Homework 5

Group 5: Piyusha Kulkarni, Carolina Munoz, Vikrant Nakod, Hareesh Rajendran

04/06/2020

```
if(!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(tidyverse, reshape, gplots, ggmap,
               mlbench, data.table, gridExtra,GGally,grid,caret,e1071, fpp2, gains, pROC, caret, data.ta
```

```
## Installing package into 'C:/Users/Home Laptop/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)

## Warning: dependency 'caTools' is not available

## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contri
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'gplots' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Home Laptop\AppData\Local\Temp\RtmpqgnALw\downloaded_packages

##
## gplots installed

## Warning in pacman::p_load(tidyverse, reshape, gplots, ggmap, mlbench, data.table, : Failed to instal
## gplots
```

```
search()
```

```
##  [1] ".GlobalEnv"         "package:MASS"       "package:pROC"
##  [4] "package:gains"      "package:fpp2"       "package:expsmooth"
##  [7] "package:fma"        "package:forecast"   "package:e1071"
## [10] "package:caret"      "package:lattice"    "package:grid"
## [13] "package:GGally"     "package:gridExtra"  "package:data.table"
## [16] "package:mlbench"    "package:ggmap"      "package:reshape"
## [19] "package:forcats"    "package:stringr"    "package:dplyr"
## [22] "package:purrr"      "package:readr"      "package:tidyr"
## [25] "package:tibble"     "package:ggplot2"    "package:tidyverse"
## [28] "package:pacman"     "package:stats"      "package:graphics"
## [31] "package:grDevices"  "package:utils"      "package:datasets"
## [34] "package:methods"    "Autoloads"          "package:base"
```

```r
theme_set(theme_classic())
```

```r
spammail.df <- read.csv("spambase.data", header = FALSE)

colnames(spammail.df)<- c("word_freq_make","word_freq_address","word_freq_all","word_freq_3d","word_fre


str(spammail.df)
```

```
## 'data.frame':    4601 obs. of  58 variables:
##  $ word_freq_make            : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ word_freq_address         : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ word_freq_all             : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ word_freq_3d              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_our             : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
##  $ word_freq_over            : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ word_freq_remove          : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ word_freq_internet        : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
##  $ word_freq_order           : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
##  $ word_freq_mail            : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ word_freq_receive         : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ word_freq_will            : num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ word_freq_people          : num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
##  $ word_freq_report          : num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ word_freq_addresses       : num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ word_freq_free            : num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ word_freq_business        : num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ word_freq_email           : num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ word_freq_you             : num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ word_freq_credit          : num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ word_freq_your            : num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ word_freq_font            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_000             : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ word_freq_money           : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ word_freq_hp              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_hpl             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_george          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_650             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_lab             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_labs            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_telnet          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_857             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_data            : num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ word_freq_415             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_85              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_technology      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_1999            : num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ word_freq_parts           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_pm              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_direct          : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word_freq_cs              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_meeting         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_original        : num  0 0 0.12 0 0 0 0 0 0.3 0 ...
```

```
##  $ word_freq_project         : num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ word_freq_re              : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word_freq_edu             : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word_freq_table           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word_freq_conference      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ char_freq_;               : num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ char_freq_(               : num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206 0.271 0.03 ...
##  $ char_freq_[               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ char_freq_!               : num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ char_freq_$               : num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ char_freq_#               : num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ capital_run_length_average: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ capital_run_length_longest: int  61 101 485 40 40 15 4 11 445 43 ...
##  $ capital_run_length_total  : int  278 1028 2259 191 191 54 112 49 1257 749 ...
##  $ s_ns_class                : int  1 1 1 1 1 1 1 1 1 1 ...
```

Question 1: Examine how each predictor differs between the spam and non-spam e-mails by comparing the spam-class average and non-spam-class average. Identify 10 predictors for which the difference between the spam-class average and non-spam class average is highest.

```
spammail.df$s_ns_class <- factor(spammail.df$s_ns_class)
spam.df <- spammail.df[spammail.df$s_ns_class == 1, ]
nospam.df <- spammail.df[spammail.df$s_ns_class == 0, ]
#str(spam.df)
#str(non_spam.df)
aggr_mean <- aggregate(spammail.df[,1:57],by=list(spammail.df$s_ns_class),FUN= mean)
aggr_mean <- aggr_mean[,-1]
#str(aggr_mean)
sorted <- abs(aggr_mean[2,]-aggr_mean[1,])
sorted <- sorted[,-1]
head(t(sort(sorted, decreasing = TRUE)),10)
```

```
##                                    2
## capital_run_length_total   309.1484684
## capital_run_length_longest  86.1787801
## capital_run_length_average   7.1418640
## word_freq_george             1.2637155
## word_freq_you                0.9941987
## word_freq_your               0.9416680
## word_freq_hp                 0.8779941
## word_freq_free               0.4447750
## word_freq_hpl                0.4228216
## char_freq_!                  0.4037291
```

Answer 1: The top 10 predictors for which the difference is highest are- capital_run_length_total
capital_run_length_longest
capital_run_length_average

word_freq_george
word_freq_you
word_freq_your
word_freq_hp
word_freq_free
word_freq_hpl
char_freq_!

```
model.df <- spammail.df[,c(57,56,55,27,19,21,25,16,26,52,58)]
levels(model.df$s_ns_class) <- c("non-spam","spam")

#Spilt data
model <- model.df
set.seed(42)

training.index <- createDataPartition(model.df$s_ns_class, p = 0.8, list = FALSE)
model.train <- model[training.index, ]
model.valid <- model[-training.index, ]




# Normalize the data

norm.values  <- preProcess(model.train, method = c("center", "scale"))
model.train.norm <- predict(norm.values, model.train)
model.valid.norm <- predict(norm.values, model.valid)
```

Question 2: Perform a linear discriminant analysis using the training dataset. Include only 10 predictors identified in the question above in the model.

```
lda <- lda(s_ns_class~., data = model.train.norm)
lda
```

```
## Call:
## lda(s_ns_class ~ ., data = model.train.norm)
##
## Prior probabilities of groups:
##  non-spam      spam
## 0.6059207 0.3940793
##
## Group means:
##          capital_run_length_total capital_run_length_longest
## non-spam               -0.2091781                 -0.1654593
## spam                    0.3216240                  0.2544037
##          capital_run_length_average word_freq_george word_freq_you
## non-spam                -0.09069094        0.1461452    -0.2253719
## spam                     0.13944279       -0.2247070     0.3465229
```

```
##          word_freq_your word_freq_hp word_freq_free word_freq_hpl `char_freq_!`
## non-spam     -0.3140727    0.2091631     -0.1978717     0.1946780    -0.2334628
## spam          0.4829058   -0.3216009      0.3042397    -0.2993291     0.3589632
##
## Coefficients of linear discriminants:
##                                     LD1
## capital_run_length_total    0.40238920
## capital_run_length_longest  0.07649678
## capital_run_length_average  0.06305829
## word_freq_george           -0.21008128
## word_freq_you               0.21917369
## word_freq_your              0.55596353
## word_freq_hp               -0.21864368
## word_freq_free              0.37122977
## word_freq_hpl              -0.17369664
## `char_freq_!`               0.37851762
```

Question 3: What are the prior probabilities?

Answer 3: Prior probabilities of groups: non-spam spam 0.6059207 0.3940793

Question 4: What are the coefficients of linear discriminants? Explain.

Answer 4: The coefficients of linear discriminant are as follows-

Coefficients of linear discriminants: LD1 capital_run_length_total 0.40238920 capital_run_length_longest 0.07649678 capital_run_length_average 0.06305829 word_freq_george -0.21008128 word_freq_you 0.21917369 word_freq_your 0.55596353 word_freq_hp -0.21864368 word_freq_free 0.37122977 word_freq_hpl -0.17369664 `char_freq_!` 0.37851762

The coefficients of linear discriminants provide the equation for the discriminant functions that is if we multiply each value of LD1 by the corresponding elements of the predictor variable and sum them you get a score for each respondent.Here, as we have two classes spam and non spam only one LD function is generated.

Question 5: Generate linear discriminants using your analysis.How are they used in classifying spams and non-spams?

```
#Question 5

pred2.train <- predict(lda, model.train.norm)
head(pred2.train$posterior,n=20)
```

```
##       non-spam      spam
## 3  0.179066960 0.8209330
## 5  0.656378811 0.3436212
## 6  0.875516880 0.1244831
## 7  0.485083508 0.5149165
```

```
## 8  0.876512591 0.1234874
## 9  0.203656589 0.7963434
## 10 0.546758136 0.4532419
## 11 0.547287480 0.4527125
## 12 0.620003730 0.3799963
## 13 0.419611109 0.5803889
## 14 0.697402030 0.3025980
## 15 0.009249456 0.9907505
## 17 0.301242042 0.6987580
## 18 0.141328458 0.8586715
## 19 0.575791058 0.4242089
## 20 0.385746340 0.6142537
## 22 0.110052379 0.8899476
## 23 0.338317455 0.6616825
## 25 0.828247130 0.1717529
## 26 0.110155496 0.8898445
```

```
head(pred2.train$x,n=20)
```

```
##            LD1
## 3   1.44648249
## 5   0.01804783
## 6  -0.84001235
## 7   0.48338906
## 8  -0.84604736
## 9   1.34175571
## 10  0.32061465
## 11  0.31920834
## 12  0.12181859
## 13  0.65763241
## 14 -0.10555398
## 15  3.52093208
## 17  0.99798753
## 18  1.63187202
## 19  0.24297586
## 20  0.75035937
## 22  1.82008454
## 23  0.88568850
## 25 -0.59157646
## 26  1.81939173
```

Answer 5: The linear discriminant shown above is the optimal linear combination of predictors that maximizes the separation between classes and minimizes the variation within a class. A discriminant function is used to calculate a record's stastical distance from each class.Then, the record's probability of belonging to each class is calculated. From there, the probability values are compared to the chosen cut-off point and assigned to a class. In this case, Linear discrimninants are used to clasify a observation weather it is spam or non spam. Example: in the 3rd observation the posterior probabilty for observation being non-spam is 0.179 and for spam is 0.821. Therefore, LDA will assign class as spam when using cut-off of 0.5.

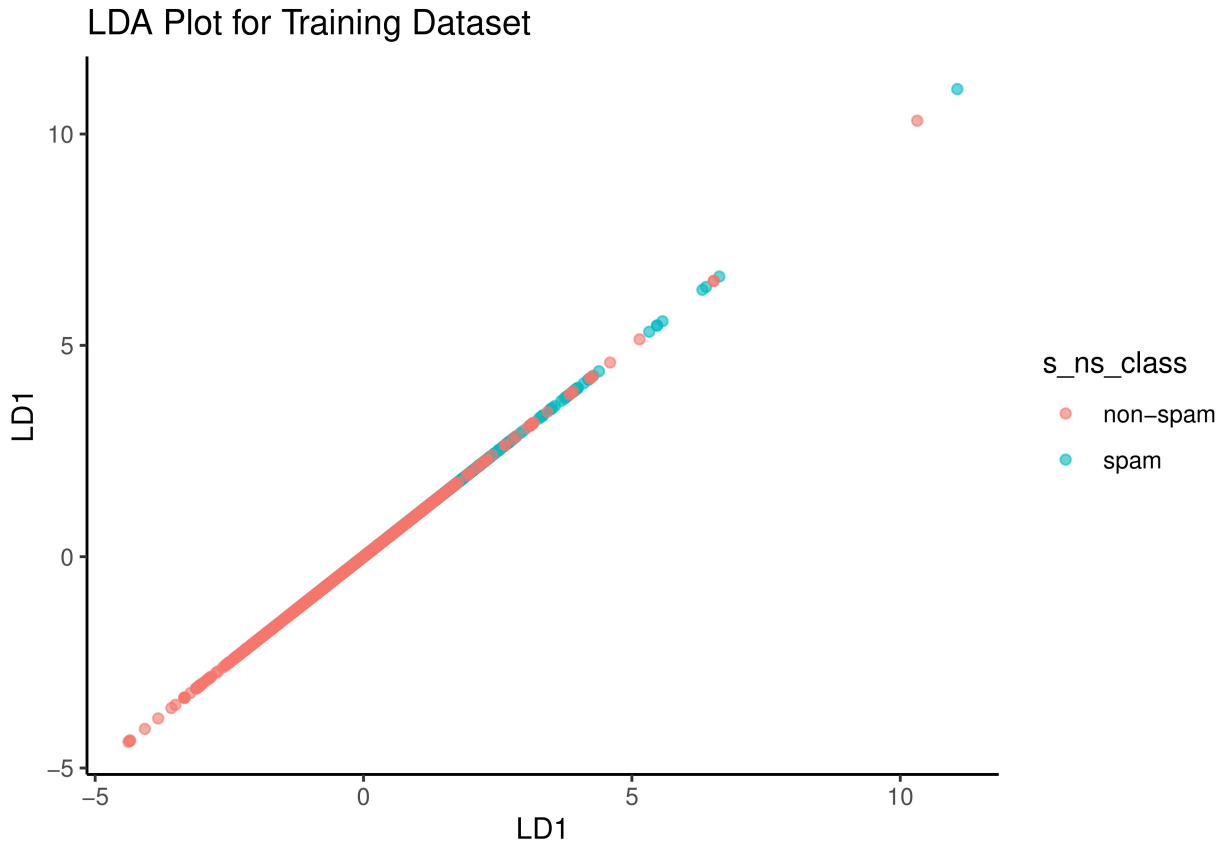
Question 6: How many linear discriminants are in the model? Why

Answer 6:Number of linear discrimants are always one less than number of classes (n-1). In the above scenario we only have two classes and those are Spam and Non-Spam. Therefore, there is only 1 Linear

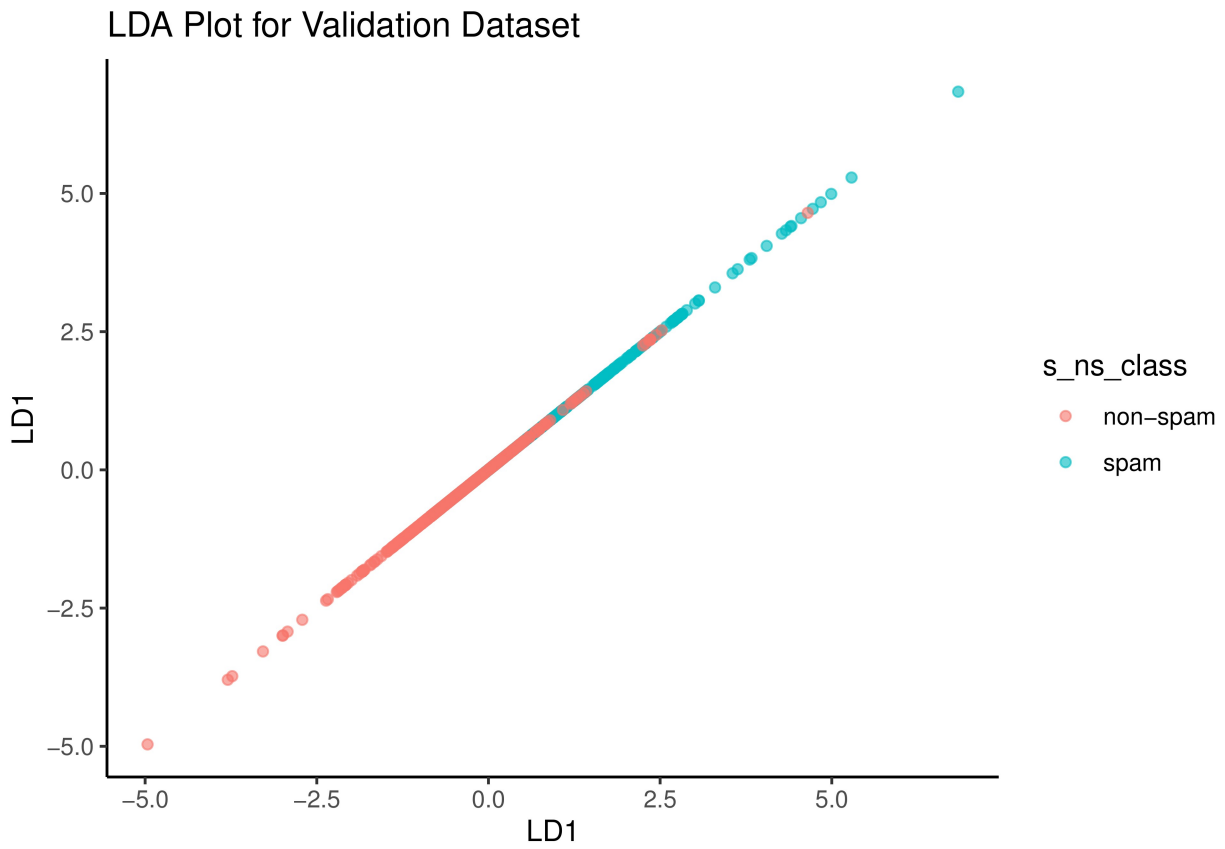
discriminant in this model

Question 7: Generate LDA plot using the training and validation data. What information is presented in these plots? How are they different?

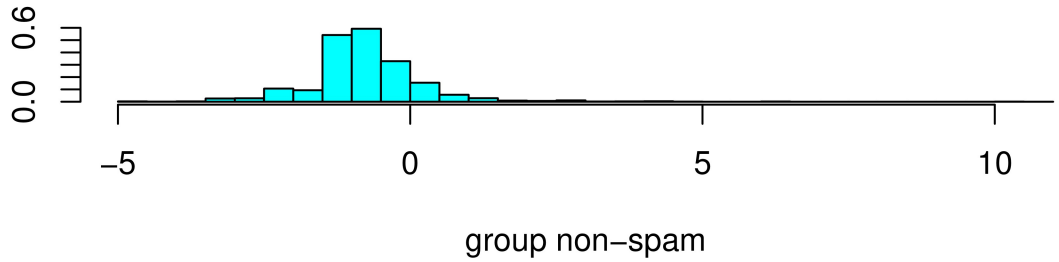
```
lda1 <- lda(s_ns_class~., data = model.train.norm)
lda2 <- lda(s_ns_class~., data = model.valid.norm)


lda.plot <- cbind(model.train.norm, predict(lda1)$x)
ggplot(lda.plot, aes(LD1,LD1)) +
  geom_point(aes(color=s_ns_class), alpha=0.6)+
    ggtitle("LDA Plot for Training Dataset")
```

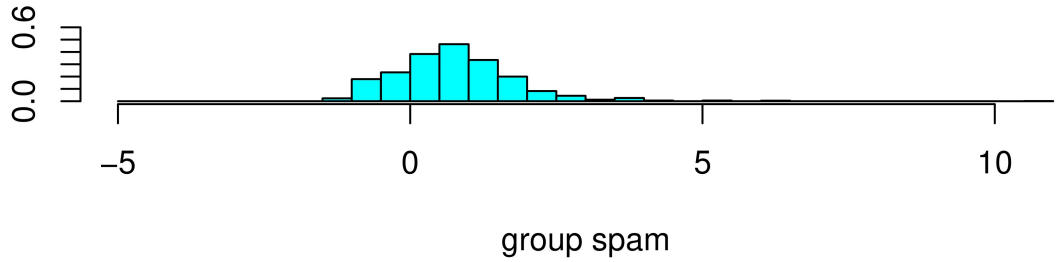



```
lda2.plot <- cbind(model.valid.norm, predict(lda2)$x)
ggplot(lda2.plot, aes(LD1,LD1)) +
  geom_point(aes(color=s_ns_class), alpha=0.6) +
    ggtitle("LDA Plot for Validation Dataset")
```




```
#histogram
plot(lda)
```





Answer7: From both scatter plots we can see a straight line with positive slope. As LD1 increases, the posterior probability for record to be classified as spam increases. This can be shown in the graph through red where its regular(non-spam) and blue when its spam.However, there is no clear separation bewtween classes on either scatter plot.

There are also a few differences between both scatter plots. For instance, the training LDA plot shows that the LD1 values range from -5 to slighlty over 10. Only 6 records have an LD1 value over 5, the rest of the records have an LD1 value between -5 and 5. On the other hand, the validation LDA plot shows that the range of LD1 values starts at -5 but does not reach 10, with most of the records having an LD1 value between -5 and 5.

From the histogram, we can read from the plot that LDA of spam emails tend towards values more than zero i.e postive and lDA of non spam(regular) emails tend towards values lower than zero i.e negative. Therfore, we can say that if positive LDA score will be more likely a spam email whereas negative LDA score has more chances of beind identifies as non-spam.

Question 8 :Generate the relevant confusion matrix. What are the sensitivity and specificity?

```
prediction.valid <- predict(lda, model.valid.norm)
names(prediction.valid)
```

```
## [1] "class"     "posterior" "x"
```

```r
# Confusion matrix

acc1 <- table(prediction.valid$class, model.valid.norm$s_ns_class)  # pred v actual
confusionMatrix(acc1, positive = "spam")
```

```
## Confusion Matrix and Statistics
##
##
##            non-spam spam
##   non-spam      507  120
##   spam           50  242
##
##                Accuracy : 0.815
##                  95% CI : (0.7884, 0.8396)
##     No Information Rate : 0.6061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.599
##
##  Mcnemar's Test P-Value : 1.209e-07
##
##             Sensitivity : 0.6685
##             Specificity : 0.9102
##          Pos Pred Value : 0.8288
##          Neg Pred Value : 0.8086
##              Prevalence : 0.3939
##          Detection Rate : 0.2633
##    Detection Prevalence : 0.3177
##       Balanced Accuracy : 0.7894
##
##        'Positive' Class : spam
##
```

```r
mean(prediction.valid$class == model.valid.norm$s_ns_class)* 100 #percentage accuracy
```

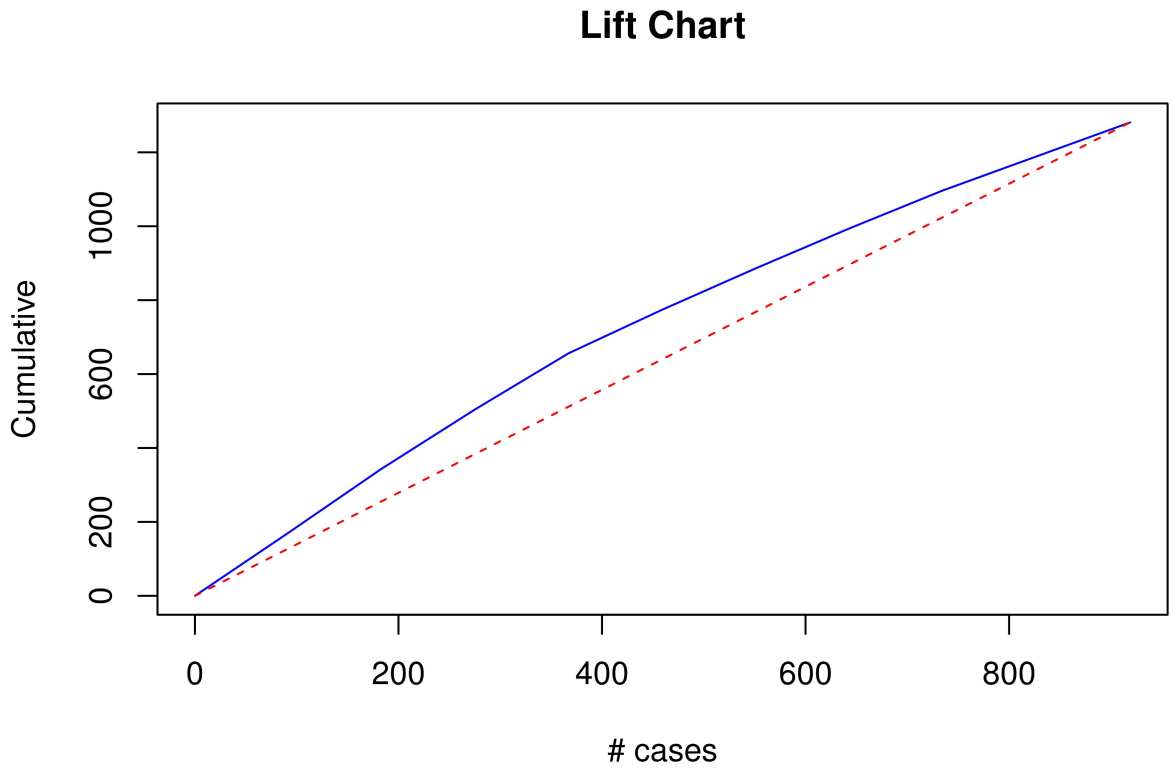
```
## [1] 81.50163
```

Answer 8: From above output, following are the results: Sensitivity : 0.6740
Specificity : 0.9013

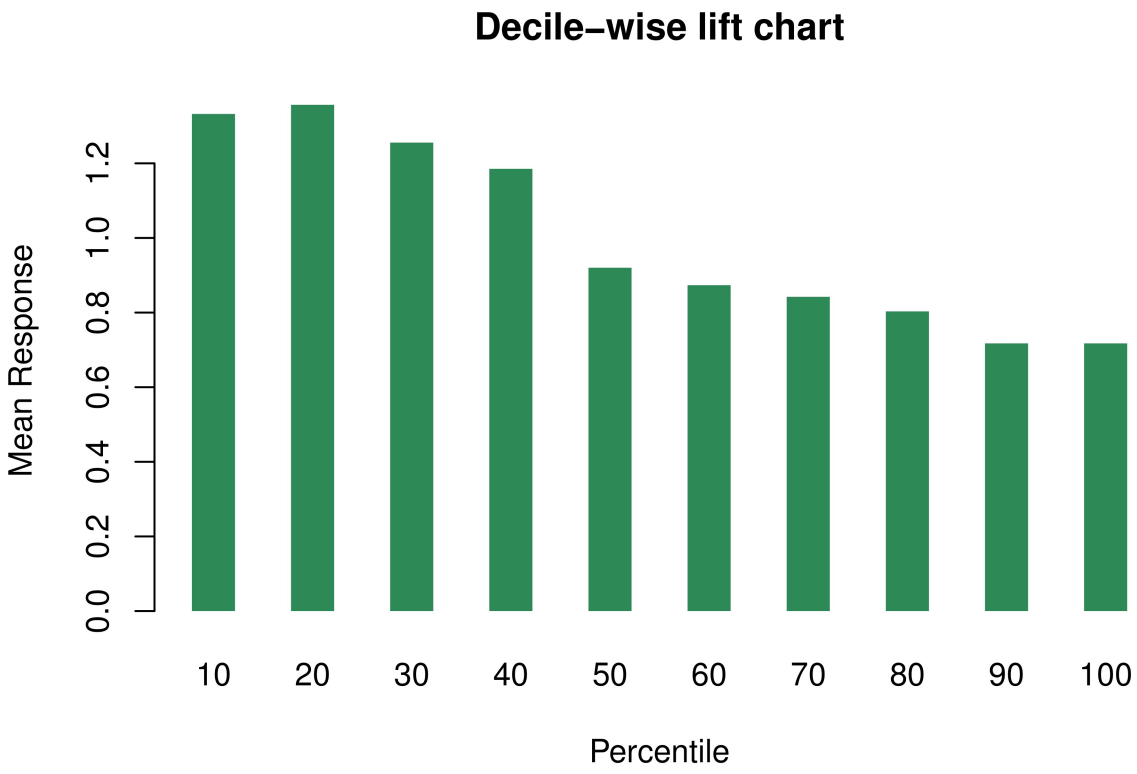
Question 9: Generate lift and decile charts for the validation dataset and evaluate the effectiveness of the model in identifying spams.

```r
#Calculating the gain

gain <- gains(as.numeric(model.valid.norm$s_ns_class),prediction.valid$x[,1] , groups=10)
spam <- as.numeric(model.valid.norm$s_ns_class)
```

```
plot(c(0, gain$cume.pct.of.total*sum(spam)) ~ c(0, gain$cume.obs),
     xlab = "# cases", ylab = "Cumulative", main="Lift Chart", type="l",
     col="blue1")
lines(c(0,sum(spam))~c(0,dim(model.valid)[1]), col="red1", lty=2)
```




```
#generating decile-wise lift chart
barplot(gain$mean.resp / mean(spam), names.arg = gain$depth, xlab = "Percentile", space = 1.3,
        ylab = "Mean Response", main = "Decile-wise lift chart", col = "seagreen", border = NA)
```



Answer 9: LIFT CHART: The above lift chart gives us enough evidence to say that the model created outperforms the naive model at identifying spams.Althought the area between curves is not as significant, the model was still able to identify more spams faster than the naive model.

DECILE CHART: We know that a model exhibiting a good staircase decile analysis is one you can consider moving forward with. Athough second decile is higher than the first, decile charts after that shows ideal conditions from second decile and hence we can infer that odel is doing good as first two deciles shows the maximum variation and the decreases gradually from left to right as it should.

Question 10:Does accuracy of model changes if you use a probability threshold of 0.2. Explain your answer.

```
#nimish code
pred_5 <- factor( ifelse(prediction.valid$posterior[,2] >= 0.5, "spam","non-spam") )
pred_2 <- factor( ifelse(prediction.valid$posterior[,2] >= 0.2, "spam", "non-spam") )

confusionMatrix(table(pred_5,model.valid.norm$s_ns_class), positive = "spam")
```

```
## Confusion Matrix and Statistics
##
##
## pred_5     non-spam spam
##   non-spam      507  120
##   spam           50  242
```

```
##
##                Accuracy : 0.815
##                  95% CI : (0.7884, 0.8396)
##     No Information Rate : 0.6061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.599
##
##  Mcnemar's Test P-Value : 1.209e-07
##
##             Sensitivity : 0.6685
##             Specificity : 0.9102
##          Pos Pred Value : 0.8288
##          Neg Pred Value : 0.8086
##              Prevalence : 0.3939
##          Detection Rate : 0.2633
##    Detection Prevalence : 0.3177
##       Balanced Accuracy : 0.7894
##
##        'Positive' Class : spam
##
```

```
confusionMatrix(table(pred_2,model.valid.norm$s_ns_class), positive = "spam")
```

```
## Confusion Matrix and Statistics
##
##
## pred_2     non-spam spam
##   non-spam      347   28
##   spam          210  334
##
##                Accuracy : 0.741
##                  95% CI : (0.7114, 0.7691)
##     No Information Rate : 0.6061
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5015
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9227
##             Specificity : 0.6230
##          Pos Pred Value : 0.6140
##          Neg Pred Value : 0.9253
##              Prevalence : 0.3939
##          Detection Rate : 0.3634
##    Detection Prevalence : 0.5919
##       Balanced Accuracy : 0.7728
##
##        'Positive' Class : spam
##
```

Answer 10: After setting the threshold of probability of 0.2, we can clearly see that the accuracu of the model reduces. This is happening becauase now model is trying to identify an email to be spam when the

probabilty is over 0.2. Hence it misclassifies some emails as spam. Therefore the accuracy of the model decreases from 81.5% to 74.1%. We can also see the changes in sensitivity and specificity.