Exercise 4

Tobias Raidl, 11717659

2023-11-21

Contents

1

Is any data preprocessing necessary or advisable? Categorical variables EmpLen, Home and Status need to be numerically encoded. I use one-hot-encoding for EmpLen and Home and binary encoding for Status because it only contains 2 unique classes.

```
mtrc_nr = 11717659
library(ROCit)
```

```
## Warning: package 'ROCit' was built under R version 4.1.3
```

```
library(mltools)
```

```
## Warning: package 'mltools' was built under R version 4.1.3
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.1.3
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df = Loan
df = one_hot(as.data.table(df), cols=c("EmpLen", "Home"))
df$Status = ifelse(df$Status == "CO", 1, 0)
df = select(df, -c(Term, EmpLen_U, Home_RENT, Score))

set.seed(mtrc_nr)
sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(2/3,1/3))
train  <- df[sample, ]
test   <- df[!sample, ]

model = lm(Status~., train)
```

2

What do you conclude when inspecting the outcome of summary()? Some coefficients are NA. I suspect the cause of this being some variables being linearly related to others. (multicollinearity)

```
str(df)
```

```
## Classes 'data.table' and 'data.frame':   900 obs. of  11 variables:
##  $ Amount      : num  67.6 23 54 24.3 43.2 ...
##  $ 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_A    : int  0 0 0 1 1 0 0 0 0 0 ...
##  $ EmpLen_B    : int  0 0 0 0 0 1 0 0 1 0 ...
##  $ EmpLen_C    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ EmpLen_D    : int  1 1 1 0 0 0 1 1 0 1 ...
##  $ Home_MORTGAGE: int  0 0 1 0 1 0 0 1 1 1 ...
##  $ Home_OWN    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Income      : num  126400 30900 111900 66000 71900 ...
##  $ Status      : num  1 1 0 0 1 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(model)
```

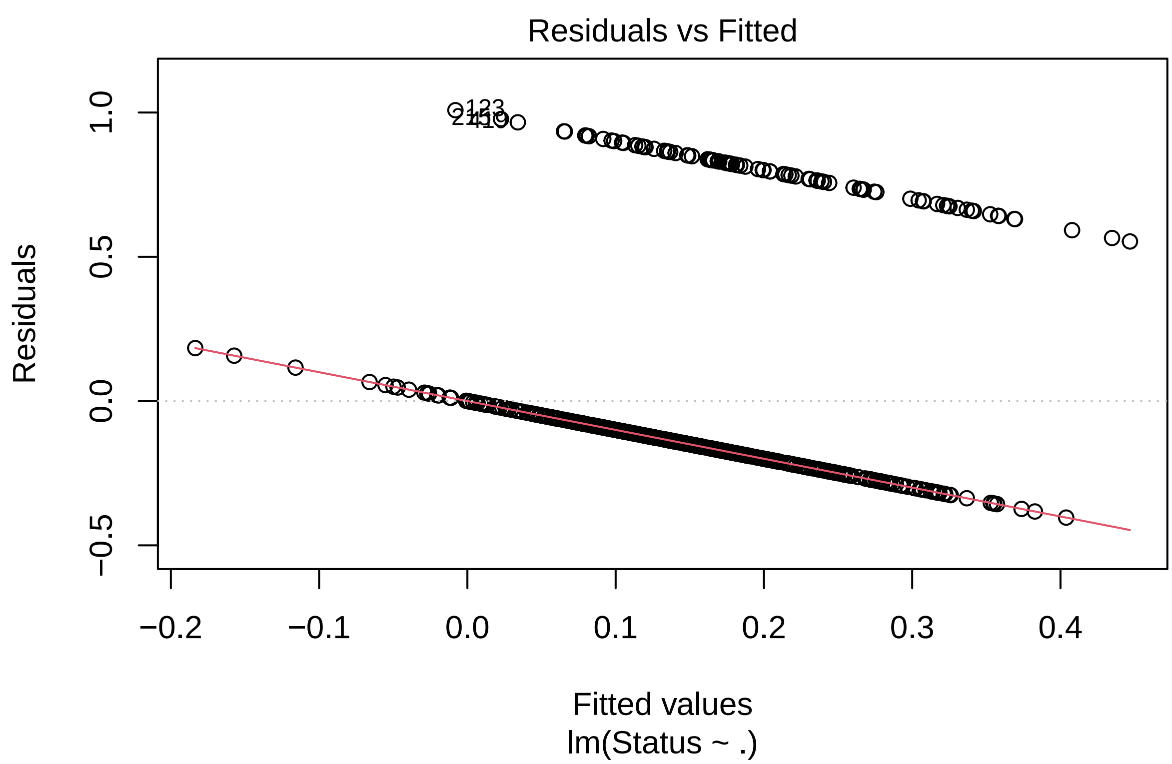
```
##
## Call:
## lm(formula = Status ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40385 -0.18471 -0.12258 -0.04668  1.00807
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.175e+00  1.343e+00   1.620   0.1059
## Amount         1.409e-03  7.808e-04   1.805   0.0717 .
```

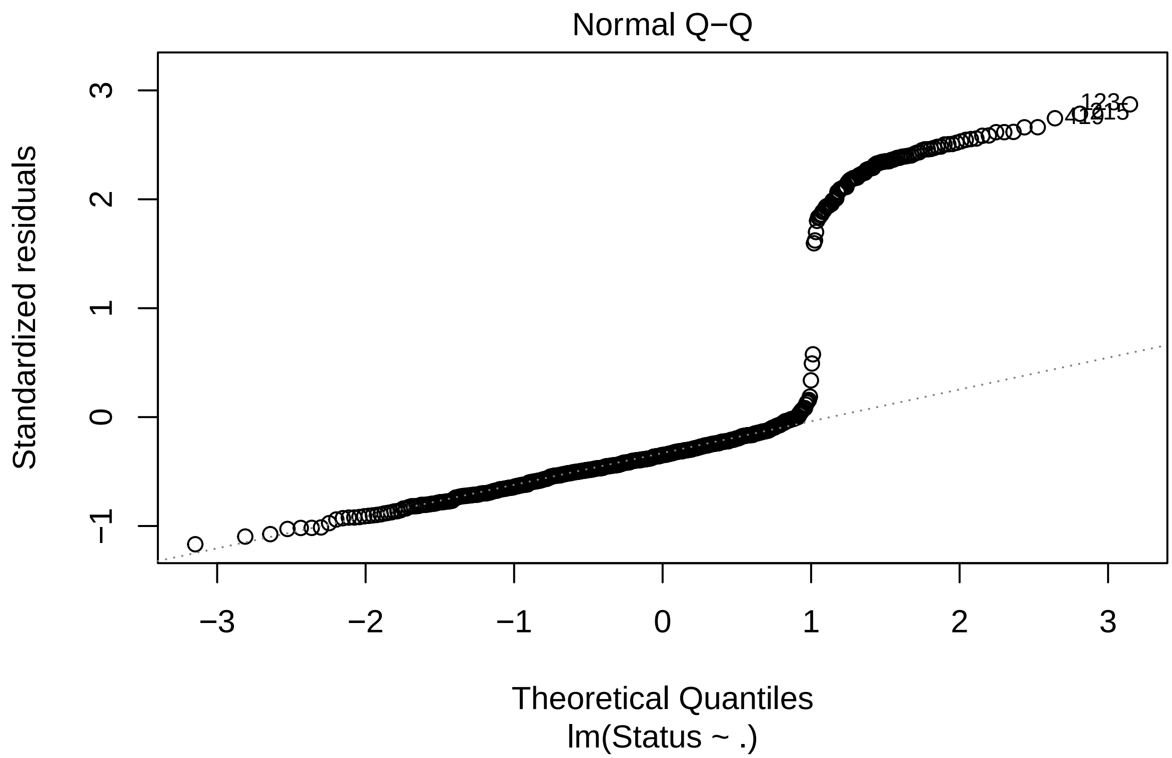
```
## IntRate        5.503e+00  2.354e+00   2.338   0.0197 *
## ILR           -8.255e+01  5.055e+01  -1.633   0.1030
## EmpLen_A      -1.162e-01  6.425e-02  -1.808   0.0711 .
## EmpLen_B      -1.123e-01  6.431e-02  -1.746   0.0813 .
## EmpLen_C      -4.979e-02  6.763e-02  -0.736   0.4619
## EmpLen_D      -1.425e-01  6.180e-02  -2.306   0.0214 *
## Home_MORTGAGE  9.268e-03  3.218e-02   0.288   0.7734
## Home_OWN      -3.641e-03  5.161e-02  -0.071   0.9438
## Income        -5.595e-07  3.547e-07  -1.577   0.1152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3534 on 596 degrees of freedom
## Multiple R-squared:  0.06308,    Adjusted R-squared:  0.04736
## F-statistic: 4.013 on 10 and 596 DF,  p-value: 2.444e-05
```

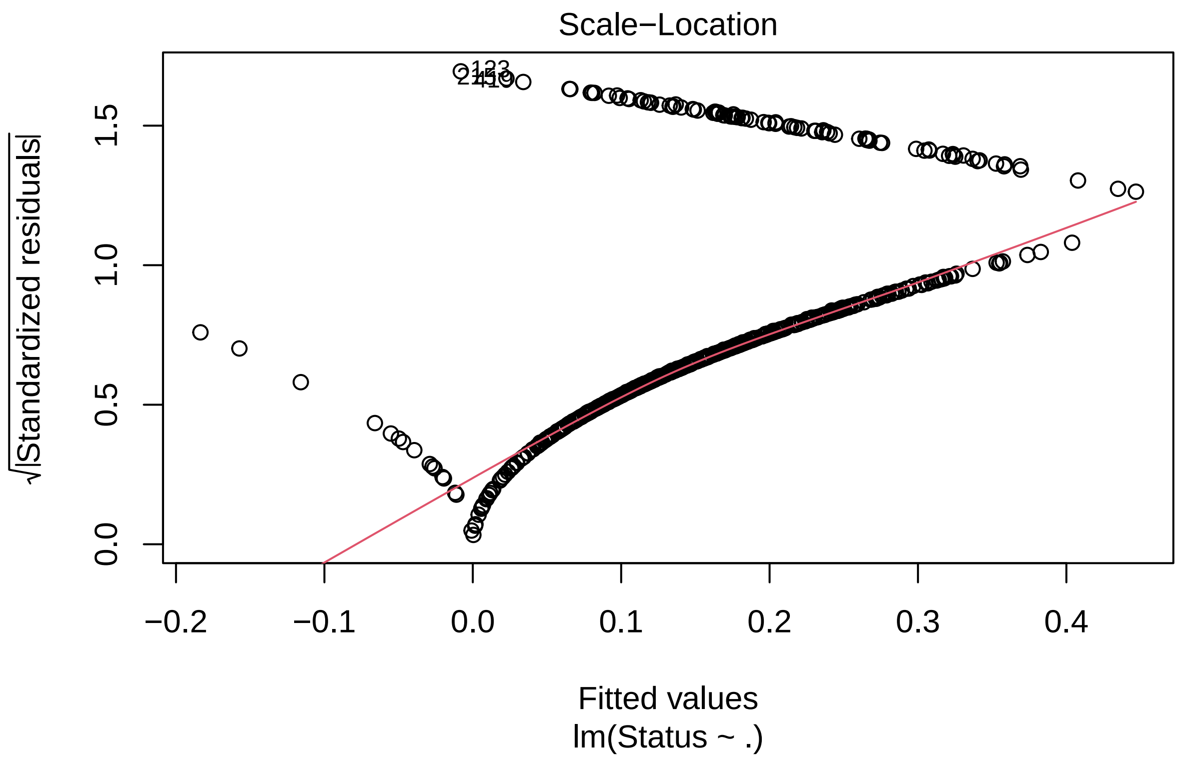
3

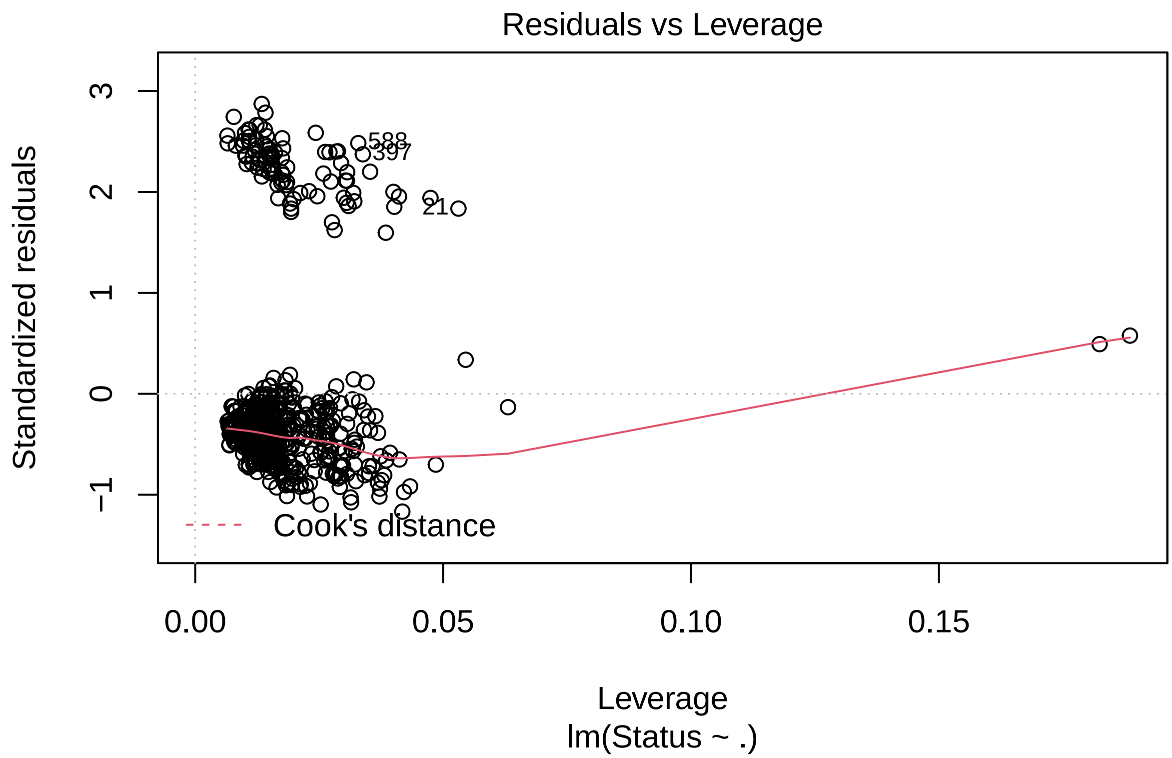
Shall we be worried looking at plot()? No because this is not a regression task.

```r
plot(model)
```




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

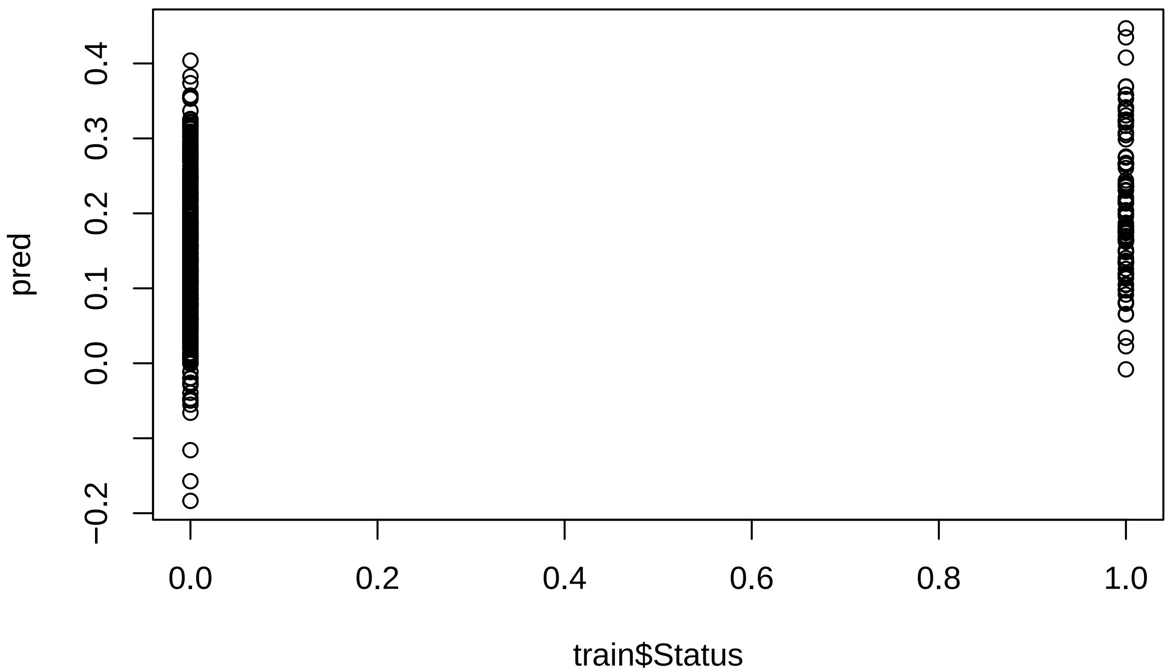

Scale−Location
√|Standardized residuals|
Fitted values
lm(Status ~ .)
0.0
0.5
1.0
1.5
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
123
2419



4

Which cutoff value would be useful in order to obtain reasonable class predictions? None as my model sucks, but i guess like 0.2 or somethihng like that.

```
pred = predict(model, train, type="response")
plot(x=train$Status, y=pred)
```




```
class.pred = as.numeric(pred>0.2)
```

5

Which conclusions can you draw from these numbers? The model either sucks or i made a major mistake. With a cutoff value of 0.2 we receive an accuracy of 0.71

```
t = table(train$Status, class.pred)
t
```

```
##    class.pred
##       0   1
##   0 384 129
##   1  50  44
```

```
accuracy = (t[1,1]+t[2,2])/sum(t)
paste("accuracy:", accuracy)
```

```
## [1] "accuracy: 0.705107084019769"
```

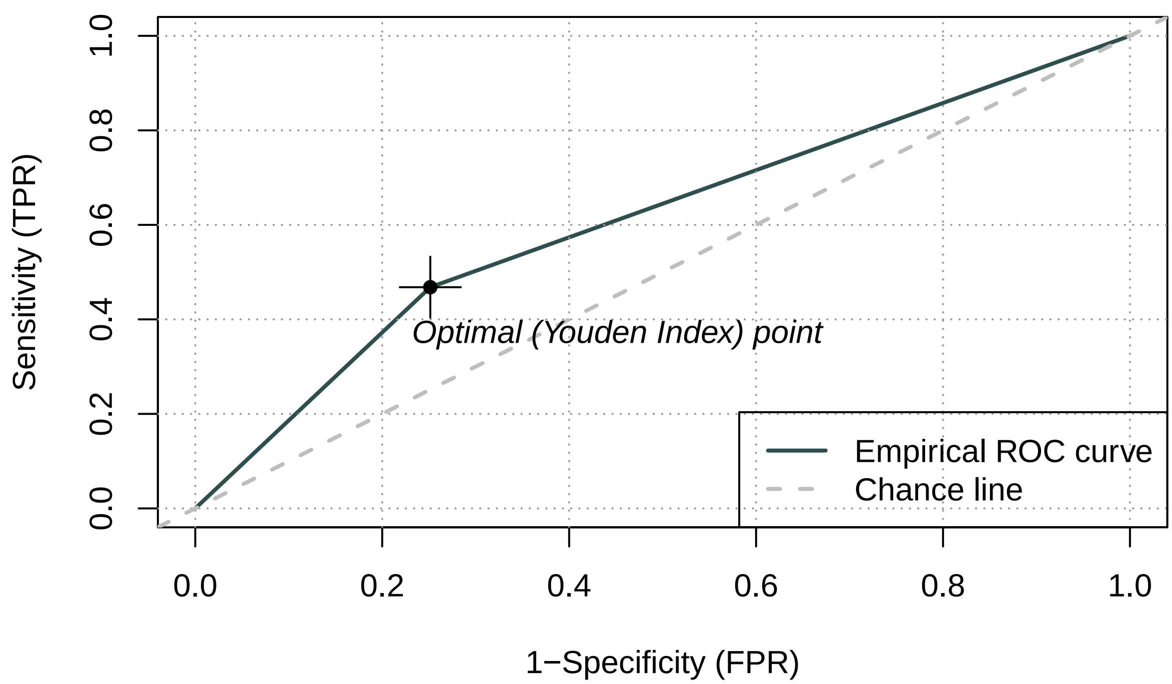
6

Which value would indicate the quality of your classifier? Is the classifier doing a good job? The AUC indicates the quality of our classifier. 1 would be ideal 0.5 would be the same as random picking. We receive an AUC of 0.6. This is visualized by the plot where the AUC is depicted as the area under the curve.

```r
roc = rocit(class.pred, train$Status)
summary(roc)
```

```
##
##  Method used: empirical
##  Number of positive(s): 94
##  Number of negative(s): 513
##  Area under curve: 0.6083
```

```r
plot(roc)
```

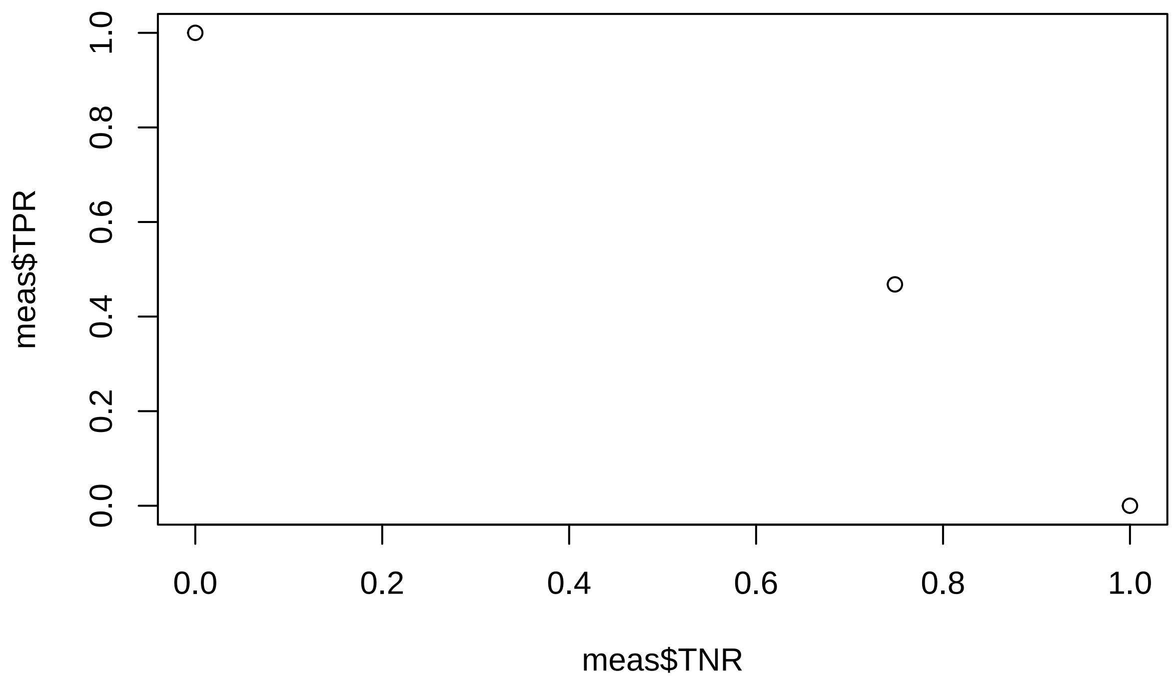


7

```r
meas = measureit(class.pred, train$Status, measure=c("TPR", "TNR"))
meas
```

```
##   Cutoff     Depth TP  FP  TN FN       TPR      TNR
## 1    Inf 0.0000000  0   0 513 94 0.0000000 1.000000
## 2      1 0.2850082 44 129 384 50 0.4680851 0.748538
## 3      0 1.0000000 94 513   0  0 1.0000000 0.000000
```

```r
plot(meas$TNR, meas$TPR)
```




```
cutoff.optim = 0.285
```

8

What are your final conclusions? We now get an accuracy of 0.82 which is an improvement of 10% using this optimal cutoff point instead of the informally estimated one.

```
pred = predict(model, test, type="response")
class.pred.optim = as.numeric(pred>cutoff.optim)
t = table(class.pred.optim, test$Status)
t
```

```
##
## class.pred.optim   0   1
##                0 239  35
##                1  17   2
```

```
accuracy = (t[1,1]+t[2,2])/sum(t)
paste("accuracy:", accuracy)
```

```
## [1] "accuracy: 0.822525597269625"
```