Exercise 4

Nikolaus Czernin

```
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# install.packages("ROCit")
library("ROCit")
```

```
set.seed(11721138)
```

```
# Preprocessing
load("Loan.Rdata")
```

```
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
```

```
suppressWarnings(
  Loan %>%
    .[sapply(., is.numeric)] %>%
```

```
    cor(use = "complete.obs") %>%
    round(3)
)
```

```
##          Amount Term IntRate    ILR Income  Score
## Amount    1.000   NA  -0.146 -0.149  0.497 -0.035
## Term         NA    1      NA     NA     NA     NA
## IntRate  -0.146   NA   1.000  0.988 -0.215  0.850
## ILR      -0.149   NA   0.988  1.000 -0.213  0.839
## Income    0.497   NA  -0.215 -0.213  1.000 -0.584
## Score    -0.035   NA   0.850  0.839 -0.584  1.000
```

The summary shows that the column Term is constant, we therefore can drop it.

The correlation-table shows that some of the numeric columns are highly correlated. I opt to deselect them fully, to avoid having a rank-deficcient model later.

```
Loan <- Loan %>%
  # Scale only numeric columns
  mutate(across(where(is.numeric), ~ if (sd(.) > 0) scale(.) else .)) %>%
  # remove the constant variable Term
  select(-Term) %>%
  # remove the variables with colearity
  select(-Score) %>%
  # 1-hot encode the response levels
  mutate(Status = ifelse(Status == "CO", 1, 0))
```

```
N <- nrow(Loan)
train_ids <- sample(1:N, (N %/% 3) * 2)

train <- Loan[train_ids, ]

test <- Loan[-train_ids, ]

summary(Loan)
```

```
##       Amount.V1           IntRate.V1            ILR.V1        EmpLen
##  Min.   :-1.4501320   Min.   :-1.740472   Min.   :-1.885735   A:198
##  1st Qu.:-0.7410807   1st Qu.:-0.772345   1st Qu.:-0.816657   B:198
##  Median :-0.3257248   Median :-0.040650   Median :-0.282118   C:141
##  Mean   : 0.0000000   Mean   : 0.000000   Mean   : 0.000000   D:305
##  3rd Qu.: 0.4281549   3rd Qu.: 0.611405   3rd Qu.: 0.786960   U: 58
##  Max.   : 2.7965501   Max.   : 3.224602   Max.   : 3.459655
##        Home          Income.V1          Status
##  MORTGAGE:429   Min.   :-1.359480   Min.   :0.0000
##  OWN     : 95   1st Qu.:-0.646346   1st Qu.:0.0000
##  RENT    :376   Median :-0.220695   Median :0.0000
##                 Mean   : 0.000000   Mean   :0.1456
##                 3rd Qu.: 0.311927   3rd Qu.:0.0000
##                 Max.   : 9.562607   Max.   :1.0000
```

We need the response variable to be a numerical variable, so I transform it into 2 1-hot encodings. If we were to use only 1 such variable, which would be enough, 1 could signify a loan being charged-off, which is typically the thing you would want to predict for credit risk estimation.

I mean-scaled all numeric variables, because their distributions were not all similar.

Least Squares Classification

```
lm.full <- lm(Status~., data=train)
lm.full
```

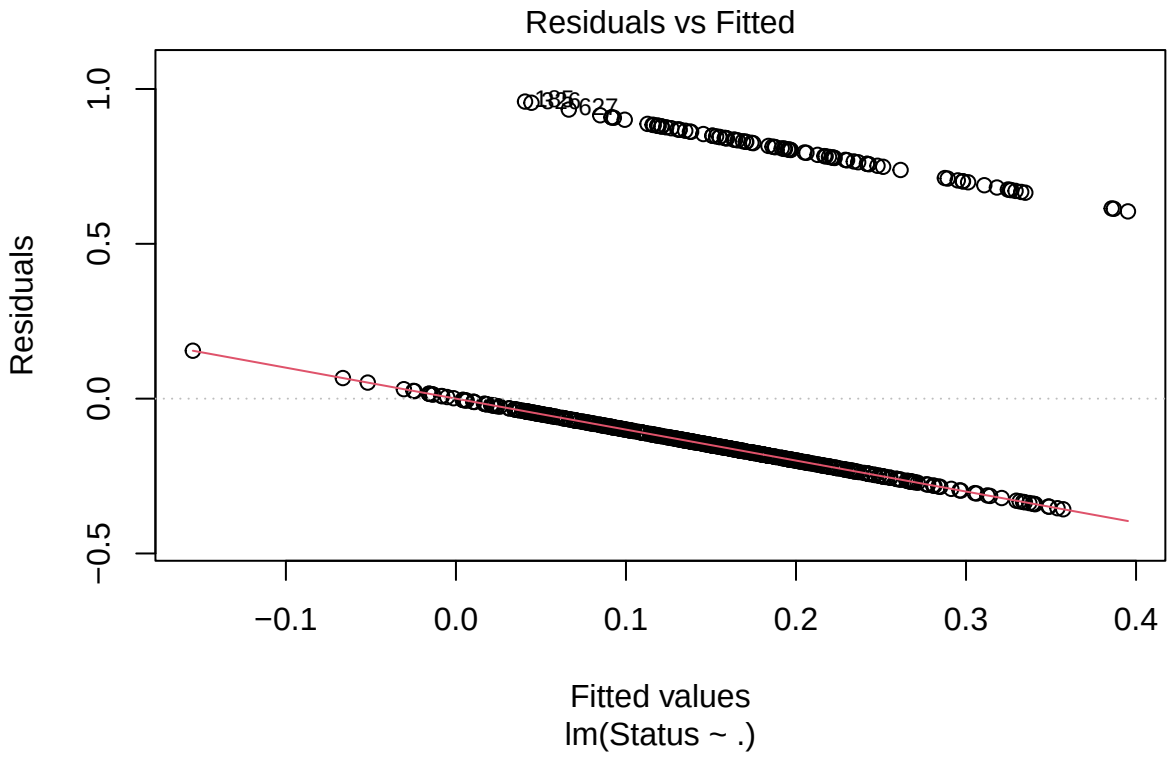
```
##
## Call:
## lm(formula = Status ~ ., data = train)
##
## Coefficients:
## (Intercept)       Amount      IntRate          ILR      EmpLenB      EmpLenC
##   0.1419978    0.0329514    0.1411922   -0.0724572    0.0220943    0.0381069
##     EmpLenD      EmpLenU      HomeOWN     HomeRENT       Income
##   0.0016131    0.1044852   -0.0543130   -0.0002162   -0.0193875
```

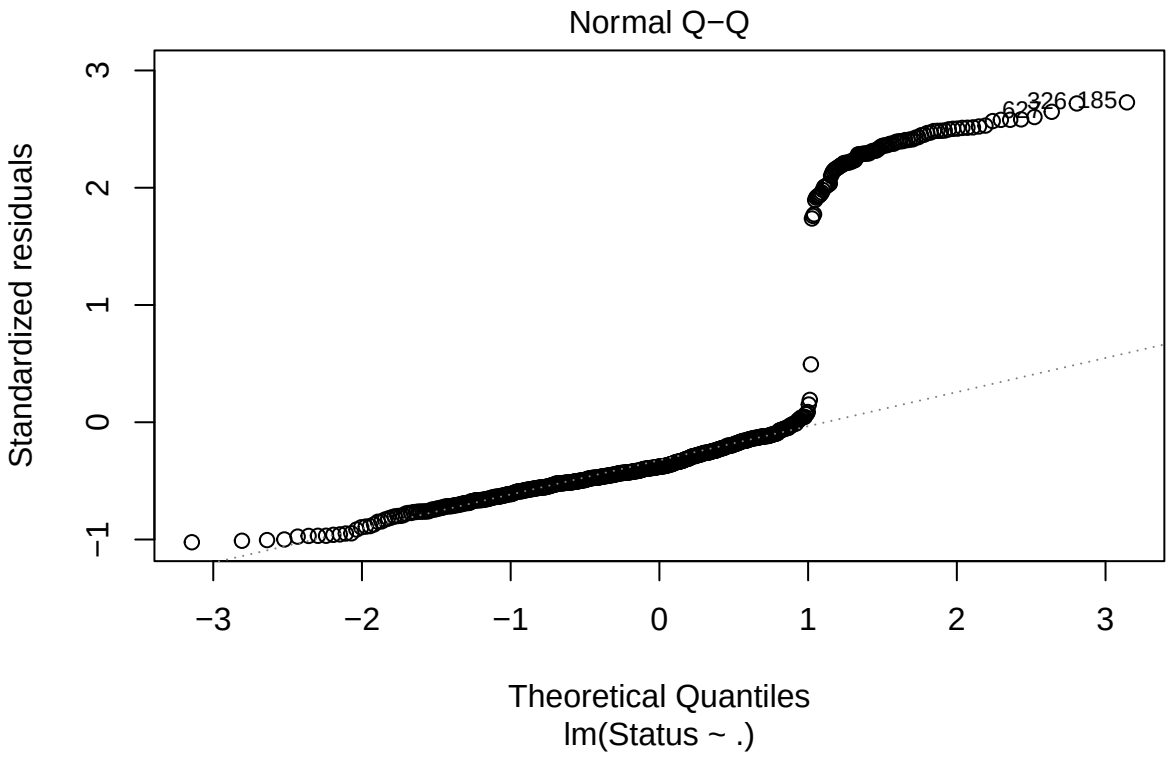
```
summary(lm.full)
```

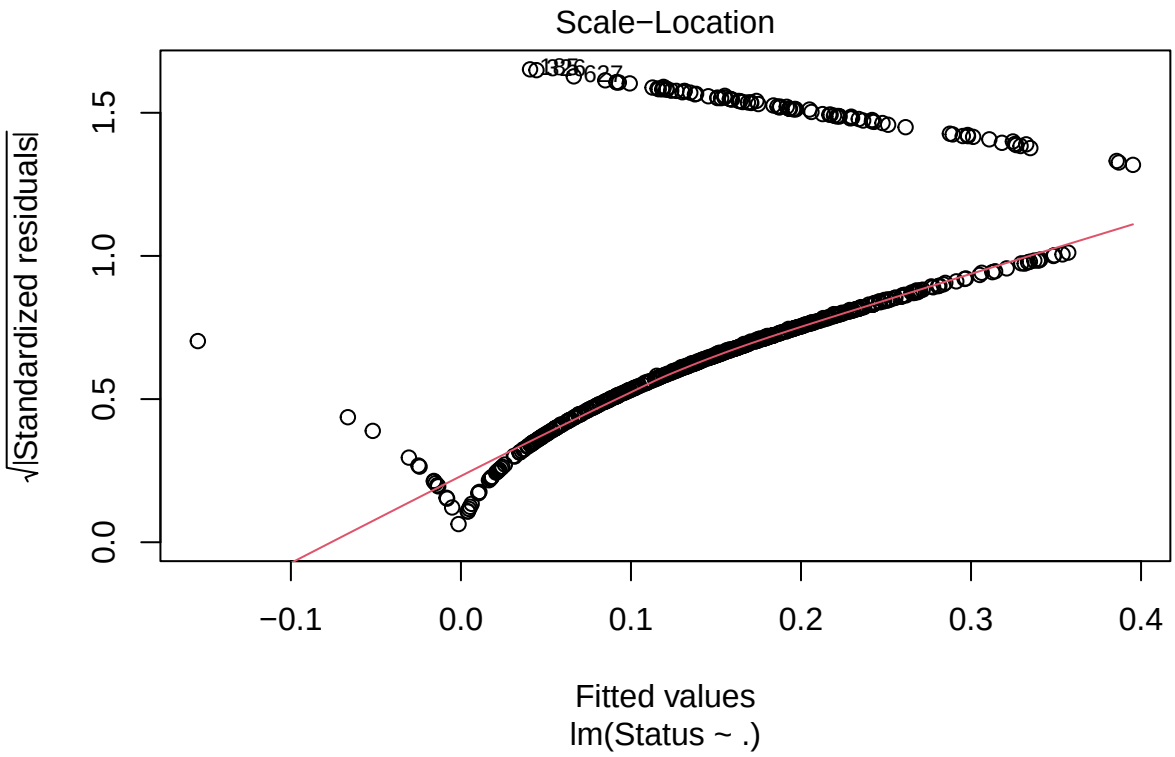
```
##
## Call:
## lm(formula = Status ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35718 -0.18350 -0.13339 -0.04511  0.95936
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1419978  0.0371258   3.825 0.000145 ***
## Amount       0.0329514  0.0170577   1.932 0.053869 .
## IntRate      0.1411922  0.0938875   1.504 0.133158
## ILR         -0.0724572  0.0937967  -0.772 0.440133
## EmpLenB      0.0220943  0.0439629   0.503 0.615457
## EmpLenC      0.0381069  0.0503337   0.757 0.449302
## EmpLenD      0.0016131  0.0409061   0.039 0.968558
## EmpLenU      0.1044852  0.0639655   1.633 0.102906
## HomeOWN     -0.0543130  0.0496987  -1.093 0.274909
## HomeRENT    -0.0002162  0.0324211  -0.007 0.994682
## Income      -0.0193875  0.0174204  -1.113 0.266198
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3541 on 589 degrees of freedom
## Multiple R-squared:  0.05202,    Adjusted R-squared:  0.03592
## F-statistic: 3.232 on 10 and 589 DF,  p-value: 0.0004532
```

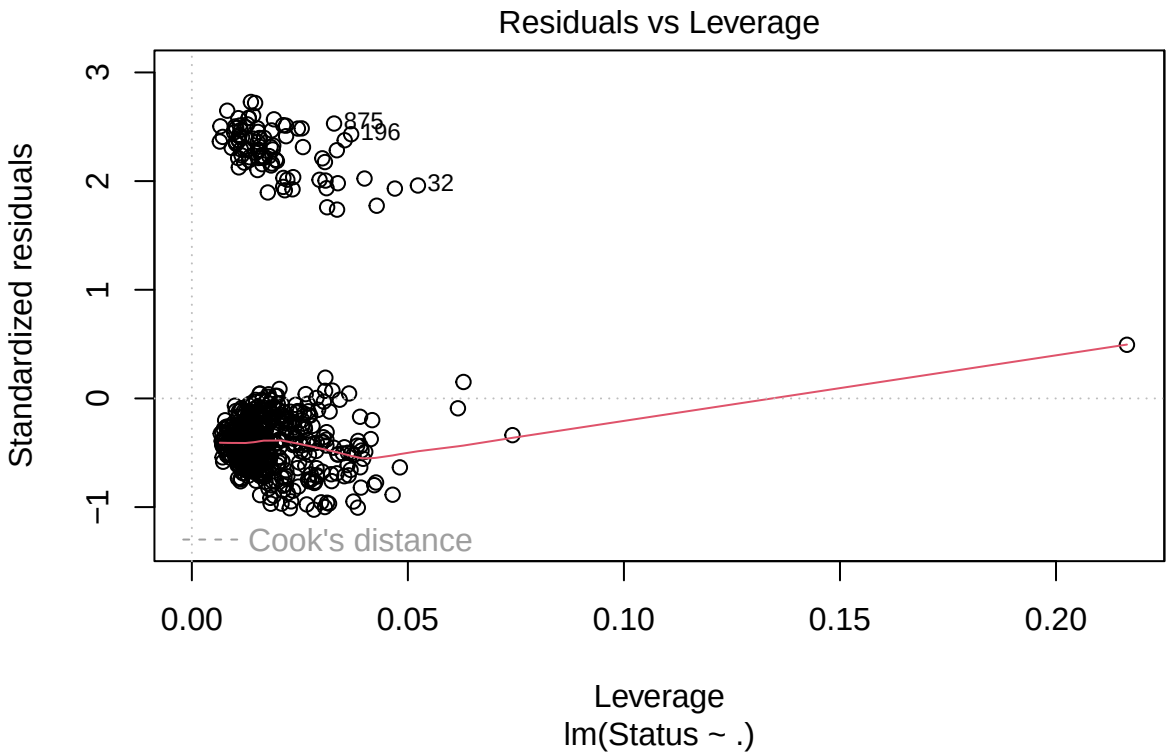
Because we included both response variable encodings, we get 2 symmetrical sets of coefficients for estimating either level of the response Status. 2 of the coefficients are significant, Score is even NA.

```r
plot(lm.full)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Status ~ .)
−3
−2
−1
0
1
2
3
185
326

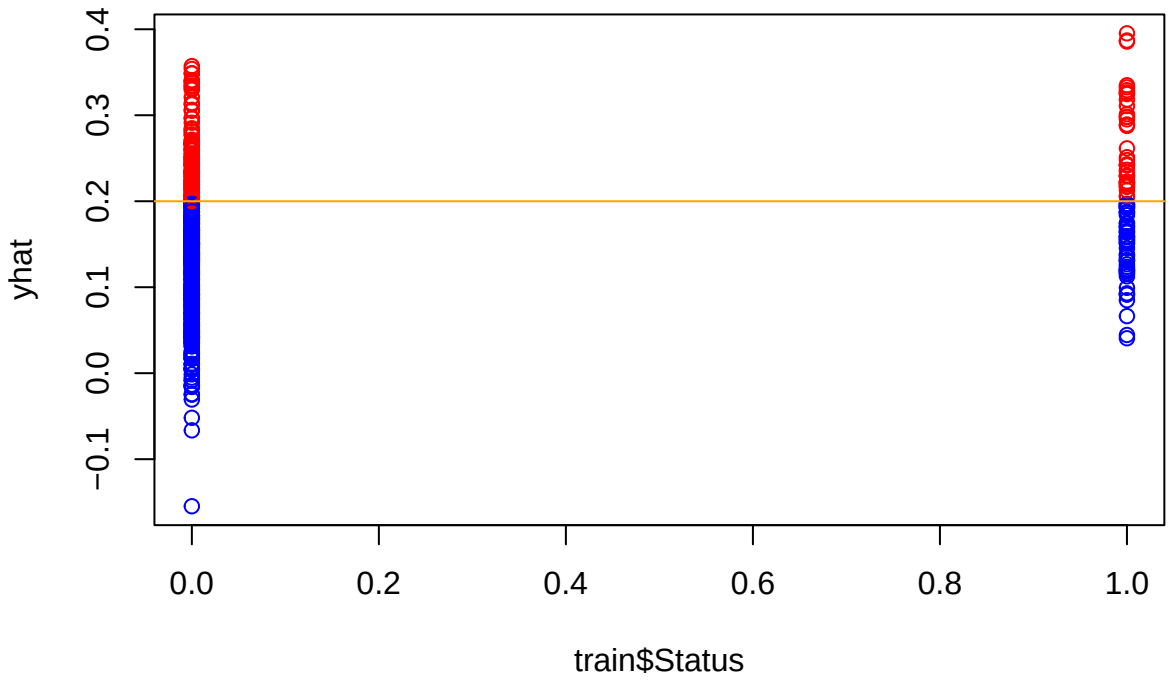

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
−0.1
0.0
0.1
0.2
0.3
0.4
Fitted values
lm(Status ~ .)



The residuals vs fitted plot shows two line patterns, only 1 of which is close to the red line. In a linear regression context, this would be alarming, but since we have 2 distint and mutually binary exclusive binary numbers to predict here, this is fine. Each line pattern corresponds with 1 single level of response.
The other 3 plots also show 2 distinct groups of observations.

Making predictions

```
cutoff <- 0.2
yhat <- predict(lm.full, newdata=train)
plot(train$Status, yhat, col=ifelse(yhat>cutoff, "red", "blue"))
abline(h=cutoff, col="orange")
```



It looks like something is going wrong here. The model estimates all all repsonse values to be under 0.4, with complete overlap between the values that are in reality 0 or 1. Visually, there is no clear pick for a position of a cutoff line to separate the two response classes.
For the sake of picking one, I went with 0.2.

```
cm <- table(train$Status, yhat>cutoff) %>% print()
```

```
##
##     FALSE TRUE
##   0   389  119
##   1    54   38
```

```
acc <- sum(diag(cm)) / sum(cm)
print(paste("Accuracy:", acc))
```

```
## [1] "Accuracy: 0.711666666666667"
```

```
recall <- cm[2,2] / sum(cm[2,])
print(paste("Recall:", recall %>% round(4)))
```

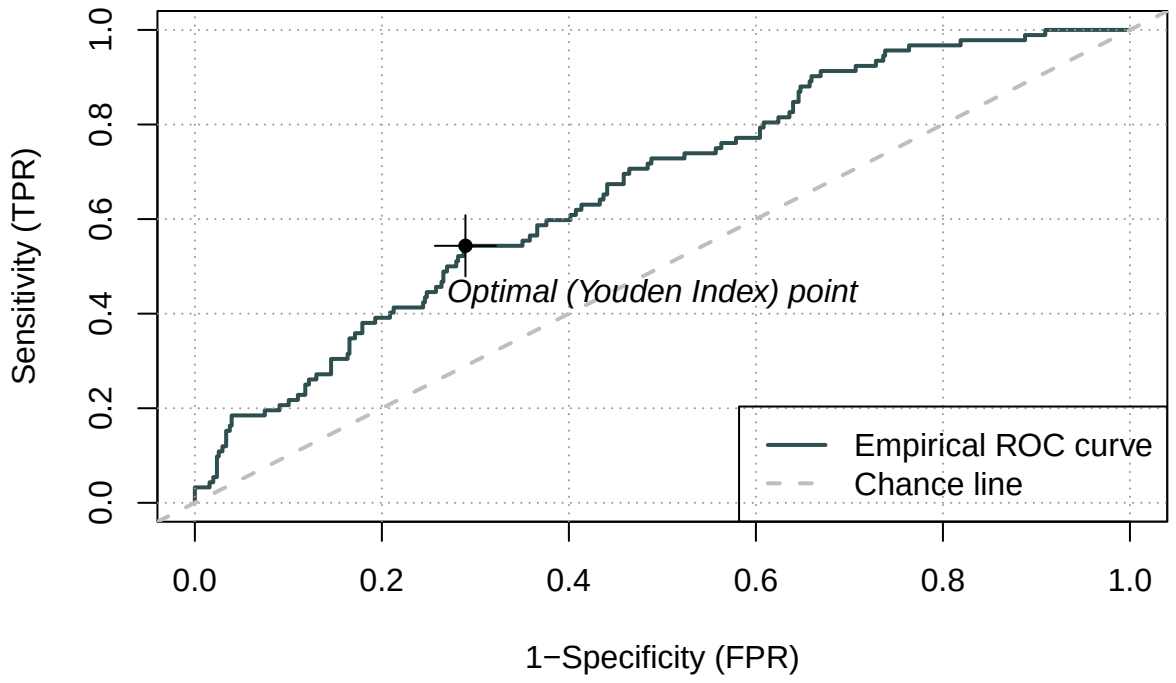
```
## [1] "Recall: 0.413"
```

The bottom row shows the values of the training data that are actually 1, i.e. real payback failures. The sum of the bottom row is much lower than the top row, i.e. the data labels are not balanced. Still, our model was not able to reflect this imbalance in its predictions. The column sums are roughly similar, ~300, meaning it was generally on the fence about the class of each observations, at the selected cutoff at least.

```
roc <- rocit(score=yhat, class=train$Status)
roc %>% summary()
```

```
##
##  Method used: empirical
##  Number of positive(s): 92
##  Number of negative(s): 508
##  Area under curve: 0.6669
```

```
roc %>% plot()
```



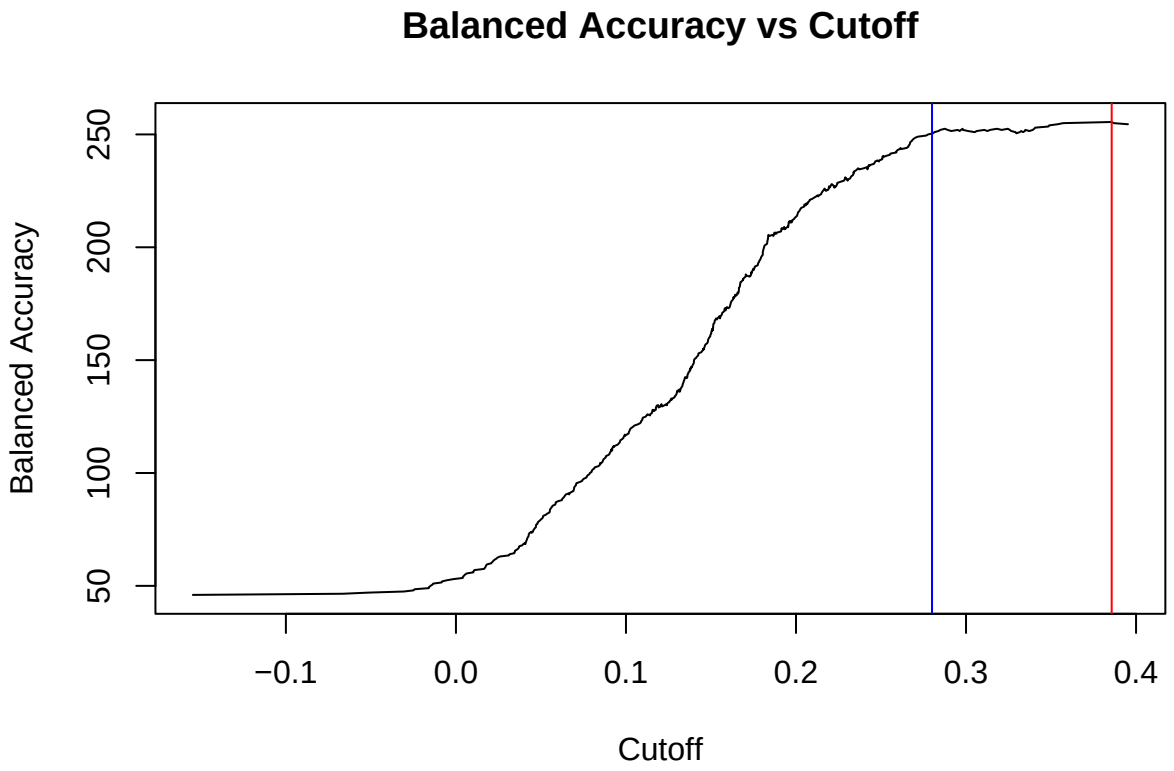
Our current AUC is 0.6669, not too close to 0.5, which would be proper bad, but there is certainly room for improvement.
The algorithm shows in the plot that the optimal cutoff point would be at 0.3, rather than 0.2 like I thought earlier.

```
roc2 <- measureit(yhat, train$Status, measure=c("TPR","TNR"))
roc2.BACC = (roc2$TP + roc2$TN) / 2


optimal_bacc <- which.max(roc2.BACC)
optimal_cutoff <- optimal_bacc %>% roc2$Cutoff[.]

plot(roc2$Cutoff, roc2.BACC, type = "l", xlab = "Cutoff", ylab = "Balanced Accuracy",
main = "Balanced Accuracy vs Cutoff")
```

```r
abline(v=0.28, col="blue")
abline(v=optimal_cutoff, col="red")
```



Now, interestingly, he maximum value of the balanced accuracy is at ~0.386 (red line), even higher than in the plot above. Just judging visually, the elbow point of the cutoff against the Balanced Accuracy seems to be ~0.28 (blue line).

Working with the new cutoff values

```r
cm <- table(train$Status, yhat>optimal_cutoff) %>% print()
```

```
##
##     FALSE TRUE
##   0   508    0
##   1    90    2
```

```r
acc <- sum(diag(cm)) / sum(cm)
print(paste("Accuracy:", acc %>% round(4)))
```

```
## [1] "Accuracy: 0.85"
```

```
recall <- cm[2,2] / sum(cm[2,])
print(paste("Recall:", recall %>% round(4)))
```

```
## [1] "Recall: 0.0217"
```

Well, now the accuracy is way higher than before and the model really does reflect the dataset's class imbalance. I would argue that the model is now not much better, as it barely ever classifies an observation to be an actual payback failure, which is exactly the type of case you would not want to miss in a real world scenario. This is observable in the recall, which is super low and was higher with the cutoff at 0.2.

```
custom_cutoff <- 0.28
cm <- table(train$Status, yhat>custom_cutoff) %>% print()
```

```
##
##     FALSE TRUE
##   0   484   24
##   1    75   17
```

```
acc <- sum(diag(cm)) / sum(cm)
print(paste("Accuracy:", acc %>% round(4)))
```

```
## [1] "Accuracy: 0.835"
```

```
recall <- cm[2,2] / sum(cm[2,])
print(paste("Recall:", recall %>% round(4)))
```

```
## [1] "Recall: 0.1848"
```

My updated custom pick for a cutoff, 0.28, does a little better, with a slightly improved Recall, but still probably problematic in a real world application.