Predicting Creditablility of a Customer

The German Credit Data contains data on 20 variables and the classification whether an applicant is considered a Good or a Bad credit risk for 1000 loan applicants. A predictive model developed on this data is expected to provide a bank manager guidance for making a decision whether to approve a loan to a prospective applicant based on customer's profiles.

Problem: Predicting a customer's ability to repay a loan.

Variables:

- **Creditability: (Target variable)**This refers to the borrower's overall creditworthiness and ability to repay the loan.(Categorical)
- **Account Balance:** This indicates the current balance in the applicant's bank account. No account(1), No balance(2), Some balance(3)(Categorical)
- **Duration of Credit :** This specifies the length of the loan term in months.(Numerical)
- **Payment Status of Previous Credit :** This reflects the applicant's history of repaying previous loans on time. Some problems(1), Paid up(2), No problems(in this bank)(3)(Categorical)
- **Purpose :** This indicates the reason for which the applicant is seeking the loan.(Categorical)
- **Credit Amount :** This represents the total amount of money the applicant is applying to borrow.(Numerical)
- **Value of Savings/Stocks :** This indicates the value of the applicant's savings and investments.(Numerical)
- **Length of Current Employment :** This specifies how long the applicant has been employed in their current job.(Numerical)
- **Installment per Cent :** This represents the percentage of the loan amount that the applicant will repay in each installment. (Numerical)
- **Sex and Marital Status :** This captures the applicant's gender and marital status.(Categorical)
- **Guarantors :** This indicates whether the applicant has a guarantor who agrees to repay the loan if the applicant defaults.(Categorical)
- **Duration at Current Address :** This specifies how long the applicant has been living at their current address.(Numerical)

- **Most Valuable Available Asset :** This refers to the applicant's most valuable asset, which could be a car, house, or other property.
- **Age in Years :** This indicates the applicant's age.(Numerical)
- **Concurrent Credits :** This specifies the number of other loans or lines of credit that the applicant currently has.(Numerical)
- **Type of Apartment :** This indicates the type of apartment the applicant lives in (e.g., single-family home, apartment building).(categorical)
- **Number of Credits at This Bank** : This specifies the number of existing loans or credit lines that the applicant has with the bank where they are applying for the new loan.(Numerical)
- **Occupation :** This indicates the applicant's occupation or job title.(Categorical)
- **Number of Dependents :** This specifies the number of people who rely on the applicant for financial support.(Numerical)
- **Foreign Worker :** This indicates whether the applicant is a foreign worker.(Categorical)

Needed Packages:

```
install.packages("caret")
install.packages("rpart")
install.packages("rpart.plot")
install.packages("rsample") # For splitting
install.packages("glmnet")
install.packages("ROSE")
```

```
library(caret)
library(rpart)
library(rpart.plot)
library(rsample) # For splitting
library(glmnet)
library(ROSE)
library(readr)
```

```
data <- read_delim("german.csv", delim = ";",
    escape_double = FALSE, trim_ws = TRUE)
```

Check the structure of the data:

```
str(data)
```

```
spc_tbl_ [1,000 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Creditability                    : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ Account_Balance                  : num [1:1000] 1 1 2 1 1 1 1 1 4 2 ...
$ Duration_of_Credit_monthly       : num [1:1000] 18 9 12 12 12 10 8 6 18 24 ...
$ Payment_Status_of_Previous_Credit: num [1:1000] 4 4 2 4 4 4 4 4 4 2 ...
$ Purpose                          : num [1:1000] 2 0 9 0 0 0 0 0 3 3 ...
$ Credit_Amount                    : num [1:1000] 1049 2799 841 2122 2171 ...
$ Value_Savings_Stocks             : num [1:1000] 1 1 2 1 1 1 1 1 1 3 ...
$ Length_of_current_employment     : num [1:1000] 2 3 4 3 3 2 4 2 1 1 ...
$ Instalment_per_cent              : num [1:1000] 4 2 2 3 4 1 1 2 4 1 ...
$ Sex_Marital_Status               : num [1:1000] 2 3 2 3 3 3 3 3 2 2 ...
$ Guarantors                       : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
$ Duration_in_Current_address      : num [1:1000] 4 2 4 2 4 3 4 4 4 4 ...
$ Most_valuable_available_asset    : num [1:1000] 2 1 1 1 2 1 1 1 3 4 ...
$ Age_years                        : num [1:1000] 21 36 23 39 38 48 39 40 65 23 ...
$ Concurrent_Credits               : num [1:1000] 3 3 3 3 1 3 3 3 3 3 ...
$ Type_of_apartment                : num [1:1000] 1 1 1 1 2 1 2 2 2 1 ...
$ No_of_Credits_at_this_Bank       : num [1:1000] 1 2 1 2 2 2 2 1 2 1 ...
$ Occupation                       : num [1:1000] 3 3 2 2 2 2 2 2 1 1 ...
$ No_of_dependents                 : num [1:1000] 1 2 1 2 1 2 1 2 1 1 ...
$ Telephone                        : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
$ Foreign_Worker                   : num [1:1000] 1 1 1 2 2 2 2 2 1 1 ...
- attr(*, "spec")=
 .. cols(
 ..   Creditability = col_double(),
 ..   Account_Balance = col_double(),
 ..   Duration_of_Credit_monthly = col_double(),
 ..   Payment_Status_of_Previous_Credit = col_double(),
 ..   Purpose = col_double(),
 ..   Credit_Amount = col_double(),
 ..   Value_Savings_Stocks = col_double(),
 ..   Length_of_current_employment = col_double(),
 ..   Instalment_per_cent = col_double(),
 ..   Sex_Marital_Status = col_double(),
 ..   Guarantors = col_double(),
 ..   Duration_in_Current_address = col_double(),
 ..   Most_valuable_available_asset = col_double(),
 ..   Age_years = col_double(),
 ..   Concurrent_Credits = col_double(),
 ..   Type_of_apartment = col_double(),
 ..   No_of_Credits_at_this_Bank = col_double(),
 ..   Occupation = col_double(),
 ..   No_of_dependents = col_double(),
 ..   Telephone = col_double(),
 ..   Foreign_Worker = col_double()
```

```
  .. )
 - attr(*, "problems")=<externalptr>
```

Splitting the data into train and test set: It allows us to train the model on one part and evaluate its performance on another, ensuring that the model generalizes well to new data and avoids overfitting.

```
index <- sample( nrow(data), round(nrow(data)*0.80))
```

```
train <- data[index,]
test <- data[-index,]
```

Building logistic regression model:

```
model <- glm(Creditability  ~ ., data = train, family = "binomial")
model
```

```
Call:  glm(formula = Creditability ~ ., family = "binomial", data = train)

Coefficients:
                        (Intercept)                     Account_Balance
                          -4.032e+00                          6.629e-01
       Duration_of_Credit_monthly  Payment_Status_of_Previous_Credit
                          -2.819e-02                          3.735e-01
                            Purpose                       Credit_Amount
                           2.245e-02                          -5.731e-05
              Value_Savings_Stocks       Length_of_current_employment
                           2.592e-01                          1.057e-01
               Instalment_per_cent                 Sex_Marital_Status
                          -2.996e-01                          2.905e-01
                         Guarantors        Duration_in_Current_address
                           3.527e-01                          -7.719e-03
    Most_valuable_available_asset                           Age_years
                          -1.830e-01                          3.303e-04
                Concurrent_Credits                  Type_of_apartment
                           2.395e-01                          4.412e-01
       No_of_Credits_at_this_Bank                          Occupation
                          -9.482e-02                          -6.956e-02
                  No_of_dependents                           Telephone
                          -3.955e-02                          2.602e-01
```

```
                    Foreign_Worker
                         1.008e+00

Degrees of Freedom: 799 Total (i.e. Null);  779 Residual
Null Deviance:      980.7
Residual Deviance: 751.2    AIC: 793.2
```

Print the summary of the model:

```
summary(model)
```

```
Call:
glm(formula = Creditability ~ ., family = "binomial", data = train)

Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                     -4.032e+00  1.116e+00  -3.612 0.000303 ***
Account_Balance                  6.629e-01  8.048e-02   8.237  < 2e-16 ***
Duration_of_Credit_monthly      -2.819e-02  1.029e-02  -2.739 0.006170 **
Payment_Status_of_Previous_Credit  3.735e-01  9.772e-02   3.822 0.000132 ***
Purpose                          2.245e-02  3.529e-02   0.636 0.524792
Credit_Amount                   -5.731e-05  4.731e-05  -1.211 0.225781
Value_Savings_Stocks             2.592e-01  6.548e-02   3.958 7.56e-05 ***
Length_of_current_employment     1.057e-01  7.918e-02   1.335 0.181985
Instalment_per_cent             -2.996e-01  9.382e-02  -3.194 0.001405 **
Sex_Marital_Status               2.905e-01  1.280e-01   2.269 0.023283 *
Guarantors                       3.527e-01  1.980e-01   1.781 0.074891 .
Duration_in_Current_address     -7.719e-03  8.628e-02  -0.089 0.928716
Most_valuable_available_asset   -1.830e-01  1.024e-01  -1.786 0.074053 .
Age_years                        3.303e-04  9.087e-03   0.036 0.971003
Concurrent_Credits               2.395e-01  1.247e-01   1.921 0.054708 .
Type_of_apartment                4.412e-01  1.904e-01   2.317 0.020504 *
No_of_Credits_at_this_Bank      -9.482e-02  1.831e-01  -0.518 0.604669
Occupation                      -6.956e-02  1.574e-01  -0.442 0.658447
No_of_dependents                -3.955e-02  2.684e-01  -0.147 0.882868
Telephone                        2.602e-01  2.144e-01   1.214 0.224883
Foreign_Worker                   1.008e+00  6.238e-01   1.617 0.105964
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 980.75  on 799  degrees of freedom
Residual deviance: 751.21  on 779  degrees of freedom
AIC: 793.21

Number of Fisher Scoring iterations: 5
```

Calculating the probabilities and dividing the possibilities into two groups:

```
predicted_probs <- predict(model, test, type= "response")
head(predicted_probs)
```

```
        1         2         3         4         5         6
0.7131098 0.5161790 0.2542570 0.3244019 0.7409832 0.7807688
```

```
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)
head(predicted_classes)
```

```
1 2 3 4 5 6
1 1 0 0 1 1
```

This code block is used to measure the performance of a classification model. The first four lines calculate the values of TP, FP, TN, and FN based on the model's predictions. The next line shows the class distribution in the training set. Finally, the confusionMatrix() function evaluates the model's performance by comparing predicted and actual classes, providing metrics such as accuracy, precision, recall scores.

```
confusionMatrix(table(ifelse(test$Creditability == "1", "1", "0"),
predicted_classes), positive = "1")
```

```
Confusion Matrix and Statistics

   predicted_classes
      0   1
  0  25  33
  1  16 126

               Accuracy : 0.755
                 95% CI : (0.6894, 0.8129)
    No Information Rate : 0.795
```

```
   P-Value [Acc > NIR] : 0.92927

                 Kappa : 0.3486

 Mcnemar's Test P-Value : 0.02227

            Sensitivity : 0.7925
            Specificity : 0.6098
         Pos Pred Value : 0.8873
         Neg Pred Value : 0.4310
             Prevalence : 0.7950
         Detection Rate : 0.6300
   Detection Prevalence : 0.7100
      Balanced Accuracy : 0.7011

       'Positive' Class : 1
```

Accuracy: It expresses the ratio of correctly predicted samples. In this case,the accuracy rate is approximately 75%.

95% CI (Confidence Interval): It is the 95% confidence interval of the accuracy rate. This indicates the probability that the accuracy estimate will be within a certain range. (0.684,0.8084)

Kappa (Kappa Statistic): It indicates how much better the model performs compared to a random model. The higher the value, the better the model performs. In this case, the kappa value is 0.375.

Sensitivity (Sensitivity): It expresses the true positive rate. That is, it shows how many of the true positives were correctly detected. In this case, the sensitivity rate is approximately 80%

Specificity (Specificity): It expresses the true negative rate. That is, it shows how many of the true negatives were correctly identified. In this case, the specificity rate is approximately 60%.

Pos Pred Value (Positive Predictive Value): It expresses the accuracy of positive predictions. That is, it shows how many of the predicted positives were actually positive. In this case, the positive predictive value is approximately 85.71%.

Neg Pred Value (Negative Predictive Value): It expresses the accuracy of negative predictions. That is, it shows how many of the predicted negatives were actually negative. In this case, the negative predictive value is approximately 50%.

Prevalence (Prevalence): It expresses the rate of the positive class in the dataset. In this case, the prevalence of the positive class in the dataset is approximately 75%.

Detection Rate: It expresses the rate at which the model correctly detects the positive class. In this case, the detection rate is approximately 60%.

Balanced Accuracy: It is the weighted average of sensitivity and specificity. Balanced accuracy is calculated based on the number of samples in each class and better reflects performance on imbalanced datasets. In this case, the balanced accuracy rate is 70%.

Control overfitting in the model:

```
X_train <- model.matrix(Creditability ~ ., data = train)[,-1]
y_train <- train$Creditability

glmnet_model <- cv.glmnet(X_train, y_train, family = "binomial", alpha = 1)

# Optimum lambda value
best_lambda <- glmnet_model$lambda.min

#final model
final_model <- glmnet(X_train, y_train,
                      family = "binomial",
                      alpha = 1, lambda = best_lambda)

x <- model.matrix(Creditability ~ ., data = test)[,-1]
y <- test$Creditability
predictions <- predict(final_model, newx = x, type = "response")

predicted_classes <- ifelse(predictions > 0.5, 1, 0)

accuracy <- mean(predicted_classes == y)
accuracy
```

```
[1] 0.75
```

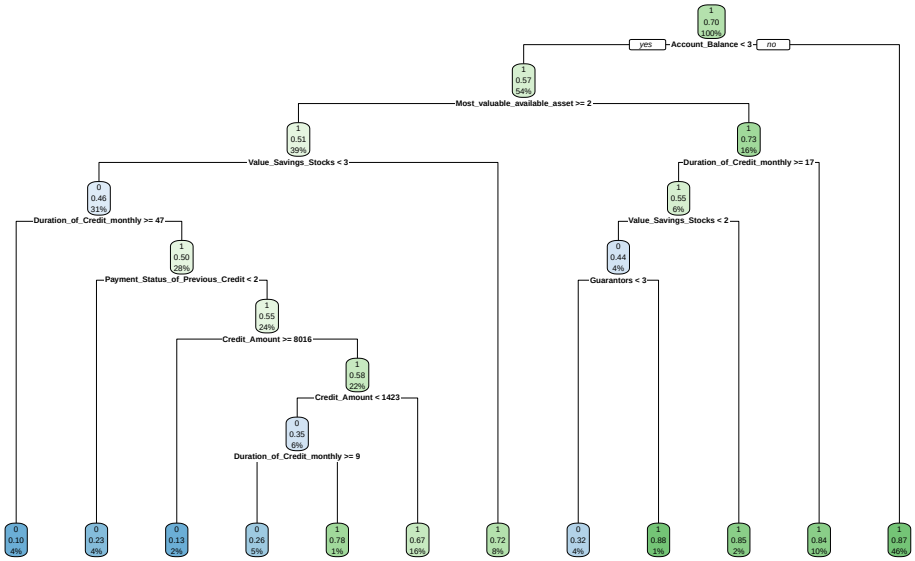
This code cross-validates the Lasso regression to select the optimal lambda value and builds a final model with that value. Predictions are then made for the test data set and the accuracy of the predictions is calculated as %75.

Training decision tree model:

Splitting the data into two parts as train and test.

```
train_index <- createDataPartition(data$Creditability, p = 0.8, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
```

```
model_dt <- rpart(Creditability~ ., data =train_data, method = "class")
rpart.plot(model_dt)
```



Check the model performance:

```
model_preds <- predict(model_dt, test_data ,type = "class")


# Assuming model_preds needs conversion
modelpf <- as.factor(model_preds)

# Assuming test_data$Creditability needs conversion
testdcf <- as.factor(test_data$Creditability)


confusionMatrix(modelpf,
                testdcf,
                positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  22  10
         1  42 126

               Accuracy : 0.74
                 95% CI : (0.6734, 0.7993)
    No Information Rate : 0.68
    P-Value [Acc > NIR] : 0.03883

                  Kappa : 0.3114

 Mcnemar's Test P-Value : 1.716e-05

            Sensitivity : 0.9265
            Specificity : 0.3438
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.6875
             Prevalence : 0.6800
         Detection Rate : 0.6300
   Detection Prevalence : 0.8400
      Balanced Accuracy : 0.6351

       'Positive' Class : 1
```

Accuracy: It expresses the ratio of correctly predicted samples. In this case,the accuracy rate is approximately 71.5%.

95% CI (Confidence Interval): It is the 95% confidence interval of the accuracy rate. This indicates the probability that the accuracy estimate will be within a certain range. (0.6471,0.7764)

Kappa (Kappa Statistic): It indicates how much better the model performs compared to a random model. The higher the value, the better the model performs. In this case, the kappa value is 0.2213.

Sensitivity (Sensitivity): It expresses the true positive rate. That is, it shows how many of the true positives were correctly detected. In this case, the sensitivity rate is approximately 89.29%

Specificity (Specificity): It expresses the true negative rate. That is, it shows how many of the true negatives were correctly identified. In this case, the specificity rate is approximately 30%.

Pos Pred Value (Positive Predictive Value): It expresses the accuracy of positive predictions. That is, it shows how many of the predicted positives were actually positive. In this case, the positive predictive value is approximately 74.85%.

Neg Pred Value (Negative Predictive Value): It expresses the accuracy of negative predictions. That is, it shows how many of the predicted negatives were actually negative. In this case, the negative predictive value is approximately 54.55%.

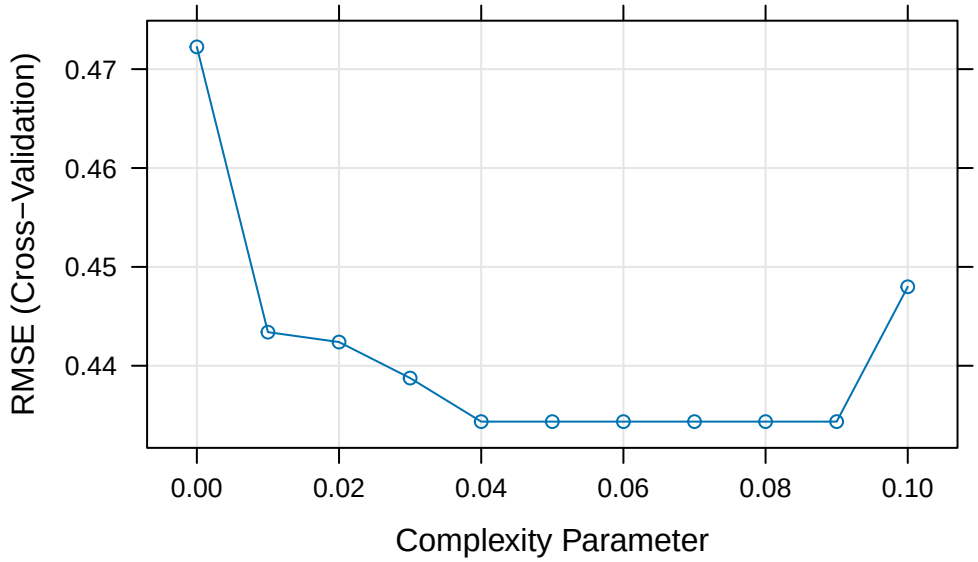
Prevalence (Prevalence): It expresses the rate of the positive class in the dataset. In this case, the prevalence of the positive class in the dataset is approximately 70%.

Detection Rate: It expresses the rate at which the model correctly detects the positive class. In this case, the detection rate is approximately 62.50%.

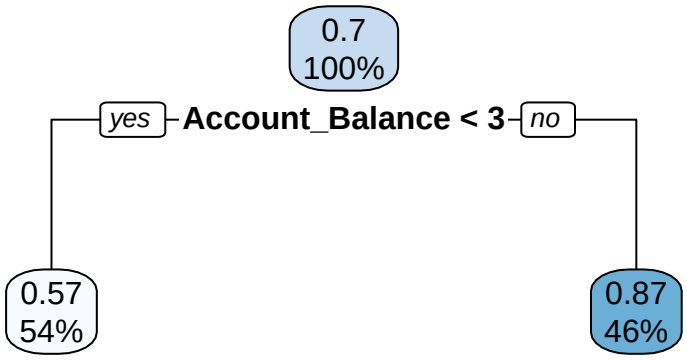
Balanced Accuracy: It is the weighted average of sensitivity and specificity. Balanced accuracy is calculated based on the number of samples in each class and better reflects performance on imbalanced datasets. In this case, the balanced accuracy rate is 59.64%.

10 Fold Cross validation in order to control overfitting:

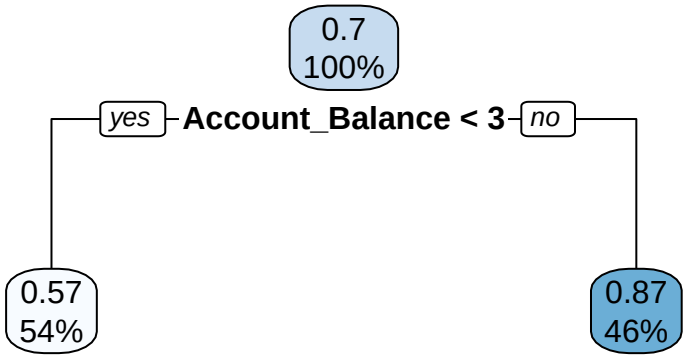
```
overfit_control <- trainControl(method = "cv", number = 10)
model_dt <- train(Creditability ~ .,
                  data = train_data,
                  method = "rpart",
                  trControl = overfit_control,
                  tuneGrid = expand.grid(cp = seq(0, 0.1, 0.01)))
plot(model_dt)
```




```
rpart.plot(model_dt$finalModel)
```




```
rpart.plot(model_dt$finalModel)
```



We try to ensure the consistency of result metrics and evaluate the results accurately by reducing randomness with 10-fold cross-validation.

This code aims to train and visualize a decision tree model for the "Creditability" prediction task using cross-validation to avoid overfitting and grid search to find the best model complexity parameter. Visualization helps understand the decision-making process of the final model.

Check if there is an imbalance problem:

```
table(train_data$Creditability)
```

```
  0   1
236 564
```

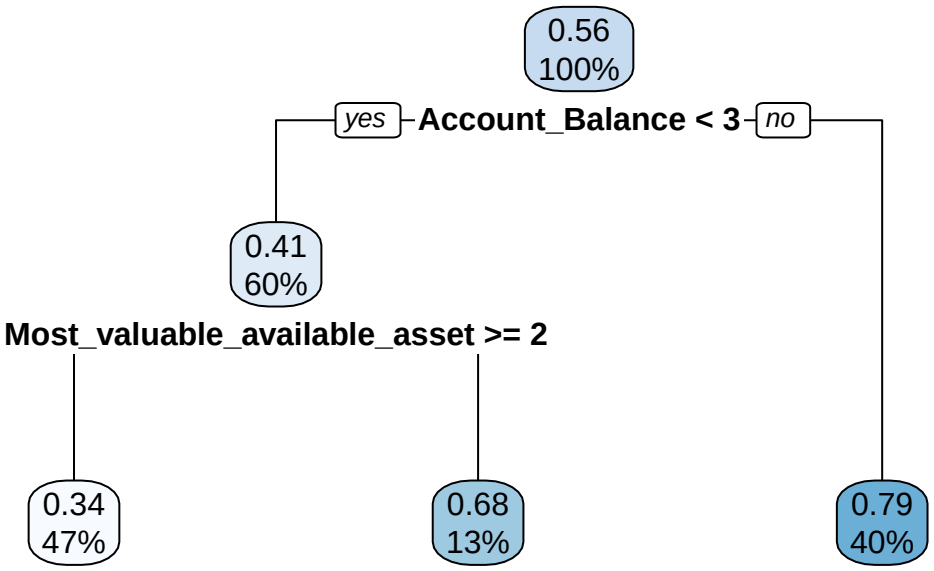
Since there are 236 observations for class 0 and 564 observations for class 1, we can say that there is an imbalance problem. Imbalance usually occurs when one class has many more observations than another. This may cause the model to fail to learn evenly across classes and perform better against the larger class.

To solve the imbalance problem:

```
# sampling
train_oversampled <- ovun.sample(Creditability ~ .,
                                 data = train_data,
                                 method = "over",
                                 N = 1000, seed = 123)

model_dt_balanced <- train(Creditability ~ .,
                           data = train_oversampled$data,
                           method = "rpart",
                           trControl = overfit_control)

predictions_balanced <- predict(model_dt_balanced, newdata = test_data)
rpart.plot(model_dt_balanced$finalModel)
```

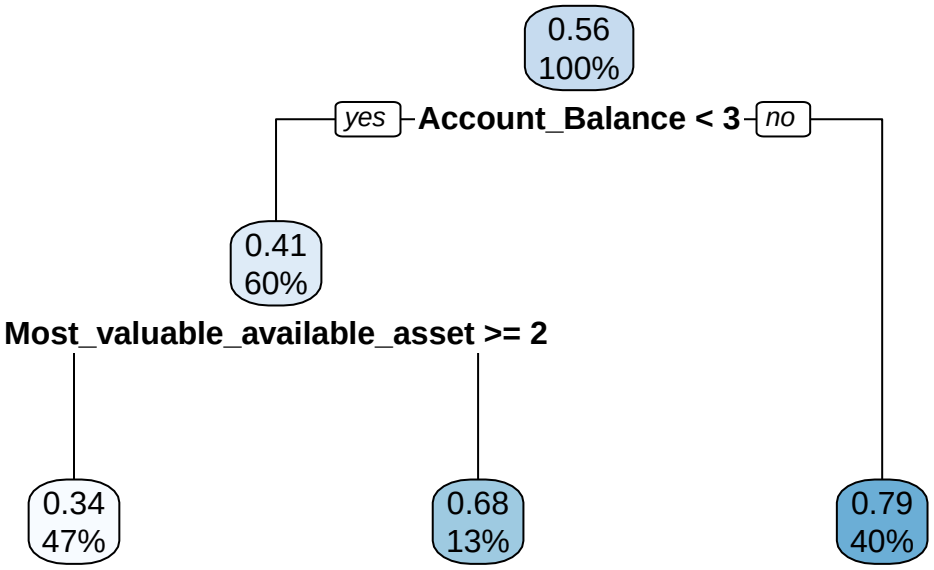


Improving prediction performance of decision tree model tuning hyperparameters:

```
pruned_model_dt <- prune(model_dt_balanced$finalModel,
                         cp = 0.01,
                         minsplit = 100)

predictions_pruned <- predict(pruned_model_dt, newdata = test_data)

rpart.plot(pruned_model_dt)
```



- The decision tree suggests that account balance and credit duration are important factors in determining creditworthiness.
- Individuals with low account balance and short credit duration are more likely to be classified as "No" (not creditable).
- Individuals with high account balance and short credit duration are more likely to be classified as "Yes" (creditable).