TURNING DATA INTO GUARDRAILS

DECODING FINANCIAL VULNERABILITY THROUGH BEHAVIOURAL SIGNS

DEBANWITA HAJRA

TURNING DATA INTO GUARDRAILS

DECODING FINANCIAL VULNERABILITY THROUGH BEHAVIOURAL SIGNS

A Dissertation Submitted in Fulfillment of the Requirements for the Award of

Master of Technology in Cryptology and Security

by

Debanwita Hajra

[Roll No.: CRS2308]

on Completion of her Internship from

Hongkong & Shanghai Banking Corporation (HSBC), Kolkata

where she worked as an

Analyst Intern

Under the Guidance of

Mr. Debasish Mishra

Vice President

Decision Sciences

Hongkong & Shanghai Banking Corporation, Bangalore

&

Prof. Anisur Rahaman Molla

Associate Professor

Cryptology and Security Research Unit

Indian Statistical Institute, Kolkata

Indian Statistical Institute (ISI)
Kolkata - 700108, India

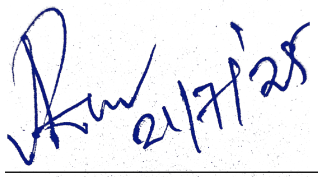
July 2025

CERTIFICATE

This is to certify that the project titled **Turning Data Into Guardrails, Decoding Financial Vulnerability Through Behavioural Signs**, submitted by **Debanwita Hajra** to the Indian Statistical Institute, Kolkata, in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Cryptology and Security**, is a bonafide record of work carried out by her under my supervision and guidance.

The project has fulfilled all the requirements as per the regulations of this Institute. The contents of this project, in full or in parts, have not been submitted to any other institute or university for the award of any degree or diploma.

Mr. Debasish Mishra
Vice President, Decision Sciences
Hongkong & Shanghai Banking Corporation, Bangalore



Prof. Anisur Rahaman Molla
Associate Professor
Cryptology and Security Research Unit
Indian Statistical Institute, Kolkata

Abstract

This report presents the work I did during my internship at Hongkong and Shanghai Banking Corporation (HSBC), Kolkata. As a financial institution, the strength of the bank is fundamentally rooted in the behavior and reliability of its customers. Understanding this behavior is not only desirable; it is essential for the security, risk mitigation and future strategic planning of the bank. To do this, banks must invest in a thorough analysis of the financial behavior of their customers to detect early signs of risk and act accordingly.

I worked in the Finance Support Team within the Data and Analytics division, where our focus was to build data-backed tools and insights to help identify what can cause financial vulnerability. My internship focused on feature engineering and dashboard making to support and strengthen financial decision-making frameworks. Key areas of my work included

- Developing features like overdraft utilization, debt-to-income ratio, and essential spend indicators
- Automating SQL pipelines to extract and aggregate large-scale banking data
- Designing interactive dashboards in Looker to visualize customer behaviors (e.g., gambling, BNPL overuse)
- Analyzing correlations, outlier patterns, and vulnerability signals in transactional datasets
- Presenting a theoretical exploration of NLP embedding techniques including TF-IDF, CBOW, Skip-Gram, and LSTM

The project blended technical depth with domain knowledge to build insights from financial data, aiming to identify early risk markers and guide responsible lending. The internship not only enhanced my analytical and engineering skills, but also offered a valuable experience in solving real-world problems collaboratively within a data science team.

Keywords: Financial Analytics, Overdraft, BNPL, Risk Modeling, Feature Engineering, Looker, Word Embeddings, SQL, Customer Vulnerability

Contents

1 Introduction **4**

2 My Team - Its Goals and Responsibilities **6**
2.1 Prediction of Future Information from Existing Data 6
2.2 Presenting Relevant Insights Based on Existing Data to Business 7

3 Problem Statements and Methods to Solve it **8**
3.1 Problem Statements 8
3.1.1 Feature Creation: 8
3.1.2 Making Looker Dashboards for Visualization: 8
3.1.3 Creating Slides on Natural Language Programming (NLP) 8
3.2 Methods Taken: 8
3.2.1 Feature Creation: 8
3.2.2 Making Looker Dashboards for Visualization: 10

4 The First Problem - Feature Creation **14**
4.1 Feature creation from Overdraft data 14
4.1.1 Overdraft-Based Feature Aggregation 16
4.1.2 Interpretation of Features 18
4.1.3 Feature Selection using Information Value (IV) 18
4.1.4 Correlation Analysis 19
4.1.5 Automated Feature Engineering Procedure 23
4.2 Debt-to-Income Ratio Features 24
4.3 Essential Spend-to-Income Ratio Features 27

5 The Second Problem - Making Looker Dashboards for Visualization **31**
5.1 Looker Dashboard: Overview and Implementation 31
5.1.1 What Is Looker? 31
5.1.2 Looker's Ownership and Purpose 31
5.1.3 Data Access and Architecture 31
5.1.4 Looker Studio vs. Enterprise Looker 31
5.1.5 User Licenses and Subscription Model 32
5.1.6 Free vs. Paid Versions 32
5.1.7 How Looker Is Used 32
5.1.8 LookML Example 32
5.1.9 Accessing Looker: Who Subscribes? 33
5.1.10 Getting Started 33
5.1.11 Capabilities Available 33
5.1.12 Example Workflow 34

5.1.13 Conclusion 34
5.1.14 Gambling Pattern Analysis Dashboard 34
5.1.15 BNPL Overuse Analysis Dashboard 37

6 Creating Slides for a Knowledge Sharing Session **41**

7 Impact and Outcomes **42**

Future Work **44**

Conclusion **45**

Acknowledgment **46**

Bibliography **46**

Chapter 1

Introduction

What does financial vulnerability look like before it arrives? Can data whisper the story of an impending default before it becomes a crisis?

In an era where financial decisions intertwine seamlessly with digital behavior, understanding a customer's financial journey has become not just an advantage—but a necessity—for institutions striving to ensure trust, security, and sustainability. Modern banking systems, with their dynamic flows of transactions and evolving credit structures, demand more than static analysis; they require a lens that is adaptive, predictive, and deeply human.

Banks do not merely manage money—they manage people's hopes, habits, and hardships. Every transaction tells a story. A sudden increase in spending, a delayed repayment, an unexplained dip into overdraft—all signal potential shifts in a customer's financial health. For a bank whose foundational promise is to protect both capital and confidence, decoding these behavioral signs becomes a pillar of responsible financial practice.

This dissertation presents the work carried out during my internship at the Hongkong and Shanghai Banking Corporation (HSBC), Kolkata, under the Finance Support Team within the Data and Analytics division. The project aimed to build a data-driven understanding of financial vulnerability—how it manifests, how it can be anticipated, and how insight can be transformed into intelligent, preventive decision-making.

The focus has been threefold:

- Engineering meaningful features from raw financial data,
- Predicting vulnerability with modeling and pattern recognition,
- Visualizing these insights through dynamic dashboards for strategic visibility.

By observing spending patterns, debt behavior, overdraft usage, and demographic nuances, this work contributes to the evolving framework of customer-centric risk analysis. The approach blends domain intuition with statistical modeling, and merges structured financial signals with behavioral inference. At the heart of it lies a simple idea—that financial vulnerability is not just a number, but a narrative waiting to be understood.

It is in this spirit that I have had the privilege to work with the Finance Support Team within the Decision Sciences division—where data is not just stored or processed, but thoughtfully translated into action. Their vision and collaborative energy

shaped the essence of this work and brought to life the power of analytics in service of meaningful decision-making.
The next chapter has the introduction to the Team that I have worked with and their Key Responsibilities.

Chapter 2

My Team - Its Goals and Responsibilities

Team: Data and Analytics – Finance Support Team (FST)

Goal: To analyze and predict who could potentially not be able to repay bank loans. This helps in risk profiling and ensures timely intervention by the finance team. The team uses data-driven techniques to detect risk patterns, understand customer profiles, and support decision making for responsible credit issuance.

Responsibilities: [We mainly talk about those responsibilities which are relevant for this report and not the others] The work that our team does is, in one word, to try Predicting who can be the Vulnerable Customers.

Terminology Alert:
Vulnerable Customers: Customers who consistently do not pay back for a threshold period of time or ask the bank for financial support with respect to their repayments.

The methods it uses in order to get to its goals regarding vulnerabilities can be classified in either of two categories, they are :

- Prediction of Future Information from Existing Data
- Presenting Relevant Insights Based on Existing Data to Business

The following is a brief discussion on the above two points.

2.1 Prediction of Future Information from Existing Data

Some Key components of the work are

- Model Development that includes:

  - BASE SELECTION: Analyzing the collected past data sets to select train data and test data
  - FEATURE CREATION: Creating Feature that potentially can help decide the Target Variable (Vulnerability) and doing Feature Engineering on them.
  - MODEL SELECTION FOR SUPERVISED LEARNING: Feeding features to a model that would help identify high-risk customer segments
  - EVALUATION: Testing and evaluating model performance
  - Hyperparameter Tuning
  - Model Governance
- Doing Financial Health Check on different Products availed by Customers like Mortgages, Credit cards etc.

2.2 Presenting Relevant Insights Based on Existing Data to Business

This work involved structuring and visualizing data sets into meaningful dashboards for business understanding.

Key components of the work:

- Generating dynamic Looker dashboards to visualize key performance indicators (KPIs) as required

Chapter 3

Problem Statements and Methods to Solve it

As a part of the Finance Support Team, the following are the areas I worked on:

3.1 Problem Statements

3.1.1 Feature Creation:

"Which engineered features derived from a bank's overdraft data, debt information, spending structure over essential items and Income data can effectively serve as indicators—however partial—of financial vulnerability?"

3.1.2 Making Looker Dashboards for Visualization:

"Can a bank learn about a customer's financial stability through their BNPL usage or Participation in Gambling? Which products are being bought, and what do the evolving trends reveal about the risks involved? Can all of these be summarized in a Looker Dashboard?"

3.1.3 Creating Slides on Natural Language Programming (NLP)

"Creating Slides on Natural Language Programming (NLP) for our knowledge sharing session and larger audience."

3.2 Methods Taken:

3.2.1 Feature Creation:

A key part of my contribution was designing and creating features that could help the model identify financially vulnerable customers. These features were derived from transactional and financial behavior data. I tried to keep them realistic. Checked for Information values.

Generic Steps:

Step 0: Took the Customer Base

- I started with a fixed set of customers (the "customer base").
- It was ensured that the vulnerable customers are not a negligible part of the set of customers that I was working with.
- It was ensured that unique customer identifiers and required attributes are consistently available across all datasets.

Step 1: Data Extraction

- Relevant data (overdraft, income, debt, essential spend) was extracted from account-level and transaction-level tables.
- A dynamic `input reference date` was used to pull data from the previous 30, 90, and 180 days.
- SQL procedures automated this step to ensure consistency.

Step 2: Account-Level Aggregation

For overdraft features:

- I used daily account snapshot data to compute:
  - Maximum *authorized overdraft limit* in the time window
  - Average daily balance and overdraft usage
  - Daily unauthorized overdraft amounts
  - Number of days the balance was negative (OD Days)
- Normalize for non-uniform presence of accounts across days

For income and debt:

- Income: I extracted last 6 months' data, aggregated monthly at customer level
- Debt: I extracted latest total outstanding debt per customer

Step 3: Customer-Level Aggregation

- I aggregated account-level data to customer level:
  - Average metrics across accounts (e.g., `avg_daily_od_balance`)
  - Sum total values where relevant (e.g., `total_od_amount`, `total_income`)
- I normalized time-based metrics based on presence days across accounts

Step 4: Feature Construction

I decided different features that are realistic and has good information value and also got them checked by my seniors.
Some features are written here. Deatiled elaboration is written in the next chapter.
Overdraft Features:

- `avg_daily_od_balance`
- `od_days_percent`
- `od_utilization_ratio`
- `unauth_od_total`

Debt-to-Income Ratios:

- Computed for 30, 90, and 180-day windows
- Income is averaged from the monthly aggregates

Essential Spend-to-Income Ratios:

- Total essential spend computed over 30, 90, and 180-day windows
- Ratios formed with corresponding income aggregates

Step 5: Final Output Table

- A consolidated customer-level table was constructed with:
  - All engineered features
  - Target labels (if applicable)
- This table serves as input to both visualization and modeling layers

Step 6: Reusability and Automation

- Entire logic was encapsulated in a parameterized `SQL procedure`
- Accepts a dynamic `input date`
- Intermediate tables were kept as temporary views, while the final output was materialized as a physical table

3.2.2 Making Looker Dashboards for Visualization:

For the Looker Dashboard presentation I learned to use Looker and built the dashboards keeping the Key Performance Indicator (KPI) in mind.

In order to bring analytical insights closer to decision-making, we used Looker—a modern business intelligence platform—for data exploration and interactive dashboarding. This section outlines the journey from learning Looker's core components to creating automated backend pipelines and user-facing interfaces that empower non-technical users to extract meaningful patterns.

Step 0: Learning Looker: An Introduction

Looker is a cloud-based data analytics platform developed by **Looker Data Sciences**, now a part of Google Cloud. Unlike traditional BI tools, Looker uses a modeling language called **LookML** to define semantic layers between raw data and business metrics.

Key initial concepts learned:

- **Connection Setup:** Integrating Looker with Google BigQuery as the data warehouse
- **Model-View-Explore Architecture:** Modular separation between how data is fetched (view), how it's structured (model), and how it's used (explore)
- **LookML Syntax:** For defining dimensions, measures, joins, and persistent derived tables (PDTs)
- **User Interface:** Drag-and-drop visual building, filters, pivoting, and drill-down capabilities

Step 1: Data Preparation Using SQL Procedures

To support Looker dashboards with optimized and aggregated datasets, I wrote SQL procedures that generate all relevant tables. These procedures are:

- **Parameterized:** Accepts input such as reference date or time span (e.g., 30, 90, or 180 days)
- **Efficient:** Uses temporary tables or views to stage intermediate steps
- **Final Output:** Materializes a customer-level summary table used by Looker

Examples of tables generated:

- `gambling_summary` – capturing overall and category-wise gambling spend behavior
- `bnpl_insight` – with spend by brand, category, and repayment behavior
- `overdraft_features`, `debt_to_income`, `essential_spend_to_income` – behavioral feature tables with fixed schema

Step 2: Creating Views in LookML

Each final table was represented by a Looker `.view` file. A view defines how a single table (or subquery) is exposed to Looker users.

Step 3: Model Configuration

A model file was created to link all views and define:

- **Explores:** How multiple views can be joined
- **Connection details:** Which BigQuery project and dataset to use
- **Access settings:** Who can explore what

```
explore: gambling_summary {
  view_name: gambling_summary

  join: customer_demographics {
    type: left_outer
    sql_on: ${gambling_summary.customer_id}
    =
    ${customer_demographics.customer_id} ;;
  }
}
```

Step 4: Exploration and Dashboard Making

Once views and models were defined, I could freely `explore` data:

- Choose dimensions (e.g., customer type, region)
- Add filters (e.g., is_lottery = Yes)
- Visualize results using bar charts, line plots, pie charts, or sankey diagrams

Step 5: Dashboards Created

Three major dashboards were developed:

1. **Gambling Insights:**
    - Summary KPIs: transaction count, value, average transaction
    - Trend over time: merchant-level seasonality
    - Behavioral clusters: relation with loan default or vulnerability (sankey diagram)
2. **BNPL Behavior:**
    - Spend category, brand preferences, and income group
    - Repayment behaviors and frequency analysis
    - Demographic overlays: age, location, and gender
3. **Overdraft Features and Risk:**
    - Time-windowed metrics (30, 90, 180 days)
    - Distribution plots (via integrated analytics)
    - Flags for extreme OD utilization or unauthorized usage

Step 6: Automation and Reusability

To make this system sustainable:

- SQL procedures were designed to be re-runnable with minimal change
- Looker's schedules auto-refresh tiles based on data updates
- Models can easily extend to new tables (e.g., credit card behavior)

Step 7: mpact and Learnings

Using Looker, we were able to move from raw SQL queries to interactive and shareable dashboards within a secure, governed platform. The semantic layer reduced ad hoc errors, while visuals provided real-time insights into customer vulnerability patterns. This experience helped bridge data engineering and storytelling—empowering stakeholders to act on data directly.

The next three chapters talk about each problem elaborately,

Chapter 4

The First Problem - Feature Creation

The work on feature Creation from different data source is documented as follows:

4.1 Feature creation from Overdraft data

Terminology Alert: What Is an Overdraft?

An **overdraft** is a short-term credit facility that allows a customer **spend more money than they have in your bank account**, usually up to a pre-approved limit. It's essentially a loan from the bank that activates when their account balance drops to zero or below.

Key Features

- **Linked to Current Accounts:** Most commonly available on current accounts, not savings accounts
- **Pre-approved Limit:** You can overdraw up to a set amount agreed with your bank
- **Interest & Fees:** You pay interest only on the amount you use, and some banks charge overdraft fees
- **Flexible Repayment:** No fixed EMIs—repayment happens whenever funds are added back to the account
- **Types:**
  - **Arranged Overdraft:** Pre-agreed with the bank
  - **Unarranged Overdraft:** Happens without prior approval and may incur higher charges
  - **Secured vs. Unsecured:** Some overdrafts require collateral; others don't. Collateral is a valuable asset that a borrower pledges to a lender as security for a loan. It acts like a safety net for the lender—if the borrower fails to repay, the lender can seize the collateral to recover their money

Understanding Overdraft Eligibility:

An overdraft facility is provided **only with current accounts**, not with savings accounts. This distinction is vital when analyzing customer's financial behavior.

Current Account

- **Purpose:** Designed for *daily transactions* like paying bills, receiving salary, and shopping
- **Access:** Comes with a *debit card*, supports *direct debits* and *standing orders*
- **Overdrafts:** Often includes *overdraft facilities* for short-term borrowing
- **Interest:** Usually *low or no interest*, though some premium accounts offer perks
- **Flexibility:** Unlimited access to funds, ideal for managing everyday finances

Savings Account

- **Purpose:** Meant for *storing money* and earning interest over time
- **Access:** May have *limits on withdrawals* or require notice periods
- **Overdrafts:** *Not available*—you can't spend more than your balance
- **Interest:** Typically *higher interest rates* than current accounts

Quick Comparison

Feature	**Current Account**	**Savings Account**
Main Use	Daily spending	Saving and earning interest
Interest	Low or none	Higher, often tax-free
Overdraft	Available	Not available
Access	Instant and unlimited	May be restricted
Debit Card	Yes	Usually no

Hence this chapter analyzes seven features derived from overdraft behavior over rolling windows (30, 90, and 180 days) for use in predictive modeling of bank default risk.

4.1.1 Overdraft-Based Feature Aggregation

To create overdraft-related features that reflect the financial stress behavior of customers over different time spans using **account-level daily snapshot data**.

Understanding how customers use their overdraft facilities provides critical insight into financial behavior and stress. This section outlines the methodology for creating overdraft-based features, starting from raw account-level data and culminating in meaningful customer-level indicators.

Step 0: Customer Base Preparation

- Work began with a fixed customer base, verified to include a sufficient proportion of vulnerable customers.
- Each customer had consistent identifiers available across all relevant tables.

Step 1: Account-Level Data Extraction

- Using a dynamic `input date`, data for the past **180 days** was extracted by joining the customer base with the daily account-level snapshot table.
- Three observation windows were considered: **30**, **90**, and **180** days.

Step 2: Authorized Limit Stability Check

- It was observed that the `authorized overdraft limit` remained constant for most customers throughout the 180-day period.
- Only a negligible fraction showed variations; hence, this limit was treated as static within each observation window.

Step 3: Time Window Segmentation and Aggregation

For each time window (30, 90, 180 days), the following were computed:

- **Maximum Authorized Limit** over the time window per account.
- **Average Daily Balance** and **Average Overdraft Balance**.
- **Total Unauthorized Overdraft Usage**, **Maximum Overdraft**, and count of:
  - **Overdraft Days**: When balance was negative
  - **Total Active Days**: When account appeared in the snapshot

Step 4: Mathematical Definitions

Let A be the set of all accounts for a customer, and D_a be the days an account $a \in A$ appears in the snapshot. For each account a and day $d \in D_a$:

- $b_{a,d}$: End-of-day balance
- $o_{a,d} = \max(0, -b_{a,d})$: Overdraft amount (if any)

- L_a: Authorized overdraft limit (static within time window)

Now we define the core metrics:
Average Daily Balance (All Balances):

$$\text{AVG\_DAILY\_BALANCE} = \frac{1}{|A|} \sum_{a \in A} \left(\frac{1}{|D_a|} \sum_{d \in D_a} b_{a,d} \right)$$

Average Daily Overdraft Balance (Negative Only):

$$\text{AVG\_DAILY\_OD\_BALANCE} = \frac{1}{|A|} \sum_{a \in A} \left(\frac{1}{|D_a|} \sum_{d \in D_a} \mathbf{1}_{(b_{a,d}<0)} \cdot b_{a,d} \right)$$

Total Unauthorized Usage:

$$\text{TOTAL\_UNAUTH\_USAGE} = \sum_{a \in A} \left(\max_{d \in D_a} \max(0, o_{a,d} - L_a) \right)$$

Maximum Overdraft Observed:

$$\text{TOTAL\_OD} = \sum_{a \in A} \left(\max_{d \in D_a} o_{a,d} \right)$$

OD Days and Total Days:

$$\text{OD\_DAYS} = \sum_{a \in A} \sum_{d \in D_a} \mathbf{1}_{(b_{a,d}<0)}, \quad \text{TOTAL\_DAYS} = \sum_{a \in A} |D_a|$$

Overdraft Ratio:

$$\text{OD\_RATIO} = \begin{cases} 100 \times \frac{OD\_DAYS}{TOTAL\_DAYS}, & \text{if } TOTAL\_DAYS > 0 \\ 0, & \text{otherwise} \end{cases}$$

OD Utilization Percent:

$$\text{OD\_UTILIZATION\_PERCENT} = \begin{cases} 100 \times \frac{\text{MAX\_OD}}{\max_{a \in A} L_a}, & \text{if } \max_{a \in A} L_a > 0 \\ 999999, & \text{otherwise} \end{cases}$$

Step 5: Customer-Level Consolidation

- Aggregated all account-level values at the customer level using:
  - Average (for daily metrics)
  - Sum (for totals and counts)
- Features were created for each time window (30, 90, 180 days)

Step 6: Output Table Construction

- Final customer-level table contained:
  - All overdraft-derived features
  - Time window indicators
- This table was used for downstream modeling and visualization

Step 7: Automation and Reusability

- All logic was encapsulated into a parameterized SQL `PROCEDURE` using dynamic SQL with `EXECUTE IMMEDIATE`.
- Intermediate tables were stored as temporary views.
- Final consolidated table was materialized for easy access in Looker and modeling environments.

4.1.2 Interpretation of Features

- **AVG_DAILY_BALANCE**: Customer's average financial position over all accounts.
- **AVG_DAILY_OD_BALANCE**: Mean depth of financial stress when below zero.
- **TOTAL_OD** and **TOTAL_UNAUTH_USAGE**: Highlight extreme stress conditions and policy violations.
- **OD_DAYS** and **OD_RATIO**: Reflect behavioral persistence of liquidity issues.
- **OD_UTILIZATION_PERCENT**: Captures saturation and dependency on credit limit.

4.1.3 Feature Selection using Information Value (IV)

After engineering the key behavioral features — including overdraft utilization metrics, debt-to-income ratios, and essential spend-to-income ratios — it was essential to assess their predictive utility. For this purpose, we adopted the **Information Value (IV)** framework, a robust metric often used in credit scoring and binary classification tasks to quantify how well a feature separates two classes: in our case, vulnerable vs. non-vulnerable customers.

Definition of Information Value (IV)

Information Value is based on the concept of **Weight of Evidence (WoE)**, which compares the distribution of good (non-vulnerable) and bad (vulnerable) outcomes across different bins of a feature. For a feature divided into k bins, let:

- g_i: Proportion of non-vulnerable customers in bin i
- b_i: Proportion of vulnerable customers in bin i

Then, the **Weight of Evidence** for bin i is:

$$\text{WoE}_i = \ln\left(\frac{g_i}{b_i}\right)$$

And the **Information Value** for the feature is:

$$\text{IV} = \sum_{i=1}^{k} (g_i - b_i) \cdot \ln\left(\frac{g_i}{b_i}\right)$$

Interpretation of IV Values

- **IV less than 0.02**: Not useful for prediction
- **IV between 0.02 and 0.1**: Weak predictive power
- **IV between 0.1 and 0.3**: Medium predictive power
- **IV between 0.3 and 0.5**: Strong predictive power
- **IV greater than 0.5**: Suspiciously powerful (may indicate data leakage)

Observations on Our Features

The following observations emerged after computing IV values:

- `OD_UTILIZATION_PERCENT`, `OD_RATIO`, and `AVG_DAILY_OD_BALANCE` exhibited **IV ¿ 0.3**, making them strong predictors of financial vulnerability. These features capture sustained stress and dependency on credit buffers.

Conclusion

By combining domain understanding with IV-based ranking, we could systematically select a set of features that not only hold behavioral relevance but also offer measurable predictive strength. These selected features feed directly into downstream modeling and visualization stages, allowing us to focus on signals that matter.

4.1.4 Correlation Analysis

Let $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$ represent the customer-level feature vector, where:

$$
\begin{aligned}
x_1 &= \text{OD\_DAYS} \\
x_2 &= \text{AVG\_DAILY\_BALANCE} \\
x_3 &= \text{AVG\_DAILY\_OD\_BALANCE} \\
x_4 &= \text{TOTAL\_UNAUTH\_USAGE} \\
x_5 &= \text{TOTAL\_OD} \\
x_6 &= \text{OD\_RATIO} \\
x_7 &= \text{OD\_UTILIZATION\_PERCENT}
\end{aligned}
$$

Define the sample Pearson correlation matrix R as:

$$R_{i,j} = \frac{\text{Cov}(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}}, \quad \text{for } i, j = 1, \ldots, 7$$

Key Correlation Observations:

- x_1 **(OD_DAYS) and** x_6 **(OD_RATIO)** show *strong positive correlation*, since both are derived from the number of days in overdraft. However, OD_RATIO normalizes OD_DAYS by total appearance, adding interpretability across different activity levels.

- x_2 **(AVG_DAILY_BALANCE) and** x_3 **(AVG_DAILY_OD_BALANCE)** exhibit *moderate negative correlation*, as customers with consistently high balances tend to avoid being in overdraft. Yet, overlap exists due to variability in daily cash flow behavior.

- x_4 **(TOTAL_UNAUTH_USAGE) and** x_7 **(OD_UTILIZATION_PERCENT)** appear *relatively correlated*. Unauthorized usage spikes imply utilization remains high.

- x_5 **(TOTAL_OD)** moderately correlates with both OD_DAYS and OD_UTILIZATION, but only reflects peak stress rather than persistence.

- Other pairs, such as (x_4, x_6) or (x_3, x_7), were observed to be *weakly correlated*, indicating dimensional uniqueness across features.

Conclusion: Overall, the majority of pairwise correlations fall below the standard multicollinearity threshold ($|R_{i,j}| < 0.7$), preserving feature diversity and allowing robust inclusion in downstream predictive models without risk of redundancy or variance inflation.

Exploratory Visual Diagnostics

To better understand how each feature behaves across customer groups, we performed a series of visual analyses. These plots are essential for:

- Identifying patterns in behavior between defaulters and repayers

- Detecting extreme values or long-tail distributions

- Validating whether the features carry meaningful, separable signals

Useful Visuals [Created by Dummy Data]:

1. **KDE (Kernel Density Estimation) Plots:**

    - Plot `avg_daily_balance` to look at liquidity behavior where
      1 := Vulnerable
      0:= Non-Vulnerable

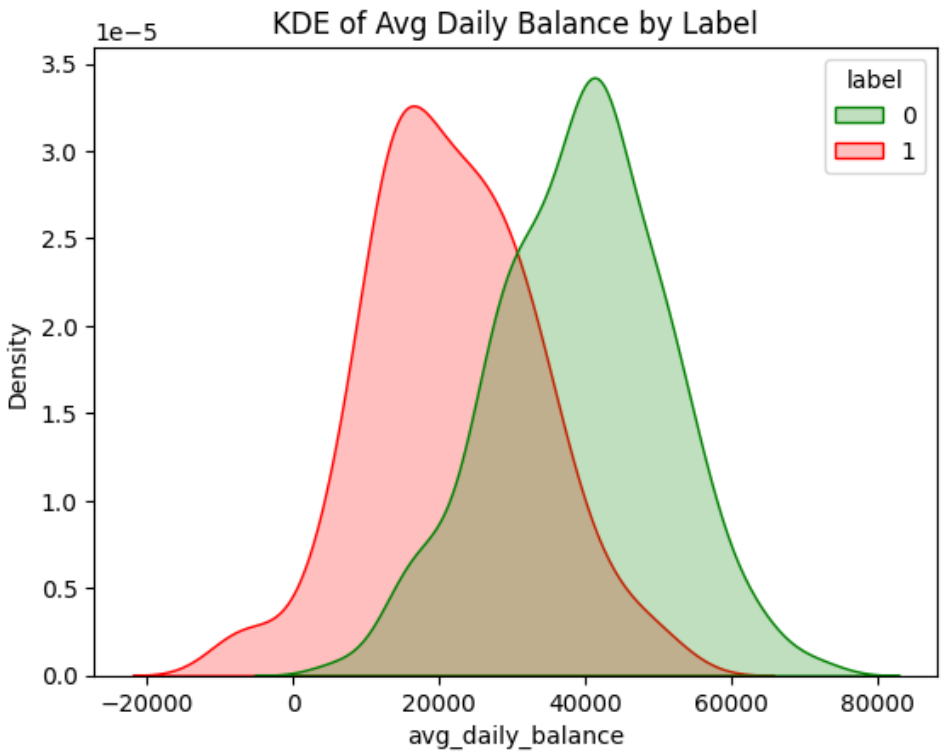


Figure 4.1: KDE of Avg Daily Balance by Label

- Separates KDE curves for Vulnerable customers vs. non-Vulnerable customers
- Interpretation: We look for shift in central tendency or shape

2. **Boxplots:**

- For `max_unauth_usage` and `max_od`
- Shows stress extremes and outliers
- Helps detect volatile financial behavior

3. **Scatter Plots:**

- Look at

max_unauth_usagevs. od_utilization_percent

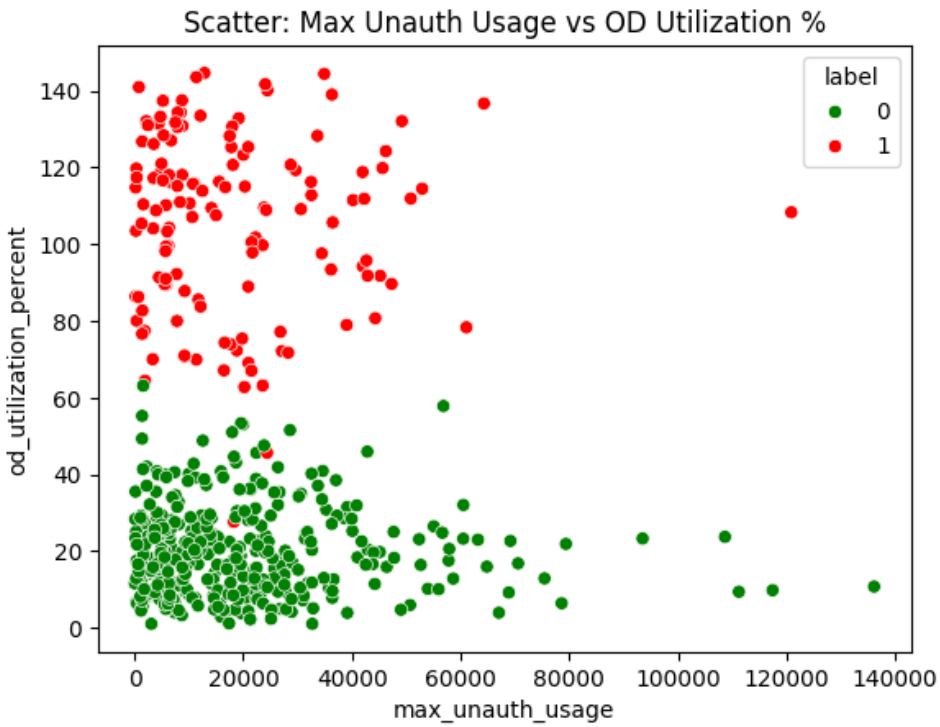


Figure 4.2: Histogram of OD Utilization Percent

- Use color to label defaulters vs. non-defaulters
- Purpose: To visually assess feature independence and cluster separation

Pre-processing Before Plotting:

- Winsorize top 1% of `od_utilization_percent` to reduce extreme distortion
- Apply $\log(x + 1)$ transformation to:
  - `max_unauth_usage`
  - `max_od`
- Convert continuous features like `od_ratio` into buckets to categorize low, medium, high:
  - Low: 0–0.25
  - Medium: 0.25–0.75
  - High: 0.75–1

Conclusion

These visual diagnostics provide:

- Intuition behind how features behave across customer types
- Support for feature selection and transformation decisions
- Early validation of predictive potential, even before modeling

They form an important bridge between data understanding and formal statistical modeling. Figures such as KDE plots, PCA biplots, and boxplots should be visually inspected and included in the appendix or analysis section.

4.1.5 Automated Feature Engineering Procedure

To streamline the entire feature generation process, we designed a reusable **SQL stored procedure** that accepts a single input parameter: the reference start date. From this input, the procedure computes features for the preceding **180-day window**.

Input Parameter:

- `START_DATE` (date): The anchor date from which the 180-day backward period is computed.

Procedure Workflow:

1. **Date Filtering:** Extracts account-level daily snapshot data within the window `START_DATE - 180` to `START_DATE` using dynamic filtering.
2. **Customer-Account Join:** Performs a `LEFT JOIN` between the customer base and daily account data using account identifiers.
3. **Authorized Limit Stability Check:** Computes the maximum arranged overdraft limit per account and discards accounts with limit variations, under the assumption that such variation is negligible.
4. **Window-Based Aggregations:** Constructs three temporary tables for 30-day, 90-day, and 180-day windows. Each table:
    - Calculates max authorized limit
    - Computes average balance, overdraft depth, and frequency metrics
    - Produces daily features at the account level
5. **Customer-Level Aggregation:** For each window, aggregates account-level statistics to the customer level using `GROUP BY CUSTOMER_ID`.
6. **Final Feature Table Creation:** Combines selected customer-level features into a single final output table, which is created as a **physical table** in the database for downstream usage.
7. **Temporary Table Management:** All intermediate computations (30-day, 90-day, 180-day windows) are stored in `temporary tables` to avoid storage overhead.

Dynamic SQL Usage: Many segments of the procedure utilize `EXECUTE IMMEDIATE` for dynamic query generation. String interpolation is handled via standard `%s` formatting to inject runtime values such as date ranges or table suffixes. This ensures flexibility and parameterization across time windows.

Benefits:

- Fully reusable for any reporting or model date
- Ensures consistency in feature definition
- Efficient memory usage through temporary tables
- Enables batch runs or scheduling for automation

4.2 Debt-to-Income Ratio Features

The following section summarizes how debt-income features were made.

What is Debt?

Debt refers to the total amount a customer owes the bank or third-party lenders. This includes formal liabilities such as:

- Personal loans
- Mortgages
- Credit card dues
- BNPL (Buy-Now-Pay-Later) balances

For this feature, we aggregate all outstanding debt **as on the input date** at the customer level, across all linked accounts.

What is Income?

Income is defined as the sum of regular credits into a customer's account, typically salary or business inflow. To obtain a stable signal, we use:

- **Daily account-level credits** from transaction data aggregated to **customer-month level** Over a rolling **180-day window** before the input date

This provides monthly income information for each customer over the past 6 months.

Time-Windowed Debt-to-Income Ratios

We construct a final table with three Debt-to-Income (DTI) ratios, based on the last:

- **1 month**
- **3 months**
- **6 months**

Let:

$$\begin{aligned} D_i &= \text{Total Debt for customer } i \text{ (on input date)} \\ I_i^{(T)} &= \text{Sum of income over last } T \text{ days} \\ \text{DTI}_i^{(T)} &= \frac{D_i}{I_i^{(T)}} \end{aligned}$$

Where T takes values 30, 90, and 180.

Data Pipeline Overview

The following procedure is implemented in SQL and parameterized by `input_date`:

1. **Debt Extraction:**
    - Left join the customer base with the debt table
    - Filter the snapshot to only include rows as of the `input_date`
    - Aggregate debt to the customer level
2. **Income Extraction:**
    - Join the customer base with income transaction table
    - Restrict to last 180 days from `input_date`
    - Group data by customer and month to get monthly income
    - Create aggregations for the last 1, 3, and 6 months
3. **Final Table Construction:**
    - Join the debt and income aggregations
    - Compute DTI for 1, 3, and 6 months
    - Store in a permanent output table

Final Output Table Schema

The final output table has the following columns:

Column	Description
cust_num	Unique customer identifier
debt	Total debt as of input date
last_month_income	Sum of income in the past 30 days
last_month_dti	Debt-to-income ratio for past 30 days
last_3_months_income	Sum of income in past 90 days
last_3_months_dti	Debt-to-income ratio for past 90 days
last_6_months_income	Sum of income in past 180 days
last_6_months_dti	Debt-to-income ratio for past 180 days

Implementation Notes

The SQL procedure uses `EXECUTE IMMEDIATE` with templated date logic to ensure flexibility. Temporary tables are used for intermediate stages, and only the final aggregated customer-level table is persisted.

This structured multi-window DTI approach enables risk segmentation at various time scales and can capture both sudden financial shocks and long-term financial behavior.

Procedure for Feature Creation

To generate the final DTI feature table, a parameterized SQL procedure was developed. The following steps summarize the complete logic:

1. **Debt Extraction from Bureau Data:**
    - Extract the `total outstanding debt` for each customer from the bureau snapshot as of the `input_date`.
    - This provides a consolidated debt value per customer.
2. **Income Extraction from Transaction Data:**
    - Pull categorized transaction-level data from the last 180 days using a dynamic filter based on the `input_date`.
    - Filter for income-relevant credits (e.g., salary, pensions, business inflows).
    - Aggregate daily credits to the customer level.
3. **Time Windowed Aggregation:**
    - Using temporary tables, compute customer-level income over:
        - **Last 30 days**
        - **Last 90 days**
        - **Last 180 days**

- Each window forms a temporary summary with `cust_num` and corresponding income sum.

4. **Final Join and Ratio Computation:**
   - Join the debt table with all three income aggregations on `cust_num`.
   - Compute the DTI ratios as:

$$\texttt{last\_month\_dti} = \frac{\texttt{debt}}{\texttt{last\_month\_income}},$$

$$\texttt{last\_3\_months\_dti} = \frac{\texttt{debt}}{\texttt{last\_3\_months\_income}},$$

$$\texttt{last\_6\_months\_dti} = \frac{\texttt{debt}}{\texttt{last\_6\_months\_income}}$$

5. **Persisting the Final Table:**
   - Store the final joined and computed data into a permanent physical table for downstream use.
   - Example output table: `dti_features_final`

This approach enables automated DTI tracking at multiple horizons and supports vulnerability modeling, financial health profiling, and customer segmentation initiatives.

4.3 Essential Spend-to-Income Ratio Features

This section summarizes how Essential spend features were made.

What is Essential Spend?

Essential spend refers to routine and unavoidable expenditures that a customer incurs, such as:

- Rent and housing-related payments
- Utility bills (electricity, gas, water)
- Grocery and food expenses
- Educational expenses
- Basic healthcare and insurance

These are considered necessary expenses and reflect the cost of living rather than discretionary spending. In our system, transactions are classified as "essential" using merchant category codes (MCCs) and keyword tagging in the transaction descriptions.

What is Income?

As in the DTI ratio, income is defined as the inflow into the customer's account, calculated from transaction-level credits, typically:

- Salary credits
- Pension or subsidies
- Business-related inflows

Income is aggregated from account-level data to customer-month level over a rolling 180-day window.

Time-Windowed Spend-to-Income Ratios

We compute three time-specific Essential Spend-to-Income (ESI) ratios:

- **ESI_30:** Last 30 days of essential spending over income
- **ESI_90:** Last 90 days of essential spending over income
- **ESI_180:** Last 180 days of essential spending over income

Let $S_i^{(T)}$ be the total essential spend for customer i over the past T days, and $I_i^{(T)}$ the total income for the same window. Then:

$$\text{ESI}_i^{(T)} = \frac{S_i^{(T)}}{I_i^{(T)}}, \quad T \in \{30, 90, 180\}$$

Data Pipeline Overview

The process is implemented via SQL, taking a dynamic `input_date` as the anchor point.

1. **Essential Spend Extraction:**
    - Classify and filter transactions marked as essential
    - Aggregate essential spend to customer-month level for the past 6 months
2. **Income Extraction:**
    - Same process as previous features: join income transactions, group by month
    - Aggregate for the last 30, 90, and 180 days
3. **Final Ratio Computation:**
    - Join customer-level essential spend and income data
    - Compute spend-to-income ratios for each time window
    - Store results in a persistent table, with temp tables used for intermediate aggregation

Final Output Table Schema

Column	Description
cust_num	Unique customer identifier
last_month_ess_spend	Total essential spend in past 30 days
last_month_income	Total income in past 30 days
esi_30	Essential Spend-to-Income ratio for 30 days
ess_spend_90	Total essential spend in past 90 days
income_90	Total income in past 90 days
esi_90	Essential Spend-to-Income ratio for 90 days
ess_spend_180	Total essential spend in past 180 days
income_180	Total income in past 180 days
esi_180	Essential Spend-to-Income ratio for 180 days

Implementation Notes

The SQL procedure uses `EXECUTE IMMEDIATE` statements to dynamically substitute dates and table logic. Temporary views store intermediate aggregations, while a single permanent output table holds all computed ratios.

This feature reflects a customer's cost-of-living burden relative to earnings and helps uncover risk due to high basic expenditure load — even in the absence of formal debt.

Procedure for Feature Creation

A parameterized SQL procedure was designed to automate the feature engineering. The key steps are outlined below:

1. **Transaction Data Extraction:**
    - Pull categorized account-level transaction data for the past **180 days** from the `input_date`.
    - Filter to include only transactions marked as "essential spend".
2. **Essential Spend Aggregation:**
    - Create temporary tables to aggregate essential spend at `customer-level` for the last:
        - **30 days**
        - **90 days**
        - **180 days**
3. **Income Aggregation:**
    - Extract income-related credits from transaction data for the same time windows.
    - Aggregate to customer level using temporary tables similar to essential spend.

4. **Join and Ratio Computation:**
    - Join essential spend and income aggregates by `cust_num`.
    - Compute the **Essential Spend-to-Income Ratio** for each window:

$$\texttt{essential\_spend\_ratio} = \frac{\texttt{essential\_spend}}{\texttt{income}}$$

    - This is computed separately for 30-day, 90-day, and 180-day windows.
5. **Final Table Storage:**
    - Store the results in a consolidated physical output table for future modeling or dashboarding.

This methodology enables fine-grained understanding of spending patterns relative to income and serves as a key indicator for potential financial vulnerability.

Chapter 5

The Second Problem - Making Looker Dashboards for Visualization

5.1 Looker Dashboard: Overview and Implementation

5.1.1 What Is Looker?

Looker is a cloud-hosted Business Intelligence (BI) and data analytics platform, now part of Google Cloud, which enables users to explore and visualize real-time data directly from their data warehouses—without the need for data duplication. It operates entirely in-browser and follows a Software-as-a-Service (SaaS) delivery model. Looker bridges the gap between analytics and operational workflows, allowing both data engineers and business users to collaborate seamlessly.

5.1.2 Looker's Ownership and Purpose

Originally built by Looker Data Sciences (founded 2012), the platform was acquired by Google in 2019 for $2.6 billion and has since been integrated into Google Cloud. Its mission is to provide a governed semantic layer for organizational data, ensuring consistent and trusted metrics across reports and dashboards.

5.1.3 Data Access and Architecture

Looker queries live data using an *in-database architecture*. It connects—via standard SQL—to data warehouses like BigQuery, Snowflake, Redshift, and Cloud SQL, executing queries in real time. Looker does not store data internally, instead fetching fresh data on demand, which minimizes storage overhead and ensures up-to-date results.

5.1.4 Looker Studio vs. Enterprise Looker

Looker offers two related products:

- **Looker Studio (formerly Data Studio):** Free tier suited for individuals and small teams, with over 800 connectors (e.g., Google Sheets, MySQL)
- **Looker Enterprise (Google Cloud Looker):** Advanced BI and embedded analytics platform with LookML modeling, governance, and API integrations

5.1.5 User Licenses and Subscription Model

Looker Enterprise is sold via annual or multi-year contracts and is generally priced based on:

- **Platform Edition:** Standard, Enterprise, or Embedded, each supporting a base number of users and API calls.
- **User Roles:** - *Viewer Users*: $30–$50/month — can view/report only - *Standard Users*: $60/month — can create dashboards, run queries - *Developer Users*: $125+/month — can edit LookML, develop models
- **Overall licensing:** Typically starts around $35K/year for small teams; can exceed $150K+ for larger setups.

5.1.6 Free vs. Paid Versions

Looker Studio (Free) includes:

- Basic reporting and visualization
- Integration with Google ecosystem (e.g. Sheets, Analytics)

Looker Enterprise adds:

- LookML-based data modeling
- Governance control and versioning via Git
- In-browser SQL Runner, API access, embedding analytics
- Developer workflows and sandbox environments

5.1.7 How Looker Is Used

- Analysts define models in **LookML**, a semantic layer reducing SQL for users .
- Business users explore data via the **Explore UI**, filtering and visualizing datasets freely.
- Dashboards organize visuals into reusable, shareable spaces with filters, scheduling, and alerts.
- Embedded analytics allow integrations into internal tools and customer applications.

5.1.8 LookML Example

For example, a simplified view for and imaginary project named: project, dataset named: dataset & table named : `gambling_summary` with imaginary columns `customer_id`, `gambling_spend`, `gambling_type` etc.

```
view: gambling_summary {
  sql_table_name: project.dataset.gambling_summary ;;

  dimension: customer_id {
    type: string
    sql: ${TABLE}.customer_id ;;
  }

  measure: total_gambling_spend {
    type: sum
    sql: ${TABLE}.gambling_spend ;;
  }

  dimension: is_lottery {
    type: yesno
    sql: ${TABLE}.gambling_type = 'Lottery' ;;
  }

  # Additional measures and dimensions...
}
```

This defines a reusable semantic model linked to your final feature table and exposes fields for dashboard use.

5.1.9 Accessing Looker: Who Subscribes?

Large enterprises subscribe to the paid Looker platform, often across business units. Most organizations maintain a few Developer and Standard licenses, with broader Viewer access. In contrast, individual analysts or small teams often use Looker Studio (free or Pro) via Google Workspace.

5.1.10 Getting Started

- **Enterprise:** Contact Google Cloud sales for a personalized evaluation, demo, and onboarding.
- **Looker Studio Pro:** Subscription is available per Workspace admin; priced per user/project basis.
- **Free Option:** Anyone can sign up for Looker Studio and embed Google ecosystem data without cost.

5.1.11 Capabilities Available

Enterprise Looker unlocks:

- Centralized LookML modeling with consistent metrics
- Real-time querying via SQL runner and embedded analytics

- Fine-grained permissions and scheduled reports
- API integration and Git version support

Looker Studio (free) allows:

- Ad-hoc visualizations, filtering, and sharing
- Google data connectors (Sheets, Analytics)

5.1.12 Example Workflow

1. Model your final features (e.g., overdraft, DTI, ESI) in a database table.
2. Define a LookML project with models and views.
3. Use Explore to validate distributions, trends, and variations.
4. Build dashboards with KPIs and drill-downs.
5. Share or embed dashboards, schedule runs, and set alerts.

5.1.13 Conclusion

Looker is a powerful, scalable BI platform providing real-time, governed analytics across corporate data. With its semantic modeling, flexible user roles, and embedded capabilities, it is ideal for structured dashboards like your OD, DTI, and ESI monitoring tools.

5.1.14 Gambling Pattern Analysis Dashboard

To understand financial vulnerability through gambling behavior, I created a Looker dashboard backed by a custom procedure and machine learning-based detection model.

Understanding Gambling in the UK Context

The United Kingdom has one of the most liberalized and highly regulated gambling markets in the world. Gambling is legal and widespread, with activities ranging from betting shops and online casinos to lotteries and sports wagering. It is regulated by the **UK Gambling Commission**, which sets standards for licensing, advertising, responsible gambling, and consumer protection.

Key Types of Gambling in the UK:

- **Betting:** Horse racing, football, and other sports via bookmakers or apps
- **Online Gambling:** Including virtual slots, poker, and live casino games
- **National Lottery:** Widely accessible and heavily advertised
- **Scratchcards and Instant Wins:** Often sold at convenience stores

- **Fixed-Odds Betting Terminals (FOBTs):** Found in betting shops (with usage caps due to policy changes)

Gambling and Financial Risk: While gambling is a legal leisure activity, it can lead to significant financial problems when done compulsively or excessively. The UK Gambling Commission estimates that a small but concerning portion of the population are at risk of *problem gambling* — characterized by:

- Repeated losses leading to debt accumulation
- Use of credit (e.g., overdrafts, BNPL) to fund gambling activity
- Attempts to recover losses through higher-stakes bets
- Deterioration in personal finances, relationships, or mental health

Regulatory Measures and Data Indicators: Banks and financial institutions are increasingly expected to identify and support customers at risk of gambling harm. Some key indicators include:

- High-frequency transactions to known gambling merchants
- Gambling transactions exceeding income or occurring during late-night hours
- Overlap between gambling behavior and credit product usage (e.g., loans, overdrafts, BNPL)

Relevance to This Work: In this internship, gambling-related transactions were studied using transaction-level data joined with merchant classification and income data. A previously trained model was used to classify merchants as gambling-related based on names, behaviors, and linkages. Features such as gambling-to-income ratio, frequency, merchant trends, and overlaps with vulnerability markers were visualized via Looker dashboards to assess financial risk.

Automated Data Pipeline and Procedure

The foundation of the dashboard is a SQL-based **procedure** I designed, which accepts user-defined `start_date` and `end_date` as arguments.

The procedure:

- Extracts **gambling-related transactions** from the main transaction table using a pre-trained classifier model.
- Brings the transaction level Gambling data to customer level
- Pulls income data in customer level
- Joins with customer income data to compute the **Gambling-Expense-to-Income Ratio** (GSIR)
- Stores final outputs as `views` which are consumed by the Looker model
- Creates intermediate `views` which are connected to a Looker model to expose fields for visualization.

Gambling Detection Model

A machine learning model (built outside this dashboard) determines if a transaction is gambling-related using:

- Merchant name keyword tagging
- Transaction frequency and time-of-day heuristics
- Brand-level associations and merchant clusters

Only the transactions classified as gambling are passed into this dashboard pipeline.

Dashboard Layout and Features

The dashboard consists of multiple dynamic tiles and charts, allowing business users to filter by:

- Type of gambling (`Betting`, `Casino`, `Lottery`)
- Date range (linked to the procedure inputs)
- Customer geography and demographics

Summary Tiles and KPIs

The top section includes a summary panel with key metrics:

- Total number of gambling transactions
- Total transaction value
- Unique customers engaging in gambling
- Median and average transaction value for gambling

These metrics auto-update based on selected filters and time ranges.

Behavioral Insights and Visuals

The dashboard also includes:

- **Line Chart:** Trend of transaction count per merchant over time
- **Pie Chart:** Ratio of gamblers who are vulnerable vs. not vulnerable
- **Horizontal Bar Graph:** Top gamblers by spending amount
- **Sankey Diagram:** Flow from gambling behavior to loan burden and financial vulnerability
- **Boxplots:** Distribution of gambling-to-income ratio across customer segments

Dynamic Date Controls and Filters

Tiles and filters such as:

- `Start Date` and `End Date` (user-editable)
- `Gambling Type Category`
- `Merchant Brand`
- `Transaction Channel (Online vs Offline)`

are linked directly to the stored procedure and views to ensure the dashboard is interactive and self-refreshing.

Business Impact

This dashboard is designed to:

- Track behavioral risks in near-real time
- Segment high-risk customer cohorts
- Support policy decisions around risk-based loan issuance and financial support

By integrating engineered features like `Gambling-to-Income Ratio` and linking them with customer vulnerability labels, this solution creates a powerful, data-driven lens on financial stressors in the banking ecosystem.

5.1.15 BNPL Overuse Analysis Dashboard

Buy Now Pay Later (BNPL) schemes offer flexible short-term credit, but repeated or high-volume usage may signal financial stress. I developed a Looker dashboard that captures BNPL spending behaviors and links them to income and risk profiles.

Understanding BNPL in the UK Context

Buy Now, Pay Later (BNPL) is a rapidly growing form of short-term credit that allows consumers to spread the cost of purchases over a few weeks or months, often with no initial interest. In the UK, BNPL usage surged during and after the COVID-19 pandemic, particularly among younger consumers and in online retail spaces.

Key Characteristics of BNPL:

- **Instant credit:** Offered at the checkout point — both online and in-store — allowing users to delay payment without traditional credit checks.
- **No interest (initially):** Most BNPL plans offer 0% interest if repaid on time within the agreed period.
- **Providers:** Popular UK BNPL providers include *Klarna, Clearpay, Laybuy, PayPal Pay Later*, and *Zilch*.

- **Retail Integration:** Embedded into e-commerce platforms and apps, often as one-click options for easy access.

Repayment Terms: When Is the Money Returned?

BNPL repayment schedules typically fall into one of the following formats:

- **Pay in 30 Days:** The customer is required to repay the full amount within 30 days of purchase — often interest-free.
- **Pay in 3 or 4 Installments:** Payments are automatically split over a few weeks (e.g., 3 monthly payments or 4 biweekly payments). The first installment is usually due at the time of purchase.
- **Flexible Finance:** Some providers offer longer repayment plans (e.g., 6–12 months), which may include interest after a certain grace period.

BNPL providers automatically charge the customer's linked debit or credit card on the scheduled dates. Failing to make timely repayments can lead to:

- Late fees or penalty charges (depending on provider policies)
- A negative impact on credit history (especially with providers that report to credit bureaus)
- Potential involvement of debt collection agencies

These timelines and consequences highlight the importance of **tracking BNPL commitments** in customer risk profiling. A high number of open BNPL contracts or delayed repayments may signal early financial distress — an area this internship aimed to detect via dashboards and feature engineering.

Regulatory Concerns and Risks:

- **Debt Accumulation:** Customers may accumulate multiple BNPL debts across platforms without consolidated tracking.
- **Credit Reporting:** Historically, many BNPL providers did not report to credit bureaus, limiting visibility of liabilities.
- **Financial Conduct Authority (FCA):** The UK's FCA has raised concerns about lack of affordability checks and aims to bring BNPL under tighter regulation.
- **Target Demographic:** BNPL is particularly popular among 18–30 year olds — a segment more vulnerable to economic shocks and overconsumption.

Common Use Cases:

- Fashion and apparel
- Electronics and lifestyle products

- Online grocery or cosmetic purchases

BNPL offers convenience and short-term financial flexibility. However, its misuse or overuse may lead to behavioral red flags — such as debt stacking, impulsive consumption, or invisible liabilities — which are of direct interest in risk profiling and customer vulnerability analysis.

In this internship, BNPL behaviors were studied not only in terms of volume and frequency, but also their relationship with income, category of spend, and vulnerability patterns across demographic groups.

Automated SQL Procedure and Input Parameters

I created a SQL **procedure** that serves as the backbone of the dashboard. It accepts a `start_date` and `end_date` as parameters and performs the following:

- Extracts BNPL-related transactions from the main transaction table
- Categorizes spend by merchant brand and product category
- Brings the transaction level BNPL data to customer level
- Pulls income data in customer level
- Joins with customer income data to compute the **BNPL-to-Income Ratio**
- Stores final outputs as `views` which are consumed by the Looker model

Detection of BNPL Transactions

BNPL transactions are filtered using merchant code classifications, merchant name patterns, and narrative keywords. The identification logic was defined in collaboration with domain experts and encoded in the data model.

Dashboard Design and Filter Controls

The Looker dashboard is structured to explore BNPL behavior across various customer segments and transaction patterns. It features:

- Filters for:
  - Date Range (linked to procedure)
  - BNPL Provider / Brand
  - Age group

Key Metrics Displayed

- Number of BNPL transactions and total value
- Number of unique BNPL users
- Average and median BNPL spend per user
- BNPL-to-Income Ratio (customer level)

Visual Elements and Behavioral Patterns

The dashboard contains dynamic charts and visual summaries:

- **Bar Graph:** BNPL transaction volume by category and brand
- **Pie Chart:** Distribution of users by BNPL-to-Income Ratio buckets
- **Sankey Diagram:** Relationship between BNPL usage and vulnerability indicators
- **Heatmaps:** BNPL spending intensity across regions and age groups
- **Trend Line:** Monthly change in BNPL usage over time

Output Tables and Data Aggregations

I aggregated data at the customer level across selected time windows (e.g., last 30, 90, 180 days). Each record includes:

- Total BNPL spend
- Total income
- BNPL-to-Income ratio for the chosen time window
- Count of distinct brands used
- Average repayment delay (if available)

Business Use and Impact

This dashboard helps financial analysts and risk managers:

- Spot customers with escalating BNPL usage
- Identify risky cohorts (e.g., high spenders in low-income brackets)
- Correlate BNPL behavior with default probability or loan eligibility
- Take early preventive action or trigger financial guidance

Conclusion

By automating the pipeline and connecting BNPL features to customer context, this dashboard enables real-time business monitoring. It transforms transactional noise into behavioral intelligence — supporting responsible lending and early intervention.

Chapter 6

Creating Slides for a Knowledge Sharing Session

In the final phase of my internship, I created slides and presented with My senior (Skip-level-manager) the natural language processing (NLP) techniques, particularly **word embeddings**, which transform textual data into numerical representations suitable for machine learning models.

The slides covered:

- A brief transition from **Count Vectors** and **TF-IDF** to **Word2Vec**
- Simple ideas behind **CBOW** and **Skip-Gram**
- Introduction to contextual embeddings like **LSTM** and **GRU**

More than a technical presentation, it was a fulfilling interactive session. The team was deeply engaged, asking questions about the meaning of "similarity" between words, real-world use cases, and the interpretability of embeddings.

The discussion stayed focused on the intuition behind embeddings — how words with similar meanings or contexts are placed closer together in high-dimensional space. Team members shared their thoughts on how this mathematical representation of language helps machines "understand" relationships between words.

There was active curiosity about how words like *bank*, *loan*, and *default* might behave in vector space, and how these representations evolve as models become more contextual.

The session was well-received, and I was happy to learn later that some of my slides and visual explanations were used in larger team presentations — a fulfilling sign that the ideas resonated beyond the session itself.

Chapter 7

Impact and Outcomes

This chapter presents the observed effects and real-world significance of the work carried out during my internship at HSBC. From the perspective of both technical performance and business insight generation, the project delivered tangible value and promising outcomes.

Modeling Outcomes and Predictive Strength

The feature engineering process, especially the extraction and normalization of behavioral signals like `OD_UTILIZATION_PERCENT`, `OD_RATIO`, and `DEBT_TO_INCOME_180D`, played a crucial role in improving the performance of downstream models. These overdraft-related features showed consistently **high feature importance scores** across models, highlighting their relevance in assessing customer vulnerability.

The predictive models built on top of these and many other features demonstrated strong performance, with the best-performing model achieving an **AUC (Area Under the ROC Curve) score exceeding 95%**. This indicates that the engineered features were not only interpretable but also powerful in distinguishing between vulnerable and non-vulnerable customer profiles.

Business Impact through Looker Dashboarding

Alongside model development, a parallel focus was placed on visualization and insight generation through the Looker BI platform. The Looker dashboards created during the internship played an important role in business storytelling and decision support. These dashboards translated backend data procedures into meaningful visual layers, enabling quick exploration of customer behaviors, spending patterns, and risk signals.

The visualizations and dashboards were appreciated for their clarity, flexibility, and real-time adaptability through parameterization. Key outputs — such as gambling pattern summaries, BNPL usage segmentation, and overdraft dependency charts — were showcased in team discussions and management presentations. Notably, some of the insights and slides were **leveraged for broader stakeholder presentations**, amplifying the visibility of the work and its strategic value.

Presentation Was Successfully Done

The presentation of the project findings was well received by both the business and technical teams. The Looker dashboards, along with the supporting insights and slides, were appreciated for their clarity and practical relevance. Key parts of the analysis were leveraged in wider discussions and internal reviews, validating the usefulness of the work beyond the scope of the internship.

Conclusion of Internship

The internship was completed successfully, with all deliverables submitted and well received. The work covered the full spectrum of data science responsibilities — from data extraction and transformation to modeling, visualization, and business communication.

Beyond technical output, the internship fostered a deeper understanding of financial data ecosystems and the importance of interpreting customer behavior not just for prediction, but for trust, support, and inclusion. The experience, mentorship, and collaborative energy from the Finance Support team in the Decision Sciences unit greatly shaped both the quality of the work and the joy in delivering it.

Future Work

While this internship laid the groundwork for understanding vulnerability through financial behavior, several directions remain open for future exploration:

- **Enhance feature engineering by incorporating temporal patterns and sequence-aware models :** We used aggregated features (e.g., max OD, avg spend). In future, we could use time-aware features, like ”increasing trend of debt”, ”regular end-of-month OD” or use models that understand sequence
- **Deepen integration with Looker for drill-through analytics and real-time alerting:** Right now, our Looker dashboards show visualizations. Future work could include interactive dashboards where clicking on a customer shows more details, or setting up alerts if some metric crosses a threshold.
- **Expand the behavioral signal library:** We focused on OD, BNPL, gambling. In future, We can explore other patterns like: sudden salary stop, frequent failed payments, new loan applications, gambling-to-income ratio spike. Basically: find new “signals” that might hint at financial stress.

These directions not only aim to improve model performance but also to make risk analytics more adaptive, interpretable and impactful in decision-making.

Conclusion

In this dissertation, I explored how financial behavior can be translated into meaningful indicators of vulnerability. By engineering features around overdraft usage, debt and income dynamics, essential spending, and behavioral triggers like BNPL and gambling, I was able to develop interpretable signals that supported robust predictive models.

I designed reusable SQL procedures, built automated data pipelines, and created interactive Looker dashboards to translate data into real business insights. The overdraft-related features showed strong predictive power, contributing to a model AUC above 95%.

This experience deepened my understanding of data analytics in a financial context and strengthened my ability to combine domain intuition with technical implementation. It was both a challenging and fulfilling journey.

Acknowledgment

I would like to express my heartfelt gratitude to the people who guided, supported, and encouraged me throughout this internship. Their mentorship and kindness shaped both my learning and experience.

- **Mr. Debasish Mishra**, my manager and mentor at HSBC, for his unwavering support, clear guidance and for entrusting me with meaningful responsibilities that accelerated my growth
- **Mr. Prasenjit Talukder**, whose calm energy, clarity of thought and ever-approachable nature made every discussion enriching and encouraging
- **Mr. Dipankar Nath**, for his remarkable patience and clarity, answering every question, no matter how small, with depth and kindness from the very beginning of my internship
- **Prof. Anisur Rahaman Molla**, my academic guide at ISI, for his insightful feedback, steady encouragement and ever-prompt responses
- **My Team at HSBC**, for their warmth, humor, and collaborative spirit — their positivity made even the most complex challenges feel lighter
- **Indian Statistical Institute**, for granting me the opportunity to work under this program and for shaping my academic journey with its vibrant and rigorous environment
- **My Classmates at ISI**, for their constant encouragement, shared learning and friendly motivation that kept me focused and grounded
- **My Parents**, for their unconditional support, love and belief in me throughout every step of this journey

Your mentorship, positivity, and support helped me push boundaries, learn quickly and complete this work with confidence and joy.

Bibliography

[1] SQL procedural language (SQL PL) by IBM: https://www.ibm.com/docs/en/i/7.4.0?topic=reference-sql-procedural-language-sql-pl

[2] Geeks for Geeks on Procedural Language/SQL: https://www.geeksforgeeks.org/category/plsql/

[3] Looker Documentation: https://cloud.google.com/looker/docs

[4] Buy now pay later: what you need to know by HSBC UK: https://www.hsbc.co.uk/financial-fitness/managing-debt/buy-now-pay-later-what-you-need-to-know/

[5] A financial safety net, just in case you need it by HSBC UK: https://www.hsbc.co.uk/current-accounts/products/overdrafts/#:~:text=An%20overdraft%20is%20a%20way,you're%20into%20your%20overdraft.

[6] Normalized Nerd - YouTube Channel: https://www.youtube.com/c/NormalizedNerd