# PREDICT OF CREDIT RISK

### BERKAY TAŞTEMEL HOMEWORK-2

My task is try to train a logistic regression model on the `customer_data` data set to predict the customer credit risk. -label is 1, the customer is in high credit risk -label is 0, the customer is in low credit risk

### PACKAGES AND DATA:

```
#install.packages("caret")
#install.packages("ROCR")
#install.packages("ggplot2")
#install.packages("pROC")
#install.packages("ROSE")
#install.packages("mlbench")
#install.packages("magrittr")
#install.packages("dplyr")
library(mlbench)
library(ROSE)
```

```
Loaded ROSE 0.0-4
```

```
library(caret)
```

```
Zorunlu paket yükleniyor: ggplot2
```

```
Zorunlu paket yükleniyor: lattice
```

```
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```
library(ROCR)
library(ggplot2)
library(readr)
data1 <- read.csv("customer_data.csv", header=TRUE, sep="," , stringsAsFactors=FALSE, stri
payment_data <- na.exclude(data1)
str(data1)
```

```
'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 ...
```

Before training the model, we divide it under the name of train and test.

```
set.seed(123)
index <- sample(1 : nrow(data1), round(nrow(data1) * 0.80))
train <- data1[index, ]
test  <- data1[-index, ]
```

Now, we factor all of the observations we use with the help of as.factor. Next, we train our logistic regression model.

```r
data1$label = as.factor(data1$label)
data1$fea_1 = as.factor(data1$fea_1)
data1$fea_3 = as.factor(data1$fea_3)
data1$fea_4 = as.factor(data1$fea_4)
data1$fea_5 = as.factor(data1$fea_5)
data1$fea_6 = as.factor(data1$fea_6)
data1$fea_7 = as.factor(data1$fea_7)
data1$fea_8 = as.factor(data1$fea_8)
data1$fea_9 = as.factor(data1$fea_9)
data1$fea_10 = as.factor(data1$fea_10)
data1$fea_11 = as.factor(data1$fea_11)
myformula <- formula(label ~ fea_1+fea_3 + fea_4 + fea_5 + fea_6 + fea_7 + fea_8 + fea_9 +
myglm <- glm(myformula , data = train , family= "binomial")
summary(myglm)
```

```
Call:
glm(formula = myformula, family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9606  -0.6961  -0.5864  -0.4014   2.9000

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.157e+00  1.310e+00  -1.647   0.0997 .
fea_1        1.223e-01  6.872e-02   1.779   0.0752 .
fea_3        7.808e-02  1.095e-01   0.713   0.4760
fea_4       -7.668e-06  1.794e-06  -4.274 1.92e-05 ***
fea_5        5.262e-01  3.769e-01   1.396   0.1627
fea_6        1.438e-03  3.735e-02   0.039   0.9693
fea_7       -1.921e-02  3.098e-02  -0.620   0.5352
fea_8       -3.158e-03  7.688e-03  -0.411   0.6812
fea_9       -1.034e-02  1.050e-01  -0.098   0.9215
fea_10      -1.038e-07  6.521e-07  -0.159   0.8735
fea_11       5.645e-04  8.574e-04   0.658   0.5103
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 860.52  on 899  degrees of freedom
```

```
Residual deviance: 826.52  on 889  degrees of freedom
AIC: 848.52

Number of Fisher Scoring iterations: 5
```

## MEASURING MODEL PERFORMANCE :

We need to check the performance of the model on the test set. So we calculate the predicted values of the target variable in the test set.

```r
predicted_label <- predict(myglm , test[,-1], type = "response")
head(predicted_label)
```

```
        1         3         7        12        14        15
0.2320107 0.2048246 0.2429548 0.1586377 0.1797198 0.2584427
```

predicted_label shows the credit risk status here. -label is 1, the customer is in high credit risk. -label is 0, the customer is in low credit risk.

## ROC CURVE AND AUC VALUE :

```r
probabilities <- predict(myglm, newdata = test, type = "response")
roc_curve <- roc(test$label , probabilities)
```

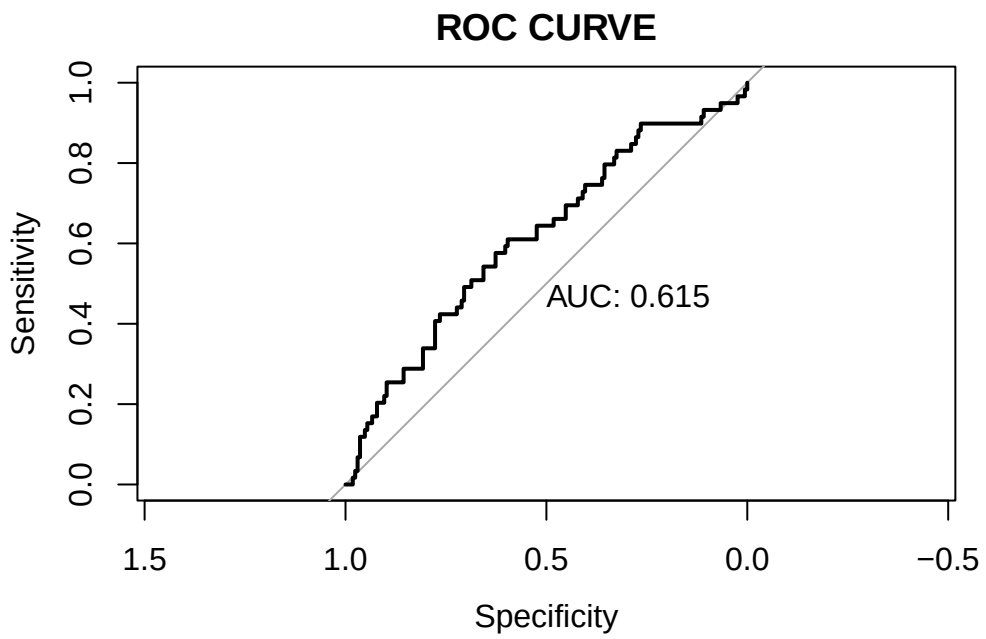
```
Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

```r
auc(roc_curve)
```

```
Area under the curve: 0.615
```

```r
plot(roc_curve, print.auc = TRUE , col = "black" , main= "ROC CURVE")
```




Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC): These metrics are used to evaluate the performance of binary classification models. The ROC curve plots the true positive rate against the false positive rate at various thresholds, while AUC is a measure of the area under the ROC curve. Higher values of AUC indicate better performance.

### IMBALANCE:

Imbalanced regression is a term used to describe a situation where the distribution of the response variable in a regression problem is not evenly distributed across the range of values. In other words, the response variable may have more values in one range compared to another, resulting in an imbalanced dataset.

```
table(train$label)
```

```
  0   1
734 166
```

```
sum(train$label == 0) / nrow(train)  # proportion of negative class
```

```
[1] 0.8155556
```

```
sum(train$label == 1) / nrow(train)  # proportion of positive class
```

```
[1] 0.1844444
```

### OVER-SAMPLING:

Oversampling is a technique used in machine learning to handle imbalanced datasets. When the class distribution in a dataset is imbalanced, i.e., one class has significantly fewer instances than the other(s), a machine learning algorithm may be biased towards the majority class, resulting in poor predictive performance on the minority class. We use "ROSE" (Random Over-Sampling Examples).

```
table(train$label)
```

```
  0   1
734 166
```

```
customer_oversampled <- ROSE(label ~ ., data = train, seed = 1)$data
table(customer_oversampled$label)
```

```
  0   1
401 380
```

### UNDER-SAMPLING:

Under-sampling is a technique used to balance an imbalanced dataset by reducing the number of instances in the majority class to match the number of instances in the minority class. We use downSample function which in the caret package.

```
table(train$label)
```

```
  0   1
734 166
```

```
train$label <- as.factor(train$label)
customer_undersampled <-ovun.sample(label ~ ., data = train, method = "over", N = 1000)$da
table(customer_undersampled$label)
```

```
  0   1
646 354
```

## OVER AND UNDER-SAMPLING COMPARE:

### Over-Sampling:

```
set.seed(123)
train_index <- createDataPartition(customer_oversampled$label, p = 0.8, list = FALSE)
train_balanced <- customer_oversampled[train_index, ]
test_balanced  <- customer_oversampled[-train_index, ]
model_over <- glm(label ~ ., data= train_balanced , family = "binomial")
predict_over <- predict(model_over , test_balanced[,-1], type = "response")
head(predict_over)
```

```
        1         3         9        17        22        27
0.5838746 0.4609678 0.3800289 0.5060739 0.4209211 0.6290017
```

```
probabilities_over <- predict(model_over, newdata = test_balanced, type = "response")
roc_curve <- roc(test_balanced$label , probabilities_over)
```

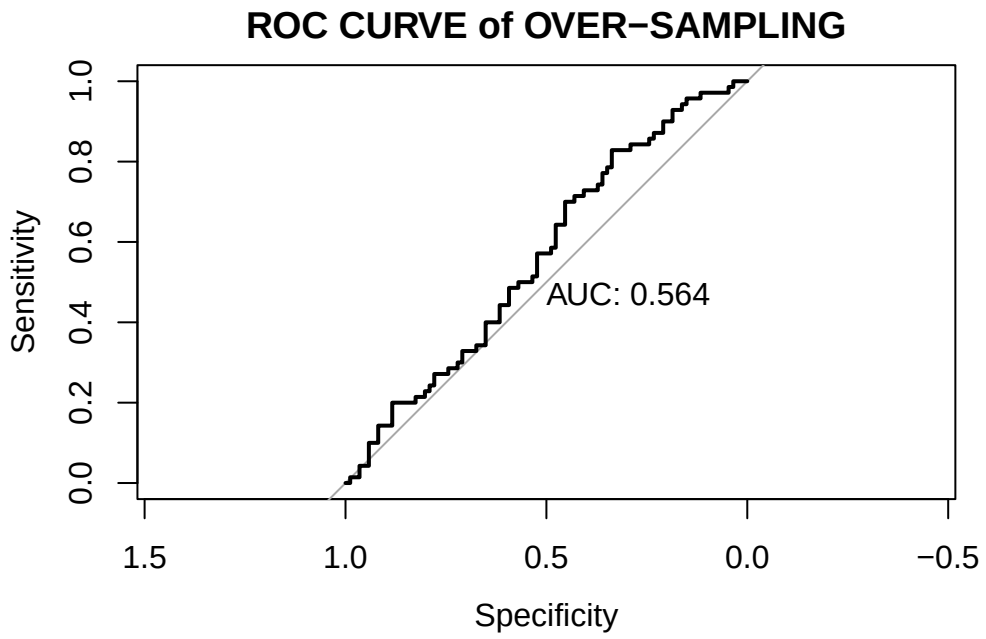
```
Setting levels: control = 0, case = 1
```

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

```
auc(roc_curve)
```

```
Area under the curve: 0.5645
```

```
plot(roc_curve, print.auc = TRUE , col = "black" , main= "ROC CURVE of OVER-SAMPLING")
```




**Under-Sampling:**

```
set.seed(123)
train_indexU <- createDataPartition(customer_undersampled$label, p = 0.8, list = FALSE)
train_balancedU <- customer_undersampled[train_indexU, ]
test_balancedU  <- customer_undersampled[-train_indexU, ]
model_under <- glm(label ~ ., data= train_balancedU , family = "binomial")
predict_under <- predict(model_under , test_balancedU[,-1], type = "response")
head(predict_under)
```

```
        1         3         9        15        18        22
0.2990662 0.2611045 0.2399256 0.0247614 0.3919241 0.2016201
```

```
probabilities_under <- predict(model_under, newdata = test_balancedU, type = "response")
roc_curve <- roc(test_balancedU$label , probabilities_under)
```

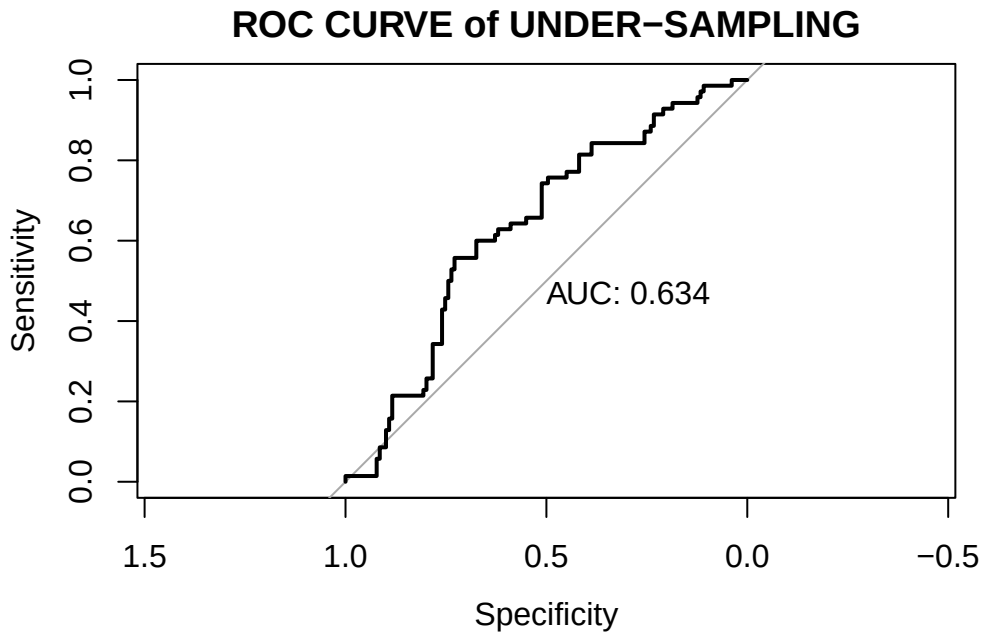
```
Setting levels: control = 0, case = 1
```

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

```
auc(roc_curve)
```

```
Area under the curve: 0.6342
```

```
plot(roc_curve, print.auc = TRUE , col = "black" , main= "ROC CURVE of UNDER-SAMPLING")
```




RESULT: Under-Sampling performed better, because higher values of AUC indicate better performance.