Predicting Creditability of a Customer

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

Variables:

- **Class:** (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)

```r
library(readr)
train <- read_csv("train.csv")
```

```
Rows: 900 Columns: 21
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (21): Class, account_balance, duration_of_credit, payment_status_of_prev...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
test <- read_csv("test.csv")
```

```
Rows: 100 Columns: 20
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (20): account_balance, duration_of_credit, payment_status_of_previous_cr...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Used packages:

```
install.packages("ranger")
```

```
Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
(as 'lib' is unspecified)
```

```
library(ranger)

install.packages("caret")
```

```
Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
(as 'lib' is unspecified)
```

```
library(caret)
```

```
Loading required package: ggplot2
```

```
Loading required package: lattice
```

```
head(train) #to see first 6 obs. of train data
```

```
# A tibble: 6 x 21
  Class account_balance duration_of_credit payment_status_of_previous_~1 purpose
  <dbl>           <dbl>              <dbl>                         <dbl>   <dbl>
1     1               1                 18                             4       2
2     1               1                  9                             4       0
3     1               2                 12                             2       9
4     1               1                 12                             4       0
5     1               1                 12                             4       0
6     1               1                 10                             4       0
# i abbreviated name: 1: payment_status_of_previous_credit
# i 16 more variables: credit_amount <dbl>, value_savings <dbl>,
#   length_of_current_employment <dbl>, instalment <dbl>,
#   sex_marital_status <dbl>, guarantors <dbl>,
#   duration_in_current_address <dbl>, asset <dbl>, age <dbl>,
#   concurrent_credits <dbl>, type_of_apartment <dbl>,
#   no_of_credits_in_this_bank <dbl>, occupation <dbl>, ...
```

```r
head(test) #to see first 6 obs. of test data
```

```
# A tibble: 6 x 20
  account_balance duration_of_credit payment_status_of_previous_credit purpose
            <dbl>              <dbl>                             <dbl>   <dbl>
1               4                 12                                 4       3
2               4                 15                                 2       1
3               4                 18                                 2       9
4               4                 36                                 4       3
5               4                 12                                 2       3
6               2                 18                                 4       0
# i 16 more variables: credit_amount <dbl>, value_savings <dbl>,
#   length_of_current_employment <dbl>, instalment <dbl>,
#   sex_marital_status <dbl>, guarantors <dbl>,
#   duration_in_current_address <dbl>, asset <dbl>, age <dbl>,
#   concurrent_credits <dbl>, type_of_apartment <dbl>,
#   no_of_credits_in_this_bank <dbl>, occupation <dbl>, no_of_dependents <dbl>,
#   telephone <dbl>, foreign_worker <dbl>
```

As we can see test data does not includes Class column. So we are going to try making best predictions for test data.

Check structure of the train data:

```r
str(train)
```

```
spc_tbl_ [900 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Class                            : num [1:900] 1 1 1 1 1 1 1 1 1 1 ...
 $ account_balance                  : num [1:900] 1 1 2 1 1 1 1 1 4 2 ...
 $ duration_of_credit               : num [1:900] 18 9 12 12 12 10 8 6 18 24 ...
 $ payment_status_of_previous_credit: num [1:900] 4 4 2 4 4 4 4 4 4 2 ...
 $ purpose                          : num [1:900] 2 0 9 0 0 0 0 0 3 3 ...
 $ credit_amount                    : num [1:900] 1049 2799 841 2122 2171 ...
 $ value_savings                    : num [1:900] 1 1 2 1 1 1 1 1 1 3 ...
 $ length_of_current_employment     : num [1:900] 2 3 4 3 3 2 4 2 1 1 ...
 $ instalment                       : num [1:900] 4 2 2 3 4 1 1 2 4 1 ...
 $ sex_marital_status               : num [1:900] 2 3 2 3 3 3 3 3 2 2 ...
 $ guarantors                       : num [1:900] 1 1 1 1 1 1 1 1 1 1 ...
 $ duration_in_current_address      : num [1:900] 4 2 4 2 4 3 4 4 4 4 ...
 $ asset                            : num [1:900] 2 1 1 1 2 1 1 1 3 4 ...
 $ age                              : num [1:900] 21 36 23 39 38 48 39 40 65 23 ...
 $ concurrent_credits                 : num [1:900] 3 3 3 3 1 3 3 3 3 3 ...
 $ type_of_apartment                  : num [1:900] 1 1 1 1 2 1 2 2 2 1 ...
 $ no_of_credits_in_this_bank         : num [1:900] 1 2 1 2 2 2 2 1 2 1 ...
 $ occupation                         : num [1:900] 3 3 2 2 2 2 2 2 1 1 ...
 $ no_of_dependents                   : num [1:900] 1 2 1 2 1 2 1 2 1 1 ...
 $ telephone                          : num [1:900] 1 1 1 1 1 1 1 1 1 1 ...
 $ foreign_worker                     : num [1:900] 1 1 1 2 2 2 2 2 1 1 ...
 - attr(*, "spec")=
  .. cols(
  ..   Class = col_double(),
  ..   account_balance = col_double(),
  ..   duration_of_credit = col_double(),
  ..   payment_status_of_previous_credit = col_double(),
  ..   purpose = col_double(),
  ..   credit_amount = col_double(),
  ..   value_savings = col_double(),
  ..   length_of_current_employment = col_double(),
  ..   instalment = col_double(),
  ..   sex_marital_status = col_double(),
  ..   guarantors = col_double(),
  ..   duration_in_current_address = col_double(),
  ..   asset = col_double(),
  ..   age = col_double(),
  ..   concurrent_credits = col_double(),
  ..   type_of_apartment = col_double(),
  ..   no_of_credits_in_this_bank = col_double(),
  ..   occupation = col_double(),
  ..   no_of_dependents = col_double(),
  ..   telephone = col_double(),
  ..   foreign_worker = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

Check the missing values:

```
anyNA(train)
```

```
[1] FALSE
```

```
anyNA(test)
```

```
[1] FALSE
```

Check imbalance problem:

```
table(train$Class)
```

```

  0   1 
266 634 
```

Because class levels 0 has 266 observations and 1 has 634 observations we can say that there is a imbalance situation in the target variable. To solve this situation bagging tree method used in this work.

Grid search is used to find best hyperparameter value:

```
control <- trainControl(method = "cv", number = 5, search = "grid")
```

```
grid <- expand.grid(
    mtry = c(5, 10, 15),
    splitrule = "gini",
    min.node.size = c(1, 3, 5)
   )
```

```
train$Class <- factor(train$Class, levels = c("0", "1"))


bt_cv <- train(Class ~ ., data = train,
               method = "ranger",
               trControl = control,
               tuneGrid = grid)
```

Conducting the final model by using best values:

```
bt_final <- ranger(Class ~ ., data = train,
                   mtry = bt_cv$bestTune$mtry,
                   min.node.size = bt_cv$bestTune$min.node.size,
                   splitrule = bt_cv$bestTune$splitrule)
```

Predict the probabilities of the test data:

```
predictions <- predict(bt_final, data = test)$predictions
```

Classification of calculated probability values as 0 and 1:

```
prediction_class <- ifelse(as.numeric(predictions) > 0.5, 1, 0)
```

Creating a data frame with predictions:

```
sonuc <- data.frame(ID = seq(1:100), class= prediction_class)
```

Converting data frame to csv file:

```
#write.csv(sonuc8, "elifk8.csv",row.names = FALSE)
```