CREDIT RISK MODELING

PREDICTING LOAN DEFAULTS WITH MACHINE LEARNING

RICHA CHAUDHARI

B.A. APPLIED MATHEMATICS (GPA: 3.94)

SOUTHERN NEW HAMPSHIRE UNIVERSITY

DECEMBER 2025

Abstract

This project develops and compares machine learning models to predict loan defaults using the "Give Me Some Credit" dataset. LightGBM with class weighting achieved optimal performance with 70.4% recall and 28.3% precision, providing a practical solution for imbalanced credit risk assessment.

Introduction to Research Question

Credit risk modeling is a cornerstone of modern financial institutions, directly impacting profitability and stability. This problem is critical in banking and finance, as accurately identifying high-risk customers helps institutions minimize losses and maintain profitability. This project focuses on developing a binary classification model to predict whether a borrower will experience serious delinquency of 90 days past due within 2 years. These models help banks identify the likelihood of loan default, reducing time spent on manual assessments and improving lending decisions. (Corporate Finance Institute, n.d.). However, real-world credit data presents significant challenges, such as severe class imbalance where defaults are rare, and complex, non-linear relationships between borrower characteristics and their likelihood of default. This research will apply and evaluate advanced techniques to address these challenges, aiming to build a model that is not only statistically sound but also operationally viable for financial decision-making. (Rogojan, Croicu, & Iancu, 2023).

I selected this area because it combines statistical reasoning, feature engineering, and model evaluation in a real-world context which align closely with my background in applied mathematics. I am particularly interested in understanding how different predictive models such as logistic regression, random forest, and xgboost, can classify borrowers as likely to default or not.

Recent advancements in machine learning have significantly improved how lenders assess creditworthiness beyond traditional credit scores. By building a model that predicts loan default probability, I aim to explore how preprocessing techniques (such as SMOTE, class weighting, etc. for class imbalance) and model tuning affect precision, accuracy and recall, which are key metrics for financial decision-making.

Research Question

The primary research question for this study is:

“To what extent do data balancing techniques, including class weighting and sampling methods, improve the performance of logistic regression, random forest, LightGBM, and XGBoost models in predicting credit default within imbalanced financial datasets, and which approach optimally balances recall against precision?”

By systematically testing preprocessing, feature engineering, and threshold tuning, I will determine which approach yields the best trade-off between recall (detecting defaulters) and precision (avoiding false alarms).

Information and Data Sources

The primary data source is the "Give Me Some Credit" dataset from Kaggle, which contains 150,000 anonymized records of credit applicants with 12 predictive features, including age, MonthlyIncome, DebtRatio, and past delinquency history. The target variable, SeriousDlqin2yrs, indicates whether a borrower was 90+ days past due within two years. The dataset is publicly available and widely used in credit scoring research, making it appropriate for academic analysis. (Kaggle, 2011)

Each record represents an individual borrower with the target variable SeriousDlqin2yrs, indicating whether the person defaulted within two years (1 = default, 0 = non-default). Key predictors include but not limited to:

- RevolvingUtilizationOfUnsecuredLines – “Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits” (Kaggle, 2011)

- DebtRatio – Monthly debt payments, alimony, living costs divided by monthly gross income. (Kaggle, 2011)
- MonthlyIncome – Monthly Income
- NumberOfTimes90DaysLate – historical credit behavior indicator (Kaggle, 2011)

Assessment of Information

The gathered information is highly appropriate for creating an effective mathematical model. The Kaggle dataset provides a realistic, large-scale sample of the relevant features used in the credit industry. However, the dataset is highly imbalanced, with far fewer default cases than non-defaults, which can bias models toward predicting "no default."

A limitation of the dataset is that income data includes missing values and extreme outliers. I plan to mitigate this by using median imputation, log transformations, and feature scaling. Additionally, I will cap unrealistic debt ratios and replace placeholder values (like "98" in past-due columns) with more meaningful imputations.

Despite these limitations, this dataset is widely accepted in academic research, and preprocessing steps will ensure reliable modeling. The information supports the creation of an effective mathematical model through both statistical and machine learning approaches, allowing for a robust comparison of interpretability and predictive power.

Selection

For this project, I selected LightGBM as my main predictive model and compared it with weighted logistic regression, Random Forest, and XGBoost. Since the problem is to predict whether a customer will default or not, this falls under supervised classification. The dataset is also highly imbalanced, with only about 6.8% of the records representing default cases, so I needed a model that handles imbalance without losing important information. LightGBM is well suited for this because it uses gradient boosting, which captures non-linear patterns, complicated interactions between variables, and subtle behavior changes that linear models cannot detect. It also supports class weighting directly, which allowed me to use the full dataset without oversampling or undersampling. Research on credit-risk and fraud detection problems has shown that LightGBM often performs better than other tree-based or linear models when dealing with extreme class imbalance, because of its leaf-wise growth strategy and built-in regularization (Zhao et al., 2024).

Models	**Precision**	**Recall**	**F1**	**Threshold**
Weighted Logistic	0.1151	0.7601	19.99	0.43
Weighted Random Forest	0.2324	0.7007	34.90	0.64
Balanced Random Forest	0.2849	0.6908	40.34	0.12
Xgboost	0.271	0.709	39.2	0.58
LightGBM	0.2825	0.7040	40.32	0.61

To support this choice, I compared LightGBM with weighted logistic regression, Random Forest, and XGBoost using five-fold cross-validation. LightGBM showed the strongest and most consistent performance, achieving a cross-validation AUC of 0.8636 with very low variance across folds, along with slightly better generalization on the independent test set (AUC = 0.8766). Threshold-based comparison further supported this result, as weighted logistic regression achieved high recall only at the cost of extremely low precision, making it impractical for operational use. Tree-based models performed substantially better, with Random Forest, XGBoost, and LightGBM achieving similar recall levels near 0.70 while maintaining higher precision. Among these, LightGBM provided the most balanced trade-off between precision and recall at its optimal threshold and showed more stable threshold behavior, which is critical for minimizing false alarms while still identifying high-risk borrowers. These results gave me confidence in selecting LightGBM as the final model.

Model Creation and Process

Before building the model, I first split the dataset using a stratified method so the class distribution would remain consistent in both training and testing sets. After the split, all cleaning and feature engineering steps were done strictly on the training data and then applied to the test data in the same way. This prevents any information from the test set from leaking into the model, which is one of the main causes of overfitting. Out of 12 columns, only Monthly_Income and Number_Of_Dependents have missing values of 29,731 and 3,924 respectively. To interpret whether the missing values were random or related to other variables, I calculated the mean for groups where these values were missing vs. not missing. Customers with missing income displayed extremely high debt ratios (1,673%) and slightly lower delinquency rates, indicating that missing income is not random and can serve as a useful predictor. Similarly, individuals with missing dependent information were older (average age 60) and showed a lower default rate (4.56% vs. 6.74%), supporting the idea that missingness has signal value.

Based on these findings, I created binary indicator variables such as Income_Missing, and Dependents_Missing, along with highdebtflag to capture a high-risk subgroup. After this, the dataset was split into 85% training and 15% testing. To prevent data leakage, the steps below were performed separately on both the training and testing sets to prepare the data for analysis:

- Replaced missing values in Monthly_Income with the median calculated from the training set.
- Replaced missing values in Number_Of_Dependents with 0 because we previously concluded that the average age for missing dependents is 60, so it makes sense to input 0.
- Created the highdebtflag binary flag in both the training and testing sets.

- The value '98' appeared to be an outlier/placeholder in past-due columns, so I replaced it with the median.
- Unrealistic ages < 18 were replaced with the median adult age.

Since DebtRatio is highly skewed, I capped it at the 99th percentile of the training set and applied the same cap to the test set. This prevents extremely large debt ratios from dominating the model. For the three past-due variables, I created new binary flags to indicate extreme delinquency whenever the value exceeded 10 and then capped the original variables at 10. This allows the model to keep useful information about unusually high delinquency without letting rare outliers distort the distribution. I also handled extreme values in RevolvingUtilizationOfUnsecuredLines by creating a flag to mark unusually high utilization and then capping the values using the same threshold as in the training data. To make income less skewed and more stable for modeling, I added a log-transformed version of MonthlyIncome.

Process

To understand how explanatory variables relate to default, I used a simple weighted logistic regression and diagnostic checks. The logistic model and p-values helped identify which variables had significant linear associations in log-odds space, while VIF analysis (performed on the unweighted logistic model) confirmed that multicollinearity was low (VIFs < 5). These linear checks informed feature selection and helped me interpret directionality. Because some relationships appeared non-linear and interactive in exploratory plots, I chose a boosted-tree model (LightGBM) for the main model so it could learn non-linear effects and interactions automatically, while keeping a weighted logistic regression as a transparent baseline for interpretation and regulatory relevance.

Model training emphasized generalization and class imbalance handling. I calculated a scale_pos_weight from the training set to correct for the roughly 14:1 class imbalance and used it in LightGBM rather than oversampling or undersampling, which can introduce bias or duplicate noise. LightGBM was trained with regularization (min_data_in_leaf, feature_fraction, bagging_fraction, lambda_l1, lambda_l2) and early stopping on a held-out validation set to prevent overfitting. Performance was validated with k-fold (five-fold) cross validation and reported on the untouched test set using AUC, precision, recall, and F1. I also performed a threshold sweep to find operational decision thresholds that trade recall and precision according to business priorities.

Tools

The model was developed using Jupyter Notebook with R for statistical modeling and visualization. R was chosen over Python for its superior statistical modeling capabilities and more robust implementation of weighted regression models in financial applications. While alternatives like Excel lack the computational power for large-scale machine learning, and specialized tools like Minitab are less suited for advanced predictive modeling, R provides the ideal balance of statistical rigor and machine learning capabilities for this credit risk analysis. Libraries such as dplyr, caret, pROC, and VIF diagnostic packages supported preprocessing, feature selection, evaluation, and ROC analysis. The LightGBM library was used to build and train the final ensemble model, and five-fold stratified cross-validation was employed to assess performance and guide model selection. (Manikandan, 2024) .These tools were selected because they support transparent modeling, robust analysis, and efficient testing, providing an appropriate framework for answering the research question.

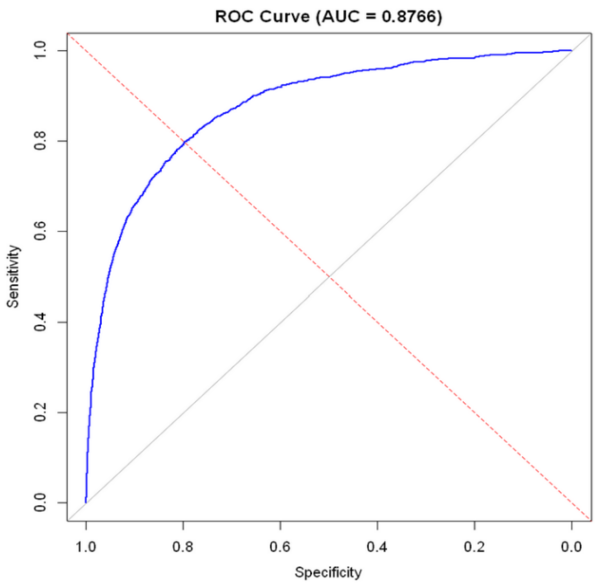
Analysis

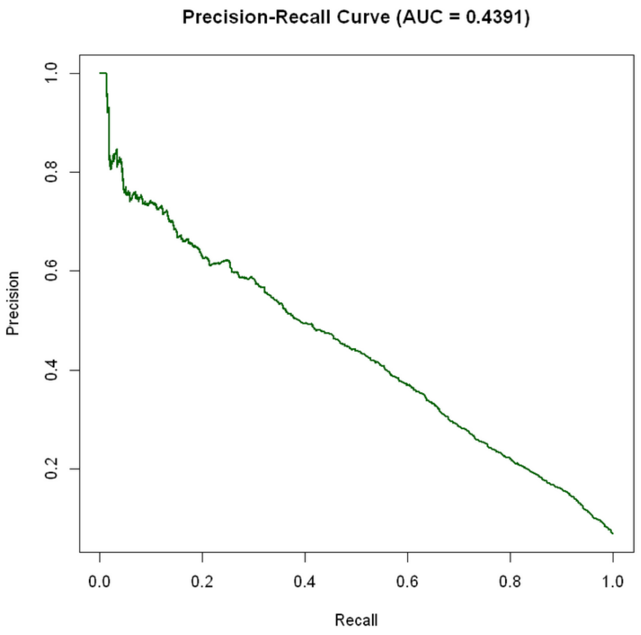
In this project, I trained a LightGBM model using five-fold stratified cross-validation with regularization and early stopping. Although the maximum number of boosting iterations was set to 3,000, training stopped automatically at iteration 295 because the validation AUC stopped improving for 80 consecutive rounds. During training, the AUC increased rapidly at first and then improved more gradually before leveling off, which indicates that the model learned meaningful patterns without overfitting.

The cross-validation results were consistent across all folds, with a mean AUC of 0.8636 and very little variation. This suggests that the model generalizes well and is not sensitive to how the data is split. When evaluated on the independent test set, the model achieved an AUC of 0.8766, which is slightly higher than the cross-validation average. Since the difference is small (less than 0.02), this further supports that the model did not overfit and performs well on unseen data.

LightGBM's histogram-based training method handled the dataset efficiently using automatic multi-threading and 1,236 feature bins. The model started with an initial score of −2.64, reflecting the low baseline default rate, and updated through 295 boosting rounds. Regularization techniques such as L2 penalties, feature subsampling, bagging, and tree constraints helped control model complexity, while the low learning rate ensured stable and gradual learning. Overall, the model demonstrated strong predictive performance while remaining well-regularized. (GeeksforGeeks, 2023).

The ROC curve for the final LightGBM model shows strong class separation. The curve rises quickly toward the top-left corner and stays well above the diagonal reference line, indicating performance far better than random guessing. The test-set AUC of 0.8766 confirms that the model distinguishes effectively between default and non-default cases across a wide range of thresholds.





Because the dataset is highly imbalanced, the Precision–Recall curve provides additional insight into performance on the minority class. The PR curve begins with high precision at low recall and quickly declines as recall increases, which is expected as the model identifies more difficult cases. The PR AUC of 0.4391 is substantially higher than the baseline default rate of approximately 0.066, showing that the model captures meaningful structure in the data rather than relying on chance.

Thresholds	**Accuracy**	**Precision**	**Recall**	**F1**
0.60	0.8545	0.2766	0.7165	0.3991
0.61	0.8595	0.2825	0.7040	0.4032
0.65	0.8776	0.3108	0.6777	0.4246
0.70	0.8949	0.3470	0.6341	0.4486
0.75	0.9105	0.3886	0.5702	0.4642

To determine an appropriate classification threshold, I evaluated precision, recall, accuracy, and F1 score across several probability cutoffs. Lower thresholds increased recall but reduced precision, reflecting the typical trade-off in credit-risk modeling. The F1 score peaked around thresholds between 0.60 and 0.62, indicating the most balanced trade-off between identifying defaults and limiting false positives. Accuracy remained high across all thresholds due to class imbalance, making F1 a more informative metric in this context. Based on these results, a threshold near 0.61 provides a reasonable balance for practical use.

Limitations

Although the LightGBM model performs well overall, it has several important limitations. The dataset is highly imbalanced, which causes accuracy to remain high even when default cases are not well identified. This imbalance also explains the noticeable gap between ROC AUC and Precision–Recall AUC, showing that predicting the minority class remains challenging. Small changes in the classification threshold can lead to large shifts in precision and recall, which limits how confidently the model can be used without careful threshold selection.

Despite its strong predictive performance, LightGBM has several limitations compared to traditional credit assessment methods. Its advantages are most pronounced with large datasets, while simpler models may perform more reliably when data is limited. In addition, tuning LightGBM requires careful selection of hyperparameters and can be computationally demanding. Another key limitation is interpretability, as the ensemble structure makes it harder to clearly explain individual predictions to stakeholders. Prior research suggests that combining LightGBM with more interpretable traditional models may help balance predictive performance and transparency (Brown et al., 2024).

Finally, the dataset lacks richer credit-related variables, such as detailed payment histories or external credit scores. This restricts the model's predictive power and may prevent it from capturing more nuanced risk patterns. Although regularization and early stopping help reduce overfitting, boosting models always carry some risk of learning patterns specific to the training data.

Approach

To answer the research question, I followed a structured modeling process. First, I separated the predictor variables from the target variable, SeriousDlqin2yrs, and kept the existing training and test split so that final evaluation would be performed on completely unseen data.

I selected LightGBM because it performs well on large, tabular datasets and is commonly used in risk modeling problems. Since the dataset is highly imbalanced, I applied class weighting using the scale_pos_weight parameter so the model would place more emphasis on default cases during training.

During model training, I used regularization techniques including L1 and L2 penalties, feature subsampling, and bagging to control model complexity. Early stopping was also applied to prevent overfitting by halting training once validation performance stopped improving. These steps helped ensure that the final model remained stable and generalizable. (GeeksforGeeks, 2023).

After training, I evaluated the model using multiple performance metrics rather than relying on accuracy alone. I examined ROC and Precision–Recall curves to assess overall discrimination and minority-class performance. I also conducted threshold analysis by testing multiple probability cutoffs and computing precision, recall, accuracy, and F1 score for each threshold. This allowed me to understand how different decision thresholds affect model behavior in a class-imbalanced setting.

Overall, the approach followed a clear sequence: data preparation, regularized model training, prevention of overfitting, and evaluation using both graphical and quantitative metrics. This process provided a complete understanding of the model's performance and its suitability for the research question.

Applicability

The goal of this model is to predict whether an individual is likely to default on a loan within two years using applicant financial data. LightGBM is well suited for this type of structured, tabular credit-risk data, which makes it an appropriate choice for addressing the research question.

The model achieved a strong ROC AUC of approximately 0.87, indicating good separation between default and non-default cases. Threshold analysis further supports practical use, as it allows decision-makers to adjust the trade-off between recall and precision depending on risk tolerance. For example, lenders who prioritize identifying high-risk applicants may choose a lower threshold to increase recall, while those focused on reducing false positives may select a higher threshold.

Early stopping and regularization help ensure that the model generalizes well to new data, making it suitable for use on unseen loan applications. Although class imbalance limits precision at higher recall levels, the model remains useful for ranking applicants by risk and supporting probability-based decision-making rather than relying on a fixed cutoff.

Overall, the model effectively answers the research question by providing reliable default probability estimates and supporting risk-based credit decisions, while remaining consistent with the practical constraints of real-world credit screening.

Regulatory Compliance and Interpretability Constraints

While LightGBM achieved superior predictive performance (AUC: 0.877), its implementation in regulated lending environments requires careful consideration. Canadian financial institutions operating under the Bank Act and provincial consumer protection legislation (such as Ontario's Consumer Reporting Act) must provide transparent, explainable lending decisions. Additionally, the Office of the Superintendent of Financial Institutions (OSFI) guidelines emphasize model risk management and interpretability for financial decision-making. LightGBM's ensemble structure, while excellent for capturing complex patterns, lacks the transparency of logistic regression, whose coefficients directly translate to actionable decline reasons. For production deployment in Canadian financial institutions, a hybrid approach using LightGBM for initial risk screening and logistic regression for final, explainable decisions would balance predictive power with regulatory compliance, consistent with research on combining ensemble methods with interpretable models (Brown et al., 2024). This represents a key industry insight: in credit risk modeling, particularly within Canada's regulated financial sector, the optimal model often depends not only on metrics but on operational constraints and regulatory requirements.

Citation:

Corporate Finance Institute. (n.d.). *Credit risk analysis models – Overview, credit risk types, factors.* Retrieved October 30, 2025, from https://corporatefinanceinstitute.com/resources/commercial-lending/credit-risk-analysis-models/

Rogojan, L., Croicu, A., & Iancu, L. (2023). *Modern approaches in credit risk modeling: A literature review*. *Proceedings of the International Conference on Business Excellence, 17*(1), 1617–1627. https://doi.org/10.2478/picbe-2023-0145

Kaggle. (2011). *Give Me Some Credit* [Dataset]. Retrieved from https://www.kaggle.com/c/GiveMeSomeCredit

Bakırarar, Batuhan & ELHAN, Atilla. (2023). Class Weighting Technique to Deal with Imbalanced Class Problem in Machine Learning: Methodological Research. Turkiye Klinikleri Journal of Biostatistics. 15. 19-29. 10.5336/biostatic.2022-93961.

Zhao, Xiaosong & Liu, Yong & Zhao, Qiangfu. (2024). Improved LightGBM for Extremely Imbalanced Data and Application to Credit Card Fraud Detection. IEEE Access. PP. 1-1. 10.1109/ACCESS.2024.3487212.

Manikandan, Veera. (2024). Enhancing Energy Efficiency and Classification Modeling Through a Combined Approach of LightGBM and Stratified KFold Cross. Electric Power Components and Systems. 10.1080/15325008.2024.2315213.

GeeksforGeeks. (2023, December 9). *LightGBM Feature parameters*. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/lightgbm-feature-parameters/

Brown, Emily & Williams, David & Smith, Jessica & Johnson, Michael & Deshmukh, Aarav & Rodriguez, Sophia. (2024). Comparative Analysis of LightGBM with Traditional Credit Assessment Methods. 10.13140/RG.2.2.29039.65444.

Appendix:

Step 1: Review Data and Interpretation of missing values

```
#loading training dataset
credit_data <- read.csv("cs-training.csv", header = TRUE, sep = ",")
credit_data$X <- NULL
#Print number of columns
print("Number of Columns: ")
ncol(credit_data)
print("Number of Rows: ")
nrow(credit_data)
#head(credit_data)

#checking columns var types
data.frame(Column = names(credit_data), type = sapply(credit_data, class))
# count total rows
n <- nrow(credit_data)

# count number of default cases (where target variable = 1)
default_count <- sum(credit_data$SeriousDlqin2yrs == 1)

# calculate imbalance percentage
percentage <- (default_count / n)

print(default_count)
print(percentage * 100)

#checking missing values in all columns
missing_values <- data.frame(Column = names(credit_data), Missing =
as.vector(colSums(is.na(credit_data))),
Percent =  as.vector(colSums(is.na(credit_data))/nrow(credit_data)*100))

print(missing_values)

#we have two columns with missing data (Monthly_Income - 29731, Number_Of_Dependents - 3924)
#check if missing values in these columns are important or not.

#install.packages("dplyr")
library(dplyr)

# 1. Check if missing income is random or relates to other financial variables -
#the code below will calculate mean of each group where the income is missing vs income is not
missing

income_analysis <- credit_data %>%
  mutate(Income_Missing = is.na(MonthlyIncome)) %>%
  group_by(Income_Missing) %>%
  summarise(
    Count = n(),
    Default_Rate = mean(SeriousDlqin2yrs, na.rm = TRUE),
    Avg_DebtRatio = mean(DebtRatio, na.rm = TRUE),
    Avg_CreditLines = mean(NumberOfOpenCreditLinesAndLoans, na.rm = TRUE),
    Avg_RealEstateLoans = mean(NumberRealEstateLoansOrLines, na.rm = TRUE),
      Avg_Age = mean(age, na.rm = TRUE)
  )

print(" INCOME MISSINGNESS ANALYSIS ")
print(income_analysis, n = Inf, width = Inf)
```

```
#interpretation

# 2. Check if missing dependents relates to age or other factors
dependents_analysis <- credit_data %>%
  mutate(Dependents_Missing = is.na(NumberOfDependents)) %>%
  group_by(Dependents_Missing) %>%
  summarise(
    Count = n(),
    Default_Rate = mean(SeriousDlqin2yrs, na.rm = TRUE),
    Avg_Age = mean(age, na.rm = TRUE),
    Avg_Income = mean(MonthlyIncome, na.rm = TRUE)
  )

print(" DEPENDENTS MISSINGNESS ANALYSIS ")
print(dependents_analysis)

#checking duplicate rows
sum(duplicated(credit_data))
```

Step 2: Data Cleaning

```
# Loading required libraries
library(caret)
library(dplyr)

# 1. STRATIFIED SPLIT (FIXED)
set.seed(143)
trainIndex <- createDataPartition(credit_data$SeriousDlqin2yrs, p = 0.85, list = FALSE)
train.data <- credit_data[trainIndex, ]
test.data <- credit_data[-trainIndex, ]

cat("Training samples:", nrow(train.data), "\n")
cat("Test samples:", nrow(test.data), "\n")
cat("Training class distribution:", table(train.data$SeriousDlqin2yrs), "\n")

#average age for missing dependents is 62, so we can safely input as 0 and create binary flag
#to avoid data leakage first  create binary missing flag in training set and replace NA with 0
train.data$Dependents_Missing <- ifelse(is.na(train.data$NumberOfDependents), 1, 0)
train.data$NumberOfDependents[is.na(train.data$NumberOfDependents)] <- 0

#similar process for testing set
test.data$Dependents_Missing <- ifelse(is.na(test.data$NumberOfDependents), 1, 0)
test.data$NumberOfDependents[is.na(test.data$NumberOfDependents)] <- 0


#missing income rows have high debt ratio
#creating missing income flag for training and testing set
train.data$Income_Missing <- ifelse(is.na(train.data$MonthlyIncome), 1, 0)
test.data$Income_Missing <- ifelse(is.na(test.data$MonthlyIncome), 1, 0)

#creating median from training set and applies to both
median_income <- median(train.data$MonthlyIncome, na.rm = TRUE)
train.data$MonthlyIncome[is.na(train.data$MonthlyIncome)] <- median_income
test.data$MonthlyIncome[is.na(test.data$MonthlyIncome)] <- median_income

#fixing high debt ratio
#creating high debt ratio flag in training and testing set
train.data$highdebtflag <- ifelse(train.data$DebtRatio > 1, 1, 0)
test.data$highdebtflag <- ifelse(test.data$DebtRatio > 1, 1, 0)

#capping extreme values
```

```
cap_value <- quantile(train.data$DebtRatio, 0.99, na.rm = TRUE)
#apply to both set
train.data$DebtRatio <- pmin(train.data$DebtRatio, cap_value)
test.data$DebtRatio <- pmin(test.data$DebtRatio, cap_value)

#capping pastdue col
#creating list of 3 col
past_due_cols <- c(
  "NumberOfTime30.59DaysPastDueNotWorse",
  "NumberOfTime60.89DaysPastDueNotWorse",
  "NumberOfTimes90DaysLate"
)

# Apply to TRAIN
#looping through each col
for (col in past_due_cols) {
  train.data[[paste0(col, "_extreme")]] <- ifelse(train.data[[col]] > 10, 1, 0)  #creating
extreme value flag
  train.data[[col]] <- ifelse(train.data[[col]] > 10, 10, train.data[[col]])  #capping
original value at 10
}

# Apply SAME transformation to TEST
for (col in past_due_cols) {
  test.data[[paste0(col, "_extreme")]] <- ifelse(test.data[[col]] > 10, 1, 0)
  test.data[[col]] <- ifelse(test.data[[col]] > 10, 10, test.data[[col]])
}

# Fix unrealistic ages < 18 using training median
age_median_train <- median(train.data$age[train.data$age >= 18], na.rm = TRUE)
train.data$age[train.data$age < 18] <- age_median_train
test.data$age[test.data$age < 18] <- age_median_train

#capping revolving utilization of credit after creating flag
cap <- 1.5
# TRAIN
train.data$RevolvingUtilization_extreme <-
  ifelse(train.data$RevolvingUtilizationOfUnsecuredLines > cap_value, 1, 0)

train.data$RevolvingUtilizationOfUnsecuredLines <-
  ifelse(train.data$RevolvingUtilizationOfUnsecuredLines > cap_value,
         cap_value,
         train.data$RevolvingUtilizationOfUnsecuredLines)

# TEST
test.data$RevolvingUtilization_extreme <-
  ifelse(test.data$RevolvingUtilizationOfUnsecuredLines > cap_value, 1, 0)

test.data$RevolvingUtilizationOfUnsecuredLines <-
  ifelse(test.data$RevolvingUtilizationOfUnsecuredLines > cap_value,
         cap_value,
         test.data$RevolvingUtilizationOfUnsecuredLines)

#applying log transformation to income
train.data$MonthlyIncome_log <- log(train.data$MonthlyIncome + 1)
test.data$MonthlyIncome_log <- log(test.data$MonthlyIncome + 1)

#converting test and train data target variable
train.data$SeriousDlqin2yrs <- factor(train.data$SeriousDlqin2yrs, levels = c(0,1), labels =
c("NonDefault","Default"))
```

```
test.data$SeriousDlqin2yrs  <- factor(test.data$SeriousDlqin2yrs, levels = c(0,1), labels =
c("NonDefault","Default"))
```

Step 3: Main Model - LightGBM

```
library(lightgbm)
library(pROC)
library(dplyr)

# 1. Prepare data
predictor_cols <- setdiff(names(train.data), "SeriousDlqin2yrs")
X <- as.matrix(train.data[, predictor_cols])
y <- as.numeric(train.data$SeriousDlqin2yrs == "Default")
dtrain <- lgb.Dataset(data = X, label = y)

# Class imbalance handling
scale_pos_weight <- sum(y == 0) / sum(y == 1)

# 2. Define hyperparameters
params <- list(
  objective = "binary",
  metric = "auc",
  learning_rate = 0.02,
  num_leaves = 40,
  max_depth = -1,
  min_data_in_leaf = 40,
  feature_fraction = 0.9,
  bagging_fraction = 0.9,
  bagging_freq = 1,
  lambda_l1 = 0.0,
  lambda_l2 = 1.0,
  scale_pos_weight = scale_pos_weight
)

# 3. Run 5-fold cross-validation
set.seed(143)

cv_results <- lgb.cv(
  params = params,
  data = dtrain,
  nrounds = 3000,
  nfold = 5,
  stratified = TRUE,   #  for class imbalance
  early_stopping_rounds = 80,
  verbose = 1
)

# Best iteration (based on validation folds)
best_iter <- cv_results$best_iter
best_iter
final_model <- lgb.train(
  params = params,
  data = dtrain,
  nrounds = best_iter
)
x_test <- as.matrix(test.data[, predictor_cols])
y_test <- as.numeric(test.data$SeriousDlqin2yrs == "Default")

pred_test <- predict(final_model, x_test)
```

```
test_auc <- roc(y_test, pred_test)$auc
test_auc
cv_auc <- cv_results$best_score
test_auc <- roc(y_test, pred_test)$auc
cat(sprintf("CV AUC: %.4f, Test AUC: %.4f\n", cv_auc, test_auc))
```

Step 4: Accuracy, precision, recall and F1 Evaluation at different Thresholds

```
# Threshold metrics
thresholds <- seq(0.1, 0.9, by = 0.01)

results <- data.frame()

for (t in thresholds) {
  pred_class <- ifelse(pred_test >= t, 1, 0)

  TP <- sum(pred_class == 1 & y_test == 1)
  FP <- sum(pred_class == 1 & y_test == 0)
  FN <- sum(pred_class == 0 & y_test == 1)
  TN <- sum(pred_class == 0 & y_test == 0)

  Accuracy  <- (TP + TN) / (TP + FP + FN + TN)
  Precision <- TP / (TP + FP)
  Recall    <- TP / (TP + FN)
  F1        <- 2 * Precision * Recall / (Precision + Recall)

  results <- rbind(results, data.frame(
    Threshold = t,
    Accuracy  = round(Accuracy, 4),
    Precision = round(Precision, 4),
    Recall    = round(Recall, 4),
    F1        = round(F1, 4),
    TP = TP, FP = FP, FN = FN, TN = TN
  ))
}

print(results)

Step 5: ROC curve and Precision- Recall curve

# ROC Curve
roc_obj <- roc(y_test, pred_test)

plot(
  roc_obj,
  main = sprintf("ROC Curve (AUC = %.4f)", auc(roc_obj)),
  col = "blue",
  lwd = 2
)

abline(a = 0, b = 1, col = "red", lty = 2)

# Precision-Recall Curve
library(PRROC)

pr_obj <- pr.curve(
  scores.class0 = pred_test[y_test == 1],
  scores.class1 = pred_test[y_test == 0],
  curve = TRUE
)

plot(
  pr_obj$curve[, 1], pr_obj$curve[, 2],
  type = "l", col = "darkgreen", lwd = 2,
  xlab = "Recall", ylab = "Precision",
  main = sprintf("Precision-Recall Curve (AUC = %.4f)", pr_obj$auc.integral)
)

Step 4: Baseline Weighted logistic model
# Train weighted logistic regression model
weighted_model <- train(
  x = train.data[, c( "RevolvingUtilizationOfUnsecuredLines",
  "RevolvingUtilization_extreme",
  "DebtRatio", "debtratio_sq",
  "highdebtflag",
  "MonthlyIncome_log",
  "Income_Missing",
  "NumberOfDependents",
  "Dependents_Missing",
  "age",  "age_sq",
  "NumberOfOpenCreditLinesAndLoans",
  "NumberRealEstateLoansOrLines",
  "NumberOfTimes90DaysLate",
  "NumberOfTimes90DaysLate_extreme" )],
  y = train.data$SeriousDlqin2yrs,   #specifying target variables
  method = "glm",                    #logistic
  family = "binomial",
  weights = model_weights,    #applies calculated class weight
  trControl = trainControl(method = "cv", number = 5, classProbs = TRUE)   #5-fold cross-
validation, returns class probabilities
)


# Predict probabilities on test set - Generates probability scores for "Default" class on test
data
test.data$weighted_pred <- predict(weighted_model, newdata = test.data, type = "prob")[,
"Default"]
```

Step 5: Weighted Random Forest

```
weighted_rf <- randomForest(
  x = train.data[, predictor_cols],           # ALL FEATURES from train_final
  y = train.data$SeriousDlqin2yrs,
  ntree = 150,
  mtry = floor(sqrt(length(predictor_cols))),
  classwt = class_weights,
  importance = TRUE,
  strata = train.data$SeriousDlqin2yrs,
  sampsize = rep(min(table(train.data$SeriousDlqin2yrs)), 2)  # Balanced sampling
)
```

Step 6: Balanced Random Forest

```
balanced_rf <- ranger(
  formula = formula,
  data = train.data,                          # Using train_final
  num.trees = 150,
  mtry = floor(sqrt(length(predictor_cols))),
  probability = TRUE,
  replace = TRUE,
  sample.fraction = 0.7,
  strata = train.data$SeriousDlqin2yrs,
  classification = TRUE,
  importance = "impurity"
)
```

Step 7: Weighted Xgboost

```
scale_pos_weight <- sum(y_train == 0) / sum(y_train == 1)

params <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  max_depth = 6,
  eta = 0.1,
  scale_pos_weight = scale_pos_weight
)

set.seed(143)
xgb_model <- xgboost(
  data = x_train,
  label = y_train,
  params = params,
  nrounds = 100,
  verbose = 0
)
```