Exercise 5 - Advanced Methods for Regression and Classification

12433732 - Stefan Merdian

2024-11-17

```r
library(ROCit)
```

```
## Warning: package 'ROCit' was built under R version 4.4.2
```

```r
data("Loan", package = "ROCit")
```

```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
head(Loan)
```

```
##   Amount Term IntRate   ILR EmpLen     Home Income Status    Score
## 1  67.57   36  0.1838 0.035      D     RENT 126400     CO 200.9581
## 2  22.97   36  0.1198 0.032      D     RENT  30900     CO 179.6058
## 3  54.05   36  0.1166 0.032      D MORTGAGE 111900     FP 161.7622
## 4  24.32   36  0.1733 0.034      A     RENT  66000     FP 196.6619
## 5  43.24   36  0.1723 0.034      A MORTGAGE  71900     CO 203.4912
## 6  16.22   36  0.1355 0.033      B     RENT  27614     FP 186.3070
```

```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
```

Status is factor variable, therefore we convert it to a numeric data. We ensure that the variable are mapped as 0 & 1.

```
indicator <- Loan$Status

Loan$Status <- ifelse(Loan$Status == "CO", 0, 1)

#data.frame(Status = indicator, NumericStatus = Loan$Status)
```

1)

```
set.seed(123)

sample <- sample(c(TRUE, FALSE), nrow(Loan), replace=TRUE, prob=c(0.7,0.3))
train_data <- Loan[sample, ]
test_data <- Loan[!sample, ]
```

Is any data preprocessing necessary or advisable?

We have some issues here:

1. Least squares regression is influenced by the scale of the predictors because it minimizes the sum of squared residuals. Predictors with larger scales can disproportionately impact the regression coefficients, leading to biased results. To address this, all numeric variables should be standardized, ensuring that each predictor contributes fairly regardless of its original scale.
2. The Term column in the dataset contains only a single unique value (36). Variables with no variation do not provide any useful information for the model and can cause issues during regression. Therefore, the Term column should be removed before building the model.
3. Also the 'Term' predictor has just one value for all data points. So we can leave it out, since it wouldn't have any impact for our model.

We split the data before preprocessing to avoid data leakage. This ensures that the test set remains completely unseen during training, so the model is evaluated on data it has never encountered, giving a more realistic measure of its performance.

Preprocessing

```
# Remove the 'Term' column
train_data <- train_data[, !names(train_data) %in% "Term"]
test_data <- test_data[, !names(test_data) %in% "Term"]
```

```r
# # Standardizing
numeric_cols <- sapply(train_data, is.numeric)
numeric_cols["Status"] <- FALSE
train_scaled <- scale(train_data[numeric_cols])

scaling_params <- attr(train_scaled, "scaled:center")
scaling_scales <- attr(train_scaled, "scaled:scale")

test_scaled <- scale(test_data[numeric_cols], center = scaling_params, scale = scaling_scales)

train_data[numeric_cols] <- train_scaled
test_data[numeric_cols] <- test_scaled
```

Model

```r
lm_model <- lm(Status ~ ., data = train_data)
```

2)

```r
summary(lm_model)
```

```
##
## Call:
## lm(formula = Status ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02289  0.03543  0.11461  0.18501  0.37481
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.886783   0.035333  25.098  < 2e-16 ***
## Amount      -0.022789   0.015871  -1.436  0.15152
## IntRate     -0.322009   0.089372  -3.603  0.00034 ***
## ILR          0.254804   0.089544   2.846  0.00458 **
## EmpLenB     -0.016486   0.041920  -0.393  0.69426
## EmpLenC     -0.049511   0.045568  -1.087  0.27767
## EmpLenD     -0.024123   0.038317  -0.630  0.52922
## EmpLenU     -0.031020   0.061449  -0.505  0.61387
## HomeOWN      0.002512   0.046050   0.055  0.95652
## HomeRENT    -0.017876   0.030476  -0.587  0.55771
## Income       0.023655   0.016195   1.461  0.14461
## Score              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3409 on 623 degrees of freedom
## Multiple R-squared:  0.06273,    Adjusted R-squared:  0.04769
## F-statistic:  4.17 on 10 and 623 DF,  p-value: 1.317e-05
```

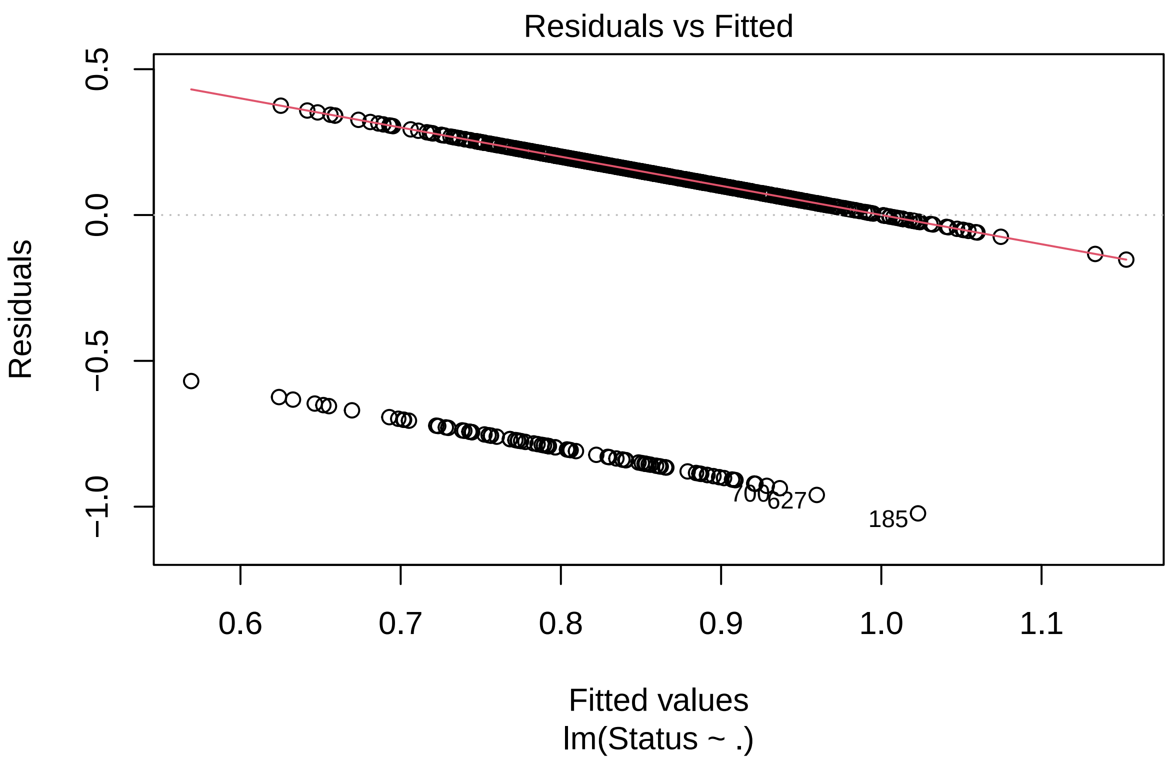
What do you conclude?

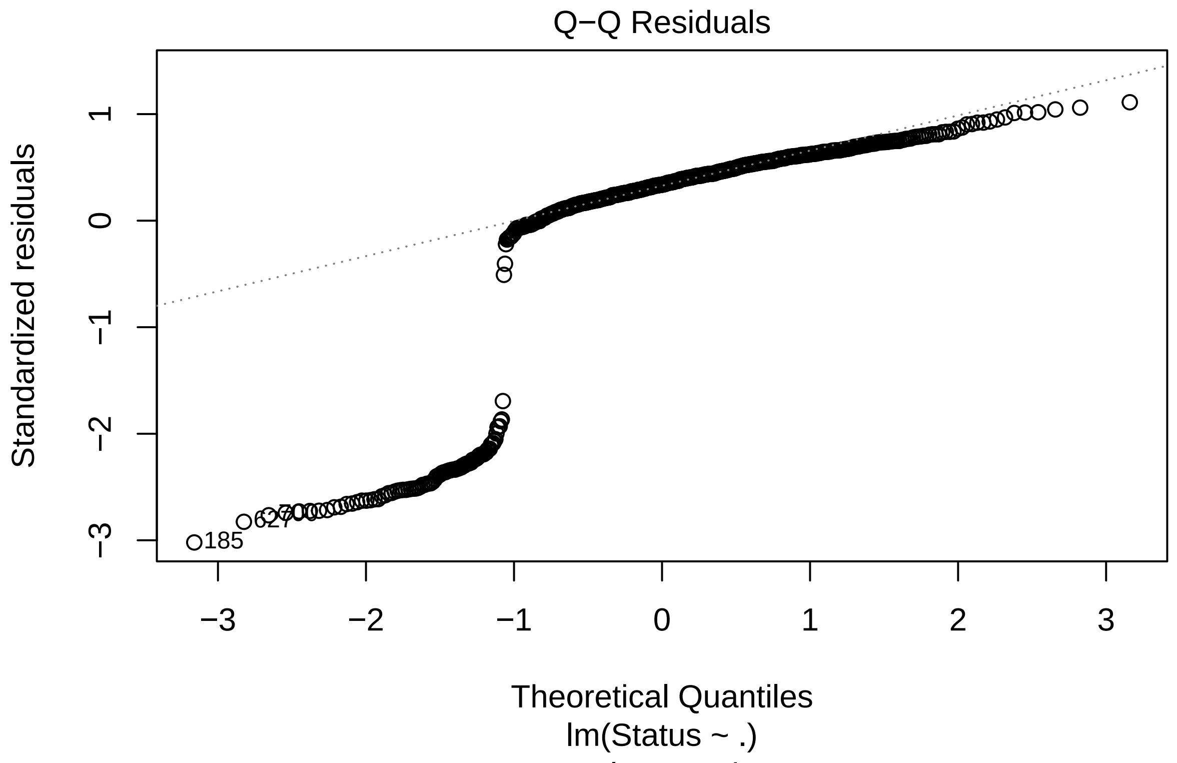
- The line "(1 not defined because of singularities)" indicates that one of the predictors is perfectly collinear with others or provides no additional information. This means the model could not assign a coefficient to it because of redundancy. In this case, 'Score' did not receive a coefficient due to perfect collinearity.
- 'IntRate' ans 'HomeOWN' are significant.
- Many other variables have high p-values, suggesting they do not contribute significantly to predicting Status. They are not helpful in explaining the variation in Status in this linear model.
- R-squared value of 6.27% indicates that only a small fraction of the variance in Status is explained by this model, which is quite low. The Adjusted R-squared of 4.77% further confirms that the model does not improve substantially even with the added predictors.

Conclusion: Some Predictors highly correlated, causing no coefficient in one estimates. Overall we can see this linear model is not suitable for this problem. Since Status is a binary variable, using linear regression (lm()) might not be ideal.

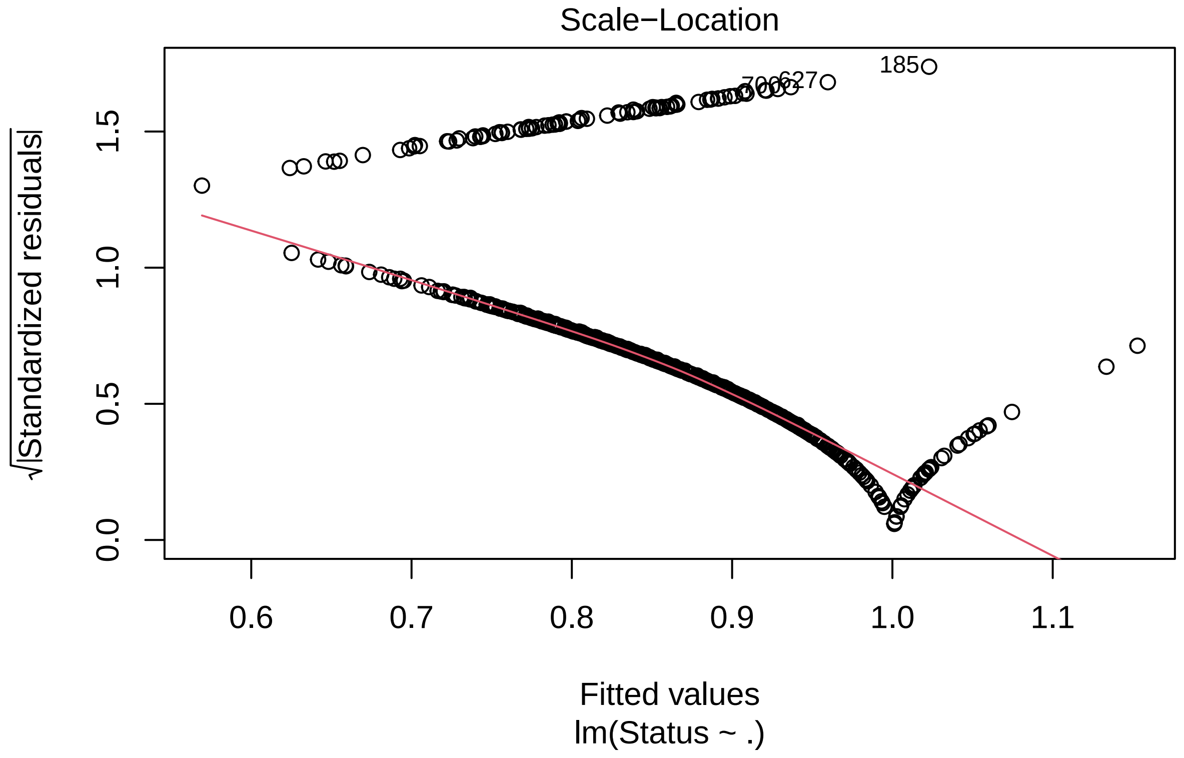
3)

```r
plot(lm_model)
```

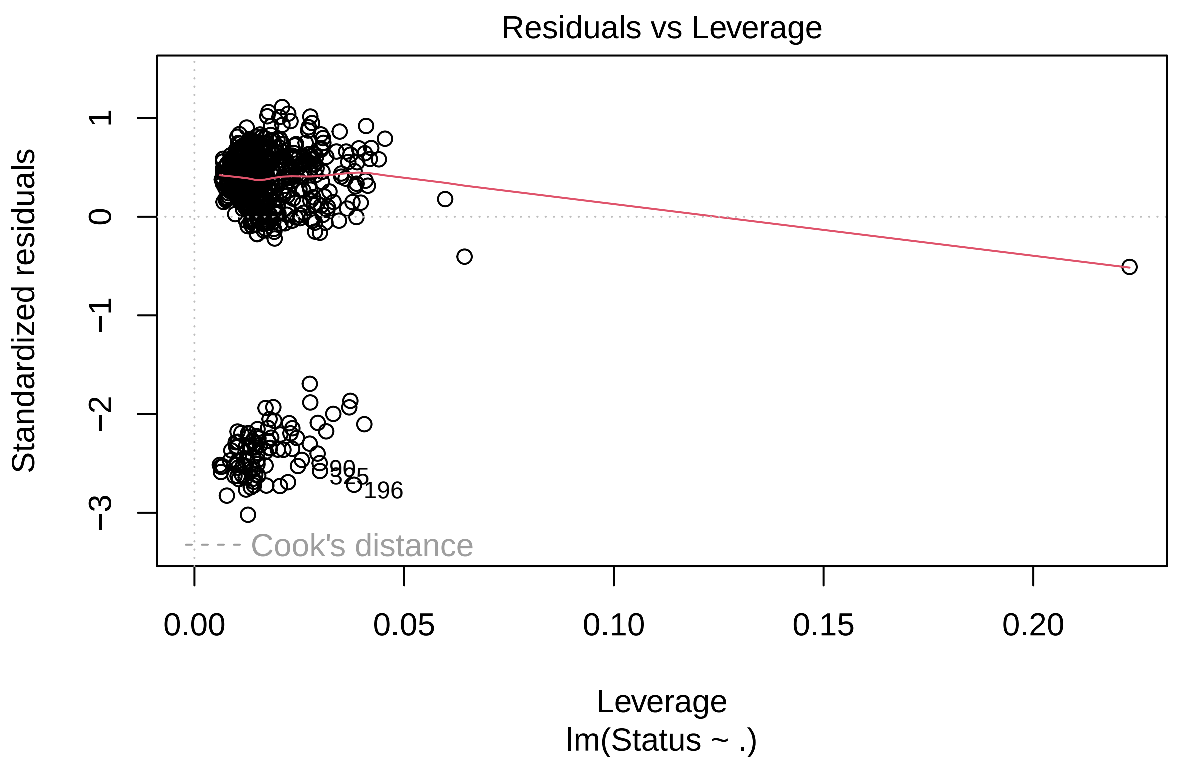




Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(Status ~ .)
185
627
70



Scale−Location
√|Standardized residuals|
Fitted values
lm(Status ~ .)
185
70
627



Shall we be worried?

These diagnostics overall indicate that the assumptions of linear regression are violated. Given that 'Status' is a binary variable, just shows that lm() is not ideal for binary prediction. This can lead to biased predictions, poor generalization, and unreliable statistics.

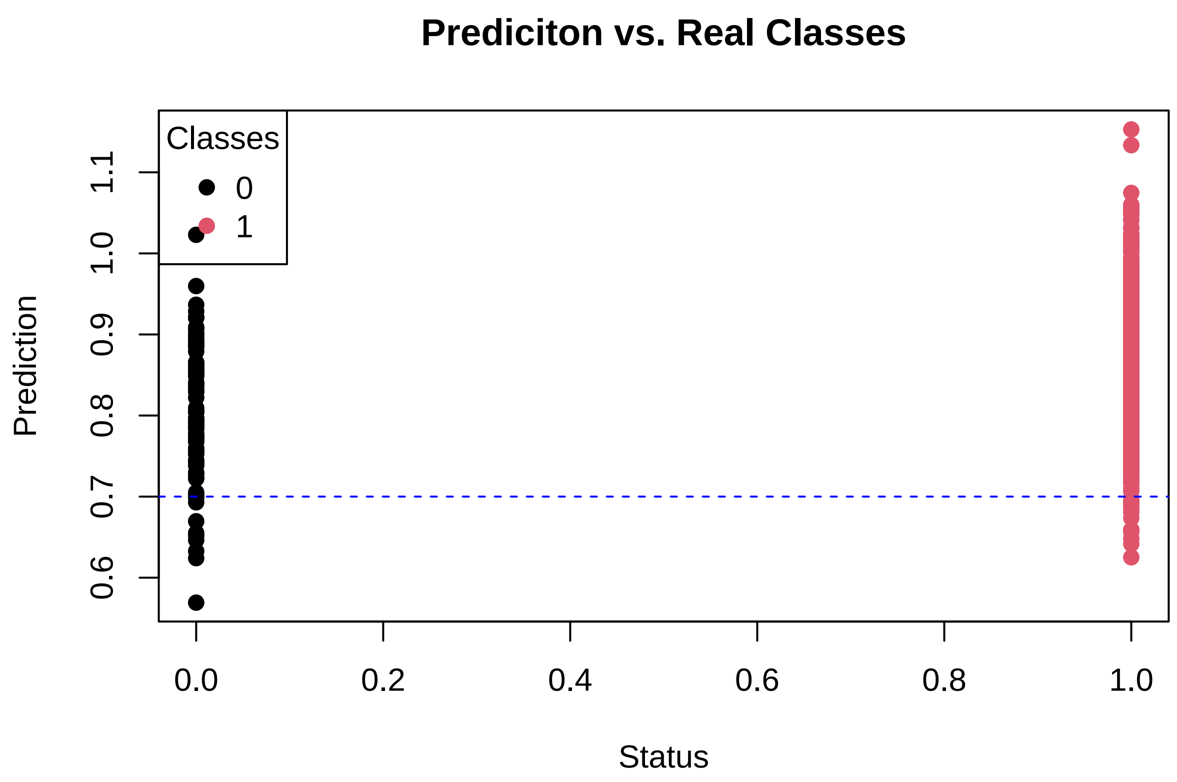
4)

```
train_predictions <- predict(lm_model, newdata = train_data)


plot(train_data$Status, train_predictions,
     col = as.factor(train_data$Status),
     pch = 19,
     xlab = "Status",
     ylab = "Prediction",
     main = "Prediciton vs. Real Classes")


abline(h = 0.7, col = "blue", lty = 2)

legend("topleft",
       legend = unique(train_data$Status),
       col = 1:length(unique(train_data$Status)),
       pch = 19,
       title = "Classes")
```



Which cutoff value would be useful in order to obtain reasonable class predictions?

To visualize the predicted values compared to the actual class labels, we create a scatter plot where the x-axis shows the true values and the y-axis shows the predicted ones. It's pretty clear that the model isn't great since it's predicting values between 0 and 1 (and even some above 1), which doesn't make sense for a binary variable.

We can see that class 0 generally has lower prediction values compared to class 1. Given this, we can consider a cutoff around 0,7. This is still not ideal! There is probably no good option here.

5)

```r
cutoff <- 0.7

predicted_classes <- ifelse(train_predictions > cutoff, 1, 0)

con_matrix = table(Actual = train_data$Status, Predicted = predicted_classes)
print(con_matrix)
```

```
##       Predicted
## Actual   0   1
##      0   9  81
##      1  13 531
```

Which conclusions can you draw from these numbers?

- True Positives (TP): 531 instances of class 1 were correctly predicted as 1.

- True Negatives (TN): 9 instances of class 0 were correctly predicted as 0.
- False Positives (FP): 81 instances of class 0 were incorrectly predicted as 1.
- False Negatives (FN): 13 instances of class 1 were incorrectly predicted as 0.

Based on these values, we can see that the model performs well at identifying the majority class 1, with high recall (97.6%) and decent precision (86.8%). However, the model performs poorly at identifying the minority class 0, with low recall (10%) and precision (10%).

The high number of False Positives (81) and the low number of True Negatives (9) indicate that the model struggles to correctly identify class 0.

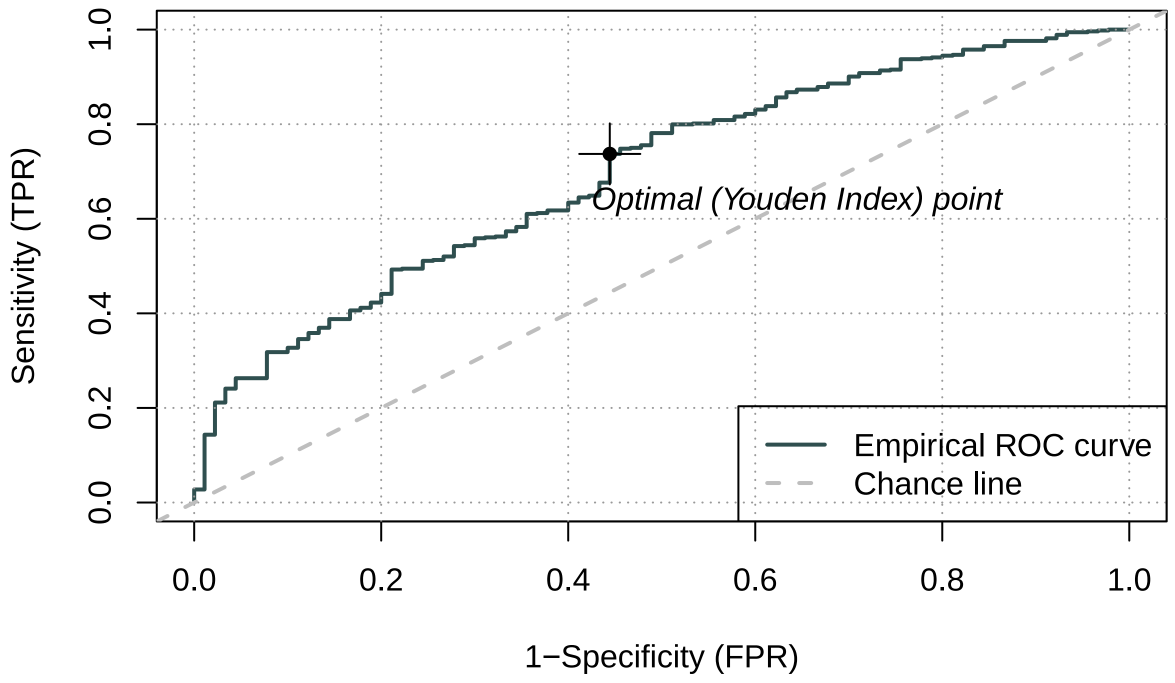
Overall, the model classifies most of the data as class 1. Since class 1 is heavily represented in the dataset, the model achieves higher accuracy for this class. However, as a whole, the model does not function effectively as a proper classifier, particularly in distinguishing the minority class 0.

6)

```r
roc <- rocit(train_predictions,train_data$Status)
summary(roc)
```

```
##
##  Method used: empirical
##  Number of positive(s): 544
##  Number of negative(s): 90
##  Area under curve: 0.6958
```

```r
plot(roc)
```



We can see in the summary, that there are 549 positive cases (class 1) and 90 negative cases (class 0). This confirms that the dataset is highly imbalanced, with many more positives than negatives. The AUC value is0.6958, which indicates the model has limited ability to distinguish between class 0 and class 1.

A good classifier would have an AUC of 1, while an AUC of 0.5 means the model is no better than random guessing, so our model performs slightly better than random but is far from ideal.

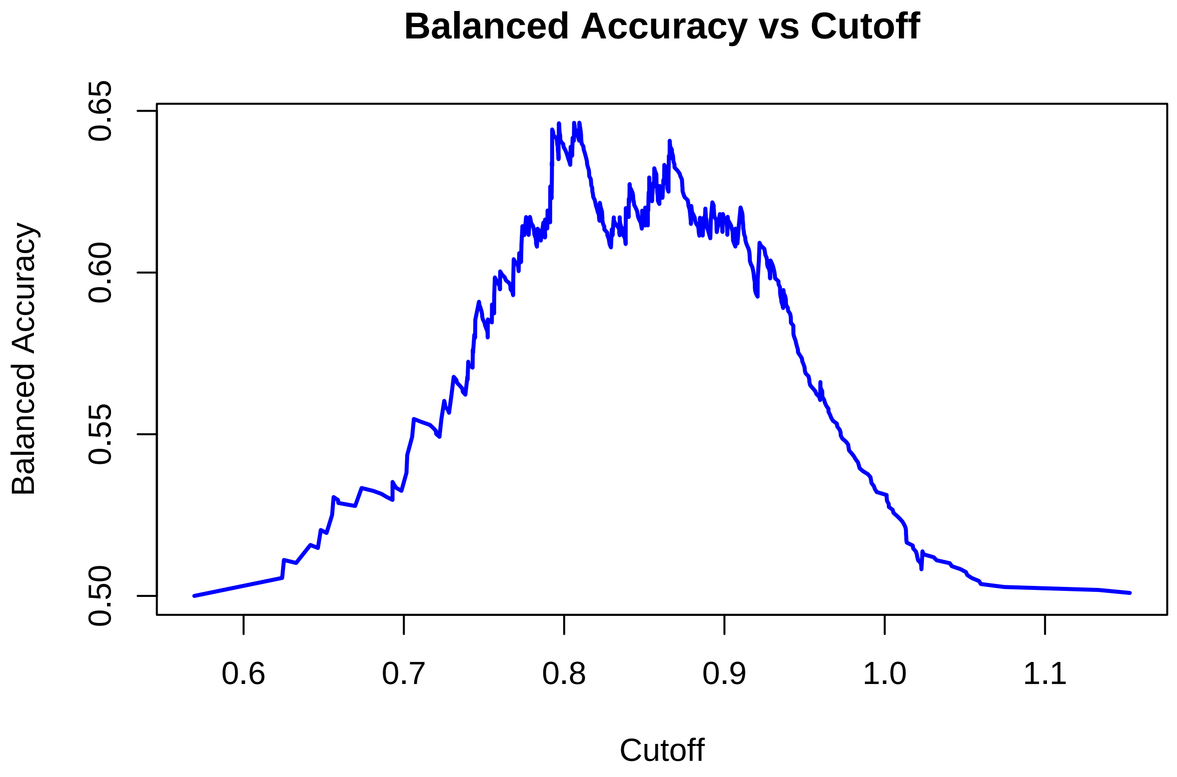
The ROC Curve:

The dotted diagonal line represents a model that performs no better than random guessing. So at least we can see our Model is better than rando guessing. The point marked as "Optimal (Youden Index) point" identifies the threshold where the model achieves the best balance between TPR and FPR. As you move along the curve, increasing TPR comes at the cost of increasing FPR. So our optimal point is around 0.75.

7)

```r
measure <- measureit(train_predictions,train_data$Statu,measure=c("TPR","TNR"))
measure$BalancedAccuracy <- (measure$TPR + measure$TNR) / 2

plot(measure$BalancedAccuracy ~ measure$Cutoff, type = "l", col = "blue", lwd = 2,
     xlab = "Cutoff", ylab = "Balanced Accuracy",
     main = "Balanced Accuracy vs Cutoff")
```



```r
# optimal cutoff with the highest Balanced Accuracy
optimal_cutoff <- measure$Cutoff[which.max(measure$BalancedAccuracy)]
cat("Optimal Cutoff (Highest Balanced Accuracy):", optimal_cutoff, "\n")
```

```
## Optimal Cutoff (Highest Balanced Accuracy): 0.8094848
```

8)

```r
test_predictions <- predict(lm_model, newdata = test_data)

test_predicted_classes <- ifelse(test_predictions > optimal_cutoff, 1, 0)

con_matrix = table(Actual = test_data$Status, Predicted = test_predicted_classes)
print(con_matrix)
```

```
##       Predicted
## Actual   0   1
##      0  14  27
##      1  58 167
```

What are your final conclusions?

On the test data and using the optimal cutoff, the model correctly classified 167 instances of class 1 (True Positives) and also correctly classified 14 instances of class 0 (True Negatives). However, it incorrectly classified 27 instances of class 0 as class 1 (False Positives) and 58 instances of class 1 as class 0 (False Negatives). The model struggles with imbalanced classes and has difficulty correctly identifying the minority class, which is class 0.

While it performs moderately well on identifying class 1, its performance for class 0 remains quite poor. The imbalance between the classes remains an issue, resulting in low performance in recognizing the minority class (class 0). Overall, there are many things we could try to improve the model, but it is clear that linear regression is not optimal for this classification problem. However, it actually performed better than initially expected, but still not good enough to be used effectively in practice.