Credit Risk Prediction

Hüseyin Durmaz

```r
#install.packages("ranger")
#install.packages("randomFores")
#install.packages("rpart")
#install.packages("rpart.plot")
#install.packages("tidymodels")
#install.packages("dplyr")
#install.packages("pROC")
#install.packages("caret")
#install.packages("gbm")
library(ranger)
library(randomForest)
library(rpart)
library(rpart.plot)
library(tidymodels)
library(dplyr)
library(pROC)
library(ROCR)
library(caret)
library(gbm)

customer <- read.csv("C:/Users/husey/OneDrive/Masaüstü/customer_data.csv")
```

```r
str(customer)
```

```
'data.frame':   1125 obs. of  13 variables:
 $ label : int  1 0 0 1 0 0 1 1 0 0 ...
 $ id    : int  54982665 59004779 58990862 58995168 54987320 59005995 59001917 54984789 5898
 $ fea_1 : int  5 4 7 7 7 6 4 5 5 4 ...
 $ fea_2 : num  1246 1277 1298 1336 NA ...
 $ fea_3 : int  3 1 1 1 2 3 3 3 3 2 ...
```

```
$ fea_4 : num  77000 113000 110000 151000 59000 56000 35000 78000 218000 35000 ...
$ fea_5 : int  2 2 2 2 2 2 2 2 2 2 ...
$ fea_6 : int  15 8 11 11 11 6 8 15 15 8 ...
$ fea_7 : int  5 -1 -1 5 5 -1 9 -1 5 5 ...
$ fea_8 : int  109 100 101 110 108 100 85 111 112 101 ...
$ fea_9 : int  5 3 5 3 4 3 5 3 4 3 ...
$ fea_10: int  151300 341759 72001 60084 450081 60091 60069 60030 151300 60029 ...
$ fea_11: num  245 207 1 1 197 ...
```

Problem, Features and Target

Problem: Credit risk prediction aims to assess the repayment capability of financial institution's customers in advance, detect potential high-risk customers, and minimize financial losses.

Features : This dataset consists of 13 variables and 1125 observations. Label = 1 indicates that the bank providing the credit sees the customer as a high-risk individual, while label = 0 indicates that the bank sees the customer as a low-risk individual. We are performing our operations using the "Payment_data.csv" dataset.

The target is label that what we are going to analyze. Its about credit risks.

Train a logistic regression model, a decision tree, and a random forest model.

```r
# We are splitting the dataset into two separate parts, with 80% (train) and 20% (test) pr

set.seed(30)  #  adding seed
trainfinal <- createDataPartition(newcustomer$label,
                                  p = 0.8,
                                  list = FALSE)

#Training data and test
train <- newcustomer[trainfinal, ]

test <- newcustomer[-trainfinal, ]
```

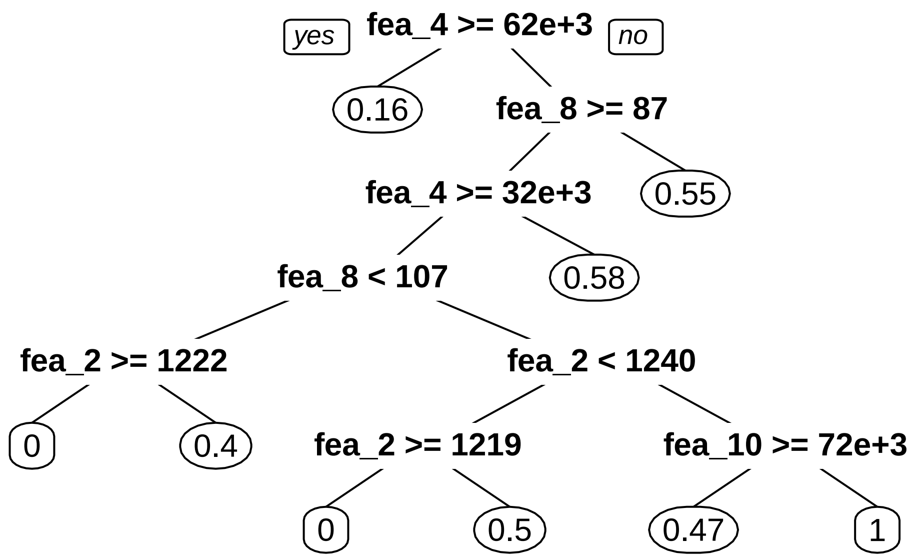
Logistic regression model

```r
modelg <- glm(label ~ .,
                data = train,
                family = "binomial")

summary(modelg)
```

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9114  -0.6940  -0.5991  -0.4562   2.9024

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.584e-02  2.898e+00   0.016  0.98738
fea_1        6.503e-02  7.349e-02   0.885  0.37623
fea_2       -7.642e-04  2.067e-03  -0.370  0.71159
fea_3        1.315e-01  1.219e-01   1.078  0.28085
fea_4       -5.607e-06  1.927e-06  -2.910  0.00361 **
fea_5        1.831e-01  3.908e-01   0.469  0.63936
fea_6        9.715e-03  3.901e-02   0.249  0.80334
fea_7       -4.396e-02  3.572e-02  -1.231  0.21847
fea_8       -5.876e-03  7.907e-03  -0.743  0.45736
fea_9       -6.327e-02  1.086e-01  -0.583  0.56002
fea_10      -5.467e-08  6.947e-07  -0.079  0.93728
fea_11       8.737e-05  8.979e-04   0.097  0.92249
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 752.5  on 780  degrees of freedom
Residual deviance: 730.5  on 769  degrees of freedom
AIC: 754.5

Number of Fisher Scoring iterations: 5
```

In above, We train a linear regression model by using the `train` data that we splited in previous step and the model formula. Then, we assigned the output of `glm()` function to the `modelg` object.

```r
modelfinal <- rpart(label ~ ., data = train)
prp(modelfinal)
```



Random Forest

```r
train$label <- as.factor(train$label)

modelrandomf <- randomForest(label ~ ., data = train,
                             type = "classification")

tahmin <- predict(modelrandomf, train)

modelrandomf
```

```
Call:
 randomForest(formula = label ~ ., data = train, type = "classification")
               Type of random forest: classification
```

```
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 19.72%
Confusion matrix:
    0  1 class.error
0 625 10  0.01574803
1 144  2  0.98630137
```

The model is a classification type random forest consisting of 500 trees. It tries 3 variables at each split during the construction of the trees.

The out-of-bag (OOB) estimate of the error rate for the model is calculated to be 19.72%. The OOB estimate reflects the accuracy of the model's predictions on the data points that were not used during the training process.

The confusion matrix is given as follows:

For samples with class 0 label, out of 635 examples, 625 were correctly predicted and 10 were misclassified. For samples with class 1 label, out of 146 examples, only 2 were correctly predicted while 144 were misclassified.

The class errors are also provided:

The error rate for class 0 is calculated as 0.01574803. The error rate for class 1 is calculated as 0.98630137.

When evaluating the performance of the model, it can be observed that it performs well in predicting class 0 (high accuracy rate and low error rate). However, it exhibits weak performance in predicting class 1 (low accuracy rate and high error rate).

This situation may be attributed to factors such as data imbalance or insufficient samples belonging to class 1. Measures such as obtaining more data or adjusting the sample balance to address these issues can be taken to improve the model's ability to predict class 1. Additionally, exploring different hyperparameter settings or feature selection strategies may also be beneficial.

We are plotting the ROC curve and calculating the AUC value.

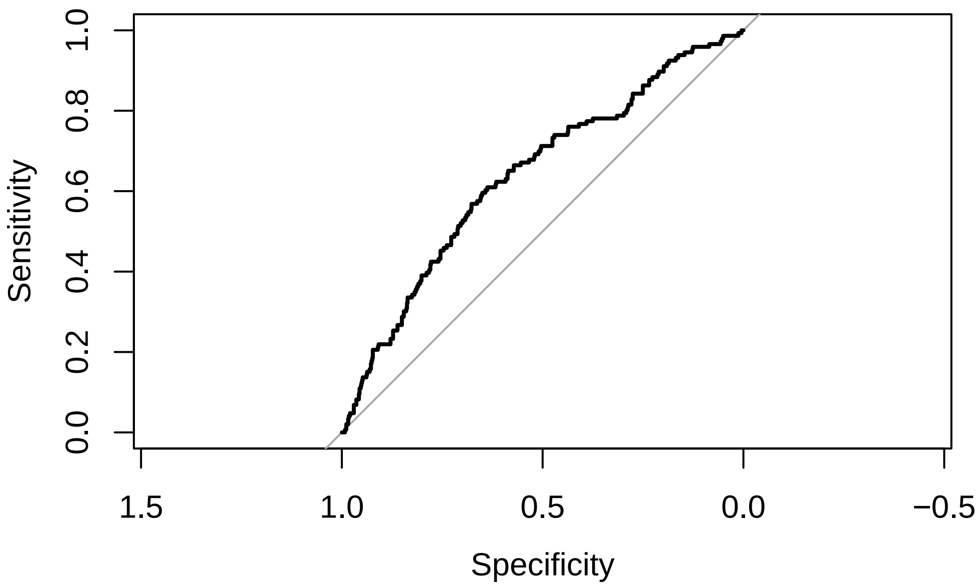
```r
tahmn <- predict(modelg,
                  newdata = train)
```

```r
roc <- roc(response = train$label,
           predictor = tahmn)
```

```
Setting levels: control = 0, case = 1
```

```
Setting direction: controls < cases
```

```r
auc <- auc(roc)
plot(roc)
```




```r
auc
```

```
Area under the curve: 0.6392
```

- The AUC value of 0.6392 indicates that the model demonstrates acceptable performance overall, but there is room for improvement. It suggests that the model's classification ability needs to be optimized further in order to enhance its performance.

Training Bagging Trees for comparing

```r
set.seed(30)
bttrain <- ranger(label ~ .,
                    data = train,
                    mtry = 11)

bttrain
```

```
Ranger result

Call:
 ranger(label ~ ., data = train, mtry = 11)

Type:                             Classification
Number of trees:                  500
Sample size:                      781
Number of independent variables:  11
Mtry:                             11
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             19.59 %
```

Training random forest

```r
set.seed(30)
rftrain <- ranger(label ~ .,data = train)
rftrain
```

```
Ranger result

Call:
 ranger(label ~ ., data = train)

Type:                             Classification
Number of trees:                  500
Sample size:                      781
Number of independent variables:  11
Mtry:                             3
```

```
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             19.85 %
```

Bt Confussion Matrix

```
test$label = as.factor(test$label)
tahminbt <- predict(bttrain,
                    test)

confusionMatrix(tahminbt$predictions,
                test$label)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 151  37
         1   2   5

               Accuracy : 0.8
                 95% CI : (0.7369, 0.8537)
    No Information Rate : 0.7846
    P-Value [Acc > NIR] : 0.3366

                  Kappa : 0.1519

 Mcnemar's Test P-Value : 5.199e-08

            Sensitivity : 0.9869
            Specificity : 0.1190
         Pos Pred Value : 0.8032
         Neg Pred Value : 0.7143
             Prevalence : 0.7846
         Detection Rate : 0.7744
   Detection Prevalence : 0.9641
      Balanced Accuracy : 0.5530

       'Positive' Class : 0
```

Rf Confussion Matrix

```r
tahminrf <- predict(rftrain,
                    test)

confusionMatrix(tahminrf$predictions,
                test$label)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 153  40
         1   0   2

               Accuracy : 0.7949
                 95% CI : (0.7313, 0.8492)
    No Information Rate : 0.7846
    P-Value [Acc > NIR] : 0.403

                  Kappa : 0.0728

 Mcnemar's Test P-Value : 6.984e-10

            Sensitivity : 1.00000
            Specificity : 0.04762
         Pos Pred Value : 0.79275
         Neg Pred Value : 1.00000
             Prevalence : 0.78462
         Detection Rate : 0.78462
   Detection Prevalence : 0.98974
      Balanced Accuracy : 0.52381

       'Positive' Class : 0
```

Accuracy: The accuracy value in the first confusion matrix is 0.8 (80%), while in the second confusion matrix it is stated as 0.7949 (79.49%). Although the accuracy values between the two matrices are almost the same, the accuracy value in the first matrix is slightly higher.

Sensitivity: In the first confusion matrix, the sensitivity value (also known as recall or true positive rate) is 0.9869 (98.69%), whereas in the second confusion matrix, the sensitivity value is stated as 1.0000 (100%). In the first matrix, the ability to correctly classify class 0 is high, while in the second matrix, the accuracy of classifying class 0 correctly is at the maximum level.

The first matrix exhibits a higher performance in terms of overall accuracy, while the second matrix has a higher sensitivity in correctly classifying class 0.

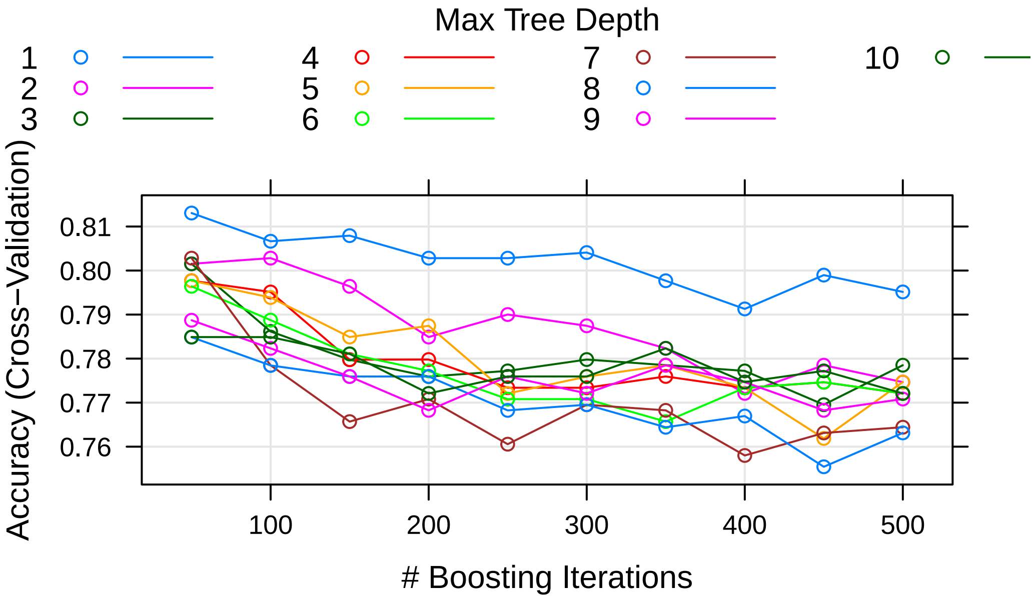
#Let's use one of the tool we learned during the courses.

```
set.seed(30)
gbm_fit <- train(label ~ .,
                 data = train,

                 method = "gbm",
                 trControl = trainControl(method = "cv", number = 5),
                 verbose = FALSE,

                 tuneLength = 10)
```

```
gbm_fit$bestTune
```

```
  n.trees interaction.depth shrinkage n.minobsinnode
1      50                 1       0.1             10
```

```
plot(gbm_fit)
```



GBM Performance

```
gbm_preds <- predict(gbm_fit, test)

confusionMatrix(gbm_preds,
                test$label)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 153  42
         1   0   0

               Accuracy : 0.7846
                 95% CI : (0.7202, 0.8401)
    No Information Rate : 0.7846
    P-Value [Acc > NIR] : 0.5412

                  Kappa : 0
```

```
Mcnemar's Test P-Value : 2.509e-10

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.7846
         Neg Pred Value :    NaN
             Prevalence : 0.7846
         Detection Rate : 0.7846
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

The confusion matrix and accuracy statistic can be used to evaluate the performance of the model. However, it is observed that the model can only predict class 0 and fails to predict class 1. This suggests that the model struggles to recognize class 1 or is biased towards predicting class 0 due to data imbalance. A more detailed analysis and appropriate measures may be necessary to address these issues.