National College of Ireland

Configuration Manual

MSc Research Project
FinTech

Sreeram Natarajan
Student ID: 23292661

School of Computing
National College of Ireland

Supervisor: Faithful Chiagoziem ONWUEGBUCHE

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sreeram Natarajan
Student ID:	23292661
Programme:	FinTech
Year:	2024
Module:	MSc Research Project
Supervisor:	Faithful Chiagoziem ONWUEGBUCHE
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	3305
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	8th August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sreeram Natarajan
23292661

1 Software Environment and Dependencies

This project was developed using Python in a Jupyter Notebook environment. All scripts, models, and visualizations were created using widely used libraries for machine learning and data analysis. This section gives a summary of the tools and system setup used throughout the project.

1.1 Programming Language and IDE

- Programming Language: Python 3.10
- Development Environment: Jupyter Notebook (via Anaconda)

1.2 Main Libraries and Packages

The project used the following Python libraries:

- pandas, numpy - for handling and processing data
- scikit-learn - for machine learning models and evaluation
- xgboost - for gradient boosting classification
- imbalanced-learn - for SMOTE and other class balancing methods
- shap - to explain model predictions
- matplotlib, seaborn - for plots and visualizations
- ydata-profiling - for quick exploratory data analysis

1.3 System Configuration

All work was done on a personal laptop with the following setup:

- Device: ASUS VivoBook X415EA
- Operating System: Windows 11 (64-bit)
- Processor: Intel Core i3-1115G4 @ 3.00GHz
- RAM: 8 GB (7.7 GB usable)

1.4 Documentation Tools

- Initial drafts were written in Microsoft Word
- Final report and configuration manual were prepared in Overleaf using LaTeX

2 Dataset Configuration and Preprocessing

2.1 Dataset Source and Filtering

The dataset used for this project was obtained from Lending Club via Kaggle[1]. It contains loan records issued between 2007 and 2020, with approximately 1 million entries originally available. For the purpose of this research, only records with specific loan statuses were retained to focus on delinquent and defaulted loans. The selected statuses were Late (16-30 days), Late (31-120 days), Default, Charged Off, and In Grace Period. After applying these filters, a total of 135,883 records were used for model training and evaluation.

2.2 Target Variable Creation

The original dataset did not include a recovery-specific target variable. To enable supervised learning, a new column named `recovery_action` was created using rule-based logic. This logic was based on real-world financial recovery practices and considered borrower-level attributes such as loan status, loan amount, annual income, debt-to-income ratio (DTI), term, and home ownership.

Several restructuring-related actions such as EMI reduction, moratorium, and term extension were grouped under a single label called `Restructure`. This grouping helped simplify the classification problem and reduce class imbalance. The engineered target variable was included among the final set of features used for modeling.

2.3 Initial Cleaning and Feature Reduction

The raw dataset contained 142 columns. A two-stage feature reduction process was followed:

- Columns with more than 80 percent missing values were removed.
- An additional 96 columns were manually dropped based on domain knowledge, redundancy, irrelevance, or data leakage concerns.

After this, only 16 relevant columns were retained for model development, including the engineered `recovery_action` column. A complete list of all columns that were dropped or retained, along with the reason for each decision, is provided in Appendix A.

[1] `https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1`

2.4 Preprocessing and Transformation

The following transformations were performed to prepare the dataset for modeling:

- Percentage fields such as `int_rate` and `revol_util` were converted to numeric format.
- The `term` column was cleaned to extract numeric duration.
- Missing values in `dti` and `revol_util` were imputed using the median.
- Categorical variables were encoded using one-hot encoding.
- Numeric features were scaled using the `StandardScaler` from the `scikit-learn` library.

3 Recovery Action Label Construction

3.1 Business Logic

The original Lending Club dataset did not contain any predefined labels indicating the recovery strategy to be taken for delinquent or defaulted loans. To enable supervised learning, a new target variable named `recovery_action` was engineered using domain-specific business rules.

This logic emulates real-world recovery decisions taken by financial institutions and was based on borrower-level attributes such as `loan_status`, `loan_amnt`, `annual_inc`, `dti`, `home_ownership`, `term`, and `debt_settlement_flag`. Depending on the combination of these attributes, a specific recovery action was assigned to each borrower.

For example, borrowers with high unpaid loan amounts were classified under "Legal Action", while early-stage delinquents were issued a "Reminder". Borrowers experiencing moderate financial distress were assigned to one of several restructuring options, such as moratorium or EMI reduction. These were later unified under a common label "Restructure" to simplify classification and reduce class sparsity.

3.2 Mapping Rules

The full decision logic used to construct the `recovery_action` variable is presented in Table 1. Each rule is derived from practical lending policies and reflects the borrower's financial situation and stage of delinquency.

Table 1: Rule-Based Logic for Assigning Recovery Actions Based on Loan Status and Borrower Profile

Loan Status	**Condition**	**Recovery Action**	**Reason**
debt_settlement_flag	debt_settlement_flag == 'Y'	Settled - No Further Action	Loan already settled; no further action required
Charged Off	loan > 10000	Legal Action	High-value loss; initiate legal process
Charged Off	3000 ≤ loan ≤ 10000 AND 20000 ≤ income ≤ 40000 AND home_ownership NOT IN ('OWN', 'MORTGAGE')	One-Time Settlement Offer	Mid-sized loan, moderate income, no property ownership; partial recovery is practical
Charged Off	income < 15000 AND loan < 3000 AND dti > 60	Outsource to Collection Agency	Low income, small loan, high DTI; internal recovery not viable
Charged Off	loan ≤ 10000	Legal Notice	Small loan; send formal legal warning
Default	loan > 20000	Legal Action	Large default; pursue legal recovery
Default	else	Outsource to Collection Agency	Small default; better handled by third-party agency
Late (31-120) or Late (16-30 days)	dti > 40 AND income < 35000	Restructure - Moratorium	High DTI and low income; temporary payment pause to avoid escalation
Late (31-120) or Late (16-30 days)	dti > 40 AND income ≥ 35000	Restructure - EMI Reduction	Moderate financial stress; reduce payment load
Late (31-120) or Late (16-30 days)	home_ownership == 'RENT' AND dti > 35	Restructure - EMI Reduction	Financially vulnerable renters; reduced EMI support
Late (31-120) or Late (16-30 days)	30 < dti ≤ 40 AND term contains "36"	Restructure - Term Extension	Medium term loans; extended terms ease payment burden
Late (31-120) or Late (16-30 days)	else	Reminder	Early-stage delinquency; send repayment reminders
In Grace Period	always	Reminder	Minimal delay; gentle reminder
Others	always	Reminder	Default fallback
Error Handling	any exception	Unknown	Unknown reason due to error

3.3 Final Class Distribution

After applying the rule-based logic and grouping restructuring categories under a single "Restructure" label, the final distribution of recovery actions was as follows:

Table 2: Final Distribution of Recovery Action Classes

Recovery Action	**Record Count**
Legal Action	69,295
Legal Notice	30,926
Reminder	13,800
Settled – No Further Action	13,005
One-Time Settlement Offer	7,687
Restructure	1,066
Outsource to Collection Agency	104

This distribution shows significant class imbalance, with the majority of records falling under "Legal Action" and "Legal Notice". To mitigate this imbalance, techniques such as SMOTE, class weighting, and random undersampling were applied during model training (discussed in later sections).

4 Experimental Environment Setup

The project was developed in Python 3.10 using Jupyter Notebook through the Anaconda distribution. Key libraries such as pandas, scikit-learn, xgboost, shap, and imbalanced-learn were used for data handling, modeling, and explanation. A detailed list of packages was already provided in Section 1.

4.1 File Structure

The main project files were stored under a folder named `Final Project`. This included the dataset, Jupyter Notebook, and output files used throughout the research. The contents were arranged as follows:

- `Loan_default_2007_2020.csv` - the filtered dataset used for modeling, based on Lending Club data downloaded from Kaggle
- `lending_club.ipynb` - the main Jupyter Notebook used for data preprocessing, model training, and evaluation
- `loan_default_eda_report.html` - automated profiling report generated using ydata-profiling
- `Appendix.xlsx` - internal tracking of dropped and retained columns (used during feature selection)
- `References/` - folder containing all research papers used for the literature review

This structure helped keep the project organized during development, with all related data, notebooks, and outputs stored in a single location.

5 Key Implementation Code Snippets

This section presents key Python code snippets from the Jupyter Notebook `lending_club.ipynb`, used for data preprocessing, feature engineering, model training, evaluation, and SHAP-based explainability. The code is provided in sequential order to illustrate the step-by-step machine learning pipeline used in this project.

5.1 Loading the Dataset

The following code loads the Lending Club dataset and displays its shape and the first few rows.

```
%matplotlib inline
import pandas as pd

df = pd.read_csv(r"D:\\SREERAM\\IRELAND\\NCI\\Semester3\\
   Practicum\\Jupyter EDA analysis\\Loan_default_2007_2020.csv")

print("Shape before cleaning:", df.shape)
df.head()
```

Listing 1: Loading the dataset and displaying basic info

5.2 Recovery Action Label Assignment

The function below assigns recovery actions to loan records using domain-driven rules. It leverages loan status, DTI, income, term, and home ownership to mirror institutional recovery strategies.

```
def assign_recovery_action(row):
    try:
        if row.get('debt_settlement_flag', 'N') == 'Y':
            return 'Settled - No Further Action'
        status = row['loan_status']
        loan = row.get('loan_amnt', 0)
        income = row.get('annual_inc', 0)
        dti = row.get('dti', 0)
        term = str(row.get('term', ''))
        home = row.get('home_ownership', 'UNKNOWN')
        if status == 'Charged Off':
            if loan > 10000:
                return 'Legal Action'
            elif 3000 <= loan <= 10000 and 20000 <= income <= 40000 and
                home not in ['OWN', 'MORTGAGE']:
                return 'One-Time Settlement Offer'
            elif income < 15000 and loan < 3000 and dti > 60:
                return 'Outsource to Collection Agency'
            elif loan <= 10000:
                return 'Legal Notice'
        elif status == 'Default':
            if loan > 20000:
                return 'Legal Action'
            else:
                return 'Outsource to Collection Agency'
        elif status in ['Late (31-120 days)', 'Late (16-30 days)']:
            if dti > 40 and income < 35000:
                return 'Restructure - Moratorium'
            elif dti > 40 and income >= 35000:
                return 'Restructure - EMI Reduction'
            elif home == 'RENT' and dti > 35:
                return 'Restructure - EMI Reduction'
            elif 30 < dti <= 40 and '36' in term:
                return 'Restructure - Term Extension'
            else:
                return 'Reminder'
        elif status == 'In Grace Period':
            return 'Reminder'
        else:
            return 'Reminder'
    except:
        return 'Unknown'

df['recovery_action'] = df.apply(assign_recovery_action, axis=1)
df['recovery_action'].value_counts()
```

Listing 2: Rule-based function to assign recovery actions

5.3 Dropping High-Missing Columns

To reduce noise and sparsity in the dataset, columns with more than 80% missing values were removed using the following code:

```
# Drop columns with more than 80% missing values
df = df.loc[:, df.isnull().mean() < 0.8]

# Check new shape and remaining columns
print("Shape after dropping high-missing columns:", df.shape)

# Optional: Preview remaining data
df.head()
```

Listing 3: Dropping columns with more than 80% missing values

5.4 Manual Feature Selection Based on Relevance

The dataset originally included over 140 columns. A subset of 16 relevant features (plus the target) was retained, and the rest were dropped. The logic used is shown below.

```
# Step 1: List of useful columns to keep
cols_to_keep = [
    'loan_amnt', 'term', 'int_rate', 'installment', 'grade', '
       loan_status',
    'annual_inc', 'dti', 'home_ownership', 'open_acc', 'total_acc',
    'revol_util', 'delinq_2yrs', 'pub_rec', 'debt_settlement_flag',
    'recovery_action'  # target
]

# Step 2: Get list of all columns
all_columns = df.columns.tolist()

# Step 3: Determine which columns to drop
cols_to_drop = [col for col in all_columns if col not in cols_to_keep]

# Step 4: Drop unwanted columns
df = df[cols_to_keep]

# Step 5: Print summary
print(f"Columns KEPT ({len(cols_to_keep)}):")
print(cols_to_keep)
print("\nColumns DROPPED ({0}):".format(len(cols_to_drop)))
print(cols_to_drop)

# Optional: Preview remaining data
df.head()
```

Listing 4: Selecting relevant columns and dropping others

5.5 EDA Profiling Using `ydata_profiling`

To gain an overview of the cleaned dataset, an interactive exploratory data analysis (EDA) report was generated using the `ydata_profiling` library (formerly `pandas_profiling`).

```
from ydata_profiling import ProfileReport

# Generate new profile on cleaned dataset
profile = ProfileReport(df, title="Cleaned Loan Dataset EDA",
    explorative=True)

# Show inside notebook
profile.to_notebook_iframe()
```

Listing 5: Generating an automated EDA report with ydata_profiling

5.6 Data Transformation and Cleanup

Before model training, the dataset required several transformations to ensure compatibility with machine learning workflows. This included converting percentage strings, imputing missing values, simplifying target labels, and confirming class distribution.

```
# Step 1: Convert Percentage Strings to Float
df['int_rate'] = df['int_rate'].str.rstrip('%').astype(float)
df['revol_util'] = df['revol_util'].str.rstrip('%').astype(float)
df['term'] = df['term'].str.extract('(\d+)').astype(int)

# Step 2: Impute Missing Values (Retain All Rows)
df['dti'] = df['dti'].fillna(df['dti'].median())
df['revol_util'] = df['revol_util'].fillna(df['revol_util'].median())

# Step 3: Group "Restructure - ..." Classes Together
df['recovery_action'] = df['recovery_action'].replace({
    'Restructure - Moratorium': 'Restructure',
    'Restructure - EMI Reduction': 'Restructure',
    'Restructure - Term Extension': 'Restructure'
})

# Step 4: Confirm Clean Class Distribution
print("Recovery Action Class Distribution:")
print(df['recovery_action'].value_counts())
df.head()
```

Listing 6: Converting percentages, imputing missing values, and grouping labels

5.7 Logistic Regression with SMOTE and Feature Scaling

This experiment uses Logistic Regression with SMOTE to address class imbalance. The pipeline includes target encoding, feature scaling, oversampling, model training, and evaluation using accuracy, balanced accuracy, and ROC AUC.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score,
    balanced_accuracy_score, confusion_matrix, roc_auc_score

# Step 1: Encode Target Labels (recovery_action)
le = LabelEncoder()
df['recovery_encoded'] = le.fit_transform(df['recovery_action'])

# Step 2: Feature/Target Split
X = df.drop(['recovery_action', 'recovery_encoded'], axis=1)
y = df['recovery_encoded']
X = pd.get_dummies(X, drop_first=True)

# Step 3: Scale Numerical Features
numerical_cols = ['loan_amnt', 'term', 'int_rate', 'installment', '
    annual_inc',
                  'dti', 'open_acc', 'total_acc', 'revol_util', '
                     delinq_2yrs', 'pub_rec']
scaler = StandardScaler()
X[numerical_cols] = scaler.fit_transform(X[numerical_cols])

# Step 4: Stratified Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42)

# Step 5: Apply SMOTE to Balance Classes
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
    y_train)

# Step 6: Train Logistic Regression Model
model_scaled = LogisticRegression(max_iter=1000)
model_scaled.fit(X_train_resampled, y_train_resampled)

# Step 7: Evaluate Model
y_pred_scaled = model_scaled.predict(X_test)
y_proba_scaled = model_scaled.predict_proba(X_test)

print("Classification Report: LR + SMOTE\n")
print(classification_report(y_test, y_pred_scaled, target_names=le.
    classes_))
print("Accuracy:", accuracy_score(y_test, y_pred_scaled))
print("Balanced Accuracy:", balanced_accuracy_score(y_test,
    y_pred_scaled))
print("ROC AUC (Macro Average):", roc_auc_score(y_test, y_proba_scaled,
    multi_class='ovr', average='macro'))
```

Listing 7: Logistic Regression + SMOTE + Scaling

5.8 Best Performing Model: XGBoost with Class Weighting and SHAP Explainability

The best performing model was XGBoost trained using class weighting to handle class imbalance. This approach was selected based on consistent performance across key evaluation metrics such as accuracy, balanced accuracy, and ROC AUC. The following subsections illustrate the training, evaluation, and explainability of this model.

5.8.1 XGBoost Training and Evaluation

The XGBoost model was trained on the original imbalanced dataset using class weighting. This approach eliminated the need for resampling while improving prediction fairness across underrepresented classes. The following code demonstrates the training process and evaluation using accuracy, balanced accuracy, and macro-averaged ROC AUC.

```
from xgboost import XGBClassifier
import numpy as np
from sklearn.utils.class_weight import compute_class_weight

# Calculate class weights manually
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train),
    y=y_train
)
weights_dict = dict(zip(np.unique(y_train), class_weights))
sample_weights = y_train.map(weights_dict)

# Initialize and train model
xgb_model = XGBClassifier(
    objective='multi:softprob',
    eval_metric='mlogloss',
    use_label_encoder=False,
    random_state=42)
xgb_model.fit(X_train, y_train, sample_weight=sample_weights)

# Evaluate model
from sklearn.metrics import classification_report, accuracy_score,
   balanced_accuracy_score, roc_auc_score
from sklearn.preprocessing import label_binarize
y_pred_xgb = xgb_model.predict(X_test)
y_pred_proba = xgb_model.predict_proba(X_test)
print("Classification Report (XGBoost with Class Weight):")
print(classification_report(y_test, y_pred_xgb, target_names=le.
   classes_))
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("Balanced Accuracy:", balanced_accuracy_score(y_test, y_pred_xgb)
   )

# Multiclass ROC AUC
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
roc_auc = roc_auc_score(y_test_bin, y_pred_proba, average='macro',
   multi_class='ovr')
print("ROC AUC Score (Macro):", roc_auc)
```

Listing 8: Training and evaluating XGBoost with class weighting

5.8.2 SHAP Explainability and Feature Insights

SHAP (SHapley Additive exPlanations) was used to interpret the XGBoost model. The first part visualizes global feature impact using a summary plot. The second part computes mean absolute SHAP values per class to reveal the most influential features driving classification decisions across different recovery actions.

```
import shap
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# SHAP summary plot
explainer = shap.TreeExplainer(xgb_model)
X_test_sample = X_test.sample(100, random_state=42)
shap_values = explainer.shap_values(X_test_sample)
shap.summary_plot(shap_values, X_test_sample, feature_names=
    X_test_sample.columns)

# Grouped SHAP bar plot
feature_names = X_test_sample.columns.tolist()
shap_vals_array = np.array(shap_values).transpose(2, 0, 1)  # shape:
    (7, 100, 27)

class_map = {
    0: "Legal Action", 1: "Legal Notice", 2: "One-Time Settlement Offer
        ",
    3: "Outsource to Collection Agency", 4: "Reminder",
    5: "Restructure", 6: "Settled - No Further Action"
}

mean_shap_by_class = {}
for class_id in range(shap_vals_array.shape[0]):
    class_name = class_map[class_id]
    mean_abs_shap = np.abs(shap_vals_array[class_id]).mean(axis=0)
    mean_shap_by_class[class_name] = mean_abs_shap

df_plot = pd.DataFrame(mean_shap_by_class, index=feature_names).T
top_features = df_plot.mean().sort_values(ascending=False).head(10).
    index
df_top = df_plot[top_features]

plt.figure(figsize=(12, 6))
df_top.plot(kind='bar', ax=plt.gca())
plt.title("Mean |SHAP| Feature Importance Across Classes (XGBoost + CB)
    ")
plt.xlabel("Class")
plt.ylabel("Mean |SHAP| Value")
plt.xticks(rotation=45, ha='right')
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5), title='Top
    Features')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Listing 9: SHAP explainability: summary plot and grouped feature importance

6 Model Training and Evaluation

6.1 Model Setup and Data Split

The final dataset included 135,883 records. It was split into training and test sets using an 80:20 ratio, with stratification to maintain class distribution. To handle the class imbalance in recovery action labels, three techniques were used:

- SMOTE - synthetic oversampling of minority classes
- Class weighting - assigning higher importance to underrepresented classes
- Random undersampling - reducing samples from majority classes

6.2 Models and Experiments

Three machine learning models were implemented using the scikit-learn and xgboost libraries:

- Logistic Regression
- Random Forest
- XGBoost

Each model was trained with all three balancing methods, resulting in a total of nine experiments. The table below shows the test set performance of each model:

Table 3: Performance Metrics of All Models on the Test Set

Model	**Technique**	**Accuracy**	**Balanced Accuracy**	**Macro ROC AUC**	**Macro F1-score**
Logistic Reg.	SMOTE	0.9585	0.9634	0.9953	0.91
Logistic Reg.	Class Balanced	0.9450	0.9653	0.9962	0.88
Logistic Reg.	Undersampled	0.8289	0.8843	0.9826	0.69
Random Forest	SMOTE	0.9722	0.9638	0.9984	0.90
Random Forest	Class Balanced	0.9993	0.9835	0.9999	0.99
Random Forest	Undersampled	0.9722	0.9638	0.9984	0.90
XGBoost	SMOTE	0.9768	0.9832	0.9989	0.89
XGBoost	Class Balanced	0.9986	0.9927	0.9999	1.00
XGBoost	Undersampled	0.9768	0.9832	0.9989	0.89

The best-performing model was XGBoost with class weighting (Model 3B), which outperformed all others across all evaluation metrics.

6.3 Detailed Results for Best Model (3B)

To understand how the best model performed in detail, the following outputs were captured from Jupyter Notebook:

- Classification report showing precision, recall, and F1-score for each recovery action
- Confusion matrix to visualize misclassifications
- ROC curve showing AUC scores for all classes
- SHAP beeswarm plot showing global feature influence
- SHAP bar plot showing top features per class

	precision	recall	f1-score	support
Legal Action	1.00	1.00	1.00	13859
Legal Notice	1.00	1.00	1.00	6185
One-Time Settlement Offer	0.99	1.00	1.00	1538
Outsource to Collection Agency	1.00	0.95	0.98	21
Reminder	1.00	1.00	1.00	2760
Restructure	1.00	1.00	1.00	213
Settled – No Further Action	1.00	1.00	1.00	2601
accuracy			1.00	27177
macro avg	1.00	0.99	1.00	27177
weighted avg	1.00	1.00	1.00	27177

✅ Accuracy: 0.9986385546601906
✅ Balanced Accuracy: 0.9927110894976348
✅ Multiclass ROC AUC Score (macro average): 0.9999644046161544

Figure 1: Classification Report for Model 3B (XGBoost with Class Weighting)

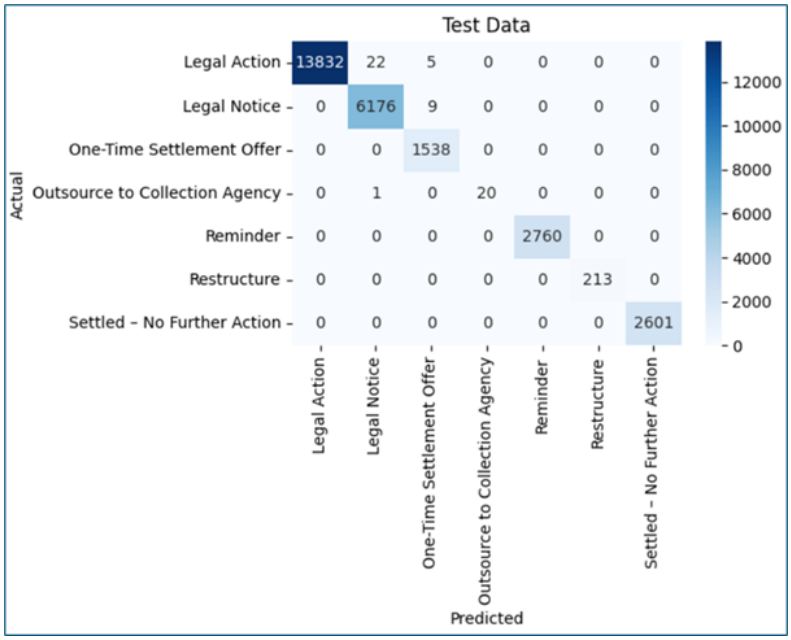

Figure 2: Confusion Matrix for Model 3B (XGBoost with Class Weighting)

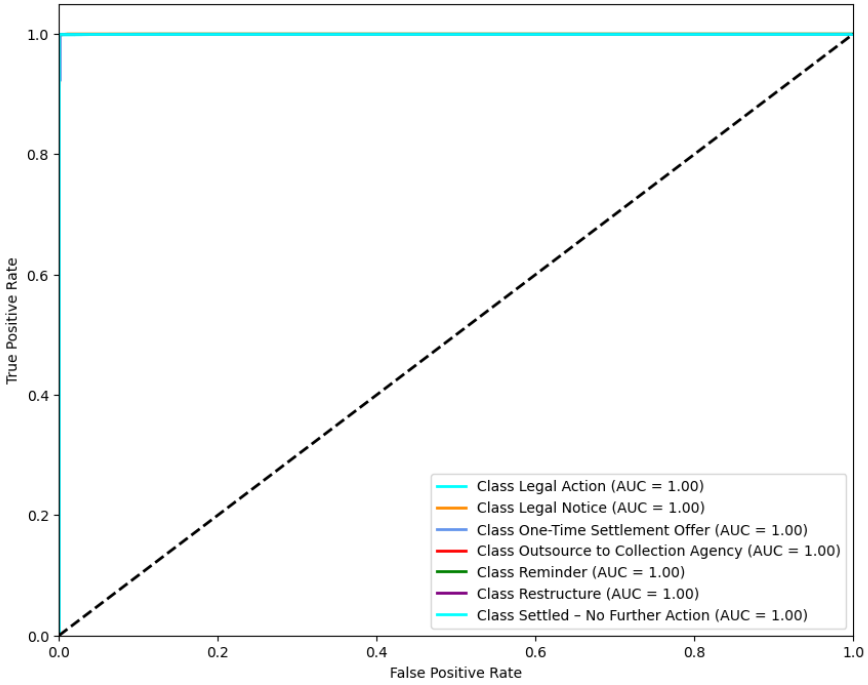

Figure 3: ROC Curve for Model 3B

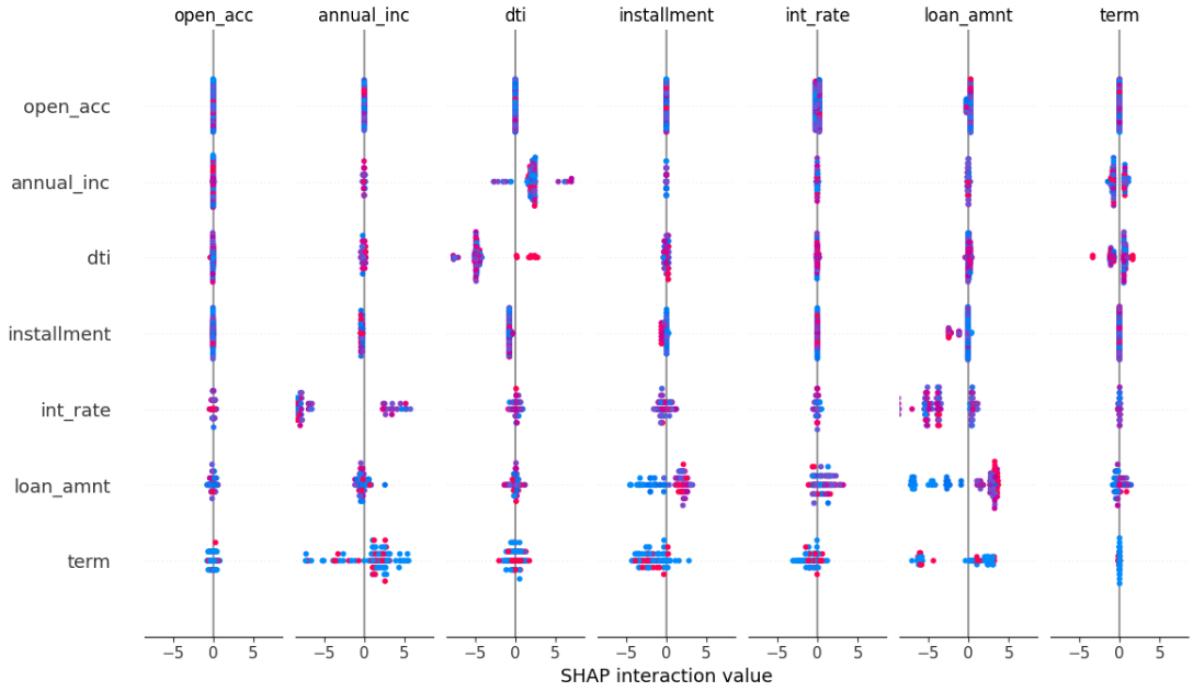

Figure 4: SHAP Beeswarm Plot for Model 3B

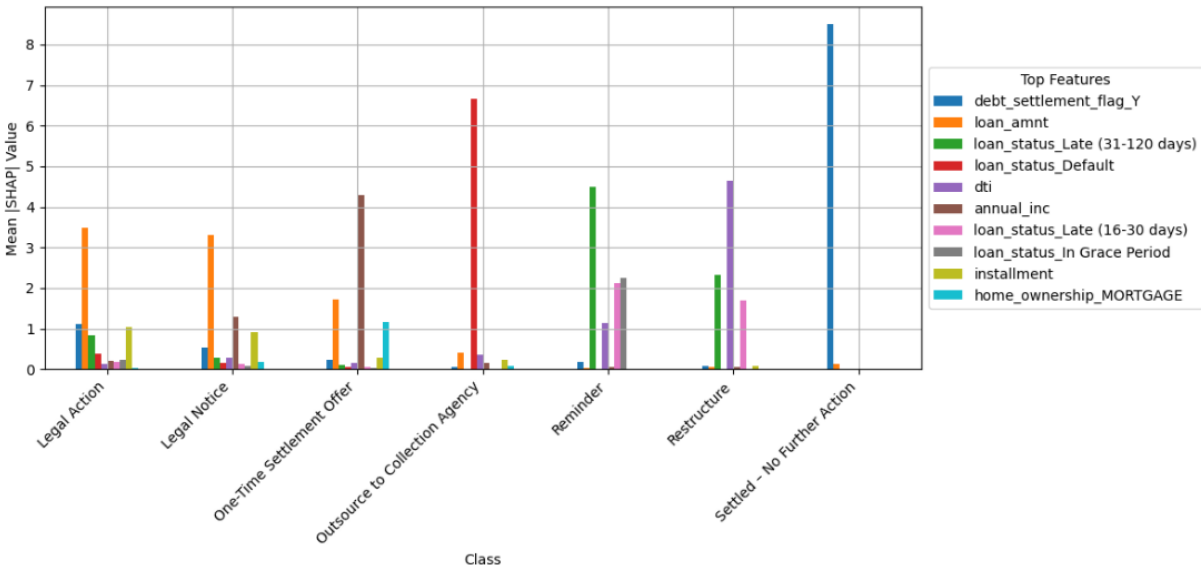

Figure 5: SHAP Bar Plot Showing Top Features per Recovery Class for Model 3B

6.4 Train-Test Performance Comparison

To ensure that models generalized well and did not overfit, training and test accuracy were compared. The table below shows the accuracy gap across all nine experiments:

Table 4: Train vs Test Accuracy and Gap for All Models

Model	**Technique**	**Train Acc**	**Test Acc**	**Accuracy Gap**
Logistic Reg.	SMOTE	0.9725	0.9585	0.0140
Logistic Reg.	Class Balanced	0.9457	0.9450	0.0007
Logistic Reg.	Undersampled	0.9260	0.8289	0.0971
Random Forest	SMOTE	1.0000	0.9722	0.0278
Random Forest	Class Balanced	1.0000	0.9993	0.0007
Random Forest	Undersampled	1.0000	0.9722	0.0278
XGBoost	SMOTE	1.0000	0.9768	0.0232
XGBoost	Class Balanced	0.9994	0.9986	0.0008
XGBoost	Undersampled	1.0000	0.9768	0.0232

There was no significant gap between training and test accuracy for any of the models, indicating good generalization and no signs of overfitting, especially in the case of Model 3B.

7 Reproducibility Guidelines

This section provides a brief guide on how to reproduce the modeling and evaluation steps carried out in this project.

7.1 Instructions for Replication

To replicate the results shown in this manual:

1. Set up a Python 3.10 environment using Anaconda.
2. Install the required libraries:
    - `pandas`, `numpy`, `scikit-learn`, `xgboost`
    - `imbalanced-learn`, `matplotlib`, `seaborn`
    - `shap`, `ydata-profiling`
3. Open the Jupyter Notebook `lending_club.ipynb`.
4. Run the cells in sequence to perform preprocessing, model training, evaluation, and SHAP-based explanation.

All visual outputs and metrics can be regenerated by executing the notebook using the provided filtered dataset.

A Appendix A: Dropped and Retained Columns

Table 5: Final List of Columns Retained and Dropped with Justification

Sl No	Column Name	Status	Reason
1	id	Dropped	Unique identifier - irrelevant for ML model training
2	loan_amnt	Retained	Loan amount approved - key financial exposure indicator
3	funded_amnt	Dropped	Redundant with 'loan_amnt' - same approved loan amount
4	funded_amnt_inv	Dropped	Redundant with 'loan_amnt' - investor-funded portion of loan
5	term	Retained	Loan term - impacts affordability and repayment structure
6	int_rate	Retained	Interest rate - proxy for borrower creditworthiness
7	installment	Retained	Installment amount - reflects monthly repayment stress
8	grade	Retained	Platform-assigned grade - categorizes credit risk
9	sub_grade	Dropped	Redundant with 'grade' - finer credit risk tier
10	emp_title	Dropped	High cardinality - unstructured free-text field
11	emp_length	Dropped	Low predictive value - removed for simplification
12	home_ownership	Retained	Home ownership status - indicates financial stability
13	annual_inc	Retained	Declared annual income - input for affordability/risk
14	verification_status	Dropped	Low predictive value - removed for simplification
15	issue_d	Dropped	Low predictive value - removed for simplification
16	loan_status	Retained	Defines the loan's repayment stage - primary trigger for recovery action
17	pymnt_plan	Dropped	Post-loan outcome - potential data leakage
18	url	Dropped	High cardinality - unstructured free-text field
19	purpose	Dropped	Low predictive value - removed for simplification
20	title	Dropped	High cardinality - unstructured free-text field
21	zip_code	Dropped	High cardinality - unstructured free-text field

Continued on next page

Sl No	Column Name	Status	Reason
22	addr_state	Dropped	Low predictive value - removed for simplification
23	dti	Retained	Debt-to-income ratio - crucial risk assessment metric
24	delinq_2yrs	Retained	Number of delinquencies in past 2 years - historical risk behavior
25	earliest_cr_line	Dropped	Low predictive value - removed for simplification
26	fico_range_low	Dropped	Correlated with retained variables - limited added value
27	fico_range_high	Dropped	Correlated with retained variables - limited added value
28	inq_last_6mths	Dropped	Correlated with retained variables - limited added value
29	mths_since_last_delinq	Dropped	Low predictive value - removed for simplification
30	mths_since_last_record	Dropped	Low predictive value - removed for simplification
31	open_acc	Retained	Number of open credit lines - shows active credit exposure
32	pub_rec	Retained	Number of derogatory public records - risk signal
33	revol_bal	Dropped	Low predictive value - removed for simplification
34	revol_util	Retained	Revolving credit utilization - behavioral financial indicator
35	total_acc	Retained	Total number of credit accounts - credit experience metric
36	initial_list_status	Dropped	Low predictive value - removed for simplification
37	out_prncp	Dropped	Low predictive value - removed for simplification
38	out_prncp_inv	Dropped	Low predictive value - removed for simplification
39	total_pymnt	Dropped	Post-loan outcome - potential data leakage
40	total_pymnt_inv	Dropped	Post-loan outcome - potential data leakage
41	total_rec_prncp	Dropped	Low predictive value - removed for simplification
42	total_rec_int	Dropped	Low predictive value - removed for simplification

Continued on next page

Sl No	Column Name	Status	Reason
43	total_rec_late_fee	Dropped	Low predictive value - removed for simplification
44	recoveries	Dropped	Post-loan outcome - potential data leakage
45	collection_recovery_fee	Dropped	Low predictive value - removed for simplification
46	last_pymnt_d	Dropped	Post-loan outcome - potential data leakage
47	last_pymnt_amnt	Dropped	Post-loan outcome - potential data leakage
48	next_pymnt_d	Dropped	Post-loan outcome - potential data leakage
49	last_credit_pull_d	Dropped	Post-loan outcome - potential data leakage
50	last_fico_range_high	Dropped	Correlated with retained variables - limited added value
51	last_fico_range_low	Dropped	Correlated with retained variables - limited added value
52	collections_12_mths_ex_med	Dropped	Low predictive value - removed for simplification
53	mths_since_last_major_derog	Dropped	Low predictive value - removed for simplification
54	policy_code	Dropped	Low predictive value - removed for simplification
55	application_type	Dropped	Low predictive value - removed for simplification
56	annual_inc_joint	Dropped	Dropped due to >80% missing values
57	dti_joint	Dropped	Dropped due to >80% missing values
58	verification_status_joint	Dropped	Low predictive value - removed for simplification
59	acc_now_delinq	Dropped	Low predictive value - removed for simplification
60	tot_coll_amt	Dropped	Low predictive value - removed for simplification
61	tot_cur_bal	Dropped	Low predictive value - removed for simplification
62	open_acc_6m	Dropped	Low predictive value - removed for simplification
63	open_act_il	Dropped	Low predictive value - removed for simplification
64	open_il_12m	Dropped	Low predictive value - removed for simplification
65	open_il_24m	Dropped	Low predictive value - removed for simplification

Continued on next page

Sl No	Column Name	Status	Reason
66	mths_since_rcnt_il	Dropped	Low predictive value - removed for simplification
67	total_bal_il	Dropped	Low predictive value - removed for simplification
68	il_util	Dropped	Low predictive value - removed for simplification
69	open_rv_12m	Dropped	Low predictive value - removed for simplification
70	open_rv_24m	Dropped	Low predictive value - removed for simplification
71	max_bal_bc	Dropped	Low predictive value - removed for simplification
72	all_util	Dropped	Low predictive value - removed for simplification
73	total_rev_hi_lim	Dropped	Low predictive value - removed for simplification
74	inq_fi	Dropped	Correlated with retained variables - limited added value
75	total_cu_tl	Dropped	Low predictive value - removed for simplification
76	inq_last_12m	Dropped	Correlated with retained variables - limited added value
77	acc_open_past_24mths	Dropped	Low predictive value - removed for simplification
78	avg_cur_bal	Dropped	Low predictive value - removed for simplification
79	bc_open_to_buy	Dropped	Low predictive value - removed for simplification
80	bc_util	Dropped	Low predictive value - removed for simplification
81	chargeoff_within_12_mths	Dropped	Low predictive value - removed for simplification
82	delinq_amnt	Dropped	Correlated with retained variables - limited added value
83	mo_sin_old_il_acct	Dropped	Low predictive value - removed for simplification
84	mo_sin_old_rev_tl_op	Dropped	Low predictive value - removed for simplification
85	mo_sin_rcnt_rev_tl_op	Dropped	Low predictive value - removed for simplification

Continued on next page

Sl No	Column Name	Status	Reason
86	mo_sin_rcnt_tl	Dropped	Low predictive value - removed for simplification
87	mort_acc	Dropped	Low predictive value - removed for simplification
88	mths_since_recent_bc	Dropped	Low predictive value - removed for simplification
89	mths_since_recent_bc_dlq	Dropped	Low predictive value - removed for simplification
90	mths_since_recent_inq	Dropped	Low predictive value - removed for simplification
91	mths_since_recent_revol_delinq	Dropped	Low predictive value - removed for simplification
92	num_accts_ever_120_pd	Dropped	Low predictive value - removed for simplification
93	num_actv_bc_tl	Dropped	Low predictive value - removed for simplification
94	num_actv_rev_tl	Dropped	Low predictive value - removed for simplification
95	num_bc_sats	Dropped	Low predictive value - removed for simplification
96	num_bc_tl	Dropped	Low predictive value - removed for simplification
97	num_il_tl	Dropped	Low predictive value - removed for simplification
98	num_op_rev_tl	Dropped	Low predictive value - removed for simplification
99	num_rev_accts	Dropped	Low predictive value - removed for simplification
100	num_rev_tl_bal_gt_0	Dropped	Low predictive value - removed for simplification
101	num_sats	Dropped	Low predictive value - removed for simplification
102	num_tl_120dpd_2m	Dropped	Low predictive value - removed for simplification
103	num_tl_30dpd	Dropped	Low predictive value - removed for simplification
104	num_tl_90g_dpd_24m	Dropped	Low predictive value - removed for simplification
105	num_tl_op_past_12m	Dropped	Low predictive value - removed for simplification

Continued on next page

Sl No	Column Name	Status	Reason
106	pct_tl_nvr_dlq	Dropped	Low predictive value - removed for simplification
107	percent_bc_gt_75	Dropped	Low predictive value - removed for simplification
108	pub_rec_bankruptcies	Dropped	Redundant with 'pub_rec' - captures overlapping risk signal
109	tax_liens	Dropped	Dropped due to >80% missing values
110	tot_hi_cred_lim	Dropped	Low predictive value - removed for simplification
111	total_bal_ex_mort	Dropped	Low predictive value - removed for simplification
112	total_bc_limit	Dropped	Low predictive value - removed for simplification
113	total_il_high_credit_limit	Dropped	Low predictive value - removed for simplification
114	revol_bal_joint	Dropped	Dropped due to >80% missing values
115	sec_app_fico_range_low	Dropped	Dropped due to >80% missing values
116	sec_app_fico_range_high	Dropped	Dropped due to >80% missing values
117	sec_app_earliest_cr_line	Dropped	Dropped due to >80% missing values
118	sec_app_inq_last_6mths	Dropped	Dropped due to >80% missing values
119	sec_app_mort_acc	Dropped	Dropped due to >80% missing values
120	sec_app_open_acc	Dropped	Dropped due to >80% missing values
121	sec_app_revol_util	Dropped	Dropped due to >80% missing values
122	sec_app_open_act_il	Dropped	Dropped due to >80% missing values
123	sec_app_num_rev_accts	Dropped	Dropped due to >80% missing values
124	sec_app_chargeoff_within_12_mths	Dropped	Dropped due to >80% missing values
125	sec_app_collections_12_mths_ex_med	Dropped	Dropped due to >80% missing values
126	hardship_flag	Dropped	Dropped due to >80% missing values
127	hardship_type	Dropped	Dropped due to >80% missing values
128	hardship_reason	Dropped	Dropped due to >80% missing values
129	hardship_status	Dropped	Dropped due to >80% missing values
130	deferral_term	Dropped	Dropped due to >80% missing values
131	hardship_amount	Dropped	Dropped due to >80% missing values
132	hardship_start_date	Dropped	Dropped due to >80% missing values
133	hardship_end_date	Dropped	Dropped due to >80% missing values
134	payment_plan_start_date	Dropped	Dropped due to >80% missing values
135	hardship_length	Dropped	Dropped due to >80% missing values

Continued on next page

Sl No	Column Name	Status	Reason
136	hardship_dpd	Dropped	Dropped due to >80% missing values
137	hardship_loan_status	Dropped	Dropped due to >80% missing values
138	orig_projected_additional_accrued_interest	Dropped	Dropped due to >80% missing values
139	hardship_payoff_balance_amount	Dropped	Dropped due to >80% missing values
140	hardship_last_payment_amount	Dropped	Dropped due to >80% missing values
141	debt_settlement_flag	Retained	Indicates settled accounts - overrides all other recovery conditions
142	recovery_action	Retained	Target variable - assigned using rule-based logic