Credit Card Default Prediction - By Jeri

2024-01-20

Introduction

Credit card default prediction is an important task in the financial industry, helping institutions assess and manage potential risks associated with lending. In this coursework, I will delve into a credit risk assessment using a real-world credit risk dataset. The dataset provides information on various factors with the main goal of predicting whether a credit card holder will default on payment. The dataset is divided into a training dataset (creditdefault_train.csv) and a test dataset (creditdefault_test.csv). This analysis is not only an exploration into the performance of various classification algorithms but also an opportunity to understand the key factors influencing credit card default. The ability to predict default accurately is crucial for financial institutions to make informed decisions and mitigate potential risks. I will be working and submitting on my own and will aim to address the challenges of model underfitting and overfitting.

Problem Formulation

This document presents a detailed analysis of the credit default dataset with the aim of predicting credit card default based on 23 input variables. The response variable, denoted as Y, represents whether a credit card holder will default or not, with '1' indicating default and '0' denoting no default. The dataset includes information on key factors such as the amount of credit given, gender, education level, marital status, age, history of past payments, amount of bill statements, and previous payment amounts. Each variable contributes to a better understanding of the creditworthiness of an individual. This is helpful to financial institutions as accurately predicting a credit default will help with their decision-making processes. For this project, I will employ machine learning models taught in class to develop a predictive framework that helps with risk mitigation by identifying potential default cases.

My analysis follows a structured approach, starting with data exploration and preprocessing. I will split the training set into the training set (80%) and validation set (20%) for testing purposes. I will then follow this by constructing the Decision Tree, Bagging, Random Forest and Gradient Boosting models. Additionally, cross-validation would be used to further test. The performance of each model will then be evaluated using the Accuracy, Precision, Recall and F1 score on both the training and validation sets. This was chosen rather than MSE since this is a classification problem and not a regression. This is a binary classification as there are only two possible outcomes, non-default or default. I will then test the chosen model on the unseen data, which will be the test data set provided.

My report will cover the following key steps:

1. **Data Exploration and Preprocessing:** Checking for missing values, handling duplicates, and exploring the relationships between variables.

2. **Model Building:** Constructing Decision Tree, Bagging, Random Forest and Gradient Boosting models.

3. **Model Evaluation:** Comparing the performance of different models on the validation set using Accuracy, Precision, Recall and F1 score through a summary table.

4. **Final Model Selection:** Choosing the best-performing model for further evaluation of the unseen data (test set).

Throughout my analysis, I will provide visualisations and insightful commentary to give a clear understanding of the relationships between predictor variables and credit default. My document will then conclude with a comprehensive evaluation of the selected model on the test set.

My analysis aims to provide valuable insights into the factors influencing credit default and to develop a predictive model for practical applications in the financial industry.

1.1 Import and view Dataset

```
#check working directory
getwd()
```

```
## [1] "/Users/jerid/Desktop/Jeri Coursework 2. Final"
```

```
#import csv file for both training and test sets
creditdefaulttrain <- read.csv("creditdefault_train.csv", header = TRUE)
creditdefaulttest <- read.csv("creditdefault_test.csv", header = TRUE)
# I would like to display the structure of the data
str(creditdefaulttrain)
```

```
## 'data.frame':	15000 obs. of  24 variables:
##  $ Y  : int  1 0 0 0 0 0 0 1 0 1 ...
##  $ X1 : int  20000 50000 50000 50000 500000 100000 630000 70000 130000 450
000 ...
##  $ X2 : int  2 2 1 1 1 2 2 1 2 2 ...
##  $ X3 : int  2 2 2 1 1 2 2 2 3 1 ...
##  $ X4 : int  1 1 1 2 2 2 2 2 2 1 ...
##  $ X5 : int  24 37 57 37 29 23 41 30 39 40 ...
##  $ X6 : int  2 0 -1 0 0 0 -1 1 0 -2 ...
##  $ X7 : int  2 0 0 0 0 -1 0 2 0 -2 ...
##  $ X8 : int  -1 0 -1 0 0 -1 -1 2 0 -2 ...
##  $ X9 : int  -1 0 0 0 0 0 -1 0 0 -2 ...
##  $ X10: int  -2 0 0 0 0 0 -1 0 0 -2 ...
##  $ X11: int  -2 0 0 0 0 -1 -1 2 -1 -2 ...
##  $ X12: int  3913 46990 8617 64400 367965 11876 12137 65802 38358 5512 ...
##  $ X13: int  3102 48233 5670 57069 412023 380 6500 67369 27688 19420 ...
##  $ X14: int  689 49291 35835 57608 445007 601 6500 65701 24489 1473 ...
##  $ X15: int  0 28314 20940 19394 542653 221 6500 66782 20616 560 ...
##  $ X16: int  0 28959 19146 19619 483003 -159 6500 36137 11802 0 ...
```

```
##  $ X17: int  0 29547 19131 20024 473944 567 2870 36894 930 0 ...
##  $ X18: int  0 2000 2000 2500 55000 380 1000 3200 3000 19428 ...
##  $ X19: int  689 2019 36681 1815 40000 601 6500 0 1537 1473 ...
##  $ X20: int  0 1200 10000 657 38000 0 6500 3000 1000 560 ...
##  $ X21: int  0 1100 9000 1000 20239 581 6500 3000 2000 0 ...
##  $ X22: int  0 1069 689 1000 13750 1687 2870 1500 930 0 ...
##  $ X23: int  0 1000 679 800 13770 1542 0 0 33764 1128 ...
```

```
#summary
summary(creditdefaulttrain)
```

```
##        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
```

```r
# I will check for duplicates in the entire data frame
duplicates <- creditdefaulttrain[duplicated(creditdefaulttrain), ]
# I will now display the dimensions of the duplicates in the data
dim(duplicates)
```

```
## [1] 11 24
```

The dataset covers a variety of information about credit card users. Within the credit default dataset, there are 11 rows that display duplication. While the existence of duplicate rows has the potential to introduce biases in my analytical processes, it seems more likely in this case that various sources have assessed the creditworthiness of individuals and assigned comparable ratings across the evaluated variables. As a result, I have chosen to keep all observations. This could enrich my analysis with additional meaningful insights. Also, there does not seem to be any extremely high or low values in most categories. The credit amount (X1) and age (X5) have diverse ranges. The response variable (Y), indicating if a user defaulted (1) or not (0), shows a potential imbalance, with a mean of 0.2212. This suggests more non-default cases. The dataset seems balanced for categorical variables like gender (X2), education (X3), and marital status (X4). Still, I need to check if there are extreme values in variables related to amounts (X12 to X23). Overall, the dataset looks good, but I will dig deeper into specific variables through EDA.

1.2 Checking the missing values in Dataframes

Missing Values

There are no missing values in the dataset, which is ideal for modeling as it simplifies the data preprocessing stage.

```r
# Check for missing values
missing_values <- sum(is.na(creditdefaulttrain))
print(missing_values)
```

```
## [1] 0
```

```r
missing_values_test <- sum(is.na(creditdefaulttest))
```

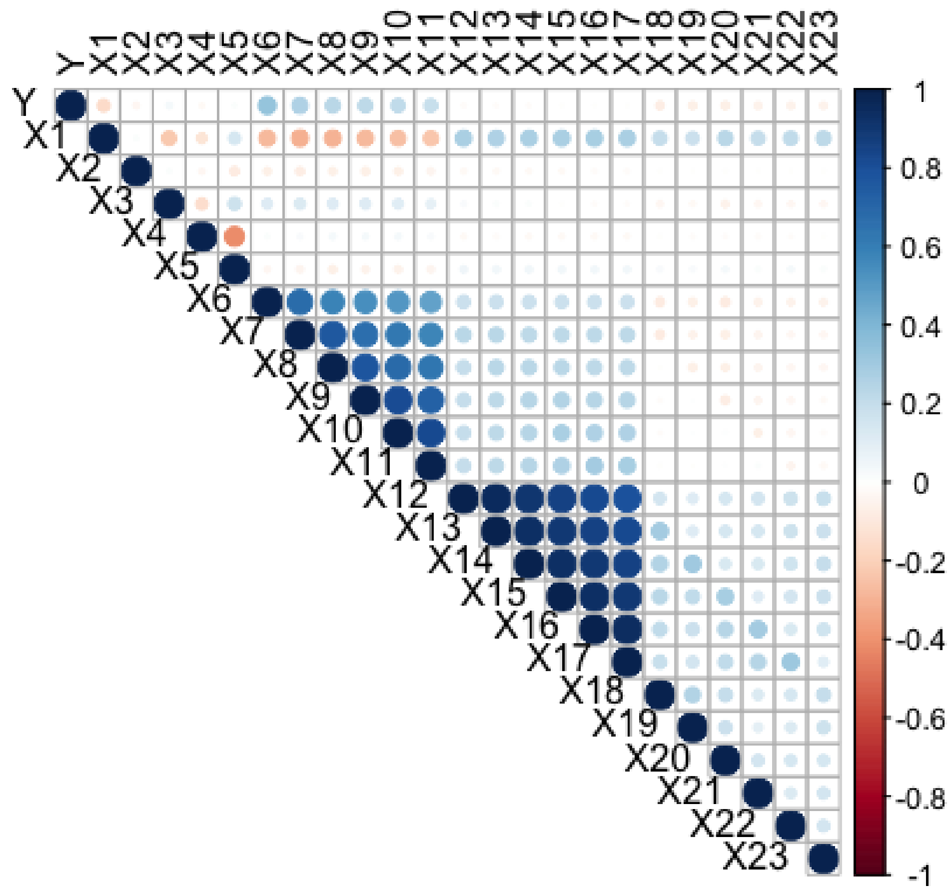
1.3 EDA on training set

Loading libraries and checking training data.

I will load the necessary libraries and examine the data summary. The dataset comprises 24 variables and some of them may be correlated. To explore this, I'll assess the strength of the correlation between Y (credit default) and other variables by creating a correlation matrix plot.

```r
# I will load necessary libraries, including ggplot2 for plotting and corrplo
t for visualising the correlations.
library(ggplot2)
library(corrplot)

## corrplot 0.92 loaded

# I will now look at the first few rows of the training data and test data
head(creditdefaulttrain)

##   Y     X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11    X12    X13    X14    X15
X16
## 1 1  20000  2  2  1 24  2  2 -1 -1  -2  -2   3913   3102    689      0
0
## 2 0  50000  2  2  1 37  0  0  0  0   0   0  46990  48233  49291  28314  28
959
## 3 0  50000  1  2  1 57 -1  0 -1  0   0   0   8617   5670  35835  20940  19
146
## 4 0  50000  1  1  2 37  0  0  0  0   0   0  64400  57069  57608  19394  19
619
## 5 0 500000  1  1  2 29  0  0  0  0   0   0 367965 412023 445007 542653 483
003
## 6 0 100000  2  2  2 23  0 -1 -1  0   0  -1  11876    380    601    221   -
159
##      X17   X18   X19   X20   X21   X22   X23
## 1      0     0   689     0     0     0     0
## 2  29547  2000  2019  1200  1100  1069  1000
## 3  19131  2000 36681 10000  9000   689   679
## 4  20024  2500  1815   657  1000  1000   800
## 5 473944 55000 40000 38000 20239 13750 13770
## 6    567   380   601     0   581  1687  1542

head(creditdefaulttest)

##   Y     X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11   X12   X13   X14   X15   X16
X17
## 1 1 120000  2  2  2 26 -1  2  0  0   0   2  2682  1725  2682  3272  3455
3261
## 2 0  90000  2  2  2 34  0  0  0  0   0   0 29239 14027 13559 14331 14948 1
5549
## 3 0 140000  2  3  1 28  0  0  2  0   0   0 11285 14096 12108 12211 11793
3719
## 4 0  20000  1  3  2 35 -2 -2 -2 -2  -1  -1     0     0     0     0 13007 1
3912
## 5 0 200000  2  3  2 34  0  0  2  0   0  -1 11073  9787  5535  2513  1828
3731
## 6 0 260000  2  1  2 51 -1 -1 -1 -1  -1   2 12261 21670  9966  8517 22287 1
3668
##     X18  X19  X20   X21  X22  X23
## 1     0 1000 1000  1000    0 2000
## 2  1518 1500 1000  1000 1000 5000
```

```
## 3   3329    0   432   1000 1000 1000
## 4      0    0     0 13007 1122    0
## 5   2306   12    50   300 3738   66
## 6 21818 9966 8583 22301    0 3640

# Computing the variable correlations
cor_matrix <- cor(creditdefaulttrain)

# I will now plot the correlation matrix
corrplot(cor_matrix, method = "circle", type = "upper", tl.col = "black")
```



Correlation among variables Summary and insights based on the correlation matrix:

In summary, the correlation matrix reveals associations between different variables in the dataset. The response variable (Y), indicating a credit card default, shows a negative correlation with the amount of credit (X1) and the repayment status in September (X6). On the other hand, it shows weaker positive correlations with gender (X2), education (X3), and age (X5). The amount of credit, X1, shows a negative correlation with both Y and X6 but positive correlations with bill statements and previous payments (X7 to X23).

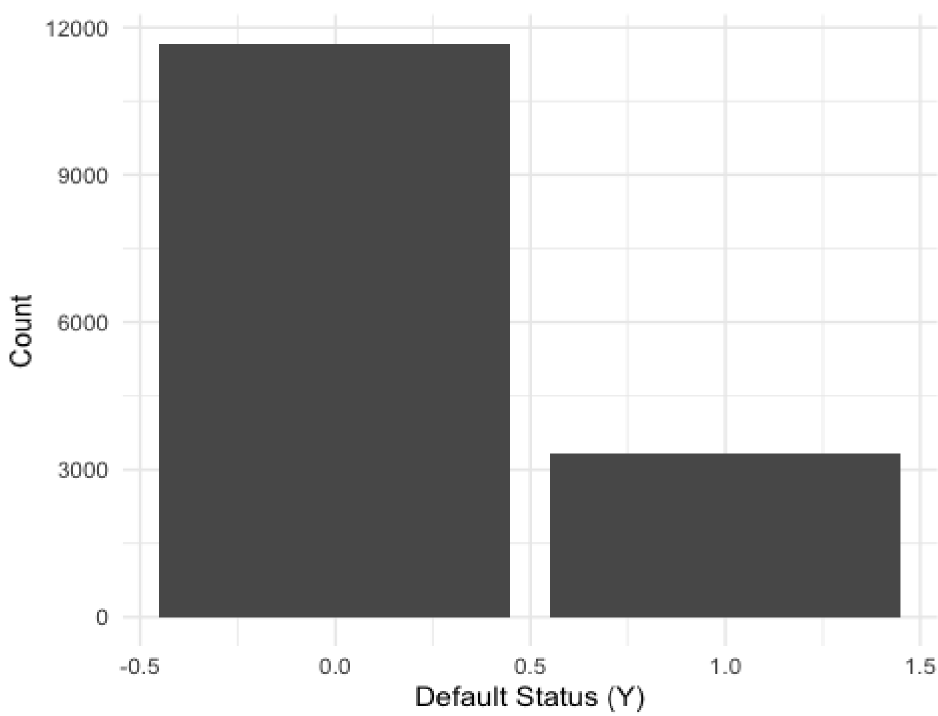
Distribution

This analysis aims to focus on the prediction of credit card default. Y is what I aim to predict using other attributes. Now, I will look into a summary of the credit card default by creating a table and visualising it through a histogram. This will give an overview of the distribution of credit card defaults.

```
#Creating a table for the counting of variables in Y
table(creditdefaulttrain$Y)

##
##     0     1
## 11682  3318
```

Using the table() function showed that the majority of instances belong to the non-default category (0), with a count of 11,682. However, there are 3,318 instances of credit card default (1). This distribution provides a crucial insight into the dataset and shows a higher amount of non-default cases. To gain a visual representation, I will create a histogram to illustrate the distribution of credit card default.

```
#Histogram for distibution of credit card default
theme_set(theme_minimal())
ggplot(creditdefaulttrain,aes(Y)) + geom_histogram(stat="count") + xlab("Defa
ult Status (Y)") + ylab("Count")

## Warning in geom_histogram(stat = "count"): Ignoring unknown parameters:
## `binwidth`, `bins`, and `pad`
```

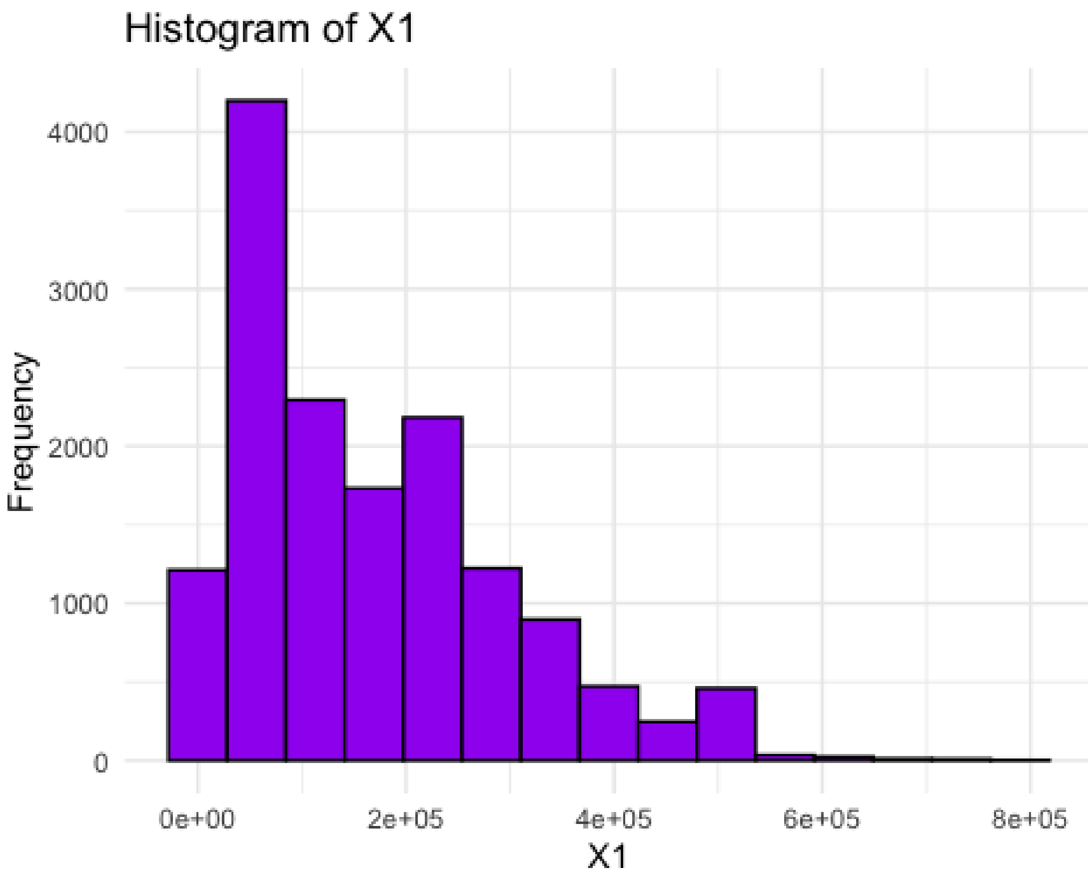


Further Visualisations

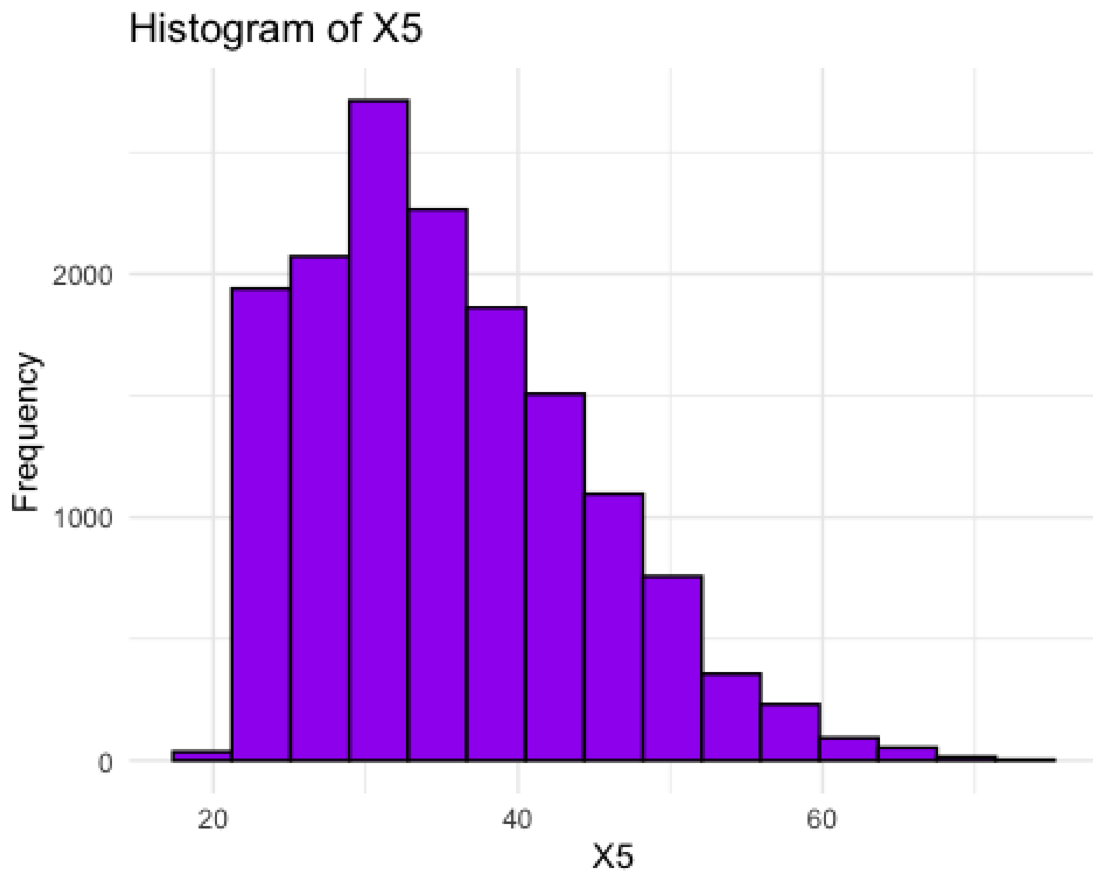
Histograms for Numerical Features and Bar Charts for Categorical Features

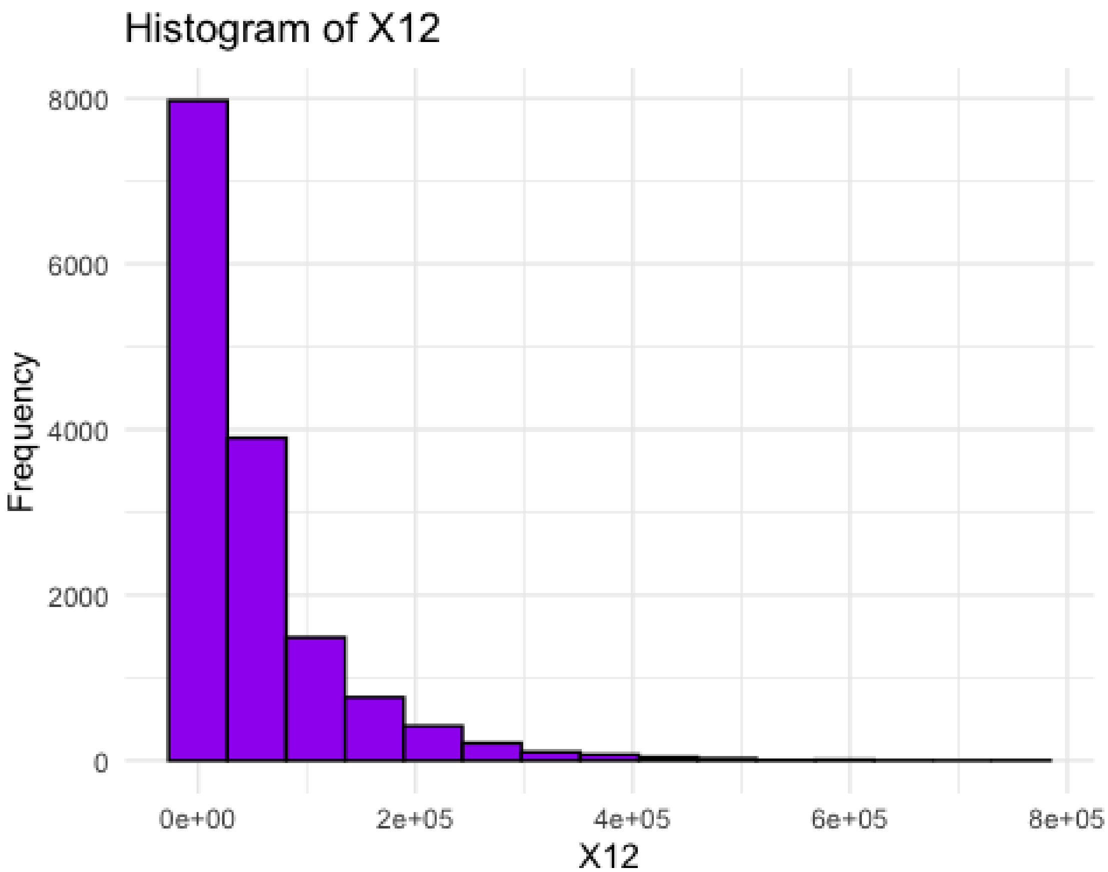
I will create further histograms below to provide insights into the distribution of numerical features. This includes the amount of credit (X1), age (X5), amount of bill statement (X12 to X17), and amount of previous payment (X18 to X23). I will then create Bar Charts to provide insights on the categorical variables like gender (X2), education (X3), marital status (X4), and repayment status from April to September 2005 (X6 to X11).

```
# First I will define numerical features
numerical_features <- c('X1', 'X5', paste0('X', 12:23))

# Then I will define categorical features
categorical_features <- c('X2', 'X3', 'X4', paste0('X', 6:11))

# Next I will define a function to plot histogram or bar chart based on featu
re type
plot_feature <- function(data, feature, feature_type) {
  if (feature_type == "numerical") {
    p <- ggplot(data, aes(!!sym(feature))) +
          geom_histogram(bins=15, fill="purple", color="black") +
          labs(title=paste("Histogram of", feature), x=feature, y="Frequency
") +
          theme_minimal()
  } else if (feature_type == "categorical") {
    p <- ggplot(data, aes(!!sym(feature))) +
          geom_bar(fill="orange", color="black") +
          labs(title=paste("Bar Chart of", feature), x=feature, y="Count") +
          theme_minimal()
  }
  print(p)
}

# Plotting the histograms for numerical features
for (feature in numerical_features) {
  plot_feature(creditdefaulttrain, feature, "numerical")
}
```

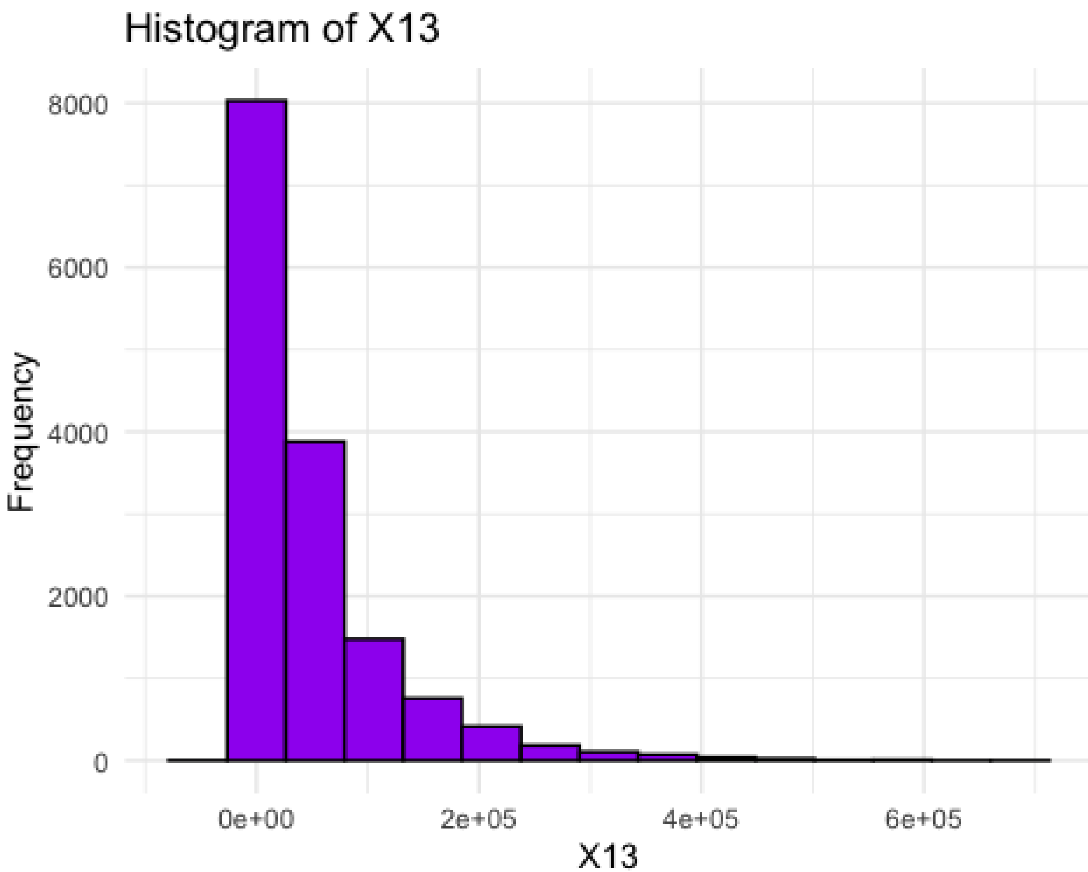


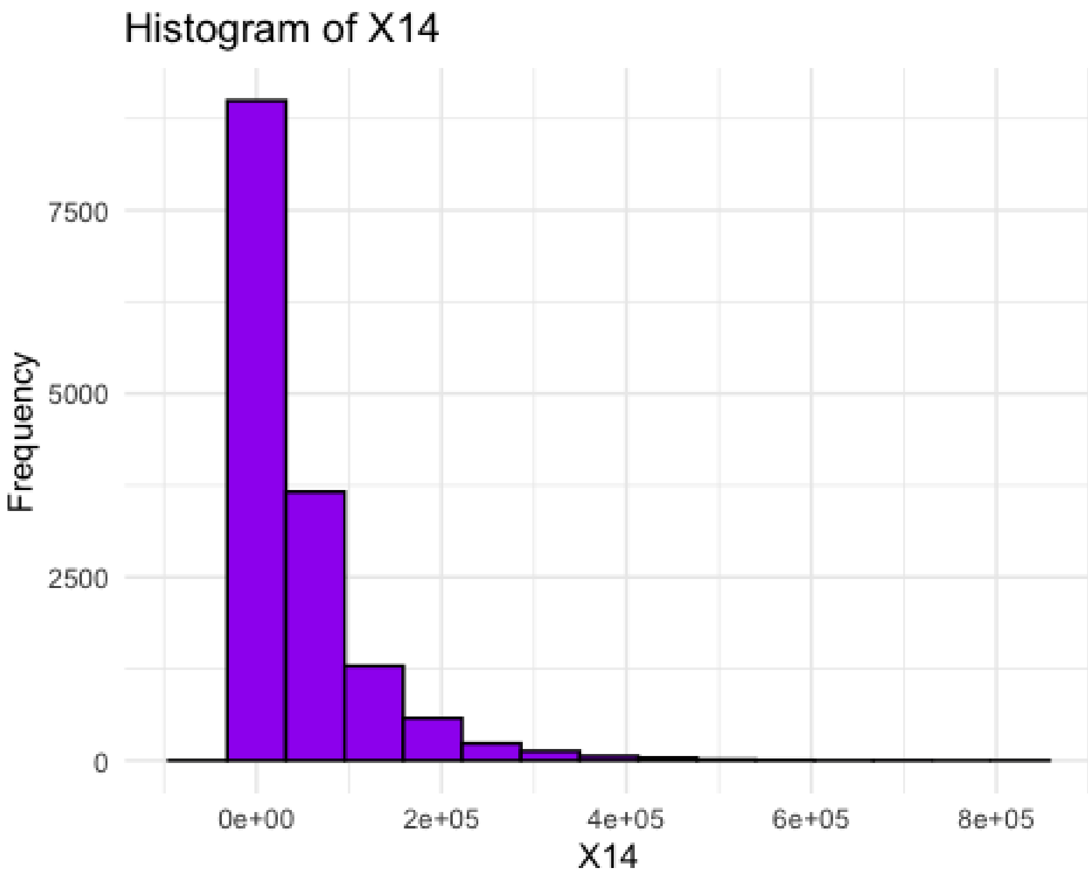
Histogram of X1
Frequency
4000
3000
2000
1000
0
0e+00
2e+05
4e+05
6e+05
8e+05
X1

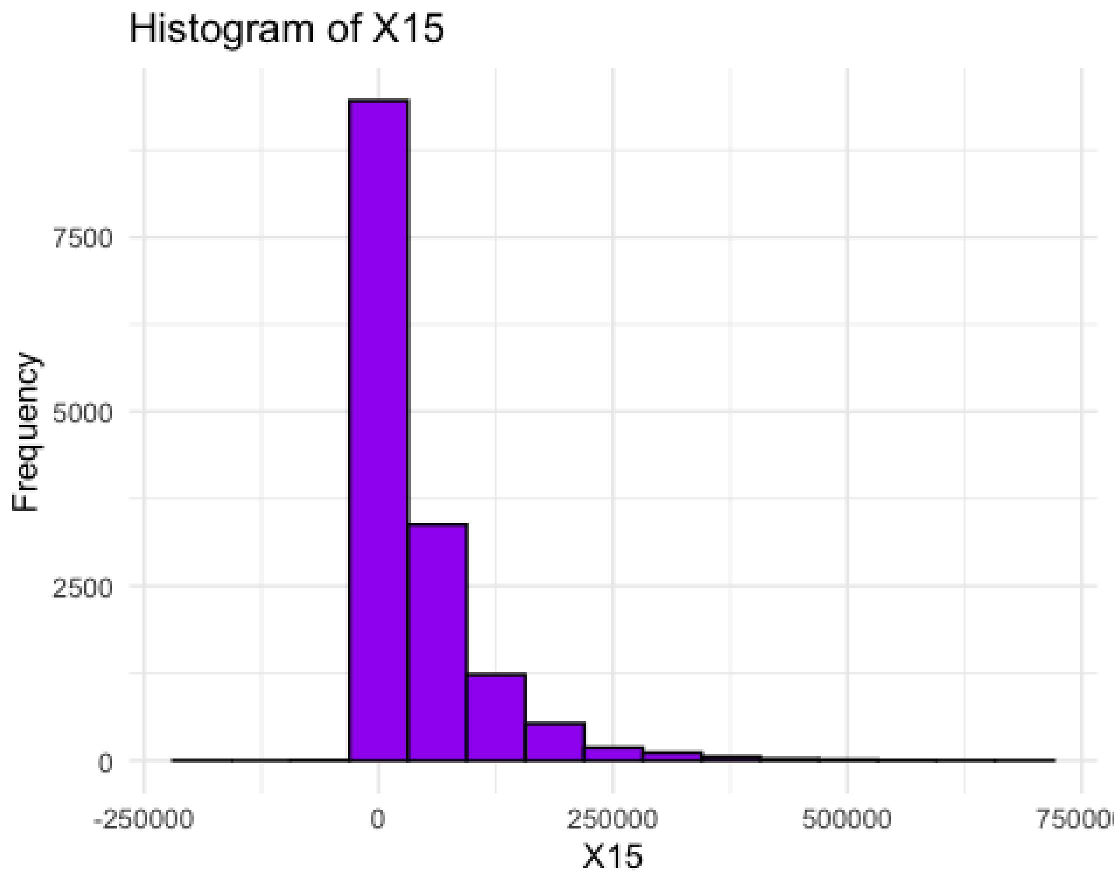

Histogram of X5
Frequency
2000
1000
0
20
40
60
X5

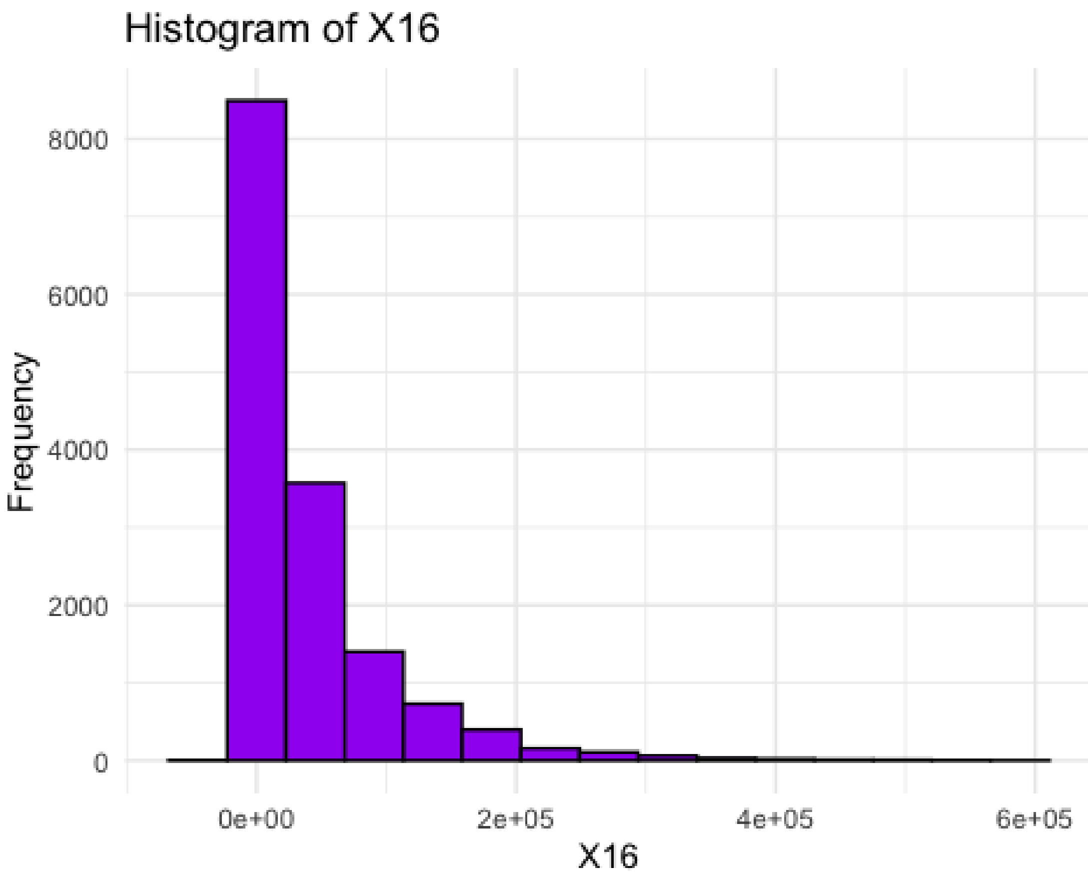


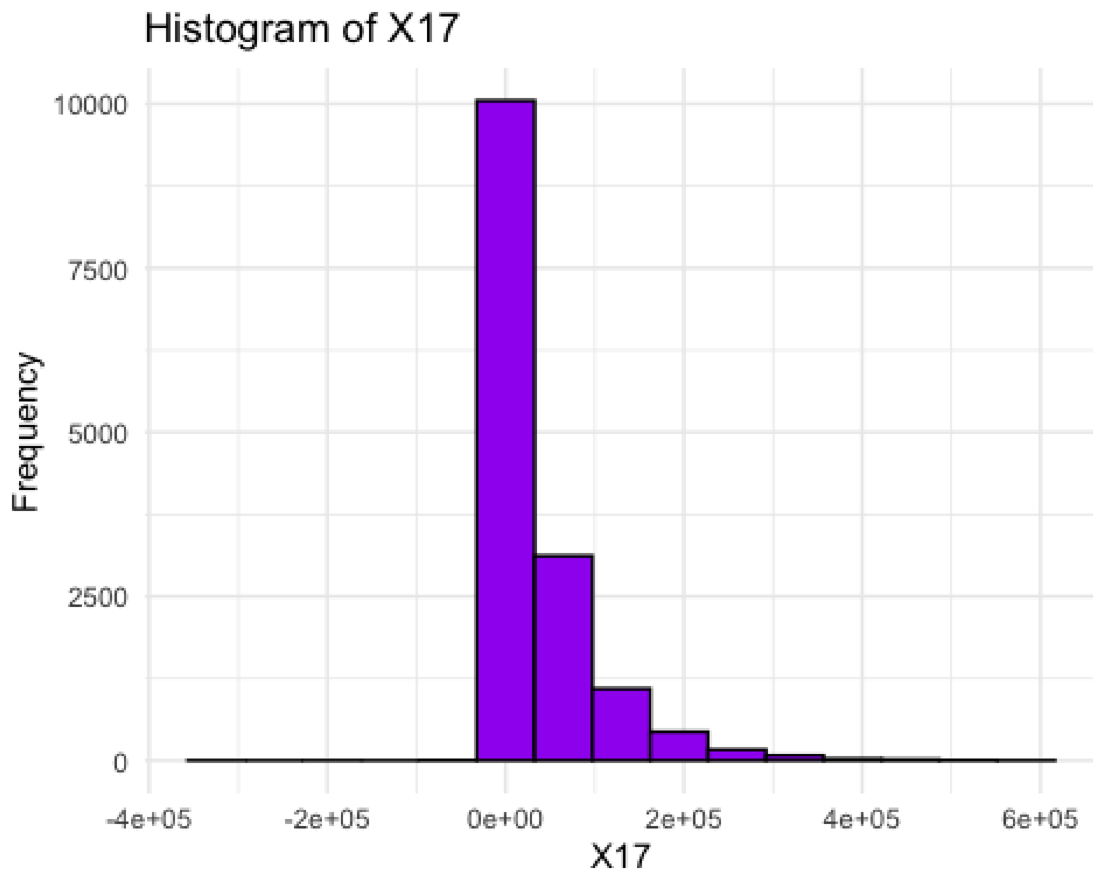
Histogram of X12
Frequency
8000
6000
4000
2000
0
0e+00
2e+05
4e+05
6e+05
8e+05
X12

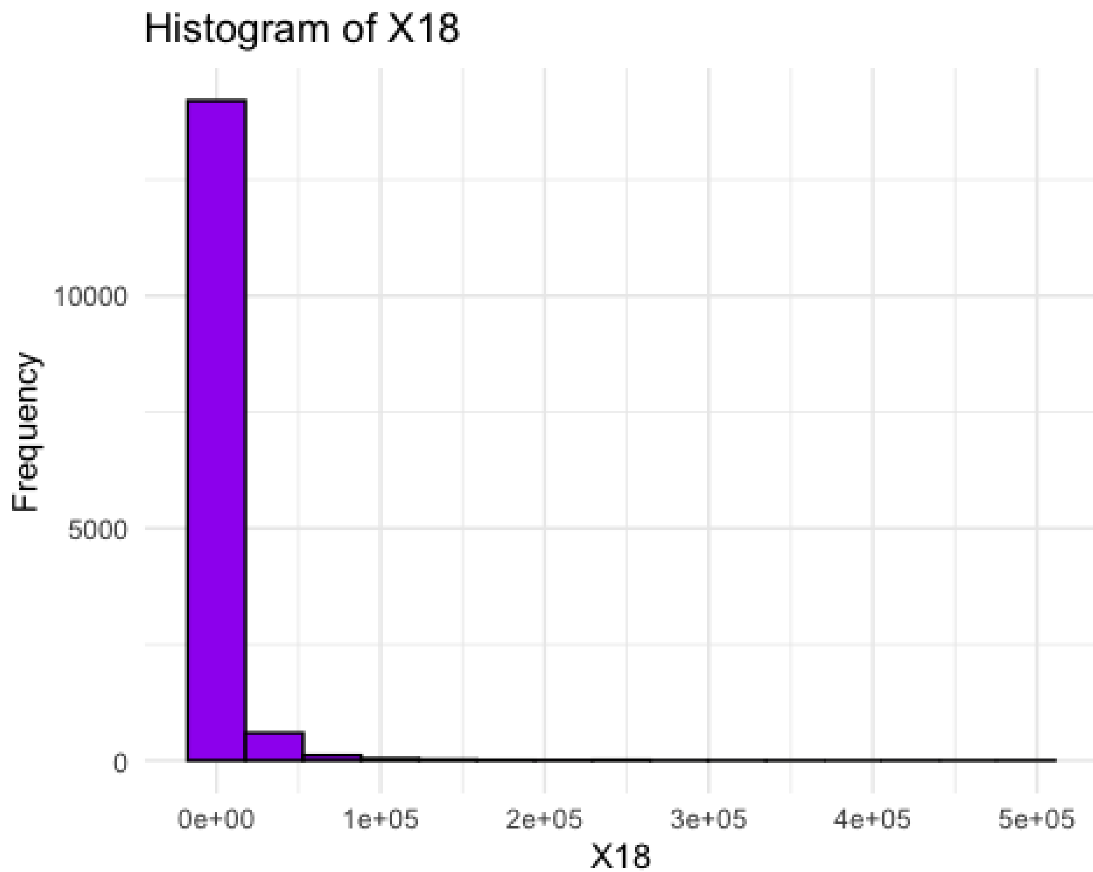

Histogram of X13
Frequency
8000
6000
4000
2000
0
0e+00
2e+05
4e+05
6e+05
X13


Histogram of X14
Frequency
7500
5000
2500
0
0e+00
2e+05
4e+05
6e+05
8e+05
X14

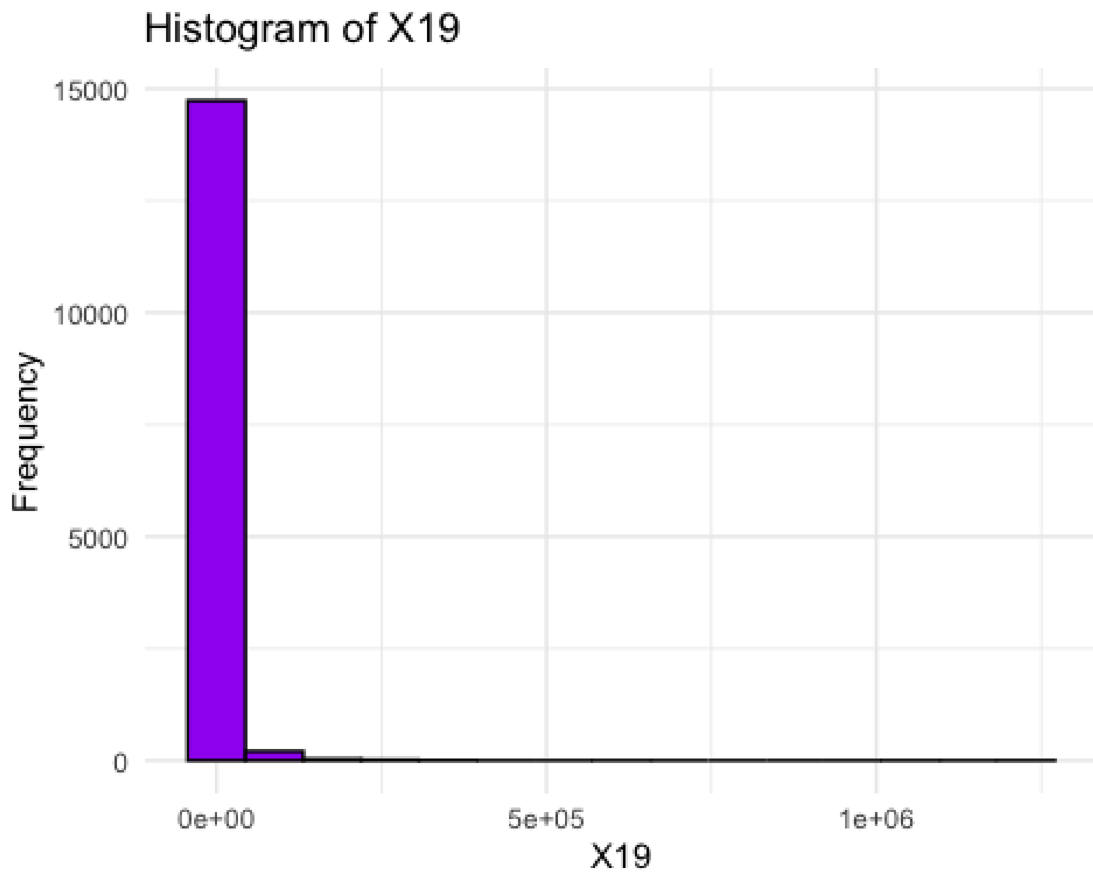

Histogram of X15
Frequency
7500
5000
2500
0
-250000
0
250000
500000
X15


Histogram of X16
Frequency
8000
6000
4000
2000
0
0e+00
2e+05
4e+05
6e+05
X16

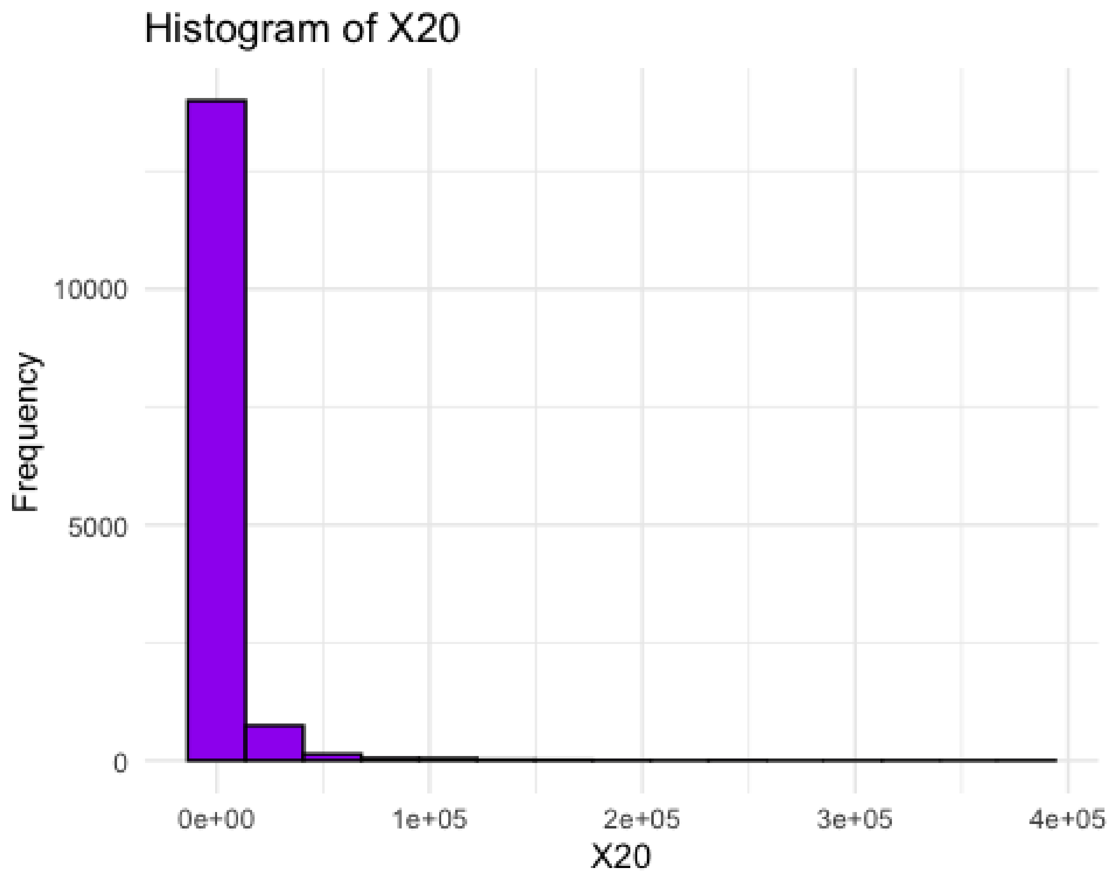

Histogram of X17
10000
7500
5000
2500
0
Frequency
-4e+05
-2e+05
0e+00
2e+05
4e+05
6e+05
X17

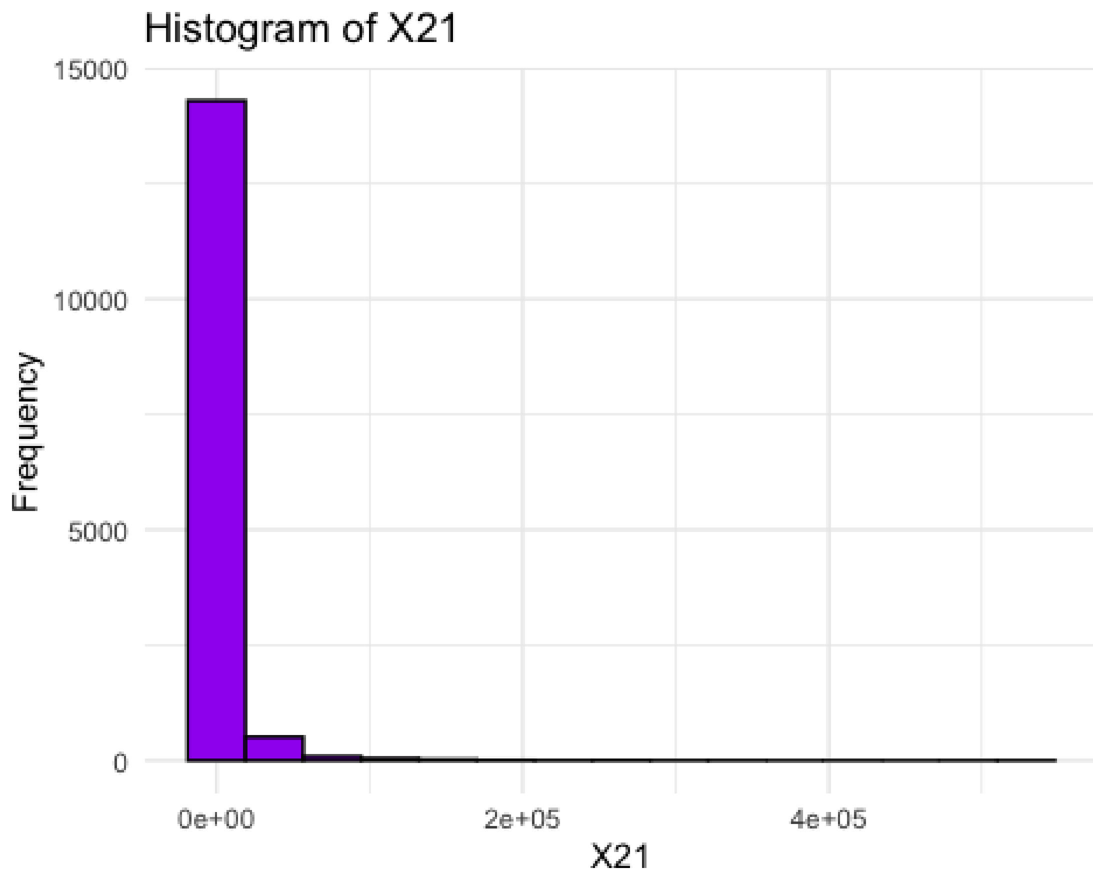


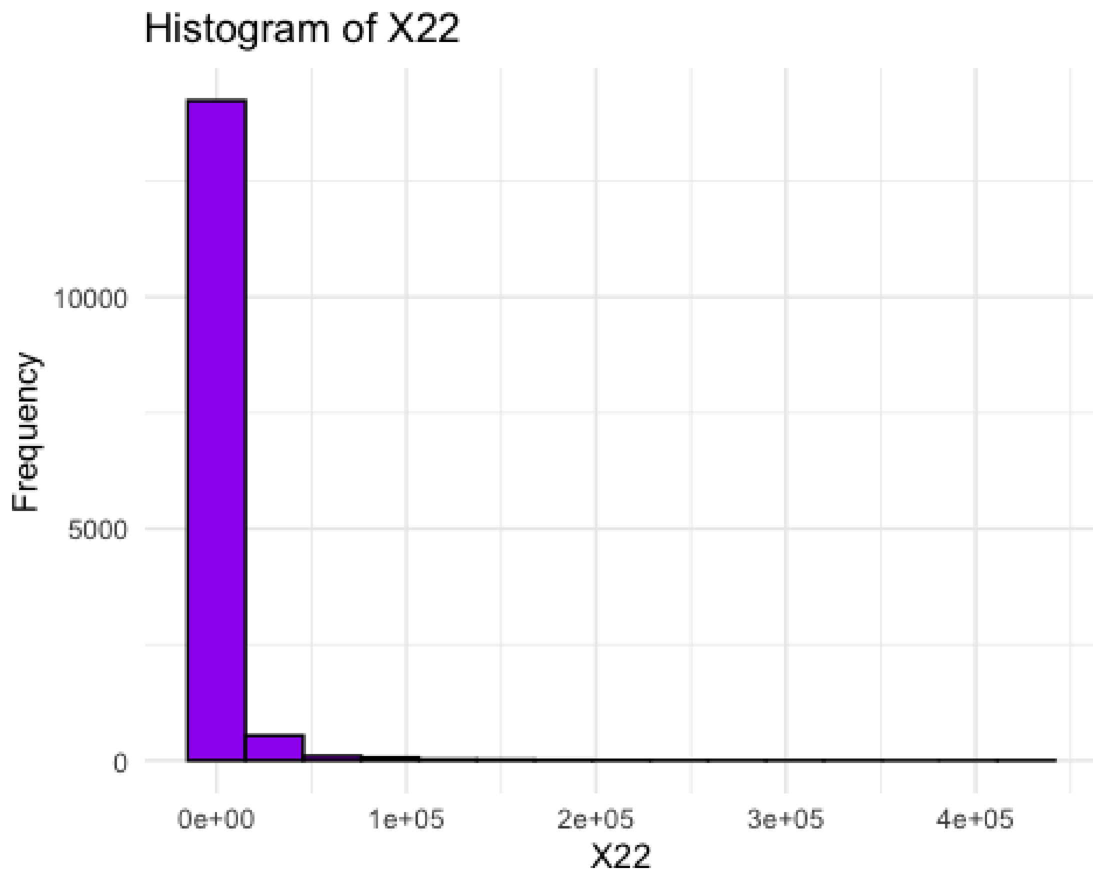
Histogram of X18
Frequency
10000
5000
0
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
X18



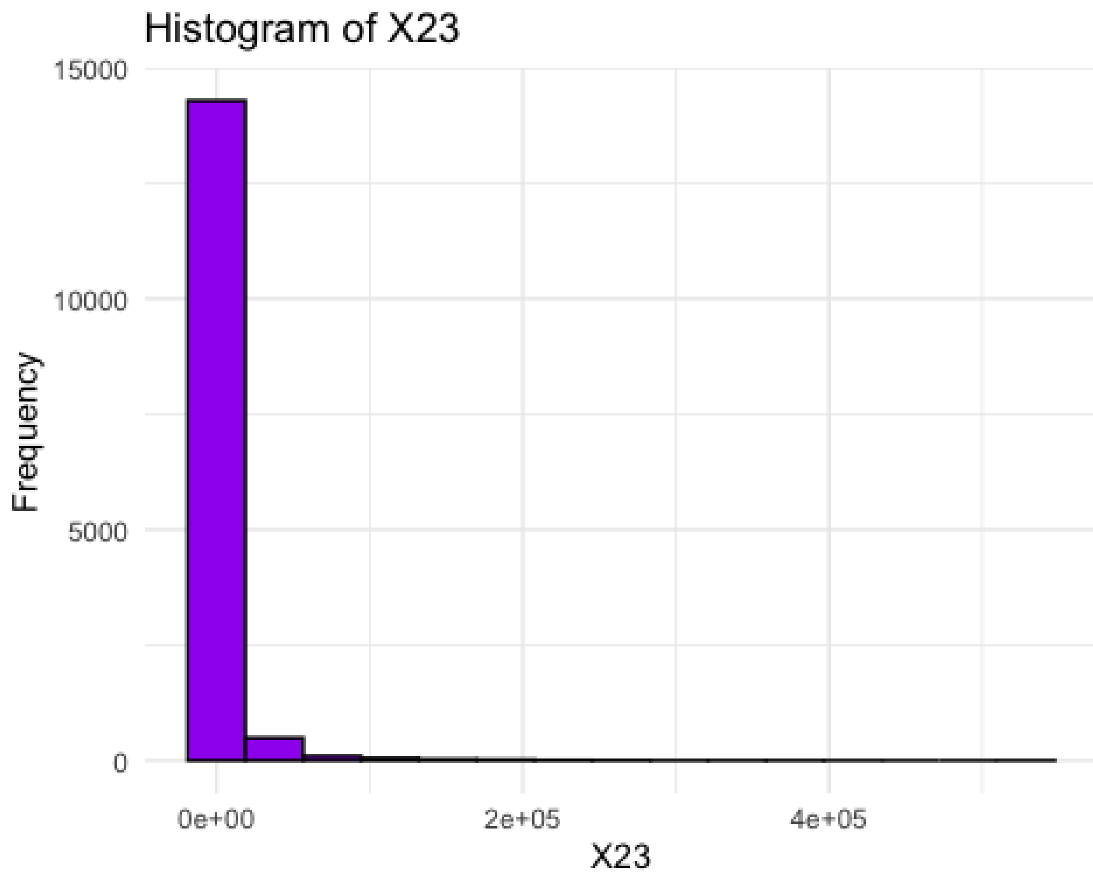
Histogram of X19
15000
10000
5000
0
Frequency
0e+00
5e+05
1e+06
X19


Histogram of X20
Frequency
10000
5000
0
0e+00
1e+05
2e+05
3e+05
4e+05
X20

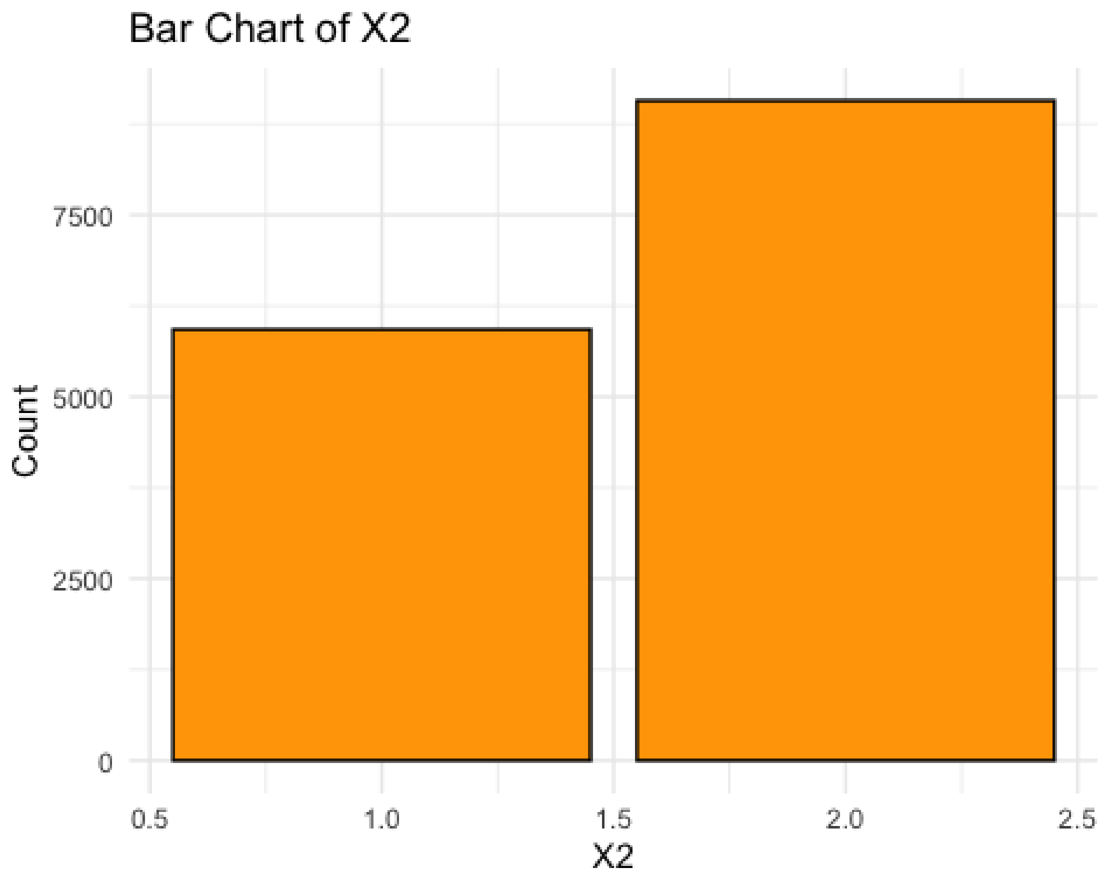

Histogram of X21
15000
10000
5000
0
Frequency
0e+00
2e+05
4e+05
X21

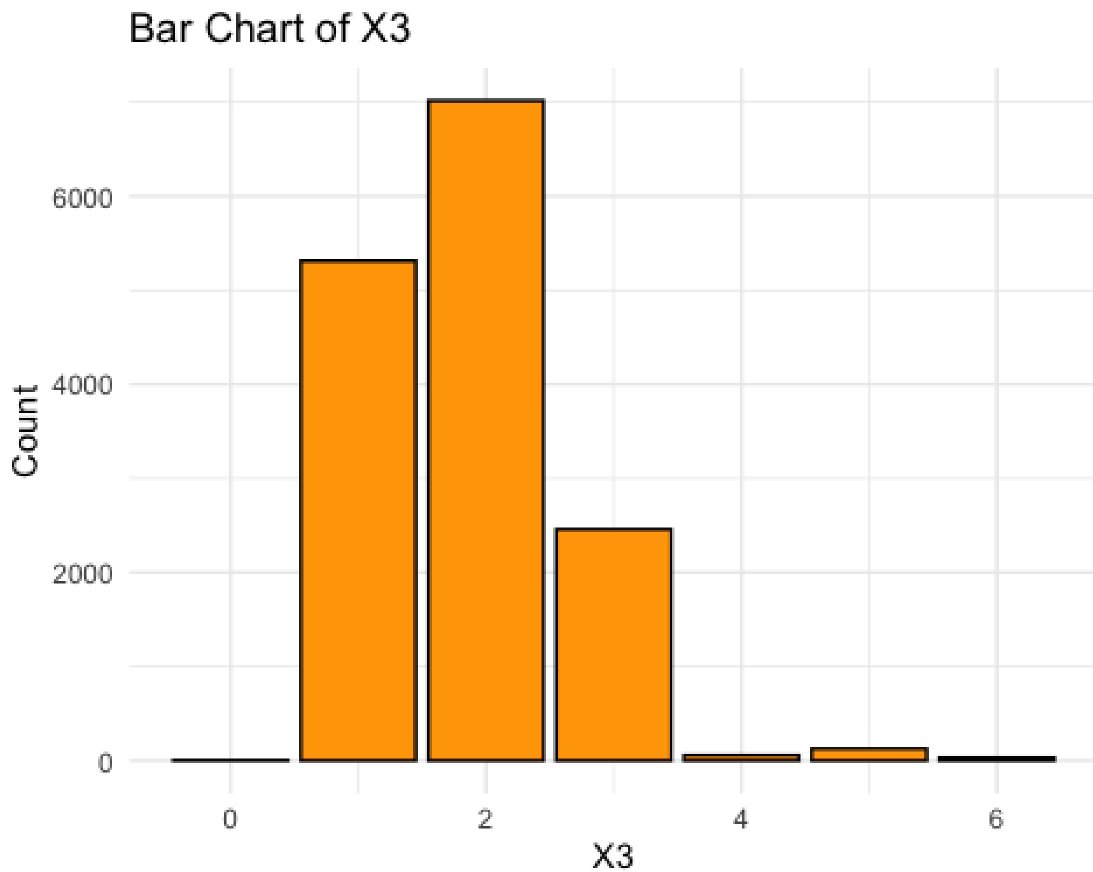


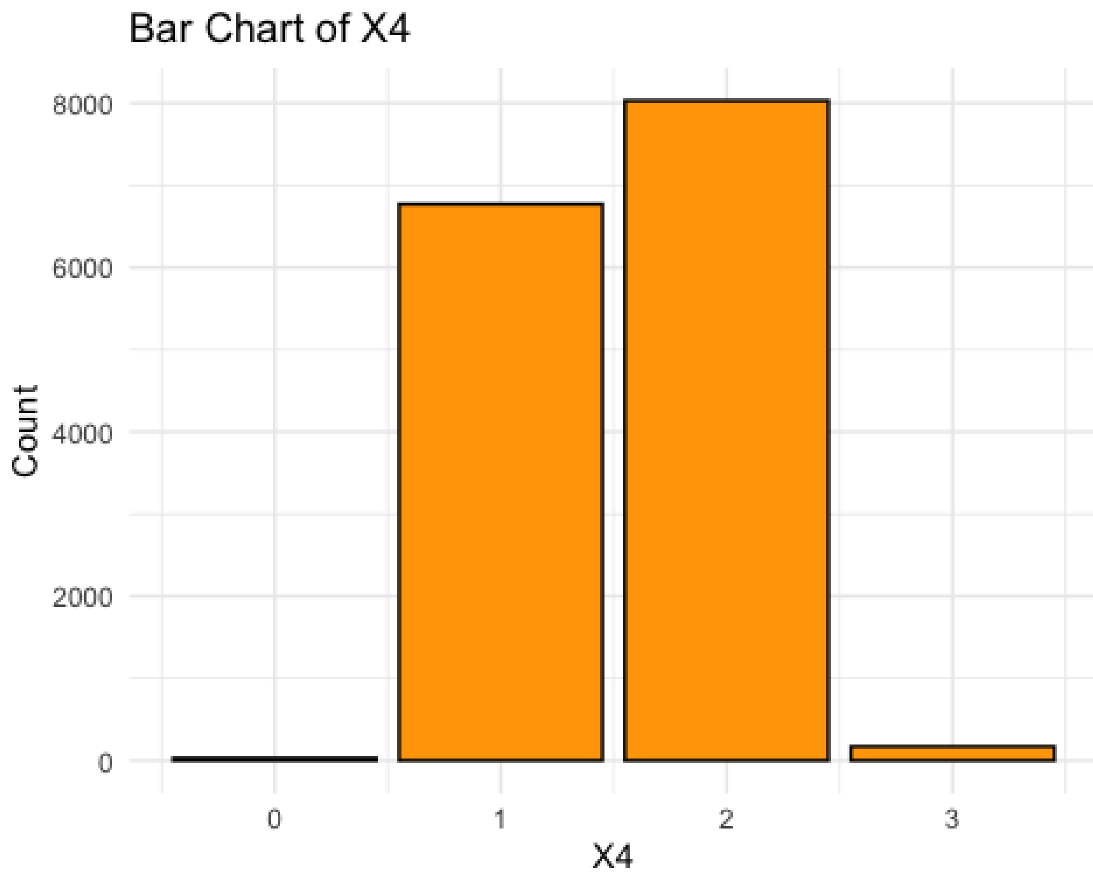
Histogram of X22
Frequency
10000
5000
0
0e+00
1e+05
2e+05
3e+05
4e+05
X22



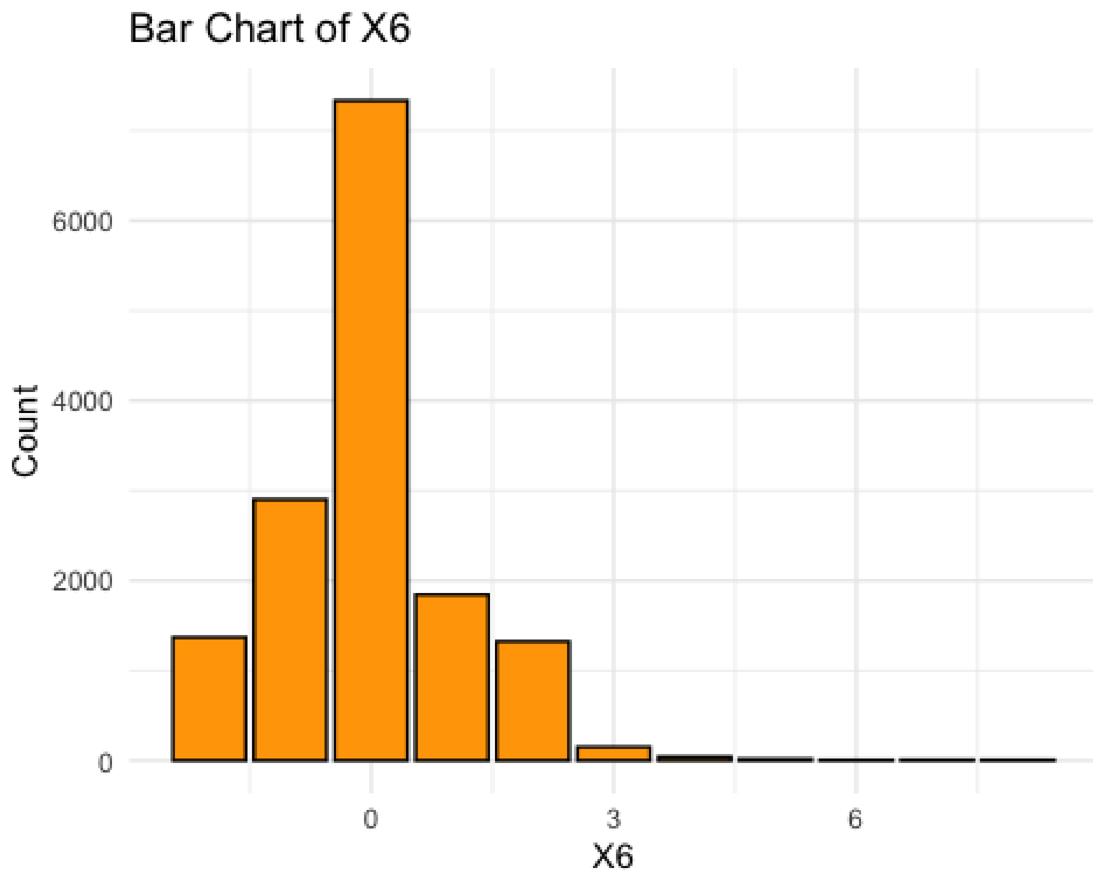

```
# Plotting the bar charts for categorical features
for (feature in categorical_features) {
  plot_feature(creditdefaulttrain, feature, "categorical")
}
```


Bar Chart of X2
Count
7500
5000
2500
0
0.5
1.0
1.5
2.0
2.5
X2

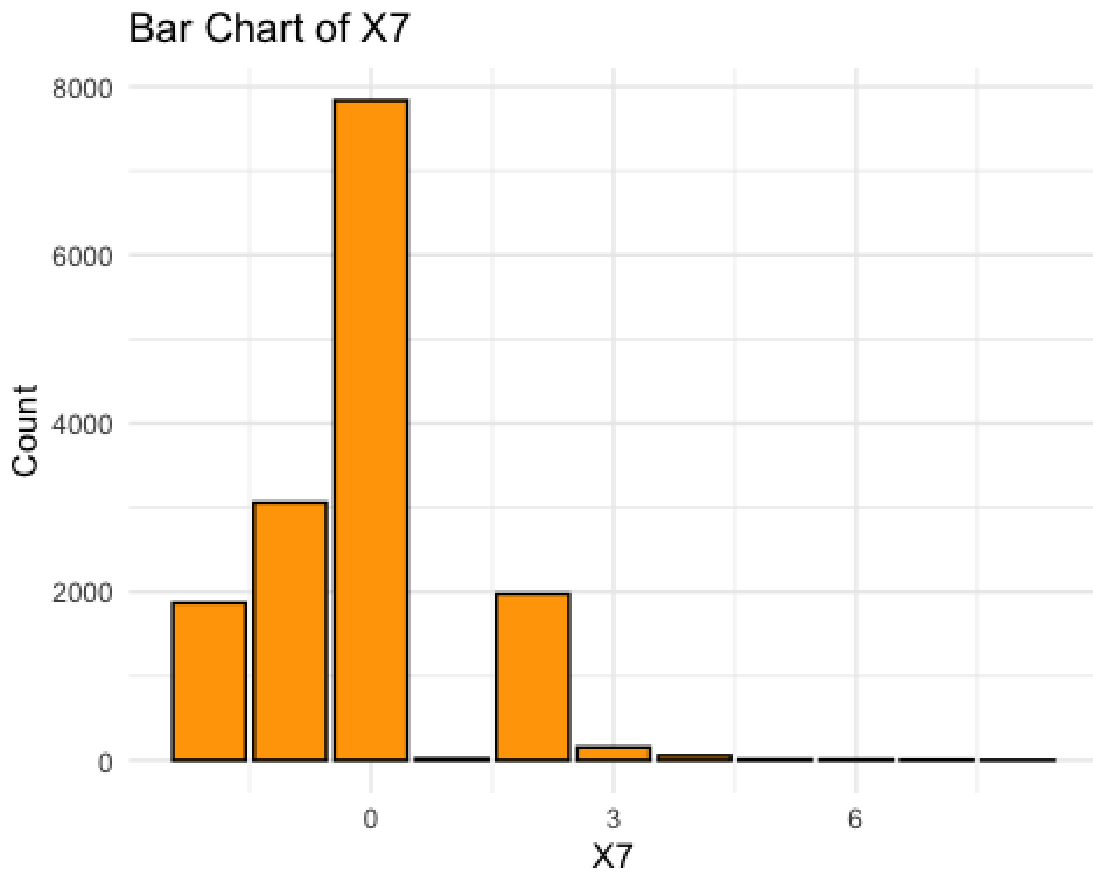

Bar Chart of X3
Count
6000
4000
2000
0
0
2
4
6
X3

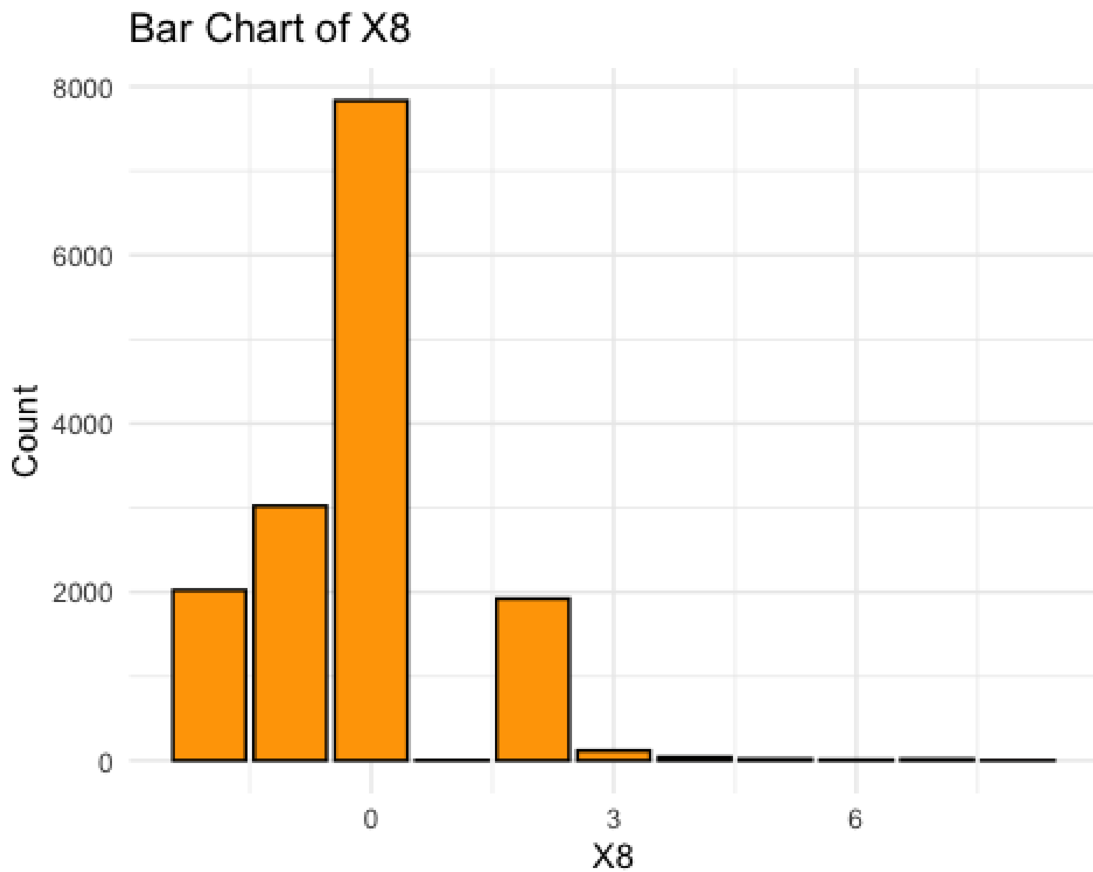


Bar Chart of X4
8000
6000
4000
2000
0
Count
0
1
2
3
X4

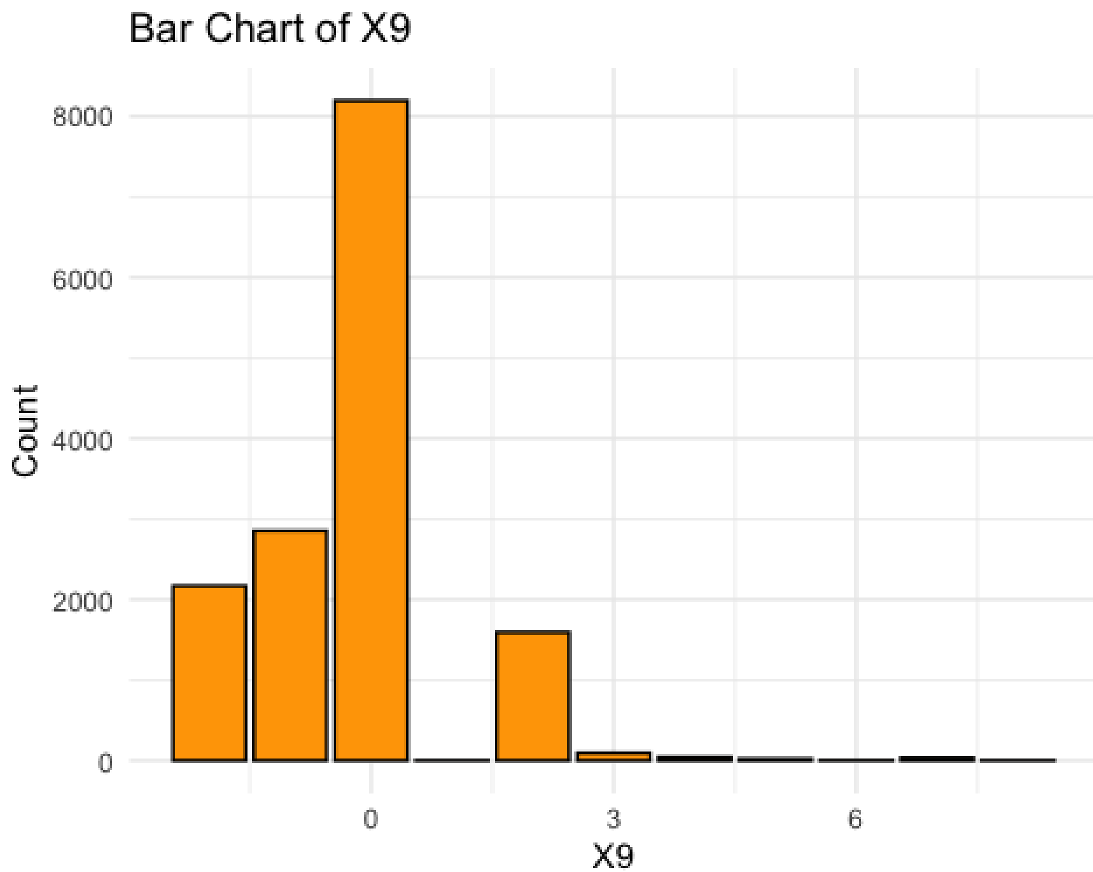


Bar Chart of X6
Count
6000
4000
2000
0
0
3
6
X6

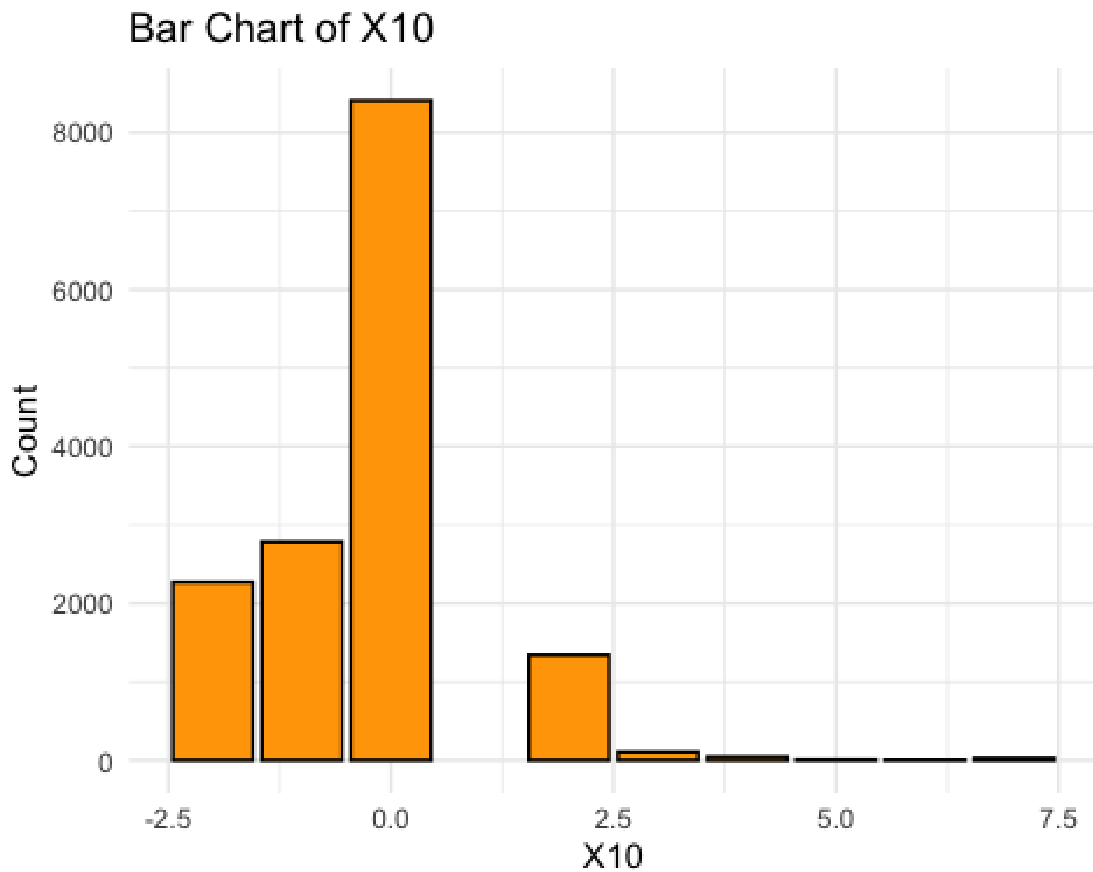

Bar Chart of X7
8000
6000
4000
2000
0
Count
0
3
6
X7

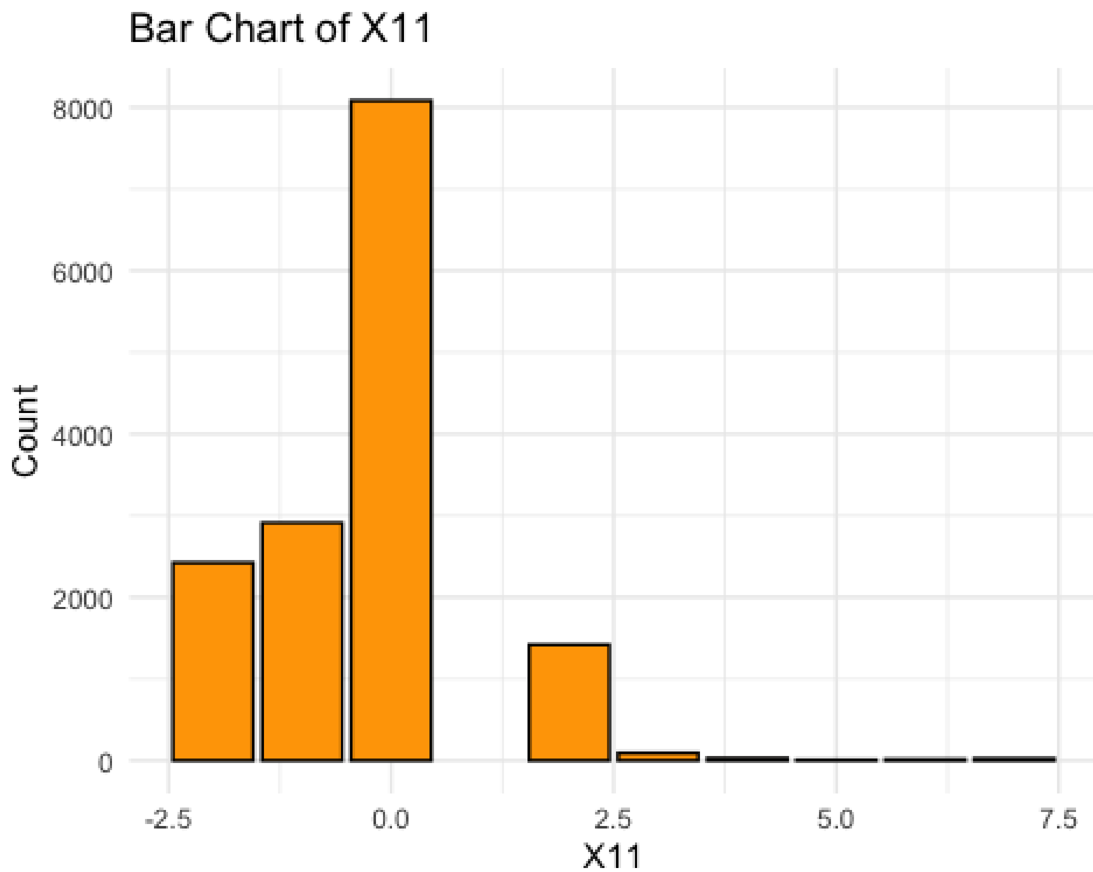


Bar Chart of X8
8000
6000
4000
2000
0
Count
0
3
6
X8



Bar Chart of X9
8000
6000
4000
2000
0
Count
0
3
6
X9


Bar Chart of X10
Count
8000
6000
4000
2000
0
-2.5
0.0
2.5
5.0
7.5
X10



Findings and Insights from the Histograms

Numerical Histograms

X1 (Credit Amount): Shows a right-skewed distribution. This illustrates that a larger amount of individuals have lower credit amounts given, while less individuals have higher credit amounts given. This is expected as it is common for customers to be granted lower credit limits.

X5 (Age): Shows a slightly right-skewed distribution. This indicates that there are more middle-aged customers and less older customers. It seems the most common age is within the 30s.

X12-17 (Bill Statement Amounts): Generally show a heavily right-skewed distribution. This suggests that more individuals have lower bill statement amounts, while a very small amount has higher bill amounts. Also, there are some customers with a bill amount of zero, suggesting that the bills have been fully paid or the credit has been unused.

X18-23 (Previous Payment Amounts): Also shows a heavily right-skewed distribution. This implies that more customers are making minimum or partial payments towards their bill amounts. This could be a potential indicator of credit risk as customers may be struggling to pay off their credit balance.

Categorical Histograms

X2 (Gender): Shows a gender imbalance with more female (2) customers in the credit card default dataset.

X3 (Education): Shows the most common education level among customers was university (2). The second most common was graduate school (1) and the third was high school (3).

X4 (Marital Status): Shows that single and married are the most common, while others were less common. There were more single (2) customers than married (1).

X6-X11 (Repayment Status): Shows the tallest bar is 'pay duly' (0). Since there are less bars after 0 and more frequency of bars below 0, it suggests that only a few amount of customers have longer payment delays Payment history is usually a critical variable for predicting credit default risk.

1.4 Preparing Data for Modelling through Data Split and Decision Trees

Since I am working alone, I will be working on Decision Trees, Bagging, Random Forest and gradient Boosting. I will not do one-hot encoding and scaling as tree-based models can handle categorical features and numerical data differently. However, I will need to feature engineer my data to prepare it for the modelling.

I will first split the data into the training and validation sets to train the models on the training set and then evaluate the model on the validation set. This is because I will be able to see its performance on the validation set and keep it unbiased.

Data Preprocessing

```
# Load the rpart package and .plot for better tree visualisation.
library(caret)

## Loading required package: lattice

library(rpart)
library(rpart.plot)

# Setting the seed for reproducibility
set.seed(123)

# I will now split the data into training and validation sets. 80% will be fo
r training and the rest will be for the validation set.

splitindex <- createDataPartition(creditdefaulttrain$Y, p = 0.8, list = FALSE
)
train_set <- creditdefaulttrain[splitindex, ]
val_set <- creditdefaulttrain[-splitindex, ]
```

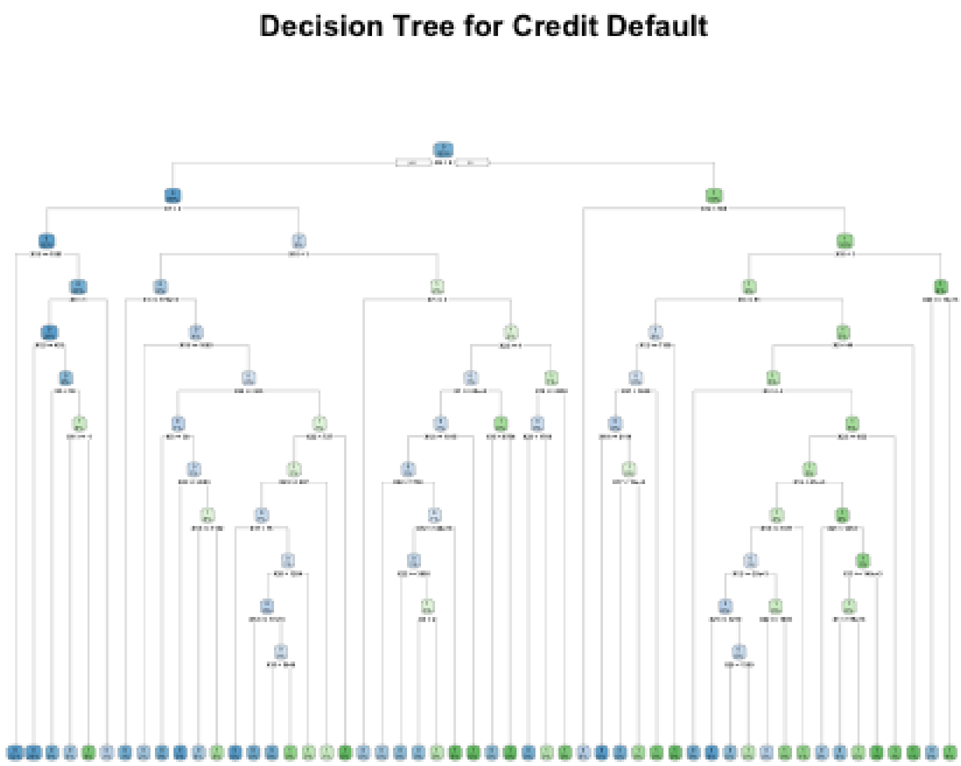
```r
# I will now convert the target variable Y to a factor to enforce a classific
ation context. Since Y is either 0 or 1, models like randomforest could mista
kenly pick it up as a regression problem.

train_set$Y <- factor(train_set$Y)
val_set$Y <- factor(val_set$Y, levels = levels(train_set$Y))

# Checking the unique values in the target variable

unique(train_set$Y)

## [1] 1 0
## Levels: 0 1

unique(val_set$Y)

## [1] 1 0
## Levels: 0 1

#Confirming that the data type of target variable is an integer for modelling
class(train_set$Y)

## [1] "factor"
```

2. Model Building

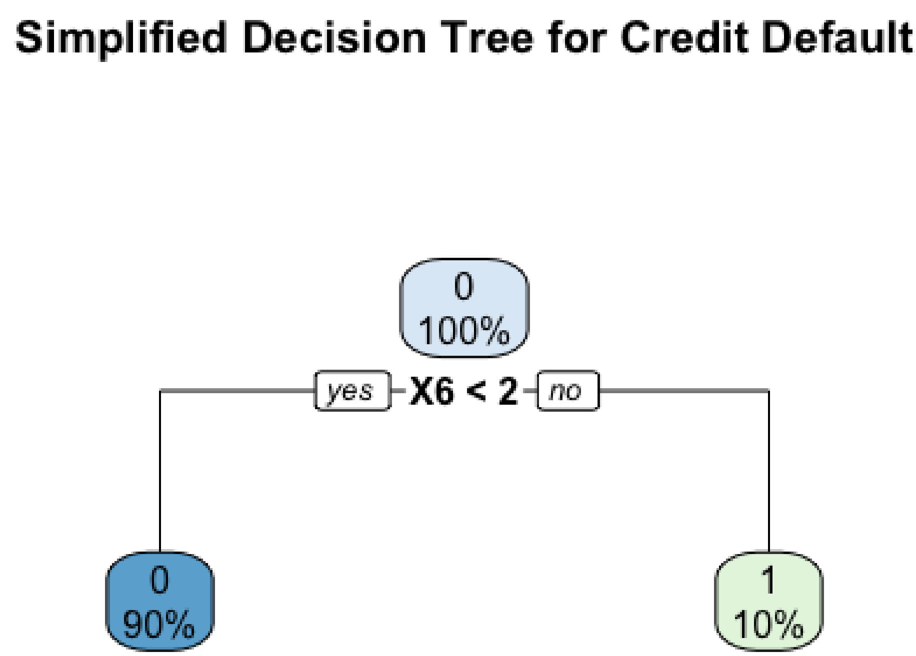
2.1 Decision Tree Model

I will use Decision Trees to analyse the importance of each variable in predicting the the target variable Y. I will then use this information to select the variables that are the most predictive. To prevent overfitting I will use the cp parameter to control the size of the tree and tune it.

```r
# Fitting the decision tree with a smaller complexity parameter
decision_tree_model <- rpart(Y ~ ., data = creditdefaulttrain, method = "clas
s", cp = 0.001)

# Plotting the decision tree using rpart.plot
rpart.plot(decision_tree_model, main = "Decision Tree for Credit Default", ex
tra = 100)
```




```r
# Fitting a simpler decision tree model with a higher cp value
decision_tree_model_simpler <- rpart(Y ~ ., data = creditdefaulttrain, method
= "class", cp = 0.01)

# Plotting the simpler tree
rpart.plot(decision_tree_model_simpler, main = "Simplified Decision Tree for
Credit Default", extra = 100)
```




```r
# Printing a summary of the decision tree model
summary(decision_tree_model)
```

```
## Call:
## rpart(formula = Y ~ ., data = creditdefaulttrain, method = "class",
##     cp = 0.001)
##   n= 15000
##
##             CP nsplit rel error    xerror       xstd
## 1  0.194996986      0 1.0000000 1.0000000 0.01532056
## 2  0.003616637      1 0.8050030 0.8050030 0.01412144
## 3  0.001808318      2 0.8013864 0.8092224 0.01415036
## 4  0.001657625      7 0.7890295 0.8089210 0.01414829
## 5  0.001607394      9 0.7857143 0.8098252 0.01415447
## 6  0.001506932     14 0.7772755 0.8107294 0.01416065
## 7  0.001306008     15 0.7757685 0.8125377 0.01417298
## 8  0.001205546     18 0.7718505 0.8152502 0.01419142
## 9  0.001024714     27 0.7591923 0.8167571 0.01420164
## 10 0.001000000     51 0.7323689 0.8203737 0.01422610
##
## Variable importance
##  X6  X7  X9 X14 X12 X19 X10 X15 X16 X13  X8 X11 X18 X20 X17  X1 X21  X5 X2
2 X23
```

```
##  45  10   4   4   4   4   3   3   3   3   3   2   2   2   2   1   1   1
1   1
##
## Node number 1: 15000 observations,    complexity param=0.194997
##   predicted class=0  expected loss=0.2212  P(node) =1
##     class counts: 11682  3318
##    probabilities: 0.779 0.221
##   left son=2 (13449 obs) right son=3 (1551 obs)
##   Primary splits:
##       X6  < 1.5      to the left,  improve=821.8068, (0 missing)
##       X7  < 1.5      to the left,  improve=602.7759, (0 missing)
##       X8  < 1.5      to the left,  improve=420.1133, (0 missing)
##       X9  < 0.5      to the left,  improve=392.9725, (0 missing)
##       X10 < 1        to the left,  improve=381.0148, (0 missing)
##   Surrogate splits:
##       X9  < 2.5      to the left,  agree=0.902, adj=0.052, (0 split)
##       X10 < 2.5      to the left,  agree=0.901, adj=0.043, (0 split)
##       X7  < 2.5      to the left,  agree=0.900, adj=0.035, (0 split)
##       X8  < 3.5      to the left,  agree=0.900, adj=0.034, (0 split)
##       X11 < 3.5      to the left,  agree=0.899, adj=0.027, (0 split)
##
## Node number 2: 13449 observations,    complexity param=0.001808318
##   predicted class=0  expected loss=0.1649937  P(node) =0.8966
##     class counts: 11230  2219
##    probabilities: 0.835 0.165
##   left son=4 (12285 obs) right son=5 (1164 obs)
##   Primary splits:
##       X7  < 1.5      to the left,  improve=148.47100, (0 missing)
##       X6  < 0.5      to the left,  improve=116.73630, (0 missing)
##       X8  < 0.5      to the left,  improve= 97.01030, (0 missing)
##       X10 < 1        to the left,  improve= 92.42547, (0 missing)
##       X9  < 0.5      to the left,  improve= 90.74921, (0 missing)
##   Surrogate splits:
##       X8  < 2.5      to the left,  agree=0.916, adj=0.024, (0 split)
##       X6  < 0.5      to the left,  agree=0.915, adj=0.017, (0 split)
##       X9  < 2.5      to the left,  agree=0.914, adj=0.011, (0 split)
##       X10 < 3.5      to the left,  agree=0.914, adj=0.004, (0 split)
##
## Node number 3: 1551 observations,    complexity param=0.003616637
##   predicted class=1  expected loss=0.2914249  P(node) =0.1034
##     class counts:   452  1099
##    probabilities: 0.291 0.709
##   left son=6 (52 obs) right son=7 (1499 obs)
##   Primary splits:
##       X12 < 568      to the left,  improve=11.293420, (0 missing)
##       X11 < 1        to the left,  improve=10.145630, (0 missing)
##       X10 < 1        to the left,  improve= 9.771396, (0 missing)
##       X9  < 1        to the left,  improve= 9.473076, (0 missing)
##       X7  < -0.5     to the left,  improve= 6.521702, (0 missing)
##   Surrogate splits:
```

```
##       X13 < 436      to the left,  agree=0.973, adj=0.192, (0 split)
##
## Node number 4: 12285 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.1421245  P(node) =0.819
##     class counts: 10539  1746
##    probabilities: 0.858 0.142
##   left son=8 (7797 obs) right son=9 (4488 obs)
##   Primary splits:
##       X19 < 1500.5   to the right, improve=46.42812, (0 missing)
##       X21 < 785.5    to the right, improve=45.19388, (0 missing)
##       X20 < 458.5    to the right, improve=43.51802, (0 missing)
##       X12 < 795.5    to the right, improve=41.87559, (0 missing)
##       X18 < 1598.5   to the right, improve=37.58896, (0 missing)
##   Surrogate splits:
##       X14 < 1493.5   to the right, agree=0.864, adj=0.629, (0 split)
##       X15 < 1648.5   to the right, agree=0.781, adj=0.401, (0 split)
##       X18 < 1438.5   to the right, agree=0.779, adj=0.395, (0 split)
##       X16 < 1775.5   to the right, agree=0.764, adj=0.355, (0 split)
##       X20 < 1007.5   to the right, agree=0.757, adj=0.336, (0 split)
##
## Node number 5: 1164 observations,    complexity param=0.001808318
##   predicted class=0  expected loss=0.4063574  P(node) =0.0776
##     class counts:   691   473
##    probabilities: 0.594 0.406
##   left son=10 (797 obs) right son=11 (367 obs)
##   Primary splits:
##       X10 < 1        to the left,  improve=10.238700, (0 missing)
##       X1  < 45000    to the right, improve= 9.248521, (0 missing)
##       X11 < 1        to the left,  improve= 8.796260, (0 missing)
##       X22 < 900.5    to the right, improve= 6.916787, (0 missing)
##       X19 < 9285     to the right, improve= 4.930666, (0 missing)
##   Surrogate splits:
##       X11 < 1        to the left,  agree=0.857, adj=0.548, (0 split)
##       X9  < 1        to the left,  agree=0.803, adj=0.376, (0 split)
##       X21 < 16.5     to the right, agree=0.722, adj=0.117, (0 split)
##       X8  < 2.5      to the left,  agree=0.702, adj=0.054, (0 split)
##       X18 < 4481     to the left,  agree=0.698, adj=0.041, (0 split)
##
## Node number 6: 52 observations
##   predicted class=0  expected loss=0.3846154  P(node) =0.003466667
##     class counts:    32    20
##    probabilities: 0.615 0.385
##
## Node number 7: 1499 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.2801868  P(node) =0.09993333
##     class counts:   420  1079
##    probabilities: 0.280 0.720
##   left son=14 (853 obs) right son=15 (646 obs)
##   Primary splits:
##       X10 < 1        to the left,  improve=8.705316, (0 missing)
```

```
##       X11 < 1        to the left,  improve=8.595524, (0 missing)
##       X9  < 1        to the left,  improve=7.610946, (0 missing)
##       X3  < 4        to the right, improve=4.454567, (0 missing)
##       X22 < 23722.5  to the right, improve=3.981169, (0 missing)
##   Surrogate splits:
##       X11 < 1        to the left,  agree=0.874, adj=0.707, (0 split)
##       X9  < 1        to the left,  agree=0.868, adj=0.693, (0 split)
##       X8  < 1        to the left,  agree=0.734, adj=0.384, (0 split)
##       X21 < 17.5     to the right, agree=0.702, adj=0.308, (0 split)
##       X7  < 1        to the left,  agree=0.644, adj=0.173, (0 split)
##
## Node number 8: 7797 observations
##   predicted class=0  expected loss=0.1091445  P(node) =0.5198
##     class counts:  6946   851
##    probabilities: 0.891 0.109
##
## Node number 9: 4488 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.1994207  P(node) =0.2992
##     class counts:  3593   895
##    probabilities: 0.801 0.199
##   left son=18 (4308 obs) right son=19 (180 obs)
##   Primary splits:
##       X9  < 1        to the left,  improve=15.08871, (0 missing)
##       X10 < 1        to the left,  improve=14.94045, (0 missing)
##       X21 < 787.5    to the right, improve=14.93408, (0 missing)
##       X12 < 432.5    to the right, improve=13.69944, (0 missing)
##       X20 < 451      to the right, improve=10.22917, (0 missing)
##   Surrogate splits:
##       X8  < 2.5      to the left,  agree=0.965, adj=0.133, (0 split)
##       X10 < 1        to the left,  agree=0.964, adj=0.100, (0 split)
##       X11 < 3.5      to the left,  agree=0.960, adj=0.006, (0 split)
##
## Node number 10: 797 observations,    complexity param=0.001607394
##   predicted class=0  expected loss=0.3613551  P(node) =0.05313333
##     class counts:   509   288
##    probabilities: 0.639 0.361
##   left son=20 (188 obs) right son=21 (609 obs)
##   Primary splits:
##       X1  < 175000   to the right, improve=8.656123, (0 missing)
##       X19 < 2003     to the right, improve=5.516231, (0 missing)
##       X7  < 2.5      to the left,  improve=4.969240, (0 missing)
##       X23 < 7075     to the right, improve=4.638978, (0 missing)
##       X20 < 1659.5   to the right, improve=4.624459, (0 missing)
##   Surrogate splits:
##       X12 < 156814   to the right, agree=0.817, adj=0.223, (0 split)
##       X13 < 150157.5 to the right, agree=0.816, adj=0.218, (0 split)
##       X14 < 151438.5 to the right, agree=0.811, adj=0.197, (0 split)
##       X20 < 5001.5   to the right, agree=0.807, adj=0.181, (0 split)
##       X15 < 132829.5 to the right, agree=0.806, adj=0.176, (0 split)
##
```

```
## Node number 11: 367 observations,    complexity param=0.001808318
##   predicted class=1  expected loss=0.4959128  P(node) =0.02446667
##     class counts:   182   185
##    probabilities: 0.496 0.504 
##   left son=22 (47 obs) right son=23 (320 obs)
##   Primary splits:
##       X7  < 2.5      to the right, improve=2.887605, (0 missing)
##       X22 < 802      to the right, improve=2.819693, (0 missing)
##       X19 < 8821     to the right, improve=1.762090, (0 missing)
##       X1  < 25000    to the right, improve=1.380161, (0 missing)
##       X5  < 49.5     to the left,  improve=1.357515, (0 missing)
##   Surrogate splits:
##       X8  < 2.5      to the right, agree=0.899, adj=0.213, (0 split)
##       X9  < 4.5      to the right, agree=0.883, adj=0.085, (0 split)
##       X20 < 24050    to the right, agree=0.877, adj=0.043, (0 split)
##       X10 < 3.5      to the right, agree=0.875, adj=0.021, (0 split)
##       X11 < 3.5      to the right, agree=0.875, adj=0.021, (0 split)
## 
## Node number 14: 853 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.3270809  P(node) =0.05686667
##     class counts:   279   574
##    probabilities: 0.327 0.673 
##   left son=28 (74 obs) right son=29 (779 obs)
##   Primary splits:
##       X5  < 50.5     to the right, improve=4.845731, (0 missing)
##       X3  < 4        to the right, improve=3.521830, (0 missing)
##       X22 < 23722.5  to the right, improve=3.050928, (0 missing)
##       X16 < 39524    to the left,  improve=2.942741, (0 missing)
##       X23 < 619      to the right, improve=2.898759, (0 missing)
##   Surrogate splits:
##       X3 < 5.5      to the right, agree=0.914, adj=0.014, (0 split)
## 
## Node number 15: 646 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.2182663  P(node) =0.04306667
##     class counts:   141   505
##    probabilities: 0.218 0.782 
##   left son=30 (10 obs) right son=31 (636 obs)
##   Primary splits:
##       X20 < 12507    to the right, improve=4.714325, (0 missing)
##       X7  < 2.5      to the right, improve=4.278658, (0 missing)
##       X14 < 1932     to the left,  improve=2.219773, (0 missing)
##       X12 < 111326.5 to the right, improve=2.136036, (0 missing)
##       X16 < 51025    to the right, improve=1.860202, (0 missing)
##   Surrogate splits:
##       X17 < 347111   to the right, agree=0.986, adj=0.1, (0 split)
##       X18 < 24852    to the right, agree=0.986, adj=0.1, (0 split)
## 
## Node number 18: 4308 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.1910399  P(node) =0.2872
##     class counts:  3485   823
```

```
##    probabilities: 0.809 0.191
##   left son=36 (2929 obs) right son=37 (1379 obs)
##   Primary splits:
##       X12 < 432.5    to the right, improve=15.61430, (0 missing)
##       X21 < 787.5    to the right, improve=14.21564, (0 missing)
##       X18 < 1596.5   to the right, improve=12.79762, (0 missing)
##       X16 < 786      to the right, improve=12.46099, (0 missing)
##       X6  < 0.5      to the left,  improve=11.30499, (0 missing)
##   Surrogate splits:
##       X6  < 0.5      to the left,  agree=0.815, adj=0.422, (0 split)
##       X7  < -1.5     to the right, agree=0.808, adj=0.400, (0 split)
##       X13 < 434.5    to the right, agree=0.805, adj=0.391, (0 split)
##       X18 < 421      to the right, agree=0.805, adj=0.390, (0 split)
##       X8  < -1.5     to the right, agree=0.744, adj=0.199, (0 split)
##
## Node number 19: 180 observations
##   predicted class=0  expected loss=0.4  P(node) =0.012
##     class counts:   108    72
##    probabilities: 0.600 0.400
##
## Node number 20: 188 observations
##   predicted class=0  expected loss=0.2287234  P(node) =0.01253333
##     class counts:   145    43
##    probabilities: 0.771 0.229
##
## Node number 21: 609 observations,    complexity param=0.001607394
##   predicted class=0  expected loss=0.4022989  P(node) =0.0406
##     class counts:   364   245
##    probabilities: 0.598 0.402
##   left son=42 (285 obs) right son=43 (324 obs)
##   Primary splits:
##       X19 < 1604.5   to the right, improve=5.627494, (0 missing)
##       X20 < 1659.5   to the right, improve=5.346756, (0 missing)
##       X16 < 24659.5  to the right, improve=4.999104, (0 missing)
##       X22 < 900.5    to the right, improve=4.840996, (0 missing)
##       X15 < 27837.5  to the right, improve=4.477308, (0 missing)
##   Surrogate splits:
##       X14 < 30406    to the right, agree=0.695, adj=0.347, (0 split)
##       X12 < 34799    to the right, agree=0.681, adj=0.319, (0 split)
##       X15 < 33006.5  to the right, agree=0.680, adj=0.316, (0 split)
##       X13 < 33808    to the right, agree=0.675, adj=0.305, (0 split)
##       X16 < 20859    to the right, agree=0.667, adj=0.288, (0 split)
##
## Node number 22: 47 observations
##   predicted class=0  expected loss=0.3404255  P(node) =0.003133333
##     class counts:    31    16
##    probabilities: 0.660 0.340
##
## Node number 23: 320 observations,    complexity param=0.001808318
##   predicted class=1  expected loss=0.471875  P(node) =0.02133333
```

```
##     class counts:   151   169
##    probabilities: 0.472 0.528
##   left son=46 (216 obs) right son=47 (104 obs)
##   Primary splits:
##       X22 < 0.5      to the right, improve=3.494462, (0 missing)
##       X1  < 25000    to the right, improve=2.777455, (0 missing)
##       X19 < 3450.5   to the right, improve=2.523275, (0 missing)
##       X5  < 49.5     to the left,  improve=2.456642, (0 missing)
##       X15 < 993      to the right, improve=1.549432, (0 missing)
##   Surrogate splits:
##       X20 < 1.5      to the right, agree=0.769, adj=0.288, (0 split)
##       X21 < 446      to the left,  agree=0.734, adj=0.183, (0 split)
##       X17 < 95.5     to the right, agree=0.691, adj=0.048, (0 split)
##       X11 < 2.5      to the left,  agree=0.688, adj=0.038, (0 split)
##       X23 < 5914     to the left,  agree=0.684, adj=0.029, (0 split)
##
## Node number 28: 74 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.5  P(node) =0.004933333
##     class counts:    37    37
##    probabilities: 0.500 0.500
##   left son=56 (66 obs) right son=57 (8 obs)
##   Primary splits:
##       X12 < 7150     to the right, improve=2.522727, (0 missing)
##       X13 < 6996.5   to the right, improve=2.522727, (0 missing)
##       X15 < 6124.5   to the right, improve=2.522727, (0 missing)
##       X21 < 3658.5   to the left,  improve=2.286255, (0 missing)
##       X19 < 1119.5   to the right, improve=2.069930, (0 missing)
##   Surrogate splits:
##       X13 < 6996.5   to the right, agree=0.973, adj=0.750, (0 split)
##       X14 < 7246.5   to the right, agree=0.959, adj=0.625, (0 split)
##       X15 < 6124.5   to the right, agree=0.946, adj=0.500, (0 split)
##       X16 < 5004.5   to the right, agree=0.932, adj=0.375, (0 split)
##       X8  < -0.5     to the right, agree=0.905, adj=0.125, (0 split)
##
## Node number 29: 779 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.3106547  P(node) =0.05193333
##     class counts:   242   537
##    probabilities: 0.311 0.689
##   left son=58 (685 obs) right son=59 (94 obs)
##   Primary splits:
##       X5  < 43.5     to the left,  improve=3.036019, (0 missing)
##       X22 < 23722.5  to the right, improve=2.803659, (0 missing)
##       X23 < 648      to the right, improve=2.743508, (0 missing)
##       X3  < 4        to the right, improve=2.367327, (0 missing)
##       X19 < 13093.5  to the left,  improve=2.186425, (0 missing)
##   Surrogate splits:
##       X19 < 71391.5  to the left,  agree=0.882, adj=0.021, (0 split)
##       X20 < 18510.5  to the left,  agree=0.881, adj=0.011, (0 split)
##
## Node number 30: 10 observations
```

```
##   predicted class=0  expected loss=0.3  P(node) =0.0006666667
##     class counts:     7     3
##    probabilities: 0.700 0.300 
## 
## Node number 31: 636 observations
##   predicted class=1  expected loss=0.2106918  P(node) =0.0424
##     class counts:   134   502
##    probabilities: 0.211 0.789 
## 
## Node number 36: 2929 observations
##   predicted class=0  expected loss=0.16183  P(node) =0.1952667
##     class counts:  2455   474
##    probabilities: 0.838 0.162 
## 
## Node number 37: 1379 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.2530819  P(node) =0.09193333
##     class counts:  1030   349
##    probabilities: 0.747 0.253 
##   left son=74 (1323 obs) right son=75 (56 obs)
##   Primary splits:
##       X5  < 54.5     to the left,  improve=11.831040, (0 missing)
##       X21 < 785      to the right, improve= 7.720639, (0 missing)
##       X16 < 985.5    to the right, improve= 5.910745, (0 missing)
##       X20 < 1.5      to the right, improve= 5.482475, (0 missing)
##       X17 < 533      to the right, improve= 5.183766, (0 missing)
## 
## Node number 42: 285 observations
##   predicted class=0  expected loss=0.3298246  P(node) =0.019
##     class counts:   191    94
##    probabilities: 0.670 0.330 
## 
## Node number 43: 324 observations,    complexity param=0.001607394
##   predicted class=0  expected loss=0.4660494  P(node) =0.0216
##     class counts:   173   151
##    probabilities: 0.534 0.466 
##   left son=86 (140 obs) right son=87 (184 obs)
##   Primary splits:
##       X22 < 902.5    to the right, improve=5.105881, (0 missing)
##       X23 < 7191.5   to the right, improve=4.482386, (0 missing)
##       X18 < 6        to the right, improve=3.750506, (0 missing)
##       X19 < 1526.5   to the left,  improve=3.310229, (0 missing)
##       X5  < 37.5     to the right, improve=2.900835, (0 missing)
##   Surrogate splits:
##       X17 < 20013    to the right, agree=0.750, adj=0.421, (0 split)
##       X21 < 921      to the right, agree=0.741, adj=0.400, (0 split)
##       X16 < 16737    to the right, agree=0.707, adj=0.321, (0 split)
##       X23 < 843      to the right, agree=0.685, adj=0.271, (0 split)
##       X14 < 24681    to the right, agree=0.679, adj=0.257, (0 split)
## 
## Node number 46: 216 observations,    complexity param=0.001808318
```

```
##   predicted class=0  expected loss=0.4768519  P(node) =0.0144
##     class counts:   113   103
##    probabilities: 0.523 0.477 
##   left son=92 (183 obs) right son=93 (33 obs)
##   Primary splits:
##       X1  < 25000    to the right, improve=3.774480, (0 missing)
##       X23 < 2506.5   to the right, improve=2.373722, (0 missing)
##       X16 < 549.5    to the right, improve=2.092510, (0 missing)
##       X20 < 2119.5   to the right, improve=2.003229, (0 missing)
##       X12 < 1015     to the right, improve=1.700886, (0 missing)
## 
## Node number 47: 104 observations,    complexity param=0.001657625
##   predicted class=1  expected loss=0.3653846  P(node) =0.006933333
##     class counts:    38    66
##    probabilities: 0.365 0.635 
##   left son=94 (42 obs) right son=95 (62 obs)
##   Primary splits:
##       X19 < 2550     to the right, improve=4.679310, (0 missing)
##       X20 < 1250     to the left,  improve=3.969072, (0 missing)
##       X8  < 1        to the right, improve=3.284533, (0 missing)
##       X23 < 890.5    to the left,  improve=2.958122, (0 missing)
##       X9  < 1        to the right, improve=2.954173, (0 missing)
##   Surrogate splits:
##       X21 < 3450     to the right, agree=0.721, adj=0.310, (0 split)
##       X14 < 69464    to the right, agree=0.692, adj=0.238, (0 split)
##       X12 < 67055.5  to the right, agree=0.683, adj=0.214, (0 split)
##       X15 < 70722    to the right, agree=0.683, adj=0.214, (0 split)
##       X23 < 2910.5   to the right, agree=0.683, adj=0.214, (0 split)
## 
## Node number 56: 66 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.4545455  P(node) =0.0044
##     class counts:    36    30
##    probabilities: 0.545 0.455 
##   left son=112 (53 obs) right son=113 (13 obs)
##   Primary splits:
##       X21 < 3658.5   to the left,  improve=3.206228, (0 missing)
##       X17 < 20021    to the left,  improve=3.155844, (0 missing)
##       X22 < 2039     to the left,  improve=2.801347, (0 missing)
##       X12 < 18315    to the left,  improve=2.051449, (0 missing)
##       X18 < 1552.5   to the left,  improve=1.850350, (0 missing)
##   Surrogate splits:
##       X12 < 98635    to the left,  agree=0.939, adj=0.692, (0 split)
##       X18 < 4350     to the left,  agree=0.939, adj=0.692, (0 split)
##       X1  < 115000   to the left,  agree=0.924, adj=0.615, (0 split)
##       X13 < 98319.5  to the left,  agree=0.924, adj=0.615, (0 split)
##       X14 < 94805    to the left,  agree=0.924, adj=0.615, (0 split)
## 
## Node number 57: 8 observations
##   predicted class=1  expected loss=0.125  P(node) =0.0005333333
##     class counts:     1     7
```

```
##    probabilities: 0.125 0.875
##
## Node number 58: 685 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.3270073  P(node) =0.04566667
##     class counts:   224   461
##    probabilities: 0.327 0.673
##   left son=116 (8 obs) right son=117 (677 obs)
##   Primary splits:
##       X3  < 4        to the right, improve=2.896594, (0 missing)
##       X22 < 23722.5  to the right, improve=2.480815, (0 missing)
##       X18 < 2.5      to the right, improve=2.395815, (0 missing)
##       X23 < 652      to the right, improve=2.339889, (0 missing)
##       X16 < 41320    to the left,  improve=2.057868, (0 missing)
##
## Node number 59: 94 observations
##   predicted class=1  expected loss=0.1914894  P(node) =0.006266667
##     class counts:    18    76
##    probabilities: 0.191 0.809
##
## Node number 74: 1323 observations
##   predicted class=0  expected loss=0.239607  P(node) =0.0882
##     class counts:  1006   317
##    probabilities: 0.760 0.240
##
## Node number 75: 56 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.4285714  P(node) =0.003733333
##     class counts:    24    32
##    probabilities: 0.429 0.571
##   left son=150 (25 obs) right son=151 (31 obs)
##   Primary splits:
##       X11 < -1.5     to the right, improve=5.709862, (0 missing)
##       X13 < 88       to the right, improve=5.709862, (0 missing)
##       X18 < 88       to the right, improve=5.709862, (0 missing)
##       X16 < 119.5    to the right, improve=5.568229, (0 missing)
##       X10 < -1.5     to the right, improve=4.647823, (0 missing)
##   Surrogate splits:
##       X10 < -1.5     to the right, agree=0.911, adj=0.80, (0 split)
##       X16 < 34       to the right, agree=0.875, adj=0.72, (0 split)
##       X7  < -1.5     to the right, agree=0.857, adj=0.68, (0 split)
##       X8  < -1.5     to the right, agree=0.857, adj=0.68, (0 split)
##       X9  < -1.5     to the right, agree=0.839, adj=0.64, (0 split)
##
## Node number 86: 140 observations,    complexity param=0.001306008
##   predicted class=0  expected loss=0.3642857  P(node) =0.009333333
##     class counts:    89    51
##    probabilities: 0.636 0.364
##   left son=172 (48 obs) right son=173 (92 obs)
##   Primary splits:
##       X5  < 37.5     to the right, improve=3.553002, (0 missing)
##       X23 < 4418     to the right, improve=2.366782, (0 missing)
```

```
##       X18 < 67       to the right, improve=2.029569, (0 missing)
##       X17 < 18385.5  to the left,  improve=1.939326, (0 missing)
##       X1  < 145000   to the left,  improve=1.805263, (0 missing)
##   Surrogate splits:
##       X4  < 1.5      to the left,  agree=0.721, adj=0.187, (0 split)
##       X19 < 1538.5   to the right, agree=0.679, adj=0.063, (0 split)
##       X12 < 137119   to the right, agree=0.671, adj=0.042, (0 split)
##       X13 < 139399.5 to the right, agree=0.671, adj=0.042, (0 split)
##       X22 < 964      to the left,  agree=0.671, adj=0.042, (0 split)
##
## Node number 87: 184 observations,    complexity param=0.001607394
##   predicted class=1  expected loss=0.4565217  P(node) =0.01226667
##     class counts:    84   100
##    probabilities: 0.457 0.543
##   left son=174 (175 obs) right son=175 (9 obs)
##   Primary splits:
##       X22 < 726.5    to the left,  improve=2.257999, (0 missing)
##       X23 < 297      to the right, improve=2.130435, (0 missing)
##       X16 < 4651.5   to the left,  improve=1.983038, (0 missing)
##       X21 < 787      to the left,  improve=1.677265, (0 missing)
##       X1  < 55000    to the right, improve=1.535328, (0 missing)
##
## Node number 92: 183 observations,    complexity param=0.001506932
##   predicted class=0  expected loss=0.4371585  P(node) =0.0122
##     class counts:   103    80
##    probabilities: 0.563 0.437
##   left son=184 (176 obs) right son=185 (7 obs)
##   Primary splits:
##       X12 < 1015     to the right, improve=2.567632, (0 missing)
##       X16 < 549.5    to the right, improve=2.567632, (0 missing)
##       X15 < 1487     to the right, improve=2.196407, (0 missing)
##       X22 < 1750     to the left,  improve=1.920260, (0 missing)
##       X23 < 2506.5   to the right, improve=1.898963, (0 missing)
##   Surrogate splits:
##       X15 < 1487     to the right, agree=0.989, adj=0.714, (0 split)
##       X17 < 1109     to the right, agree=0.989, adj=0.714, (0 split)
##       X13 < 524.5    to the right, agree=0.984, adj=0.571, (0 split)
##       X14 < 132      to the right, agree=0.984, adj=0.571, (0 split)
##       X16 < 549.5    to the right, agree=0.978, adj=0.429, (0 split)
##
## Node number 93: 33 observations,    complexity param=0.001205546
##   predicted class=1  expected loss=0.3030303  P(node) =0.0022
##     class counts:    10    23
##    probabilities: 0.303 0.697
##   left son=186 (10 obs) right son=187 (23 obs)
##   Primary splits:
##       X15 < 9753.5   to the left,  improve=4.522003, (0 missing)
##       X12 < 8447     to the left,  improve=4.185548, (0 missing)
##       X14 < 10227.5  to the left,  improve=3.503304, (0 missing)
##       X1  < 15000    to the left,  improve=3.005328, (0 missing)
```

```
##       X16 < 9124     to the left,  improve=3.005328, (0 missing)
##   Surrogate splits:
##       X16 < 9445     to the left,  agree=0.939, adj=0.8, (0 split)
##       X12 < 8447     to the left,  agree=0.909, adj=0.7, (0 split)
##       X14 < 8848.5   to the left,  agree=0.909, adj=0.7, (0 split)
##       X1  < 15000    to the left,  agree=0.848, adj=0.5, (0 split)
##       X13 < 8167.5   to the left,  agree=0.848, adj=0.5, (0 split)
##
## Node number 94: 42 observations,    complexity param=0.001657625
##   predicted class=0  expected loss=0.452381  P(node) =0.0028
##     class counts:    23    19
##    probabilities: 0.548 0.452
##   left son=188 (19 obs) right son=189 (23 obs)
##   Primary splits:
##       X23 < 1700     to the left,  improve=4.058952, (0 missing)
##       X21 < 1680     to the left,  improve=2.972487, (0 missing)
##       X5  < 45.5     to the left,  improve=1.750700, (0 missing)
##       X13 < 11325.5  to the right, improve=1.750700, (0 missing)
##       X3  < 2.5      to the right, improve=1.609524, (0 missing)
##   Surrogate splits:
##       X21 < 2497.5   to the left,  agree=0.881, adj=0.737, (0 split)
##       X17 < 29293    to the left,  agree=0.810, adj=0.579, (0 split)
##       X16 < 29908.5  to the left,  agree=0.786, adj=0.526, (0 split)
##       X1  < 45000    to the left,  agree=0.738, adj=0.421, (0 split)
##       X14 < 25689    to the left,  agree=0.738, adj=0.421, (0 split)
##
## Node number 95: 62 observations
##   predicted class=1  expected loss=0.2419355  P(node) =0.004133333
##     class counts:    15    47
##    probabilities: 0.242 0.758
##
## Node number 112: 53 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.3773585  P(node) =0.003533333
##     class counts:    33    20
##    probabilities: 0.623 0.377
##   left son=224 (14 obs) right son=225 (39 obs)
##   Primary splits:
##       X19 < 2114     to the right, improve=5.418481, (0 missing)
##       X20 < 2024.5   to the right, improve=1.896358, (0 missing)
##       X23 < 12       to the right, improve=1.814751, (0 missing)
##       X17 < 20021    to the left,  improve=1.422902, (0 missing)
##       X18 < 1552.5   to the left,  improve=1.422902, (0 missing)
##   Surrogate splits:
##       X14 < 48614.5  to the right, agree=0.849, adj=0.429, (0 split)
##       X12 < 84040    to the right, agree=0.811, adj=0.286, (0 split)
##       X13 < 82758    to the right, agree=0.811, adj=0.286, (0 split)
##       X20 < 2024.5   to the right, agree=0.811, adj=0.286, (0 split)
##       X15 < 45532.5  to the right, agree=0.792, adj=0.214, (0 split)
##
## Node number 113: 13 observations
```

```
##   predicted class=1  expected loss=0.2307692  P(node) =0.0008666667
##     class counts:     3    10
##    probabilities: 0.231 0.769
##
## Node number 116: 8 observations
##   predicted class=0  expected loss=0.25  P(node) =0.0005333333
##     class counts:     6     2
##    probabilities: 0.750 0.250
##
## Node number 117: 677 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.3220089  P(node) =0.04513333
##     class counts:   218   459
##    probabilities: 0.322 0.678
##   left son=234 (473 obs) right son=235 (204 obs)
##   Primary splits:
##       X23 < 652      to the right, improve=2.629796, (0 missing)
##       X22 < 23722.5  to the right, improve=2.581100, (0 missing)
##       X3  < 1.5      to the right, improve=2.505626, (0 missing)
##       X18 < 2.5      to the right, improve=2.374325, (0 missing)
##       X16 < 41320    to the left,  improve=2.204392, (0 missing)
##   Surrogate splits:
##       X21 < 593.5    to the right, agree=0.799, adj=0.333, (0 split)
##       X17 < 10649    to the right, agree=0.786, adj=0.289, (0 split)
##       X16 < 12905    to the right, agree=0.777, adj=0.260, (0 split)
##       X11 < -1.5     to the right, agree=0.762, adj=0.211, (0 split)
##       X22 < 634      to the right, agree=0.761, adj=0.206, (0 split)
##
## Node number 150: 25 observations
##   predicted class=0  expected loss=0.32  P(node) =0.001666667
##     class counts:    17     8
##    probabilities: 0.680 0.320
##
## Node number 151: 31 observations
##   predicted class=1  expected loss=0.2258065  P(node) =0.002066667
##     class counts:     7    24
##    probabilities: 0.226 0.774
##
## Node number 172: 48 observations
##   predicted class=0  expected loss=0.2083333  P(node) =0.0032
##     class counts:    38    10
##    probabilities: 0.792 0.208
##
## Node number 173: 92 observations,    complexity param=0.001306008
##   predicted class=0  expected loss=0.4456522  P(node) =0.006133333
##     class counts:    51    41
##    probabilities: 0.554 0.446
##   left son=346 (21 obs) right son=347 (71 obs)
##   Primary splits:
##       X23 < 2053     to the right, improve=3.543712, (0 missing)
##       X22 < 1854     to the right, improve=3.002676, (0 missing)
```

```
##       X5  < 33.5     to the left,  improve=2.966478, (0 missing)
##       X18 < 1132     to the right, improve=2.887416, (0 missing)
##       X20 < 1571.5   to the right, improve=2.652355, (0 missing)
##   Surrogate splits:
##       X20 < 3099     to the right, agree=0.859, adj=0.381, (0 split)
##       X16 < 46474.5  to the right, agree=0.848, adj=0.333, (0 split)
##       X17 < 47831    to the right, agree=0.848, adj=0.333, (0 split)
##       X15 < 85336    to the right, agree=0.837, adj=0.286, (0 split)
##       X21 < 3119     to the right, agree=0.837, adj=0.286, (0 split)
##
## Node number 174: 175 observations,    complexity param=0.001607394
##   predicted class=1  expected loss=0.4742857  P(node) =0.01166667
##     class counts:    83    92
##    probabilities: 0.474 0.526
##   left son=348 (86 obs) right son=349 (89 obs)
##   Primary splits:
##       X23 < 297      to the right, improve=3.083309, (0 missing)
##       X13 < 511      to the right, improve=1.665532, (0 missing)
##       X21 < 3400     to the left,  improve=1.601905, (0 missing)
##       X22 < 568      to the right, improve=1.485132, (0 missing)
##       X18 < 1974.5   to the left,  improve=1.371797, (0 missing)
##   Surrogate splits:
##       X17 < 75       to the right, agree=0.777, adj=0.547, (0 split)
##       X21 < 279.5    to the right, agree=0.709, adj=0.407, (0 split)
##       X20 < 269.5    to the right, agree=0.703, adj=0.395, (0 split)
##       X16 < 5655     to the right, agree=0.686, adj=0.360, (0 split)
##       X11 < -0.5     to the right, agree=0.674, adj=0.337, (0 split)
##
## Node number 175: 9 observations
##   predicted class=1  expected loss=0.1111111  P(node) =0.0006
##     class counts:     1     8
##    probabilities: 0.111 0.889
##
## Node number 184: 176 observations,    complexity param=0.001205546
##   predicted class=0  expected loss=0.4204545  P(node) =0.01173333
##     class counts:   102    74
##    probabilities: 0.580 0.420
##   left son=368 (66 obs) right son=369 (110 obs)
##   Primary splits:
##       X22 < 1750     to the left,  improve=2.912121, (0 missing)
##       X12 < 46115    to the left,  improve=2.300257, (0 missing)
##       X14 < 22888.5  to the left,  improve=2.132292, (0 missing)
##       X13 < 47301    to the left,  improve=2.114108, (0 missing)
##       X23 < 2506.5   to the right, improve=1.738726, (0 missing)
##   Surrogate splits:
##       X14 < 46003.5  to the left,  agree=0.756, adj=0.348, (0 split)
##       X17 < 47221.5  to the left,  agree=0.756, adj=0.348, (0 split)
##       X16 < 40554.5  to the left,  agree=0.750, adj=0.333, (0 split)
##       X13 < 44633    to the left,  agree=0.744, adj=0.318, (0 split)
##       X15 < 44028.5  to the left,  agree=0.739, adj=0.303, (0 split)
```

```
##
## Node number 185: 7 observations
##   predicted class=1  expected loss=0.1428571  P(node) =0.0004666667
##     class counts:     1     6
##    probabilities: 0.143 0.857
##
## Node number 186: 10 observations
##   predicted class=0  expected loss=0.3  P(node) =0.0006666667
##     class counts:     7     3
##    probabilities: 0.700 0.300
##
## Node number 187: 23 observations
##   predicted class=1  expected loss=0.1304348  P(node) =0.001533333
##     class counts:     3    20
##    probabilities: 0.130 0.870
##
## Node number 188: 19 observations
##   predicted class=0  expected loss=0.2105263  P(node) =0.001266667
##     class counts:    15     4
##    probabilities: 0.789 0.211
##
## Node number 189: 23 observations
##   predicted class=1  expected loss=0.3478261  P(node) =0.001533333
##     class counts:     8    15
##    probabilities: 0.348 0.652
##
## Node number 224: 14 observations
##   predicted class=0  expected loss=0  P(node) =0.0009333333
##     class counts:    14     0
##    probabilities: 1.000 0.000
##
## Node number 225: 39 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.4871795  P(node) =0.0026
##     class counts:    19    20
##    probabilities: 0.487 0.513
##   left son=450 (16 obs) right son=451 (23 obs)
##   Primary splits:
##       X17 < 18616.5  to the left,  improve=3.748049, (0 missing)
##       X12 < 18315    to the left,  improve=3.102564, (0 missing)
##       X18 < 1552.5   to the left,  improve=2.876653, (0 missing)
##       X13 < 50145    to the left,  improve=2.640405, (0 missing)
##       X14 < 42447.5  to the left,  improve=2.640405, (0 missing)
##   Surrogate splits:
##       X16 < 17844.5  to the left,  agree=0.897, adj=0.750, (0 split)
##       X15 < 17434    to the left,  agree=0.872, adj=0.688, (0 split)
##       X14 < 17162.5  to the left,  agree=0.821, adj=0.562, (0 split)
##       X13 < 16076.5  to the left,  agree=0.795, adj=0.500, (0 split)
##       X12 < 15190.5  to the left,  agree=0.769, adj=0.438, (0 split)
##
## Node number 234: 473 observations,    complexity param=0.001024714
```

```
##   predicted class=1  expected loss=0.3509514  P(node) =0.03153333
##     class counts:   166   307
##    probabilities: 0.351 0.649
##   left son=468 (242 obs) right son=469 (231 obs)
##   Primary splits:
##       X16 < 41399    to the left,  improve=3.843058, (0 missing)
##       X17 < 29756    to the left,  improve=3.680316, (0 missing)
##       X21 < 1466.5   to the left,  improve=2.651556, (0 missing)
##       X15 < 38199    to the left,  improve=2.388666, (0 missing)
##       X23 < 3027     to the left,  improve=2.232082, (0 missing)
##   Surrogate splits:
##       X17 < 39931    to the left,  agree=0.962, adj=0.922, (0 split)
##       X15 < 40884.5  to the left,  agree=0.941, adj=0.879, (0 split)
##       X14 < 51861.5  to the left,  agree=0.899, adj=0.792, (0 split)
##       X13 < 57016.5  to the left,  agree=0.884, adj=0.762, (0 split)
##       X12 < 52989    to the left,  agree=0.869, adj=0.732, (0 split)
##
## Node number 235: 204 observations
##   predicted class=1  expected loss=0.254902  P(node) =0.0136
##     class counts:    52   152
##    probabilities: 0.255 0.745
##
## Node number 346: 21 observations
##   predicted class=0  expected loss=0.1904762  P(node) =0.0014
##     class counts:    17     4
##    probabilities: 0.810 0.190
##
## Node number 347: 71 observations,    complexity param=0.001306008
##   predicted class=1  expected loss=0.4788732  P(node) =0.004733333
##     class counts:    34    37
##    probabilities: 0.479 0.521
##   left son=694 (36 obs) right son=695 (35 obs)
##   Primary splits:
##       X18 < 1132     to the right, improve=3.739794, (0 missing)
##       X20 < 1571.5   to the right, improve=2.722894, (0 missing)
##       X13 < 17776.5  to the right, improve=2.404141, (0 missing)
##       X12 < 18517    to the right, improve=2.292101, (0 missing)
##       X16 < 41666    to the right, improve=2.222334, (0 missing)
##   Surrogate splits:
##       X8  < 1        to the right, agree=0.803, adj=0.600, (0 split)
##       X19 < 924      to the left,  agree=0.761, adj=0.514, (0 split)
##       X14 < 38395.5  to the right, agree=0.746, adj=0.486, (0 split)
##       X15 < 20245.5  to the right, agree=0.746, adj=0.486, (0 split)
##       X12 < 22708    to the right, agree=0.718, adj=0.429, (0 split)
##
## Node number 348: 86 observations,    complexity param=0.001205546
##   predicted class=0  expected loss=0.4302326  P(node) =0.005733333
##     class counts:    49    37
##    probabilities: 0.570 0.430
##   left son=696 (8 obs) right son=697 (78 obs)
```

```
##   Primary splits:
##       X17 < 75       to the left,  improve=3.265355, (0 missing)
##       X1  < 110000   to the right, improve=3.203166, (0 missing)
##       X14 < 5751     to the left,  improve=2.550987, (0 missing)
##       X15 < 5848.5   to the left,  improve=2.550987, (0 missing)
##       X23 < 455.5    to the left,  improve=2.339819, (0 missing)
##   Surrogate splits:
##       X16 < 158      to the left,  agree=0.942, adj=0.375, (0 split)
##       X11 < -1.5     to the left,  agree=0.930, adj=0.250, (0 split)
##
## Node number 349: 89 observations
##   predicted class=1  expected loss=0.3820225  P(node) =0.005933333
##     class counts:    34    55
##    probabilities: 0.382 0.618
##
## Node number 368: 66 observations
##   predicted class=0  expected loss=0.3030303  P(node) =0.0044
##     class counts:    46    20
##    probabilities: 0.697 0.303
##
## Node number 369: 110 observations,    complexity param=0.001205546
##   predicted class=0  expected loss=0.4909091  P(node) =0.007333333
##     class counts:    56    54
##    probabilities: 0.509 0.491
##   left son=738 (98 obs) right son=739 (12 obs)
##   Primary splits:
##       X12 < 126319.5 to the left,  improve=3.158689, (0 missing)
##       X2  < 1.5      to the right, improve=3.078286, (0 missing)
##       X13 < 124767.5 to the left,  improve=2.788366, (0 missing)
##       X14 < 143290   to the left,  improve=1.613281, (0 missing)
##       X15 < 126655   to the left,  improve=1.600866, (0 missing)
##   Surrogate splits:
##       X13 < 126312.5 to the left,  agree=0.991, adj=0.917, (0 split)
##       X14 < 127320   to the left,  agree=0.973, adj=0.750, (0 split)
##       X15 < 129602.5 to the left,  agree=0.973, adj=0.750, (0 split)
##       X16 < 125226   to the left,  agree=0.964, adj=0.667, (0 split)
##       X17 < 136781   to the left,  agree=0.964, adj=0.667, (0 split)
##
## Node number 450: 16 observations
##   predicted class=0  expected loss=0.25  P(node) =0.001066667
##     class counts:    12     4
##    probabilities: 0.750 0.250
##
## Node number 451: 23 observations
##   predicted class=1  expected loss=0.3043478  P(node) =0.001533333
##     class counts:     7    16
##    probabilities: 0.304 0.696
##
## Node number 468: 242 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.4132231  P(node) =0.01613333
```

```
##     class counts:   100   142
##    probabilities: 0.413 0.587
##   left son=936 (127 obs) right son=937 (115 obs)
##   Primary splits:
##       X18 < 1611     to the right, improve=6.058145, (0 missing)
##       X21 < 4366     to the right, improve=4.945462, (0 missing)
##       X12 < 84871.5  to the right, improve=4.943303, (0 missing)
##       X13 < 49281    to the right, improve=4.046281, (0 missing)
##       X14 < 47506.5  to the right, improve=2.957803, (0 missing)
##   Surrogate splits:
##       X13 < 21582    to the right, agree=0.649, adj=0.261, (0 split)
##       X14 < 21205    to the right, agree=0.628, adj=0.217, (0 split)
##       X7  < 1        to the left,  agree=0.624, adj=0.209, (0 split)
##       X1  < 45000    to the right, agree=0.612, adj=0.183, (0 split)
##       X12 < 20623.5  to the right, agree=0.612, adj=0.183, (0 split)
##
## Node number 469: 231 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.2857143  P(node) =0.0154
##     class counts:    66   165
##    probabilities: 0.286 0.714
##   left son=938 (12 obs) right son=939 (219 obs)
##   Primary splits:
##       X21 < 1472     to the left,  improve=3.673842, (0 missing)
##       X13 < 138542.5 to the right, improve=3.155844, (0 missing)
##       X12 < 121545   to the right, improve=3.064757, (0 missing)
##       X22 < 10938    to the right, improve=2.842835, (0 missing)
##       X14 < 143386.5 to the right, improve=2.812987, (0 missing)
##   Surrogate splits:
##       X17 < 26605.5  to the left,  agree=0.952, adj=0.083, (0 split)
##       X23 < 999.5    to the left,  agree=0.952, adj=0.083, (0 split)
##
## Node number 694: 36 observations
##   predicted class=0  expected loss=0.3611111  P(node) =0.0024
##     class counts:    23    13
##    probabilities: 0.639 0.361
##
## Node number 695: 35 observations
##   predicted class=1  expected loss=0.3142857  P(node) =0.002333333
##     class counts:    11    24
##    probabilities: 0.314 0.686
##
## Node number 696: 8 observations
##   predicted class=0  expected loss=0  P(node) =0.0005333333
##     class counts:     8     0
##    probabilities: 1.000 0.000
##
## Node number 697: 78 observations,    complexity param=0.001205546
##   predicted class=0  expected loss=0.474359  P(node) =0.0052
##     class counts:    41    37
##    probabilities: 0.526 0.474
```

```
##   left son=1394 (54 obs) right son=1395 (24 obs)
##   Primary splits:
##       X20 < 1284     to the left,  improve=2.564103, (0 missing)
##       X12 < 17543.5  to the right, improve=2.483643, (0 missing)
##       X1  < 110000   to the right, improve=2.191873, (0 missing)
##       X23 < 442.5    to the left,  improve=2.191873, (0 missing)
##       X13 < 18357    to the right, improve=1.282051, (0 missing)
##   Surrogate splits:
##       X16 < 30503    to the left,  agree=0.756, adj=0.208, (0 split)
##       X15 < 30539.5  to the left,  agree=0.744, adj=0.167, (0 split)
##       X17 < 19940    to the left,  agree=0.744, adj=0.167, (0 split)
##       X21 < 2586.5   to the left,  agree=0.744, adj=0.167, (0 split)
##       X5  < 23.5     to the right, agree=0.718, adj=0.083, (0 split)
##
## Node number 738: 98 observations,    complexity param=0.001205546
##   predicted class=0  expected loss=0.4489796  P(node) =0.006533333
##     class counts:    54    44
##    probabilities: 0.551 0.449
##   left son=1476 (34 obs) right son=1477 (64 obs)
##   Primary splits:
##       X22 < 3650     to the right, improve=3.535752, (0 missing)
##       X14 < 93309    to the right, improve=3.054792, (0 missing)
##       X23 < 4615     to the right, improve=2.712523, (0 missing)
##       X2  < 1.5      to the right, improve=2.520811, (0 missing)
##       X16 < 118910   to the right, improve=1.828685, (0 missing)
##   Surrogate splits:
##       X14 < 93309    to the right, agree=0.806, adj=0.441, (0 split)
##       X15 < 79182    to the right, agree=0.806, adj=0.441, (0 split)
##       X16 < 88487.5  to the right, agree=0.806, adj=0.441, (0 split)
##       X17 < 79924    to the right, agree=0.806, adj=0.441, (0 split)
##       X13 < 85914.5  to the right, agree=0.796, adj=0.412, (0 split)
##
## Node number 739: 12 observations
##   predicted class=1  expected loss=0.1666667  P(node) =0.0008
##     class counts:     2    10
##    probabilities: 0.167 0.833
##
## Node number 936: 127 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.480315  P(node) =0.008466667
##     class counts:    66    61
##    probabilities: 0.520 0.480
##   left son=1872 (69 obs) right son=1873 (58 obs)
##   Primary splits:
##       X12 < 28922    to the right, improve=3.237157, (0 missing)
##       X1  < 25000    to the right, improve=3.233967, (0 missing)
##       X14 < 47678    to the right, improve=3.141899, (0 missing)
##       X21 < 4066     to the right, improve=3.087135, (0 missing)
##       X13 < 49204    to the right, improve=2.935578, (0 missing)
##   Surrogate splits:
##       X13 < 29439.5  to the right, agree=0.937, adj=0.862, (0 split)
```

```
##       X14 < 29780    to the right, agree=0.890, adj=0.759, (0 split)
##       X15 < 27600.5  to the right, agree=0.748, adj=0.448, (0 split)
##       X1  < 45000    to the right, agree=0.732, adj=0.414, (0 split)
##       X16 < 19623.5  to the right, agree=0.661, adj=0.259, (0 split)
##
## Node number 937: 115 observations
##   predicted class=1  expected loss=0.2956522  P(node) =0.007666667
##     class counts:    34    81
##    probabilities: 0.296 0.704
##
## Node number 938: 12 observations
##   predicted class=0  expected loss=0.3333333  P(node) =0.0008
##     class counts:     8     4
##    probabilities: 0.667 0.333
##
## Node number 939: 219 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.2648402  P(node) =0.0146
##     class counts:    58   161
##    probabilities: 0.265 0.735
##   left son=1878 (72 obs) right son=1879 (147 obs)
##   Primary splits:
##       X13 < 140102.5 to the right, improve=4.081827, (0 missing)
##       X12 < 121545   to the right, improve=4.019450, (0 missing)
##       X14 < 142162.5 to the right, improve=3.616442, (0 missing)
##       X15 < 127474.5 to the right, improve=2.800237, (0 missing)
##       X23 < 1737.5   to the right, improve=2.585470, (0 missing)
##   Surrogate splits:
##       X12 < 136705.5 to the right, agree=0.982, adj=0.944, (0 split)
##       X14 < 138505.5 to the right, agree=0.968, adj=0.903, (0 split)
##       X15 < 132993.5 to the right, agree=0.918, adj=0.750, (0 split)
##       X16 < 133262   to the right, agree=0.895, adj=0.681, (0 split)
##       X17 < 129400   to the right, agree=0.877, adj=0.625, (0 split)
##
## Node number 1394: 54 observations,    complexity param=0.001205546
##   predicted class=0  expected loss=0.3888889  P(node) =0.0036
##     class counts:    33    21
##    probabilities: 0.611 0.389
##   left son=2788 (20 obs) right son=2789 (34 obs)
##   Primary splits:
##       X12 < 16736    to the right, improve=2.266667, (0 missing)
##       X13 < 17024    to the right, improve=1.500000, (0 missing)
##       X14 < 15834    to the right, improve=1.225490, (0 missing)
##       X16 < 14677.5  to the right, improve=1.225490, (0 missing)
##       X1  < 90000    to the right, improve=1.184637, (0 missing)
##   Surrogate splits:
##       X13 < 15827    to the right, agree=0.963, adj=0.90, (0 split)
##       X14 < 15834    to the right, agree=0.963, adj=0.90, (0 split)
##       X15 < 15409    to the right, agree=0.926, adj=0.80, (0 split)
##       X16 < 15986    to the right, agree=0.870, adj=0.65, (0 split)
##       X17 < 15536.5  to the right, agree=0.833, adj=0.55, (0 split)
```

```
##
## Node number 1395: 24 observations
##   predicted class=1  expected loss=0.3333333  P(node) =0.0016
##     class counts:     8    16
##    probabilities: 0.333 0.667
##
## Node number 1476: 34 observations
##   predicted class=0  expected loss=0.2647059  P(node) =0.002266667
##     class counts:    25     9
##    probabilities: 0.735 0.265
##
## Node number 1477: 64 observations,    complexity param=0.001205546
##   predicted class=1  expected loss=0.453125  P(node) =0.004266667
##     class counts:    29    35
##    probabilities: 0.453 0.547
##   left son=2954 (18 obs) right son=2955 (46 obs)
##   Primary splits:
##       X3  < 1.5      to the left,  improve=3.626963, (0 missing)
##       X22 < 3055     to the left,  improve=3.227522, (0 missing)
##       X16 < 73284    to the left,  improve=1.968750, (0 missing)
##       X17 < 74098    to the left,  improve=1.968750, (0 missing)
##       X15 < 75203    to the left,  improve=1.513236, (0 missing)
##   Surrogate splits:
##       X1  < 175000   to the right, agree=0.750, adj=0.111, (0 split)
##       X13 < 17261.5  to the left,  agree=0.750, adj=0.111, (0 split)
##       X15 < 15037.5  to the left,  agree=0.734, adj=0.056, (0 split)
##       X16 < 14602.5  to the left,  agree=0.734, adj=0.056, (0 split)
##       X17 < 16127    to the left,  agree=0.734, adj=0.056, (0 split)
##
## Node number 1872: 69 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.3768116  P(node) =0.0046
##     class counts:    43    26
##    probabilities: 0.623 0.377
##   left son=3744 (11 obs) right son=3745 (58 obs)
##   Primary splits:
##       X21 < 3210     to the right, improve=3.716142, (0 missing)
##       X18 < 4866     to the right, improve=3.321051, (0 missing)
##       X17 < 19257    to the right, improve=2.960923, (0 missing)
##       X7  < 1        to the right, improve=2.840580, (0 missing)
##       X20 < 1358.5   to the left,  improve=2.684789, (0 missing)
##   Surrogate splits:
##       X20 < 7862     to the right, agree=0.884, adj=0.273, (0 split)
##       X10 < -0.5     to the left,  agree=0.870, adj=0.182, (0 split)
##       X12 < 225363   to the right, agree=0.870, adj=0.182, (0 split)
##       X13 < 28715    to the left,  agree=0.870, adj=0.182, (0 split)
##       X18 < 14006    to the right, agree=0.870, adj=0.182, (0 split)
##
## Node number 1873: 58 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.3965517  P(node) =0.003866667
##     class counts:    23    35
```

```
##    probabilities: 0.397 0.603
##   left son=3746 (21 obs) right son=3747 (37 obs)
##   Primary splits:
##       X22 < 1600     to the right, improve=3.259264, (0 missing)
##       X23 < 2683.5   to the right, improve=2.207896, (0 missing)
##       X1  < 25000    to the right, improve=1.735945, (0 missing)
##       X5  < 36.5     to the right, improve=1.674900, (0 missing)
##       X14 < 26288.5  to the left,  improve=1.599200, (0 missing)
##   Surrogate splits:
##       X23 < 2683.5   to the right, agree=0.776, adj=0.381, (0 split)
##       X12 < 1828     to the left,  agree=0.690, adj=0.143, (0 split)
##       X15 < 1409     to the left,  agree=0.690, adj=0.143, (0 split)
##       X8  < 2.5      to the right, agree=0.672, adj=0.095, (0 split)
##       X14 < 653      to the left,  agree=0.672, adj=0.095, (0 split)
##
## Node number 1878: 72 observations,    complexity param=0.001024714
##   predicted class=1  expected loss=0.4027778  P(node) =0.0048
##     class counts:    29    43
##    probabilities: 0.403 0.597
##   left son=3756 (18 obs) right son=3757 (54 obs)
##   Primary splits:
##       X1  < 175000   to the left,  improve=4.898148, (0 missing)
##       X21 < 6604     to the left,  improve=4.201389, (0 missing)
##       X13 < 175817.5 to the left,  improve=3.112963, (0 missing)
##       X5  < 39.5     to the left,  improve=2.938889, (0 missing)
##       X12 < 154660.5 to the left,  improve=2.892857, (0 missing)
##   Surrogate splits:
##       X12 < 173855.5 to the left,  agree=0.944, adj=0.778, (0 split)
##       X13 < 160555   to the left,  agree=0.903, adj=0.611, (0 split)
##       X14 < 148952.5 to the left,  agree=0.861, adj=0.444, (0 split)
##       X16 < 92120.5  to the left,  agree=0.806, adj=0.222, (0 split)
##       X15 < 92192    to the left,  agree=0.792, adj=0.167, (0 split)
##
## Node number 1879: 147 observations
##   predicted class=1  expected loss=0.1972789  P(node) =0.0098
##     class counts:    29   118
##    probabilities: 0.197 0.803
##
## Node number 2788: 20 observations
##   predicted class=0  expected loss=0.2  P(node) =0.001333333
##     class counts:    16     4
##    probabilities: 0.800 0.200
##
## Node number 2789: 34 observations,    complexity param=0.001205546
##   predicted class=0  expected loss=0.5  P(node) =0.002266667
##     class counts:    17    17
##    probabilities: 0.500 0.500
##   left son=5578 (16 obs) right son=5579 (18 obs)
##   Primary splits:
##       X15 < 5848.5   to the left,  improve=3.777778, (0 missing)
```

```
##       X16 < 4978     to the left,  improve=3.777778, (0 missing)
##       X17 < 2398     to the left,  improve=3.777778, (0 missing)
##       X2  < 1.5      to the left,  improve=2.922807, (0 missing)
##       X14 < 5751     to the left,  improve=2.922807, (0 missing)
##   Surrogate splits:
##       X16 < 4978     to the left,  agree=1.000, adj=1.000, (0 split)
##       X17 < 2398     to the left,  agree=1.000, adj=1.000, (0 split)
##       X14 < 5751     to the left,  agree=0.971, adj=0.938, (0 split)
##       X13 < 4449.5   to the left,  agree=0.941, adj=0.875, (0 split)
##       X9  < -0.5     to the left,  agree=0.912, adj=0.812, (0 split)
##
## Node number 2954: 18 observations
##   predicted class=0  expected loss=0.2777778  P(node) =0.0012
##     class counts:    13     5
##    probabilities: 0.722 0.278
##
## Node number 2955: 46 observations
##   predicted class=1  expected loss=0.3478261  P(node) =0.003066667
##     class counts:    16    30
##    probabilities: 0.348 0.652
##
## Node number 3744: 11 observations
##   predicted class=0  expected loss=0  P(node) =0.0007333333
##     class counts:    11     0
##    probabilities: 1.000 0.000
##
## Node number 3745: 58 observations,    complexity param=0.001024714
##   predicted class=0  expected loss=0.4482759  P(node) =0.003866667
##     class counts:    32    26
##    probabilities: 0.552 0.448
##   left son=7490 (30 obs) right son=7491 (28 obs)
##   Primary splits:
##       X20 < 1358.5   to the left,  improve=4.099179, (0 missing)
##       X18 < 4866     to the right, improve=3.199459, (0 missing)
##       X7  < 1        to the right, improve=3.194379, (0 missing)
##       X17 < 19257    to the right, improve=3.145796, (0 missing)
##       X8  < 1        to the right, improve=2.667433, (0 missing)
##   Surrogate splits:
##       X15 < 30563.5  to the left,  agree=0.707, adj=0.393, (0 split)
##       X22 < 1250     to the left,  agree=0.690, adj=0.357, (0 split)
##       X19 < 1670.5   to the left,  agree=0.655, adj=0.286, (0 split)
##       X7  < 1        to the right, agree=0.638, adj=0.250, (0 split)
##       X16 < 29498.5  to the left,  agree=0.638, adj=0.250, (0 split)
##
## Node number 3746: 21 observations
##   predicted class=0  expected loss=0.3809524  P(node) =0.0014
##     class counts:    13     8
##    probabilities: 0.619 0.381
##
## Node number 3747: 37 observations
```

```
##   predicted class=1  expected loss=0.2702703  P(node) =0.002466667
##     class counts:    10    27
##    probabilities: 0.270 0.730
##
## Node number 3756: 18 observations
##   predicted class=0  expected loss=0.2777778  P(node) =0.0012
##     class counts:    13     5
##    probabilities: 0.722 0.278
##
## Node number 3757: 54 observations
##   predicted class=1  expected loss=0.2962963  P(node) =0.0036
##     class counts:    16    38
##    probabilities: 0.296 0.704
##
## Node number 5578: 16 observations
##   predicted class=0  expected loss=0.25  P(node) =0.001066667
##     class counts:    12     4
##    probabilities: 0.750 0.250
##
## Node number 5579: 18 observations
##   predicted class=1  expected loss=0.2777778  P(node) =0.0012
##     class counts:     5    13
##    probabilities: 0.278 0.722
##
## Node number 7490: 30 observations
##   predicted class=0  expected loss=0.2666667  P(node) =0.002
##     class counts:    22     8
##    probabilities: 0.733 0.267
##
## Node number 7491: 28 observations
##   predicted class=1  expected loss=0.3571429  P(node) =0.001866667
##     class counts:    10    18
##    probabilities: 0.357 0.643
```

```r
# Predicting it on the training set
train_pred <- predict(decision_tree_model, newdata = creditdefaulttrain, type
= "class")

# Calculating the accuracy
accuracy <- sum(train_pred == creditdefaulttrain$Y) / nrow(creditdefaulttrain
)
print(paste("Accuracy on training set:", accuracy))
```

```
## [1] "Accuracy on training set: 0.838"
```

```r
# Getting the variable importance
importance <- decision_tree_model$variable.importance
print(importance)
```

```
##          X6          X7          X9         X14         X12         X19
## 830.9478328 193.8908903  75.9919839  74.9323010  68.4416386  65.5345839
```

```
##         X10         X15         X16         X13          X8         X11 
##  61.8959984  58.9178083  58.8469615  49.1204135  48.0461311  42.4198405 
##         X18         X20         X17          X1         X21          X5 
##  38.0061788  34.8267026  34.3501253  26.1206264  25.1687114  23.4794680 
##         X22         X23          X3          X4 
##  22.7189424  17.3529393   6.5890394   0.6661879
```

```
# Saving the decision tree plot as a PNG file to see easier.
png("decision_tree.png", width = 1200, height = 800)
rpart.plot(decision_tree_model, main = "Decision Tree for Credit Default", ex
tra = 100)
dev.off()
```

```
## quartz_off_screen 
##                 2
```

Findings from the Decision Trees on the Training Data

Simplified Decision Tree: More individuals (90%) who have X6 < 2, which means payment delay for less than 2 months in September, did not default (0). Therefore, this suggests that historically when a customer pays in less than 2 months, they are very likely to pay on time. However, only 10% of individuals who delayed payments of 2 months or more, X6>=2, defaulted (1). This implies that while late payment is a strong indicator of default, not all customers who are late with payment will necessarily default.

Full Decision Tree: According to class, the nodes at the top near the root are generally the most significant as they are the most important in predicting the variable. At the top, the root node also starts with X6. This suggests that the repayment status in September is a very important predictor. This then splits into X7, indicating again that the history of repayments is critical in predicting defaults. It also splits into X12, which represents bill amounts, suggesting that the recent financial activity of an individual is a predictive factor for default risk. Other variables, such as X5 (Age) are used for split points, which also suggests that it is predictive.

Accuracy on training set: 0.838

###Evaluating the Decision Tree Model on the validation data set

```
# Predicting on the validation set
validationPred_DT <- predict(decision_tree_model, newdata = val_set, type = "
class")

# Calculating the accuracy on the validation set
accuracy_DT <- sum(validationPred_DT == val_set$Y) / nrow(val_set)

# Calculating the confusion matrix
conf_matrix_DT <- confusionMatrix(validationPred_DT, val_set$Y)

# Calculating the precision for class 1
precision_DT <- conf_matrix_DT$byClass["Pos Pred Value"]
```

```r
# Calculating the recall for class 1
recall_DT <- conf_matrix_DT$byClass["Sensitivity"]

# Calculating the F1 Score for class 1
f1_score_DT <- 2 * (precision_DT * recall_DT) / (precision_DT + recall_DT)

# Printing the metric results
print(paste("Accuracy (Decision Tree):", accuracy_DT))

## [1] "Accuracy (Decision Tree): 0.837"

print(paste("Precision (Decision Tree):", precision_DT))

## [1] "Precision (Decision Tree): 0.844293272864701"

print(paste("Recall (Decision Tree):", recall_DT))

## [1] "Recall (Decision Tree): 0.96668109043704"

print(paste("F1 Score (Decision Tree):", f1_score_DT))

## [1] "F1 Score (Decision Tree): 0.901351623966109"
```

The analysis of the Decision Tree model for credit default prediction shows a high level of accuracy, with an 83.8% success rate on the training set and a similar 83.7% on the validation set. This consistency between training and validation sets suggests that the model is reliable and generalises well to new data. In addition, the model's performance has a precision of 84.43%. This means that when the model predicts a default, it is correct approximately 84.43% of the time. Furthermore, the model shows a great recall or sensitivity of 96.67%. This suggests that it correctly identifies nearly 96.67% of all actual defaults. The F1 Score, which balances precision and recall, is also very high at 0.9014. This high score suggests that the model is efficient and accurately predicts defaults while maintaining a balanced approach to false positives and negatives.

2.1 Bagging Model

This model uses bagging (bootstrap aggregating) to train multiple decision trees on different subsets of the training data and then it averages their predictions.

```r
# Loading the library for randomForest
library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

# Training the bagging model using randomForest with all features. I will set mtry in randomForest to the number of features used in the model. This will result in bagging
baggingModel <- randomForest(Y ~ ., data = train_set, mtry = ncol(train_set) - 1, ntree = 500, importance = TRUE)

# I will also look at the importance of each variable
importance(baggingModel)
```

```
##              0           1 MeanDecreaseAccuracy MeanDecreaseGini
## X1   22.613398  11.0148106            27.957507        254.18598
## X2    5.371665  -3.4604804             2.774737         45.30083
## X3    4.338427  -1.4217233             3.058769         85.50910
## X4   12.537935  -0.2244654            11.267602         53.05905
## X5   22.539413   2.5631826            21.847770        310.24048
## X6  106.033579  90.8046741           152.918970        678.43080
## X7   59.430898   1.2661332            62.510709        135.22598
## X8   28.474562  -2.5253971            30.555435         41.18518
## X9   32.255693  -0.1287262            34.961874         42.12609
## X10  26.375690   2.1981591            30.509256         46.39195
## X11  28.220649   0.2947598            30.874520         42.56616
## X12  21.009696  21.9883524            33.039066        277.87976
## X13  33.811828  -9.6757372            35.728152        191.88346
## X14  44.685449 -16.8299875            46.112579        179.08159
## X15  35.962592  -7.9319693            39.594157        174.26499
## X16  34.158331  -5.8689201            36.745729        165.31185
## X17  22.690843   4.5984690            29.390938        181.72672
## X18  29.110735  -9.7097456            29.042006        197.59577
## X19  27.358336   5.8898747            32.614498        219.71122
## X20  22.913125   5.6967408            27.896823        201.50035
## X21  26.626967  -0.4987388            29.531401        190.91080
## X22  27.167096   2.0995445            31.572014        187.03460
## X23  29.863784   1.3248387            32.531086        204.05100
```

```r
# Predicting on the training set
trainingPred <- predict(baggingModel, newdata = train_set, type = "class")

# Calculating the accuracy on the training set
trainingAccuracy <- sum(trainingPred == train_set$Y) / nrow(train_set)
print(paste("Accuracy on training set:", trainingAccuracy))
```

```
## [1] "Accuracy on training set: 0.99975"
```

```r
# Confusion matrix on the training set
conf_matrix_train <- confusionMatrix(trainingPred, train_set$Y)
print(conf_matrix_train)
```

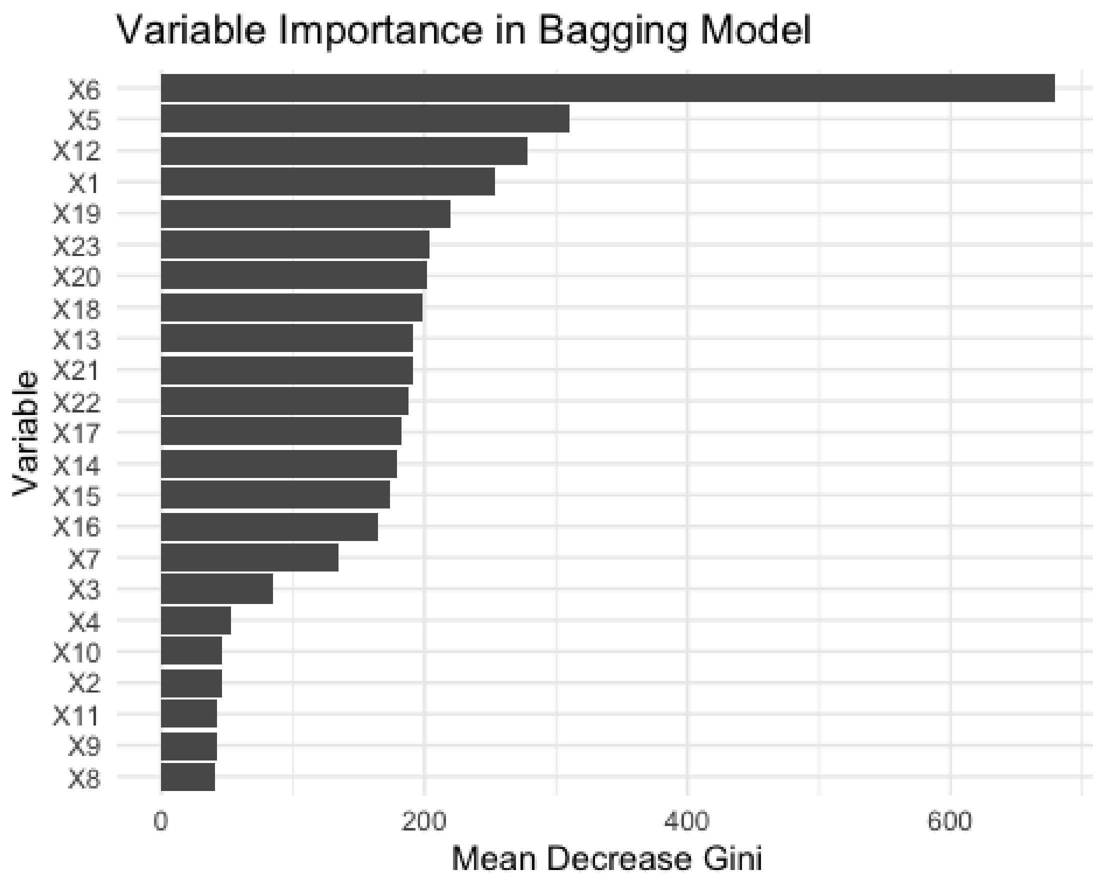
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9371    3
##          1    0 2626
##
##                Accuracy : 0.9998
##                  95% CI : (0.9993, 0.9999)
##     No Information Rate : 0.7809
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9993
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9989
##          Pos Pred Value : 0.9997
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7809
##          Detection Rate : 0.7809
##    Detection Prevalence : 0.7812
##       Balanced Accuracy : 0.9994
##
##        'Positive' Class : 0
##
```

I constructed 500 trees and utilised all the available variables in the data set to ensure that it was a comprehensive analysis. i assessed the variable importance in this model to see what feature significantly contributes to the predictions. I measured this in terms of Mean Decrease in Accuracy and Mean Decrease Gini and identified that variables such as X6, X7, X5, X15, and X14 as particularly influential in the model's decision-making process. This is similar to the decision tree model variables that were influential.

Visualising the variable importance of the bagging model

```r
library(ggplot2)
library(randomForest)

# Getting the variable importance
importance_data <- importance(baggingModel)
feature_names <- row.names(importance_data)
importance_df <- data.frame(Feature = feature_names, MeanDecreaseGini = impor
tance_data[, "MeanDecreaseGini"])
```

```r
# I will order the data frame by MeanDecreaseGini
importance_df <- importance_df[order(importance_df$MeanDecreaseGini, decreasi
ng = TRUE), ]

# Creating the plot
ggplot(importance_df, aes(x = reorder(Feature, MeanDecreaseGini), y = MeanDec
reaseGini)) +
  geom_bar(stat = "identity") +
  coord_flip() +  # I used this to flip the axes to get a horizontal bar plot
for better visuals.
  labs(x = "Variable", y = "Mean Decrease Gini", title = "Variable Importance
in Bagging Model") +
  theme_minimal()
```



Findings from the Bagging Model on the Training Data

The model achieved great accuracy on the training set, with a success rate of 99.975%. This high level of accuracy suggests a strong fit to the training data. The accompanying confusion matrix further highlights the model's predictive strength, showing 9371 true negatives and 2626 true positives, with only 3 instances of false negatives. This translates to a sensitivity rate of 100% and a specificity rate of 99.89%, indicating the model's adeptness in accurately identifying both positive (default) and negative (non-default) cases. The model's agreement between predicted and actual classifications is quantified by a Kappa statistic of 0.9993. This suggests a good level of agreement, which is above what might occur by chance. Additionally, the Mcnemar's Test, with a P-Value of 0.2482, suggests no significant bias in the model's error rate between the two classes.

Despite the models' great performance on the training set, it is very important to further evaluate this on a validation. This will show whether the model is overfitting, especially if the accuracy is near perfect, like in this instance. Overfitting usually occurs when the model is excessively tuned to the specific patterns and noise of the training data. This potentially compromises the effectiveness of new, unseen data.

Evaluating the Bagging Model on the Validation Data Set

```
# Predicting on the validation set using the bagging model
validationPred <- predict(baggingModel, newdata = val_set, type = "class")

# Calculating the accuracy on the validation set
accuracy_Bagging <- sum(validationPred == val_set$Y) / nrow(val_set)

# Calculating the confusion matrix
conf_matrix_Bagging <- confusionMatrix(validationPred, val_set$Y)

# Calculating the precision for class 1
precision_Bagging <- conf_matrix_Bagging$byClass["Pos Pred Value"]

# Calculating the recall for class 1
recall_Bagging <- conf_matrix_Bagging$byClass["Sensitivity"]

# Calculate F1 Score for class 1
f1_score_Bagging <- 2 * (precision_Bagging * recall_Bagging) / (precision_Bag
ging + recall_Bagging)

# Printing the metrics
print(paste("Accuracy (Bagging Model):", accuracy_Bagging))

## [1] "Accuracy (Bagging Model): 0.813666666666667"

print(paste("Precision (Bagging Model):", precision_Bagging))

## [1] "Precision (Bagging Model): 0.837962962962963"

print(paste("Recall (Bagging Model):", recall_Bagging))

## [1] "Recall (Bagging Model): 0.939852877542189"

print(paste("F1 Score (Bagging Model):", f1_score_Bagging))

## [1] "F1 Score (Bagging Model): 0.885988170507852"
```

When the Bagging Model was applied to the validation set, the model's accuracy dropped to 81.33%. This suggests that while the model is still performing well, it did not capture the validation data as effectively as the training data. This is a common indicator of overfitting. The Recall for defaults is also very high at 93.94%, which suggests that the model is very effective at identifying most of the actual default cases in the validation set. It means that the model can catch a large majority of potential defaults, which is crucial for financial institutions since undetected defaults can lead to significant financial losses. Furthermore, the F1 score is 88.58% which is very good. Additionally, the model's precision is good, which means that defaulting customers are more likely to be classed correctly, which mitigates any potential risks and losses of the business.

However, it is important to note that the drop in accuracy from the training to the validation set indicates that the model may be overfitted to the training data and might not generalize as well to new, unseen data. Even though the model seems powerful in detecting defaults, it is important to monitor how the model performs in a real-world scenario and continue to adjust and validate the model with new data.

2.2 Random Forest

```
# No need to load random forest as it was previoudly loaded for bagging.

# Setting the seed for reproducibility
set.seed(123)

# Training the Random Forest model
randomForestModel <- randomForest(Y ~ ., data = train_set, mtry = sqrt(ncol(t
rain_set) - 1), ntree = 500)

# Summarising the model
print(randomForestModel)

##
## Call:
##  randomForest(formula = Y ~ ., data = train_set, mtry = sqrt(ncol(train_se
t) -      1), ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 18.06%
## Confusion matrix:
##      0   1 class.error
## 0 8879 492   0.0525024
## 1 1675 954   0.6371244

# Predicting on the training set
trainingPredictions <- predict(randomForestModel, newdata = train_set)

# Calculating the accuracy on the training set
trainingAccuracy <- sum(trainingPredictions == train_set$Y) / nrow(train_set)
cat("Accuracy on training set:", trainingAccuracy, "\n")
```

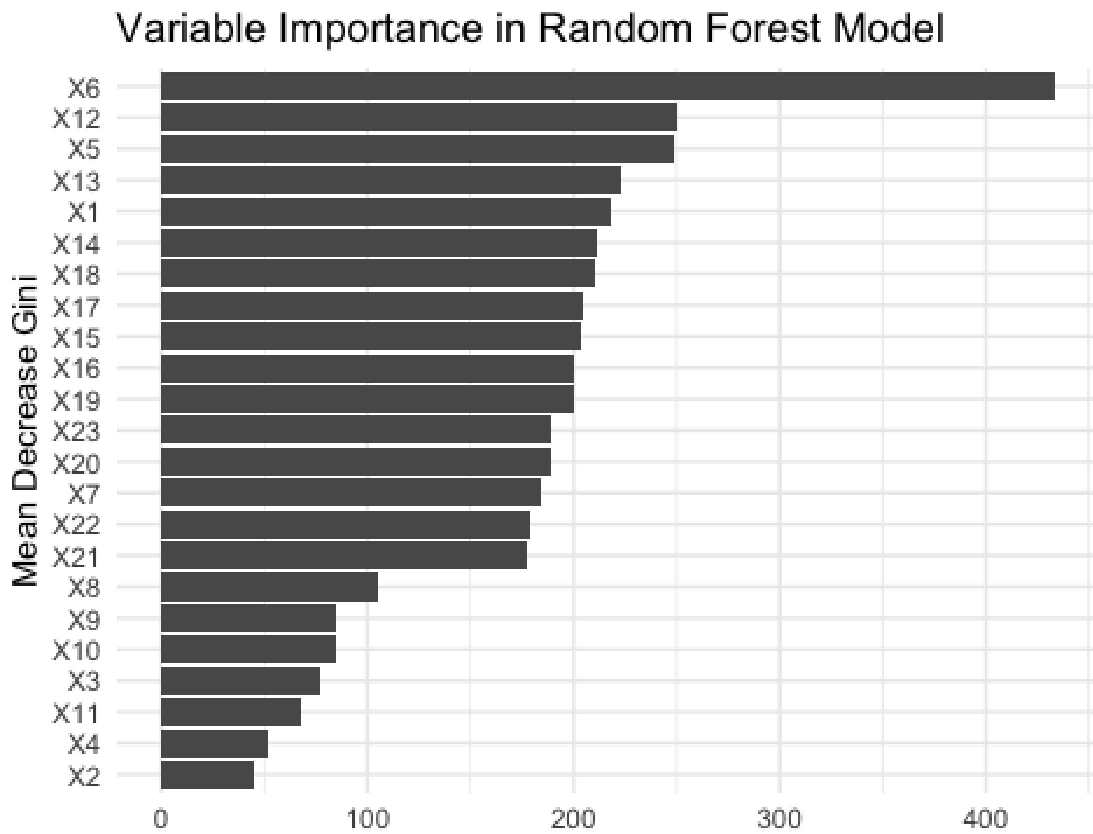
```
## Accuracy on training set: 0.996
```

```r
# Confusion matrix on the training set
confusionMatrixTrain <- table(Predicted = trainingPredictions, Actual = train
_set$Y)
print(confusionMatrixTrain)
```

```
##          Actual
## Predicted    0    1
##         0 9369   46
##         1    2 2583
```

Visualising the Random Forest Model on the variable importance

```r
# Getting the variable importance
importance_data_rf <- importance(randomForestModel)
feature_names_rf <- row.names(importance_data_rf)
importance_df_rf <- data.frame(Feature = feature_names, MeanDecreaseGini = im
portance_data_rf[, "MeanDecreaseGini"])

# Ordering the data frame for random forest (rf) by MeanDecreaseGini
importance_df_rf <- importance_df_rf[order(importance_df_rf$MeanDecreaseGini,
decreasing = TRUE), ]

# Create the plot using ggplot2
ggplot(importance_df_rf, aes(x = reorder(Feature, MeanDecreaseGini), y = Mean
DecreaseGini)) +
  geom_bar(stat = "identity") +
  coord_flip() +  # Make the bar plot horizontal
  labs(title = "Variable Importance in Random Forest Model", x = "Mean Decrea
se Gini", y = "") +
  theme_minimal()  # Use a minimal theme for a cleaner look
```



Evaluating the Random Forest Model on the validation data set

```
# Predict on the validation set using the bagging model
validationPredictions <- predict(randomForestModel, newdata = val_set)

# Calculate accuracy on the validation set
accuracy_RF <- sum(validationPredictions == val_set$Y) / nrow(val_set)

# Calculate confusion matrix
conf_matrix_RF <- confusionMatrix(validationPredictions, val_set$Y)

# Calculate precision for class 1
precision_RF <- conf_matrix_RF$byClass["Pos Pred Value"]

# Calculate recall for class 1
recall_RF <- conf_matrix_RF$byClass["Sensitivity"]

# Calculate F1 Score for class 1
f1_score_RF <- 2 * (precision_RF * recall_RF) / (precision_RF + recall_RF)

# Print metrics
cat("Accuracy (Random Forest):", accuracy_RF, "\n")
```

```
## Accuracy (Random Forest): 0.815
cat("Precision (Random Forest):", precision_RF, "\n")
## Precision (Random Forest): 0.8379523
cat("Recall (Random Forest):", recall_RF, "\n")
## Recall (Random Forest): 0.9420164
cat("F1 Score (Random Forest):", f1_score_RF, "\n")
## F1 Score (Random Forest): 0.8869424
```

Findings from the Random Forest Model on the Training Data and Validation Data

The Random Forest Model was trained with 500 trees and I ensured that it considered all the variables, excluding Y at each split. The model achieved an extremely high accuracy of 99.6% on the training set. This indicates that it was able to correctly predict the default status for nearly all of the training instances. The Out-Of-Bag (OOB) error estimate is 18.06%, which is a measure of prediction error for the trees in the forest when they are not using the bootstrapped sample. However, it is important to note that there is a significant difference between OOB error and training accuracy, which could suggest the model is overfitting. The confusion matrix on the training set also shows that the model predicted the majority of non-defaults (class 0) and defaults (class 1) correctly with only 48 instances of class 0 being incorrectly classified as class 1. There were 2 instances of class 1 being incorrectly classified as class 0.

The recall for class 1 on the validation set is 94.20%. This means the model correctly identifies 94.2% of all actual defaults. This is important for credit default prediction as failing to detect defaults could be costly. The F1 score also seems good.

However, just like the Bagging model, the model is very accurate on the training set but shows signs of overfitting as it is reduced on the validation set. Again the model has a high recall, but the drop in performance from the training to the validation set suggests that there needs to be a careful evaluation of the model and a consideration of techniques, such as further tuning to reduce overfitting.

2.3 Gradient Boosting

I installed the new gbm 3 package instead of using the library gbm as it fixes bugs and is an improved version.

```
#install.packages("devtools")
#library("devtools")
#install_github("gbm-developers/gbm3")
#install_github("gbm-developers/gbm3", force = TRUE)
```

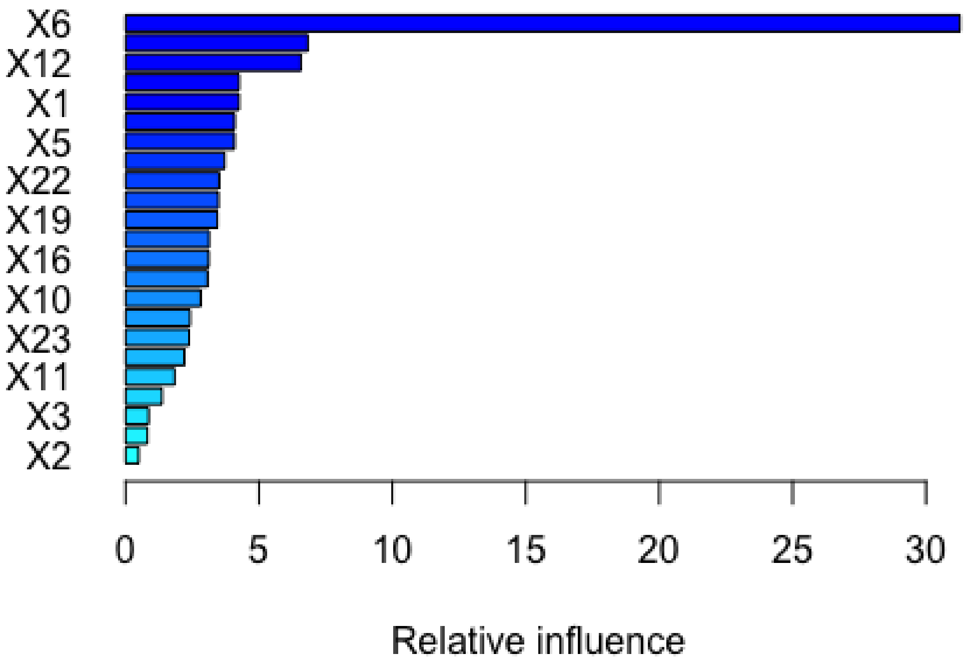
Next I will run the model.

```
# Loading the libraries for this model
library(gbm3)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

# Set seed for reproducibility
set.seed(123)

# Training the Gradient Boosting model
gbm_model <- gbm(Y ~ ., data = train_set, distribution = "bernoulli",
                 n.trees = 500, interaction.depth = 3, shrinkage = 0.1, cv.fo
lds = 5)

# Summarize the model
print(gbm_model)

## gbm(formula = Y ~ ., distribution = "bernoulli", data = train_set,
##     n.trees = 500, interaction.depth = 3, shrinkage = 0.1, cv.folds = 5)
## A gradient boosted model with Bernoulli loss function.
##  500 iterations were performed.
## The best cross-validation iteration was 136.
## There were 23 predictors of which 23 had non-zero influence.
##
## Cross-validation confusion matrix:
##      0   1 Pred. Acc.
## 0 8903 468      95.01
## 1 1671 958      36.44
##
## Cross-validation prediction Accuracy = 82.17%

# Specifying the number of trees for prediction
n_trees <- 500

# Preparing the training set
train_x <- train_set[, -1]
train_y <- train_set$Y
```

```r
# Predicting this on the training data
train_predictions <- predict(gbm_model, newdata = train_x, n.trees = n_trees,
type = "response")
train_predictions_binary <- ifelse(train_predictions > 0.5, 1, 0)
train_predictions_binary <- factor(train_predictions_binary, levels = levels(
train_y))

# Calculating the Accuracy for  training sets
train_accuracy <- sum(train_predictions_binary == train_y) / length(train_y)
print(paste("Accuracy on training set:", train_accuracy))

## [1] "Accuracy on training set: 0.841833333333333"

# Calculating Precision and Recall for training sets
conf_matrix_train <- confusionMatrix(train_predictions_binary, train_y)
precision_train <- conf_matrix_train$byClass["Pos Pred Value"]
recall_train <- conf_matrix_train$byClass["Sensitivity"]
print(paste("Precision on training set:", precision_train))

## [1] "Precision on training set: 0.854674893213099"

print(paste("Recall on training set:", recall_train))

## [1] "Recall on training set: 0.96083662362608"
```

###Visualising gradient boost

```r
# Fitting the Gradient Boosting model
gbm_model <- gbm(Y ~ ., data = train_set, distribution = "bernoulli",
                 n.trees = 500, interaction.depth = 3, shrinkage = 0.1, cv.fo
lds = 5)

# Plotting the relative influence of variables on the graph
summary(gbm_model, plot = TRUE)
```




```
##     var    rel_inf
## X6   X6 31.2721258
## X7   X7  6.8301208
## X12 X12  6.5712034
## X13 X13  4.2542461
## X1   X1  4.2533435
## X18 X18  4.0861242
## X5   X5  4.0795452
## X15 X15  3.7136848
## X22 X22  3.5119233
## X20 X20  3.4783000
## X19 X19  3.4406752
## X14 X14  3.1304039
## X16 X16  3.1152536
## X17 X17  3.0702676
## X10 X10  2.8133397
## X21 X21  2.4160635
## X23 X23  2.3804440
## X8   X8  2.2076501
## X11 X11  1.8401548
## X9   X9  1.3639940
## X3   X3  0.8612936
```

```
## X4   X4  0.8190136
## X2   X2  0.4908293
```

Evaluating the Gradient Boosting Model on the validation data set

```
# Prepare the validation set
val_x <- val_set[, -1]  # Exclude the target variable
val_y <- val_set$Y      # Target variable

# Predict on the validation data with the specified number of trees
val_predictions <- predict(gbm_model, newdata = val_x, n.trees = n_trees, typ
e = "response")

# Converting the probabilities to binary class labels based on a threshold (e
.g., 0.5)
val_predictions_binary <- ifelse(val_predictions > 0.5, 1, 0)

# Converting the val_predictions_binary to a factor with the same levels as v
al_y
val_predictions_binary <- factor(val_predictions_binary, levels = levels(val_
y))

# Calculate accuracy on the validation set
accuracy_GB <- sum(val_predictions_binary == val_y) / length(val_y)

# Calculate confusion matrix
conf_matrix_GB <- confusionMatrix(val_predictions_binary, val_y)

# Calculate precision for class 1
precision_GB <- conf_matrix_GB$byClass["Pos Pred Value"]

# Calculate recall for class 1
recall_GB <- conf_matrix_GB$byClass["Sensitivity"]

# Calculate F1 Score for class 1
f1_score_GB <- 2 * (precision_GB * recall_GB) / (precision_GB + recall_GB)

# Print metrics
print(paste("Accuracy (Gradient Boosting):", accuracy_GB))

## [1] "Accuracy (Gradient Boosting): 0.817333333333333"

print(paste("Precision (Gradient Boosting):", precision_GB))

## [1] "Precision (Gradient Boosting): 0.839168911119661"

print(paste("Recall (Gradient Boosting):", recall_GB))

## [1] "Recall (Gradient Boosting): 0.943747295543055"

print(paste("F1 Score (Gradient Boosting):", f1_score_GB))
```

```
## [1] "F1 Score (Gradient Boosting): 0.888391038696538"
```

Findings from the Gradient Boosting Model on the Training Data and Validation Data

For the training data, the GBM model was trained with 500 decision trees, with each tree having a maximum depth of 3. It also employed a Bernoulli loss function, which is suitable for binary outcomes. The model's training process included cross-validation with 5 folds to see the model's performance and mitigate overfitting. The optimal number of trees identified during this process was 136. This suggested that adding more trees beyond this point did not contribute to improving the model's ability to generalize. The model's accuracy on the training set was approximately 84.18%. The precision of the model, which reflects its ability to predict true positives out of all positive predictions, stood at about 85.47%. This indicates that when the model forecasted a default event, it was accurate around 85% of the time. Moreover, the model showed a high recall of 96.08% on the training set. Again, this is important for identifying the majority of actual defaults. In terms of the cross-validation performed during the training phase, the confusion matrix revealed a predictive accuracy of 82.17%, which indicates that the model can generalise well across different subsets of the data.

For the validation set, the accuracy dropped very slightly to 82.17%. A slight reduction is generally anticipated as models do tend to drop in accuracy with new unseen data. The precision, recall and F1 score were also good, suggesting that the model was able to identify defaults. The high recall and F1 Score shows that the model is effective for predicting credit default. However, since there is a slight discrepancy between the training and cross-validation accuracy, it suggests that the model could be enhanced through further hyperparameter optimisation. Nevertheless, this model seemed to be an improvement from the Bagging and Random Forest Model.

3.Model Selection

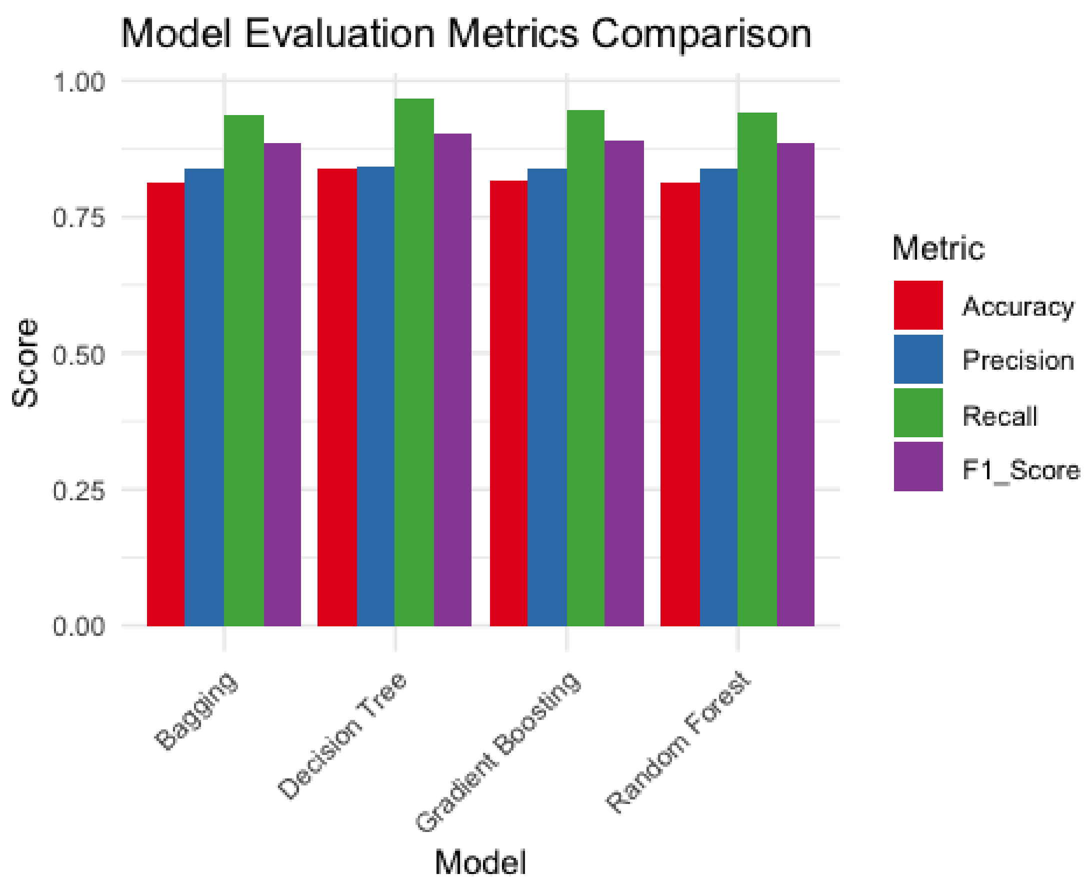
Since I have now evaluated all the models, I will select the one that best performs. To do this, I will create a summary table that includes the evaluation metrics for each model. This table will allow me to compare the models side by side.

```r
# Creating a data frame to hold the evaluation metrics for each model
model_comparison <- data.frame(
  Model = c("Decision Tree", "Bagging", "Random Forest", "Gradient Boosting")
,
  Accuracy = c(0.837, 0.8133, 0.815, 0.8167),
  Precision = c(0.8443, 0.8379, 0.8380, 0.8377),
  Recall = c(0.9667, 0.9394, 0.9420, 0.9450),
  F1_Score = c(0.9014, 0.8858, 0.8869, 0.8882)
)

# Printing the summary table for comparison
print(model_comparison)
```

```
##              Model Accuracy Precision Recall F1_Score
## 1     Decision Tree   0.8370    0.8443 0.9667   0.9014
## 2           Bagging   0.8133    0.8379 0.9394   0.8858
## 3     Random Forest   0.8150    0.8380 0.9420   0.8869
## 4 Gradient Boosting   0.8167    0.8377 0.9450   0.8882
```

3.1 Visualisation of the summary table for all the models

```
# Reshape the data for plotting
long_model_comparison <- reshape2::melt(model_comparison, id.vars = "Model")

# Plot
ggplot(long_model_comparison, aes(x = Model, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(y = "Score", x = "Model", title = "Model Evaluation Metrics Comparison
") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_brewer(palette = "Set1") +
  guides(fill = guide_legend(title = "Metric"))
```



From the above,I can see the Decision Tree model performs the best. However, I would like to further test this through cross validation.

4. Final Model Selection

4.1 Cross Validation

I will perform a 10-fold cross-validation across multiple models with default settings. I will use the train function from the caret package to be able to conduct a consistent and comparable cross-validation across all the models.

```
# Ensuring the target variable is a factor for classification again
creditdefaulttrain$Y <- as.factor(creditdefaulttrain$Y)

# Set seed for reproducibility
set.seed(123)

# Define control parameters for the 10-fold cross-validation
control <- trainControl(method = "cv", number = 10)

# Decision Tree with cross-validation
model_dt_cv <- train(Y ~ ., data = creditdefaulttrain, method = "rpart",
                     trControl = control, tuneLength = 10)
print(model_dt_cv)

## CART
##
## 15000 samples
##    23 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13500, 13500, 13500, 13500, 13500, 13501, ...
## Resampling results across tuning parameters:
##
##   cp            Accuracy   Kappa
##   0.0009041591  0.8133995  0.3525123
##   0.0010247137  0.8155330  0.3603713
##   0.0012055455  0.8167994  0.3619771
##   0.0013060076  0.8176659  0.3623002
##   0.0015069319  0.8191993  0.3664209
##   0.0016073940  0.8196659  0.3673267
##   0.0016576251  0.8196659  0.3673267
##   0.0018083183  0.8199993  0.3694561
##   0.0036166365  0.8205994  0.3579574
##   0.1949969861  0.8009334  0.2015892
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.003616637.

# Random Forest with Bagging and cross-validation
model_bagging_cv <- train(Y ~ ., data = creditdefaulttrain, method = "rf",
```

```r
                              trControl = control, tuneLength = 10)
print(model_bagging_cv)
```

```
## Random Forest
##
## 15000 samples
##    23 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13499, 13500, 13500, 13500, 13500, 13501, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8187996  0.3645208
##    4    0.8176008  0.3710212
##    6    0.8178670  0.3744904
##    9    0.8168007  0.3714588
##   11    0.8172674  0.3734502
##   13    0.8166673  0.3710188
##   16    0.8162005  0.3701349
##   18    0.8168001  0.3720444
##   20    0.8172009  0.3743998
##   23    0.8168001  0.3728739
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# Gradient Boosting Machine with cross-validation
model_gb_cv <- train(Y ~ ., data = creditdefaulttrain, method = "gbm",
                     trControl = control, tuneLength = 10, verbose = FALSE)
print(model_gb_cv)
```

```
## Stochastic Gradient Boosting
##
## 15000 samples
##    23 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13501, 13500, 13500, 13499, 13500, 13500, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##    1                   50     0.8189327  0.3498117
##    1                  100     0.8193993  0.3579244
##    1                  150     0.8195327  0.3613426
##    1                  200     0.8200659  0.3646913
```

```
##    1              250     0.8211327  0.3700941
##    1              300     0.8203992  0.3674009
##    1              350     0.8202659  0.3679642
##    1              400     0.8203992  0.3681216
##    1              450     0.8200658  0.3670248
##    1              500     0.8194658  0.3648240
##    2               50     0.8217326  0.3786211
##    2              100     0.8220658  0.3827370
##    2              150     0.8223992  0.3851719
##    2              200     0.8213992  0.3818315
##    2              250     0.8218658  0.3824590
##    2              300     0.8212659  0.3810693
##    2              350     0.8217327  0.3822825
##    2              400     0.8217325  0.3825047
##    2              450     0.8211990  0.3808673
##    2              500     0.8213992  0.3819260
##    3               50     0.8219994  0.3808204
##    3              100     0.8217994  0.3833939
##    3              150     0.8216660  0.3848226
##    3              200     0.8217991  0.3843001
##    3              250     0.8217326  0.3848079
##    3              300     0.8217329  0.3847830
##    3              350     0.8207992  0.3801941
##    3              400     0.8200658  0.3780276
##    3              450     0.8191325  0.3745605
##    3              500     0.8209323  0.3813075
##    4               50     0.8221327  0.3806660
##    4              100     0.8223326  0.3856299
##    4              150     0.8217325  0.3847663
##    4              200     0.8217991  0.3865693
##    4              250     0.8217990  0.3870145
##    4              300     0.8204661  0.3812408
##    4              350     0.8195327  0.3783429
##    4              400     0.8188660  0.3761938
##    4              450     0.8180659  0.3747956
##    4              500     0.8185992  0.3766889
##    5               50     0.8221994  0.3823376
##    5              100     0.8215995  0.3836057
##    5              150     0.8220660  0.3860786
##    5              200     0.8214660  0.3848743
##    5              250     0.8199323  0.3788078
##    5              300     0.8209323  0.3832794
##    5              350     0.8197991  0.3801003
##    5              400     0.8194659  0.3806290
##    5              450     0.8190657  0.3781484
##    5              500     0.8187991  0.3793391
##    6               50     0.8230663  0.3869537
##    6              100     0.8217331  0.3860916
##    6              150     0.8209999  0.3845002
##    6              200     0.8209997  0.3842165
```

```
##    6                 250      0.8207329  0.3823976
##    6                 300      0.8203327  0.3816372
##    6                 350      0.8204659  0.3829173
##    6                 400      0.8197992  0.3820431
##    6                 450      0.8185994  0.3764695
##    6                 500      0.8178659  0.3745027
##    7                  50      0.8216662  0.3810593
##    7                 100      0.8212664  0.3854852
##    7                 150      0.8207994  0.3842636
##    7                 200      0.8199331  0.3824546
##    7                 250      0.8198662  0.3817787
##    7                 300      0.8183992  0.3773069
##    7                 350      0.8193326  0.3808011
##    7                 400      0.8173324  0.3745594
##    7                 450      0.8161991  0.3709818
##    7                 500      0.8163991  0.3715879
##    8                  50      0.8224660  0.3836779
##    8                 100      0.8205995  0.3815179
##    8                 150      0.8194660  0.3783233
##    8                 200      0.8182663  0.3755387
##    8                 250      0.8175995  0.3738396
##    8                 300      0.8179999  0.3758483
##    8                 350      0.8174662  0.3740339
##    8                 400      0.8158663  0.3683576
##    8                 450      0.8182666  0.3775146
##    8                 500      0.8158663  0.3702362
##    9                  50      0.8242659  0.3898883
##    9                 100      0.8227993  0.3893504
##    9                 150      0.8212659  0.3868059
##    9                 200      0.8215989  0.3876611
##    9                 250      0.8208658  0.3849810
##    9                 300      0.8184661  0.3786504
##    9                 350      0.8181991  0.3779851
##    9                 400      0.8162661  0.3729996
##    9                 450      0.8162666  0.3724534
##    9                 500      0.8152662  0.3681965
##   10                  50      0.8224663  0.3844648
##   10                 100      0.8222659  0.3868230
##   10                 150      0.8211328  0.3836694
##   10                 200      0.8199993  0.3803606
##   10                 250      0.8206659  0.3841893
##   10                 300      0.8172660  0.3726575
##   10                 350      0.8181327  0.3766973
##   10                 400      0.8165994  0.3722058
##   10                 450      0.8146661  0.3667390
##   10                 500      0.8131327  0.3633157
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth =
##  9, shrinkage = 0.1 and n.minobsinnode = 10.
```

From my results, it seems that the Decision Tree may be the best choice. Firstly, it has the highest recall, which means that it minimises missed defaults (false negatives). Secondly, it also leads with the precision and F1 score. Although the Gradient Boosting model slightly leads, in terms of accuracy, it is not as important as the recall score for predicting credit default as it could still have a higher amount of false negatives. This can be critical as missing a default could lead to significant financial losses, making recall more important for predicting default than overall accuracy.

4.1 Testing the Chosen Model on the Unseen Test Set Data

After evaluating the models, I will test how well it genralises to the unseen data. In this case, I will validate it on the separate test set to assess the performance on the unseen data.

```
# First I will ensure that the target variable 'Y' is in the correct format,
factor.
creditdefaulttest$Y <- as.factor(creditdefaulttest$Y)

# Next I will use the trained Decision Tree model to predict on the test data
set
test_pred <- predict(decision_tree_model, newdata = creditdefaulttest, type =
"class")

# Calculate accuracy, precision, recall, and F1 score
conf_matrix_test <- confusionMatrix(test_pred, creditdefaulttest$Y)
accuracy_test <- sum(test_pred == creditdefaulttest$Y) / nrow(creditdefaultte
st)
precision_test <- conf_matrix_test$byClass["Pos Pred Value"]
recall_test <- conf_matrix_test$byClass["Sensitivity"]
f1_score_test <- 2 * (precision_test * recall_test) / (precision_test + recal
l_test)

# Print the metrics
print(paste("Accuracy on test set:", accuracy_test))

## [1] "Accuracy on test set: 0.814133333333333"

print(paste("Precision on test set:", precision_test))

## [1] "Precision on test set: 0.836130007558579"

print(paste("Recall on test set:", recall_test))

## [1] "Recall on test set: 0.946926896079438"

print(paste("F1 Score on test set:", f1_score_test))

## [1] "F1 Score on test set: 0.888086062941554"
```

The Decision Tree model applied to the test dataset for predicting credit card defaults has achieved an accuracy of 81.41%, indicating its ability to correctly predict credit defaults in many cases. Furthermore, the Precision was 83.61%. This high precision rate is important, particularly in financial industries, as it implies a lower rate of false positives—cases where a default is predicted incorrectly. More importantly, the model does really well in recall rate, with a high rate of 94.69%. Recall measures the model's ability to identify actual defaults, and a high recall rate is important in credit default prediction. This is because the consequences of failing to identify a default (a false negative) can have a greater impact than incorrectly predicting a default. Therefore, the high recall shows that the model is highly effective in identifying most of the true default cases. In addition, The F1 Score, which balances precision and recall, is 88.81%. This suggests that there is a strong overall performance of the model. The model is not only good at identifying defaults accurately but does so with a great rate of correctly identifying non-default cases as well. The Decision Tree model shows a reliable performance in predicting credit card defaults, making it a valuable tool in the financial industry where accurately identifying potential defaults is very important.

Conclusion

In conclusion, this classification-focused coursework aimed to address a comprehensive set of tasks, ranging from the initial understanding of the problem to the final evaluation of the selected model. The analysis began by formulating the classification problem. The aim was to predict credit card default and gain meaningful insights from the provided credit default data set. Through rigorous data splitting, exploration, and preprocessing, I gained valuable insights into the dataset's structure, effectively handled missing values and identified relevant variables.

The model building phase included the creation of the Decision Tree Model, followed by Bagging, Random Forest, Gradient Boosting and cross-validation for the optimal model selection. The validation set played a crucial role in comparing the models. This was important as it provided a better understanding of the trade-offs between the different types of models and the ability to generalise. This led to the identification of the best-performing model.

The final evaluation of the chosen model on the test set provided a robust assessment of the Decision Trees' performance on unseen data, reflecting its potential for real-world applications. Throughout my analysis, I aimed to consider whether the model overfitted and what variables significantly contributed to predicting credit card default. The analysis generally showed that the variables X6-X11, history of past payments, were very significant in predicting credit card default. Furthermore, demographic factors, such as age and education also significantly influenced credit card default.

Overall, my findings presented in this report contribute to a better understanding of the dataset and showcase the effectiveness of models in addressing the specified problem. My findings show that the Decision Tree was the best model due to the high recall rate.