Advanced Methods for Regression and Classification

Exercise 5

Bosse Behrens , st.id: 12347333

First we laod the data

```r
library(ROCit)
data("Loan")
```

Task 1

We take a look at the structrue of our data.

```r
str(Loan)
```

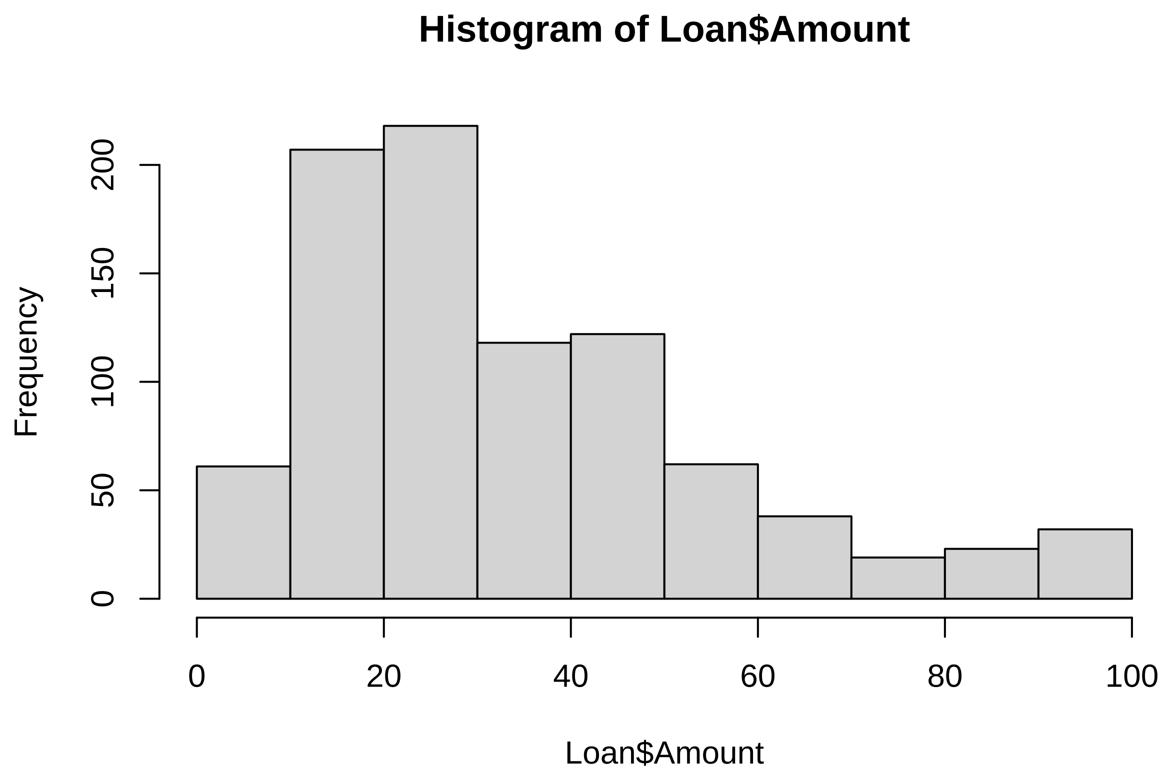
```
## 'data.frame':    900 obs. of  9 variables:
##  $ Amount : num  67.6 23 54 24.3 43.2 ...
##  $ Term   : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ IntRate: num  0.184 0.12 0.117 0.173 0.172 ...
##  $ ILR    : num  0.035 0.032 0.032 0.034 0.034 0.033 0.035 0.03 0.031 0.034 ...
##  $ EmpLen : Factor w/ 5 levels "A","B","C","D",..: 4 4 4 1 1 2 4 4 2 4 ...
##  $ Home   : Factor w/ 3 levels "MORTGAGE","OWN",..: 3 3 1 3 1 3 3 1 1 1 ...
##  $ Income : num  126400 30900 111900 66000 71900 ...
##  $ Status : Factor w/ 2 levels "CO","FP": 1 1 2 2 1 2 2 2 2 2 ...
##  $ Score  : num  201 180 162 197 203 ...
```

```r
summary(Loan)
```

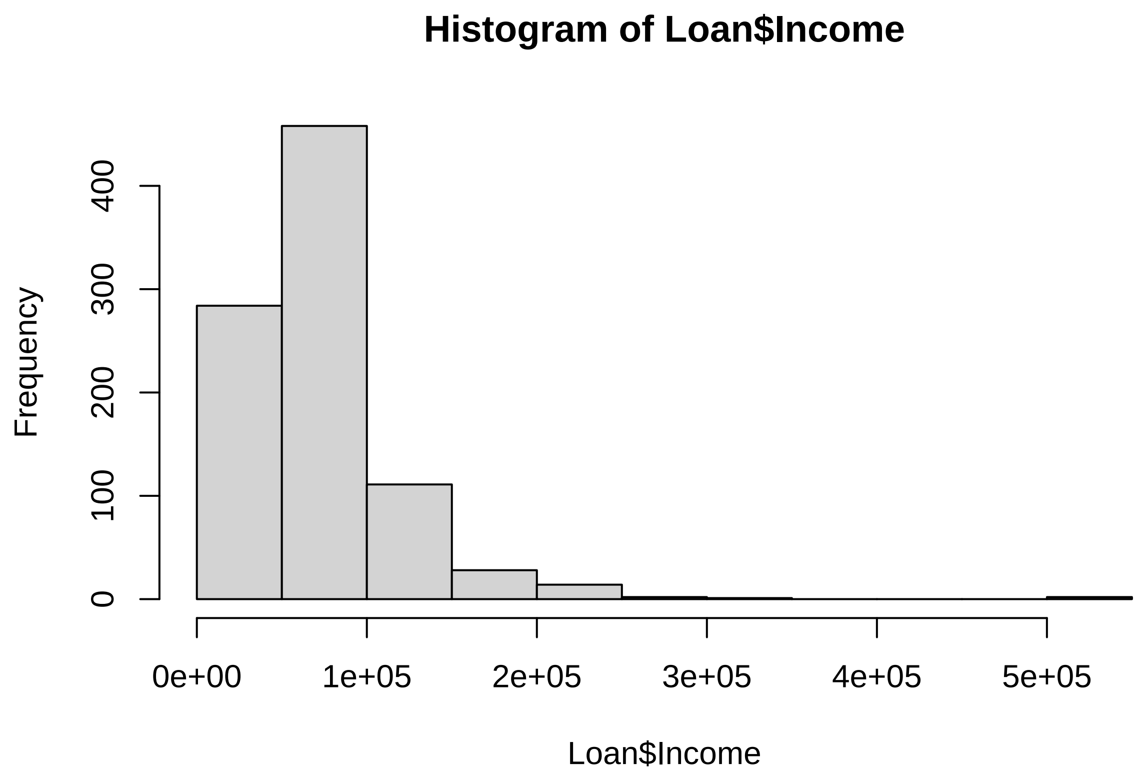
```
##      Amount           Term       IntRate            ILR           EmpLen
##  Min.   : 2.70   Min.   :36   Min.   :0.0830   Min.   :0.03000   A:198
##  1st Qu.:18.04   1st Qu.:36   1st Qu.:0.1219   1st Qu.:0.03200   B:198
##  Median :27.03   Median :36   Median :0.1513   Median :0.03300   C:141
##  Mean   :34.08   Mean   :36   Mean   :0.1529   Mean   :0.03353   D:305
##  3rd Qu.:43.34   3rd Qu.:36   3rd Qu.:0.1775   3rd Qu.:0.03500   U: 58
##  Max.   :94.59   Max.   :36   Max.   :0.2825   Max.   :0.04000
##        Home          Income        Status       Score
##  MORTGAGE:429   Min.   : 11900   CO:131   Min.   : -5.449
##  OWN     : 95   1st Qu.: 43900   FP:769   1st Qu.:169.358
##  RENT    :376   Median : 63000            Median :189.120
##                 Mean   : 72903            Mean   :187.440
##                 3rd Qu.: 86900            3rd Qu.:205.064
##                 Max.   :502000            Max.   :269.171
```

the summary suggests that Amount and Income are both very reight-skewed. We take a look at the ditributions.

```r
hist(Loan$Amount)
```



```r
hist(Loan$Income)
```



We apply some preprocessing. first we need to transform the response Status from a factor to a numerical variable to be able to regress on it (with 0 and 1 as new values). Amount and Income we log-transform so prevent the skewedness. At last the rpedictor term can be removed sicne it has the same value for every observation and carries therefore no variance and predictive value.

```
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag

## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
Loan <- Loan %>% select(-Term)
Loan <- Loan %>% mutate(Amount = log(Amount))
Loan <- Loan %>% mutate(Income = log(Income))
Loan$Status <- as.numeric(Loan$Status) - 1
```

Setting up the train/test split.

```
set.seed(12347333)
n <- nrow(Loan)
train <- sample(1:n, round((2/3) * n))
test<-(1:n)[-train]
```

```r
y_train <- Loan[train,"Status"]
y_test <- Loan[test, "Status"]
```

Building the model with lm().

```r
lm_model <- lm(Status ~ ., data = Loan[train, ])
```

Task 3

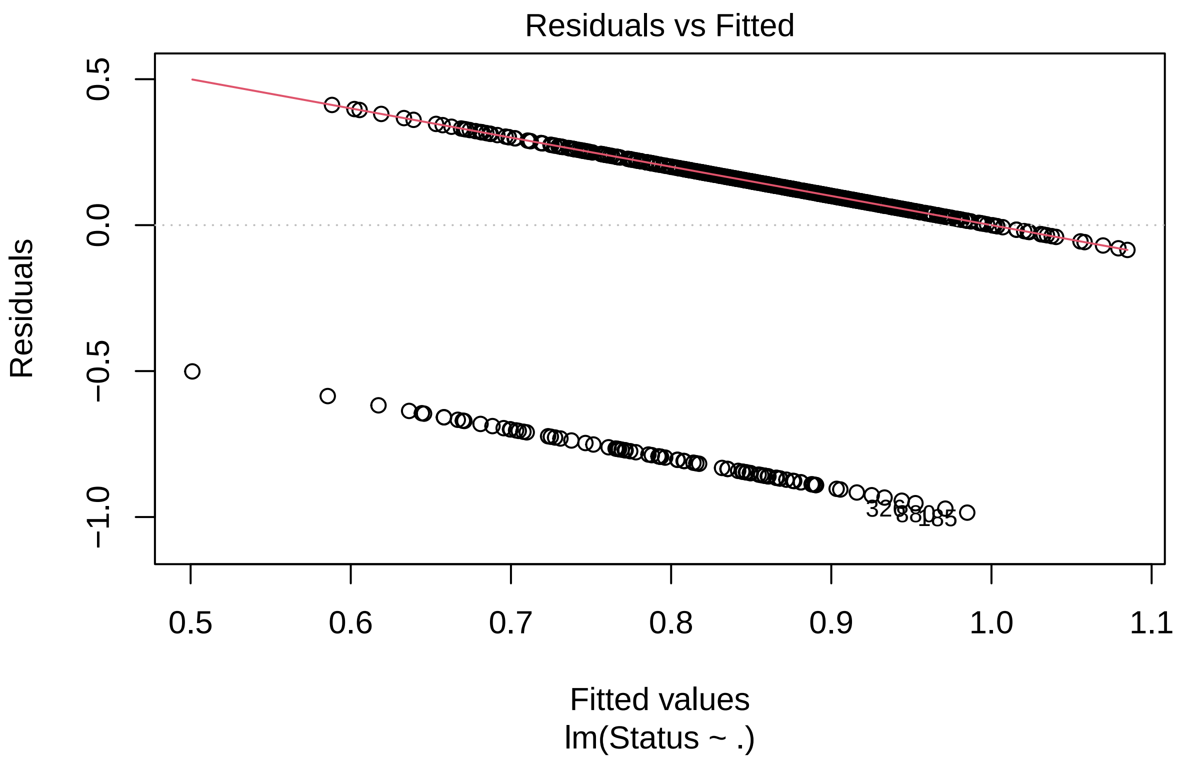
```r
summary(lm_model)
```

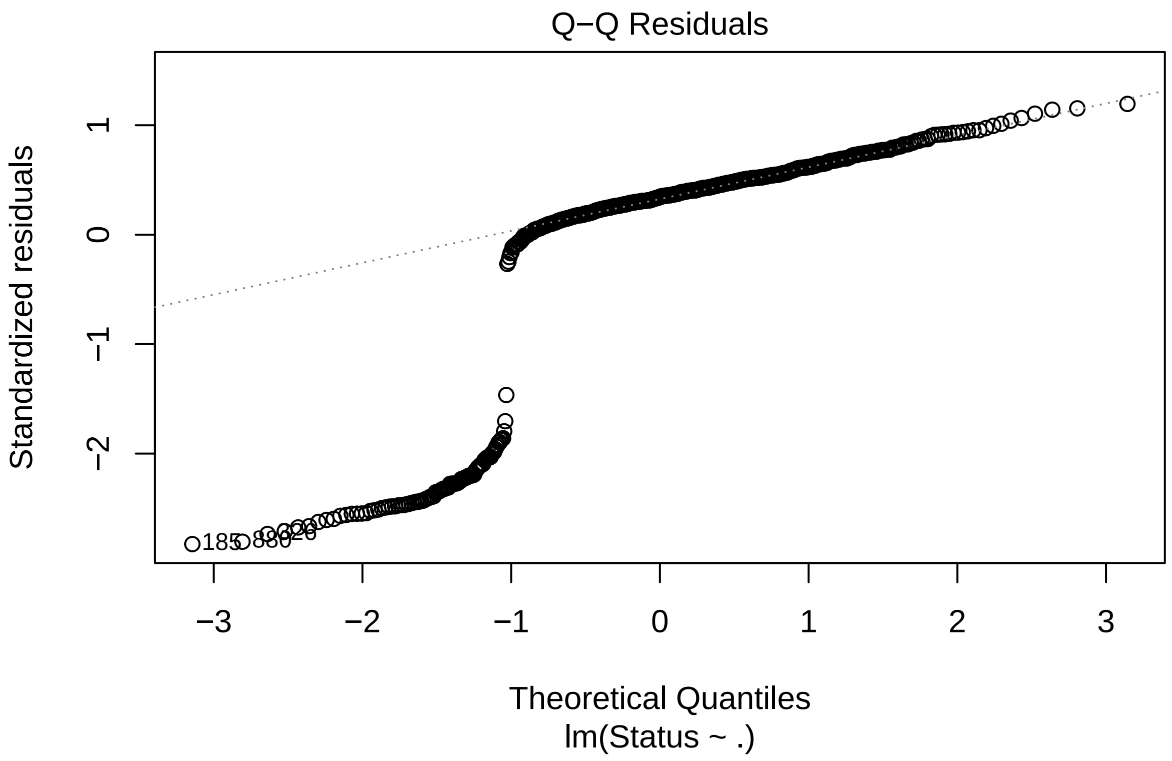
```
##
## Call:
## lm(formula = Status ~ ., data = Loan[train, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98487  0.04486  0.12060  0.18150  0.41173
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.5034421  1.4932255  -1.007   0.3144
## Amount      -0.0653548  0.0351585  -1.859   0.0635 .
## IntRate     -4.7122005  2.4876245  -1.894   0.0587 .
## ILR         65.2518894 49.9217807   1.307   0.1917
## EmpLenB     -0.0158530  0.0431309  -0.368   0.7133
## EmpLenC     -0.0366832  0.0486490  -0.754   0.4511
## EmpLenD      0.0217495  0.0396073   0.549   0.5831
## EmpLenU     -0.1294033  0.0687017  -1.884   0.0601 .
## HomeOWN      0.0546547  0.0527666   1.036   0.3007
## HomeRENT     0.0271796  0.0323619   0.840   0.4013
## Income       0.0927385  0.0600731   1.544   0.1232
## Score        0.0003605  0.0016303   0.221   0.8251
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3509 on 588 degrees of freedom
## Multiple R-squared:  0.06221,    Adjusted R-squared:  0.04467
## F-statistic: 3.546 on 11 and 588 DF,  p-value: 7.661e-05
```

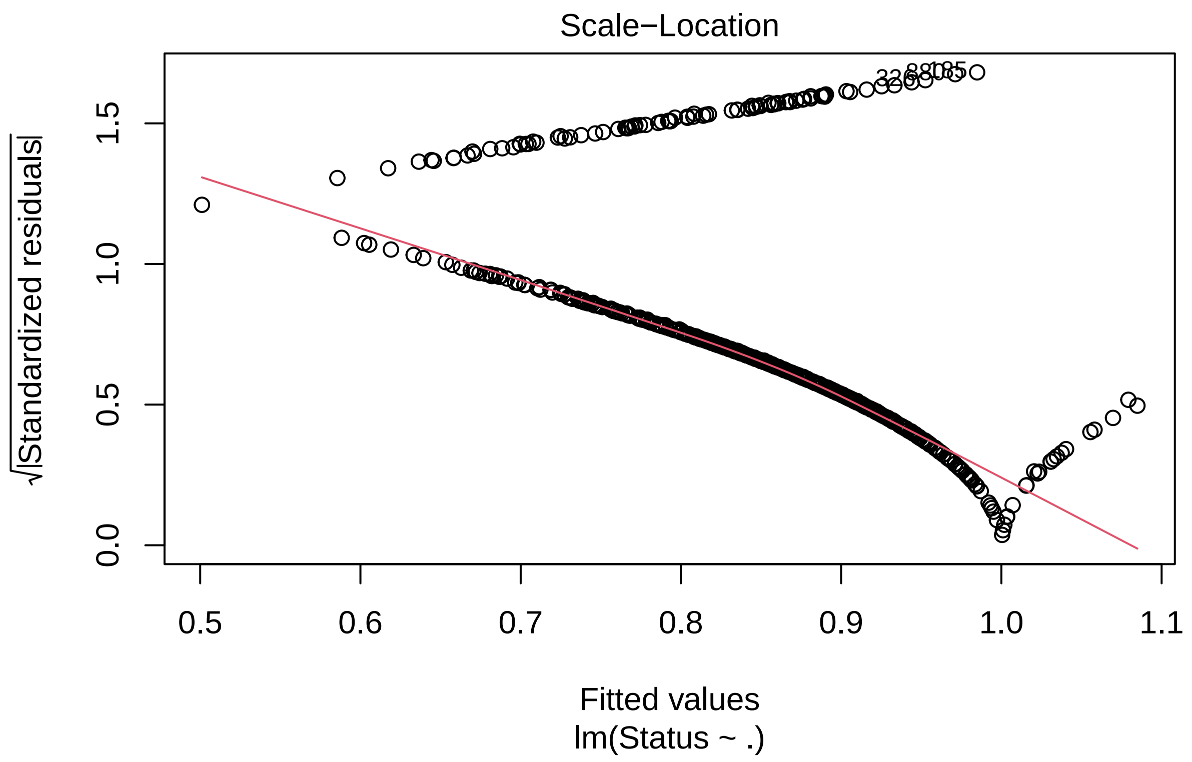
Only 3 predictors are significant and that only at the 0.1 significance level. Furthermore the adjusted R-squared is 0.04467. We can conclude that the model/predictors is/are not very well-suited and cannot explain most of the variance.

Task 3

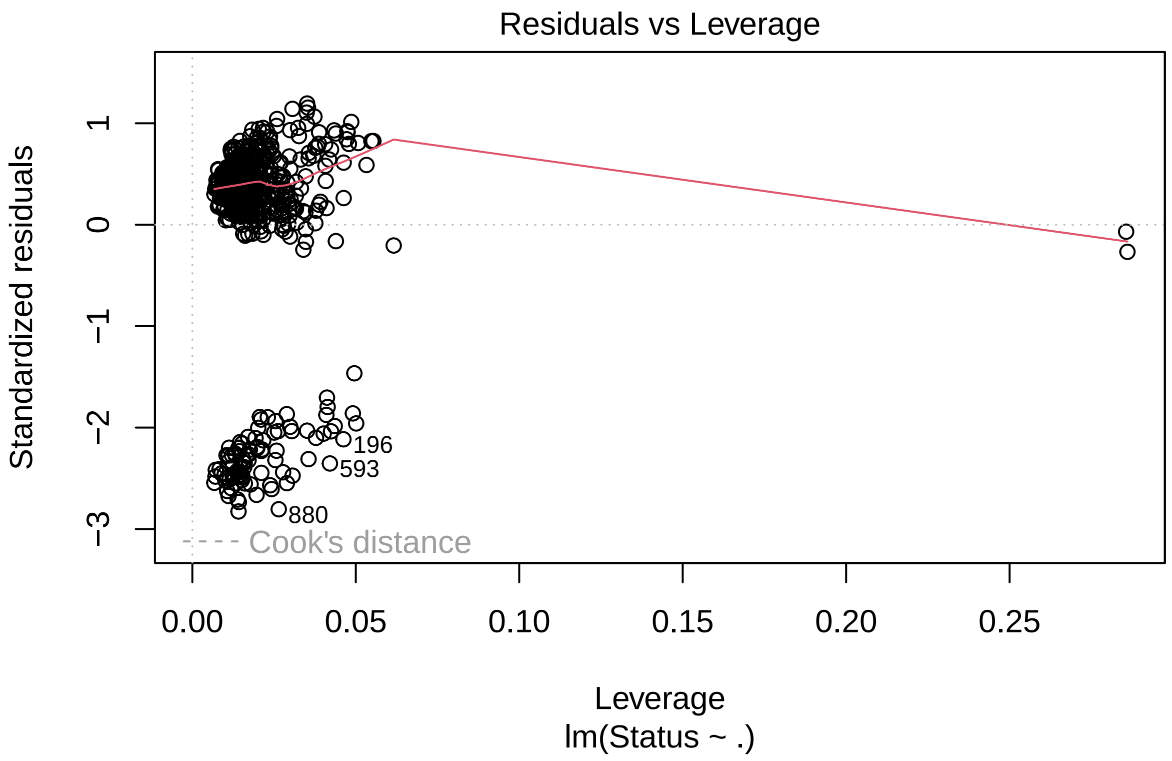
```r
plot(lm_model)
```


Residuals vs Fitted
Residuals
0.5
0.0
−0.5
−1.0
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Fitted values
lm(Status ~ .)
326
88
185


Q−Q Residuals
Standardized residuals
1
0
−1
−2
−3
−2
−1
0
1
2
3
185
880
26
Theoretical Quantiles
lm(Status ~ .)



Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Fitted values
lm(Status ~ .)
326
88
185

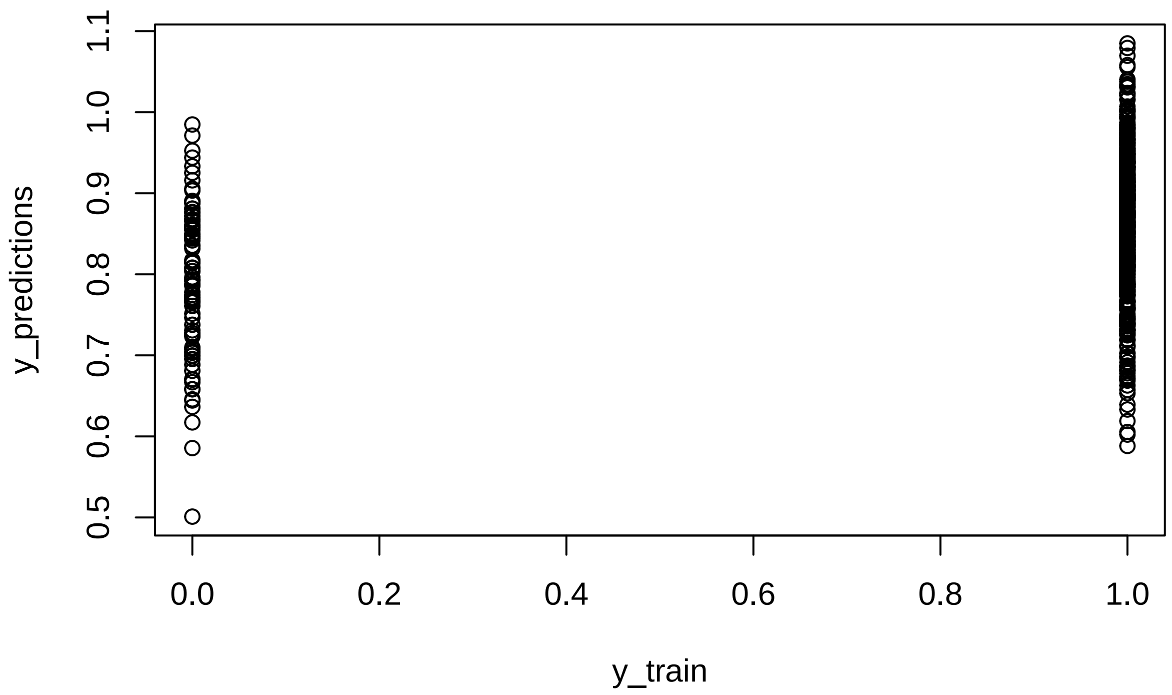


Wen can see in the residuals vs fitted values two lines that are the two classes. These lines run parallely and have a large overlap in terms of the fitted values. This is reason to worry about since it means the model cannnot really distinguish between the classes.

Task 4

```r
y_predictions <- predict(lm_model, newdata = Loan[train, ])

plot(y_train, y_predictions)
```



As noticed before the predictions for both classes have a very large overlap. A good cutoff value should divide both classes in their prediciton. This is not really possible in this case. A cutoff value optimized by some metric like the balanced accuracy can not be easily concluded by just looking at the plot so the only thing we can do is choosing the cutoff value so either all positives are classified as such or all negatives as negatives. To classify all positives correctly is the mroe common approach sow e choose to do that. The resulting cutoff value would be ≈ 0.58.

Task 5

We use the cotoff value for the classification for thre predicitons.

```r
cutoff <- 0.58  # Initial cutoff value
train_pred_class <- ifelse(y_predictions >= cutoff, 1, 0)

# Confusion Matrix
confusion_matrix <- table(Actual = Loan[train, ]$Status, Predicted = train_pred_class)
print(confusion_matrix)
```

```
##       Predicted
## Actual   0   1
##      0   1  90
##      1   0 509
```

We chose the cutoff value so at least all positives get classified correctly. the resulting confusion matrix shows that by doing so also almost all negatives get falsely classified as positives. The conclusion is that the model is very bad and distinguishing between the two classes.

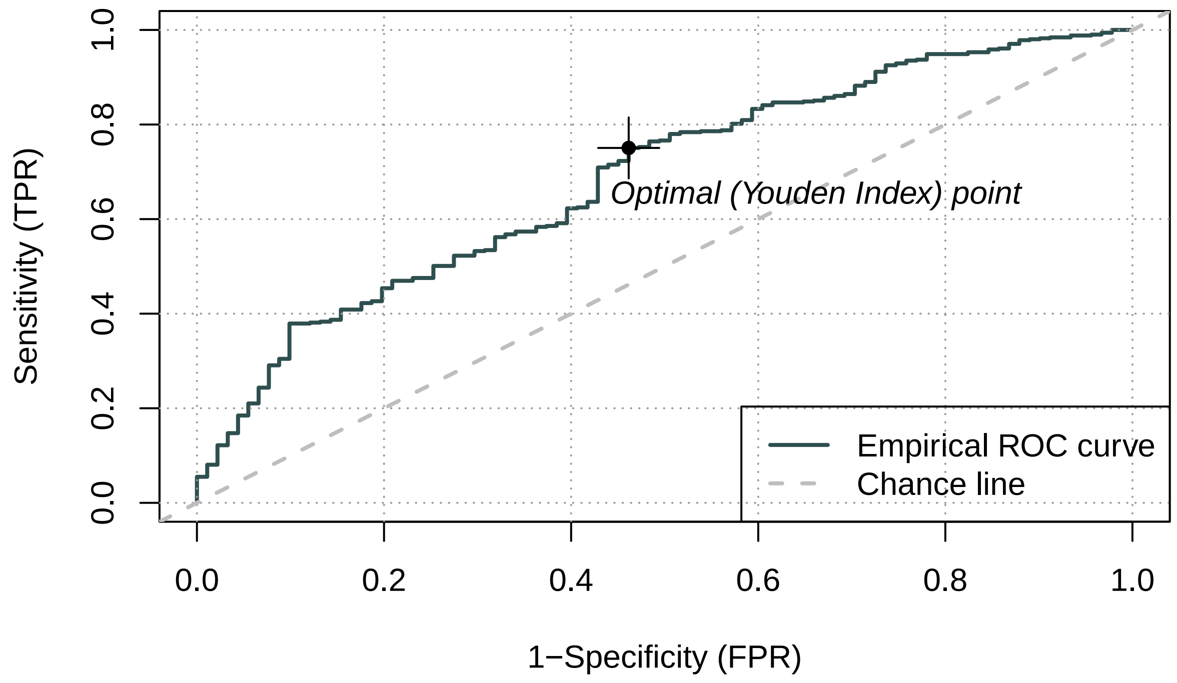
Task 6

```r
rocit_model <- rocit(class = Loan[train, ]$Status, score = y_predictions)

summary(rocit_model)
```

```
##
##  Method used: empirical
##  Number of positive(s): 509
##  Number of negative(s): 91
##  Area under curve: 0.6864
```

```r
plot(rocit_model)
```



The area under curve (AUC) value indicates the quality of the classification. This value should ideally be 1 which means it perfectly classifies everything. A value of 0.5 refers to random group assignment and thus a poor classifier. The value in our case is 0.6863 which is not very good and means the classifier does not perform well. The "top-leftmost" point which is indicated in the plot would taken to compute the optimal cutoff value.

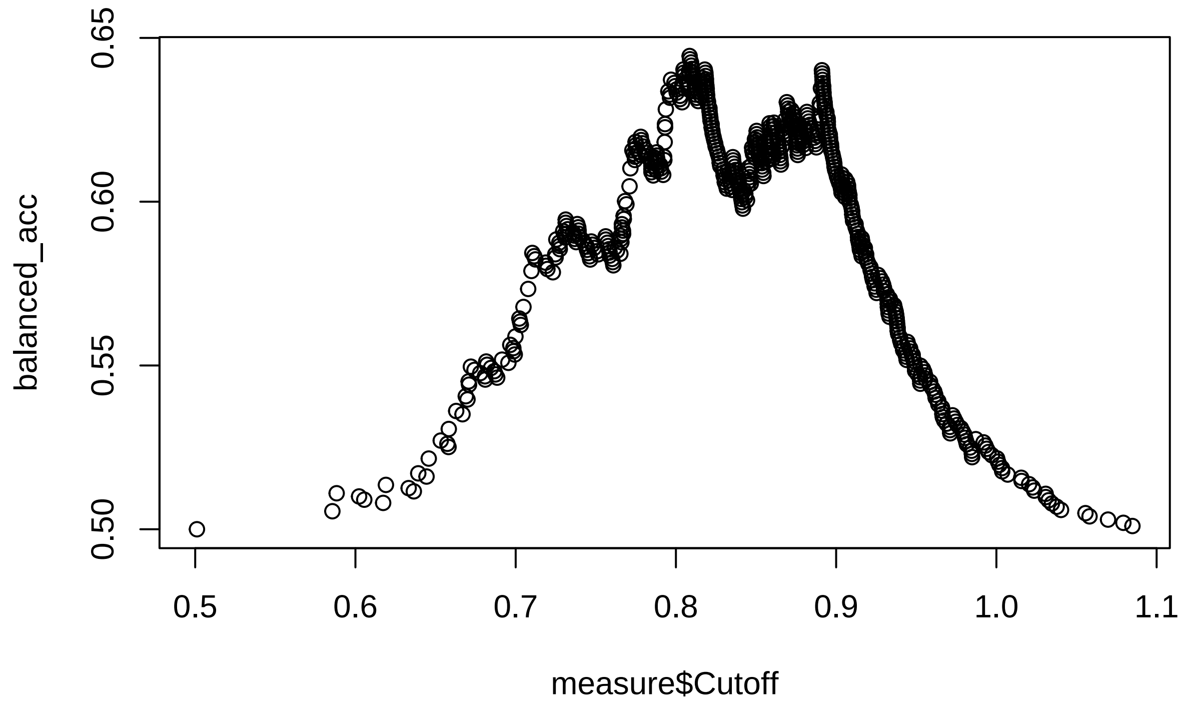
Task 7

We compute the table with the TPR and TNR values and compute the balanced accuracy using these. Then we plot the balanced accuracy vs the cutoff value. The optimal cutoff value would be the maximum in the plot, meaning where the balanced accuracy is the highest.

```r
measure <- measureit(y_predictions, Loan[train, ]$Status, measure=c("TPR","TNR"))
```

```r
balanced_acc <- (measure$TPR + measure$TNR)/2

plot(measure$Cutoff, balanced_acc)
```



```r
optimal_cutoff <- measure$Cutoff[which.max(balanced_acc)]
print(optimal_cutoff)
```

```
## [1] 0.8085497
```

We get the resulting optimal cutoff value of 0.8085497.

Task 8

Using the computed optimal cutoff value from the previous task we then use this to make the class predictions with our model on the test data.

```r
y_test_pred <- predict(lm_model, newdata = Loan[test, ])
test_pred_class <- ifelse(y_test_pred >= optimal_cutoff, 1, 0)

test_confusion_matrix <- table(Actual = Loan[test, ]$Status, Predicted = test_pred_class)
print(test_confusion_matrix)
```

```
##       Predicted
## Actual   0   1
##      0  20  20
##      1  78 182
```

Optimally only values on the diagonal of the confusion matrix should be non-zero and all others as low as possible. As we can see for the test data still many observations are not correctly predicted with only half of the negatives and about two thirds of the positives. This is slightly better than on the training data with initial cutoff value we got by just looking at the plot, but overall still not good. This means the model might just not be very well-suited for this classification problem or there are further underlying structures in the data we failed to capture so far.