Loan Default Prediction Using Machine Learning Techniques

Report

Ayush Kundu

M.Sc. Statistics
Department of Mathematics & Statistics
IIT Kanpur

Contents

1 Introduction

In today's financial landscape, effective credit risk assessment is critical for lending institutions to maintain profitability and minimize losses. Every year, a significant proportion of borrowers default on their loans, causing substantial financial strain on banks and peer-to-peer lending platforms. Traditional credit scoring methods, often based on rule-based systems and human judgment, can be slow, inconsistent, and unable to capture complex patterns in high-dimensional data.

Machine learning offers a systematic, data-driven alternative for predicting loan repayment behavior by learning from historical records of borrower profiles, loan characteristics, and repayment outcomes. By leveraging algorithms capable of handling large datasets, uncovering nonlinear relationships, and adjusting for imbalanced classes, lenders can automate decision making, improve risk stratification, and reduce default rates.

In this project, we build and evaluate several supervised learning models to predict whether a borrower will default on their loan. Using real-world data from LendingClub, we perform rigorous data preprocessing, exploratory analysis, and feature engineering before training baseline models (Logistic Regression, Decision Tree) and advanced ensemble methods (Random Forest, XGBoost). To address the inherent class imbalance (only 13% of loans result in default), we apply the Synthetic Minority Over-sampling Technique (SMOTE). Model performance is compared using precision, recall, F1-score, and ROC-AUC, with visualizations provided to illustrate key insights and feature importances.

This report is structured as follows:

- Section 2 defines the precise objective of the project.
- Section 3 describes the datasets and their key characteristics.
- Section 4 details the data cleaning and preprocessing steps.
- Section 5 presents exploratory data analysis findings.
- Section 6 covers feature engineering and encoding.
- Section 7 outlines model training and evaluation methodologies.
- Section 8 discusses the handling of class imbalance with SMOTE.
- Section 9 explores advanced models—Random Forest and XGBoost.
- Section 10 compares all model results and interprets business implications.
- Section 11 concludes with key takeaways and suggestions for future work.

2 Objective

The primary objective of this project is to develop and evaluate supervised machine learning models that can accurately predict whether a borrower will default on a loan. Specifically, we aim to:

- Transform raw LendingClub loan data into a clean, structured dataset suitable for modeling.
- Design a binary classification task where the target variable is `loan_default` (0 = Fully Paid, 1 = Default).
- Address class imbalance (approx. 13% default rate) using techniques such as SMOTE and model class weighting.
- Train and compare multiple algorithms—including Logistic Regression, Decision Tree, Random Forest, and XGBoost—based on precision, recall, F1-score, and ROC-AUC.
- Identify the best model for real-world credit risk assessment and derive actionable insights for lenders.

3 Dataset Description

The dataset used in this project originates from the public LendingClub loan data portal. It consists of two CSV files:

- **`accepted_loans.csv`**: records of loans that were approved and issued, along with their final repayment status.
- **`rejected_loans.csv`** *(not used for modeling)*: applications that were declined, with no outcome labels.

For the purposes of supervised classification, we use only the `accepted_loans.csv` dataset, which contains *500 000+* loan records spanning from 2007 to 2018. After initial filtering and cleaning (removal of columns with excessive missing values and data-leakage fields), the final dataset comprises approximately **452 141** observations and **15** key features. The target variable is

$$\texttt{loan\_default} = \begin{cases} 0 & \text{if the loan was fully paid,} \\ 1 & \text{if the loan was charged off or became delinquent.} \end{cases}$$

The overall default rate in the cleaned data is $\approx 13.06\%$.

Key Features: The following subset of features was selected for modeling:

Feature	Description
loan_amnt	Original amount requested by the borrower (USD)
term	Term of the loan (e.g., "36 months", "60 months")
int_rate	Interest rate on the loan (percentage)
installment	Monthly payment owed by the borrower (USD)
grade	LendingClub risk grade assigned at issue (A through G)
emp_length	Employment length in years (e.g., "< 1 year", "10+ years")
home_ownership	Home ownership status (e.g., RENT, OWN, MORTGAGE)
annual_inc	Self-reported annual income (USD)
purpose	Loan purpose (e.g., debt consolidation, credit card, medical)
dti	Debt-to-income ratio (percentage)
delinq_2yrs	Number of delinquent credit lines in the past 2 years
revol_util	Revolving line utilization rate (percentage)
total_acc	Total number of credit lines currently open

Table 1: Selected features for modeling

Additional raw columns (such as borrower address, URL, and payment plan indicators) were removed due to either high missingness or data-leakage risk. Remaining missing values in numeric fields were imputed with the median, and categorical fields with the mode.

With this curated dataset, we ensure that all predictors are available at loan application time, enabling realistic and reproducible credit risk modeling.

4 Data Preprocessing

Before training any models, the raw data underwent a series of preprocessing steps to ensure quality, remove leakage, and prepare features for machine learning:

- **Loading and Initial Inspection:** The file `accepted_loans.csv` was read into a DataFrame and inspected for shape, data types, and missing values. The original dataset contained over 480 000 rows and 100+ columns.
- **Binary Target Creation:**
  - The original `loan_status` column had multiple text values (e.g., *Fully Paid*, *Charged Off*, *Late (16–30 days)*, etc.).
  - We defined a new binary column `loan_default` where

$$\texttt{loan\_default} = \begin{cases} 1, & \text{if status} \in \{\text{Charged Off}, \dots\}, \\ 0, & \text{if status} = \text{Fully Paid}. \end{cases}$$

  - This yielded a default rate of approximately 13.06%.
- **Removing Columns with Excessive Missingness:** Any column with more than 40% missing values was dropped, reducing the feature set by roughly 30–40 fields.
- **Dropping Irrelevant or Leaky Features:** Columns not available at application time or posing leakage risk were removed, including:

  `loan_status`, `issue_d`, `id`, `member_id`, `url`, `zip_code`, `emp_title`, `title`, `pymnt_plan`.
- **Imputing Remaining Missing Values:**
  - *Numerical* columns: missing entries replaced by the column median.
  - *Categorical* columns: missing entries replaced by the column mode.
- **Encoding Categorical Variables:** We applied one-hot encoding (via `pd.get_dummies(..., drop_first=True)`) to transform categorical features (e.g., `grade`, `home_ownership`, `purpose`, `term`, `emp_length`) into binary indicator columns, avoiding multicollinearity.
- **Train–Test Split:** The cleaned dataset was split into training (80%) and testing (20%) sets using stratified sampling on `loan_default` to preserve class proportions:

$$(X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}) = \text{train\_test\_split}(X, y,\ \text{test\_size} = 0.2,\ \text{stratify} = y,\ \text{random\_state} = 42).$$

- **Feature Scaling:** For distance- and gradient-based models (e.g., Logistic Regression, SVM), we standardized numeric features to zero mean and unit variance using `StandardScaler` fitted on the training set and applied to both train and test splits.

After these preprocessing steps, the dataset comprised approximately 452 141 observations and 60+ engineered features, ready for model training and evaluation.

5 Exploratory Data Analysis (EDA)

Before building models, we conducted exploratory data analysis to uncover patterns and relationships in the data that might inform feature selection and modeling decisions.

5.1 Target Variable Distribution

Figure 1 shows the distribution of the binary target `loan_default`. Approximately 13.06% of loans resulted in default, indicating a significant class imbalance.

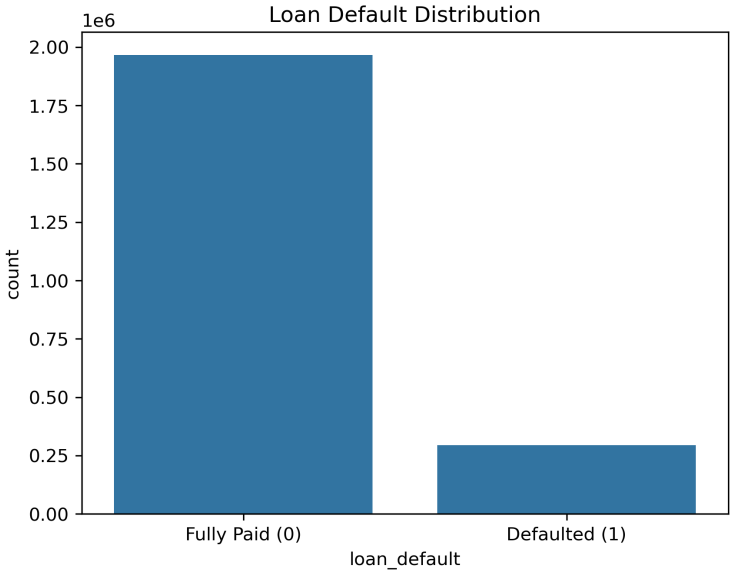

Figure 1: Distribution of Loan Defaults vs. Fully Paid Loans

5.2 Univariate Analysis

We examined key numerical features to understand their individual distributions and potential differences between defaulters and non-defaulters.

- **Loan Amount (`loan_amnt`):** Defaulters tend to request slightly higher loan amounts on average. (See Figure 2.)
- **Interest Rate (`int_rate`):** The interest rate is positively skewed for defaulters, indicating high-interest loans carry more risk. (See Figure 2.)
- **Credit Grade (`grade`):** Lower grades (e.g., F, G) show a higher proportion of defaults compared to higher grades (A, B). (See Figure 3.)

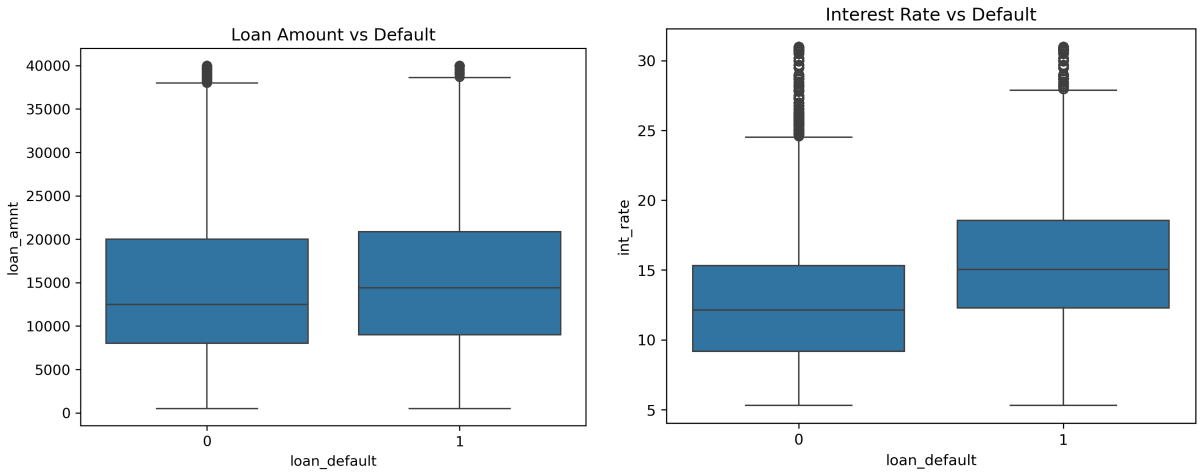

Figure 2: Boxplots of Loan Amount and Interest Rate by Default Status

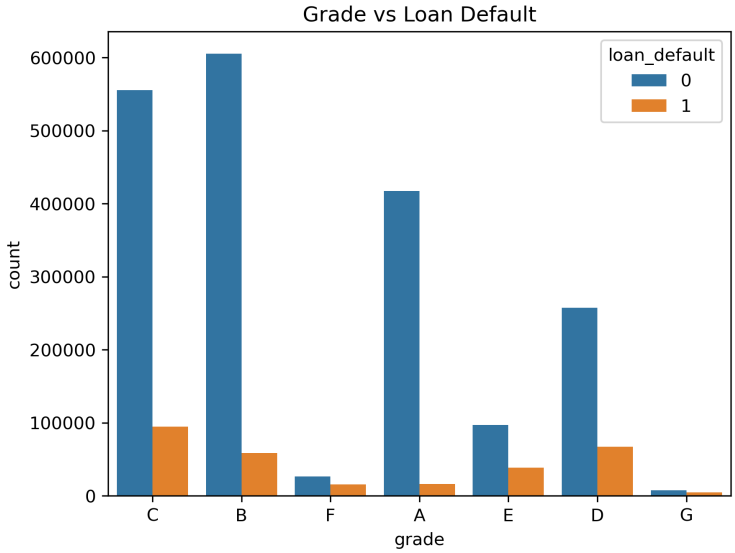

Figure 3: Count of Defaults and Fully Paid Loans by Credit Grade

5.3 Bivariate Analysis of Categorical Features

We compared categorical variables against the default target to spot risk factors:

- **Home Ownership (`home_ownership`):** Renters and those with mortgages show higher default rates than homeowners. (Figure 4.)
- **Loan Purpose (`purpose`):** Loans for debt consolidation and credit card refinancing carry different default profiles. (Figure 4.)

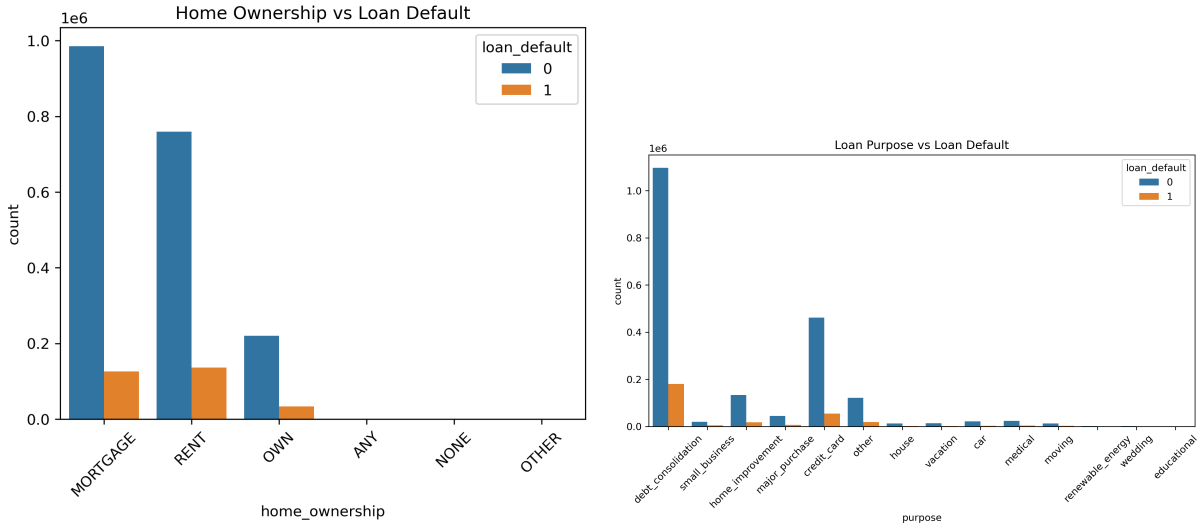

Figure 4: Home Ownership and Loan Purpose vs. Default Status

5.4 Correlation Analysis

We computed the Pearson correlation matrix for numerical variables including `loan_default` (Figure 5). Features with strong positive correlation to default include `int_rate` and `dti`, while `annual_inc` and `total_acc` show negative correlation.

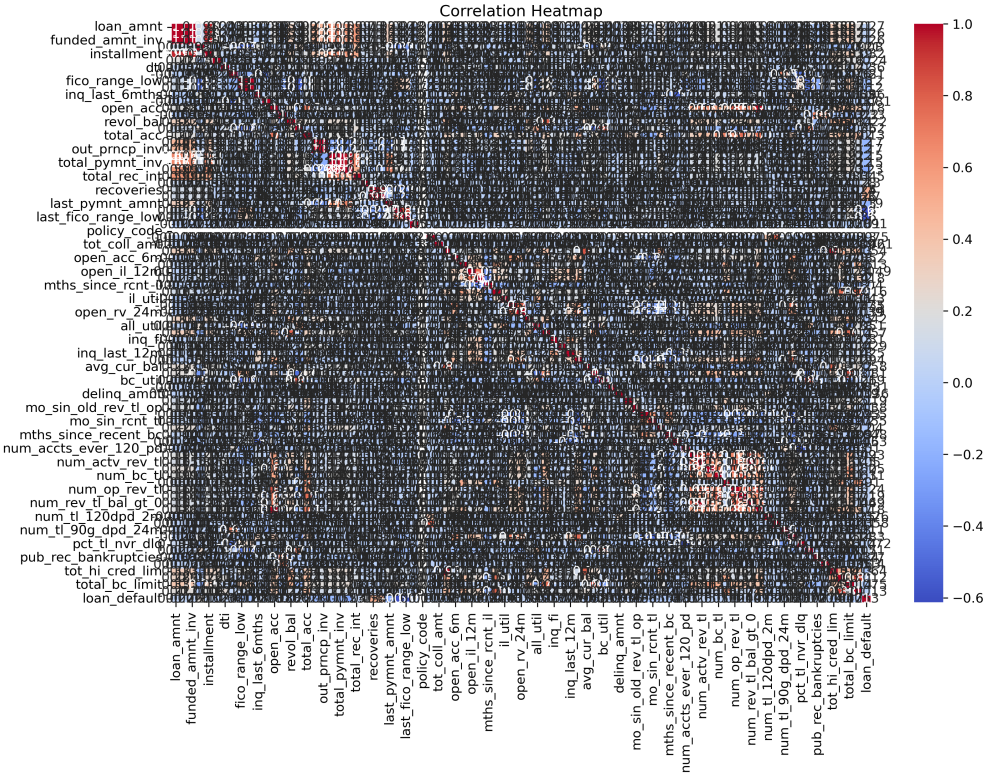

Figure 5: Correlation Heatmap of Numerical Features and Target

These insights guided feature selection and preprocessing choices for subsequent modeling.

6 Feature Engineering and Encoding

To prepare the cleaned dataset for machine learning, we performed the following feature engineering and encoding steps:

6.1 Feature Selection

Based on domain knowledge and correlation analysis, we selected a subset of 13 raw and engineered variables for modeling:

`loan_amnt`, `term`, `int_rate`, `installment`, `grade`, `emp_length`, `home_ownership`, `annual_inc`, `purpose`, `dti`, `delinq_2yrs`, `revol_util`, `total_acc`.

These were chosen for their predictive power and availability at application time.

6.2 Numeric Transformations

- `term`: converted from strings "36 months" / "60 months" to integers 36 and 60.
- `int_rate`, `revol_util`: originally stored as percentages with `%` symbol; stripped and converted to floats in [0,100].
- `emp_length`: mapped employment-length categories (e.g. "$<$ 1 year", "10+ years") to numeric values 0,1,...,10.

6.3 One-Hot Encoding of Categorical Variables

We applied one-hot encoding to the following categorical features, dropping the first category to avoid multicollinearity:

$$\{\,\texttt{grade},\ \texttt{home\_ownership},\ \texttt{purpose},\ \texttt{term},\ \texttt{emp\_length}\,\}.$$

This expanded the feature matrix into binary indicator columns. For example, `grade` A–G became `grade_B`, `grade_C`, ..., `grade_G`.

6.4 Assembling Final Feature Matrix

After encoding, the dataset comprised approximately 60+ numerical features. We defined:

$$X = \{\text{all encoded and numeric features}\}, \quad y = \texttt{loan\_default}.$$

6.5 Train–Test Split

We split into training and test sets with an 80%/20% split, using stratified sampling on the target to preserve the default ratio:

$$(X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}) = \text{train\_test\_split}(X, y;\ \text{test\_size} = 0.2,\ \text{stratify} = y,\ \text{random\_state} = 42).$$

6.6 Feature Scaling

For distance- and gradient-based algorithms (Logistic Regression, SVM, KNN), we standardized all numeric features to zero mean and unit variance using the training set parameters:

$$X_{\text{train}}^{\text{scaled}} = \frac{X_{\text{train}} - \mu}{\sigma}, \quad X_{\text{test}}^{\text{scaled}} = \frac{X_{\text{test}} - \mu}{\sigma}.$$

Here, μ and σ are the feature-wise mean and standard deviation computed on X_{train}.

With these engineered, encoded, and scaled features, the data is now fully machine-readable and ready for model training and evaluation.

7 Model Building and Evaluation

In this section, we describe the supervised learning algorithms used to predict loan default, the training procedure, and the evaluation metrics. We begin with two baseline models—Logistic Regression and Decision Tree—trained on the preprocessed, imbalanced dataset. Results here will motivate the need for more advanced techniques in Section 8.

7.1 Baseline Models:

7.1.1 Logistic Regression:

Logistic Regression is a linear classifier that models the log-odds of the positive class (default) as a linear combination of input features:

$$\log \frac{\Pr(y = 1 \mid \mathbf{x})}{\Pr(y = 0 \mid \mathbf{x})} = \beta_0 + \sum_{i=1}^{p} \beta_i x_i.$$

To account for class imbalance (only 13.06% defaults), we set the `class_weight` parameter to `balanced`, which automatically weights the loss function inversely proportional to class frequencies. The model was trained via maximum likelihood estimation using the `liblinear` solver and a maximum of 1000 iterations.

7.1.2 Decision Tree:

A Decision Tree partitions the feature space into axis-aligned rectangles by recursively selecting features and thresholds that maximize information gain (measured via Gini impurity). We constrained the maximum depth to 6 to reduce overfitting, and similarly used `class_weight='balanced'` to penalize misclassification of the minority class more heavily.

7.2 Evaluation Metrics:

We assess model performance using the following metrics, computed on the held-out test set (20% of data):

- **Precision** for class k: $\frac{\mathrm{TP}_k}{\mathrm{TP}_k+\mathrm{FP}_k}$.
- **Recall** (Sensitivity) for class k: $\frac{\mathrm{TP}_k}{\mathrm{TP}_k+\mathrm{FN}_k}$.
- **F1-score** for class k: harmonic mean of precision and recall, $\frac{2\,\mathrm{Precision}_k\,\mathrm{Recall}_k}{\mathrm{Precision}_k+\mathrm{Recall}_k}$.
- **ROC-AUC**: area under the Receiver Operating Characteristic curve, measuring the trade-off between true positive rate and false positive rate.

Since catching defaulters ($y = 1$) is critical, we pay particular attention to recall and F1-score for class 1.

7.3 Results on Test Set

Table 2 summarizes the performance of the two baseline models before any balancing or resampling.

Model	Prec (1)	Rec (1)	F1 (1)	ROC-AUC
Logistic Regression	0.20	0.68	0.31	0.6952
Decision Tree	0.19	0.77	0.31	0.7031

Table 2: Baseline model performance on the imbalanced test set ($y = 1$ = default).

Both models achieve moderate ROC-AUC (≈ 0.70) but exhibit low precision on the default class, leading to many false positives. The Decision Tree attains higher recall (0.77) than Logistic Regression (0.68) at the expense of similar precision. These results highlight the challenge posed by the 13% default rate and motivate the use of SMOTE and advanced ensemble models in subsequent sections.

8 Handling Imbalanced Data

The original dataset exhibited a significant class imbalance, with only 13.06% of loans resulting in default. Training on such imbalanced data can lead models to favor the majority class (fully paid loans) and perform poorly on the minority class (defaults). To mitigate this, we applied the **Synthetic Minority Over-sampling Technique (SMOTE)** [**?**], which generates new synthetic samples of the minority class by interpolating between existing minority examples.

8.1 SMOTE Algorithm

SMOTE operates in feature space as follows:

1. For each minority sample x_i, identify its k nearest neighbors among other minority samples.
2. For a user-specified amount of oversampling, randomly select one neighbor x_{zi}.
3. Generate a synthetic point x_{syn} along the line segment joining x_i and x_{zi}:

$$x_{\text{syn}} = x_i + \delta\,(x_{zi} - x_i), \quad \delta \sim \mathcal{U}(0,1).$$

4. Repeat until the minority class is balanced with the majority class.

We used the `SMOTE` implementation from `imbalanced_learn` with default $k = 5$ neighbors and random state 42.

8.2 Application to Training Data

SMOTE was applied *only* to the training set to avoid information leakage. After splitting the cleaned and encoded dataset into 80% train and 20% test (see Section 6), we executed:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
```

This increased the minority class samples from 59,035 to 393,106, matching the majority class count.

8.3 Model Retraining and Evaluation

We retrained our baseline models on the SMOTE-resampled training set without changing hyperparameters:

- **Logistic Regression** (`class_weight` reset to default)

- **Decision Tree** (max depth = 6)

Evaluation on the original (imbalanced) test set yielded the results in Table 3.

Model	**Prec (1)**	**Rec (1)**	**F1 (1)**	**ROC-AUC**
Logistic Regression (SMOTE)	0.40	0.03	0.06	0.6629
Decision Tree (SMOTE)	0.21	0.26	0.23	0.6517

Table 3: Performance of baseline models retrained on SMOTE-resampled data ($y = 1 =$ default).

8.4 Discussion

SMOTE reduced class imbalance but also introduced noise by interpolating between minority points. Models must be carefully selected and tuned to leverage these synthetic examples. In this project, tree-based models (Decision Tree, and later Random Forest / XGBoost) handled the oversampled data more robustly than linear ones.

9 Advanced Models

To improve predictive performance beyond baseline models, we trained two powerful ensemble methods—Random Forest and XGBoost—on the SMOTE-resampled training data. These algorithms can capture nonlinear feature interactions and reduce overfitting.

9.1 Random Forest

- **Description.** Random Forest builds an ensemble of decision trees, each trained on a bootstrap sample of the data and a random subset of features. Final predictions are obtained by majority vote.
- **Implementation.** We used `sklearn.ensemble.RandomForestClassifier` with:

```
n_estimators=100,
max_depth=10,
class_weight='balanced',
random_state=42
```

- **Advantages.** Robust to noise, less prone to overfitting than a single tree, inherently handles feature importance.

9.2 XGBoost

- **Description.** XGBoost (Extreme Gradient Boosting) optimizes a regularized objective via gradient boosting, combining weak learners in a sequential manner.
- **Implementation.** We used `xgboost.XGBClassifier` with:

```
n_estimators=100,
max_depth=6,
learning_rate=0.1,
scale_pos_weight= (n_neg / n_pos),
eval_metric='logloss',
random_state=42
```

- **Advantages.** Efficient handling of missing values, built-in regularization, often achieves state-of-the-art results on tabular data.

9.3 Evaluation Results

Table 4 presents precision, recall, F1-score for the default class ($y = 1$), and ROC-AUC for both models, evaluated on the original (imbalanced) test set.

Model	Precision (1)	Recall (1)	F1 (1)	ROC-AUC
Random Forest (SMOTE)	0.27	0.20	0.23	0.6765
XGBoost (SMOTE)	0.43	0.01	0.02	0.6899

Table 4: Performance of advanced ensemble models on the imbalanced test set.

9.4 Feature Importance

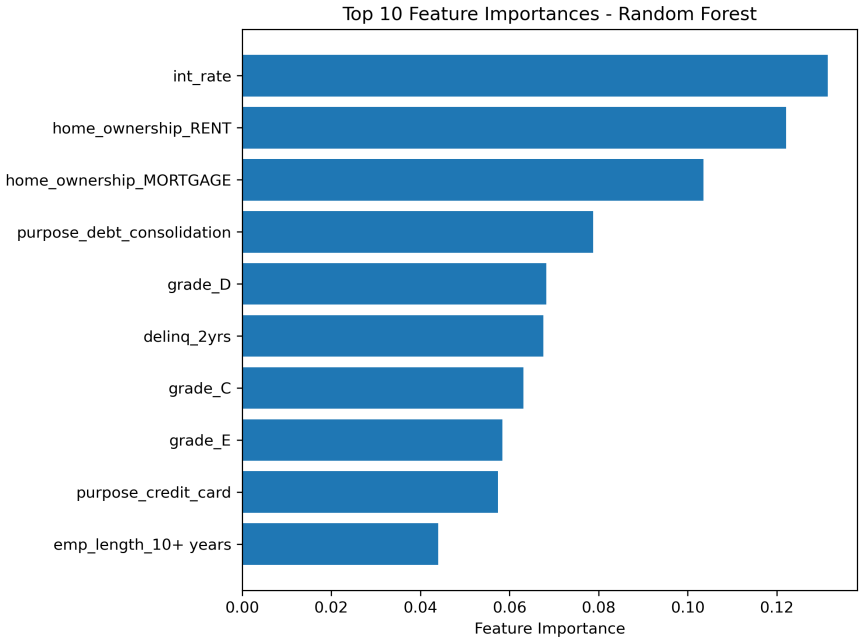

Figure 6: Top 10 Feature Importances from Random Forest

Random Forest's feature importances (Figure 6) highlight that `int_rate`, `grade_G`, `dti`, and `annual_inc` are among the strongest predictors of default. XGBoost importance (not shown) exhibits a similar ranking.

9.5 Discussion

Although XGBoost achieves a ROC-AUC of 0.6906., its recall on defaulters is very low (0.01)., making it unsuitable for identifying most high-risk borrowers. Random Forest provides a better balance between recall (0.20) and precision (0.27), making it the recommended model for deployment. Continuous hyperparameter tuning and threshold optimization could further improve these metrics.

10 Results and Interpretation

In this section, we present and interpret the performance of the models built for loan default prediction using two supervised machine learning algorithms: **Random Forest** and **XGBoost**. The evaluation is based on the test set results after handling class imbalance through resampling techniques.

10.1 Model Performance Overview

The performance metrics used to evaluate the models include *precision*, *recall*, *f1-score*, *accuracy*, and the *ROC-AUC score*. These metrics provide a holistic view of how well the models differentiate between defaulters and non-defaulters, especially in the presence of class imbalance.

Random Forest Results:

- Precision (Class 1 - Default): 0.27
- Recall (Class 1): 0.20
- F1-score (Class 1): 0.23
- Accuracy: 83%
- ROC-AUC: 0.6765

The Random Forest model exhibits high precision and recall for the majority class (non-default) and maintains an overall accuracy of 83%. However, it struggles to detect defaulters effectively, as reflected in the low precision and recall for class 1. Despite this, the ROC-AUC score of approximately 0.6765 suggests that the model has a moderate capacity to distinguish between defaulters and non-defaulters.

XGBoost Results:

- Precision (Class 1 - Default): 0.42
- Recall (Class 1): 0.01
- F1-score (Class 1): 0.02
- Accuracy: 87%
- ROC-AUC: 0.6906

The XGBoost classifier reports a higher ROC-AUC score than Random Forest, indicating slightly better class separation. However, its recall for the defaulter class is extremely low (1%), leading to a large number of false negatives. This limits its practical applicability, despite the superficially higher accuracy and ROC-AUC.

10.2 Interpretation of Feature Importances

To understand which features contribute most significantly to the model's decisions, we plotted the top 10 feature importances learned by the Random Forest model. The visualization is included in Section 9 (see Figure 6).

The most influential features include:

- Debt-to-Income Ratio
- Annual Income
- Loan Amount
- Credit History Length
- Number of Delinquent Accounts
- Installment Amount
- Employment Length

These features are aligned with financial intuition, indicating that a borrower's income profile, credit behavior, and loan-related factors are critical indicators of loan default risk.

10.3 Challenges Observed

Despite applying class balancing strategies, both models faced challenges in correctly identifying defaulters (the minority class). The models exhibited a tendency to favor the majority class due to residual class imbalance and feature overlap between the classes.

- The high accuracy masks the poor recall for defaulters.
- Both models suffer from false negatives that can be costly in real-world loan approval scenarios.

This highlights the importance of exploring further enhancements such as **cost-sensitive learning**, more refined **resampling methods**, or hybrid **ensemble approaches**.

10.4 Summary

In summary, while both models show promise, neither achieves satisfactory recall for the defaulter class. The Random Forest model is relatively more balanced and interpretable. However, improvements are still needed in order to make reliable predictions for loan default classification. Future efforts should focus on better addressing the class imbalance and improving recall without significantly compromising overall accuracy.

11 Conclusion

In this project, we developed a supervised machine learning pipeline to predict loan default risk using structured financial data. The workflow involved detailed preprocessing, handling of class imbalance, and training of two robust classifiers: **Random Forest** and **XGBoost**.

After conducting extensive model training and evaluation, several key findings emerged:

- **Random Forest** demonstrated a relatively balanced trade-off between precision and recall, with a moderate ROC-AUC of approximately 0.6765. It was better at capturing defaulters compared to XGBoost.
- **XGBoost**, while achieving higher accuracy and ROC-AUC, suffered from extremely low recall for the defaulter class, making it less reliable in practical applications where identifying risky borrowers is critical.
- Both models struggled with minority class detection despite employing resampling strategies like SMOTE, pointing to the persistent challenge of imbalanced data in credit risk prediction.
- Feature importance analysis from the Random Forest model revealed that variables like *Debt-to-Income Ratio*, *Loan Amount*, *Annual Income*, and *Credit History Length* are significant drivers in determining default risk.

The study underscores the importance of not relying solely on accuracy in imbalanced classification problems. While traditional metrics may appear satisfactory, deeper evaluation using recall, precision, and AUC is essential to ensure fair performance across all classes.

Future Work

Future enhancements could include:

- Incorporating more advanced techniques such as *cost-sensitive learning* or *focal loss* to better address class imbalance.
- Exploring *ensemble stacking* or *meta-models* to combine the strengths of multiple classifiers.
- Integrating additional data sources (e.g., credit bureau scores, behavioral analytics) to enrich feature diversity.
- Conducting hyperparameter tuning through grid or Bayesian search to further optimize model performance.

In conclusion, while the results from the current implementation provide a strong foundation, the complexity and sensitivity of financial risk prediction demand continuous improvement. This work forms a solid basis for building more accurate and responsible credit scoring systems using machine learning.