BUAN 6356 - Homework 3

Group No.9 (Shubhi Kala, Spoorthi Thatipally, Hao-Yu Lin, Loc Nguyen, Tatsat Joshi)

4/6/2020

Contents

Linear Discriminant Analysis

```
if(!require("pacman")) install.packages("pacman")
pacman::p_load(caret, data.table, leaps, forecast, tidyverse, GGally, reshape2, gains,
               MASS, grid, gridExtra)
search()
theme_set(theme_classic())
```

Read the data from the spambase url

```r
#Using fread for faster data reading and it returns data table by default
spambase.dt <- fread("http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data")

#Naming the Predictors
cnames = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.names",
cnames = gsub("([:,`])", "", as.matrix(cnames))
cnames = c(cnames[c(2:nrow(cnames))],"spam")
colnames(spambase.dt) = cnames
```

Solution to Q1

1. 10 Predictors with highest average difference

```r
# Normalize the data
  # Estimate preprocessing parameters
spambase.norm  <- preProcess(spambase.dt[,-"spam"], method = c("center", "scale"))

  # Transform the data using the estimated parameters
spambase.norm.dt <- predict(spambase.norm, spambase.dt)

# Divide the data in to spam and nonspam:
spam.dt <- filter(spambase.norm.dt[,-58], spambase.norm.dt$spam == 1)
non_spam.dt <- filter(spambase.norm.dt[,-58], spambase.norm.dt$spam == 0)

# calculate the mean for each variable
mean_spam <- sapply(spam.dt, mean)
mean_nonspam <- sapply(non_spam.dt, mean)
mean_dif <- mean_spam - mean_nonspam

# absolute value of the mean:
meandif_abs <- sapply(mean_dif, abs)

# Top 10 predictors with highest absolute difference between spam and non-spam average:
top10 <- sort(meandif_abs, decreasing = T)[c(1:10)]
top10.df <- data.frame(top10)
spam.var.names <- rownames(top10.df)
top10.df
```

```
##                              top10
## word_freq_your           0.7841941
## word_freq_000            0.6850596
## word_freq_remove         0.6795969
## char_freq_$              0.6622271
## word_freq_you            0.5599603
## word_freq_free           0.5386044
## word_freq_business       0.5385825
## word_freq_hp             0.5253205
## capital_run_length_total 0.5098533
## word_freq_our            0.4950309
```

Answer1:

- The top 10 predictors with highest average difference between spam and nonspam classes are word_freq_your,word_freq_000, word_freq_remove, char_freq_$, word_freq_you, word_freq_free, word_freq_business, word_freq_hp, capital_run_length_total, and word_freq_our. We will use these predictors for further analysis.
- We did the normalization or data cleaning before computing the difference of the spam average and non-spam average for all the predictors so that the predictors are scaled to the same dimensions.

Solution to Q2

2. Performing a linear discriminant analysis using the training dataset

```
library(caret)

# Split the data into training (80%) and validation/test set (20%)
set.seed(42)
training.index <- createDataPartition(spambase.norm.dt$spam, p = 0.8, list = FALSE)

# Training Dataset
spambase.train <- spambase.norm.dt[training.index, ]

# Validation Dataset
spambase.valid <- spambase.norm.dt[-training.index, ]

spam.train.norm <- spambase.train[, .SD, .SDcols = c(spam.var.names, "spam")]
spam.valid.norm <- spambase.valid[, .SD, .SDcols = c(spam.var.names, "spam")]

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

```
## Call:
## lda(spam ~ ., data = spam.train.norm)
##
## Prior probabilities of groups:
##         0         1
## 0.6036403 0.3963597
##
## Group means:
##   word_freq_your word_freq_000 word_freq_remove `char_freq_$` word_freq_you
## 0     -0.3096578    -0.2707002       -0.2633643    -0.2577604    -0.2136983
## 1      0.4624814     0.4193563        0.3869652     0.4106747     0.3381014
##   word_freq_free word_freq_business word_freq_hp capital_run_length_total
## 0     -0.2163270         -0.2117147    0.2193507               -0.1972143
## 1      0.3562333          0.3134029   -0.3166362                0.2987029
##   word_freq_our
## 0    -0.1976282
## 1     0.2877977
##
## Coefficients of linear discriminants:
##                                LD1
## word_freq_your            0.4019434
## word_freq_000             0.3551827
```

```
## word_freq_remove          0.4153207
## `char_freq_$`             0.2850664
## word_freq_you             0.2013096
## word_freq_free            0.3521105
## word_freq_business        0.1716707
## word_freq_hp             -0.2340641
## capital_run_length_total  0.3844893
## word_freq_our             0.2452462
```

Answer 2:

- The LDA algorithm starts by finding directions that maximize the ratio of between-class variability to within-class variability, then use these directions to predict the class of individuals.
- The linear discriminant analysis can be easily computed using the function lda() [MASS package].
- The lda() outputs contain the following elements: Here 1 : spam class and 0 : non-spam class
    1. Prior probabilities of groups: the proportion of training observations in each group. For example, there are 40% of the training observations in the spam class.
    2. Group means: group center of gravity. Shows the mean of each variable in each group.
    3. Coefficients of linear discriminants: Shows the linear combination of predictor variables that are used to form the LDA decision rule. LD1 is the discriminant function which discriminates the spam and non spam classes based on the cutoff/threashold value (By default = 0.5) The values lower than 0.5 would be predicted as non-spam wheras the values above the threshold value would be predicted as spam. The coefficients are the weights whereby the variables compose this function.

Solution to Q3

3. Prior Probalities

```
# Prior probablities of the model
lda.train$counts
```

```
##    0    1
## 2222 1459
```

```
lda.train$prior
```

```
##         0         1
## 0.6036403 0.3963597
```

Answer 3:

Here class 0 is non-spam and 1 is spam. * From the above computed output we infer that the prior Probability of spam class is 39.6% and that of non-spam class is 60.4%.

- Prior probablity is the probability of randomly selecting an observation from class 'i' from the training set. The proportion of data belonging to each group unless specified.
- Because there are 1459 observations of spam class in the original data set (3681 observations total) we know that the prior probabilities should be close to 39.6% as computed above for actual spam class unless some other prior property is specified.

Solution to Q4

4. Coefficients of Linear Discriminants

```
# Coefficients of Linear Discriminants
lda.train$scaling
```

```
##                                LD1
## word_freq_your            0.4019434
## word_freq_000             0.3551827
## word_freq_remove          0.4153207
## `char_freq_$`             0.2850664
## word_freq_you             0.2013096
## word_freq_free            0.3521105
## word_freq_business        0.1716707
## word_freq_hp             -0.2340641
## capital_run_length_total  0.3844893
## word_freq_our             0.2452462
```

Answer 4:

- The coefficients of linear discriminant is used to calculate the LD score for the classification process and on that the final classification would be performed.
- lda$scaling return a matrix which transforms observations to discriminant functions, say LD1.
- These are the coefficients for each discriminant. For example the first linear discriminant (LD1) function is the linear combination: (word_freq_your x 0.402) + (word_freq_000 x 0.355) + (word_freq_remove x 0.415) + (`char_freq_$`x 0.285) + (word_freq_you x 0.201) + (word_freq_free x 0.352) + (word_freq_business x 0.172) + (word_freq_hp x -0.234) + (capital_run_length_total x 0.384) + (word_freq_our x 0.245)

Solution to Q5

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

```
# Generate linear discriminants
lda.train$scaling
```

```
##                                LD1
## word_freq_your            0.4019434
## word_freq_000             0.3551827
## word_freq_remove          0.4153207
## `char_freq_$`             0.2850664
## word_freq_you             0.2013096
## word_freq_free            0.3521105
## word_freq_business        0.1716707
## word_freq_hp             -0.2340641
## capital_run_length_total  0.3844893
## word_freq_our             0.2452462
```

Answer 5:

- For each observation LDA score is calculated by using the Coefficient of LD and taking the weighted average of all the variables for that observation.
- The LDA model looks at the score from each function and uses the highest score to allocate a case to a category (prediction).
- These coefficients are use to calculate the posterior for each class based on the given the data of the record. Then we can assign the record to the class that has the highest posterior.
- The posterior probabilities of classes are calculated for the observation. Comparing with a threshold of 0.5 probability, the observation is either classified "spam" or "nonspam" based on whichever probability is higher than the other.

Solution to Q6

6. Number of Linear Discriminants in the model

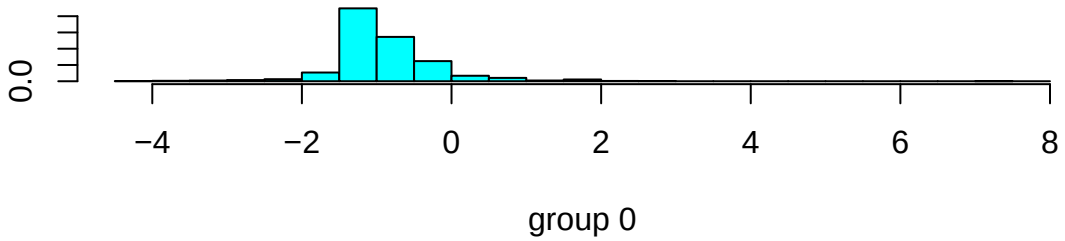
Answer 6: There is only 1 Linear Discriminant in the model since there are two classes into which the observations can be classified i.e. spam and non-spam. The number of LDA = total classes of membership - 1.

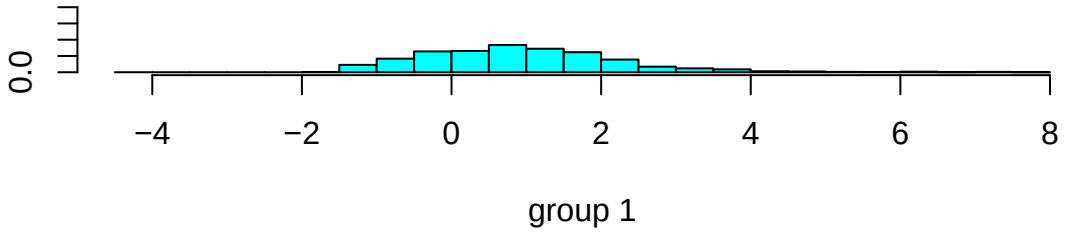
Solution to Q7

7. Generate LDA plot using the training and validation data and compare them.

```
# Plot of LDA using training data
plot(lda.train)
```

Plot from Training Dataset





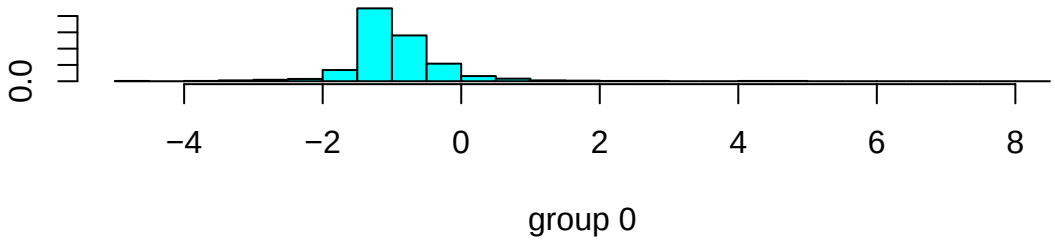
- The function plot() produces plots of the linear discriminants that obtained by computing LD1 for each of the training observations.
- From the plot we can infer that mostly the records to the left of 0 belongs to non-spam group and to the right belongs to spam group.
- The values after 0 in the non-spam graph are wrongly classified. They possibly belong to spam group.
- Similarly, the values towards the left of 0 in spam graph possibly belong to non-spam group.
- Also the LD score of non-spam class is mostly less than 0 and that of spam class is more than 0.

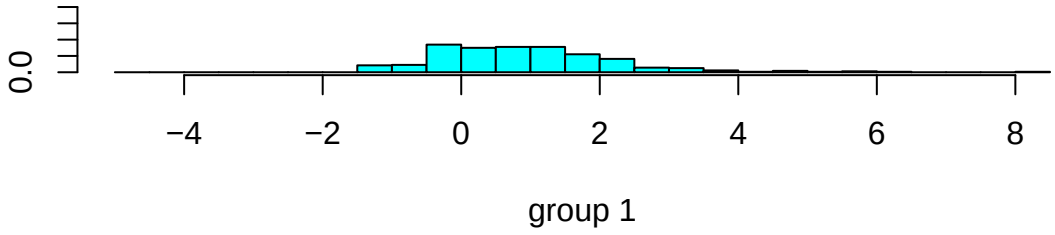
```
# Linear Discriminant Analysis of the validation dataset
lda.valid <- lda(spam~., data = spam.valid.norm)

# Predict - using Training data and plot
pred.train <- predict(lda.train, spam.train.norm)

# Predict - using Validation data
pred.valid <- predict(lda.valid, spam.valid.norm)
# LDA plot using validation data
plot(lda.valid)
```

Plot from Validation Dataset





- Using the function plot() produces plots of the linear discriminants that obtained by computing LD1 for each of the training observations.
- Plot from validation set can be interpreted in the similar manner.
- These plots illustrate the separation between spam and non-spam groups as well as overlapping areas that are potential for mix-ups when predicting classes.

How are the plots different? One plot is generated by running lda on training dataset and other one is generated by running lda on validation dataset.

Both look similar with some overlap between detection of spams and non-spams.

Solution to Q8

8. Relevant confusion matrix and measures of Sensitivity and Specificity

```
# Confusion matrix when the cutoff is default
acc1 <- table(prediction = pred.valid$class, actual = spam.valid.norm$spam)  # pred v actual
confusionMatrix(acc1, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##            actual
```

```
## prediction   0   1
##          0 541 117
##          1  25 237
##
##                Accuracy : 0.8457
##                  95% CI : (0.8207, 0.8684)
##     No Information Rate : 0.6152
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6573
##
##  Mcnemar's Test P-Value : 2.231e-14
##
##             Sensitivity : 0.6695
##             Specificity : 0.9558
##          Pos Pred Value : 0.9046
##          Neg Pred Value : 0.8222
##              Prevalence : 0.3848
##          Detection Rate : 0.2576
##    Detection Prevalence : 0.2848
##       Balanced Accuracy : 0.8127
##
##        'Positive' Class : 1
##
```

Answer 8.

Confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. As we see here, the confusion matrix is computed for Spam (1) and non-spam (0) class. On the basis of the values

Here positive = 1 means class of interest is 1 i.e. spam

Confusion matrix gives us:

Correctly predicted spam values = 237 Incorrectly predicted spam = 117. Correctly predicted non-spam values = 541. Incorrectly predicted non-spam values = 25.

From the above computation, Sensitivity : 0.6695 and Specificity : 0.9558

- Sensitivity and specificity are inversely proportional, meaning that as the sensitivity increases, the specificity decreases and vice versa.
- Sensitivity: the proportion of records which are truely classified as spam .
- Specificity: the proportion of records which are not spam are truely classified as non-spam
- Based on the results, we can conclude that:

1) The sensitivity is 67% (67 out of 100 observations are correctly classified as spam);
2) The specifity is 96% (5% of non-spam email are classified as spam)

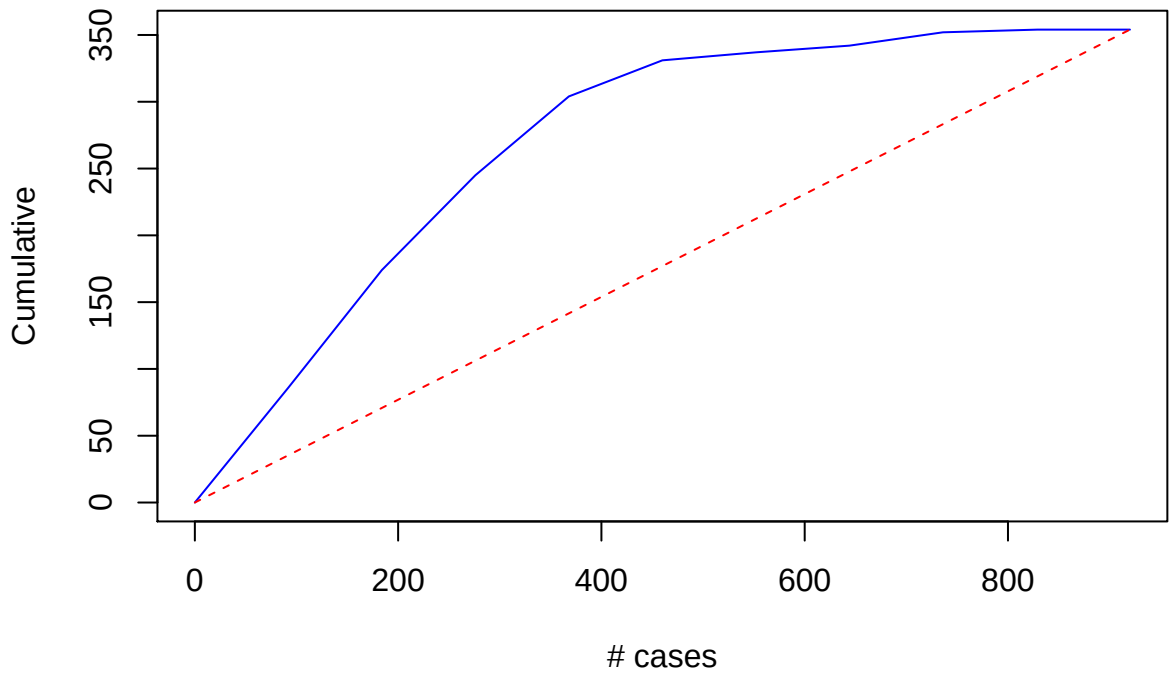
Solution to Q9

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

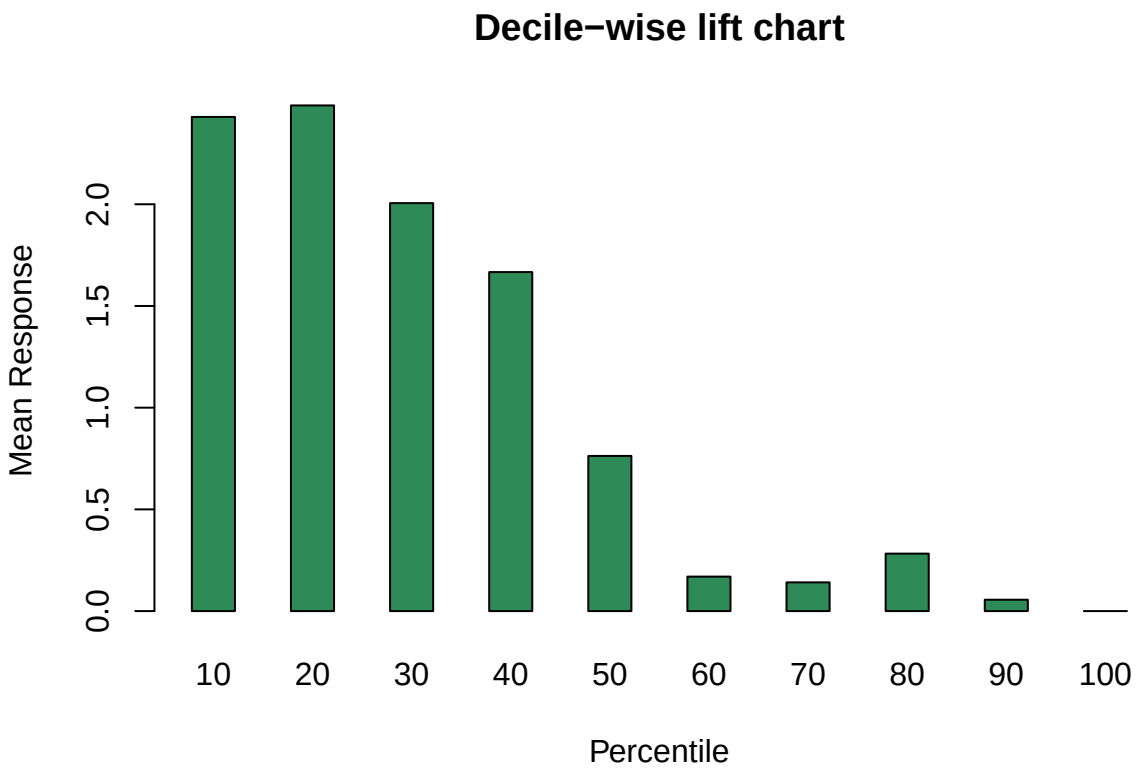
```
# Lift chart
lift.actual <- spam.valid.norm
lift.pred <- pred.valid$posterior[,2]
gain <- gains(lift.actual$spam, lift.pred)

plot(c(0, gain$cume.pct.of.total*sum(lift.actual$spam)) ~ c(0, gain$cume.obs),
     xlab = "# cases", ylab = "Cumulative", type="l",
     col="blue1")

lines(c(0,sum(lift.actual$spam))~c(0,dim(lift.actual)[1]), col="red1", lty=2)
```




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



Answer 9:

- The Lift chart is constructed with the cumulative number of cases (in descending order of probability) on the x-axis and the cumulative number of true positives(here the class of interest is spam) on the y-axis as shown below.
- True positives are those observations from the important class (here class 1) that are classified correctly.
- The dashed line is a reference line. For any given number of cases (the x-axis value), it represents the expected number of positives we would predict if we did not have a model but simply selected cases at random. It provides a benchmark against which we can see performance of the model.
- The model is pretty good in identifying spam emails. We can see the lift increases without any major interruption until the number of case reach around 300 and then continues as a straight line with a little increase until the end. This means that the model place almost all the 1 class (spam) at the beginning of the gain table (where the actual 1 class located).
- Also, when we look at the Decile-wise lift chart, we can see that with 10% of the records the model performs 2.5 times better than randomly assigning 1 and 0.
- we can say that we can classify 2.5 times the number of emails by selecting only 10% of the observations based on the model as compared to 10% observations selection randomly without a model.

Solution to Q10

10. Accuracy of model when threshold of 0.2 is taken

```
# Examine accuracy using confusion matrix
confusionMatrix(factor(1*(pred.valid$posterior[,2] > 0.2)),
                reference = factor(spam.valid.norm$spam), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 472  33
##          1  94 321
##
##                Accuracy : 0.862
##                  95% CI : (0.838, 0.8836)
##     No Information Rate : 0.6152
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7175
##
##  Mcnemar's Test P-Value : 1.014e-07
##
##             Sensitivity : 0.9068
##             Specificity : 0.8339
##          Pos Pred Value : 0.7735
##          Neg Pred Value : 0.9347
##              Prevalence : 0.3848
##          Detection Rate : 0.3489
##    Detection Prevalence : 0.4511
##       Balanced Accuracy : 0.8704
##
##        'Positive' Class : 1
##
```

Answer 10:

The Accuracy will increase by 1.89%.

- When the cutoff is default = 0.5, the accuracy is 0.846. but when cutoff is changed to 0.2 the accuracy increases to 0.862 as computed above.
- Also, we change the threshold to 0.2, the sensitivity increase dramatically from 67% to 90% with the sacrifice of specificity (from 95.6% to 83.4%).
- To decide a correct cutoff for the model, we should first identify the cost of misclassification of each class and choose the cutoff that can produce acceptable sensitivity and specificity.