Assignment 4

Loic Roldan Waals, 409763 & Nathan de Kruyf, 416677

April 16, 2018

Question 1

Part (a)

In total there are 12 emails, of which 9 are spam mails, and 3 non spam mails. The probability, derived from this data, of an email being spam is: 9 / 12 = 0.750, while the chance of not being spam is: 0.25.

Part (b)

Total number of spam emails is 9. Total number of emails which include the word "Viagra" and are marked as spam: 6. The probability of an email containing the word "viagra" given that it is spam is: 6 / 9 = 0.667.

Total number of spam emails is 9. Total number of emails which include "donation" and are marked as spam: 2. Probability of an email being spam when it contains the word "donation": 2 / 9 = 0.222 (the remainder is 0.778)

Part (c)

$$P(E = s \mid V, D) = \frac{P(V \mid E = s)\ P\ (D \mid E = s)P(E = s)}{\sum E = s, \bar{s}\ P(V|E)P(D|E)P(E)}$$

In our case we have to use the remainder of D.

In this case these are the values we fill into the formula:

- P(V|E=s) = 6 / 9
- P(D|E=s) = 7 / 9
- P(E=s) = 9 / 12
- P(V|E=ns) = 1 / 3
- P(D|E=ns) = 2 / 3
- P(E=ns) = 3 / 12

Probability according to the Naïve Bayes model that an email which contains the word "Viagra", but not the word "donation", is spam is 0.875 (see appendix for calculation).

Part (d)

We use the same formula as in **Part (c)**, but this time our values are:

- P(V|E=s) = 3 / 9
- P(D|E=s) = 2 / 9
- P(E=s) = 9 / 12

- P(V|E=ns) = 2 / 3
- P(D|E=ns) = 1 / 3
- P(E=ns) = 3 / 12

Probability according to the Naïve Bayes model that an email which contains the word "donation", but not the word "Viagra", is spam is 0.5 (see appendix for calculation).

Question 2

First we import the tweets from last weeks assignment. All the variables that are no used in this assignment are skipped. The code that is used is shown below.

```
#determine where files are read and written
setwd("C:/Users/Loic RW/Google Drive/Big Data and Business Analytics/Assignments/Assignment 4")


#read all the data
library(readr)
tweets <- read_delim("C:/Users/Loic RW/Google Drive/Big Data and Business Analytics/Assignments/Assignme
                     ";", escape_double = FALSE, col_types = cols(favorited = col_skip(),
                                                              favoriteCount = col_skip(),
                                                              isRetweet = col_skip(),
                                                              latitude = col_skip(),
                                                              longitude = col_skip(),
                                                              replyToSID = col_skip(),
                                                              replyToSN = col_skip(),
                                                              replyToUID = col_skip(),
                                                              retweeted = col_skip(),
                                                              statusSource = col_skip(),
                                                              truncated = col_skip()),
                     trim_ws = TRUE)
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
```

```
## Warning: 4 parsing failures.
## row # A tibble: 4 x 5 col     row col   expected   actual     file
```

```
tweets$retweetCount <- as.numeric(tweets$retweetCount)
```

```
## Warning: NAs introduced by coercion
```

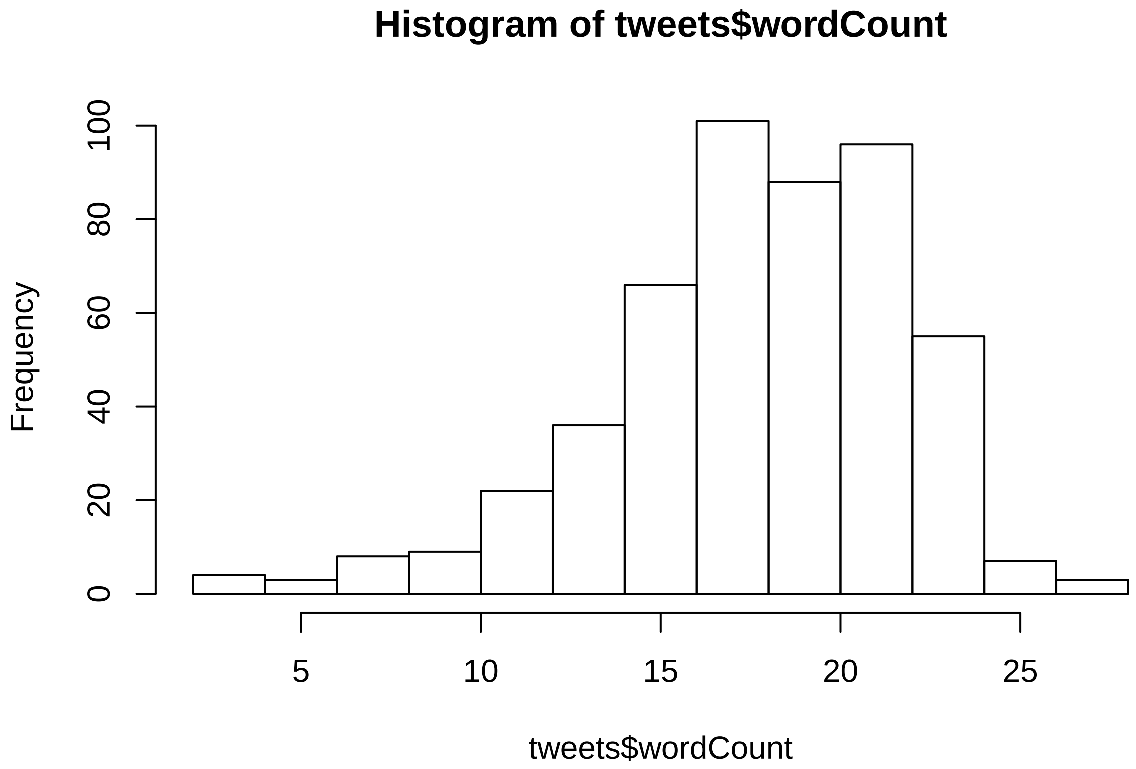
```
#delete 4 NA's for "retweetcount"
tweets <- tweets[complete.cases(tweets),]
```

Next we need to count how many words each tweet has, this number is then added to the data frame. After the word count for each of the tweets is completed we make a histogram. We then also make a scatter plot with *wordCount* and *retweetCount*. The code that is used is shown below.

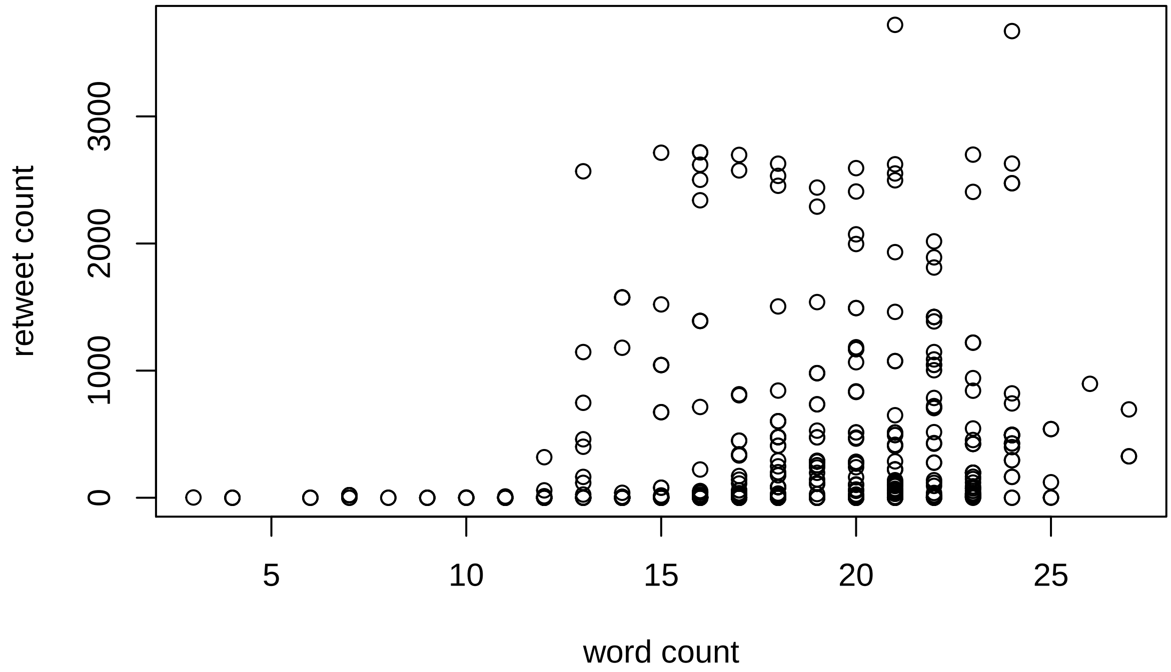
```
library(ngram)

#count the words per tweet
for (i in 1:nrow(tweets)) {
  tweets$wordCount[i] <- as.numeric(wordcount(tweets$text[i], sep = " ", count.function = sum))
}
```

```
## Warning: Unknown or uninitialised column: 'wordCount'.
```

```
#make histogram for the wordcount
hist(tweets$wordCount)
```



```
#make scatterplot
plot(tweets$wordCount, tweets$retweetCount, xlab = "word count", ylab = "retweet count")
```



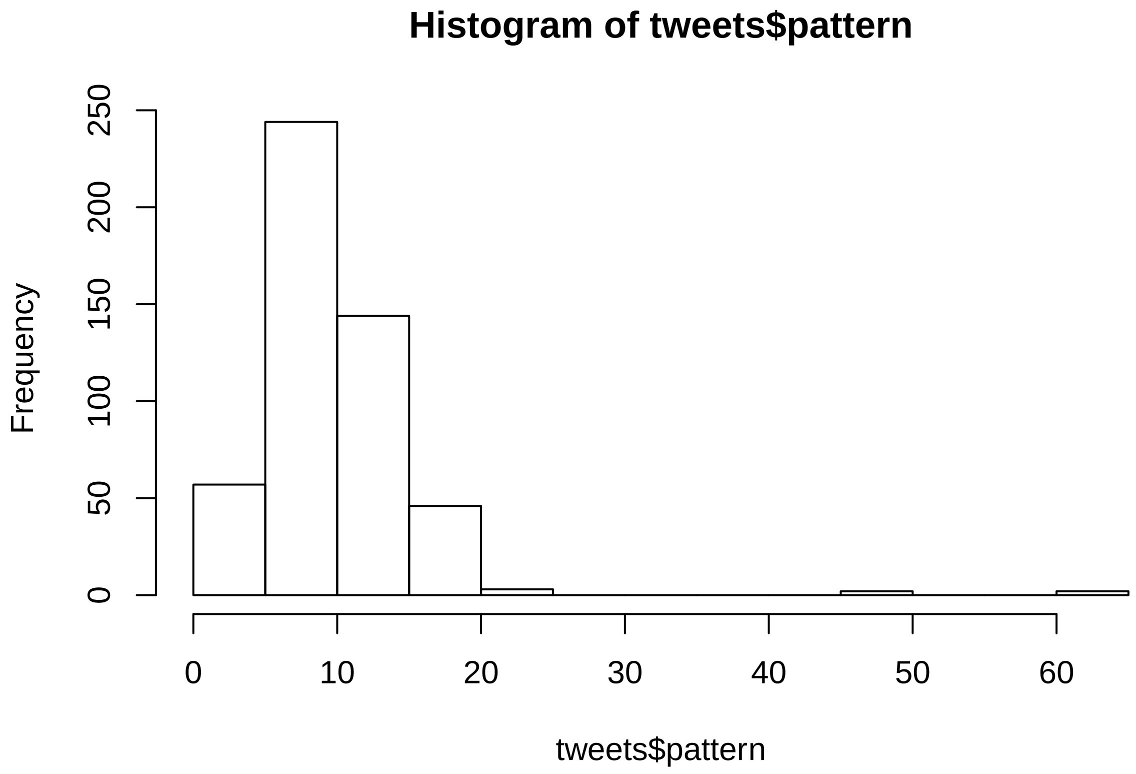
Next we calculate the frequency of the pattern “dam” per tweet. As we can see there are only two instances where the pattern “dam” is used, furthermore we also have all of our tweets within 5 minutes of each other, so the graph for average frequency of “dam” will simply be the average frequency of “dam”.

As this would not make for an exciting graph, we redo this question with the pattern “e” and plot it against average minute frequency. Here we see a more interesting result. Although it appears that there is a positive trend, 5 data points is simply not enough to make a proper inference. Additionally it makes more sense to plot the average minute frequency against the actual minutes (as average minute frequency only changes when the minute changes). The code that is used is shown below.

```
library(stringr)
library(plyr)

#count the instances of the pattern "dam"
tweets$pattern <- str_count(tweets$text, "dam")
sum(tweets$pattern)
```

```
## [1] 2
```

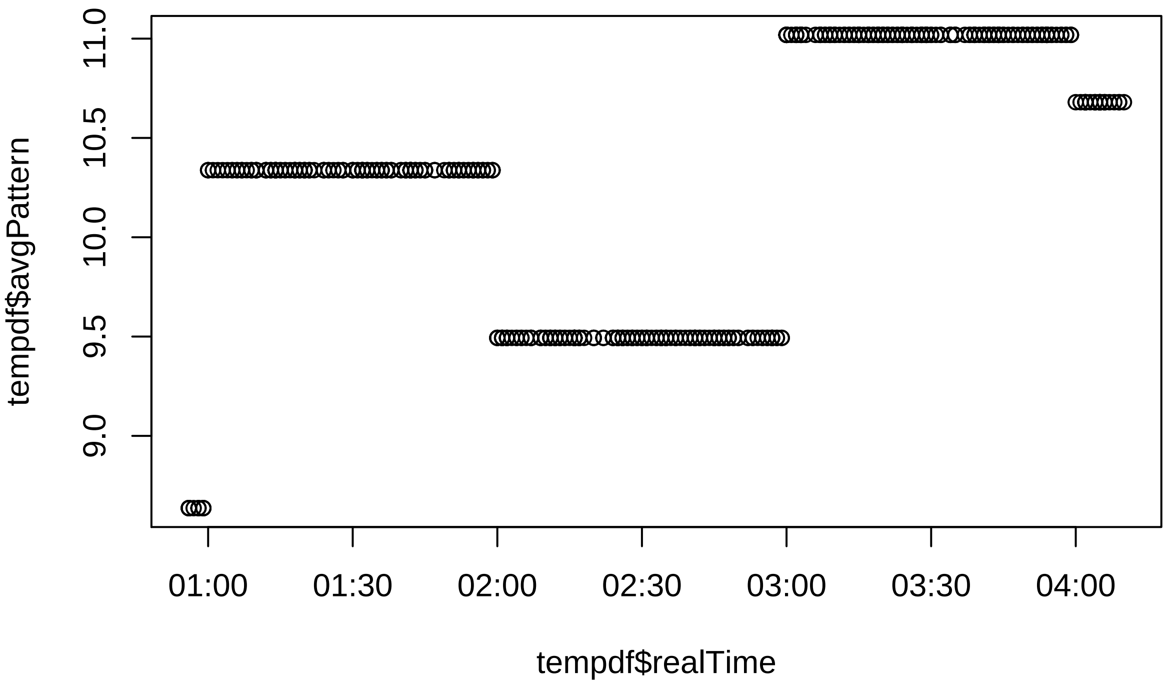
```
#count the instances of the pattern "e"
tweets$pattern <- str_count(tweets$text, "e")
hist(tweets$pattern)
```



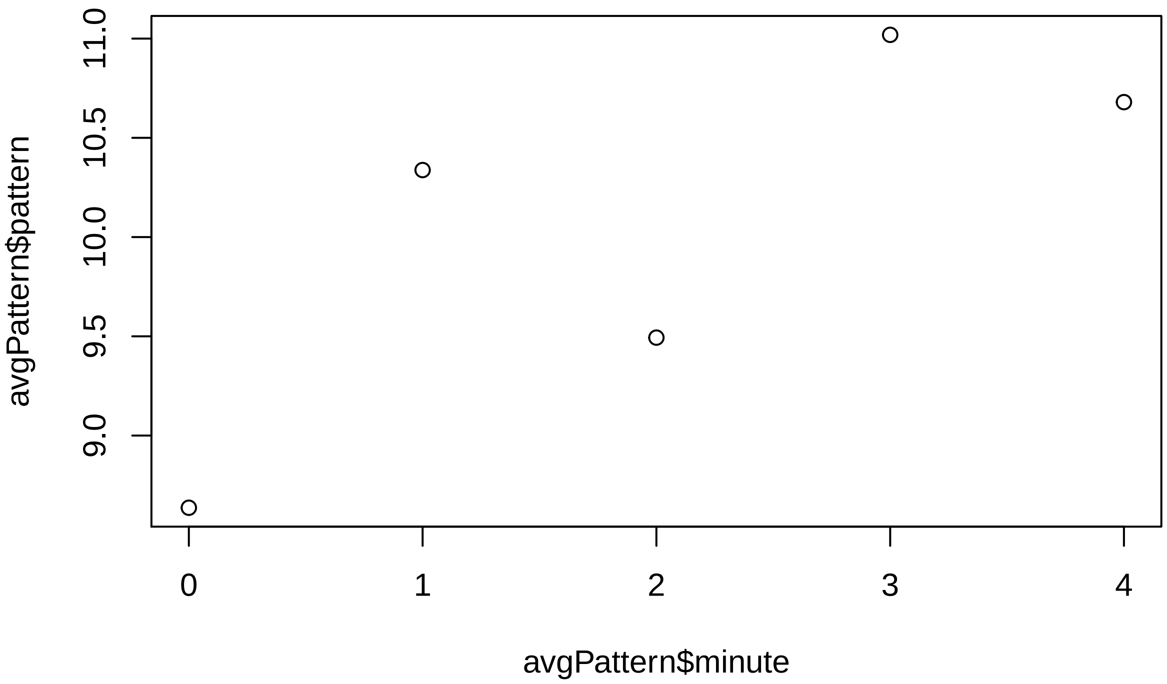
```
#make a scatterplot
#get the minutes
tweets$realTime <- as.POSIXct(tweets$created, format = "%y-%m-%d %H:%M:%S")
tweets$minute <- as.POSIXlt(tweets$realTime)$min

#get the average per minute
tempdf <- tweets[,c("pattern", "minute", "realTime")]
avgPattern <- aggregate(cbind(pattern)~minute, data = tempdf, FUN = mean)

#assign average per minute to temp dataframe
for (i in 1:nrow(tempdf)){
  for (j in 1:nrow(avgPattern))
  if (tempdf$minute[i] == avgPattern$minute[j]){
    tempdf$avgPattern[i] <- avgPattern$pattern[j]
  }
}
```

```
## Warning: Unknown or uninitialised column: 'avgPattern'.
```

```
#plot the average frequency of the pattern per minute over the time
plot(tempdf$avgPattern~tempdf$realTime)
```



```
#plot the average frequenty over the minutes
plot(avgPattern$pattern~avgPattern$minute)
```



Question 3

First we import the data and clean it. The code that is used is shown below.

```
library(readr)

#import the date and skip irrelevant features
trainTitanic <- read_csv("C:/Users/Loic RW/Google Drive/Big Data and Business Analytics/Workshops/Sessi
                         col_types = cols(Name = col_skip(), PassengerId = col_skip()))

#fix variables
trainTitanic$Survived <- as.ordered(trainTitanic$Survived)
trainTitanic$Pclass <- as.factor(trainTitanic$Pclass)
trainTitanic$Sex <- as.factor(trainTitanic$Sex)
trainTitanic$Embarked <- as.factor(trainTitanic$Embarked)

trainTitanic$Age[is.na(trainTitanic$Age)] <- mean(trainTitanic$Age, na.rm = TRUE)
trainTitanic$Cabin <- NULL
trainTitanic$Ticket <- NULL

#delete any other missing cases
trainTitanic <- trainTitanic[complete.cases(trainTitanic),]

#scale remaining integer variables
```

```
trainTitanic$Age <- scale(trainTitanic$Age)
trainTitanic$SibSp <- scale(trainTitanic$SibSp)
trainTitanic$Parch <- scale(trainTitanic$Parch)
trainTitanic$Fare <- scale(trainTitanic$Fare)
```

Now we perform the analysis. We train both a naive Bayesian (NB) classifier and a Support Vector Machine SVM. 10 fold cross validation is used for calculating the performance measures. We will measure accuracy, specificity, sensitivity and Area Under the Curve (AUC). The code that is used is shown below.

```
library(C50)
library(e1071)

mdl <- Survived ~ .

#Xval
#cross validation
nFolds = 10
myFolds <- cut(seq(1,nrow(trainTitanic)),
               breaks = nFolds,
               labels = FALSE)

#initialise accuracy variables
accNB <- rep(NA, nFolds)
accSVM <- rep(NA, nFolds)

specNB <- rep(NA, nFolds)
specSVM <- rep(NA, nFolds)

sensNB <- rep(NA, nFolds)
sensSVM <- rep(NA, nFolds)

ROCData <- data.frame(actual=numeric(), NB=numeric(), SVM=numeric())

actual <-NULL
ROCNB <- NULL
ROCSVM <- NULL

library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(pROC)
```

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

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(reshape2)

for (i in 1:nFolds) {
  cat("Analysis of fold", i, " complete ", "\n")

  #define training and test set
  testIndex <- which(myFolds == i, arr.ind = TRUE)
  crossTest <- trainTitanic[testIndex, ]
  crossTrain <- trainTitanic[-testIndex, ]

  #train the models
  rsltNB <- naiveBayes(mdl, data = crossTrain)
  rsltSVM <- svm(mdl, data = crossTrain)

  #predict values
  pdNB <- as.ordered(predict(rsltNB, crossTest, type = "class"))
  pdSVM = as.ordered(predict(rsltSVM, crossTest))

  #measure accuracy
  accNB[i] = mean(pdNB == crossTest$Survived)
  accSVM[i] = mean(pdSVM == crossTest$Survived)

  confMatrix <- table(pdNB, crossTest$Survived)
  d = confMatrix[1,1]
  c = confMatrix[1,2]
  b = confMatrix[2,1]
  a = confMatrix[2,2]
  specNB[i] = d/(d+b)
  sensNB[i] = a/(a+c)

  confMatrix <- table(pdSVM, crossTest$Survived)
  d = confMatrix[1,1]
  c = confMatrix[1,2]
  b = confMatrix[2,1]
  a = confMatrix[2,2]
  specSVM[i] = d/(d+b)
  sensSVM[i] = a/(a+c)

  #AUC
  crossTrain$Survived <- as.numeric(crossTrain$Survived)
  crossTest$Survived <- as.numeric(crossTest$Survived)

  rsltNB <- naiveBayes(mdl, data = crossTrain)
  rsltSVM <- svm(mdl, data = crossTrain)

  pdNB <- predict(rsltNB, crossTest, type = "raw")
  pdNB <- pdNB[,2]
  pdSVM = predict(rsltSVM, crossTest)

  #keep track of the values for the ROC curve
  actual <- c(actual, crossTest$Survived)
  ROCNB <- c(ROCNB, pdNB)
  ROCSVM <- c(ROCSVM, pdSVM)
}
```

```
## Analysis of fold 1  complete
## Analysis of fold 2  complete
## Analysis of fold 3  complete
## Analysis of fold 4  complete
## Analysis of fold 5  complete
## Analysis of fold 6  complete
## Analysis of fold 7  complete
## Analysis of fold 8  complete
## Analysis of fold 9  complete
## Analysis of fold 10  complete
```

```
#determine the average accuracy over the 10 folds for each model
avgAccNB = mean(accNB)
avgAccSVM = mean(accSVM)

avgSpecNB = mean(specNB)
avgSpecSVM = mean(specSVM)

avgSensNB = mean(sensNB)
avgSensSVM = mean(sensSVM)


#print all the results
avgAccNB
```

```
## [1] 0.7761747
```

```
avgAccSVM
```

```
## [1] 0.8245148
```

```
avgSpecNB
```

```
## [1] 0.8963486
```

```
avgSpecSVM
```

```
## [1] 0.8898537
```

```
avgSensNB
```

```
## [1] 0.5849951
```

```
avgSensSVM
```

```
## [1] 0.7150887
```

```r
#compare with classifying everything as did not survive
lazyPredictor <- rep(0, nrow(trainTitanic))

mean(trainTitanic$Survived == lazyPredictor)
```

```
## [1] 0.6175478
```

```r
auc(trainTitanic$Survived, lazyPredictor)
```

```
## Area under the curve: 0.5
```

```r
#ROC and AUC calculation
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```r
ROCData <- data.frame(actual, ROCNB, ROCSVM)
roc.curve(ROCData$actual, ROCData$ROCNB, col = 2)
```

```
## Area under the curve (AUC): 0.825
```

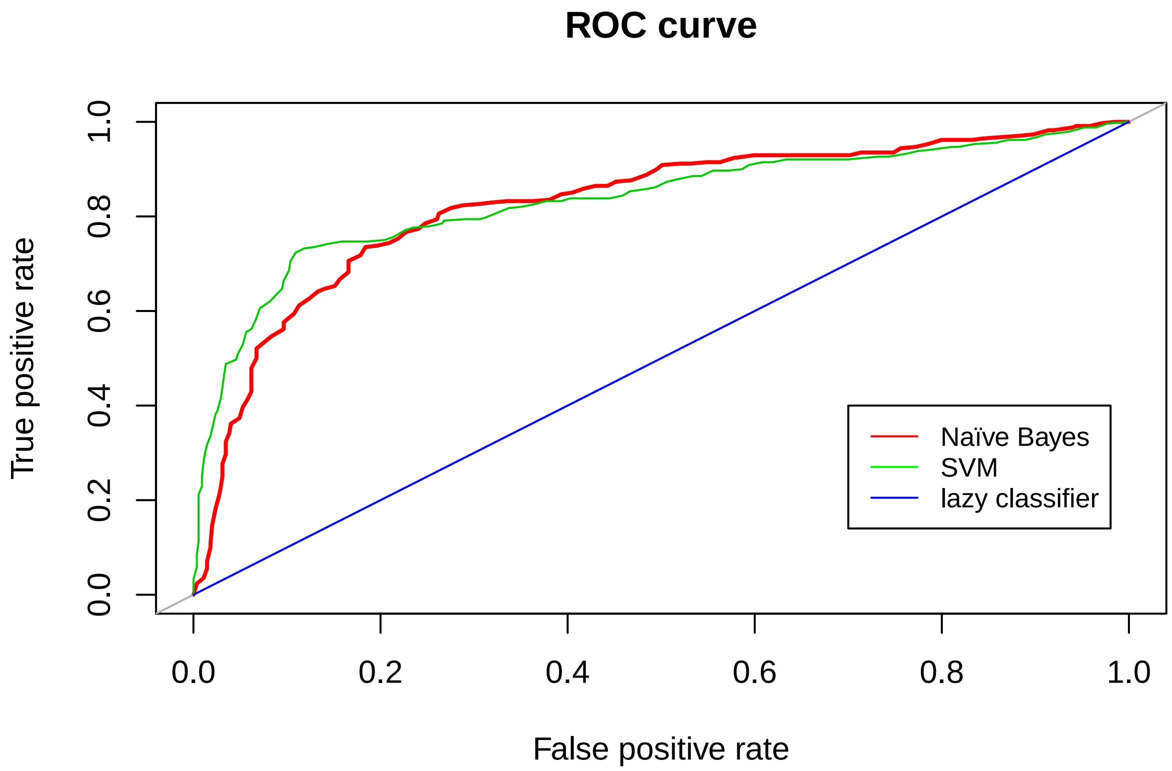
```r
roc.curve(ROCData$actual, ROCData$ROCSVM, add.roc = TRUE, col = 3)
```

```
## Area under the curve (AUC): 0.834
```

```r
roc.curve(trainTitanic$Survived, lazyPredictor, add.roc = TRUE,
          col = 4)
```

```
## Area under the curve (AUC): 0.500
```

```r
legend(0.7, 0.4, legend=c("Naïve Bayes", "SVM", "lazy classifier"),
       col=c("red", "green", "blue"), lty=1, cex=0.8)
```



```
AUCNB = as.numeric(auc(ROCData$actual, ROCData$ROCNB))
AUCSVM = as.numeric(auc(ROCData$actual, ROCData$ROCSVM))
AUCLazy =as.numeric(auc(trainTitanic$Survived, lazyPredictor))
```

As we can see both the accuracy and AUC of the SVM are higher than for NB, this indicates a higher performance as a classifier. It is noted that by guessing everything as *did not survive*, or 0, one would get an accuracy of 61.8%, this indicates a slight skew; the AUC is 0.5. Thus the SVM is the better classifier, but not by much.

From the specificity and sensitivity analysis we can conclude that of the people that did not survive, the SVM classified 89.0% correctly, while the NB classified 89.6% correctly. This difference is quite small. Of the people that did survive, the NB classified 58.5% correctly while the SVM classified 71.5% correctly, this difference is quite large.

It appears that due to the SVM's superior ability to determine who did survive, it achieves a higher performance.

Now we will examine the ROC. NB is represented with red, SVM with green and the blue line represents the case of predicting *did not survive* everywhere. The SVM performs better than the NB with an AUC of 0.834 compared to 0.825. The difference in performance can be seen in the graph as the green line is either above or at about the same level as the red line. SVM excels against the NB at the threshold intervals 0.0-0.2, everywhere else it is about the same.

$$0.875 = \frac{\frac{6}{9} * \frac{7}{9} * \frac{9}{12}}{\frac{6}{9} * \frac{7}{9} * \frac{9}{12} + \frac{1}{3} * \frac{2}{3} * \frac{3}{12}}$$

Figure 1: Calculation part (c)

$$0.5 = \frac{\frac{3}{9} * \frac{2}{9} * \frac{9}{12}}{\frac{3}{9} * \frac{2}{9} * \frac{9}{12} + \frac{2}{3} * \frac{1}{3} * \frac{3}{12}}$$

Figure 2: Calculation part (d)

Appendix