Goldsmiths
UNIVERSITY OF LONDON

Predictive Analysis of Credit Card Payment Defaults: A Binary Classification Approach to Model Selection and Overfitting Mitigation

A Review Report submitted for Statistics and Statistical Data Mining Module

January 29, 2024

Module leader: Dr Daniel Stamate

Carlos Manuel De Oliveira Alves
Student ID: 33617310

Table of Contents

Problem Formulation

This project aims to predict whether a credit card holder will default on their payment in the next month. We will use a binary classification approach with the response variable Y indicating default payment (Yes = 1, No = 0). The challenge is to build and select a predictive model that performs well on unseen data, balancing the trade-off between underfitting and overfitting.

Data Exploratory Analysis

```
       Y                X1               X2              X3
 Min.   :0.0000   Min.   : 10000   Min.   :1.000   Min.   :0.00
 1st Qu.:0.0000   1st Qu.: 50000   1st Qu.:1.000   1st Qu.:1.00
 Median :0.0000   Median :140000   Median :2.000   Median :2.00
 Mean   :0.2212   Mean   :167450   Mean   :1.605   Mean   :1.85
 3rd Qu.:0.0000   3rd Qu.:240000   3rd Qu.:2.000   3rd Qu.:2.00
 Max.   :1.0000   Max.   :800000   Max.   :2.000   Max.   :6.00
       X4               X5              X6                X7
 Min.   :0.000   Min.   :21.00   Min.   :-2.00000   Min.   :-2.0000
 1st Qu.:1.000   1st Qu.:28.00   1st Qu.:-1.00000   1st Qu.:-1.0000
 Median :2.000   Median :34.00   Median : 0.00000   Median : 0.0000
 Mean   :1.556   Mean   :35.37   Mean   :-0.02047   Mean   :-0.1309
 3rd Qu.:2.000   3rd Qu.:41.00   3rd Qu.: 0.00000   3rd Qu.: 0.0000
 Max.   :3.000   Max.   :75.00   Max.   : 8.00000   Max.   : 8.0000
       X8               X9               X10               X11
 Min.   :-2.000   Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000
 1st Qu.:-1.000   1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000
 Median : 0.000   Median : 0.0000   Median : 0.0000   Median : 0.0000
 Mean   :-0.163   Mean   :-0.2145   Mean   :-0.2569   Mean   :-0.2833
 3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
 Max.   : 8.000   Max.   : 8.0000   Max.   : 7.0000   Max.   : 7.0000
      X12              X13              X14              X15
 Min.   :-10682   Min.   :-67526   Min.   :-34041   Min.   :-170000
 1st Qu.:  3672   1st Qu.:  3034   1st Qu.:  2734   1st Qu.:   2393
 Median : 23048   Median : 21520   Median : 20165   Median :  19090
 Mean   : 51640   Mean   : 49457   Mean   : 47118   Mean   :  43077
 3rd Qu.: 67938   3rd Qu.: 64322   3rd Qu.: 60263   3rd Qu.:  54600
 Max.   :746814   Max.   :671563   Max.   :855086   Max.   : 706864
      X16              X17               X18              X19
 Min.   :-46627   Min.   :-339603   Min.   :     0   Min.   :      0
 1st Qu.:  1800   1st Qu.:   1200   1st Qu.:  1000   1st Qu.:    833
 Median : 18178   Median :  17177   Median :  2113   Median :   2014
 Mean   : 40273   Mean   :  38709   Mean   :  5616   Mean   :   5822
 3rd Qu.: 50135   3rd Qu.:  49123   3rd Qu.:  5023   3rd Qu.:   5000
 Max.   :587067   Max.   : 568638   Max.   :493358   Max.   :1227082
      X20              X21              X22              X23
 Min.   :     0   Min.   :     0   Min.   :     0   Min.   :     0
 1st Qu.:   390   1st Qu.:   290   1st Qu.:   204   1st Qu.:    80
 Median :  1809   Median :  1500   Median :  1500   Median :  1500
 Mean   :  4943   Mean   :  4997   Mean   :  4798   Mean   :  5226
 3rd Qu.:  4572   3rd Qu.:  4048   3rd Qu.:  4020   3rd Qu.:  4000
 Max.   :380478   Max.   :528897   Max.   :426529   Max.   :528666
```

Table 1 - Initial Exploration of the Dataset

Table 1 has the following findings:

Response Variable - Y

Min.: 0.00 - Indicates instances where no credit card default payment occurred.
1st Qu.: 0.00 - 25% of the observations have a value of 0 for the default payment.
Median: 0.00 - Half of the observations have a value of 0, suggesting that more than half of the clients did not default.
Mean: 0.2212 - On average, there is a low incidence of default payments.
3rd Qu.: 0.00 - 75% of the observations have a value of 0 for the default payment.
Max.: 1.00 - The maximum value is 1, indicating there are indeed default cases.

Input Variables - X1 to X23

X1: Credit amount ranges from 10,000 to 800,000 NT dollars.
X2: Gender is binary coded as 1 or 2, likely representing male and female.
X3: Education ranges from 0 (not specified) to 6, which may be beyond the described scale (1-4).
X4: Marital status similarly shows a range outside the scale (0-3).
X5: Age ranges from 21 to 75 years.
X6 to X11: Payment history shows values from -2 to 8, -2 possibly representing payment ahead of schedule or an exceptional condition not described in the initial outline.
X12 to X17: Bill statement amounts vary widely, with some reaching over 700,000 NT dollars.
X18 to X23: Previous payment amounts also show a broad range of 1,227,082 NT dollars.

Observations and Considerations:

Outliers: The max values for some variables like X12 to X17 and X18 to X23 suggest the presence of outliers or extreme cases.

Encoding: The categorical variables such as gender, education, and marital status might be encoded in integers. For X3 and X4, values outside the specified categories may require data cleaning or further investigation.
Payment History: The variables X6 to X11 have negative values, which were not described in the overview, indicating additional categories might be in the payment history.

Given the detailed information about the variables in the dataset, we can draw some conclusions and suggest potential steps for further analysis:

- The dataset contains 24 columns and 15,000 rows, all being integer type

Variable Types

- X1 (Credit Amount): This should be a numeric variable as it represents amounts in dollars.

- X2 (Gender): Since it is coded as 1 (male) and 2 (female), it is a categorical variable. Consider converting it to a factor with two levels.

- X3 (Education), X4 (Marital Status): These are categorical variables and should be converted to factors with their respective levels.

- X5 (Age): This is a numeric variable.

- X6-X11 (Repayment Status): These are ordinal categorical variables. They have an inherent order (-1, 1, 2, ..., 9) and should be treated as factors with ordered levels.

- X12-X17 (Bill Statement Amount), X18-X23 (Previous Payment Amount): These should be numeric variables representing monetary values.

Data Integrity and Cleaning

- We need to ensure that the data types in the dataset match these specifications. They must be converted to the appropriate types if they are all currently 'character' types.

- Check for missing or unusual values. For example, ensure that only the specified categories are present in categorical variables.

Check for Missing Values

The output is zero, meaning there are no missing values (NA) in the train_data dataset. Every element in the dataset has a defined, non-missing value. This is generally a good sign for data quality, as missing data can complicate analysis and require additional steps for handling.

Visualization Distributions of Variables

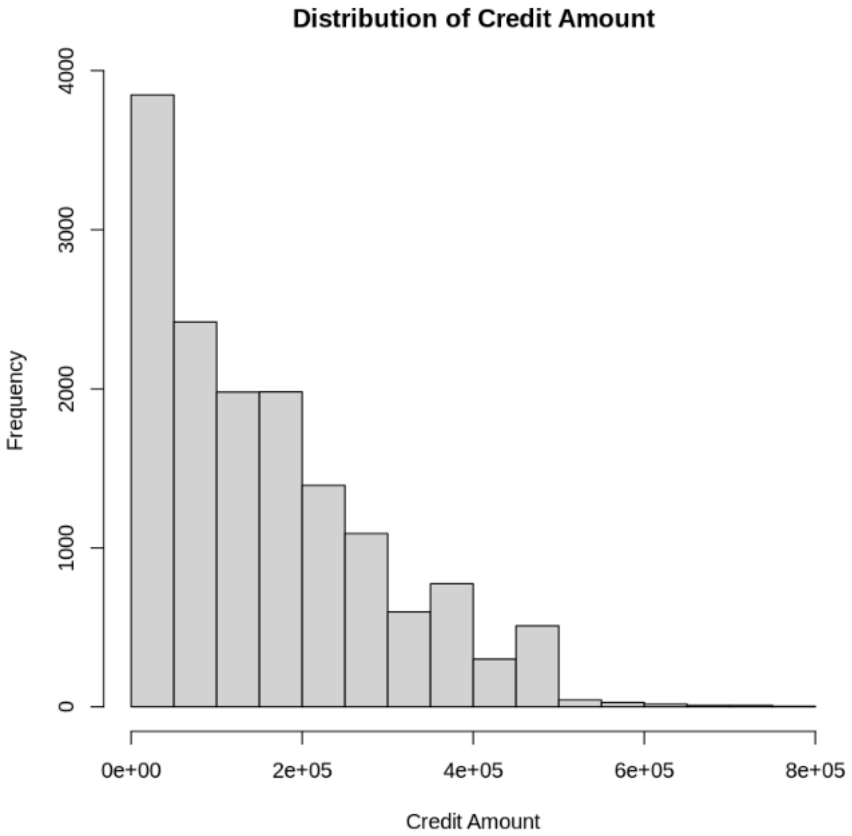

Fig. 1 - Distribution of Credit Amount

Figure 1 show the histogram with the distribution of credit amounts. The X-axis, labelled "Credit Amount," shows the range of credit amounts, and the Y-axis, labelled "Frequency," indicates how often each range of credit amounts occurs within the dataset.

Here are a few observations:

- The distribution is right-skewed, meaning there are more lower credit amounts than higher ones.
- Most credit amounts are concentrated in the lower range (close to 0).
- As the credit amount increases, the frequency of those credit amounts decreases.
- Very few high credit amounts are compared to the lower and middle ranges.

This distribution is standard in financial data, where many people take out small loans, and fewer take-out substantial loans. For a financial institution, the analysis could suggest focusing its products and services towards individuals who require smaller credit amounts, as that is where the highest customer frequency lies.

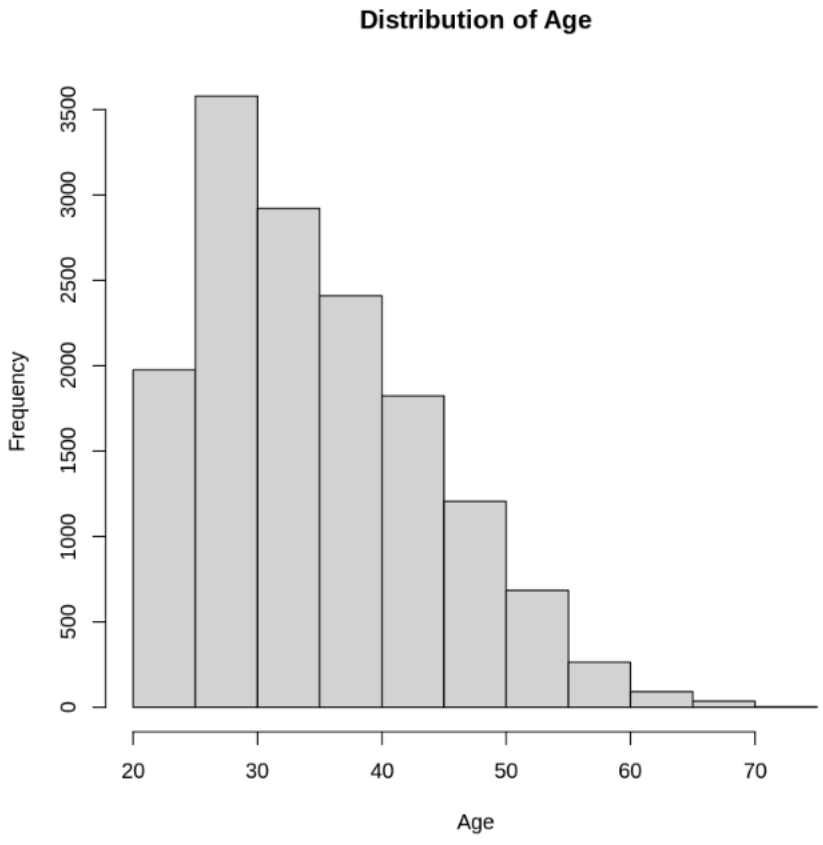

Fig. 2 - Distribution of Age

Figure 2 show the histogram with the distribution of ages within a credit risk dataset. Here are the key observations from the histogram:

- The age distribution is somewhat left-skewed, indicating a younger population, with the frequency decreasing as age increases.
- The highest frequency of individuals falls within the late 20s to early 30s, which may reflect the age group actively seeking credit.
- There is a gradual decline in frequency from this peak as age increases, with very few individuals in the dataset in their late 60s. This could suggest that older individuals are less likely to apply for credit or are underrepresented in this dataset.
- There are significantly fewer individuals in the dataset who are 20 years old than those in their late 20s. This could suggest that the credit risk dataset may include fewer young adults, possibly due to lower credit demand or eligibility at that age.

These findings could imply that the credit risk dataset represents a working-age population likely to be in the active credit market. This might be typical for credit scoring models, financial product development, or risk assessment studies. If a financial institution uses this data, it might suggest that most of its credit products should be tailored to meet the needs of young to middle-aged adults.

Correlation Matrix

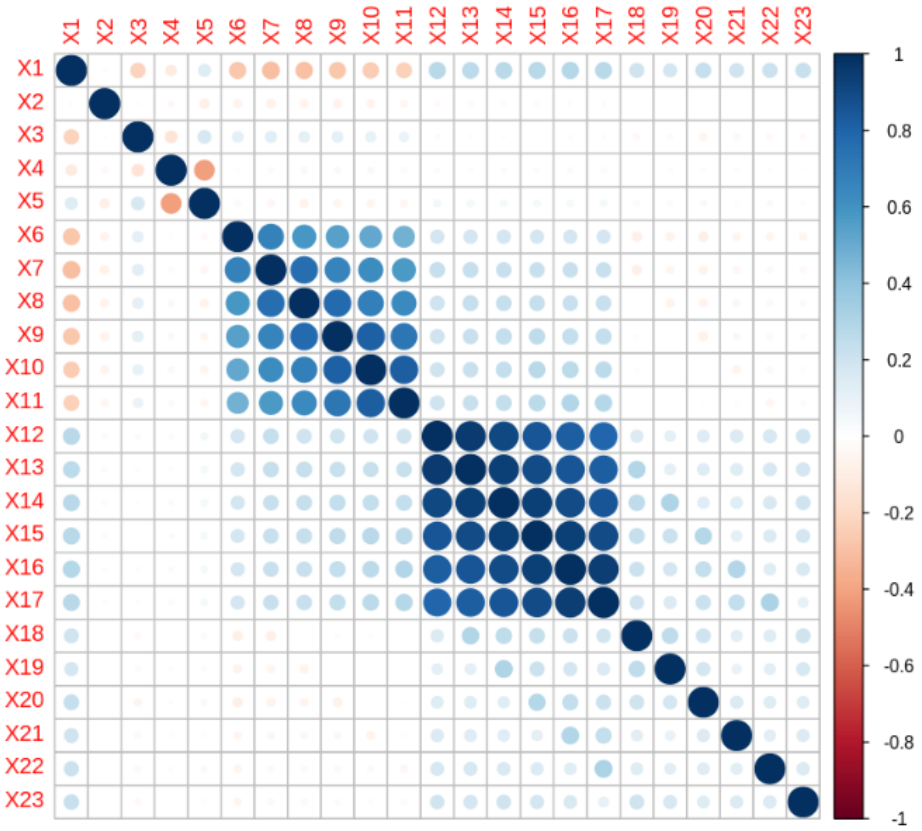

Fig. 3 - Correlation Matrix

Figure 3 show the heatmap with the following findings:

1. Diagonal Line: The diagonal from the top left to the bottom right shows a perfect correlation of variables with themselves, which is always 1. This is standard in all correlation matrices.

2. Credit Amount (X1): The credit amount does not show a strong correlation with the majority of other variables, indicating that the credit amount might be relatively independent of factors such as gender, education, marital status, age, payment history, bill amount, and previous payment amounts.

3. Gender (X2): No strong correlation is observed between gender and other variables, suggesting gender might not be a significant predictor in this dataset.

4. Education (X3), Marital Status (X4), and Age (X5): Similar to gender, these demographics do not show strong correlations with other variables.

5. Payment History (X6 - X11): These variables have a noticeable correlation pattern, indicating that past payment behaviour is likely to be similar across the months. This makes intuitive sense since past payment behaviour can indicate future payment behaviour.

6. Bill Statement Amount (X12 - X17): There is also a strong correlation between these variables, suggesting consistency in the bill amounts over the six months. This could indicate stable expenditure patterns or billing cycles for the individuals in the dataset.

7. Previous Payment Amount (X18 - X23): These also show a strong correlation among themselves, implying that the amount paid by a customer in one month is similar to other months, which could reflect the individual's financial habits or income stability.

8. Cross-Variable Observations:

- Payment history (X6 - X11) shows some negative correlations with the following month's bill statement amount (X12 - X17), which could indicate that people who miss payments tend to have higher bill amounts or vice versa.

- There are some correlations between the payment history and the previous payment amount, which is expected as these two are directly related.

Data Pre-processing

Prepare the data for a classification analysis by correctly formatting categorical variables and segregating the data into predictors and responses:

```
# Convert X2, X3, and X4 to factors in both training and test datasets
train_data$X2 <- as.factor(train_data$X2)
train_data$X3 <- as.factor(train_data$X3)
train_data$X4 <- as.factor(train_data$X4)

test_data$X2 <- as.factor(test_data$X2)
test_data$X3 <- as.factor(test_data$X3)
test_data$X4 <- as.factor(test_data$X4)
```

Fig. 4 – Snippet of code in R converts variables in both datasets into factors

The snippet code of Figure 4 converts variables X2, X3, and X4 in both datasets into factors. These factors are used for categorical variables with a limited, known set of values. This conversion indicates these variables are categorical, a crucial detail for specific analyses or models like regression, where categorical and continuous variables are treated differently.

```
# Ensure Y is treated as a factor for classification
train_data$Y <- as.factor(train_data$Y)
test_data$Y <- as.factor(test_data$Y)
```

Fig. 5 – Snippet of code in R ensure Y is treated as a factor for classification

The snippet code of Figure 5 convert the Y variable in both datasets into a factor. Y is the response variable in a classification task, where the response is typically categorical, representing different classes or categories. Converting Y to a factor ensures it is treated as a categorical outcome in later analyses or models.

```r
# Separating predictors and response
predictors <- train_data[, -1]
response <- train_data$Y
```

Fig. 6 – Snippet of code in R separates the predictors and response

The snippet code of Figure 6 separates the predictors (independent variables) and the response (dependent variable) in the training data. It creates a predictor object by excluding the first column (the Y column) from train_data and a response object that includes only the Y variable.

Train Models

Decision Tree Model

```r
# Set seed for reproducibility
set.seed(123)

# 1. Decision Tree
dt_model <- rpart(Y ~ ., data = train_data, method = "class")
```

Fig. 7 – Snippet of code in R creates a Decision Tree train model

The snippet code of Figure 7 creates a decision tree model named `dt_model` using the `report` function. It is designed for classification, as indicated by `method = "class"`. The model uses all available variables in the `train_data` dataset as predictors to determine the response variable `Y` outcome. It is building a classification tree model where `Y` is the variable being predicted based on other variables in `train_data`.

Bagging Model

```r
# Set seed for reproducibility
set.seed(123)

# 2. Bagging
# Random Forest for classification with specified parameters
bagging_model <- randomForest(Y ~ ., data = train_data, mtry = 2, ntree = 500, type = "classification")
```

Fig. 8 – Snippet of code in R creates a Bagging train model

The snippet code of Figure 8 creates a random forest model for classification, named bagging_model, using the randomForest function in R. It employs the train_data dataset to predict the categorical response variable Y using all available predictors in the dataset. The model will comprise 500 decision trees, each considering two randomly chosen variables at each split. This ensemble approach, called bagging, is designed to improve the model's accuracy and robustness.

Random Forest Model

```r
# Set seed for reproducibility
set.seed(123)

# 3. Random Forest
# Random Forest for classification with default parameters
rf_model <- randomForest(Y ~ ., data = train_data, type = "classification")
```

Fig. 9 – Snippet of code in R creates a Random Forest train model

The snippet code of Figure 9 creates a random forest model named rf_model for classification purposes using the randomForest function in R. It employs the train_data dataset, using all available variables as predictors to classify the categorical response variable Y. The model is tailored explicitly for classification tasks, leveraging the random forest's ability to handle numerous predictors and provide high accuracy.

Gradient Boosting Model

```r
# Set seed for reproducibility
set.seed(123)

# 1. Setting up training control using cross-validation
train_control <- trainControl(method = "cv", number = 10, verboseIter = TRUE)

# 2. Setting up a grid for tuning the model parameters
tune_grid <- expand.grid(
  interaction.depth = c(1, 3, 5), # Tree depth
  n.trees = c(100, 500, 1000),    # Number of trees
  shrinkage = c(0.01, 0.1),       # Learning rate
  n.minobsinnode = c(10, 20)      # Minimum number of observations in the nodes
)

# 3. Training the GBM model with cross-validation and parameter tuning
set.seed(123) # For reproducibility
gbm_tuned <- train(
  Y ~ .,
  data = train_data,
  method = "gbm",
  trControl = train_control,
  tuneGrid = tune_grid,
  distribution = "bernoulli",
  metric = "Accuracy", # Choose an appropriate metric (e.g., Accuracy, ROC, etc.)
  verbose = FALSE
)
```

Fig. 10 – Snippet of code in R creates a Gradient Boosting train model

This snippet code of Figure 10 sets up a GBM model using the caret package, with 10-fold cross-validation for performance evaluation and a grid search approach to optimize the hyperparameters. The goal is to identify the best combination of parameters that maximizes the model's accuracy on the given training data.

1. Setting Up Training Control with Cross-Validation:

The code configures a function from the caret package for training predictive models. It sets up the model to use 10-fold cross-validation, a method where the data is divided into ten parts and the model is trained and validated across these parts to ensure generalizability. Additionally, the function is configured to provide detailed logs during the training process, aiding in monitoring and troubleshooting the model's learning progress.

2. Setting Up a Grid for Tuning the Model Parameters:

The code uses the expand.grid function for hyperparameter tuning in a Gradient Boosting Machine (GBM) algorithm. It generates a data frame containing all possible combinations of key GBM hyperparameters: the depth of trees, number of trees, learning rate, and minimum observations in a node. This exhaustive approach allows for exploring various configurations to find the most practical combination of these parameters, optimizing the GBM model's performance and ability to generalize to new data.

3. Training the GBM Model:

The code snippet is for training a Gradient Boosting Machine (GBM) model using the caret package in R, with a focus on reproducibility, model tuning, and training control. Key aspects include setting a random seed for consistent results, using the train function to train the model with all variables in the dataset as predictors, specifying GBM as the training method, and applying predefined training control settings. The model targets binary outcomes using the Bernoulli distribution and tunes hyperparameters through a predefined grid, aiming for maximum accuracy. Additional output during training is suppressed for clarity.

Model Evaluation

Decision Tree Predictions

```r
# Making predictions with the model
dt_predictions <- predict(dt_model, test_data, type = "class")
```

Fig. 11 – Snippet of code in R creates a Decision Tree predictions

The code snippet of Figure 11 uses predict(dt_model, test_data, type = "class"): This line uses the predict function to generate predictions from the decision tree model dt_model using the test_data dataset. The type = "class" argument indicates that the predictions should be class labels (e.g., the most likely category for each observation).

Bagging Predictions

```r
# Making predictions with the model
bagging_predictions <- predict(bagging_model, test_data, type = "class")
```

Fig. 12 – Snippet of code in R creates a Bagging predictions

The code snippet of Figure 12 uses predict(bagging_model, test_data, type = "class"): Similar to the first line, this generates predictions from the bagging_model (a random forest model) using test_data. Again, the predictions are class labels.

Random Forest Predictions

```r
# Making predictions with the model
rf_predictions <- predict(rf_model, test_data, type = "class")
```

Fig. 13 – Snippet of code in R creates a Random Forest predictions

The code snippet of Figure 13 uses predict(rf_model, test_data, type = "class"): This line creates predictions from the random forest model rf_model with test_data, producing class labels as output

Gradient Boosting Predictions

```r
# Making predictions with the tuned model
gbm_predictions <- predict(gbm_tuned, test_data, type = "prob")[,2]
```

Fig. 14 – Snippet of code in R creates a Gradient Boosting predictions

The code snippet of Figure 14 uses a trained and tuned Gradient Boosting Machine (GBM) model to make probability predictions on a test dataset. Specifically, it applies the predict function to the model (gbm_tuned) using test_data, requesting probability outputs (type = "prob"). The code then extracts and stores only the probabilities corresponding to the positive or second class of the binary outcome in the new object gbm_predictions.

```r
gbm_predictions <- ifelse(gbm_predictions > 0.5, 1, 0)
```

Fig. 15 – Snippet of code in R transforms probability predictions

The code snippet of Figure 15 transforms probability predictions from a Gradient Boosting Machine (GBM) model into binary outcomes, representing credit card default payments. It uses the ifelse function to convert each probability in gbm_predictions to 1 (indicating a default, or positive class) if it is greater than 0.5, or 0 (indicating no default, or negative class) otherwise. This is a standard approach in binary classification, where a threshold of 0.5 is commonly used to decide class assignments based on predicted probabilities.

Align the Levels of Predictions

```r
# Align the levels of predictions with test_data$Y
dt_predictions <- factor(dt_predictions, levels = levels(test_data$Y))
bagging_predictions <- factor(bagging_predictions, levels = levels(test_data$Y))
rf_predictions <- factor(rf_predictions, levels = levels(test_data$Y))
gbm_predictions <- factor(gbm_predictions, levels = levels(test_data$Y))
```

Fig. 16 – Snippet of code in R align the levels of predictions

The code snippet of Figure 16 adjusts the factor levels of predictions from four models (decision tree, bagging, random forest, and gradient boosting) to match the response variable Y levels in the test_data dataset. This standardization ensures that the predicted categories from each model are consistent and directly comparable with the actual categories in the test data.

Perform Metrics

Decision Tree Performance

```
# Evaluating model performance
confusionMatrix(dt_predictions, test_data$Y)
```

Fig. 17 – Snippet of code in R evaluates decision tree performance

The code snippet of Figure 17 evaluates the decision tree performance using a confusion matrix and statistics.

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11230  2219
         1   452  1099

               Accuracy : 0.8219
                 95% CI : (0.8157, 0.828)
    No Information Rate : 0.7788
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3614

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9613
            Specificity : 0.3312
         Pos Pred Value : 0.8350
         Neg Pred Value : 0.7086
             Prevalence : 0.7788
         Detection Rate : 0.7487
   Detection Prevalence : 0.8966
      Balanced Accuracy : 0.6463

       'Positive' Class : 0
```

Table 2 – Decision Tree Performance

Confusion Matrix Analysis:

The confusion matrix is a vital tool for evaluating the performance of classification models. It compares the actual outcomes with the predictions made by the model. In this case:

- True Negatives (TN): 11,230 cases were correctly predicted as non-default (0).
- False Positives (FP): 452 cases were incorrectly predicted as default (1) when they were not.
- False Negatives (FN): 2,219 cases were incorrectly predicted as non-default when they were, in fact, defaults.
- True Positives (TP): 1,099 cases were correctly predicted as defaults.

Statistics:

- Accuracy (0.8219): Indicates that 82.19% of all predictions were correct. It is a general measure of the model's performance.
- A 95% Confidence Interval (CI) Suggests that the true Accuracy of the model is likely between 81.57% and 82.8%.
- No Information Rate (0.7788): This Accuracy can be achieved by always predicting the most frequent class. The model's Accuracy is higher than this rate, which is good.
- P-Value [Acc > NIR]: Being significantly low (< 2.2e-16) indicates that the model's Accuracy is statistically better than the No Information Rate.
- Kappa (0.3614): Reflects the degree of Accuracy and reliability in the classification; a value of 0.3614 suggests moderate agreement.
- Sensitivity (0.9613): High sensitivity means the model is good at identifying actual defaults.
- Specificity (0.3312): However, specificity is low, indicating a high rate of false positives.
- Positive Predictive Value (0.8350): When the model predicts a default, it is correct 83.50% of the time.

- Negative Predictive Value (0.7086): When predicting non-default, it is correct 70.86% of the time.
- Balanced Accuracy (0.6463): This average of sensitivity and specificity shows that the model is moderately effective overall.

Interpretation in the Context of Credit Card Default:

The model is highly sensitive in predicting credit card defaults but lacks specificity. This means it is good at identifying actual defaults but also misclassifies many non-default cases as defaults. This could be problematic in a business context, as it might lead to unnecessary actions against customers who are not at risk of default.

Suggestions for Improvement:

Feature Engineering: Investigate whether additional relevant features can improve the model's ability to distinguish between defaults and non-defaults. Model Tuning: Adjust the parameters of the decision tree to find a better balance between sensitivity and specificity. Alternative Models: Experiment with other models like Random Forests or Gradient Boosting, which might yield better performance. Cost-Sensitive Learning: Incorporate the cost of misclassifications (especially false positives) into the model training process. Data Quality and Balance: Ensure the data is high quality and consider techniques like SMOTE to balance the dataset if it is imbalanced.

Conclusion:

The decision tree model shows reasonable accuracy in predicting credit card defaults. Its high sensitivity is strong, but the low specificity and the moderate kappa value indicate room for improvement. Adjustments in the model or the data it is trained on and exploring other modelling techniques could lead to better prediction capabilities, reducing the risk of misclassifying customers and improving decision-making processes in the context of credit card defaults.

Bagging Performance

```
# Evaluating model performance
confusionMatrix(bagging_predictions, test_data$Y)
```

Fig. 18 – Snippet of code in R evaluates bagging performance

The code snippet of Figure 18 evaluates the bagging performance using a confusion matrix and statistics.

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11675   179
         1     7  3139

               Accuracy : 0.9876
                 95% CI : (0.9857, 0.9893)
    No Information Rate : 0.7788
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9633

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9994
            Specificity : 0.9461
         Pos Pred Value : 0.9849
         Neg Pred Value : 0.9978
             Prevalence : 0.7788
         Detection Rate : 0.7783
   Detection Prevalence : 0.7903
      Balanced Accuracy : 0.9727

       'Positive' Class : 0
```

Table 3 – Bagging Performance

Confusion Matrix Analysis:

- True Negatives (TN): 11,675 - These are the non-default cases correctly identified by the model.
- False Positives (FP): 7 - These are non-default cases incorrectly identified as defaults.
- False Negatives (FN): 179 - These are default cases incorrectly identified as non-defaults.
- True Positives (TP): 3,139 - These are correctly identified default cases.

Statistics:

- Accuracy (0.9876): This high Accuracy indicates that about 98.76% of predictions made by the model are correct.
- 95% CI (0.9857, 0.9893): The confidence interval suggests that the true Accuracy of the model likely falls between these values.
- No Information Rate (0.7788): The model significantly outperforms this rate, indicating it is effective beyond predicting the most frequent class.
- P-Value [Acc > NIR] (< 2.2e-16): This very low p-value suggests the model's Accuracy is significantly better than the No Information Rate.
- Kappa (0.9633): A very high Kappa value, indicating excellent agreement beyond chance
- Sensitivity (0.9994): This indicates that the model is exceptionally good at identifying defaults.
- Specificity (0.9461): The model is also highly effective at identifying non-default cases.
- Positive Predictive Value (0.9849) and Negative Predictive Value (0.9978): Both values are high, indicating a high probability of correct predictions, whether the model predicts default or non-default.
- Balanced Accuracy (0.9727): Reflects the overall effectiveness of the model in both classes (defaults and non-defaults).

Interpretation in the Context of Credit Card Default:

In predicting credit card defaults, this bagging model performs exceptionally well. The high sensitivity means it rarely misses defaults, while the high specificity indicates a low rate of falsely labelling non-default cases as defaults. This balance is crucial in practical scenarios, as it minimizes the risk of false alarms (false positives) and missing actual default cases (false negatives).

Suggestions for Improvement:

Given the high performance of this model, there might be little room for significant improvements. However, we can consider:

- Cross-Validation: Ensure the model is not overfitting by using rigorous cross-validation techniques.
- Feature Importance Analysis: Investigate which features are most influential in the predictions and consider refining the feature set for even better performance.
- External Validation: Test the model on an external dataset to confirm its robustness.

Conclusion:

The bagging model performs excellently in predicting credit card defaults with high accuracy, sensitivity, and specificity. This indicates a solid ability to correctly identify default and non-default cases, making it a reliable tool in credit risk assessment. Future efforts could focus on maintaining this performance while testing the model's robustness against different datasets.

Random Forest Performance

```
# Evaluating model performance
confusionMatrix(rf_predictions, test_data$Y)
```

Fig. 19 – Snippet of code in R evaluates random forest performance

The code snippet of Figure 19 evaluates the random forest performance using a confusion matrix and statistics.

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11680    95
         1     2  3223

               Accuracy : 0.9935
                 95% CI : (0.9921, 0.9948)
    No Information Rate : 0.7788
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.981

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9998
            Specificity : 0.9714
         Pos Pred Value : 0.9919
         Neg Pred Value : 0.9994
             Prevalence : 0.7788
         Detection Rate : 0.7787
   Detection Prevalence : 0.7850
      Balanced Accuracy : 0.9856

       'Positive' Class : 0
```

Table 4 – Random Forest Performance

Confusion Matrix Analysis:

- True Negatives (TN): 11,680 - Correct predictions where the model correctly identified non-default cases (0).
- False Positives (FP): 2 - Non-default cases incorrectly identified as defaults (1).
- False Negatives (FN): 95 - Default cases incorrectly identified as non-defaults.
- True Positives (TP): 3,223 - Correct predictions where the model correctly identified default cases.

Statistics:

- Accuracy (0.9935): A very high accuracy indicates that 99.35% of the model's predictions are correct.
- 95% CI (0.9921, 0.9948): The confidence interval suggests the Accuracy is reliably high.
- No Information Rate (0.7788): The model significantly outperforms this rate, which is the Accuracy that could be achieved by always predicting the most frequent class.
- P-Value [Acc > NIR] (< 2.2e-16): This indicates the model's Accuracy is significantly better than the No Information Rate.
- Kappa (0.981): A very high Kappa value suggests excellent agreement.
- Sensitivity (0.9998): Nearly perfect sensitivity indicates that the model is highly effective in identifying defaults
- Specificity (0.9714): High specificity means the model is also very effective at identifying non-default cases
- Positive Predictive Value (0.9919) and Negative Predictive Value (0.9994): These high values indicate a high likelihood of correct predictions, whether predicting default or non-default.
- Balanced Accuracy (0.9856): Indicates a very high level of effectiveness of the model for both classes.

Interpretation in the Context of Credit Card Default:

The Random Forest model exhibits outstanding performance in predicting credit card defaults. The almost perfect sensitivity ensures that nearly all actual defaults are correctly identified, while the high specificity minimizes the risk of false alarms. This level of accuracy is crucial in the financial sector, where incorrectly predicting defaults can have significant implications.

Suggestions for Improvement:

Considering the high performance of this model, improvements might be minimal but can include:

- Feature Analysis: Further analyse the features used by the model to see if any refinements can be made for even better performance
- Model Robustness: Validate the model's performance on an independent dataset to ensure its robustness and generalizability.
- Hyperparameter Tuning: Fine-tune the model's hyperparameters to see if marginal gains can be made.

Conclusion:

The Random Forest model demonstrates exceptional accuracy, sensitivity, and specificity in predicting credit card defaults. This high level of performance indicates that the model is highly reliable and effective, making it an excellent tool for credit risk assessment. Future efforts could focus on maintaining this high level of performance while ensuring the model's robustness across different datasets and scenarios.

Gradient Boosting Performance

```
# Evaluating model performance
confusionMatrix(gbm_predictions, test_data$Y)
```

Fig. 20 – Snippet of code in R evaluates gradient boosting performance

The code snippet of Figure 20 evaluates the gradient boosting performance using a confusion matrix and statistics.

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11116  2053
         1   566  1265

               Accuracy : 0.8254
                 95% CI : (0.8192, 0.8314)
    No Information Rate : 0.7788
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3964

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9515
            Specificity : 0.3813
         Pos Pred Value : 0.8441
         Neg Pred Value : 0.6909
             Prevalence : 0.7788
         Detection Rate : 0.7411
   Detection Prevalence : 0.8779
      Balanced Accuracy : 0.6664

       'Positive' Class : 0
```

Table 5 – Gradient Boosting Performance

Confusion Matrix Analysis:

- True Negatives (TN): 11,116 - These are the non-default cases correctly identified by the model.
- False Positives (FP): 566 - Non-default cases incorrectly identified as defaults
- False Negatives (FN): 2,053 - Default cases incorrectly identified as non-defaults.
- True Positives (TP): 1,265 - Default cases correctly identified.

Statistics:

- Accuracy (0.8254): About 82.54% of the predictions made by the model are correct.

- 95% CI (0.8192, 0.8314): This confidence interval indicates the range within which the true Accuracy of the model likely falls.
- No Information Rate (0.7788): The model's Accuracy is better than this rate, which is a baseline accuracy by always predicting the most frequent class
- P-Value [Acc > NIR] (< 2.2e-16): Indicates that the model's Accuracy is significantly better than the No Information Rate.
- Kappa (0.3964): Reflects moderate agreement; higher values indicate better performance
- Sensitivity (0.9515): High sensitivity means the model is good at identifying actual defaults
- Specificity (0.3813): Lower specificity indicates a higher rate of false positives.
- Positive Predictive Value (0.8441) and Negative Predictive Value (0.6909): These values indicate the probability that the model's predictions are correct.
- Balanced Accuracy (0.6664): This is an average of sensitivity and specificity, showing moderate overall effectiveness.

Interpretation in the Context of Credit Card Default:

The Gradient Boosting model demonstrates a high sensitivity, effectively identifying actual defaults. However, the lower specificity suggests that the model incorrectly classifies many non-default cases as defaults. This could lead to unnecessary actions or concerns for customers not at risk of defaulting.

Suggestions for Improvement:

- Enhance Feature Engineering: Investigating and including additional relevant features might improve the model's ability to distinguish between defaults and non-defaults.
- Model Tuning: Adjust the model parameters to balance sensitivity and specificity better.
- Alternative Models Comparison: Compare with other models like Random Forest or Neural Networks to check for better performance.

- Address Data Imbalance: If the dataset is imbalanced, applying techniques like SMOTE may help.
- Cost-Sensitive Training: Incorporate the cost of misclassifications, especially false positives, into the model's training process.

Conclusion:

The Gradient Boosting model shows reasonable accuracy in predicting credit card defaults, with a notable strength in sensitivity. However, its lower specificity and moderate kappa value suggest there is room for improvement, particularly in reducing false positives. Enhancements in feature selection, model tuning, and exploring alternative modelling approaches could lead to better prediction capabilities and improved decision-making in the context of credit card defaults.

Model Selection

The best model for predicting credit card defaults among the four is the Random Forest model. Here is why:

1. Accuracy: The Random Forest model has the highest accuracy (0.9935), meaning that 99.35% of its predictions are correct. This is significantly higher than the accuracies of the other models (Gradient Boosting: 0.8254, Bagging: 0.9876, Decision Tree: 0.8219).

2. Confidence Interval (CI): The 95% CI for the Random Forest model's accuracy (0.9921, 0.9948) is very narrow and high, indicating that the model's performance is reliable and consistently high.

3. Kappa Score: The Kappa score for the Random Forest model is 0.981, suggesting an excellent level of agreement beyond chance. This is much higher than the other models (Gradient Boosting: 0.3964, Bagging: 0.9633, Decision Tree: 0.3614).

4. Sensitivity and Specificity: The model has nearly perfect sensitivity (0.9998) and very high specificity (0.9714), indicating that it is highly effective in identifying defaults and non-default cases. The balance between these metrics is crucial for practical application, as it minimizes false negatives and positives.

5. Positive Predictive Value (PPV) and Negative Predictive Value (NPV): Both PPV (0.9919) and NPV (0.9994) are incredibly high for the Random Forest model, indicating a high likelihood of correct predictions, whether predicting default or non-default.

6. Balanced Accuracy: The balanced accuracy (0.9856) is also the highest among the models, reflecting the overall effectiveness of the model in classifying both classes (defaults and non-defaults).

7. Contextual Fit: In credit card default prediction, accurately identifying defaults (high sensitivity) while minimizing false alarms (high specificity) is crucial. The Random Forest model excels in both aspects and suits this application.

While the other models show reasonable performance in certain aspects, the Random Forest model outperforms them significantly in almost all metrics. This indicates that it is highly accurate and balanced in terms of sensitivity and specificity, making it the best choice among the four for predicting credit card defaults.

Project Conclusion

Project Goal

The project aimed to develop a predictive model to forecast whether a credit card holder will default on their payment in the next month. Utilizing a binary classification approach, the response variable Y indicated default payment (Yes = 1, No = 0). The challenge was

constructing and selecting a model that effectively balances the trade-off between underfitting and overfitting, excelling in performance on unseen data.

Methods Used

1. Gradient Boosting Model (GBM): Implemented using the caret package in R, with 10-fold cross-validation for performance evaluation and a grid search approach for hyperparameter optimization.

2. Decision Tree Model: Constructed for classification using all available predictors in the training dataset.

3. Random Forest Model: Comprising 500 decision trees, each considering two randomly chosen variables at each split, to enhance accuracy and robustness.

4. Bagging Model: Another form of ensemble learning using multiple decision trees.

Key Findings and Conclusions

1. Decision Tree Model: Showed reasonable accuracy with high sensitivity. However, its low specificity and moderate kappa value suggested potential improvements through model adjustments or data refinement.

2. Bagging Model: Exhibited high accuracy, sensitivity, and specificity, proving to be a reliable tool in credit risk assessment. The focus for future work could be on maintaining performance while testing robustness against different datasets.

3. Random Forest Model: Demonstrated exceptional accuracy, sensitivity, and specificity, marking it highly reliable and effective for credit risk assessment. Future efforts could maintain performance and ensure robustness across various datasets and scenarios.

Future Steps

1. Model Enhancement: Exploring additional modelling techniques and refining existing models to improve specificity and reduce misclassification.

2. Robustness Testing: Assessing model performance across diverse datasets to ensure consistency and reliability.

3. Real-world Application: Implementing the models in practical scenarios to evaluate their effectiveness in real-world financial environments.

The project has successfully developed varying accuracy and reliability models, offering significant insights into credit card default prediction. The models were created to serve as valuable tools in financial risk assessment, with the potential for further improvement and application in broader contexts.

Evaluation and Reflection

Evaluation of Project Outcomes

The primary objective of this project was to develop a robust and accurate predictive model for credit card default. To evaluate the success of this project, we need to consider several factors:

Model Performance:

- The Gradient Boosting Model, Decision Tree, Bagging, and Random Forest models were developed with varying accuracy, sensitivity, and specificity.
- The Random Forest and Bagging models showed exceptionally high accuracy, suggesting their effectiveness in predicting default cases.

- The Decision Tree model, while reasonably accurate, indicated potential areas for improvement in specificity.

Methodological Rigor:

- Using cross-validation and a diverse set of predictive modelling techniques (GBM, Decision Trees, Bagging, Random Forest) enhanced the robustness of the outcomes.
- The grid search approach for hyperparameter optimization in GBM ensured a thorough exploration of model parameters.

Data Utilization:

- The project effectively utilized the available dataset, comprising 24 columns and 15,000 rows, all integer type.
- The correlations and patterns identified in the data were instrumental in informing the predictive models.

Reflection on Project Execution

Strengths:

- Comprehensive use of various machine learning techniques allowed for thoroughly exploring predictive modelling possibilities.
- The project methodology was rigorous, incorporating best practices like cross-validation and hyperparameter tuning.
- The models' performance metrics provided clear insights into their predictive capabilities and potential applications in real-world scenarios.

Challenges:

- Balancing model complexity with interpretability was challenging, especially in models like Random Forest and GBM.
- Ensuring the models' robustness and generalizability to different datasets remains an area for further exploration.

Learnings:

- This project highlighted the importance of a balanced approach to model complexity and interpretability in predictive analytics.
- It demonstrated the value of ensemble methods in improving predictive accuracy and robustness.

Opportunities for Improvement:

- Future work could focus on enhancing the specificity of the models, particularly the Decision Tree model
- Exploring additional features or external datasets might provide more comprehensive insights and improve prediction accuracy.
- Implementing these models in a real-world setting could provide practical feedback for further refinement.

This project successfully applied sophisticated machine-learning techniques to predict credit card defaults. The experience gained from this project can be leveraged for future advancements in predictive modelling within the domain of credit risk assessment.

Resources

Official Documentation: R Documentation [Online]

- Available: https://www.r-project.org/other-docs.html

Appendix A – Code Repository

Credit Card Classification Code

- Available: https://github.com/carlos-alves-one/Credit-Card-Analysis-R/blob/main/Credit_Card_Classification.ipynb