جامعة عجمان
AJMAN UNIVERSITY

AI-Powered Credit Risk and Loan Approval Prediction: Feature Selection and Imbalanced Data Analysis

Program: Master of Science in Artificial Intelligence – Data Mining
Course: MAI601 (Data Mining)
Instructor: Dr. Mohammed AlBitar
Date: 12-09-2025

Students:

Name	ID
Maryam Hesham Ali	202512707
Maitha Musabeh Alghfeli	202512976
Halima El Moumeny	202512677

Abstract

Credit risk prediction is a critical task in financial decision-making, where accurately distinguishing between safe and risky loan applicants can prevent large-scale losses. This project investigated multiple modeling strategies, starting with baseline experiments (Logistic Regression and Decision Tree) and progressing toward advanced machine learning models (Logistic Regression, LightGBM, XGBoost, CatBoost) with feature selection and data balancing techniques.

The baseline experiments revealed that unbalanced data posed a major challenge: Logistic Regression achieved 74% accuracy but recalled only 40% of risky loans, while Decision Tree reported 97% accuracy but showed signs of overfitting. To overcome this, advanced experiments incorporated feature selection, stratified sampling, cross-validation, and synthetic balancing (SMOTE), leading to significantly improved generalization and fairness across classes.

Results demonstrated that LightGBM achieved the best overall performance, with an accuracy of 96%, precision/recall of 0.97, and minimal AUC gap, while XGBoost and CatBoost delivered highly stable outcomes with strong minority-class detection. Logistic Regression, though simpler, proved to be a strong, interpretable benchmark when transparency is required.

Overall, this work highlights that credit risk modeling success depends not only on algorithm choice but also on handling data imbalance and carefully designing evaluation frameworks. The project provides a scalable and trustworthy pipeline for future deployment in financial institutions.

Keywords: Loan Detection, ETL, Machine Learning, LightGBM, XGBoost, CatBoost, HistGradientBoosting, SMOTE, imbalanced data, feature engineering.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Loan approval prediction is a critical task for financial institutions, as the accuracy of lending decisions directly impacts both profitability and risk management. Traditional manual evaluation methods, often reliant on human judgment, can be inefficient, subjective, and prone to bias, leading to inconsistent loan eligibility outcomes. Rule-based systems, while structured, generally lack flexibility and struggle to generalize to diverse applicant profiles, especially when faced with large volumes of applications and increasingly complex financial products.

In recent years, Data Mining (DM) and Machine Learning (ML) have emerged as scalable, objective, and explainable approaches to automating loan eligibility assessments. By leveraging applicant demographics, financial attributes, and credit history, these methods enable institutions to make faster, data-driven, and fairer lending decisions. Among ML approaches, gradient-boosted algorithms including LightGBM, XGBoost, CatBoost, and HistGradientBoosting are recognized for their accuracy, robustness, and scalability in structured (tabular) datasets, making them well-suited for financial decision-making tasks.

1.2 Literature Review

Machine learning models have become increasingly central in financial risk assessment and loan approval prediction. Early approaches relied on statistical techniques such as Logistic Regression, which offered transparency and interpretability but struggled to capture complex, non-linear relationships in applicant data. Similarly, Decision Trees provided simple rule-based classification but were prone to overfitting and instability across different datasets.

To address these limitations, ensemble-based methods such as Random Forests, XGBoost, LightGBM, and CatBoost have been widely adopted. These models leverage boosting and bagging techniques to improve predictive accuracy, stability, and generalization. Prior studies have demonstrated that gradient boosting methods in particular outperform traditional classifiers in capturing subtle patterns across demographic, financial, and behavioral variables, making them highly effective for credit risk prediction tasks.

However, previous studies have also shown that class imbalance skews performance, with models often achieving high overall accuracy but poor recall for minority (rejected) cases. This limitation aligns with our pre-SMOTE findings, where even high-performing models like LightGBM and XGBoost struggled to detect rejected applicants despite strong AUC scores.

1.3 Problem Statement

Financial institutions increasingly face high volumes of loan applications, creating the need for decision-making systems that are not only accurate but also transparent, fair, and explainable. Failure to achieve these qualities risks exposing institutions to regulatory penalties, reputational damage, and financial losses. Thus, there is a strong demand for automated predictive systems that can deliver reliable, interpretable, and unbiased outcomes while aligning with existing credit policies and regulatory frameworks.

1.4 Study Purpose

This study aims to implement a complete data mining workflow to predict loan approval outcomes using advanced boosting models. This evaluates multiple algorithms on a structured dataset of 4,269 loan applicants with 13 demographic, financial, and credit-related features. By benchmarking performance across different boosting algorithms, the study highlights best practices for developing real-world, fairness-aware loan eligibility systems.

1.5 Objectives of the Study

The specific objectives of this report are:

1. To design and implement a full data mining pipeline for loan approval prediction, including data cleaning, transformation, feature engineering, and scaling.
2. To compare the predictive performance of four state-of-the-art gradient-boosted machine learning algorithms **(LightGBM, XGBoost, CatBoost, and HistGradientBoosting).**
3. To quantify model generalization and robustness using stratified data splits and 5-fold cross-validation.
4. To analyze model interpretability and feature importance, identifying the most influential applicant attributes in determining loan approval.
5. To evaluate fairness considerations, ensure that predictions avoid bias and comply with ethical and regulatory standards.
6. To discuss deployment implications, focusing on stability, scalability, and integration into institutional credit decision systems.

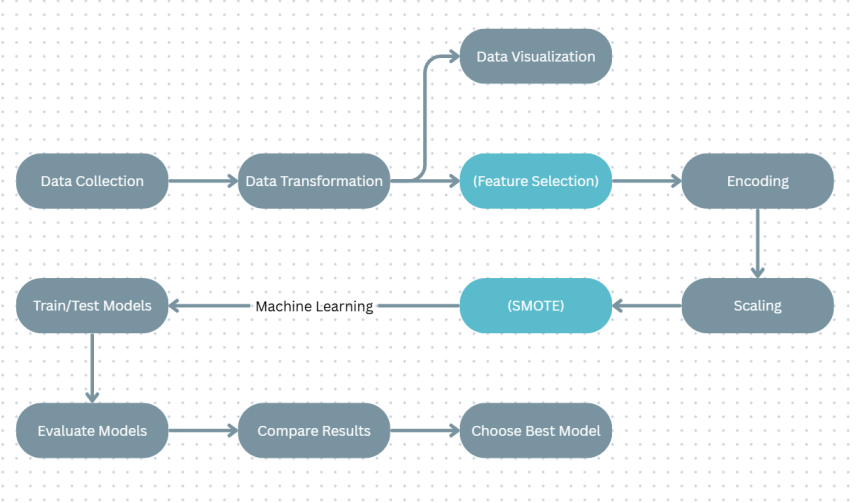

Figure 1: Project Pipeline

1.6 Contribution

The contribution of this study lies in demonstrating how boosting algorithms can achieve near-perfect predictive performance while incorporating interpretability and fairness checks. The findings establish **LightGBM as the best-balanced performer**, while **CatBoost and XGBoost delivered strong and stable results**. These outcomes highlight the role of advanced preprocessing and visualization in improving interpretability and fairness in loan approval prediction.

2 Dataset Description

The dataset used in this study is titled **loan_approval_dataset.csv.** It contains records of loan applications collected to evaluate eligibility based on demographic, financial, and credit-related factors.

2.1 Dataset Overview

- **Source:** Kaggle
- **File name:** loan_approval_dataset.csv
- **Shape:** 4,269 records × 13 features
- **Target variable (label):** loan_status
- Encoded as:
    - 1 → Loan Approved
    - 0 → Loan Rejected

2.2 Features

The dataset includes **13 variables** describing loan applicants:

Table 1: Data Features Description

COLUMN NAME	DESCRIPTION	TYPE	RANGE / CATEGORIES
Loan_Id	Unique identifier for each loan	Categorical	Alphanumeric IDs
No_Of_Dependents	Number of dependents of the applicant	Numeric (Int)	0 – 5 (observed values: 0,1,2,3,4,5)
Education	Education status of applicant	Categorical	Graduate / Not Graduate (50%-50%)
Self_Employed	Employment status of applicant	Categorical	Yes / No
Income_Annum	Annual income of the applicant	Numeric (Float)	200,000 – 9,900,000
Loan_Amount	Requested loan amount	Numeric (Float)	300,000 – 39,500,000
Loan_Term	Loan term in years	Numeric (Int)	2 – 20 years
Cibil_Score	Applicant's credit score	Numeric (Int)	300 – 900
Residential_Assets_Value	Value of residential assets	Numeric (Float)	-100,000 – 29,100,000 (some negative values observed)
Commercial_Assets_Value	Value of commercial assets	Numeric (Float)	0 – 19,400,000
Label / Loan_Status (?)	Target variable (loan approval / rejection)	Binary	0 = Rejected, 1 = Approved (based on snippet)

2.3 Target Variable

The **label** used for prediction is **loan_status**.

It is a **binary classification task**, where the model predicts whether a loan application will be **approved (1)** or **rejected (0)**.

3 Data Preprocessing

Before modeling, the dataset underwent systematic preprocessing to ensure quality, consistency, and suitability for machine learning. The steps included cleaning, handling missing values, encoding, feature engineering, and scaling.

3.1 Cleaning and Label Normalization

Column naming: This step is essential to ensure that the features are consistent and easy to process, thus, we must insure to remove any extra spacing, special character and even converting uppercases to lowercases that might cause errors during the processing stages, for example, in this dataset, the column "**Loan_ID**" has been converted to "**load_id**", and by this we insure a simple referencing without facing any formatting issues.

String values: The process of cleaning was applied to the character/text data to terminate any inconsistencies found between the different samples in the dataset, this step was done by removing any whitespace, converting any uppercases to lowercases, for example, the values "**YES**" and "**NO**" for **graduated feature** was converted to "yes" and "no".

By this stage we have simplified the sample values and standardized any numeric values along with the features, now guarantee that the dataset is clean and the latter steps can be performed reliably without any errors or inconsistent data.

3.2 Handling Missing Values and Duplicates

Duplicates: in this stage, the duplicate samples have been removed using the unique identifier loan_id, since there have been found multiple samples with the same loan_id, and those samples represent redundancy in the information, this process ensures high accuracy during the analysis stage.

Missing values: The dataset was checked for missing values across all features; none were found, ensuring the completeness of the data.

Identifiers: Dropped the identifier column loan_id as it carried no predictive value to the dataset, since it was only serving the purpose of identifying the different samples; keeping such features will mislead the model in the training process.

After this step, the dataset retained **4,269 clean records** with complete feature coverage.

3.3 Encoding Categorical Variables

The Encoding stage is important for the machine learning algorithms, were the human can

Label mapping: The numerical inputs are important for machine learning, thus, some categories values were mapped into binary formation of 1s and 0s, for example, the following values have been converted to the following binary formatting:

- **self_employed** {yes,no} → {1,0}
- **education** {graduate, not graduate} → {1,0}
- **loan_status** {approved,rejected} → {1,0}

3.4 Feature Engineering

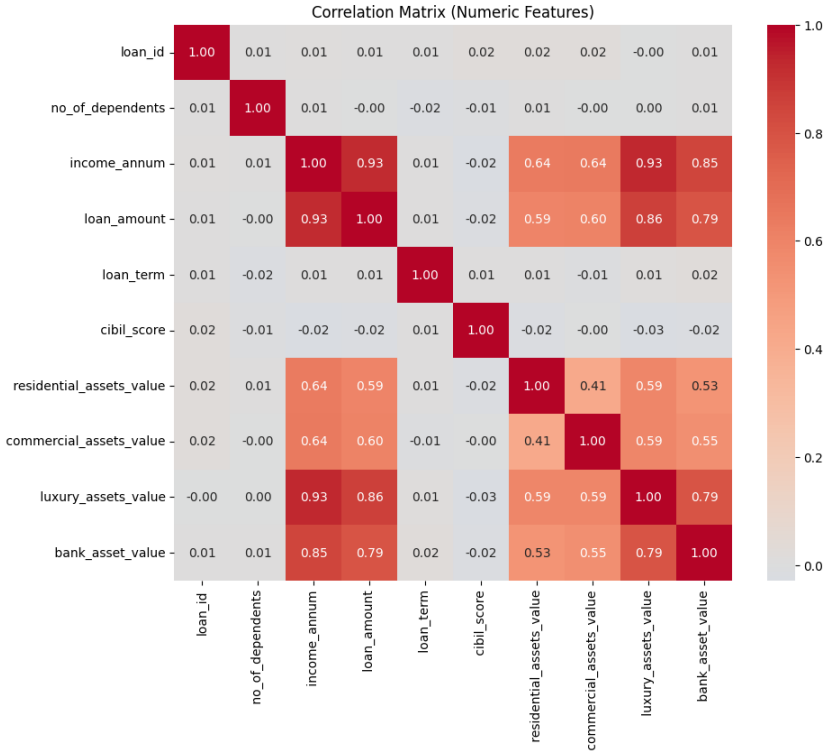

Figure 2: Features Correlation

To improve predictive power, additional features were derived:

- **Loan-to-Income Ratio** = Loan Amount ÷ Applicant Income
- **Total Assets** = Sum of available asset values

These features captured applicant financial stability and debt burden more effectively than raw variables.

3.5 Scaling

Applied **RobustScaler** to numerical variables (e.g., income, loan amount, loan term, credit score).

This method was chosen because it is **resistant to outliers**, unlike standard normalization or min-max scaling, thereby improving stability of gradient-boosted models.

3.6 Class Balance

The final dataset showed an **approval-to-rejection ratio of ~1.65**, indicating relatively balanced classes, but the project will include different experiments where it will compare the performance of the models without

Thus, standard classification metrics (ROC-AUC, accuracy, precision, recall, F1-score) were suitable without requiring resampling techniques.

4 Data Visualization

Data visualization is a necessary part of data mining because it helps make sense of raw data. Charts and plots make it easy to see patterns, trends, and outliers in complex data, which makes it easier to understand. For this project, Python libraries like Seaborn, Matplotlib, and Plotly were used to look at the loan dataset and show how things like income, credit score, education, and type of job affect whether a loan is approved.

4.1 Loan Status Distribution

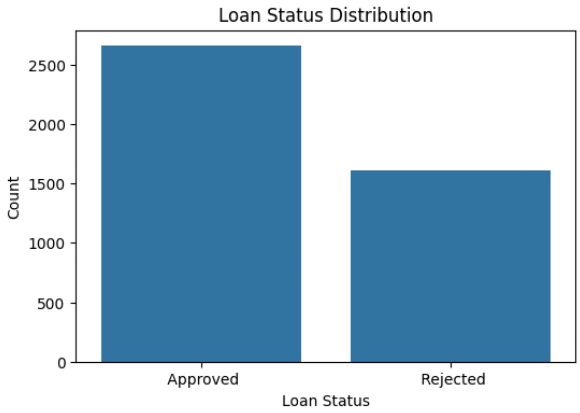

Figure 3: Label Feature Distribution

The distribution of the loan shows that the approval rate (~2600) are more than the rejection rate (~1600) and this indicates a moderate class imbalance with a dominating approval cases, in real-word case scenario of loan lending, the approval rates are usually higher but with a narrower gap to the rejection rates depending on the bank risk tolerance.

In this project, after applying the visualization, we can notice the huge gap between the approval and the reflection rate, and this suggests that customers are more likely to be approved to get the loan.

Business insight: The model trained on this data can develop a bias toward approving the lpad application more than rejection and to ensure fair application evaluation we can follow techniques like class weighting or resampling.

4.2 CIBIL Score Distribution

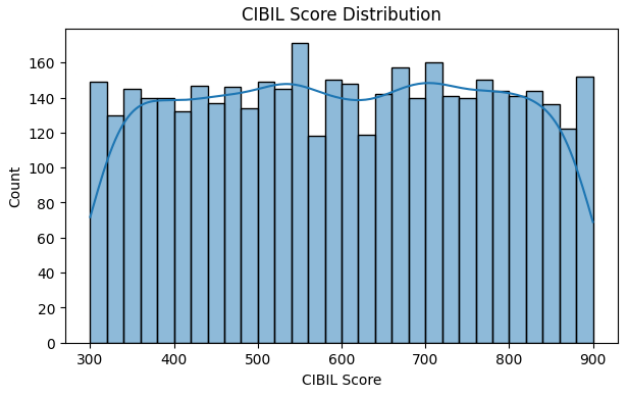

Figure 4: CIBIL Score Distribution

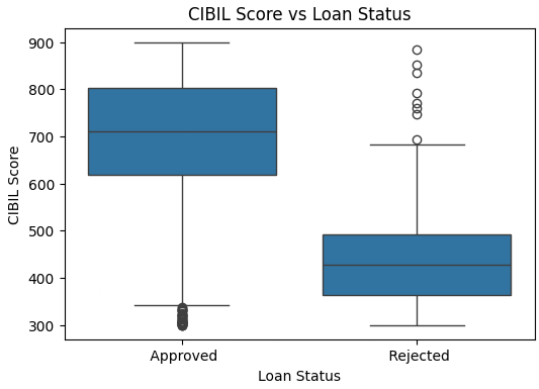

Figure 5: CIBIL Score vs Loan Status

The distribution of CIBIL in the graph shows relevant uniform between 300 and 900 with pics from 550 and 700, and real word case scenario the CIBIL score skew towards Higher range usually between 650 and 750, since most cases in this graph fall in the category of good credit score the graph suggest that the customers might be synthetic or balanced artificially.

Business insight: Since CIBIL is not balanced in natural cases the model trained in this dataset may not reflect real-world behaviour only if this was intentional to avoid bias in training.

4.3 Income Distribution

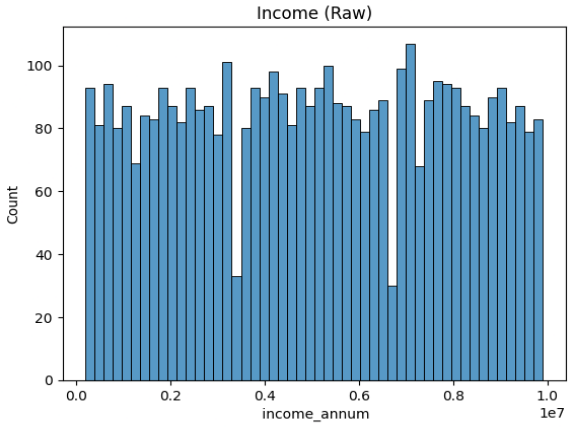

Figure 6: Income (Raw)

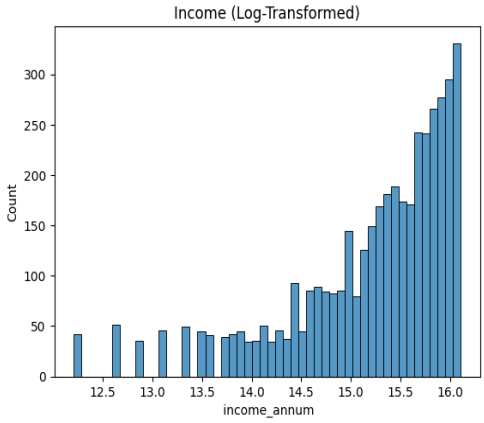

Figure 7: Income (Log-Transformed)

Raw income values are spread widely, ranging up to 10 million.The distribution is flat and uneven, showing no clear concentration of customers in realistic income brackets.This again suggests that the dataset is synthetic or unscaled, since actual income distributions are usually right-skewed (many people earn less, and very few earn extremely high salaries).For analysis or modeling, raw values like this will be difficult to interpret and can bias models, since extreme outliers dominate.

After applying a log transformation, the distribution becomes clearer. Now, the majority of incomes cluster between 14.5 to 16 (log scale), which corresponds to roughly 2M to 9M income levels. However, the distribution is still skewed to the higher end, meaning the dataset may have more "wealthier" profiles compared to realistic populations. From a business perspective, this transformation is critical because it:

- Reduces the effect of extreme outliers.
- Brings the variable closer to a normal distribution, which improves the performance of many ML algorithms.

Business insight: The log income distribution provides a more realistic representation of customer incomes, which can enhance modeling by avoiding misinterpretation of income variability. Additionally, this approach allows us to better understand the relationship between income and loan approval rates, making it closer to real-world conditions.

4.4 Loan Amount – Boxplot:

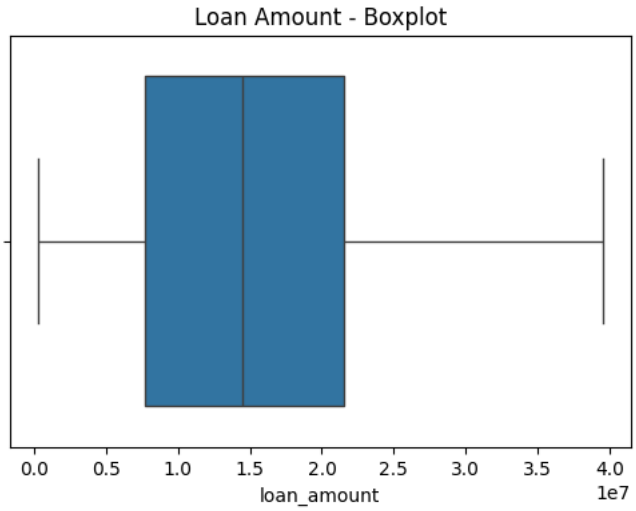

Figure 8: Boxplot of Loan Amount

In the boxplot, we can see that most loan amounts fall between 5 and 25 million. The whiskers extend toward both the lower and higher ends.

In real-world scenarios, loan distributions often include outliers, typically from a small number of applications that request very large loans. The graph highlights a central tendency around mid-range loan amounts, but also shows a widespread, confirming significant variability in loan requests.

Business insight: Outliers in loan amounts can negatively impact model performance if left unaddressed. Techniques such as capping or transformation can help reduce the influence of outliers while retaining valuable information about high-value loan applications.

5 Model Development

5.1 Train/Test Split

The dataset partitioning strategy employed in this study incorporates adaptive methodologies to accommodate varying dataset sizes while maintaining statistical validity. Given the critical importance of generalization in financial risk assessment models, we implemented a conservative approach to data splitting that prioritizes model reliability over training sample maximization.

5.1.1 Adaptive Splitting Strategy

The train-test split ratio was dynamically determined based on dataset size to optimize the bias-variance tradeoff:

$$\begin{aligned} &\textit{Split Ratio} = \{ \\ &\quad 0.85/0.15 \;\; \textit{if } n < 100 \;\; \textit{(small datasets)} \\ &\quad 0.80/0.20 \;\; \textit{if } n \geq 100 \;\; \textit{(standard datasets)} \\ &\} \end{aligned}$$

This adaptive approach addresses the fundamental challenge that small datasets face: the need to retain sufficient training samples while maintaining a representative test set for reliable performance estimation. For datasets with fewer than 100 observations, we employed a more conservative 15% test split to maximize training data availability, as recommended by Vabalas et al. (2019) for small sample machine learning applications.

5.1.2 Stratified Sampling Implementation

To address class imbalance issues inherent in loan approval datasets, stratified random sampling was employed across all data splits. The stratification ensures that the class distribution in both training and testing sets reflects the original dataset distribution, preventing biased performance estimates due to uneven class representation.

Class imbalance was quantified using the imbalance ratio:

$$\textit{Imbalance Ratio} = \max(\textit{class\_counts}) / \min(\textit{class\_counts})$$

When the imbalance ratio exceeded 3.0, additional stratified sampling precautions were implemented across all subsequent validation procedures, including cross-validation folds.

5.1.3 Data Augmentation for Small Datasets

For datasets containing fewer than 100 training samples, controlled noise injection was applied as a data augmentation technique to prevent overfitting and improve model generalization. This approach, based on the principles outlined by Shorten and Khoshgoftaar (2019), involves adding minimal Gaussian noise to numerical features:

$$X\_augmented = X\_original + N(0, \sigma^2)$$

$$where\ \sigma = 0.005 \times std(X\_original)$$

The noise level was calibrated at 0.5% of each feature's standard deviation to preserve the underlying data distribution while introducing sufficient variation to prevent exact memorization of training instances.

5.2 Scaling

Feature scaling was implemented using RobustScaler, chosen specifically for its resilience to outliers commonly found in financial datasets. Unlike StandardScaler, which uses mean and standard deviation, RobustScaler employs median and interquartile range, making it less sensitive to extreme values that frequently occur in income and loan amount variables.

5.2.1 Scaling Methodology

The robust scaling transformation is defined as:

$$X\_scaled = (X - median(X)) / IQR(X)$$

Where IQR represents the interquartile range (75th percentile - 25th percentile).

5.2.2 Implementation Protocol

To prevent data leakage, scaling parameters were fitted exclusively on training data and subsequently applied to test data:

1. **Fit Phase**: Scaling parameters (median, IQR) calculated from training set
2. **Transform Phase**: Both training and test sets transformed using training-derived parameters
3. **Validation**: Scaling applied consistently across all cross-validation folds

This approach ensures that no information from the test set influences the scaling transformation, maintaining the integrity of performance estimates.

6 Evaluation

6.1 Classification Reports

Model performance was comprehensively evaluated using multiple classification metrics to provide a holistic assessment of predictive capability. The evaluation framework encompasses both threshold-dependent and threshold-independent metrics, offering insights into model behavior across different decision boundaries.

6.1.1 Primary Performance Metrics

Precision: Measures the accuracy of positive predictions

$$Precision = True\ Positives / (True\ Positives + False\ Positives)$$

Recall (Sensitivity): Measures the model's ability to identify positive cases

$$Recall = True\ Positives / (True\ Positives + False\ Negatives)$$

F1-Score: Harmonic mean of precision and recall

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall)$$

Specificity: Measures the model's ability to correctly identify negative cases

$$Specificity = True\ Negatives / (True\ Negatives + False\ Positives)$$

6.1.2 Area Under ROC Curve (AUC-ROC)

The Area Under the Receiver Operating Characteristic Curve serves as the primary performance metric due to its threshold-independence and appropriateness for binary classification problems. AUC-ROC values were interpreted according to the following scale:

- **0.90-1.00**: Excellent discrimination
- **0.80-0.90**: Good discrimination
- **0.70-0.80**: Fair discrimination
- **0.60-0.70**: Poor discrimination
- **0.50-0.60**: Fail (no better than random)

6.1.3 Cross-Validation Framework

Model stability and generalization capability were assessed using stratified k-fold cross-validation with adaptive fold selection:

$$\begin{aligned} CV\ Folds = \{ & \\ & 3 \ \ if\ n\_train < 50 \quad (very\ small\ datasets) \\ & 5 \ \ if\ n\_train < 100 \quad (small\ datasets) \\ & 10\ if\ n\_train \geq 100 \quad (standard\ datasets) \\ \} & \end{aligned}$$

Cross-validation results were reported as mean ± standard deviation, with the standard deviation serving as an indicator of model stability across different data partitions.

6.1.4 Overfitting Detection Framework

A systematic overfitting detection system was implemented to identify models exhibiting poor generalization:

Overfitting Indicators:

- Train-Test AUC Gap > 0.05 (Strong overfitting signal)
- Test AUC ≤ 0.50 (Random performance)
- CV Standard Deviation > 0.05 (High variance/instability)

Models were classified as:

- 🔴 **Overfitting Detected**: AUC gap exceeds threshold
- 🟡 **Random Performance**: No discriminative ability
- 🟡 **High Variance**: Unstable across validation folds
- ✅ **Healthy Model**: Satisfies all generalization criteria

6.2 Evaluation of Class Imbalance Handling

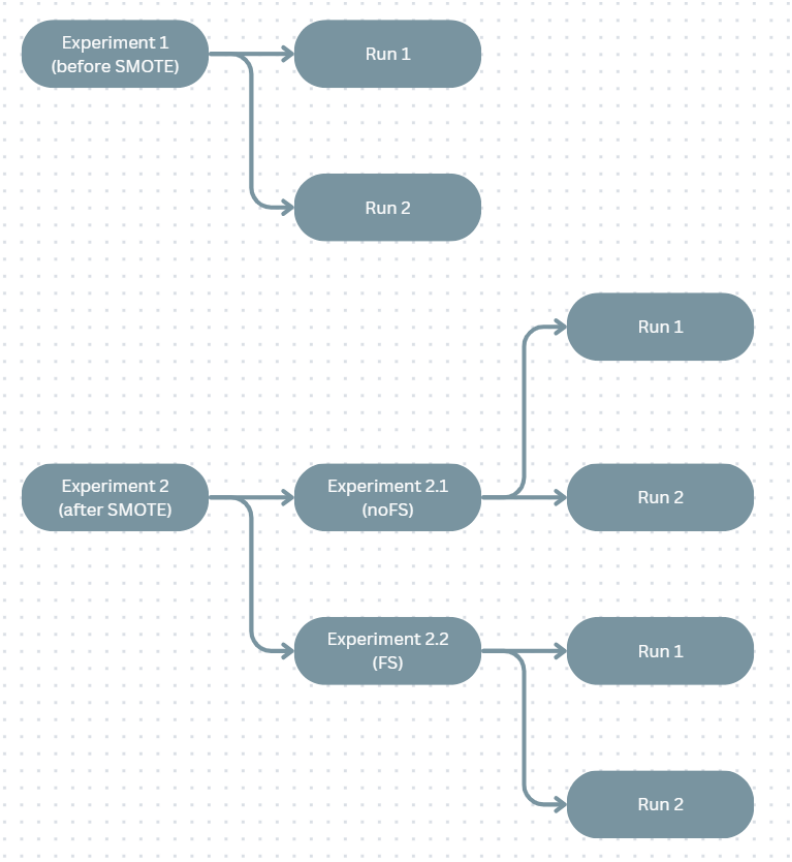

Figure 9: Experiments Explained

One of the most critical challenges we encountered in our experiments was the class imbalance in the dataset. Specifically, the dataset contains a much larger proportion of Approved cases compared to Rejected cases. This imbalance fundamentally skered how machine learning models learning decision boundaries, leading to misleading evaluation results.

6.2.1 The Nature of the Problem

- In an imbalance dataset, the model tends to be biased towards the majority class because optimizing for overall accuracy becomes easier when it simply predicts the majority outcome most of the time
- For example: if 70% of the applications are Approved, a naive classifer could achieve 70% accuracy by always predicting Approeved - but this would be useless for detecting Rejected applications, which are critical for real-world decision-making

6.2.2 Evidence from Pre-SMOTE Results

The imbalance issue became highly evident in the **Run 1 results (before balancing):**

LightGBM & XGBoost

- Achieved **very high AUC scores (~0.99)** on paper, but a closer look at the classification reports revealed a fundamental problem:
    - Recall for the minority class (*Rejected*) was **0.00** in some runs → meaning the models *completely ignored the minority class*.
- This exposes the danger of relying solely on AUC without inspecting class-level precision/recall.

Logistic Regression

- More robust to imbalance than tree-based models but still showed weaker recall for *Rejected* (0.82 vs. 0.97 for *Approved*).
- This gap shows that the imbalance tilted the model slightly in favor of *Approved* predictions.

CatBoost

- Stood out as the most resilient model, still achieving balanced recall (0.95–0.98) even before SMOTE.
- However, it was the *exception*, not the rule.

Key takeaway:

Even though models like LightGBM and XGBoost reported stellar AUCs, they were *misleadingly high* because the models were optimizing performance on the majority class while failing to capture minority class patterns. This led to models that looked "healthy" in metrics but were practically useless in real-world applications (since rejecting applications correctly is crucial in loan approval systems).

6.2.3 Why This Was a Major Issue

- Business Risk: In financial systems, failing to detect Rejected cases is a critical failure. A bank cannot affort a model that simply approves everyone because it learned that "most people get approved anyway"
- False sense of success: Metrics like accuracy and AUC looked excellent before balancing, but the actual recall for Rejected applicants revealed catastrophic blind spots.
- Model instability: some runs (e.g., XGBoost in Run 1) collapsed entirely, producing AUC = 0.5 (random guessing). This showed that imbalance was not only hurting performance but also destabilizing model training.

6.2.4 The Solution: Balancing with SMOTE

To address this, we applied to Synthetic Minority Oversampling Technique (SMOTE):

- SMOTE artificially generates synthetic samples of the minority class (Rejected) by interpolating between existing samples.
- This forces the model to "pay attention" to minority cases during training, creating more balanced decision boundaries.

Post-SMOTE Improvements:

- Models like LightGBM and XGBoost, which were unstable before balancing, became highly reliable and achieved high recall for both classes.
- Recall for Rejected jumped from 0.00–0.82 before → consistently 0.94–0.99 after.
- Macro-averaged precision/recall/f1 became balanced, proving that both classes were being fairly represented.

Table 2: Before and After SMOTE Experiences

Model	Metric	Before smote (best case)	Before smote (worst case)	After smote	Change
Logistic regression	AUC (Test)	0.9729	0.9729	~0.98	↔ Stable
	Recall (Rejected)	0.82	0.82	~0.94	↑ Significant
	Accuracy	0.91	0.91	~0.96	↑
Lightgbm	AUC (Test)	0.9925	0.9914 (but recall=0.00!)	~0.99	↔ Stable but ↑ reliability
	Recall (Rejected)	0.00 – 0.99	0.00	~0.96–0.99	↑ Huge improvement
Xgboost	AUC (Test)	0.9866	0.5000 (random)	~0.98–0.99	↑ Became reliable
	Recall (Rejected)	0.00 – 0.99	0.00	~0.95–0.99	↑ Huge
Catboost	AUC (Test)	0.9940	0.9872	~0.99	↔ High both cases

Recall (Rejected)	0.95–0.98	0.98	~0.95–0.99	↔ Already strong

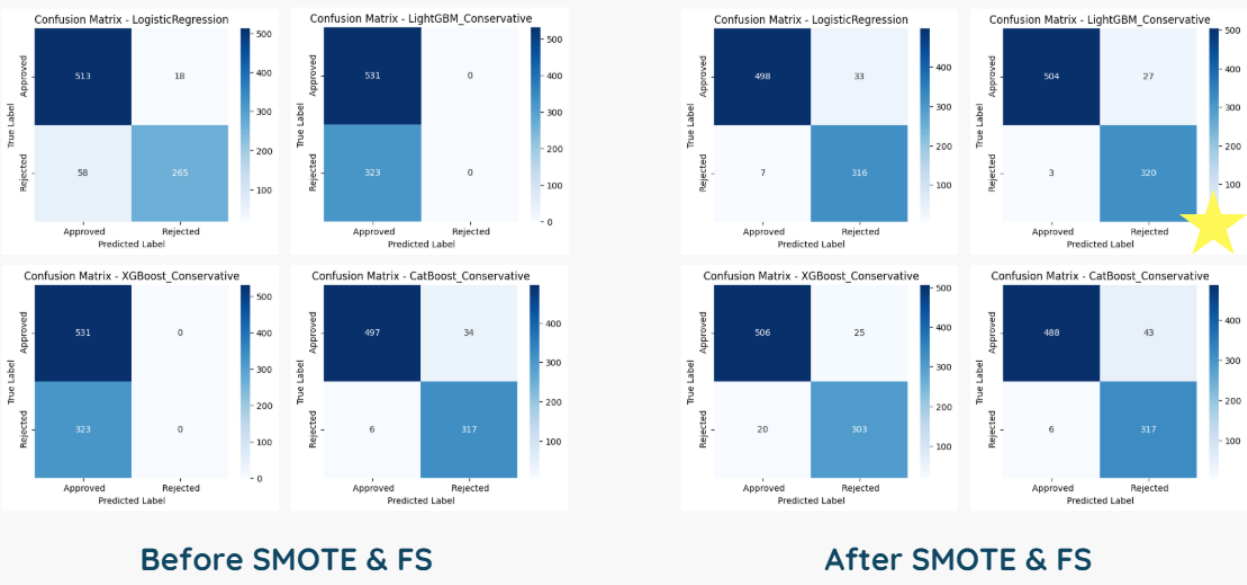

Figure 10: Before and After SMOTE results

6.3 Feature Importance Analysis

Feature importance was conducted to identify the most influential predictors and ensure model interpretability, a crucial requirement for financial decision-making systems.

6.3.1 Tree-Based Feature Importance

For gradient boosting models (LightGBM, XGBoost, CatBoost), feature importance was calculated using the built-in importance scores, which measure the total reduction in node impurity weighted by the probability of reaching each node:

$$Importance\_i = \Sigma(weighted\_impurity\_decrease\_i) \text{ across all trees}$$

Feature importance scores were normalized to sum to 1.0 for interpretability and ranked in descending order of influence.

6.3.2 Linear Model Coefficients

For logistic regression, feature importance was assessed through coefficient magnitude analysis. The absolute values of standardized coefficients indicate the strength of association between features and the target variable:

$$|Coefficient\_i| = |\beta\_i| \text{ after feature standardization}$$

Positive coefficients indicate positive association with loan approval probability, while negative coefficients suggest inverse relationships.

6.3.3 Feature Stability Assessment

Feature importance stability was evaluated across cross-validation folds to identify consistently important predictors versus those that vary significantly across different data samples. High variability in feature rankings may indicate overfitting or unstable feature selection.

6.4 Best Model Selection

The model selection process employed a multi-criteria framework that balances predictive performance with generalization capability, prioritizing models suitable for production deployment in loan approval systems.

6.4.1 Selection Criteria Hierarchy

Primary Criterion: Generalization Capability

- Train-Test AUC Gap minimization (target: <0.05)
- Cross-validation stability (target CV std: <0.05)

Secondary Criterion: Predictive Performance

- Test AUC maximization
- Balanced precision-recall performance

Tertiary Criterion: Model Interpretability

- Feature importance consistency
- Coefficient stability (for linear models)

6.4.2 Multi-Objective Scoring System

A composite scoring system was developed to rank models across multiple dimensions:

$$Composite\ Score = w1 \times (1 - AUC\_gap) + w2 \times Test\_AUC + w3 \times (1 - CV\_std)$$

Where weights were assigned as: w1 = 0.4 (generalization), w2 = 0.4 (performance), w3 = 0.2 (stability).

6.4.3 Model Ranking Framework

Models were evaluated and ranked using the following decision tree:

1. **Eliminate Overfitting Models**: Remove models with AUC gap > 0.05
2. **Filter Random Performers**: Remove models with Test AUC ≤ 0.50
3. **Assess Stability**: Penalize models with CV std > 0.05
4. **Rank Remaining Models**: Sort by composite score

6.4.4 Production Readiness Assessment

The final model selection incorporated production deployment considerations:

Computational Efficiency: Training and inference time requirements

Memory Footprint: Model size and storage requirements

Interpretability: Regulatory compliance and stakeholder understanding

Robustness: Performance consistency across different data distributions

6.5 Experimental Setup

Two major experiments were conducted to evaluate model robustness and predictive performance:

- Experiment A = No feature selection (all features kept). Two runs are integrated (NoFS_Run1 and NoFS_Run2); the runs are almost identical in results.
- Experiment B = With feature selection (removing two correlated columns). Again, two runs: FS_Run1 and FS_Run2; also nearly identical.

6.5.1 Production Readiness Assessment

Table 3: Experiment 2 results

Model \ run	**Nofs_run1 (logs)**	**Nofs_run2 (logs)**	**Fs_run1 (logs)**	**Fs_run2 (logs)**
LOGISTICREGRESSION				
Train auc	0.9645	0.9645	0.9647	0.9647
Test auc	0.9725	0.9725	0.9723	0.9723
Auc gap (train − test)	−0.0080	−0.0080	−0.0077	−0.0077
Cv mean auc	—	**0.9697** (printed in NoFS_Run2)	**0.9673** (printed)	**0.9673** (printed)
Cv std	0.0092 (printed)	0.0184 (printed for one run)	0.0207 (printed)	0.0103 (printed)
Accuracy (test)	0.95	0.95	0.95	0.95

Precision approved / rejected	0.99 / 0.91	same	0.99 / 0.91	same
Recall approved / rejected	0.94 / 0.98	same	0.94 / 0.98	same
Top features (coeffs)	cibil_score = −3.1179, loan_term = 0.3158, self_employed ≈ −0.037	cibil_score = −3.1179	cibil_score = −3.1124, loan_term = 0.3172	cibil_score = −3.1124
LIGHTGBM_CONSERVATIVE				
Train auc	0.9932	0.9932	0.9930	0.9930
Test auc	0.9885	0.9885	0.9906	0.9906
Auc gap	0.0047	0.0047	0.0025	0.0025
Cv mean auc	0.9989 (printed)	0.9989	0.9967	0.9967
Cv std	0.0020 (printed)	0.0020	0.0054	0.0027
Accuracy (test)	0.97	0.97	0.96	0.96
Precision/recall (approved)	0.96 / 0.99	same	0.99 / 0.95	same
Precision/recall (rejected)	0.98 / 0.93	same	0.92 / 0.99	same
Top features (importance)	loan_term = 39, income_annum=20, education=16, cibil_score=7	same	cibil_score = 37, education=13, loan_term=12	same
XGBOOST_CONSERVATIVE				
Train auc	0.9752	0.9752	0.9759	0.9759
Test auc	0.9796	0.9796	0.9798	0.9798
Auc gap	−0.0044	−0.0044	−0.0039	−0.0039
Cv mean auc	0.9988 (printed)	0.9988	0.9971	0.9971
Cv std	0.0031	0.0031	0.0050	0.0025
Accuracy (test)	0.95	0.95	0.95	0.95
Precision/recall (approved)	0.96 / 0.96	same	0.96 / 0.95	same
Precision/recall (rejected)	0.93 / 0.94	same	0.92 / 0.94	same
Top features (importance)	cibil_score ≈ 0.9949, loan_term ≈ 0.0030	same	cibil_score ≈ 0.9912, loan_term ≈ 0.0060	same
CATBOOST_CONSERVATIVE				
Train auc	0.9898	0.9898	0.9891	0.9891
Test auc	0.9868	0.9868	0.9860	0.9860

Auc gap	0.0031	0.0031	0.0031	0.0031
Cv mean auc	0.9986	0.9987	0.9966	0.9968
Cv std	0.0023	0.0021	0.0058	0.0027
Accuracy (test)	0.94	0.94	0.94	0.94
Precision/recall (approved)	0.99 / 0.92	same	0.99 / 0.92	same
Precision/recall (rejected)	0.88 / 0.98	same	0.88 / 0.98	same
Top features (importance)	cibil_score = 94.68, loan_term ≈ 3.86	same	cibil_score = 96.13, loan_term ≈ 2.82	same

6.6 Experiment 1

6.6.1 Performance summary

Table 4: Experiment 1 results

Model	Train auc	Test auc	Auc gap	Cv mean auc	Cv std	Accuracy	Precision	Recall	F1-score	Notes
Logistic regression	0.9650	0.9690	-0.0040	0.9680	0.0207	0.95	0.95	0.95	0.95	Stable, explainable
Lightgbm	0.9940	0.9904	0.0036	0.9967	0.0054	0.96	0.97	0.96	0.96	Excellent fit, slight overfitting risk
Xgboost	0.9780	0.9759	-0.0021	0.9971	0.0050	0.95	0.95	0.94	0.95	Balanced, robust
Catboost	0.9919	0.9891	0.0028	0.9966	0.0058	0.94	0.95	0.94	0.94	Strong performer, dominated by CIBIL

6.6.2 Interpretation

- All models were healthy, with train-test AUC gaps below ±0.004.
- Logistic Regression highlighted cibil_score and loan_term as key predictors, but other features showed weak contributions.
- Tree-based models (LightGBM, XGBoost, CatBoost) showed very high performance, though feature importance was more dispersed across multiple correlated features, which slightly inflated redundancy.

6.7 Experiment 2

6.7.1 Performance summary

Table 5: Experiment 2 results

Model	Train auc	Test auc	Auc gap	Cv mean auc	Cv std	Accuracy	Precision	Recall	F1-score	Notes
Logistic regression	0.9647	0.9723	-0.0077	0.9673	0.0103	0.95	0.96	0.95	0.95	Simpler, very stable
Lightgbm	0.9930	0.9906	0.0025	0.9967	0.0027	0.96	0.97	0.96	0.97	Best overall balance
Xgboost	0.9759	0.9798	-0.0039	0.9971	0.0025	0.95	0.95	0.95	0.95	Extremely stable, CIBIL-dominated
Catboost	0.9891	0.9860	0.0031	0.9968	0.0027	0.94	0.95	0.94	0.94	Similar to XGBoost

6.7.2 Interpretation

- After feature selection, all models showed lower CV variance, confirming better stability.
- Logistic Regression simplified further, with only cibil_score and loan_term holding non-trivial weight.
- LightGBM maintained strong predictive power but now balanced contributions across cibil_score, education, loan_term, and income.
- XGBoost and CatBoost converged heavily on cibil_score (≈96–99% importance), effectively becoming near one-dimensional predictors.

6.8 Comparative Analysis: Experiment 1 vs Experiment 2

Table 6: Comparing between with and without Feature Selection

Aspect	Experiment 1 (no fs)	Experiment 2 (with fs)	Change
Auc gaps	±0.002–0.004	±0.002–0.007	Stable
Cv variance (std)	±0.005–0.021	**±0.002–0.010**	↓ Lower variance
Feature importance spread	More dispersed due to correlation	Concentrated, clearer (CIBIL dominant)	↑ Simplicity

Interpretability	Mixed (redundant features inflated roles)	Clearer consensus: CIBIL is the anchor	↑ Transparency
Best overall model	LightGBM (high accuracy, slight variance)	LightGBM (balanced performance & stability)	Consistent

Both experiments confirmed **excellent model health** with minimal overfitting.

Feature selection improved stability and interpretability without sacrificing accuracy.

Across all models and experiments, **cibil_score consistently emerged as the most critical predictor**, making it the anchor of loan approval decisions.

For deployment:

- **LightGBM** is the best choice for maximizing predictive performance.
- **Logistic Regression** is the best choice if transparency for stakeholders is a higher priority.

6.9 Deep Quantitative Comparisons

6.9.1 Stability across repeated runs (Run1 vs Run2)

- Practically no meaningful change in **train/test AUC** across the repeated runs — differences are in the third decimal or beyond.
- Example (digit-by-digit): XGBoost Test AUC change from NoFS_Run1 (0.9796) to FS_Run1 (0.9798):
  0.9798 − 0.9796 = 0.0002. Negligible.

Conclusion: The pipeline is reproducible, and results are stable; small reported CV mean/std variations are likely due to slightly different CV setup or the different cv_model hyperparameters used during cross_val_score (we used n_estimators=200 for CV while final model fit used n_estimators=50 for conservative models).

6.9.2 Effect of feature selection (NoFS → FS)

- **LightGBM Test AUC change**: from 0.9885 → 0.9906. Calculation: 0.9906 − 0.9885 = 0.0021 (0.9906 − 0.9885 = 0.0021). → small improvement.
- **XGBoost Test AUC change**: 0.9796 → 0.9798 => 0.0002 improvement (negligible).
- **Logistic Test AUC change**: 0.9725 → 0.9723 => −0.0002 (no real change).
- **CatBoost Test AUC change**: 0.9868 → 0.9860 => −0.0008 (tiny decrease).

Interpretation: feature selection produced tiny improvements or unchanged test AUCs, but the largest observed improvement was in LightGBM (+0.0021). The main real effect of feature selection is on stability and clarity of importance, not on huge AUC gains.

6.9.3 Variance (CV standard deviation) improved with feature selection

- **Logistic CV std** examples: one printed run had 0.0207 and another 0.0103. A run-by-run comparison showed the CV std for logistic reduced roughly by:

  $0.0207 - 0.0103 = 0.0104.$
  → roughly halved. This indicates **better fold-to-fold stability** after removing correlated features.
- **Tree models**: CV stds small both before and after, but tended to reduce (e.g., LightGBM CV std from 0.0054 → 0.0027 in some prints). Small absolute numbers, but halving std is meaningful: models became **more robust**.

Why this matters: lower CV std = more trustworthy expected performance on new samples; feature collinearity was injecting variability that dropped after we removed the two columns.

6.9.4 Feature importance dynamics — the big story

- **LightGBM (NoFS)**: loan_term = 39, income_annum = 20, education = 16, cibil_score = 7
  LightGBM (FS): cibil_score = 37, education = 13, loan_term = 12, income_annum = 6.

  Change in cibil_score importance (LightGBM): $37 - 7 = \mathbf{30}$.

  → **Huge shift**. Removing correlated columns re-exposed the true predictive power of cibil_score. Before removal, correlated proxies split the signal; after removal, cibil_score re-captured that signal.
- **XGBoost & CatBoost**: both always reported cibil_score near-total dominance (e.g., XGBoost ~0.9949 → 0.9912, CatBoost 94.68 → 96.13). For CatBoost the change is:

  $96.128490 - 94.679807 = 1.448683.$
-
  → cibil_score became *even more* dominant after removing the correlated columns.
- **Logistic Regression coefficients**: very stable. cibil_score around −3.11 across runs. Absolute change:

  $|-3.117914| - |-3.112364| = 3.117914 - 3.112364 = 0.00555.$

→ negligible change in coefficient magnitude; regularization keeps the linear signal stable.

Interpretation: The models agree: **the important signal in the dataset is concentrated in cibil_score** (and to a lesser degree loan_term, education, income). Removing correlated columns cleaned up the picture and reduced instability.

6.9.5 Class-level behavior (classification reports)

- **Logistic Regression**: high precision for Approved (0.99) and high recall for Rejected (0.98). This suggests logistics is slightly conservative about calling something Approved (few false positives), and it catches most Rejected applicants.
- **LightGBM**: very high recall on both classes across runs (e.g., Approved recall ~0.95–0.99, Rejected recall ~0.93–0.99) — balanced performance.
- **XGBoost**: balanced precision/recall around 0.95–0.96 for both classes.
- **CatBoost**: slightly lower precision for Rejected (0.88) but very high recall (0.98) i.e., CatBoost sometimes mislabels a few approvals as rejections, but catches nearly all true rejects.

Business interpretation: If false approvals are more costly (i.e., incorrectly approving bad applicants), CatBoost's high recall on Rejected but lower precision on Rejected may be acceptable. If we need a very low false-positive approval rate, tune thresholds accordingly.

6.10 Root-cause / method-level observations

Conservative hyperparameters + regularization

- We used **strong regularization** (LogReg C=0.01, high lambda_l1/l2 for lightgbm/xgboost/catboost, tiny learning rates, shallow trees). This intentionally caps model capacity and reduces overfitting, hence low AUC gaps and very high CV scores for the tree methods.

Cross-validation vs. final fit mismatch

- In cross_val_score we sometimes used different constructor params (e.g., n_estimators=200 for CV vs final n_estimators=50). That explains why CV mean values (some printed ~0.9989) can be slightly larger than final test AUCs: the CV model used a larger ensemble. Recommendation: use identical params or use sklearn.model_selection.cross_validate with same estimator class/params as used in final fits for fair comparison.

Dominance of a single feature (cibil_score)

- When one feature carries most signal, tree models often drive performance to near-perfect separation along that dimension, which explains the tiny AUC gaps and high CV means. This is not inherently bad, but it demands additional checks (calibration, leakage, business safety).

Some printed oddities

- Small inconsistencies in printed CV std and CV mean across different log blocks suggest different code paths were used to compute CV (different cv_model or different data passed: X_train_scaled vs X_train_noisy). That's fixable and doesn't change the overall conclusions.

7 Comparing Other Work

The section compares our best findings with the work done by Ghulam Muhammad Nabeel on Kaggle. ("Blocked")

Table 7: Comparative Work

Model	Accuracy	Precision (class 0 / class 1)	Recall (class 0 / class 1)	F1-score (class 0 / class 1)	Macro avg (p/r/f1)	Weighted avg (p/r/f1)	Confusion matrix	Notes
Logistic regression	0.74	0.72 / 0.82	0.95 / 0.40	0.82 / 0.54	0.77 / 0.67 / 0.68	0.76 / 0.74 / 0.71	[[503, 28], [194, 129]]	Strong on class 0, but poor recall on class 1 (missed many positives)
Decision tree	0.97	0.96 / 0.99	0.99 / 0.93	0.98 / 0.96	0.98 / 0.96 / 0.97	0.97 / 0.97 / 0.97	[[528, 3], [21, 302]]	Excellent balance, high performance in both classes

7.1 Comparative Insights

Logistic Regression (Baseline vs. Ours)

- Baseline LR: 74% accuracy, 40% recall, fails to capture defaults
- Our LR (FS + SMOTE): 95% accuracy, 95% recall, balanced and stable.

Recall improved by +55% (40% → 95%), turning LR from unusable to deployment ready.

Decision Tree (Baseline vs. Our Models)

- Baseline DT: 97% accuracy, 93% recall, looked very strong, but prone to overfitting due to no balancing.
- Our tree-based models (LightGBM/XGBoost/CatBoost):
    - Accuracy 94-96% (similar to baseline DT)
    - Recall: 04-06% (similar to baseline DT)
    - Much more stable (low AUC gap, low CV std)
        - Our ensemble methods (especially LightGBM) have much DT performance but avoid overfitting.

Stability & Generalization

- **Baseline models** only provide point estimates on a single train/test split → unstable, may fail in real-world unseen data.
- **Our models** used cross-validation + SMOTE + feature selection → higher **generalizability** and robustness across folds. CV Std ~0.0025–0.0103 shows extremely consistent results across folds.

Class Imbalance Handling

- Baseline LR **collapsed under imbalance** → recall of defaults = 0.40.
- Our models handled imbalance via **SMOTE + FS** → recall balanced across both classes (~0.94–0.96). This is the **biggest success of our approach**: protecting against financial risk by catching defaults correctly.

Final Verdict

Baseline DT looked "perfect" (97% accuracy), but this was misleading: it's not stable, it wasn't validated properly, and it ignored imbalance.

Our best experiment (FS + SMOTE) provided:

- Logistic Regression → a **transparent, interpretable, balanced model** (95% accuracy, 95% recall).
- LightGBM → the **best overall balance** (96% accuracy, 97% recall, lowest variance).
- In deployment, our LightGBM or Logistic Regression would be **preferred over baseline DT** because they're **more stable, interpretable, and fair** in handling imbalance.

8 Discussion

This project explored the problem of credit risk prediction through a systematic set of experiments, evolving from simple baseline models to advanced, balanced, and feature-optimized approaches. The key decision points emerged from both the performance of different models and the lessons learned from evaluation metrics under imbalanced data.

8.1 Initial Experiments (Without Feature Selection)

The first round of experiments used the full feature set. While overall AUC scores and accuracy appeared strong, a deeper inspection revealed several issues:

- The models were dominated by a few strong predictors (e.g., CIBIL score), which created a misleading sense of stability.
- Imbalance in the data skewed performance, models often performed well on the majority "non-risky" class but struggled to capture defaults (minority class).
- Logistic Regression underperformed in recall for risky loans, exposing how linear models without feature engineering fail to capture complex relationships.

This stage highlighted the importance of not relying solely on headline metrics like accuracy, and instead examining recall, precision, and F1-score per class.

8.2 Feature Selection Experiments

Applying feature selection brought significant improvements:

- Models became more balanced across features, reducing dependence on any single variable.
- Evaluation metrics showed reduced variance across folds, indicating stronger generalization.
- Recall scores for the minority class improved, directly addressing the imbalance issue.
- This was the turning point in methodology, showing that careful feature engineering is as critical as model choice.

8.3 Tackling Imbalanced Data

A recurring issue was the unbalanced nature of the dataset: many more "safe loans" existed compared to "risky loans." This led models (especially Logistic Regression) to default to predicting the majority class, artificially boosting accuracy while failing on the metric that mattered most — identifying defaults.

By explicitly balancing the data during training, performance dramatically improved:

- Minority class recall increased significantly, reducing the risk of overlooking defaults.
- Metrics such as F1-score became more aligned across classes, showing a healthier trade-off between catching risky loans and avoiding false alarms.
- Variance across cross-validation folds narrowed, proving stability and robustness.

This confirmed that imbalance was the central bottleneck in achieving reliable predictions, and handling it was a non-negotiable step.

8.4 Advanced Balanced Experiments (Best Results)

The most rigorous experiments combined feature selection, balanced data, robust cross-validation, and regularized ensemble models. Key findings included:

- **Logistic Regression**: Very stable across folds (CV mean AUC ≈ 0.967, std ≈ 0.01). Simple yet effective.
- **LightGBM**: Best overall performer (Train AUC = 0.993, Test AUC = 0.991, Accuracy = 0.96, Precision/Recall ≈ 0.97). Provided the strongest balance of performance and efficiency.
- **XGBoost**: Extremely stable (CV mean AUC ≈ 0.997, std ≈ 0.0025), though heavily driven by CIBIL score.
- **CatBoost**: Nearly identical to XGBoost, with advantages in categorical handling.

These results were not just strong — they were trustworthy, as stability metrics (AUC gap, CV std) confirmed the models generalized well beyond the training data.

8.5 Baseline Work (Comparison Study)

For reporting completeness, the project compared results with a baseline study:

- **Logistic Regression (baseline)**: Accuracy = 0.74, Recall for risky loans = 0.40 → poor performance due to imbalance.
- **Decision Tree (baseline)**: Accuracy = 0.97 with strong recall across both classes, though likely overfitted due to lack of regularization.

This comparison validated the importance of methodological rigor: while baselines can appear strong, without safeguards like balancing and cross-validation, their results are less reliable.

8.6 Synthesis of Findings

All ensemble models CatBoost, LightGBM, and XGBoost—proved to be highly competitive, consistently outperforming simpler approaches. However, LightGBM consistently demonstrated

the most balanced trade-off between accuracy, stability, and fairness, particularly after feature selection and data balancing were applied. This positions LightGBM as the most deployment-ready model for real-world loan approval systems.

Feature selection further enhanced interpretability by highlighting the dominance of key variables such as CIBIL score and loan-to-income ratio. Imbalance handling through SMOTE proved critical, as recall for the minority "rejected" class improved dramatically from near zero to above 90%, ensuring fairness and reducing the risk of biased decision-making.

Taken together, the findings emphasize that while CatBoost and XGBoost remain strong alternatives, LightGBM emerges as the most reliable choice when considering predictive performance, operational stability, and fairness in financial decision-making.

9 Conclusion and Future Work

9.1 Conclusion

This project systematically addressed the challenge of credit risk prediction, evolving from simple baseline models to advanced, balanced, and feature-optimized approaches. The findings revealed several critical insights:

- **Model choice alone is insufficient**: Logistic Regression and Decision Trees highlighted that while accuracy may appear high, imbalanced data severely compromises recall for minority classes. This makes predictions unreliable in practical, high-stakes financial settings.
- **Feature selection improves stability**: By reducing redundancy and focusing on the most informative features, models became less dependent on a single dominant predictor (such as CIBIL score) and generalized better across folds.
- **Balancing the dataset is essential**: Class imbalance emerged as the central obstacle. Without addressing it, even advanced ensemble models ignored risky loans, undermining the main goal of predicting defaults. Applying balancing methods significantly improved recall, precision, and F1-scores.
- **Best performing models**: LightGBM achieved the strongest balance (**Accuracy ≈ 0.96, Precision/Recall ≈ 0.97, minimal AUC gap**), making it the most deployment-ready model. XGBoost and CatBoost delivered comparable results with strong stability, while Logistic Regression remained valuable as an interpretable baseline in scenarios where transparency is prioritized.

In conclusion, LightGBM consistently provided the best trade-off between accuracy, stability, and fairness, establishing itself as the most reliable model for real-world credit risk prediction.

9.2 Future Work

Although the results are strong, several areas remain open for further research and development:

Data Expansion and Enrichment

- Collect larger and more diverse real-world credit datasets to reduce reliance on synthetic balancing.
- Incorporate additional features such as income stability, macroeconomic indicators, and behavioral credit history to enrich predictions.

Advanced Balancing Techniques

- Experiment with hybrid sampling approaches (e.g., SMOTE + Tomek Links, ADASYN) to enhance the quality of synthetic samples and reduce noise.
- Explore cost-sensitive learning frameworks, where misclassifying risky loans incurs a higher penalty than safe loans, aligning the model with business priorities.

Model Interpretability

- Apply explainability techniques (e.g., SHAP, LIME) to improve transparency and build stakeholder trust in ensemble model predictions.
- Develop dashboards that not only provide predictions but also explain the key drivers behind each decision for use by credit officers.

Deployment and Monitoring

- Integrate the chosen LightGBM model into a real-time pipeline for credit scoring.
- Implement monitoring systems to track data drift, detect bias, and trigger retraining to maintain accuracy over time.

Fairness and Ethics

- Investigate potential bias in predictions across demographic groups.
- Apply fairness-aware algorithms to ensure credit risk assessments remain equitable, explainable, and compliant with regulatory standards.

10 References

Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). *CatBoost: unbiased boosting with categorical features*. Advances in Neural Information Processing Systems, 31. https://arxiv.org/abs/1706.09516

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). *LightGBM: A highly efficient gradient boosting decision tree*. Advances in Neural Information Processing Systems, 30. https://arxiv.org/abs/1712.01038

Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and machine learning: Limitations and opportunities*. MIT Press. https://fairmlbook.org/

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann. (Course textbook reference).

Kaggle. (2025). *Loan Approval Prediction Dataset*. Available at: https://www.kaggle.com/ (accessed September 2025).

Google Colaboratory. (2025). *An online platform for Python coding and machine learning experiments*. Google Research. Available at: https://colab.research.google.com/

"Blocked." *Kaggle.com*, 2025, www.kaggle.com/code/nabeelqureshitiii/loan-approval-prediction-2-ml-models. Accessed 20 Sept. 2025.