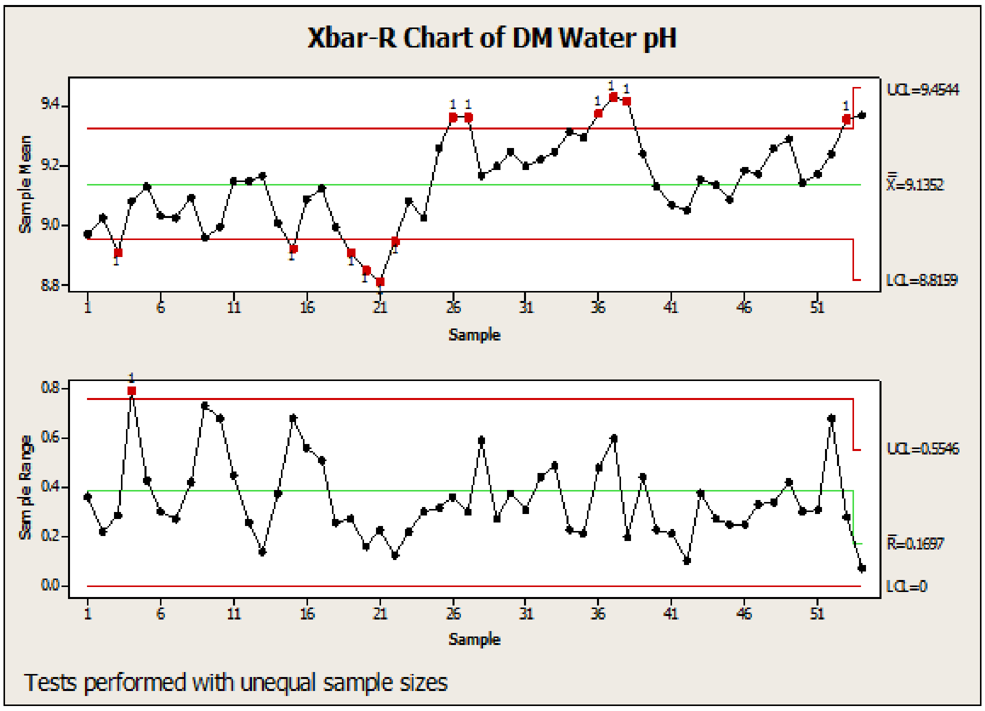


Figure 6.3: Summary statistics (above) and control chart (below) for DM water outlet pH

Table 6.5: Quantitative measures of performance for different regression models on test data set (average values of the metrics after 10-fold cross validations)

Regression Models	Data set	RMSE	MAPE	R^2	Adj R^2
Kernel SVR	DMST 1	4.06	4.50	75.66	70.29
	DMST 2	4.18	5.20	72.70	67.25
B-splines	DMST 1	4.32	5.40	69.85	63.30
	DMST 2	6.94	7.21	56.70	49.78
MARS	DMST 1	4.29	5.26	65.95	58.90
	DMST 2	6.74	7.93	57.53	47.05
RT	DMST 1	3.44	4.12	80.52	75.56
	DMST 2	3.89	4.78	76.56	71.23
ANN (with 2HL)	DMST 1	3.86	4.80	76.95	70.03
	DMST 2	4.12	5.91	70.10	64.73
BNT-2 Model	DMST 1	**3.05**	**3.40**	**85.40**	**82.50**
	DMST 2	**3.20**	**3.75**	**83.50**	**80.00**

We have shuffled the observations of the water pH data set randomly and split it into training and testing data sets in a ratio of 70 : 30. Each experiment is repeated ten times with different randomly assigned training and test sets. We will finally report the averages of the performance metrics observed over five times validations in Table 6.5. Various nonparametric regression models were applied to the data sets with default parameters, and the results were recorded. To implement Kernel SVR, we build a Gaussian kernel-based SVR model by the built-in package *e1071* in the R statistical package. The other two classical regression models, namely B-splines regression and MARS, were implemented using the built-in libraries *splines2* and *earth* in R, respectively. Then we started experimenting with the proposed hybrid BNT-2 model since it outperforms the BNT-1 model in many of the cases while implementation with UCI data sets. We first build BCART using the *bartMachine* implementation in R statistical software. We have used "minsplit" splitting rule to stop the tree, which determines the minimum number of points a node must have to be considered for splitting. From the tree, we extracted the set of all the most essential features as well as the prediction results and executed them in BNN architecture. We have used most of the default arguments present in these packages, as reported in Section 6.4.2. The training time and memory requirements are also quite low for the hybrid model compared to the 2HL ANN model. In Table 6.5, we present the results of different regression models on the water pH data sets, and the best results are displayed in bold fonts. The experimental evaluation of different regression models shows that the hybrid model outperforms other statistical and machine learning models with a significant margin. The proposed hybrid model will be useful for forecasting future values of water pH, given the values of the process variables. The model will also be helpful for the management and engineers to take further preventive action as well.

However, since our objective of the work is not only to develop a competitive prediction model for water pH level in the DM process but also to find out the optimal level of process parameters (in other words, causal variables) using the model. To keep DM water pH in the range of $8.5 - 9.2$ as specified by the boiler manual, the recommendations of our model are to maintain MB strokes, water pressure, and chemical consumption within a specified range, as shown in Table 6.6. The user of the process can't control inlet flow, and Air pressure need not be controlled as far as the recommendation of the proposed model goes. Table 6.6 gives the optimum range of controllable parameters for DMST 1 and DMST 2 based on the recommendation of regression analysis performed by an optimal hybrid model.

Table 6.6 depicts a range for all the critical process variables for which the expected DM water outlet pH will be within the required specification, i.e., $8.5-9.2$. However, to find out the exact values of the process parameters within the range prescribed in Table 6.6, a design of experiment (DOE) would be necessary, which was subsequently carried out to solve the problem. Though we have discussed only the proposed model and its accuracy level, our model also helped the manufacturing process industry to improve its water quality level and to gain monetary benefits due to a reduction in the chemical consumption.

Table 6.6: Optimal range of causal variables for achieving desired pH level

Process	Range for Water Pressure	Range for MB stroke	Range for chemical consumption	Expected range for DM water outlet pH
DMST 1	5.0-6.0	45-55	6.5-7.5	8.5-9.1
DMST 2	5.0-6.0	40-50	7.5-8.5	8.5-9.2

6.6 Concluding Remarks

This chapter presents two hybrid models that combine frequentist and Bayesian implementations of decision trees and neural networks. The BNT models are novel, first-of-their-kind proposals for nonparametric regression purposes. We find that the models performed quite better on small to medium-sized data sets than other state-of-the-art nonparametric models. Moreover, the BNT models have a significant advantage over purely frequentist hybridization. A Bayesian approach to constructing a CART or an ANN model can check the overfitting issue in the model. A BCART model allows placing priors that control the depth of the resultant trees, and BNNs with Gaussian priors are inherently regularized. This prevents the need to tune multiple parameters via cross-validation manually. Thus, the proposed BNT models overcome the deficiencies of their component models and the drawbacks of using fully frequentist or fully Bayesian models. We also show that the BNT models are consistent, which ensures their theoretical validity. When applied to solve water quality forecasting problems in a paper manufacturing industry, the proposed hybrid machine learning paradigm performs better than competing tools. The developed model was used to predict water pH levels in the DM process for a paper manufacturing sector and on a wide variety of standard regression data sets. The experimental results on UCI data sets are also promising for the proposed BNT models. It can be thought of as a new alternative to the popularly used BART model (Chipman et al., 2010) for nonparametric regression problems.

An immediate extension of this work will be to develop a hybrid methodology based on two Bayesian models, namely BCART or BART and BNN, to enhance uncertainty quantification and decision making in a fully nonparametric regression scenario. Even there are other scopes of improving the proposed BNT models for survival regression problems and lifetime data analysis. In the next chapter, we look at a different kind of regression problem, namely time series forecasting. Chapter 7 presents a hybrid framework based on linear and nonlinear models for forecasting unemployment rates for different countries.

Chapter 7

A Hybrid Time Series Model for Macroeconomic Forecasting

Related Publication:

1. Chakraborty, T., Chakraborty, A. K., Biswas, M., Banerjee, S., Bhattacharya, S. (2020). Unemployment Rate Forecasting: A Hybrid Approach. *Computational Economics*, https://doi.org/10.1007/s10614-020-10040-2.

Summary

Unemployment has always been a very focused issue causing a nation as a whole to lose its economic and financial contribution. The unemployment rate prediction of a country is a crucial factor for the country's economic and financial growth planning and a challenging job for policymakers. Traditional stochastic time series models, as well as modern nonlinear time series techniques, were employed for unemployment rate forecasting previously. But these macroeconomic data sets are mostly nonstationary and nonlinear in nature. Thus, it is atypical to assume that an individual time series forecasting model can generate a white noise error. In this chapter, we propose a hybrid model combining linear autoregressive integrated moving average (ARIMA) model and nonlinear autoregressive neural networks (ARNN) model to take advantage of the unique strength of ARIMA and ARNN models in modeling the unemployment rate data sets. The proposed hybrid approach is applied to six unemployment rate data sets from various countries, namely, Canada, Germany, Netherlands, New Zealand, Sweden, and Switzerland. The results of computational tests are very promising in comparison with other conventional methods. The results for asymptotic stationarity of the proposed hybrid approach using Markov chains and nonlinear time series analysis techniques are given in this chapter which guarantees that the proposed model cannot show 'explosive' behavior or growing variance over time. We also use simulated time series data set to show the effectiveness of the proposed hybrid forecasting model.

7.1 Introduction

Economic indicators such as GDP and labor statistics are used by investors to forecast economic trends and decide on the appropriate investment policies. In particular, the unemployment rate for any country represents one of the most important economic indicators for financial market participants due to its correlation with the country's business cycle and its influence on the monetary policy (Blanchard and Leigh, 2013). Accurate forecasting of the unemployment rate is central to economic decision-making and it helps in the design of government policy for the country's development. The study of unemployment rate forecasting started in the middle of the 1990s. Many time series models have been employed extensively for the prediction of macroeconomic variables, including unemployment. Previous studies on unemployment rate suggested an asymmetry in the unemployment rate data for various European countries (Milas and Rothman, 2008). One of the primary time series implications of such behavior is that it is inconsistent with a linear data generating process with symmetrically distributed innovations.

In previous studies on unemployment rate forecasting for various developed countries, the autoregressive integrated moving average (ARIMA) model was applied for analyzing Germany and Spain's unemployment data obtained from online search (Funke, 1992; Vicente et al., 2015). The usefulness and effectiveness of the classical linear ARIMA model were evident from the results obtained while using various European unemployment rate forecasting data sets (Edlund and Karlsson, 1993) and out-of-sample forecasts for Canadian unemployment rates (Khan Jaffur et al., 2017). But the situation was a bit different in case of unemployment rate forecasting for the USA. The threshold autoregressive (TAR) model, a classical nonlinear time series model, outperformed the linear time series models for forecasting the USA unemployment rate data set (Montgomery et al., 1998). For short term forecasting of seasonally adjusted monthly USA unemployment data sets, nonlinear models outperform the linear models (Nagao et al., 2019; Proietti, 2003).

The current progress in the area of modern statistics and machine learning have equipped the forecasters with nonlinear forecasting tools such as artificial neural networks (ANN), deep learning, and support vector machines (SVM) among many others (Katris, 2020). ANN is found to be the most accurate in forecasting unemployment over the asymmetric business cycle for the USA, Canada, UK, France, and Japan (Moshiri and Brown, 2004; Peláez, 2006). The previous results show that the nonlinear models are well-versed to seize the asymmetry of unemployment rate time series for long-term forecast horizons (Katris, 2020). Despite all these, there remains an asymmetry in unemployment rate forecasting and its elimination is bound to be a challenging job (Galbraith and van Norden, 2019).

The classical ARIMA model is competitive for forecasting stochastic time series whereas nonlinear ANN has produced favorable results for similar problems in the past decades. Nevertheless, neural nets also have the apparent drawback of finding the 'optimal' network architecture. To overcome this drawback, the autoregressive neural network (ARNN) model was proposed in some recent literature (Faraway and Chatfield, 1998). ARNN is a "white-box-like" model that fits a feed-forward neural net having one hidden layer to any time series data set with lagged values of the series as inputs (Teräsvirta et al., 2005). It has the advantages of less complexity and easy interpretability over ANN formulation (Hyndman and Athanasopoulos, 2018). The data sets at hand contain both linear and nonlinear patterns in the current problem of unemployment rate forecasting. It will be critical for policymakers to make any decision based on a single model since one can see regular changes in the dynamic behavior of the unemployment rates. By hybridizing linear and nonlinear models, one may reduce the bias and variances of the prediction error of component models (Oliveira and Torgo, 2014). Thus, combining both the linear and nonlinear models will be preferred for accurately predicting such complex autocorrelation structures (Khashei and Bijari, 2011a). Several hybrid models were applied in the past to solve various forecasting problems that arose in the stock market, financial econometrics, electricity, epidemiology, and other applied areas (Aladag et al., 2009; Arora and Taylor, 2016; Khashei and Bijari, 2011b; Pai and Lin, 2005; Terui and Van Dijk, 2002; Tümer and Akkuş, 2018; Zhang, 2003). All these hybrid models are practically shown to be useful in solving real-life forecasting problems, but there are hardly any theoretical results for asymptotic stationarity for these models in the literature.

This chapter proposes a hybrid approach that studies the relationship between linear and nonlinear components of the unemployment rate time series (Chakraborty et al., 2020a). The proposed hybrid methodology assumes an additive relationship between linear and nonlinear models, assuming that different models can capture the linear and nonlinear patterns of a time series separately, and then the forecasts can be combined. The proposed hybrid method will be appropriate for explaining variations of the unemployment rate in the presence of nonstationarity and nonlinearity in this time series. In the first phase of the proposed model, an ARIMA model is applied to catch the linear patterns of the data set. Residual error values of the ARIMA model are calculated and restored for further modeling. In the next stage, a nonlinear ARNN model is applied to capture the nonlinear trends in the data set using the residual values obtained from ARIMA. We call this two-step approach as 'hybrid ARIMA-ARNN' model. Of particular interest for the statisticians in time series forecasting literature is the question of asymptotic stationarity of the model, viz. whether the probabilistic structure of the series is constant over time or at least asymptotically constant. In this study, we have shown the asymptotic stationarity of the proposed hybrid ARIMA-ARNN model using Markov chains and nonlinear time series analysis techniques. The theoretical results for asymptotic stationarity guarantee that the model cannot have a growing variance with time. The asymptotic behavior of the proposed hybrid model is especially crucial for predictions over a larger interval of time or when using the model to generate an artificial time series. We theoretically show asymptotic stationarity for the proposed hybrid approach in Section 7.4. Finally, we test the performance of the proposed model on six unemployment rate data sets and compare our proposed model with several other state-of-the-art forecasting models in Section 7.5. A simulation study is also presented in Section 7.6 to make our results more convincing.

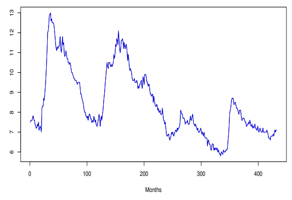
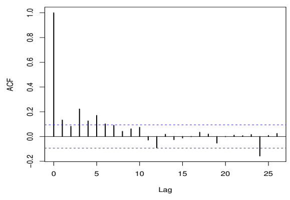
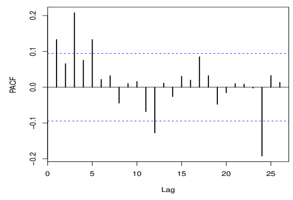
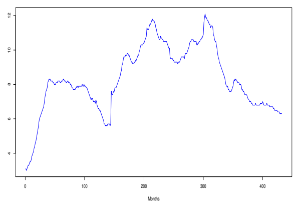
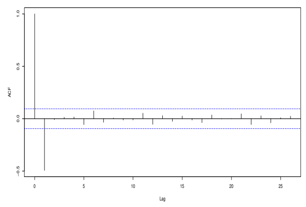
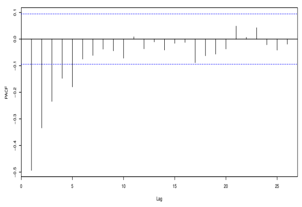
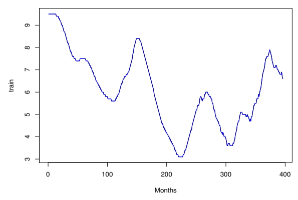
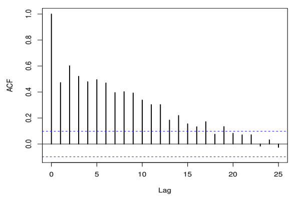
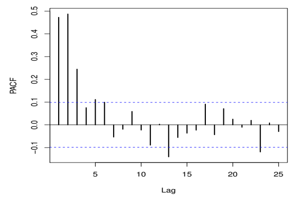
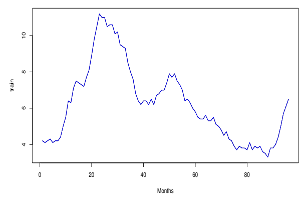
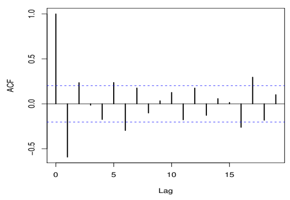
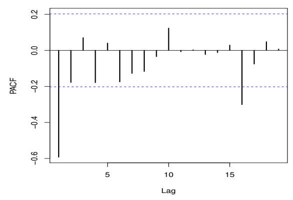
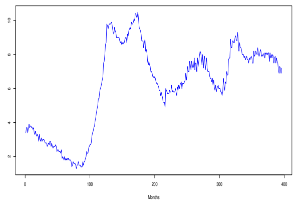
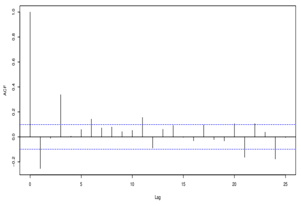
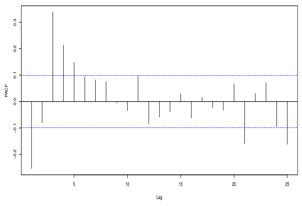
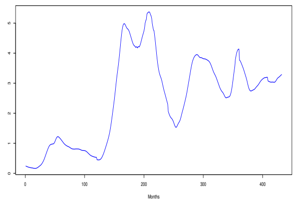
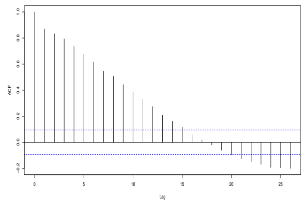
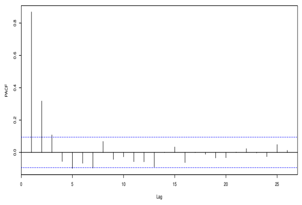
7.2 Unemployment Rate Data and its Characteristics

The unemployment rate represents the number of unemployed as a percentage of the labor force. Forecasting unemployment rate can be defined as the projected value for the number of unemployed people as a percentage of the labor force. Six seasonally adjusted monthly data sets on unemployment rates for Canada, Germany, Netherlands, Sweden, and Switzerland and one quarterly data of New Zealand were collected from the open-access data repository FRED Economic Data sets (Link: `https://fred.stlouisfed.org/`) and OECD data repository (Link: `https://data.oecd.org/`). A summary of these unemployment rate data sets is provided in Table 7.1. The plots of the training data for various countries are given in Table 7.2. The graphical presentation of the data sets confirms the presence of nonstationarity and nonlinearity in these unemployment rate data. We also use a well-known linearity test, namely the new-F test, since it covers the most extensive set of alternatives of nonlinearity (Terui and Van Dijk, 2002). The new-F test rejects the null hypothesis of linearity most strongly for all the six unemployment rate data sets.

Table 7.1: Descriptions of the unemployment rate data sets

Data	Total size	Maximum value	Minimum value	Training set size (Years)	Test set size (Years)
Canada	468	13.0	5.6	432 (1980-2015)	36 (2016-2018)
Germany	468	12.1	3.0	432 (1980-2015)	36 (2016-2018)
Netherlands	432	9.5	3.1	396 (1983-2015)	36 (2016-2018)
New Zealand	132	11.2	3.3	100 (1986-2009)	32 (2010-2018)
Sweden	432	10.5	1.3	396 (1983-2015)	36 (2016-2018)
Switzerland	468	5.37	0.1	432 (1980-2015)	36 (2016-2018)

Table 7.2: Graphical analysis of training unemployment rate data sets for different countries and its corresponding ACF and PACF plots

Country	Training data	ACF plot	PACF plot
Canada			
Germany			
Netherlands			
New Zealand			
Sweden			
Switzerland			

In this chapter, we consider approaches for the prediction of univariate unemployment rate time series. A combination of linear and nonlinear methods that take into account the specific characteristics of data can offer more accurate predictions. Some inherent characteristics of these data sets are the departure from normality and the nonlinearity of the dependence structure of the data which are also evident from previous studies of these data sets (Galbraith and van Norden, 2019; Katris, 2020; Nagao et al., 2019). To take into account the nonstationarity, the linear ARIMA model is first considered. Furthermore, since nonlinearity exists, the neural networks-based ARNN model seems suitable in the second stage of the hybrid model.

7.3 Methodology

This work proposes a hybrid model based on ARIMA and ARNN model to forecast the unemployment rates for six countries. Below we first discuss about the constituent models to be used in the hybridization.

7.3.1 ARIMA Model

ARIMA is a linear time series model, used for tracking linear tendencies in stationary time series data. ARIMA model is denoted by ARIMA(p, d, q). The parameters p and q are the order of the AR model and the MA model respectively, and d is the level of differencing (used for converting nonstationary series into a stationarity one) (Box et al., 1976). ARIMA model can be mathematically expressed as follows:

$$y_t = \theta_0 + \sum_{i=1}^{p} \phi_i y_{t-i} + \varepsilon_t - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j},$$

where y_t denotes the actual value of the variable under consideration at time t, ε_t is the random error at time t, ϕ_i and θ_j are the coefficients of the model. The necessary steps for building an ARIMA model for any given time series data set are as follows: model identification of the model (achieving stationarity), estimation of model parameters (the autocorrelation function (ACF) and the partial autocorrelation function (PACF) plots are used to select the AR and MA model parameters, respectively), and model diagnostics checking (finding the 'best' fitted forecasting model using Akaike information criterion (AIC) or the Bayesian information criterion (BIC)) (Hyndman and Athanasopoulos, 2018).

7.3.2 ARNN Model

ANN is a widely used supervised learning model, highly useful for sophisticated nonlinear time series forecasting. Any neural net architecture can be described as a network of "neurons", arranged in layers, namely the input layer, hidden layer, and output layer. The information from one layer to another layer is passed using weights that are selected using a risk minimization based 'learning algorithm'. The ARNN model is a modified neural network model especially designed for time series data sets which uses a pre-specified number of hidden neurons in its architecture (Faraway and Chatfield, 1998). It uses lagged values of the time series as inputs to the model. ARNN(p, k) is a nonlinear feed-forward neural net model with one hidden layer (having p lagged inputs) and k hidden units in the hidden layer. It also uses BIC as the criterion for comparing different models created by ARNN. Here $\hat{x}_t$ is computed using selected past observations $x_{t-j_1}, \cdots, x_{t-j_p}$ as the inputs. Thus, ARNN model with one hidden layer can be depicted with the following mathematical form:

$$\hat{x}_t = \phi_0 \left\{ w_{c_0} + \sum_k w_{k_0} \phi_k \left(w_{c_k} + \sum_i w_{i_k} x_{t-j_i} \right) \right\},$$

where $\{w_{c_k}\}$ denotes the connecting weights and ϕ_i is the activation function. Weights of the ARNN model are trained using a gradient descent backpropagation algorithm (Rumelhart et al., 1985). The ARNN(p, k) model uses p as the number of lags for an AR(p) model and k is usually set to $k = \left[\frac{(p+1)}{2}\right]$ for non-seasonal time series data (Hyndman and Athanasopoulos, 2018).

7.3.3 Proposed Hybrid ARIMA-ARNN Model

The ARIMA model is a popular classical time series model for linear data structures. In contrast, with the advent of neural networks, even nonlinear structures in the data set can be handled. The aim of developing a hybrid model based on linear and nonlinear time series models is to harness the advantages of single models and reduce the risk of failures of single models. The underlying assumption of the developed hybrid approach is that the relationship between linear and nonlinear components are additive. Even if the relationship is of multiplicative type, in the log-log scale, it becomes additive. Hence, without loss of generality, we may assume the relationship to be additive.

We propose a hybrid ARIMA-ARNN model, which is a two-step pipeline approach. In the first step of the proposed hybrid approach, an ARIMA model is built to model the linear components of time series, and a series of forecasts are generated. In the second phase, the ARIMA residuals are modeled using a nonlinear ARNN model. The formulation of the proposed hybrid ARIMA-ARNN model (Z_t) can be formally represented as follows:

$$Z_t \;=\; Y_t + N_t,$$

where Y_t is the linear part and N_t is the nonlinear part of the hybrid model. We can estimate both Y_t and N_t from the training data set. Let, $\hat{Y}_t$ be the forecast value of the ARIMA model at time t and ε_t represent the error residuals at time t, obtained from the ARIMA model. We can then write

$$\varepsilon_t = Z_t - \hat{Y}_t.$$

The residuals are modeled by the ARNN model and can be represented as follows

$$\varepsilon_t = f(\varepsilon_{t-1}, \varepsilon_{t-2}, ..., \varepsilon_{t-n}) + \varsigma_t, \quad \text{for some integer } n,$$

where f is a nonlinear function of the ARNN model and ς_t is the random shocks. Therefore, we can write the combined forecast as:

$$\hat{Z}_t = \hat{Y}_t + \hat{N}_t,$$

where $\hat{N}_t$ is the forecasted value of the ARNN model. ARNN models the left-over autocorrelations in the residuals, which ARIMA could not model. This is important because the linear ARIMA model may fail to generate white noise behavior in the forecast residuals due to the model misspecification and disturbances in the unemployment rate time series. Therefore, if the error series is modeled again, the performance of the original forecaster can be improved, even though marginally at times. A flowchart of the hybrid ARIMA-ARNN model is presented in Figure 7.1. The algorithmic representation of the proposed hybrid approach is given in Algorithm 7.1.

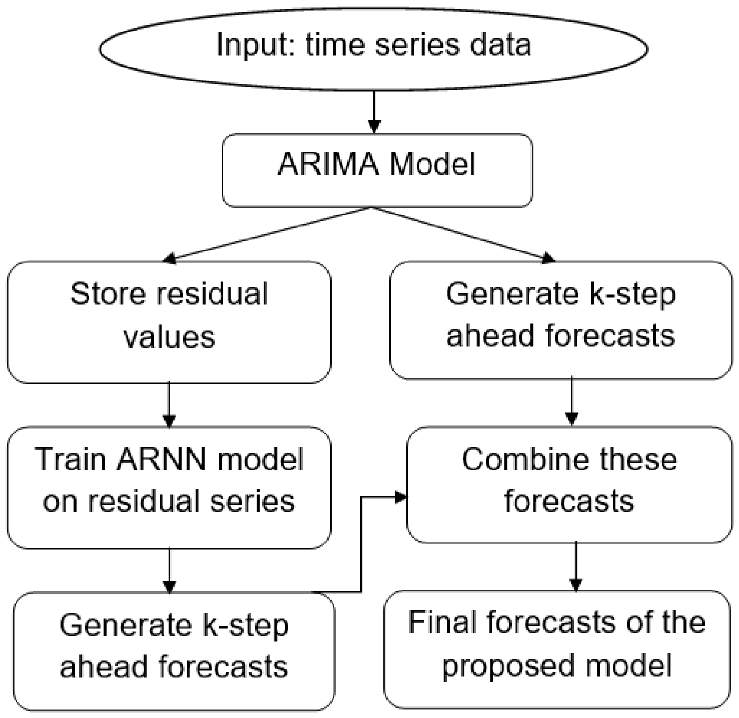

Input: time series data
ARIMA Model
Store residual values
Generate k-step ahead forecasts
Train ARNN model on residual series
Combine these forecasts
Generate k-step ahead forecasts
Final forecasts of the proposed model

Figure 7.1: Flow diagram of the proposed hybrid ARIMA-ARNN model

Algorithm 7.1: Proposed hybrid ARIMA-ARNN approach

```
1 Given a time series, input the in-sample (training) and out-of-sample (test)
   data.
2 Determine the best ARIMA($p, d, q$) model using the in-sample (training)
   data.
     • ARIMA parameters $p$, $d$, and $q$ values are selected using standard
       procedures, as described in Section 7.3.1.
     • Obtain the predictions using the selected ARIMA($p, d, q$) model for the
       training data.
     • Obtain the residual series ($\varepsilon_t$) by subtracting ARIMA predicted values
       from the original training series.
3 Determine the best ARNN($p, k$) model on the training residual series ($\varepsilon_t$).
     • Perform lag selection for the training of residual series and apply ARNN
       model with $p$ selected lagged inputs from ($\varepsilon_t$) and $k$ hidden units as
       described in Section 7.3.2.
     • Obtain predictions using the ARNN model ($\hat{\varepsilon}_t$).
4 Final predictions ($\hat{Z}_t$) are obtained by combining the ARIMA predictions
   with ARNN predictions ($\hat{\varepsilon}_t$).
```

Remark 16 *There exist large classes of time series (e.g., unemployment rate time series), such as those with nonlinear moving average components, that are not well modeled by feedforward networks or linear models, but can be modeled by the proposed hybrid ARIMA-ARNN model. Practical ability will be shown in the results of unemployment rate forecasting data analysis (refer to Section 7.5) and also with a simulated data (refer to Section 7.6) where the hybrid ARIMA-ARNN model gave the best performance over state-of-the-art models.*

7.4 Asymptotic Stationarity of the Model

ARIMA has an in-built mechanism to transform a nonstationary time series into a stationary time series by taking the differencing of the given time series (Brockwell and Lindner, 2010). ANNs are asymptotically stationary and it requires the process to be stationary while training the neural network, but when applied to a nonstationary process, the out of sample predictions becomes poor (Leoni, 2009). In the hybrid formulation based on ARIMA and ARNN model, this problem can be overcome since we deal with only the additive error terms generated by ARIMA and model it using ARNN model.

Let us consider a nonlinear ARNN model generated by additive noise of the ARIMA model. Let ε_t denote a time series generated by a nonlinear autoregressive process as defined in (7.1). Thus, the stochastic difference equation is of the form:

$$\varepsilon_t = f(\varepsilon_{t-1}, \varepsilon_{t-2}, ..., \varepsilon_{t-p}, \theta) + \varsigma_t, \tag{7.1}$$

where ς_t is an i.i.d. noise process and $f(\cdot, \theta)$ is a feedforward neural network with weight parameter vector θ. This is called an ARNN process of order p and has k hidden nodes in its one hidden layer, denoted by ARNN(p, k) model. ARNN models are a natural generalization of the classic linear autoregressive AR(p) process

$$\varepsilon_t = \alpha_1 \varepsilon_{t-1} + \cdots + \alpha_p \varepsilon_{t-p} + \varsigma_t.$$

Now, we define f as a neural network and z denote a p-dimensional input feature to the ARNN model (error residuals obtained from ARIMA model). We consider the following architecture:

$$f(z) \quad = \quad c_0 + \sum_{i=1}^{k} w_i \sigma\Big(a_i + {a_i}' z\Big) \tag{7.2}$$

where a_i, w_i, and c_0 are scalar weights, a_i are p-dimensional weight vectors and $\sigma(\cdot)$ is a bounded nonlinear sigmoidal function such as tan-hyperbolic or logistic function. Since we consider unbounded additive noise terms generated by the ARIMA model, we can not apply the results based on state space model of the Markov chains. But if we suppose $E(\varsigma_t) = 0$, then f equals the conditional expectation $E\big(\varepsilon_t | \varepsilon_{t-1}, \ldots, \varepsilon_{t-p}\big)$ and $f(\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-p})$ is the best prediction for ε_t in the mean square sense. Thus, for the unbounded noise terms, we define the following notation:

$$\begin{aligned} z_{t-1} &= \Big(\varepsilon_{t-1}, \ldots, \varepsilon_{t-p}\Big)' \\ F(z_{t-1}) &= \Big(f(z_{t-1}), \varepsilon_{t-1}, ..., \varepsilon_{t-p+1}\Big)' \\ e_t &= \Big(\varsigma_t, 0, \ldots, 0\Big)' \end{aligned}$$

Then we write scalar AR(p) models in (7.1) as a first-order vector model

$$z_t \quad = \quad F(z_{t-1}) + e_t \tag{7.3}$$

with $z_t, e_t \in \mathbb{R}^p$ (Chan and Tong, 1985). Also, we write

$$\begin{aligned} p^n(z, A) &= P\{z_{t+n} \in A | z_t = z\} \\ p(z, A) &= p^1(z, A) \end{aligned}$$

for the probability of going from point z to set $A \in \mathbb{B}$ in n steps, then $\{z_t\}$ with $p(z, A)$ forms a Markov chain with state space $(\mathbb{R}^p, \mathbb{B}, \lambda)$, where $\mathbb{B}$ is a Borel set on $\mathbb{R}^p$ and λ be the usual Lebesgue measure.

The Markov chain $\{z_t\}$ is said to be φ-irreducible, if for some σ-finite measure φ on $(\mathbb{R}^p, \mathbb{B}, \lambda)$ and for all $z \in \mathbb{R}^p$, we have

$$\sum_{n=1}^{\infty} p^n(z, A) > 0$$

whenever $\varphi(A) > 0$. This essentially means that all parts of the state space can be reached by the Markov chain irrespective of the starting point. Another interesting property of the Markov chain is *aperiodicity* that loosely means that there are no (infinitely often repeated) cycles (Tong, 1990).

Since we are interested in the long-term properties of the time series, one may ask whether certain features such as mean or variance change over time or remain constant. A time series is called weakly stationary if $E(\varepsilon_t) = \mu$ and $cov(\varepsilon_t, \varepsilon_{t+h}) = \gamma_h \; \forall \; t$, that is mean and covariance do not depend on time t. A more strong criterion is that the whole distribution of the process does not depend on the time and then the series is called strictly stationary. It is interesting to note that strong stationarity implies weak stationarity if the second order moments of the series exist (Brockwell and Lindner, 2010). We further need to define asymptotic stationarity in this context:

Definition 13 *If ε_t is strictly stationary then $P(\varepsilon_t \in A) = \Pi(A)$, for all t and $\Pi(\cdot)$ is called the stationary distribution of the series. We call the series asymptotically stationary if it converges to its stationary distribution (if it is not started with Π):*

$$\lim_{t\to\infty} P(\varepsilon_t \in A) = \Pi(A).$$

Also, trivially the series can only be stationary from the beginning if it starts with the stationary distribution such that $\varepsilon_0 \sim \Pi$.

Definition 14 *Let $\{z_t\}$, a Markov chain, is said to be geometrically ergodic if there exists a probability measure $\Pi(A)$ on the state space $(\mathbb{R}^p, \mathbb{B}, \lambda)$, and for $\rho > 1$ and for all $z \in \mathbb{R}^p$,*

$$\lim_{n\to\infty} \rho^n \|p^n(z, \cdot) - \Pi(\cdot)\| \quad = \quad 0$$

where $\|.\|$ denotes the total variation. Then Π satisfies the invariance equation

$$\Pi(A) = \int p(z, A)\Pi(dz), \; \forall \; A \in \mathbb{B}.$$

If the markov chain is geometrically ergodic then its distribution will converge to Π and the corresponding time series is called asymptotically stationary (Trapletti et al., 2000).

Lemma 7 *Let $\{z_t\}$ be defined by (7.3), and let $E|\varsigma_t| < \infty$ and the probability distribution function (PDF) of ς_t is positive everywhere in $\mathbb{R}$. Then if f is defined by (7.2), the Markov chain $\{z_t\}$ is φ-irreducible and aperiodic.*

Proof It can easily be shown that $\{z_t\}$ is φ-irreducible if the support of the PDF of ς_t is the whole real line, viz., the PDF is positive everywhere in $\mathbb{R}$ (Chan and Tong, 1985). In our case, every non-null p-dimensional hypercube can be reached in p steps with positive probability (and hence every non-null Borel set A).

A necessary and sufficient condition for $\{z_t\}$ to be *aperiodic* is to have a set A and positive integer n such that $p^n(z, A) > 0$ and $p^{n+1}(z, A) > 0$ for all $z \in A$ (Tong, 1990, p. 455). In this case, this is true for all n due to consideration of the unbounded additive noise. $\square$

Remark 17 *Lemma 7 states that the state space of the Markov chain cannot be reduced depending on the starting point. An example of a reducible Markov chain is a series that is always positive if only $z_0 > 0$ (and negative otherwise). But this cannot happen in the ARNN(p, k) model due to having unbounded additive noise terms.*

The theorem below states the necessary condition for geometric ergodicity of a markov chain (Chan and Tong, 1985).

Theorem 19 (Chan & Tong, 1985) *Suppose $\{\varepsilon_t\}$ and $\{z_t\}$ are defined as in (7.1) and (7.3), respectively. Further, let F be compact (or preserve compact set) and can be decomposed as $F = F_h + F_d$ and the following conditions hold:*
(i) $F_h(.)$ is continuous and homogeneous and $F_d(.)$ is of bounded range;
(ii) The origin is a fixed point of F_h and F_h is uniform asymptotically stable.
(iii) If $E|\varsigma_t| < \infty$ and PDF of ς_t is positive everywhere in $\mathbb{R}$;
then $\{z_t\}$ is geometrically ergodic.

Proof See (Chan and Tong, 1985, p. 671-673). $\square$

The next theorem gives the main result for asymptotic stationary of the hybrid ARIMA-ARNN model.

Theorem 20 *Let $E|\varsigma_t| < \infty$ and the PDF of ς_t is positive everywhere in $\mathbb{R}$, and $\{\varepsilon_t\}$ and $\{z_t\}$ are defined as in (7.1) and (7.3), respectively. Then if f is a nonlinear neural network as defined in (7.2), then $\{z_t\}$ is geometrically ergodic and $\{\varepsilon_t\}$ is asymptotically stationary.*

Proof The noise process ς_t satisfies $E|\varsigma_t| < \infty$ by assumption (e.g., Gaussian noise). It is also important to note that neural network activation function (logistic sigmoidal activation function in this case) is continuous compact function and has bounded range. For application to the unemployment rate problem, we considered the logistic activation function $F(z) = \frac{1}{1+e^{-z}}$ in the ARNN model during implementation.
Thus $\{z_t\}$ satisfies all the criteria to be geometrically ergodic and using Theorem 19, one can write that for the ARNN process with $F_h \equiv 0$ and $F_d \equiv F$. Thus, the series $\{\varepsilon_t\}$ is asymptotically stationary. $\square$

Remark 18 *We have found sufficient conditions for asymptotic stationarity of the proposed hybrid ARIMA-ARNN model. This is important for predictions over larger intervals of time, for example, one might train the network on an available sample and then use the trained network to generate new data with similar properties like the training sample. Theoretical results on asymptotic stationary guarantees that the proposed approach cannot have 'explosive' behavior or growing variance with time.*

7.5 Experimental Results and Discussions

Six open-access unemployment rate data sets are used to determine the effectiveness of the proposed model. The properties of these data sets are different and have been used in many previous studies (Edlund and Karlsson, 1993; Khan Jaffur et al., 2017; Moshiri and Brown, 2004; Peláez, 2006). Various linear and nonlinear models have been studied on these data sets that show highly nonlinear patterns in these regions. Mean absolute error (MAE); root mean square error (RMSE) and mean absolute percent error (MAPE) are used to evaluate the performances of the proposed model and other single models, for details refer to Section 2.5 of Chapter 2. Six unemployment rate data sets are divided into training and testing data, as described in Table 7.1. These data sets are mostly nonlinear and non-Gaussian in nature and statistical tests confirm this (see Section 7.2). Even time series plots of the data sets show nonlinearity and non-stationarity (refer to Table 7.2). We experimentally evaluate the performances of ARIMA, ANN, SVM, ARNN model, hybrid ARIMA-ANN model (Zhang, 2003), hybrid ARIMA-SVM model (Pai and Lin, 2005), in comparison with our proposed hybrid ARIMA-ARNN model for all these data sets.

We start the experimental evaluation with the classical ARIMA(p, d, q) using the "forecast" package in R statistical software. Fitting an ARIMA model, we need to specify the orders of the model. Using the ACF plot and PACF plots, we can decide the value of the model's parameters. We performed the Augmented Dickey-Fuller (ADF) test for stationarity check for the model to determine the amount of d in the ARIMA model. The 'best' fitted ARIMA model is chosen using

AIC value for each training data set. As the ARIMA model is fitted, predictions are generated for one year and three year time steps. In the second stage, residuals obtained using the ARIMA model are remodeled with ARNN(p, k) model. We employ a pre-defined Box-Cox transformation set to $\lambda = 0$ to ensure the forecast values to stay positive. The values of p and k are obtained by training the network, which is a data-dependent approach as in Hyndman and Athanasopoulos (2018). Further, both the linear and nonlinear forecasts are added together to get the final forecasts.

Other individual models like SVM were implemented using the "e1071" package in R statistical software with kernel = 'radial' of type polynomial (degree = 3). It is of form

$$K(x, y) = exp\left(-\gamma \sum_{j=1}^{p} (u - v)^2\right);$$

and γ here is a tuning parameter which accounts for the smoothness of the decision boundary and controls the variance of the model. The value of γ is chosen based on the following formulae: $\gamma = 1/(\text{data dimension})$ for all the experiments. For all the experiments, we use small values for γ, since it makes the decision line or boundary is smoother and has low variance (Hastie et al., 2009). The ARNN model was applied using the "forecast" package with 'nnetar' function and for ANN model, we have used "nnfor" package with the 'mlp' function in R. The ARNN(p, k) model parameters are chosen based on the formula given in Subsection 7.3.2. For all the experiments with the ANN model, we have used one hidden layer with the number of hidden neurons $k \approx \sqrt{n}$, where n being the sample size. We report all the choices of tuning parameters for all the models in Tables 7.3-7.8.

The experimental results are obtained as follows: ARIMA (3,1,3) was fitted to Canada unemployment rate data with AIC and log-likelihood values as -154.25 and 84.37, respectively. The ARIMA residuals were further trained with ARNN(5,3) model with an average of 20 networks. Further, we computed the predicted test outputs of the hybrid ARIMA-ARNN model and compared it with actual test outputs. The values of different performance metrics are reported in Table 7.3. For the Germany data set, ARIMA(1,2,1) having AIC = -494.05 and L = 250.03 was fitted. An ARNN(1,1) model (1-1-1 network configuration) with an average of 20 networks, each having four weights, was trained on ARIMA training residuals. Both the forecasted results of ARIMA and ARNN are added together to obtain the final forecast values based on which RMSE, MAE and MAPE values are computed and reported in Table 7.4. Similarly, we applied the proposed hybrid ARIMA-ANN model for New Zealand and Netherlands data sets and the results are reported in Table 7.6 and 7.5, respectively. ARIMA(1,1,2) with log-likelihood = -22.54 and AIC=53.07 was first fitted to the quarterly unemployment rate data set of New Zealand. The residuals are modeled with an ARNN(10,5) model with an average of 20 networks. For the Netherlands' monthly data, we fit an ARIMA(1,2,1) and ARNN(12,6) on ARIMA residuals. In the case of Sweden data, ARIMA(3,1,1) was fitted with AIC = 0.65 and L = 4.68. Further, ARIMA residuals were trained using ARNN(16,8) model (16-8-1 network configuration) with an average of 20 networks, each having 145 weights. Again, we obtain the final predicted forecast values for Sweden's test data sets by adding both the ARIMA and ARNN forecasts. The values of the performance metrics for measuring forecasting accuracy are reported in Table 7.7. Finally, ARIMA(2,1,2) was fitted to Switzerland data having AIC = -1745.46 and L equals to 877.73. The residuals obtained from the ARIMA model was trained using ARNN(3,2) model with an average of 20 networks, each of which is a 3-2-1 network with 11 weights. Finally, the forecast results of ARIMA and ARNN residual forecasts are added together to obtain the predicted forecasted values. We then compute the values of the performance measures, viz. RMSE, MAE and MAPE values and report them in Table 7.8.

For comparison purposes, we applied single ARIMA, ANN, ARNN, SVM, along with hybrid ARIMA-ANN (Zhang, 2003), hybrid ARIMA-SVM model (Pai and Lin, 2005) models for the seasonally adjusted unemployment rate data sets of these countries. All the experimental results are reported in Tables 7.3, 7.4, 7.5, 7.6, 7.7, and 7.8. Figures in ( ) for all the Tables indicate the values of the tuning parameters for each of the forecasting models. The predicted forecasts for the test data sets of the proposed hybrid model for six data sets, along with actual test values, are plotted in Figure 7.2. The performances of the proposed hybrid ARIMA-ARNN model are superior as compared to all the individual models. In comparison to other hybrid models, our proposal outperformed all the hybrid models in a significant margin. The theoretically proven asymptotic stationarity of the proposed hybrid model suggested that the model can not have a growing variance over time. The consistency in experimental results empirically approves the same. Thus, the usefulness of the proposed methodology is experimentally validated as well.

Table 7.3: Performance metrics for different forecasting models on the Canadian unemployment rate (monthly) data

Model	1-Year ahead forecast			3-Year ahead forecast		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA(3,1,3)	0.133	0.115	1.623	0.847	0.685	9.708
ANN(10)	0.137	0.117	1.095	0.837	0.614	9.365
ARNN(15,8)	0.126	0.113	1.084	0.801	0.613	9.247
SVM($\gamma = 0.5$)	0.273	0.248	1.915	0.998	0.740	10.92
Hybrid ARIMA(3,1,3)-SVM($\gamma = 0.5$)	0.145	0.135	1.135	0.835	0.711	9.595
Hybrid ARIMA(3,1,3)-ANN(5)	0.118	0.108	1.017	0.638	0.615	8.387
Hybrid ARIMA(3,1,3)-ARNN(5,3)	**0.106**	**0.098**	**0.838**	**0.627**	**0.601**	**8.017**

Table 7.4: Performance metrics for different forecasting models on the Germany unemployment rate (monthly) data

Model	1-Year ahead forecast			3-Year ahead forecast		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA(1,2,1)	0.098	0.092	1.490	0.361	0.303	5.177
ANN(10)	0.127	0.120	4.295	0.564	0.505	7.394
ARNN(5,3)	0.104	0.099	6.783	0.569	0.533	6.365
SVM($\gamma = 0.5$)	0.101	0.099	1.594	0.566	0.509	6.272
Hybrid ARIMA(1,2,1)-SVM($\gamma = 0.5$)	0.090	0.089	1.537	0.360	0.305	5.120
Hybrid ARIMA(1,2,1)-ANN(5)	0.082	0.096	1.558	0.306	0.297	4.243
Hybrid ARIMA(1,2,1)-ARNN(1,1)	**0.077**	**0.071**	**1.068**	**0.300**	**0.291**	**4.156**

Table 7.5: Performance metrics for different forecasting models on the Netherlands unemployment rate (monthly) data

Model	1-Year ahead forecast			3-Year ahead forecast		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA(1,2,1)	0.144	0.119	2.003	0.306	0.270	5.671
ANN(10)	0.228	0.174	2.906	1.304	1.048	17.935
ARNN(14,7)	0.249	0.192	3.177	0.938	0.784	14.518
SVM($\gamma = 0.5$)	0.228	0.174	2.906	1.304	1.048	17.935
Hybrid ARIMA(1,2,1)-SVM($\gamma = 0.5$)	0.145	0.120	2.023	0.308	0.272	5.706
Hybrid ARIMA(1,2,1)-ANN(5)	0.143	0.118	2.002	0.306	0.270	5.668
Hybrid ARIMA(1,2,1)-ARNN(12,6)	**0.140**	**0.114**	**1.192**	**0.300**	**0.264**	**5.529**

Table 7.6: Performance metrics for different forecasting models on the New Zealand unemployment rate (quarterly) data

Model	3-Year ahead forecast			8-Year ahead forecast		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA(1,1,2)	1.121	1.118	13.779	1.985	1.850	24.775
ANN(10)	1.630	1.461	18.398	4.796	4.167	40.109
ARNN(12,6)	1.329	1.280	17.073	1.895	1.797	24.863
SVM($\gamma = 0.5$)	1.329	1.280	17.073	1.895	1.797	23.940
Hybrid ARIMA(1,1,2)-SVM($\gamma = 0.5$)	1.059	1.028	14.249	1.978	1.842	25.182
Hybrid ARIMA(1,1,2)-ANN(5)	1.019	0.986	13.748	1.941	1.820	24.763
Hybrid ARIMA(1,1,2)-ARNN(10,5)	**0.998**	**0.944**	**11.272**	**1.318**	**1.239**	**22.992**

Table 7.7: Performance metrics for different forecasting models on the Sweden unemployment rate (monthly) data

Model	1-Year ahead forecast			3-Year ahead forecast		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA(3,1,1)	0.194	0.155	2.323	0.384	0.305	4.458
ANN(10)	0.257	0.282	2.895	0.324	0.412	**3.797**
ARNN(21,11)	0.253	0.178	3.275	**0.324**	**0.288**	4.336
SVM($\gamma = 0.5$)	0.396	0.358	4.253	0.489	0.410	5.525
Hybrid ARIMA(3,1,1)-SVM($\gamma = 0.5$)	0.198	0.157	2.252	0.391	0.312	4.544
Hybrid ARIMA(3,1,1)-ANN(5)	0.190	0.157	2.244	0.387	0.309	4.502
Hybrid ARIMA(3,1,1)-ARNN(16,8)	**0.189**	**0.151**	**2.024**	0.363	0.298	4.231

Table 7.8: Performance metrics for various forecasting models on Switzerland unemployment rate (monthly) data

Model	1-Year ahead forecast			3-Year ahead forecast		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA(2,1,2)	0.047	0.037	1.095	0.437	0.314	9.365
ANN(10)	**0.026**	0.023	1.094	0.526	0.433	11.176
ARNN(7,4)	0.027	0.028	1.715	0.498	0.340	10.924
SVM($\gamma = 0.5$)	0.045	0.035	1.135	0.535	0.511	11.295
Hybrid ARIMA(2,1,2)-SVM($\gamma = 0.5$)	0.040	0.038	1.117	0.438	0.315	9.387
Hybrid ARIMA(2,1,2)-ANN(5)	0.026	0.024	1.090	0.435	0.310	9.273
Hybrid ARIMA(2,1,2)-ARNN(3,2)	**0.026**	**0.022**	**1.038**	**0.427**	**0.301**	**8.917**

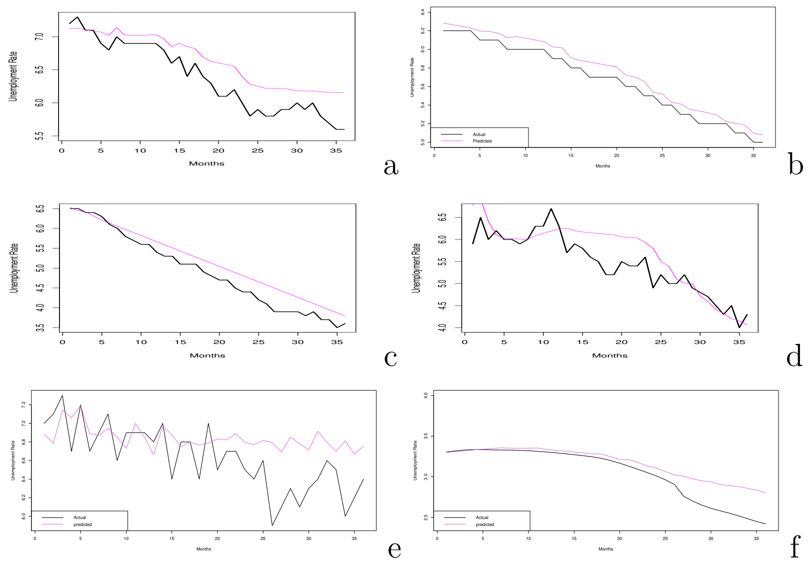


Figure 7.2: Actual vs Predicted (based on hybrid ARIMA-ARNN model) forecasts for the test data sets of the Canada (a), Germany (b), Netherlands (c), New Zealand (d), Sweden (e) and Switzerland (f) unemployment rate data sets

7.6 Simulation Study

A time series data have been synthesized in such a way that the mean between multiple segments in both the test and training data differ. Refer to Table 7.9 for the visualization of the same. The data consists of 165 points out of which 15 data points are kept as test samples (red-colored samples in the figure given in Table 7.9). We randomly draw the required number of components from a uniform distribution with varying means. We have used an R package 'GRATIS' for the time series generation which is available from `https://github.com/ykang/gratis` and a detailed description of the data generation process is given in Kang et al. (2020).All the models discussed in Section 7.5 are trained on the data set and results are reported in Table 7.11.

Table 7.9: Synthesized data set and corresponding ACF and PACF plots

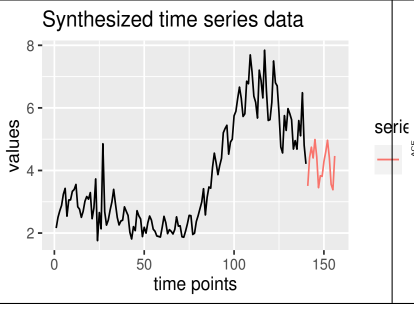
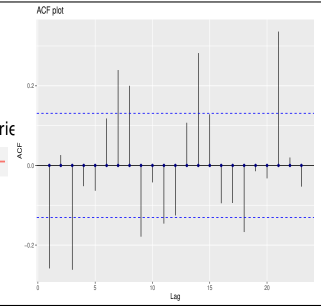
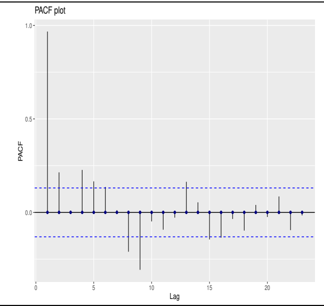
Data	ACF plot	PACF plot
		

Table 7.10: R functions and packages for implementation.

Model	R function	R package	Reference
ARIMA	auto.arima	forecast	Hyndman and Khandakar (2007)
SVM	svm	e1071	Kourentzes (2017)
ANN	mlp	nnfor	Kourentzes (2017)
ARNN	nnetar	forecast	Hyndman and Khandakar (2007)
Hybrid models	-	-	`https://github.com/tanujit123`

We now study some basic characteristics of this synthetically generated time series. KPSS tests are performed to examine the stationarity of a given time series (Kwiatkowski et al., 1992). The null hypothesis for the KPSS test is that the time series is stationary. Thus, the series is nonstationary when the p-value is less than a threshold. The synthesized series can be characterized as non-stationary as the p-value < 0.01. The value of Skewness and Kurtosis of this data is 0.4971 and - 0.7465, respectively. Among the single models, ARIMA(2,1,4) performs 'best' in terms of accuracy metrics for 15-points ahead forecasts. ARNN(16,8) also has competitive accuracy metrics. ARIMA residuals are trained with an ARNN(8,4) model to obtain the final prediction results of the hybrid ARIMA-ARNN model. Hybrid ARIMA-ARNN model improves the earlier ARIMA forecasts and has the best accuracy among all single and hybrid models (see Table 7.11). We have used the default parameters available in R statistical packages while implementing rest of the component models and the values for the tuning parameters are reported in Table 7.11. In-sample and out-of-sample forecasts obtained from ARIMA and hybrid ARIMA-ARNN models are depicted in Figure 7.3. Out-of-sample forecasts are generated using the rest of the synthesized data set as training data. A detailed summary of the implementation tools is presented in Table 7.10. The experimental results based on the analysis of the synthesized data are presented in Table 7.11 and it is clear from the table that the proposed hybrid ARIMA-ARNN model works better than the state-of-the-art models.

Table 7.11: Performance metrics with 15 points-ahead test set for synthesized data. Figures in ( ) indicate the values of the tuning parameters for each of the forecasting models.

Model	15-points ahead forecast			
	RMSE	MAE	MAPE	SMAPE
ARIMA(2,1,4)	0.718	0.609	1.689	1.607
ANN(10)	0.967	0.838	2.515	2.169
ARNN(16,8)	0.763	0.664	1.975	1.742
SVM($\gamma = 0.5$)	0.841	0.669	1.748	1.768
Hybrid ARIMA(2,1,4)-SVM($\gamma = 0.5$)	0.658	0.521	1.433	1.380
Hybrid ARIMA(2,1,4)-ANN(5)	0.711	0.605	1.690	1.599
Hybrid ARIMA(2,1,4)-ARNN(8,4)	**0.597**	**0.465**	**1.322**	**1.245**

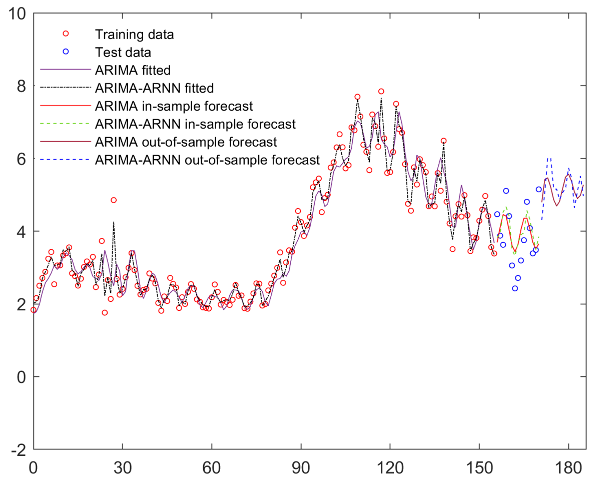


Figure 7.3: Plots of the proposed forecasting model for training, testing, and 15-points ahead forecast results on synthesized data.

7.7 Economic Implications and Conclusions

Forecasting the unemployment rate is very important for financial market participants and is a reliable indicator of labor market conditions. The release of the monthly (quarterly) unemployment rate for a country is one of the most important regular economic events for market participants. The effect of the unemployment news on stock returns is not so straightforward because of the relative importance of information on labor market conditions and monetary policy changes over time depending on the state of the economy. Considering the difficulty in predicting market reactions, forecasting the unemployment rate accurately is highly useful for investors to hedge the market risk arising from the unexpected change in business conditions and monetary policy.

In this study, we proposed a hybrid ARIMA-ARNN model using an error modeling approach that performs considerably well for unemployment rates forecasting for countries like Canada, Germany, Netherlands, New Zealand, Sweden, and Switzerland. The proposed hybrid ARIMA-ARNN model filters out linearity using the ARIMA model and predicts nonlinearities present in the error residuals with an ARNN model. The proposed hybrid ARIMA-ARNN model can explain the linear and nonlinear tendencies present in the unemployment rate data sets of developed countries better as compared to the traditional single and other hybrid models. It also yields better forecast accuracy than various single and hybrid models for most of the data sets considered in this study. Even we have experimented on simulated time series data sets to show the excellent performance of the proposed hybrid model over state-of-the-art models. The proposed model is also tested on simulated time series data set and compared with available individual and hybrid forecasting models. The results are very promising and it shows that the proposed model can perform in complex univariate time series data set involving forecasting tasks.

The proposal will be useful for macroeconomists, policymakers, and econometricians working in the field of government decision and policy makings. However, any econometric phenomena can fluctuate heavily because of various external factors over time. And those fluctuations are generally challenging to be appropriately captured for accurate forecasting. However, the proposed model may still predict with better accuracy provided the conditions stated in the main result for asymptotic stationarity of the hybrid model are satisfied. It is to be noted that over the last four decades, the unemployment rates for most of the countries considered in this study had no consistent trend at all and has asymmetrical cyclical movements. The best short-term forecasts and long-term forecasts of monthly and quarterly unemployment rate data sets are obtained using the proposed hybrid model as compared to other competitive methods. An immediate extension of this work is to see the model's application for seasonal unemployment rate data sets.

Advanced neural network models (say, recurrent neural networks (RNN), long short-term memory (LSTM) networks, and deep neural net) are highly complex, over-parameterized models and found useful when the data sets are very large (like image, audio, and video data sets) (Dunson, 2018). Since the number of data points in both the data sets used in this chapter is very limited, advanced deep learning techniques will over-fit the data sets, thus not been included in this chapter. One possible area for future research would be to expand this hybrid approach for multivariate time series forecasting problems that arise in various applied domains.

Chapter 8

Conclusions

Summary

In this chapter, we discuss the overall contributions made in this thesis and how these contributions serve as possible solutions to various applied problems in the fields of business analytics, quality control, macroeconomics, and software reliability, among many others. We also proceed to identify some of the possible future research areas stemming from the developed nonparametric hybrid methods in this thesis that can be useful in the field of data science. The identified future research scopes encompass adversarial machine learning as well as deep learning paradigms.

8.1 Contribution of the Thesis

Our primary focus in this thesis is on the development of specialized nonparametric hybrid models for various applied problems, drawn from the fields of business analytics, process control, quality prediction, macroeconomics, and software reliability engineering. In Chapters 1-2, we undertook an in-depth perusal of the existing methods for handling the various kinds of data science problems. In each of the chapters from Chapter 3 to Chapter 6, we focus on the implicit learning strategies, in particular, on the hybridization of the tree-based methods with that of neural network-based methods. In Chapter 7, we discuss the hybridization of linear and nonlinear forecasting models in a macroeconomic context. We identified some of the principal shortcomings of the current hybrid approaches which are shown below.

- A lot of research efforts have been dedicated to developing hybrid methodologies from the system's designer point of view. However, none of the existing methods present theoretical (statistical) interpretations of these hybrid predictive models regardless of their uses in practical problems of classification, regression, etc.

- The existing approaches in the field of hybrid or ensemble systems are mostly restricted on relatively balanced well-structured pattern classification and general nonparametric regression estimation data sets. These hybrid systems become infeasible for high-dimensional moderate sample-sized data sets involving both the feature selection and prediction tasks. Therefore, there is a need to develop new hybrid techniques for complex situations arising in the domain of quality, economics, software reliability, and business analytics, to name a few, with data irregularities.

- Research in the underlying hybrid algorithms is far from done. If one needs to use scalability, accuracy, robustness, statistical interpretation, and easy interpretability as the criteria to judge these hybrid techniques, no existing models can simultaneously excel in all criteria. Therefore, there is a need to extend the available approaches and design some novel hybrid models that are scalable, robust, accurate, statistically sound (have desired asymptotic properties), and easily interpretable.

- Lastly, most of the existing methods in the hybrid literature have considered two or more frequentist methods while creating the hybridization. However, there is a scope to extend hybrid methods to blend two or more different statistical paradigms, namely frequentist and Bayesian methods.

In Chapters 3-7, we have presented several novel hybrid methods for addressing the shortcomings of the current hybrid literature and applied these methods to the problems drawn from the fields of business analytics, manufacturing process control, process quality improvements, unemployment rate forecasting, and software defect predictions. In Chapters 3 and 4, we developed hybrid methodologies to deal with problems of feature selection cum pattern classification and imbalanced pattern classification, respectively. Some other hybrid methods were introduced in the context of the regression estimation problems and Bayesian nonparametric regression problems in Chapter 5 and Chapter 6, respectively. Finally, in Chapter 7, we introduced a hybrid approach combining linear and nonlinear models for time series forecasting.

In Chapter 3, we began with a motivating problem of a private business school that would like to admit a handful number of students whose probability of placement at the end of the Master's program is very high. The basis of the decision-making process is based on past students' data available to the business school. The business school administration wants to come up with a model that can help them select the essential features from various available academic characteristics of students and model it accordingly. We formulated this applied problem into a feature selection cum prediction problem and developed a hybrid model based on classification trees and artificial neural networks to solve the problem. In the first stage of the model, we employed classification trees that could suggest an essential set of features and precise classification of the business school data set.

In the second stage of the model, the important set of features along with the tree-predicted results were further trained with the neural net model, and final results were obtained. The hybrid CT-ANN model was theoretically shown to be consistent and an upper bound on the number of hidden neurons in the latter stage of the model was derived for providing a 'white-box-like' interpretation of the proposed framework. The hybrid CT-ANN model was applied to the business school data set and found very useful in terms of various performance metrics including classification accuracy. The robustness of the proposed hybrid approach was shown by applying it to other standard data sets from medical domains and toy data sets. The experimental findings suggested the efficacy and broad applicability of the proposed model.

In Chapter 4, we addressed another challenging problem from the field of imbalanced pattern classification. Traditional statistical learning algorithms perform poorly when the data sets are skewed and exhibit an unequal class distribution. An example of the imbalanced classification problem was given with software defect prediction data sets from the field of software reliability engineering. A novel hybrid methodology, Hellinger net, was developed in this chapter to handle the curse of imbalanced data sets. Hellinger net maps Hellinger distance decision tree to a two-hidden layered artificial neural network using the idea of soft pruning while training the architecture. The asymptotic results for the proposed Hellinger net model were presented, which gives the theoretical robustness of the approach presented in this chapter. Experimental validation of the Hellinger net method on software defect prediction (SDP) data sets compared with state-of-the-art techniques showed excellent performance of the model when data exhibits skew-class distribution. The general applicability of the proposed Hellinger net was shown by applying it to standard UCI imbalanced data sets from various applied domains and also on simulated data sets.

In Chapter 5, we moved our focus towards another wing of supervised learning problems, viz. regression estimation problems. The primary motivation of this chapter came from the process efficiency improvement problem in a modern paper manufacturing company. The issue of fiber-filler recovery process improvement in the paper company was first framed into a nonparametric regression estimation problem. Further, a solution methodology, namely radial basis neural tree, was developed in this chapter to capture the relationship between the recovery percentage of the fiber-filler recovery equipment with that of the process parameters as decided by preliminary statistical analysis and the process experts. An idea of parameter optimization in the hybrid model, along with its asymptotic consistency, were shown in this chapter to provide a strong statistical background for the proposed model. The RBNT model, when applied to the problem of process efficiency improvement of the paper manufacturing company, had shown outstanding results compared to other state-of-the-art methods. We also tested the asymptotic behavior of the proposed RBNT model on the simulated regression data set. The model was further extended in the next chapter where we used the same analogy to combine two contrasting paradigms (frequentist and Bayesian methods) of statistical science.

In Chapter 6, we built a hybrid model combining two self-contrasting paradigms in a joint framework. Decision trees and neural nets have both the frequentist and Bayesian counterparts, and situations exist when one approach is preferred over the other. We created two hybridization based on frequentist versions of decision trees (neural networks) and Bayesian versions of neural nets (decision trees), which can utilize the potential benefits of two ideologically different paradigms and overcome their drawbacks. The algorithms presented in this chapter, we called them 'Bayesian neural trees' (BNT), have significant benefits, such as fewer tuning parameters than advanced neural nets and white-box interpretability. The proposed BNT models attained the desired asymptotic properties under certain regularity conditions and, when applied to various regression data sets (for example, water quality prediction), performed superior to other state-of-the-art methods.

In Chapter 7, we considered the problem of unemployment rate forecasting from the time series literature. Unemployment has always been a very focused issue causing the nation to lose its economic and financial contribution. Unemployment rate data sets are mostly nonstationary and nonlinear in nature. We proposed a hybrid approach based on linear ARIMA and nonlinear ARNN models that can predict the unemployment rates more accurately. The hybrid approach is an integration of two distinct models for generating better forecasts for the test data sets. We derived the results for the asymptotic stationarity of the proposed hybrid approach using Markov chains and nonlinear time series analysis techniques. The application of the proposed approach to six unemployment rate data sets from various countries, namely, Canada, Germany, Netherlands, New Zealand, Sweden, and Switzerland, showed the superiority of the developed hybrid model.

The ideas, concepts, and methods presented in this thesis attempt to address a wide variety of applied data science problems. The developed methods like the hybrid CT-ANN model in Chapter 3 can improve predictions for both feature selection and classification problems. Additionally, the Hellinger net method in Chapter 4 presents a way to deal with the imbalanced class scenario in software defect prediction problems. The RBNT model, as described in Chapter 5, is useful for nonparametric regression problems. In Chapter 6, the blends of Bayesian and frequentist frameworks are presented, which has a wide range of applications in regression problems from diverse fields. Finally, Chapter 7 describes a hybrid time series forecasting model that generated superior forecasts of unemployment rates for different countries. All these hybrid models presented in this thesis have the desired statistical properties, are robust in nature, and are easily interpretable. Continuing research in the line of this thesis is needed to expand and adapt these hybrid predictive models so that they can operate on a richer collection of data types. Data is no longer just numerical or discrete. It can be unstructured text, video clips, or audio clips, and the amount of information available is also growing amazingly. Thus, the new developments of scalable and automated predictive learning techniques for extracting useful knowledge from a diverse source of data sets will be motivating thrust areas for future research as we move forward.

The interpretability of predictive machine learning models is important, especially in cases where ethics are involved, such as law, medicine, and finance; and other critical applications where we wish to manually verify the correctness of a model's reasoning. In some areas like Business Intelligence, it is often more important to know how each factor contributes to the prediction rather than the conclusion itself. With machine learning-based predictions becoming ubiquitous and affecting many aspects of our daily lives, the focus of research moves beyond model performance (e.g., efficiency and accuracy) to model interpretability (Doshi-Velez and Kim, 2017; Weller, 2017). This is particularly so in applications where there are ethical (Bostrom and Yudkowsky, 2014) or safety concerns and models' predictions should be explainable in order to verify the correctness of their reasoning process or justify their decisions. The hybrid models developed from Chapter 3 to Chapter 7 are scalable (the size of the data does not pose a problem), robust (work well in a wide variety of problems), accurate (achieve higher predictive accuracy), statistically sound (have desired asymptotic properties), and interpretable. However, the (limited) explainability of these hybrid models arises in the form of transparency in the context of human interpretation of algorithms, noting their benefits, motivations, difficulties for measurement, and potential concerns. There are now a number of attempts to make models explainable. Some are model-agnostic (Ribeiro et al., 2016), while most are associated with a certain type of model, e.g., rule-based classifiers (Dash et al., 2015; Malioutov et al., 2017) and neural networks (Kim et al., 2016). Hence, there is a need for future work to develop some new explanation techniques that can explain the predictions of the hybrid models in a more interpretable and faithful manner, by learning an interpretable model locally around the prediction. Such understanding will provide insights into the model, which can be used to transform an untrustworthy model or prediction into a trustworthy one. Furthermore, the hybrid models presented in this thesis opens a broader scope for future research in the direction of Bayesian nonparametrics, adversarial machine learning, and deep learning. There is also the possibility of applying a hybrid approach to software defect prediction problems using Poisson processes. We discuss some motivating thrust areas for future research in the next section.

8.2 Future Scope of Study

In this section, we attempt to identify some of the possible future scopes of study in which the research work presented in this thesis can be extended. We lay stress on the possible extension of our proposed methods on the problem of data shift, missing data problem, adversarial classification problem, Bayesian nonparametrics, and software reliability modeling.

8.2.1 Addressing Covariate Shift when Data is Imbalanced

In the present thesis, we have addressed the 'curse of imbalanced data sets' in which there is an under-represented class and a majority class in Chapter 4. However, data sets may arise where the training and test data distributions are different in the data mining paradigm, thus leading to inaccurate conclusions when building a model from the training data. This issue is popularly known as data set shift problems in the machine learning literature (Quionero-Candela et al., 2009), or more explicitly covariate shift problems (Hofer and Krempl, 2013). In this case, the attribute values have different distributions between training and test sets (López et al., 2014). In the presence of imbalance, the problem is even more critical. This problem occurs in biomedical and software engineering where data set shifts in the form of imbalanced data and covariate shifts are common. Out of several possible causes for data set shift, some most important causes are sample selection bias and nonstationary data environments (Moreno-Torres et al., 2012). In the former case, the discrepancy in distribution is due to the fact that the training examples have been obtained through a biased method, and thus do not represent reliably the operating environment where the classifier is to be deployed (test set). It commonly occurs, among others, in software defect prediction data sets in which due to cost concerns, one of the classes is sometimes sampled at a lower rate than it actually appears. The latter case arises when the training and test environments are different, which could be due to temporal or spatial change. It commonly appears in adversarial classification problems such as spam detection, fraud detection, and image recognition (Laskov and Lippmann, 2010). A possible solution may be to extend the idea of the Hellinger net presented in Chapter 4 with that of "distributional-optimally balanced stratified cross-validation" (DOB-SCV) approach introduced by López et al. (2014). The DOB-SCV method attempts to minimize covariate shifts in data by keeping data distribution as similar as possible between training and testing folds by maximizing diversity on each fold while requiring that the folds resemble each other as closely as possible. Hence, one exciting avenue of future research may be to employ our proposed Hellinger net along with DOB-SCV on real-world data sets that suffer from the problem of class imbalance in the presence of covariate shift.

8.2.2 Handling Data Irregularities with Hybrid Methods

In this thesis, we addressed some issues that arise in the field of data science, such as feature selection cum classification problem in Chapter 3, regression estimation problem in Chapter 5 with possible extension for Bayesian paradigms in Chapter 6, and nonlinear time series forecasting in Chapter 7. But due to the unprecedented success of modern deep learning methods, we lay stress on possible amalgamations between our proposed methods and state-of-the-art deep learning techniques. Deep learning methods utilize neural networks having multiple hidden layers to learn useful representations of data. Comprehensive reviews of well-known deep learning architectures, including autoencoders, convolutional neural network (CNN), deep belief network (DBN), and restricted Boltzmann machine, can be found in Goodfellow et al. (2016); Zhang et al. (2019b). Gondara and Wang (2017) suggested the use of deep denoising autoencoders for multiple imputations of continuous, categorical, and mixed data types with various missing features. Zhong et al. (2016) proposed a field-effect deep network (FEDN) for image recognition with missing features. An interesting avenue of research will be to employ our proposed hybrid CT-ANN model in the deep learning frameworks to analyze incomplete data sets. Several recent works built deep neural network-based decision trees and random forests, which can utilize the potential benefits of both the deep neural nets and tree-based algorithms (Dong et al., 2018; Feng and Zhou, 2018; Humbird et al., 2018; Kontschieder et al., 2015; Yang et al., 2018). Thus, our proposed hybrid approaches can also be extended in a deep learning framework that can be thought of as a broad generalization of the hybrid approaches presented in this thesis. This will be useful for handling the dependency structures, missing data features, and will be computationally very useful with applications in image captioning, image recreation, and natural language processing problems.

8.2.3 Building Hybrid Models for Adversarial Machine Learning Problems

State-of-the-art machine learning (ML) algorithms perform extraordinarily well on standard data but have recently been shown to be vulnerable to adversarial examples, data instances targeted at fooling those algorithms (Goodfellow et al., 2015). Predictive modeling techniques are widely used in security settings, email spam detection systems, medical fraud assessment, and computer vision in which data can be deliberately manipulated by an adversary trying to evade detection and achieve some benefit (Naveiro et al., 2019). However, most traditional classifiers are not robust to such data modifications (Dalvi et al., 2004). A few methods have been proposed to robustify classification algorithms in adversarial environments. Most of them have focused on application-specific domains, such as spam detection (Kołcz and Teo, 2009), medical fraud assessment (Ekin et al., 2018) and computer vision (Elsayed et al., 2018). Some of the approaches to such problems have focused on game-theoretic ideas with strong underlying common knowledge assumptions (Zhou et al., 2019), which are not realistic in the security realm whereas Gallego et al. (2020) provides an alternative Bayesian framework that accounts for the lack of precise knowledge about the attacker's behavior using adversarial risk analysis. (Naveiro et al., 2019) apply adversarial risk analysis (ARA) (see also Banks et al. (2020)) to the emerging field of adversarial machine learning (AML). In particular, it shows how to protect statistical classification systems from attackers trying to fool them by intentionally modifying input data in search of a benefit. This provides new tools in the field of AML which has been previously based on standard game-theoretic approaches. AML is an emerging field aimed at the protection of automated ML systems against security threats and machine learning models are adapted to the adversarial case (Rios Insua et al., 2020). In recent work, Cheng et al. (2020) proposed an advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modeling. Thus, our proposed hybrid CT-ANN model in Chapter 3 can be extended in this framework and there is a scope of future work to introduce new hybrid models to deal with adversarial attacks in the machine learning domain.

8.2.4 Combining the Poisson Processes with Hybrid Methods for the Software Defect Problems

Software defect prediction has been a significant research topic in software engineering for the last 30 years (Xie, 1995), with studies concentrating on estimating how many defects remain in a system, identifying possible associations between defects, and revealing the defect proneness of software systems (Andreou and Chatzis, 2016). Tracking and predicting quality and reliability is a major challenge in large and distributed software development projects. A number of standard distributions have been successfully used in reliability engineering theory and practice, common among these for modeling software defect inflow being exponential, Weibull, and beta distributions (Rana et al., 2016). Another line of research in reliability prediction is the use of Bayesian modeling and the Non-Homogeneous Poisson Process (NHPP). Kuo et al. (1997) presented Bayes inference methodology for NHPP models with S-shaped mean value functions. Similarly, Li and Meeker (2014) used NHPP to model the distribution and found that new parameters are required to balance the release readiness decision with the reliability criteria of software release. Yet another approach is to use the time-related analyses of defect inflow. Zhou and Davis (2005) analyzed time-related bug reporting patterns of eight popular open-source projects and found that the

Weibull distribution family is the most correct one for these projects; they showed that open source projects exhibited similar reliability growth patterns as the proprietary projects. This indicated that the NHPP models can have alternatives and therefore the study of Zhao (2006) proposed to use the beta distribution to indicate software testability. The author shows theoretically that testing effort and test values can be simultaneously expressed through the distribution. However, no validation is provided using empirical data sets. The nature of the data, i.e., rare defect events, might induce to use of Poisson or Self-exciting processes, possibly different in different modules and with intensity depending on covariates (Ruggeri et al., 2008). Thus, it becomes important to use the information on the number and time of events to study also the temporal evolution of the process. Therefore, a hybrid model can possibly be obtained combining the Poisson process with the proposed Hellinger net method in Chapter 4. The temporal aspect is very important since software developers are aiming to reduce defect events and there is future scope of study to apply our proposed hybrid method along with Poisson processes to the SDP problems.

8.2.5 Developing Bayesian Deep Neural Network driven by Recursive Gaussian Processes

In the present thesis, we discussed a novel framework combining frequentist and Bayesian machine learning methods for nonparametric regression tasks. Possible future work in this direction would be to improve Bayesian deep neural networks using recursive Gaussian processes (Lee et al., 2017) with automatic relevance determination (ARD) prior (Wipf and Nagarajan, 2008) that takes care of important feature selection problem out of all the regressor variables and induces sparsity.

Gaussian process behavior arises in many practical situations and current Gaussian process-based Bayesian deep neural networks also demonstrate its utility (Kwon et al., 2020; Matthews et al., 2018). The method proceeds by beginning with a linear Gaussian framework and then proceeds recursively such that the Gaussian process framework is preserved asymptotically even in the hidden layers. This seems to be less flexible than required in the sense that it fails to adequately address the uncertainties at every layer. Indeed, uncertainties, hence non-linear variations in the hidden layers, are expected to increasingly shift the initial Gaussian process to non-Gaussian processes. Future scope of the study will be to introduce some novel Bayesian methodology for deep neural networks based on general Gaussian process priors that coherently and satisfactorily address this issue. Given the data, all the unknown quantities can be learned in a fully Bayesian framework, using a 'look-up table' idea (Ghosh et al., 2014) combined with an efficient Markov Chain Monte Carlo (MCMC) methodology (Zhong and Ghosh, 2016). This will be useful to generalize Bayesian deep learning and can be employed in challenging big data problems.

In supervised neural network set up, we have T hidden layers and input vector $\underline{x} = (x_1, x_2, \ldots, x_p)'$, t^{th} hidden layer $h^{(t)}$ of dimension k_t, $t = 0, \ldots, T$ and output layer $\underline{y} = (y_1, y_2, \ldots, y_q)$. We set $h_j^{(0)} = w_j^{(0)'}\underline{x} + b_j^{(0)}$ for $k = 1, \ldots, k_0$ and $h_j^{(t)} = g^{(t)}\left(w^{(t)'}h_j^{(t-1)} + b_j^{(t)}\right)$, for $j = 1, \ldots, k_t$ and $t = 1, \ldots, T$ and $y_j = f(h_j^{(T)})$; $j = 1, \ldots, q$, as the final output. Let $g^{(t)}(w^{(t)'}h^{(t-1)} + b^{(t)}) = g(t, w^{(t)'}h^{(t-1)} + b^{(t)})$, where $g(\cdot,\cdot)$ is an appropriate Gaussian process. Also, $f(\cdot)$ is another appropriate Gaussian process (or transformation of Gaussian process, if y needs to be bounded, positive, etc.). The process $g(\cdot,\cdot)$ may also be some appropriate transformation of the underlying Gaussian process. Weights $w^{(t)}$ and bias $b^{(t)}$ will be assigned appropriate priors. An useful prior choice of $w^{(t)}$ seems to be the ARD prior which takes care of important feature selection problem out of all the regressor variables and induces sparsity. The function $g(t,\cdot)$ is the random function that generalizes the activation function which is considered fixed in classical neural network model. Usually, the activation function is modeled by the sigmoidal function. Although classical neural networks uses the same activation function for all the hidden layers, it makes sense to allow the activation function to vary with different layers. The random function $g(t,\cdot)$, which varies randomly with each hidden layer t, can provide a broad generalization to the existing strategies of Bayesian nonparametrics in the context of neural networks.

In a nutshell, this thesis contributes to the development of some hybrid predictive models from both the theoretical and applied viewpoints with potential applications in a wide range of applied fields, ranging from process control, software reliability engineering to business analytics, and macroeconomic data analysis. We also highlighted the need for future research in different directions.

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016. 67, 99

S. Abpeikar, M. Ghatee, G. L. Foresti, and C. Micheloni. Adaptive neural tree exploiting expert nodes to classify high-dimensional data. *Neural Networks*, 124:20–38, 2020. 6, 39, 42

R. Adhikari and R. Agrawal. A combination of artificial neural network and random walk models for financial time series forecasting. *Neural Computing and Applications*, 24(6):1441–1449, 2014. 7

P. S. Akash, M. E. Kadir, A. A. Ali, and M. Shoyaib. Inter-node hellinger distance based decision tree. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1967–1973, 2019. 4, 83, 84, 85

C. H. Aladag, E. Egrioglu, and C. Kadilar. Forecasting nonlinear time series with a hybrid methodology. *Applied Mathematics Letters*, 22(9):1467–1470, 2009. 157

C. Aldrich, C. Marais, B. Shean, and J. Cilliers. Online monitoring and control of froth flotation systems with machine vision: A review. *International Journal of Mineral Processing*, 96(1-4):1–13, 2010. 119

R. Aler, J. M. Valls, and H. Boström. Study of hellinger distance as a splitting metric for random forests in balanced and imbalanced classification datasets. *Expert Systems with Applications*, 2020. 6, 81, 84

A. S. Andreou and S. P. Chatzis. Software defect prediction using doubly stochastic poisson processes driven by stochastic belief networks. *Journal of Systems and Software*, 122:72–82, 2016. 183

D. Antanasijević, V. Pocajt, A. Perić-Grujić, and M. Ristić. Multilevel split of high-dimensional water quality data using artificial neural networks for the prediction of dissolved oxygen in the danube river. *Neural Computing and Applications*, 32:3957–3966, 2020. 148

S. Arora and J. W. Taylor. Forecasting electricity smart meter data using conditional kernel density estimation. *Omega*, 59:47–59, 2016. 157

I. S. Arvanitoyannis and T. H. Varzakas. Application of failure mode and effect analysis (fmea) and cause and effect analysis for industrial processing of common octopus (octopus vulgaris). *International Journal of Food Science & Technology*, 44(1):79–92, 2009. 121

R. Asif, A. Merceron, S. A. Ali, and N. G. Haider. Analyzing undergraduate students' performance using educational data mining. *Computers and Education*, 113:177–194, 2017. 48

A. Asuncion and D. Newman. Uci machine learning repository, 2007. 101

R. Avila, B. Horn, E. Moriarty, R. Hodson, and E. Moltchanova. Evaluating statistical model performance in water quality prediction. *Journal of Environmental Management*, 206:910–919, 2018. 148

D. Banks, V. Gallego, R. Naveiro, and D. Rios Insua. Adversarial risk analysis: An overview. *Wiley Interdisciplinary Reviews: Computational Statistics*, page e1530, 2020. 183

D. Barber and C. M. Bishop. Ensemble learning for multi-layer networks. In *Advances in Neural Information Processing Systems*, pages 395–401, 1998. 128

A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993. 60, 61

V. R. Basili, L. C. Briand, and W. L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, 22 (10):751–761, 1996. 80

G. W. Batchelder. Process for the demineralization of water, Mar. 2 1965. US Patent 3,171,799. 147

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139, 1999. 38

M. J. Bayarri and J. O. Berger. The interplay of Bayesian and frequentist analysis. *Statistical Science*, 19(1):58–80, 2004. 129

J. Berkson. Application of the logistic function to bio-assay. *Journal of the American statistical association*, 39(227):357–365, 1944. 3

B. Bhattacharya and D. P. Solomatine. Neural networks and m5 model trees in modelling water level–discharge relationship. *Neurocomputing*, 63:381–396, 2005. 108, 148

G. Biau, E. Scornet, and J. Welbl. Neural random forests. *Sankhya A*, 81(2): 347–386, 2019. 77, 84

D. R. Bickel. Blending Bayesian and frequentist methods according to the precision of prior information with applications to hypothesis testing. *Statistical Methods & Applications*, 24(4):523–546, 2015. 129

A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 299–310, 2010. 77

O. J. Blanchard and D. Leigh. Growth forecast errors and fiscal multipliers. *American Economic Review*, 103(3):117–20, 2013. 156

J. Bleich, A. Kapelner, E. I. George, and S. T. Jensen. Variable selection for bart: an application to gene regulation. *Annals of Applied Statistics*, 8(3):1750–1781, 2014. 131, 139

G. Boetticher. The promise repository of empirical software engineering data. *http://promise.site.uottawa.ca/SERepository/datasets-page.html*, 2007. 77, 80, 96

K. Boonchuay, K. Sinapiromsaran, and C. Lursinsap. Decision tree induction based on minority entropy for the class imbalance problem. *Pattern Analysis and Applications*, 20(3):769–782, 2017. 76

P. Boothe and D. Glassman. Comparing exchange rate forecasting models: Accuracy versus profitability. *International Journal of forecasting*, 3(1):65–79, 1987. 12

N. Bostrom and E. Yudkowsky. The ethics of artificial intelligence. *The Cambridge handbook of artificial intelligence*, 1:316–334, 2014. 180

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207, 2003. 2, 17, 18

G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons, 1976. 3, 160

A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997. 43

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 6

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 6, 67, 74, 80

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984. 3, 5, 27, 46, 47, 74, 80, 83, 84, 109, 128

R. P. Brent. Fast training algorithms for multilayer neural nets. *IEEE Transactions on Neural Networks*, 2(3):346–354, 1991. 77, 88, 89

B. M. Brentan, E. Luvizotto Jr, M. Herrera, J. Izquierdo, and R. Pérez-García. Hybrid regression model for near real-time urban water demand forecasting. *Journal of Computational and Applied Mathematics*, 309:532–541, 2017. 108

L. C. Briand, K. El Emam, B. G. Freimut, and O. Laitenberger. A comprehensive evaluation of capture-recapture models for estimating software defect content. *IEEE Transactions on Software Engineering*, 26(6):518–540, 2000. 74, 75, 77

P. J. Brockwell and A. Lindner. Strictly stationary solutions of autoregressive moving average equations. *Biometrika*, 97(3):765–772, 2010. 163, 165

V. G. Cancho, A. K. Suzuki, G. D. Barriga, and F. Louzada. A non-default fraction bivariate regression model for credit scoring: An application to brazilian customer data. *Communications in Statistics: Case Studies, Data Analysis and Applications*, 2(1-2):1–12, 2016. 108

G. Cao and L. Wu. Support vector regression with fruit fly optimization algorithm for seasonal electricity consumption forecasting. *Energy*, 115:734–745, 2016. 12

G. Casella and E. I. George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992. 30

O. Castillo and P. Melin. Hybrid soft computing models for systems modeling and control. *Encyclopedia of Complexity and Systems Science*, pages 4696–4713, 2009. 4, 6

C. Catal and B. Diri. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences*, 179(8):1040–1058, 2009. 80

H. B. Celikoglu. Travel time measure specification by functional approximation: application of radial basis function neural networks. *Procedia-Social and Behavioral Sciences*, 20:613–620, 2011. 109

H. B. Celikoglu and H. K. Cigizoglu. Modelling public transport trips by radial basis function neural networks. *Mathematical and computer modelling*, 45(3-4):480–489, 2007. 109

I. Chaabane, R. Guermazi, and M. Hammami. Enhancing techniques for learning decision trees from imbalanced data. *Advances in Data Analysis and Classification*, pages 1–69, 2019. 76, 84

S. Chakraborty. Bayesian additive regression tree for seemingly unrelated regression with automatic tree selection. In *Handbook of Statistics*, volume 35, pages 229–251. Elsevier, 2016. 31

S. Chakraborty, M. Ghosh, T. Maiti, and A. Tewari. Bayesian neural networks for bivariate binary data: An application to prostate cancer study. *Statistics in medicine*, 24(23):3645–3662, 2005. 35, 145

T. Chakraborty and A. K. Chakraborty. Hellinger net: A hybrid imbalance learning model to improve software defect prediction. *IEEE Transactions on Reliability*, 2020a. https://doi.org/10.1109/TR.2020.3020238. 15, 77

T. Chakraborty and A. K. Chakraborty. Superensemble classifier for improving predictions in imbalanced datasets. *Communications in Statistics: Case Studies, Data Analysis and Applications*, 6(2):123–141, 2020b. 15, 72, 77, 81, 84

T. Chakraborty, S. Chattopadhyay, and A. K. Chakraborty. A novel hybridization of classification trees and artificial neural networks for selection of students in a business school. *Opsearch*, 55(2):434–446, 2018. 14, 47, 62, 101

T. Chakraborty, A. K. Chakraborty, and S. Chattopadhyay. A novel distribution-free hybrid regression model for manufacturing process efficiency improvement. *Journal of Computational and Applied Mathematics*, 362:130–142, 2019a. 15, 109, 129

T. Chakraborty, A. K. Chakraborty, and Z. Mansoor. A hybrid regression model for water quality prediction. *Opsearch*, 56(4):1167–1178, 2019b. 16, 129

T. Chakraborty, A. K. Chakraborty, and C. A. Murthy. A nonparametric ensemble binary classifier and its statistical properties. *Statistics & Probability Letters*, 149:16–23, 2019c. 14, 47, 128, 129

T. Chakraborty, G. Kamat, and A. K. Chakraborty. Bayesian neural tree models for nonparametric regression. *arXiv preprint arXiv:1909.00515*, 2019d. *Under Review*. 16, 129

T. Chakraborty, A. K. Chakraborty, M. Biswas, S. Banerjee, and S. Bhattacharya. Unemployment rate forecasting: A hybrid approach. *Computational Economics*, 2020a. https://doi.org/10.1007/s10614-020-10040-2. 16, 157

T. Chakraborty, S. Chattopadhyay, and A. K. Chakraborty. Radial basis neural tree model for improving waste recovery process in a paper industry. *Applied Stochastic Models in Business and Industry*, 36:49–61, 2020b. 15, 109, 128, 129

K. S. Chan and H. Tong. On the use of the deterministic lyapunov function for the ergodicity of stochastic difference equations. *Advances in Applied Probability*, 17(3):666–678, 1985. 164, 166

N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pages 107–119. Springer, 2003. 98, 104

Y. Chen, A. Abraham, and J. Yang. Feature selection and intrusion detection using hybrid flexible neural tree. In *International Symposium on Neural Networks*, pages 439–444. Springer, 2005. 6, 38

Y. Chen, A. Abraham, and B. Yang. Feature selection and classification using flexible neural tree. *Neurocomputing*, 70(1-3):305–313, 2006. 6, 77

Y. Chen, B. Yang, and Q. Meng. Small-time scale network traffic prediction based on flexible neural tree. *Applied Soft Computing*, 12(1):274–279, 2012. 38

M. Cheng, F. Fang, C. Pain, and I. Navon. An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. *arXiv preprint arXiv:2003.10547*, 2020. 183

S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994. 80

H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 94(443):935–948, 1998. 4, 28, 29, 128

H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian treed models. *Machine Learning*, 48(1-3):299–320, 2002. 29, 128

H. A. Chipman, E. I. George, and R. E. McCulloch. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010. 4, 6, 29, 30, 31, 145, 147, 154

C. Chow. Statistical independence and threshold functions. *IEEE Transactions on Electronic Computers*, 1:66–68, 1965. 37

W. Chu, S. S. Keerthi, and C. J. Ong. Bayesian support vector regression using a unified loss function. *IEEE Transactions on Neural Networks*, 15(1):29–44, 2004. 4

D. A. Cieslak and N. V. Chawla. Learning decision trees for unbalanced data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 241–256. Springer, 2008. 31, 32, 76, 81, 83, 84

D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012. 4, 76, 83, 84, 88, 98

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995. 3

C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang. Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 874–883, 2017. 39, 41

T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on Electronic Computers*, (3):326–334, 1965. 50, 56, 114

N. Dalvi, P. Domingos, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004. 182

B. V. Dasarathy and B. V. Sheela. A composite classifier system design: concepts and methodology. *Proceedings of the IEEE*, 67(5):708–713, 1979. 37

S. Dash, D. M. Malioutov, and K. R. Varshney. Learning interpretable classification rules using sequential rowsampling. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3337–3341. IEEE, 2015. 180

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine learning*, pages 233–240. ACM, 2006. 43

J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009. 41

L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 1996. 3, 18, 19, 25, 33, 47, 51, 54, 57, 58, 61, 67, 90, 131, 132, 139

D. A. Dickey. Introduction to predictive modeling with examples. In *SAS Global Forum 2012*, 2012. 3

M. Dong, L. Yao, X. Wang, B. Benatallah, C. Huang, and X. Ning. Opinion fraud detection via neural autoencoder decision forest. *Pattern Recognition Letters*, 132:21–29, 2018. 7, 182

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017. 180

D. B. Dunson. Statistics in the big data era: Failures of the machine. *Statistics & Probability Letters*, 136:4–9, 2018. 5, 27, 33, 41, 46, 71, 175

P.-O. Edlund and S. Karlsson. Forecasting the swedish unemployment rate var vs. transfer function modelling. *International Journal of Forecasting*, 9(1):61–76, 1993. 12, 156, 167

T. Ekin, F. Ieva, F. Ruggeri, and R. Soyer. Statistical medical fraud assessment: exposition to an emerging field. *International Statistical Review*, 86(3):379–402, 2018. 183

C. El Moucary. Data mining for engineering schools predicting students performance and enrollment in masters programs. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 2(10):1–9, 2011. 48

G. Elsayed, S. Shankar, B. Cheung, N. Papernot, A. Kurakin, I. Goodfellow, and J. Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In *Advances in Neural Information Processing Systems*, pages 3910–3920, 2018. 183

S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, pages 524–532, 1990. 39

A. Faragó and G. Lugosi. Strong universal consistency of neural network classifiers. *IEEE Transactions on Information Theory*, 39(4):1146–1151, 1993. 47, 60, 61

J. Faraway and C. Chatfield. Time series forecasting with neural networks: a comparative study using the air line data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(2):231–250, 1998. 157, 161

J. Feng and Z.-H. Zhou. Autoencoder by forest. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 182

N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5):675–689, 1999. 74, 75, 77

A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera. Learning from imbalanced data streams. In *Learning from Imbalanced Data Sets*, pages 279–303. Springer, 2018a. 76

A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 2018b. 75, 76

R. A. Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8(4):376–386, 1938. 3

R. A. Fisher. The precision of discriminant functions. *Annals of Eugenics*, 10(1): 422–429, 1940. 3

E. Fix and J. L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley, 1951. 3

P. A. Flach. The geometry of roc space: understanding machine learning metrics through roc isometrics. In *Proceedings of the 20th International Conference on Machine Learning*, pages 194–201, 2003. 82

G. L. Foresti and T. Dolso. An adaptive high-order neural tree for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(2):988–996, 2004. 38, 42

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996. 6

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. 2

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991. 3

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 29(5):1189–1232, 2001. 6

N. Frosst and G. Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017. 6, 40, 41, 84, 109

M. Funke. Time-series forecasting of the german unemployment rate. *Journal of Forecasting*, 11(2):111–125, 1992. 156

Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1183–1192, 2017. 4

J. W. Galbraith and S. van Norden. Asymmetry in unemployment rate forecast errors. *International Journal of Forecasting*, 35(4):1613–1626, 2019. 156, 160

V. Gallego, R. Naveiro, A. Redondo, D. Rios Insua, and F. Ruggeri. Protecting classifiers from attacks. a bayesian approach. *arXiv preprint arXiv:2004.08705*, 2020. 183

F. Galton. *Natural inheritance*. Macmillan and Company, 1894. 3

D. Ganatra and U. Dinesh Kumar. A dean's dilemma: selection of students for the mba program. *Harvard Business Review, pp 1-6*, 2014. 48, 62

E. George, P. Laud, B. Logan, R. McCulloch, and R. Sparapani. Fully nonparametric bayesian additive regression trees. In *Topics in Identification, Limited Dependent Variables, Partial Observability, Experimentation, and Flexible Modeling: Part B*. Emerald Publishing Limited, 2019. 31

S. Ghosal and A. Van der Vaart. *Fundamentals of nonparametric Bayesian inference*, volume 44. Cambridge University Press, 2017. 4

A. Ghosh, S. Mukhopadhyay, S. Roy, and S. Bhattacharya. Bayesian inference in nonparametric dynamic state-space models. *Statistical Methodology*, 21:35–48, 2014. 184

M. Ghosh, T. Maiti, D. Kim, S. Chakraborty, and A. Tewari. Hierarchical bayesian neural networks: an application to a prostate cancer study. *Journal of the American Statistical Association*, 99(467):601–608, 2004. 35

S. Gmar, N. Helali, A. Boubakri, I. B. S. Sayadi, M. Tlili, and M. B. Amor. Electrodialytic desalination of brackish water: determination of optimal experimental parameters using full factorial design. *Applied Water Science*, 7(8): 4563–4572, 2017. 108, 148

L. Gondara and K. Wang. Recovering loss to followup information using denoising autoencoders. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1936–1945. IEEE, 2017. 182

L. Gong, S. Jiang, L. Bo, L. Jiang, and J. Qian. A novel class-imbalance learning approach for both within-project and cross-project defect prediction. *IEEE Transactions on Reliability*, 2019. 74, 78, 79, 81, 96, 98

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016. 4, 33, 42, 182

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 182

D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson. Using the support vector machine as a classification method for software defect prediction with static code metrics. In *International Conference on Engineering Applications of Neural Networks*, pages 223–234. Springer, 2009. 80

P. J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995. 36, 130

H. Guo and H. L. Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM SIGKDD Explorations Newsletter*, 6(1):30–39, 2004. 76

L. Guo, Y. Ma, B. Cukic, and H. Singh. Robust prediction of fault-proneness by random forests. In *15th International Symposium on Software Reliability Engineering*, pages 417–428. IEEE, 2004. 80

S. K. Gupta, S. Gupta, and R. Vijay. Prediction of student success that are going to enroll in the higher technical education. *International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)*, 3(1): 95–108, 2013. 48

L. Györfi. *Principles of nonparametric learning*, volume 434. Springer, 2002. 18, 19, 20, 21, 23, 24, 26

L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2002. 3, 18, 24, 34, 60, 115, 116, 140, 142

T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell. A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, 38(6):1276–1304, 2011. 74

M. Hamers and M. Kohler. A bound on the expected maximal deviation of averages from their means. *Statistics & Probability Letters*, 62(2):137–144, 2003. 140

L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 12(10):993–1001, 1990. 9, 39

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009. 2, 17, 28, 44, 168

D. Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992. 24, 60

H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009. 75

J. Hill and Y.-S. Su. Assessing lack of common support in causal inference using bayesian nonparametrics: Implications for evaluating the effect of breastfeeding on children's cognitive outcomes. *The Annals of Applied Statistics*, pages 1386–1420, 2013. 4

J. Hill, A. Linero, and J. Murray. Bayesian additive regression trees: A review and look forward. *Annual Review of Statistics and Its Application*, 7:251–278, 2020. 29

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 40

N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker. *Bayesian nonparametrics*, volume 28. Cambridge University Press, 2010. 4

V. Hofer and G. Krempl. Drift mining in data: A framework for addressing drift in classification. *Computational Statistics & Data Analysis*, 57(1):377–391, 2013. 181

C. Holmes and B. Mallick. Bayesian radial basis functions of variable dimension. *Neural computation*, 10(5):1217–1233, 1998. 35

C. Holt. Forecasting trends and seasonal by exponentially weighted moving averages. *ONR Memorandum*, 52, 1957. 3

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. 5, 33, 46, 67, 74, 80, 128

Y. Huang and S. Meng. A bayesian nonparametric model and its application in insurance loss prediction. *Insurance: Mathematics and Economics*, 2020. 4

K. D. Humbird, J. L. Peterson, and R. G. McClarren. Deep neural network initialization with decision trees. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1286–1295, 2018. 109, 182

R. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for r 7, 2008. *URL http://www. jstatsoft. org/v27/i03*, 2007. 172

R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018. 157, 160, 161, 168

Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, and A. Criminisi. Decision forests, convolutional networks and the models in-between. *arXiv preprint arXiv:1603.01250*, 2016. 41

O. Irsoy, O. T. Yıldız, and E. Alpaydın. Soft decision trees. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1819–1822. IEEE, 2012. 39, 41

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013. 2, 3

Z. Jelinski and P. Moranda. Software reliability research. In *Statistical computer performance evaluation*, pages 465–484. Elsevier, 1972. 79

J. M. Jerez-Aragonés, J. A. Gómez-Ruiz, G. Ramos-Jiménez, J. Muñoz-Pérez, and E. Alba-Conejo. A combined neural network and decision trees model for prognosis of breast cancer relapse. *Artificial Intelligence in Medicine*, 27(1): 45–63, 2003. 7

H. Jiang, S. Zhang, J. Jing, and C. Zhu. The improvement and analysis of the high-pressure propane recovery process. *Asia-Pacific Journal of Chemical Engineering*, 13(5):e2246, 2018. 108

X.-Y. Jing, F. Wu, X. Dong, and B. Xu. An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems. *IEEE Transactions on Software Engineering*, 43(4):321–339, 2016. 74, 75, 98

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994. 40

J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383, 2011. 4

Y. Kang, R. J. Hyndman, and F. Li. Gratis: Generating time series with diverse and controllable characteristics. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2020. 172

A. Kapelner and J. Bleich. bartmachine: Machine learning with bayesian additive regression trees. *Journal of Statistical Software*, 70(4):1–40, 2016. 145

N. B. Karayiannis. Reformulated radial basis neural networks trained by gradient descent. *IEEE Transactions on Neural Networks*, 10(3):657–671, 1999. 119

C. Katris. Prediction of unemployment rates with time series and machine learning techniques. *Computational Economics*, 55(2):673–706, 2020. 12, 156, 160

A. Kendall and Y. Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5574–5584, 2017. 128

Z. R. Khan Jaffur, N.-U.-H. Sookia, P. Nunkoo Gonpot, and B. Seetanah. Out-of-sample forecasting of the canadian unemployment rates using univariate models. *Applied Economics Letters*, 24(15):1097–1101, 2017. 156, 167

M. Khashei and M. Bijari. Which methodology is better for combining linear and nonlinear models for time series forecasting? *Journal of Industrial and Systems Engineering*, 4(4):265–285, 2011a. 157

M. Khashei and M. Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, 11(2): 2664–2675, 2011b. 157

T. M. Khoshgoftaar and Y. Liu. A multi-objective software quality classification model using genetic programming. *IEEE Transactions on Reliability*, 56(2): 237–245, 2007. 80

T. M. Khoshgoftaar and N. Seliya. Tree-based software quality estimation models for fault prediction. In *Proceedings Eighth IEEE Symposium on Software Metrics*, pages 203–214. IEEE, 2002. 80

T. M. Khoshgoftaar and N. Seliya. Analogy-based practical classification rules for software quality estimation. *Empirical Software Engineering*, 8(4):325–350, 2003. 80

B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems*, pages 2280–2288, 2016. 180

J. H. Kim and M. T. Ngo. Modelling and forecasting monthly airline passenger flows among three major australian cities. *Tourism Economics*, 7(4):397–412, 2001. 12

K. Kim. A hybrid classification algorithm by subspace partitioning through semi-supervised decision tree. *Pattern Recognition*, 60:157–163, 2016. 49, 109

S. Kim, T. Zimmermann, E. J. Whitehead Jr, and A. Zeller. Predicting faults from cached history. In *Proceedings of the 29th International Conference on Software Engineering*, pages 489–498. IEEE Computer Society, 2007. 80

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 69

M. Kohler. Nonparametric regression with additional measurement errors in the dependent variable. *Journal of Statistical Planning and Inference*, 136(10): 3339–3361, 2006. 140

M. Kohler and A. Krzyżak. Adaptive regression estimation with multilayer feedforward neural networks. *Nonparametric Statistics*, 17(8):891–913, 2005. 140, 143

T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *IEEE International Conference on Neural Networks*, volume 1, pages 61–68, 1988. 114

A. Kołcz and C. H. Teo. Feature weighting for improved classifier robustness. In *CEAS09: sixth conference on email and anti-spam*, 2009. 183

P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1467–1475, 2015. 38, 41, 182

N. Kourentzes. nnfor: Time series forecasting with neural networks. r package version 0.9. 6, 2017. 172

Z. Kovacic. Early prediction of student success: Mining students' enrolment data. *Proceedings of Informing Science and IT Education Conference (InSITE)*, pages 648–665, 2010. 48

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 4

M. Krofta. Apparatus for clarification of water, December 2, 1986. US Patent 4,626,345. 108

M. Krofta. Three zone dissolved air flotation clarifier with improved efficiency, December 8, 1998. US Patent 5,846,413. 108, 120

A. Krzyzak, T. Linder, and C. Lugosi. Nonparametric estimation and classification using radial basis function nets and empirical risk minimization. *IEEE Transactions on Neural Networks*, 7(2):475–487, 1996. 3, 115

M. Kubat. Decision trees can initialize radial-basis function networks. *IEEE Transactions on Neural Networks*, 9(5):813–821, 1998. 77

M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186, 1997. 43

M. Kuhn and K. Johnson. *Applied predictive modeling*. Springer, 2013. 8

L. I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002. 5

L. I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004. 5, 6, 8, 18, 32, 42, 56, 74, 75, 109, 114

L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2): 181–207, 2003. 9

L. Kuo, J. C. Lee, K. Choi, and T. Y. Yang. Bayes inference for s-shaped software-reliability growth models. *IEEE Transactions on Reliability*, 46(1):76–80, 1997. 183

L. A. Kurgan, K. J. Cios, R. Tadeusiewicz, M. Ogiela, and L. S. Goodenday. Knowledge discovery approach to automated cardiac spect diagnosis. *Artificial Intelligence in Medicine*, 23(2):149–169, 2001. 10, 66

D. Kwiatkowski, P. C. Phillips, P. Schmidt, Y. Shin, et al. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of econometrics*, 54(1-3):159–178, 1992. 172

Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis*, 142:106816, 2020. 184

D. Laptev and J. M. Buhmann. Convolutional decision trees for feature learning and segmentation. In *German Conference on Pattern Recognition*, pages 95–106. Springer, 2014. 40

I. H. Laradji, M. Alshayeb, and L. Ghouti. Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, 58:388–402, 2015. 79, 81

P. Laskov and R. Lippmann. Machine learning in adversarial environments, 2010. 181

Y. LeCun, Y. Bengio, and Y. Hinton. Deep learning. *Nature*, 521(7553):436, 2015. 128

D. S. Lee and S. N. Srihari. A theory of classifier combination: the neural network approach. In *Proceedings of the Third International Conference on Document Analysis and Recognition, 1995*, volume 1, pages 42–45. IEEE, 1995. 50

H. K. H. Lee. Consistency of posterior distributions for neural networks. *Neural Networks*, 13(6):629–642, 2000. 132, 138

H. K. H. Lee. *Bayesian Nonparametrics via Neural Networks*, volume 13. SIAM, 2004. 36, 132

J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017. 184

T. S. Lee and I. F. Chen. A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. *Expert Systems with Applications*, 28(4):743–752, 2005. 108

W. Leigh, R. Purvis, and J. M. Ragusa. Forecasting the nyse composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support. *Decision Support Systems*, 32(4): 361–377, 2002. 6

G. Lemaître, F. Nogueira, and C. K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. 76, 98, 104

A. Léon and L. Denoyer. Policy-gradient methods for decision trees. In *ESANN*, 2016. 40

P. Leoni. Long-range out-of-sample properties of autoregressive neural networks. *Neural Computation*, 21(1):1–8, 2009. 163

S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4):485–496, 2008. 74

A. Lhassani, M. Rumeau, D. Benjelloun, and M. Pontie. Selective demineralization of water by nanofiltration application to the defluorination of brackish water. *Water Research*, 35(13):3260–3264, 2001. 147

M. Li and W. Q. Meeker. Application of bayesian methods in reliability data analyses. *Journal of Quality Technology*, 46(1):1–23, 2014. 183

M. Lichman et al. Uci machine learning repository, 2013. 144

H. J. Lin, Y. T. Kao, F. W. Yang, and P. S. Wang. Content-based image retrieval trained by adaboost for mobile application. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(04):525–541, 2006. 6

C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. J. Li, L. Fei Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 41

M. Liu, L. Miao, and D. Zhang. Two-stage cost-sensitive learning for software defect prediction. *IEEE Transactions on Reliability*, 63(2):676–686, 2014. 74, 75, 79

W. Liu, S. Chawla, D. A. Cieslak, and N. V. Chawla. A robust decision tree algorithm for imbalanced data sets. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 766–777. SIAM, 2010. 4

W. Y. Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011. 5, 28, 128

V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013. 79

V. López, A. Fernández, and F. Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1–13, 2014. 181

E. Lughofer. Hybrid active learning for reducing the annotation effort of operators in classification systems. *Pattern Recognition*, 45(2):884–896, 2012. 6

G. Lugosi and A. Nobel. Consistency of data-driven histogram methods for density estimation and classification. *The Annals of Statistics*, 24(2):687–706, 1996. 47, 54, 55, 77, 132

G. Lugosi and K. Zeger. Nonparametric estimation via empirical risk minimization. *IEEE Transactions on Information Theory*, 41(3):677–687, 1995. 47, 57, 77, 132, 139

D. J. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992a. 128, 145

D. J. MacKay. A practical Bayesian framework for backprop networks. *Neural Computation*, 4:448–472, 1992b. 4, 34, 134

S. Mahuli, R. Rhinehart, and J. Riggs. ph control using a statistical technique for continuous on-line model adaptation. *Computers & Chemical Engineering*, 17(4):309–317, 1993. 108, 148

R. Malhotra and S. Kamal. An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data. *Neurocomputing*, 343:120–140, 2019. 74

D. M. Malioutov, K. R. Varshney, A. Emad, and S. Dash. Learning interpretable classification rules with boolean compressed sensing. In *Transparent Data Mining for Big and Small Data*, pages 95–121. Springer, 2017. 180

O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995. 11

P. Mangiameli, D. West, and R. Rampal. Model selection for medical diagnosis decision support systems. *Decision Support Systems*, 36(3):247–259, 2004. 6

G. A. Marques and J. A. S. Tenório. Use of froth flotation to separate pvc/pet mixtures. *Waste Management*, 20(4):265–269, 2000. 119

K. Mathan, P. M. Kumar, P. Panchatcharam, G. Manogaran, and R. Varadharajan. A novel gini index decision tree data mining method with neural network classifiers for prediction of heart disease. *Design Automation for Embedded Systems*, 22(3):225–242, 2018. 7

A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018. 184

P. Meedech, N. Iam-On, and T. Boongoen. Prediction of student dropout using personal profile and data mining approach. In *Intelligent and Evolutionary Systems*, pages 143–155. Springer, 2016. 48

H. N. Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, 1(1):61–80, 1993. 143

R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system aq15 and its testing application to three medical domains. *Proc. AAAI 1986*, pages 1–041, 1986. 11

C. Micheloni, A. Rani, S. Kumar, and G. L. Foresti. A balanced neural tree for pattern classification. *Neural Networks*, 27:81–90, 2012. 128

C. Milas and P. Rothman. Out-of-sample forecasting of unemployment rates with pooled stvecm forecasts. *International Journal of Forecasting*, 24(1):101–121, 2008. 156

A. L. Montgomery, V. Zarnowitz, R. S. Tsay, and G. C. Tiao. Forecasting the us unemployment rate. *Journal of the American Statistical Association*, 93(442): 478–493, 1998. 156

J. G. Moreno-Torres, T. Raeder, R. Alaiz-RodríGuez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45 (1):521–530, 2012. 181

S. Moshiri and L. Brown. Unemployment variation over the business cycles: a comparison of forecasting models. *Journal of Forecasting*, 23(7):497–511, 2004. 156, 167

P. Müller and F. A. Quintana. Nonparametric bayesian data analysis. *Statistical science*, pages 95–110, 2004. 4

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 4

S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998. 49

J. D. Musa and K. Okumoto. A logarithmic poisson execution time model for software reliability measurement. In *Proceedings of the 7th international conference on Software engineering*, pages 230–238. Citeseer, 1984. 79

S. Nagao, F. Takeda, and R. Tanaka. Nowcasting of the us unemployment rate using google trends. *Finance Research Letters*, 30:103–109, 2019. 156, 160

N. Nagappan and T. Ball. Static analysis tools as early indicators of pre-release defect density. In *Proceedings of the 27th International Conference on Software Engineering*, pages 580–586. ACM, 2005a. 80

N. Nagappan and T. Ball. Use of relative code churn measures to predict system defect density. In *Proceedings of the 27th International Conference on Software Engineering*, pages 284–292. ACM, 2005b. 80

R. Naveiro, A. Redondo, D. Rios Insua, and F. Ruggeri. Adversarial classification: An adversarial risk analysis approach. *International Journal of Approximate Reasoning*, 113:133–148, 2019. 182, 183

R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media, New York, NY, 1996. 34, 35, 128, 145

R. M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000. 4

J. A. Nelder and R. W. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972. 3

T. Nguyen, S. Gupta, S. Venkatesh, and D. Phung. A bayesian nonparametric framework for activity recognition using accelerometer data. In *2014 22nd International Conference on Pattern Recognition*, pages 2017–2022. IEEE, 2014. 4

V. Nguyen, S. Gupta, S. Rana, C. Li, and S. Venkatesh. A bayesian nonparametric approach for multi-label classification. In *Asian conference on machine learning*, pages 254–269, 2016. 4

Y. Ni, P. Müller, M. Diesendruck, S. Williamson, Y. Zhu, and Y. Ji. Scalable bayesian nonparametric clustering and classification. *Journal of Computational and Graphical Statistics*, 29(1):53–65, 2020. 4

A. Nobel. Histogram regression estimation using data-dependent partitions. *The Annals of Statistics*, 24(3):1084–1105, 1996. 111, 113, 132

S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493, 1992. 33

A. K. Ojha. Management education in india: Avoiding the simulacra effect. In *Management Education in India*, pages 55–77. Springer, 2017. 47

M. R. Oliveira and L. Torgo. Ensembles for time series forecasting. *Journal of Machine Learning Research*, 39:360–370, 2014. 157

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. 9, 38

P. Orbanz and Y. W. Teh. Bayesian nonparametric models. *Encyclopedia of machine learning*, (1), 2010. 4

C. Ordóñez, F. S. Lasheras, J. Roca-Pardiñas, and F. J. de Cos Juez. A hybrid arima–svm model for the study of the remaining useful life of aircraft engines. *Journal of Computational and Applied Mathematics*, 346:184–191, 2019. 7

Y. Ouyang, P. Nkedi-Kizza, Q. Wu, D. Shinde, and C. Huang. Assessment of seasonal variations in surface water quality. *Water Research*, 40(20):3800–3810, 2006. 148

P.-F. Pai and C.-S. Lin. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005. 6, 157, 167, 169

Z. S. Pan, S. C. Chen, G. B. Hu, and D. Q. Zhang. Hybrid neural network and c4.5 for misuse detection. In *International Conference on Machine Learning and Cybernetics, 2003*, volume 4, pages 2463–2467. IEEE, 2003. 38

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 99

A. Pektas and H. Cigizoglu. Investigating the extrapolation performance of neural network models in suspended sediment data. *Hydrological Sciences Journal*, 62 (10):1694–1703, 2017. 109

A. O. Pektaş and H. K. Cigizoglu. Ann hybrid model versus arima and arimax models of runoff coefficient. *Journal of Hydrology*, 500:21–36, 2013. 109

R. F. Peláez. Using neural nets to forecast the unemployment rate. *Business Economics*, 41(1):37–44, 2006. 156, 167

L. Pelayo and S. Dick. Evaluating stratification alternatives to improve software defect prediction. *IEEE Transactions on Reliability*, 61(2):516–525, 2012. 74, 75, 81

A. Pena Ayala. Educational data mining: A survey and a data mining-based analysis of recent works. *Expert Systems with Applications*, 41(4):1432–1462, 2014. 48

H. Pham. *Handbook of reliability engineering*. Springer Science & Business Media, 2006. 79

T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247(4945):978–982, 1990. 119

D. Pollard. *Convergence of stochastic processes.* Springer Science & Business Media, 1984. 24, 59, 92, 117, 141

D. Pollard. Empirical processes: theory and applications. In *NSF-CBMS regional conference series in probability and statistics*, pages 1–86, 1990. 24, 59, 60

M. T. Pratola, H. A. Chipman, J. R. Gattiker, D. M. Higdon, R. McCulloch, and W. N. Rust. Parallel bayesian additive regression trees. *Journal of Computational and Graphical Statistics*, 23(3):830–852, 2014. 29

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical recipes in c: the art of scientific computing. *Cambridge University Press, Cambridge, MA,*, 131:243–262, 1992. 56

T. Proietti. Forecasting the us unemployment rate. *Computational Statistics & Data Analysis*, 42(3):451–476, 2003. 156

J. R. Quinlan. Decision trees and decision-making. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):339–346, 1990. 27

J. R. Quinlan. C4. 5: Programming for machine learning. *Morgan Kauffmann*, 38:48, 1993. 49

J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning.* The MIT Press, 2009. 181

R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and W. Meding. Analyzing defect inflow distribution and applying bayesian inference method for software defect prediction in large software projects. *Journal of Systems and Software*, 117:229–244, 2016. 183

R. Ranawana and V. Palade. Multi-classifier systems: Review and a roadmap for developers. *International Journal of Hybrid Intelligent Systems*, 3(1):35–61, 2006. 5

A. Rani, G. L. Foresti, and C. Micheloni. A neural tree for classification using convex objective function. *Pattern Recognition Letters*, 68:41–47, 2015. 42

C. R. Rao. A review of canonical coordinates and an alternative to correspondence analysis using hellinger distance. *Qüestiió: quaderns d'estadística i investigació operativa*, 19(1), 1995. 31, 83

M. Rastgoo, G. Lemaitre, J. Massich, O. Morel, F. Marzani, R. Garcia, and F. Meriaudeau. Tackling the problem of data imbalancing for melanoma classification. In *Bioimaging*, 2016. 75

L. Rastrigin and R. Erenstein. Method of collective recognition. *Energoizdat, Moscow*, 595, 1981. 37

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2902–2911, 2017. 41

M. Redmond and A. Baveja. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3):660–678, 2002. 11, 144

C. Reinders, H. Ackermann, M. Y. Yang, and B. Rosenhahn. Object recognition from very few training examples for enhancing bicycle maps. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–8. IEEE, 2018. 7

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 180

D. Rios Insua, R. Naveiro, V. Gallego, and J. Poulos. Adversarial machine learning: Perspectives from adversarial risk analysis. *arXiv preprint arXiv:2003.03546*, 2020. 183

D. R. Rios Insua and P. Müller. Feedforward neural networks for nonparametric regression. In *Practical Nonparametric and Semiparametric Bayesian Statistics*, pages 181–193. Springer, 1998. 35, 36, 130, 134, 147

I. Rish. An empirical study of the naive bayes classifier. In *IJCAI Workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001. 80

V. Rocková and S. van der Pas. Posterior concentration for bayesian regression trees and forests. *Annals of Statistics*, pages 1–40, 2020. 132, 139

J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006. 66

L. Rokach. Mining manufacturing data using genetic algorithm-based feature set decomposition. *International journal of intelligent systems technologies and applications*, 4(1):57, 2008. 6

L. Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046–4072, 2009. 7

L. Rokach. *Pattern classification using ensemble methods*. World Scientific, 2010. 18

L. Rokach and O. Maimon. Data mining for improving the quality of manufacturing: a feature set decomposition approach. *Journal of Intelligent Manufacturing*, 17(3):285–299, 2006. 6

C. Romero and S. Ventura. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(6):601–618, 2010. 48

S. Rota Bulo and P. Kontschieder. Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 81–88, 2014. 7, 38, 40

F. Ruggeri, R. Soyer, and C. IMATI. Advances in bayesian software reliability modelling. *Advances in Mathematical Modelling for Reliability*, pages 149–157, 2008. 80, 184

D. E. Rumelhardt, C. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. 128

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, pp 318-362, 1985. 3, 32, 33, 77, 161

D. Ryu, O. Choi, and J. Baik. Value-cognitive boosting with a support vector machine for cross-project defect prediction. *Empirical Software Engineering*, 21(1):43–71, 2016. 98

D. Ryu, J.-I. Jang, and J. Baik. A transfer cost-sensitive boosting approach for cross-project defect prediction. *Software Quality Journal*, 25(1):235–272, 2017. 79

A. Sakar and R. J. Mammone. Growing and pruning neural tree networks. *IEEE Transactions on Computers*, 42(3):291–299, 1993. 38, 46

D. Sannen, E. Lughofer, and H. Van Brussel. Towards incremental classifier fusion. *Intelligent Data Analysis*, 14(1):3–30, 2010. 6

N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972. 23

A. Schclar, A. Tsikinovsky, L. Rokach, A. Meisels, and L. Antwarg. Ensemble methods for improving the performance of neighborhood-based collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 261–264. ACM, 2009. 6

M. J. Schervish. *Theory of statistics*. Springer Science & Business Media, 2012. 2, 3

M. Sebri. Forecasting urban water demand: A meta-regression analysis. *Journal of Environmental Management*, 183:777–785, 2016. 108

I. K. Sethi. Entropy nets: from decision trees to neural networks. *Proceedings of the IEEE*, 78(10):1605–1613, 1990. 6, 38, 39, 46, 77, 109, 128

I. K. Sethi. Decision tree performance enhancement using an artificial neural network implementation. In *Machine Intelligence and Pattern Recognition*, volume 11, pages 71–88. 1991. 39

I. K. Sethi. Neural implementation of tree classifiers. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(8):1243–1249, 1995. 39

R. Shatnawi. Improving software fault-prediction for imbalanced data. In *International Conference on Innovations in Information Technology*, pages 54–59. IEEE, 2012. 79

M. Shrimali and K. Singh. New methods of nitrate removal from water. *Environmental Pollution*, 112(3):351–359, 2001. 108

E. Siegel. *Predictive analytics: The power to predict who will click, buy, lie, or die*. John Wiley & Sons, 2013. 2

M. J. Siers and M. Z. Islam. Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. *Information Systems*, 51:62–71, 2015. 74, 75

M. A. Silgu and H. B. Celikoglu. Clustering traffic flow patterns by fuzzy c-means method: some preliminary findings. In *International Conference on Computer Aided Systems Theory*, pages 756–764. Springer, 2015. 109

K. P. Singh, A. Basant, A. Malik, and G. Jain. Artificial neural network modeling of the river water qualitya case study. *Ecological Modelling*, 220(6):888–895, 2009. 148

J. Sirat and J. Nadal. Neural trees: a new tool for classification. *Network: Computation in Neural Systems*, 1(4):423–438, 1990. 6, 38, 46, 109, 128

Q. Song, M. Shepperd, M. Cartwright, and C. Mair. Software defect association mining and defect correction effort prediction. *IEEE Transactions on Software Engineering*, 32(2):69–82, 2006. 74, 75

K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002. 41

S. M. Stigler. An attack on gauss, published by legendre in 1820. *Historia Mathematica*, 4(1):31–35, 1977. 2

S. M. Stigler. Gauss and the invention of least squares. *The Annals of Statistics*, 9(3):465–474, 1981. 3

C. Su, S. Ju, Y. Liu, and Z. Yu. Improving random forest and rotation forest for highly imbalanced datasets. *Intelligent Data Analysis*, 19(6):1409–1432, 2015. 6, 76, 81, 85, 98

A. Suárez and J. F. Lutsko. Globally optimal fuzzy decision trees for classification and regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1297–1311, 1999. 40

V. Sugumaran, V. Muralidharan, and K. Ramachandran. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing*, 21(2):930–942, 2007. 7

Z. Sun, Q. Song, and X. Zhu. Using coding-based ensemble learning to improve software defect prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1806–1817, 2012. 74, 80

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 41

A. C. Tan, D. Gilbert, and Y. Deville. Multi-class protein fold classification using a new ensemble machine learning approach. *Genome Informatics*, 14:206–217, 2003. 6

E. K. Tang, P. N. Suganthan, and X. Yao. An analysis of diversity measures. *Machine learning*, 65(1):247–271, 2006. 9

R. Tanno, K. Arulkumaran, D. C. Alexander, A. Criminisi, and A. Nori. Adaptive neural trees. *In Proceedings of the 36th International Conference on Machine Learning, Long Beach, California*, 2019. 6, 39, 41, 77

D. Tao, X. Tang, X. Li, and X. Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):1088–1099, 2006. 6

Y. W. Teh and M. I. Jordan. Hierarchical bayesian nonparametric models with applications. *Bayesian nonparametrics*, 1:158–207, 2010. 4

T. Teräsvirta, D. Van Dijk, and M. C. Medeiros. Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A re-examination. *International Journal of Forecasting*, 21(4):755–774, 2005. 157

N. Terui and H. K. Van Dijk. Combined forecasts from linear and nonlinear time series models. *International Journal of Forecasting*, 18(3):421–438, 2002. 157, 158

H. Tong. *Non-linear time series: a dynamical system approach*. Oxford University Press, 1990. 165, 166

A. Trapletti, F. Leisch, and K. Hornik. Stationary and integrated autoregressive neural network processes. *Neural Computation*, 12(10):2427–2450, 2000. 166

C. C. Tsai, M. C. Lu, and C.-C. Wei. Decision tree-based classifier combined with neural-based predictor for water-stage forecasts in a river basin during typhoons: a case study in taiwan. *Environmental Engineering Science*, 29(2): 108–116, 2012. 7, 39, 46, 67, 109, 123

K. Tsujino and S. Nishida. Implementation and refinement of decision trees using neural networks for hybrid knowledge acquisition. *Artificial Intelligence in Engineering*, 9(4):265–276, 1995. 7, 38

A. E. Tümer and A. Akkuş. Forecasting gross domestic product per capita using artificial neural networks with non-economical parameters. *Physica A*, 512: 468–473, 2018. 157

B. Turhan. On the dataset shift problem in software engineering prediction models. *Empirical Software Engineering*, 17(1-2):62–74, 2012. 80

B. Turhan and A. Bener. Analysis of naive bayes assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, 68(2):278–290, 2009. 80

P. E. Utgoff. Perceptron trees: A case study in hybrid concept representations. *Connection Science*, 1(4):377–391, 1989. 6, 38, 42, 84, 128

N. D. Vanli, M. O. Sayin, M. Mohaghegh, H. Ozkan, and S. S. Kozat. Nonlinear regression via incremental decision trees. *Pattern Recognition*, 86:1–13, 2019. 129

V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013. 2

V. Vapnik and A. Chervonenkis. Theory of pattern recognition, 1974. 2, 18, 22

R. Vedelago and G. J. Millar. Process evaluation of treatment options for high alkalinity coal seam gas associated water. *Journal of Water Process Engineering*, 23:195–206, 2018. 149

M. R. Vicente, A. J. López-Menéndez, and R. Pérez. Forecasting unemployment with internet search data: Does it help to improve predictions when job destruction is skyrocketing? *Technological Forecasting and Social Change*, 92: 132–139, 2015. 12, 156

J. Von Neumann and R. Kurzweil. *The computer and the brain*. Yale University Press, 2012. 37

H. Wan, G. Wu, M. Cheng, Q. Huang, R. Wang, and M. Yuan. Software defect prediction using dictionary learning. In *SEKE*, pages 335–340, 2017. 79

S. Wang and X. Yao. Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2):434–443, 2013. 74, 79, 80, 81, 98

S. Wang, L. L. Minku, N. Chawla, and X. Yao. Learning from data streams and class imbalance. *Connection Science*, 31:103–104, 2019a. 75

T. Wang and Q. Lin. Hybrid predictive model: When an interpretable model collaborates with a black-box model. *arXiv preprint arXiv:1905.04241*, 2019. 6

T. Wang, Z. Zhang, X. Jing, and L. Zhang. Multiple kernel ensemble learning for software defect prediction. *Automated Software Engineering*, 23(4):569–590, 2016. 79

X. Wang, Y. Zhou, Z. Zhao, L. Wang, J. Xu, and J. Yu. A novel water quality mechanism modeling and eutrophication risk assessment method of lakes and reservoirs. *Nonlinear Dynamics*, 96(2):1037–1053, 2019b. 147

A. Weller. Challenges for transparency. *arXiv preprint arXiv:1708.01870*, 2017. 180

P. R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342, 1960. 3

D. P. Wipf and S. S. Nagarajan. A new view of automatic relevance determination. In *Advances in neural information processing systems*, pages 1625–1632, 2008. 184

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996. 5, 72

D. H. Wolpert. The supervised learning no-free-lunch theorems. In *Soft computing and industry*, pages 25–42. Springer, 2002. 26, 37

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. 126

W. H. Wong and X. Shen. Probability inequalities for likelihood ratios and convergence rates of sieve mles. *Annals of Statistics*, 23(2):339–362, 1995. 138

M. Wozniak. *Hybrid classifiers: methods of data, knowledge, and classifier combination*, volume 519. Springer, 2013. 5, 6, 8, 18

J. Wu, H. Xiong, and J. Chen. Cog: local decomposition for rare class analysis. *Data Mining and Knowledge Discovery*, 20(2):191–220, 2010. 102

H. Xiao. Ndt: Neual decision tree towards fully functioned neural graph. *arXiv preprint arXiv:1712.05934*, 2017. 41

T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 177–186, 2014. 40, 41

M. Xie. *Software reliability modelling*, volume 1. World Scientific, 1991. 79

M. Xie. Software reliability models for practical applications. In *Software Quality and Productivity*, pages 211–214. Springer, 1995. 80, 183

S. K. Yadav and S. Pal. Data mining application in enrollment management: A case study. *International Journal of Computer Applications*, 41(5):1–6, 2012. 48

S. Yamada. *Software reliability modeling: fundamentals and applications*, volume 5. Springer, 2014. 79

B. Yang and X. Li. A study on software reliability prediction based on support vector machines. In *IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1176–1180. IEEE, 2007. 80

Y. Yang, I. G. Morillo, and T. M. Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018. 46, 67, 182

I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998. 12, 144

J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017. 40

L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5(Oct):1205–1224, 2004. 56

A. Yuan. Bayesian frequentist hybrid inference. *Annals of Statistics*, 37(5A): 2458–2501, 2009. 129

G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003. 6, 12, 157, 167, 169

J. Zhang, Y. Li, X. Zeng, G. Huang, Y. Li, Y. Zhu, F. Kong, M. Xi, and J. Liu. Effluent trading planning and its application in water quality management: A factor-interaction perspective. *Environmental Research*, 168:286–305, 2019a. 147

S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 52(1):1–38, 2019b. 182

L. Zhao. A new approach for software testability analysis. In *Proceedings of the 28th international conference on Software engineering*, pages 985–988, 2006. 184

J. Zheng. Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6):4537–4543, 2010. 74, 80, 81

S.-H. Zhong, Y. Liu, and K. A. Hua. Field effect deep networks for image recognition with incomplete data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(4):1–22, 2016. 182

X. Zhong and M. Ghosh. Approximate bayesian computation via sufficient dimension reduction. *arXiv preprint arXiv:1608.05173*, 2016. 184

Y. Zhou and J. Davis. Open source software reliability model: an empirical approach. *ACM SIGSOFT software engineering notes*, 30(4):1–6, 2005. 183

Y. Zhou, M. Kantarcioglu, and B. Xi. A survey of game theoretic approach for adversarial machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1259, 2019. 183

Z. H. Zhou. *Ensemble methods: foundations and algorithms.* Chapman and Hall/CRC, 2012. 5, 6, 8, 18

Z. H. Zhou and Z. Q. Chen. Hybrid decision tree. *Knowledge-based systems*, 15 (8):515–528, 2002. 77

Z. H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002. 49, 63, 67

T. Zhu, Y. Lin, and Y. Liu. Improving interpolation-based oversampling for imbalanced data learning. *Knowledge-Based Systems*, 187:104826, 2020. 75