UTD 2016 Spring

CS6301.5U1 Advanced Computational Methods for Data Science

Assignment 4 – Classification Model Building

(Logistic Regression, LDA, QDA, Naïve Bayes)

Name: Yu Zhang, Mingyue Sun

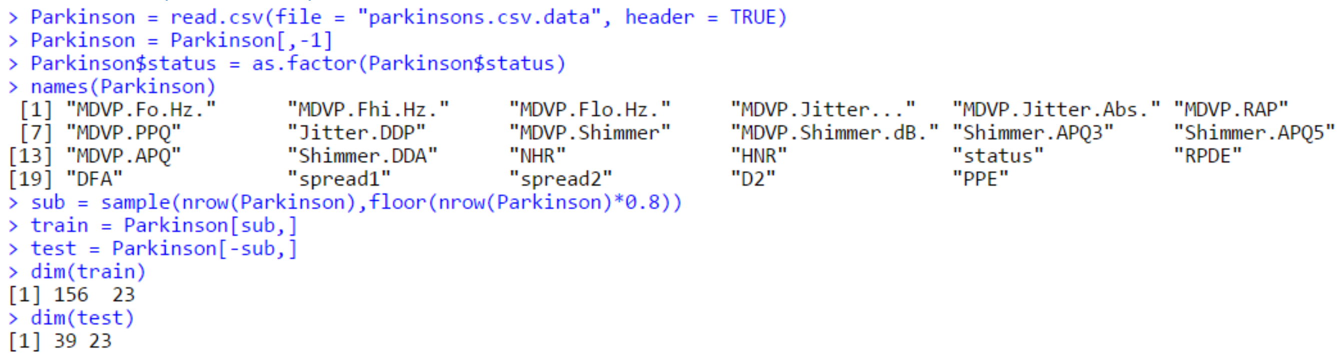
NetID: yxz141631, mxs151730

Part I. Parkinson's Disease Dataset

1.1 Overview

The main aim of the data is to discriminate healthy people from those with PD, according to "status" column which is set to 0 for healthy and 1 for PD. This dataset contains 195 instances with 23 attributes. We will split the data into train (80%) and test (20%) sets to train model and test accuracy. Use logistic regression, LDA, QDA and naïve Bayes model to predict "status" in test set and compare the accuracy between models.

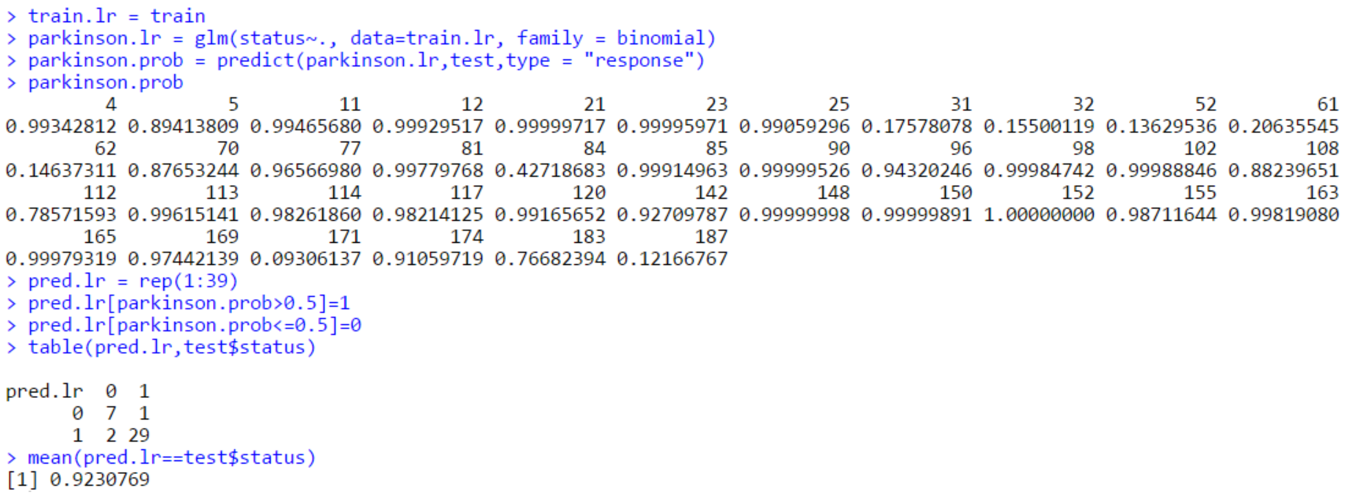
1.2 Data Import and Split



```
> Parkinson = read.csv(file = "parkinsons.csv.data", header = TRUE)
> Parkinson = Parkinson[,-1]
> Parkinson$status = as.factor(Parkinson$status)
> names(Parkinson)
 [1] "MDVP.Fo.Hz."      "MDVP.Fhi.Hz."     "MDVP.Flo.Hz."     "MDVP.Jitter..."   "MDVP.Jitter.Abs." "MDVP.RAP"
 [7] "MDVP.PPQ"         "Jitter.DDP"       "MDVP.Shimmer"     "MDVP.Shimmer.dB." "Shimmer.APQ3"     "Shimmer.APQ5"
[13] "MDVP.APQ"         "Shimmer.DDA"      "NHR"              "HNR"              "status"           "RPDE"
[19] "DFA"              "spread1"          "spread2"          "D2"               "PPE"
> sub = sample(nrow(Parkinson),floor(nrow(Parkinson)*0.8))
> train = Parkinson[sub,]
> test = Parkinson[-sub,]
> dim(train)
[1] 156  23
> dim(test)
[1] 39 23
```

After split the size of training dataset is 156 and the size of testing dataset tis 39. We will use the training dataset to train models and the test dataset to predict and calculate the accuracy of corresponding models.

1.3 Logistic Regression

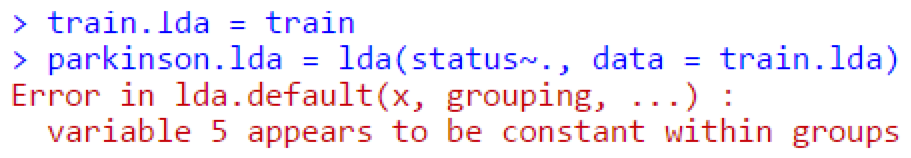


```
> train.lr = train
> parkinson.lr = glm(status~., data=train.lr, family = binomial)
> parkinson.prob = predict(parkinson.lr,test,type = "response")
> parkinson.prob
         4          5         11         12         21         23         25         31         32         52         61 
0.99342812 0.89413809 0.99465680 0.99929517 0.99999717 0.99995971 0.99059296 0.17578078 0.15500119 0.13629536 0.20635545 
        62         70         77         81         84         85         90         96         98        102        108 
0.14637311 0.87653244 0.96566980 0.99779768 0.42718683 0.99914963 0.99999526 0.94320246 0.99984742 0.99988846 0.88239651 
       112        113        114        117        120        142        148        150        152        155        163 
0.78571593 0.99615141 0.98261860 0.98214125 0.99165652 0.92709787 0.99999998 0.99999891 1.00000000 0.98711644 0.99819080 
       165        169        171        174        183        187 
0.99979319 0.97442139 0.09306137 0.91059719 0.76682394 0.12166767 
> pred.lr = rep(1:39)
> pred.lr[parkinson.prob>0.5]=1
> pred.lr[parkinson.prob<=0.5]=0
> table(pred.lr,test$status)
       
pred.lr  0  1
      0  7  1
      1  2 29
> mean(pred.lr==test$status)
[1] 0.9230769
```

The logistic regression model will return the probability of reach classification (as "parkinson.prob" shows). Since there are only two classifications, we define the hard limit of classification as if probability >0.5, the classification of the instance should be assigned as 1.

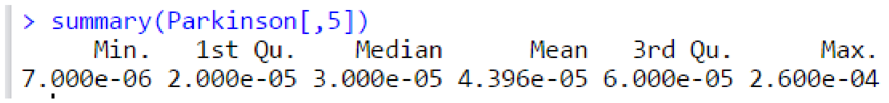
Otherwise, it would be assigned as 0. In the result of logistic regression model, the accuracy is 0.9230769. 2 of 9 “0”s were predicted as 1. 1 of 30 “1”s were predicted as 0.

1.4 LDA



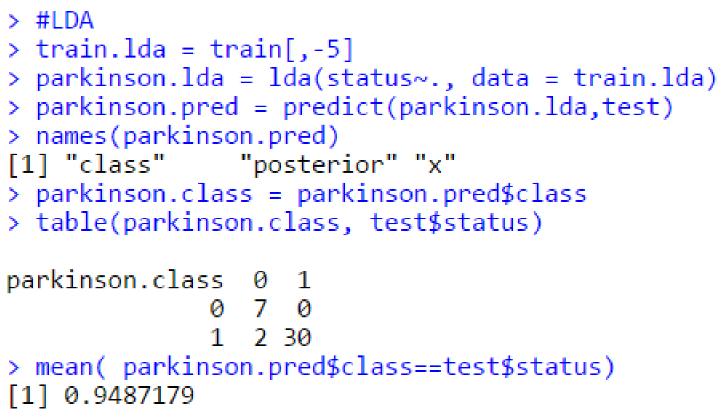
```
> train.lda = train
> parkinson.lda = lda(status~., data = train.lda)
Error in lda.default(x, grouping, ...) :
  variable 5 appears to be constant within groups
```

When we first time tried to build model based on LDA, R returns error message that values in attribute 5 is likely to be constant. So we locate the fifth attribute to check the status.



```
> summary(Parkinson[,5])
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
7.000e-06 2.000e-05 3.000e-05 4.396e-05 6.000e-05 2.600e-04 
```

According to the summary of 5th attribute we can observe that the data in this group is very small and the variance is also small. That’s why the system regard them as constant values. Therefore, we get rid of this attribute and rebuild the model.



```
> #LDA
> train.lda = train[,-5]
> parkinson.lda = lda(status~., data = train.lda)
> parkinson.pred = predict(parkinson.lda,test)
> names(parkinson.pred)
[1] "class"     "posterior" "x"        
> parkinson.class = parkinson.pred$class
> table(parkinson.class, test$status)

parkinson.class  0  1
              0  7  0
              1  2 30
> mean( parkinson.pred$class==test$status)
[1] 0.9487179
```

The accuracy of the model in this case is 0.9487179. 2 of 9 “0”s were predicted as “1”. All the “1”s are correctly predicted.

1.5 QDA

```
> #QDA
> train.qda = train
> parkinson.qda = qda(status~., data = train.qda)
> QDA.pred = predict(parkinson.qda,test)
> QDA.class = QDA.pred$class
> table(QDA.class, test$status)

QDA.class  0  1
        0  5  0
        1  4 30
> mean( QDA.class==test$status)
[1] 0.8974359
```

The accuracy of the model in this case is 0.8974359. 4 of 9 “0”s were predicted as “1”. All the “1”s are correctly predicted.

1.6 Naïve Bayes

To build naïve Bayes model on R, we need install and library “e1071” package.

```
> #Naive Bayes
> train.nb = train
> train.nb = naiveBayes(status~., data = train.nb, laplace = 3)
> NB.pred = predict(train.nb, test)
> table(NB.pred,test$status)

NB.pred  0  1
      0  8  8
      1  1 22
> mean( NB.pred==test$status)
[1] 0.7692308
```

The accuracy of the model in this case is 0.7692308. 1 of 9 “0”s were predicted as “1”. 8 of 30 “1”s were predicted as “0”.

1.7 Conclusion

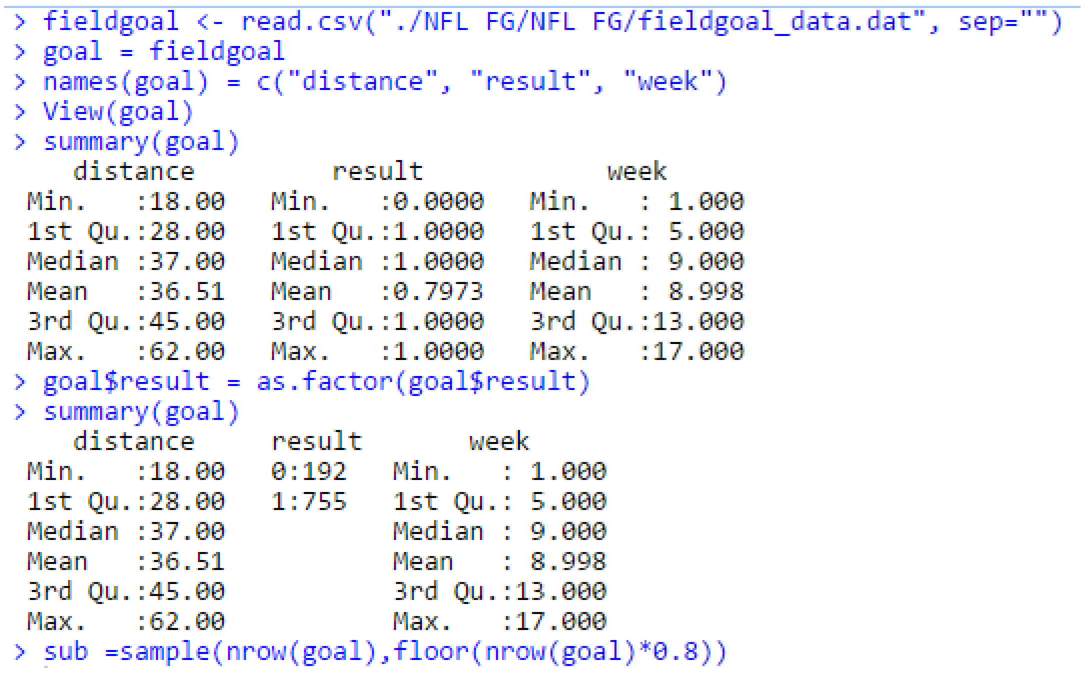
Observing the results above we can draw a conclusion that in this case, the LDA model has the best performance which has reached accuracy of 0.948719. Only 2 cases are wrongly predicted. Logistic regression and QDA also have good performances, the accuracies of which are 0.9230769 and 0.8947359. However, the Naïve Bayes model has a bad performance, the accuracy of which only has 0.7692308. The reason behind this may because of the number of attributes and size of training dataset. Since the data has 23 predictors, this is relatively large for predicting. However, the training set has only 156 instances. Many combination of cases may not even be concluded into the model. Even if we used Laplace smoothing method. This still cannot eliminate the negative influence to the performance.

Part II. NFL Field Goal Information

2.1. Overview

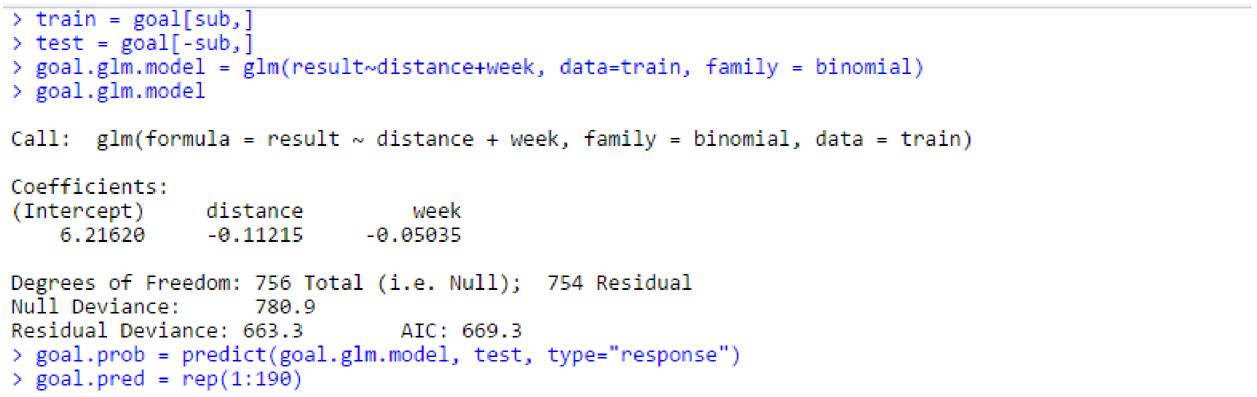
In this section, we are going to deal with a dataset which describes all field goal attempts information for a NFL season in 2003. It has three attributes: 1. Yardage, which describes how far away the goal attempt was shot, 2. Success indicator, the class variable, which tells whether the attempt was a success or a failure, 3. Week number, which is just the number of week during the season 2003. Since this is a typical classification problem, we would like to build a good model that could predict the success/failure variable based upon the other two yardage and week variables. The technique that we will use here are: 1. Logistic regression, 2. LDA, 3. QDA, 4. Naïve Bayes. And we will try to create the most accurate (with highest accuracy) possible in terms of misclassification rate.

2.2. Logistic regression (LR)



```
> fieldgoal <- read.csv("./NFL FG/NFL FG/fieldgoal_data.dat", sep="")
> goal = fieldgoal
> names(goal) = c("distance", "result", "week")
> View(goal)
> summary(goal)
    distance          result             week       
 Min.   :18.00   Min.   :0.0000   Min.   : 1.000  
 1st Qu.:28.00   1st Qu.:1.0000   1st Qu.: 5.000  
 Median :37.00   Median :1.0000   Median : 9.000  
 Mean   :36.51   Mean   :0.7973   Mean   : 8.998  
 3rd Qu.:45.00   3rd Qu.:1.0000   3rd Qu.:13.000  
 Max.   :62.00   Max.   :1.0000   Max.   :17.000  
> goal$result = as.factor(goal$result)
> summary(goal)
    distance     result       week       
 Min.   :18.00   0:192   Min.   : 1.000  
 1st Qu.:28.00   1:755   1st Qu.: 5.000  
 Median :37.00           Median : 9.000  
 Mean   :36.51           Mean   : 8.998  
 3rd Qu.:45.00           3rd Qu.:13.000  
 Max.   :62.00           Max.   :17.000  
> sub =sample(nrow(goal),floor(nrow(goal)*0.8))
```

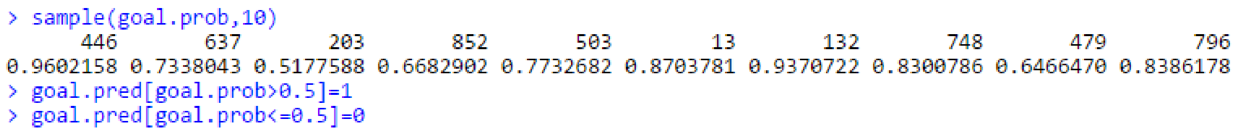
Shown above are the steps we applied in loading original dataset, converting the class variable into categorical type and splitting the original dataset into an 80% training set and a 20% testing set.



```
> train = goal[sub,]
> test = goal[-sub,]
> goal.glm.model = glm(result~distance+week, data=train, family = binomial)
> goal.glm.model

Call:  glm(formula = result ~ distance + week, family = binomial, data = train)

Coefficients:
(Intercept)     distance         week  
    6.21620     -0.11215     -0.05035  

Degrees of Freedom: 756 Total (i.e. Null);  754 Residual
Null Deviance:	    780.9 
Residual Deviance: 663.3 	AIC: 669.3
> goal.prob = predict(goal.glm.model, test, type="response")
> goal.pred = rep(1:190)
```

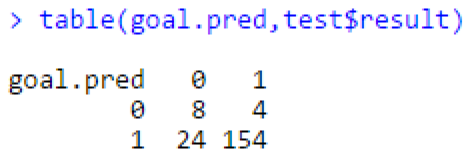
Shown above are steps that we used to build the logistic regression model with the training set and then

apply this model on the testing set.



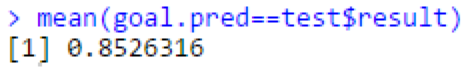
```
> sample(goal.prob,10)
      446       637       203       852       503        13       132       748       479       796 
0.9602158 0.7338043 0.5177588 0.6682902 0.7732682 0.8703781 0.9370722 0.8300786 0.6466470 0.8386178 
> goal.pred[goal.prob>0.5]=1
> goal.pred[goal.prob<=0.5]=0
```

Above is a sample of prediction result after we applied the logistic regression model to the testing set. As we can see here that some of the sample has a prediction value which is larger than 0.5 and some has a value which is equal to or less than 0.5, we would like to label those samples with value > 0.5 to success (1) and the other with value <= 0.5 to failure (0).



```
> table(goal.pred,test$result)

goal.pred   0   1
        0   8   4
        1  24 154
```

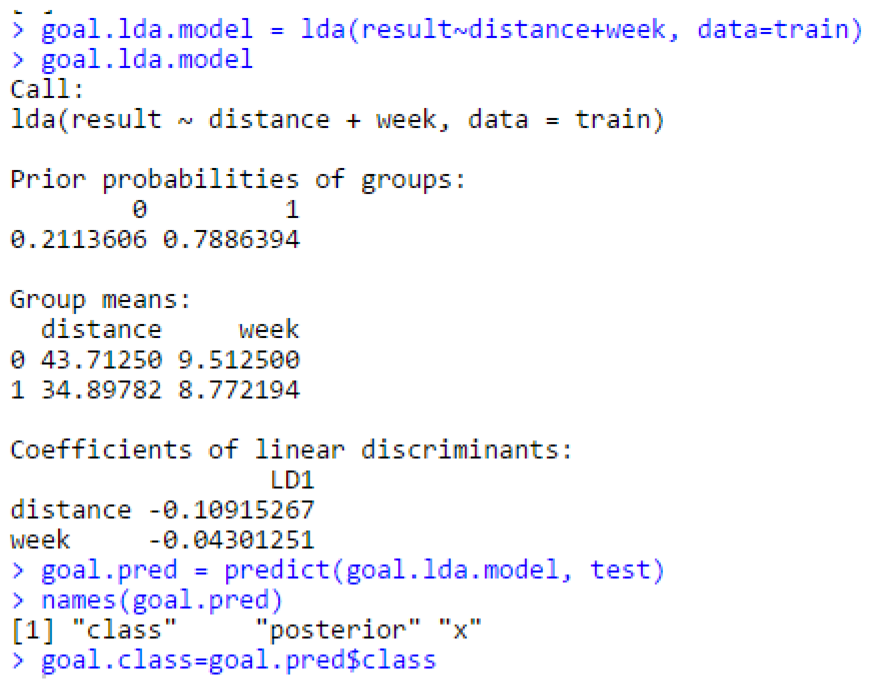
And then we created a table which summarized the prediction in comparison with the actual class label in the testing set. From the table we will see that this model predict 12 samples to be failure however it turns out only 8 of them were right (66.67%) and the other 4 were misclassified and it predicts 178 samples to be success while only 154 of them turned out to be right (86.52%)



```
> mean(goal.pred==test$result)
[1] 0.8526316
```

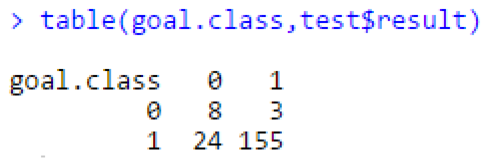
Then we calculated the overall accuracy of this model to be 85.26% (misclassification rate: 14.74%).

2.3. LDA



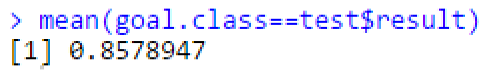
```
> goal.lda.model = lda(result~distance+week, data=train)
> goal.lda.model
Call:
lda(result ~ distance + week, data = train)

Prior probabilities of groups:
        0         1 
0.2113606 0.7886394 

Group means:
  distance     week
0 43.71250 9.512500
1 34.89782 8.772194

Coefficients of linear discriminants:
                 LD1
distance -0.10915267
week     -0.04301251
> goal.pred = predict(goal.lda.model, test)
> names(goal.pred)
[1] "class"     "posterior" "x"        
> goal.class=goal.pred$class
```

Shown above are steps that we used to build LDA model with the training set and then apply this model on the testing set.



```
> table(goal.class,test$result)

goal.class   0   1
         0   8   3
         1  24 155
```

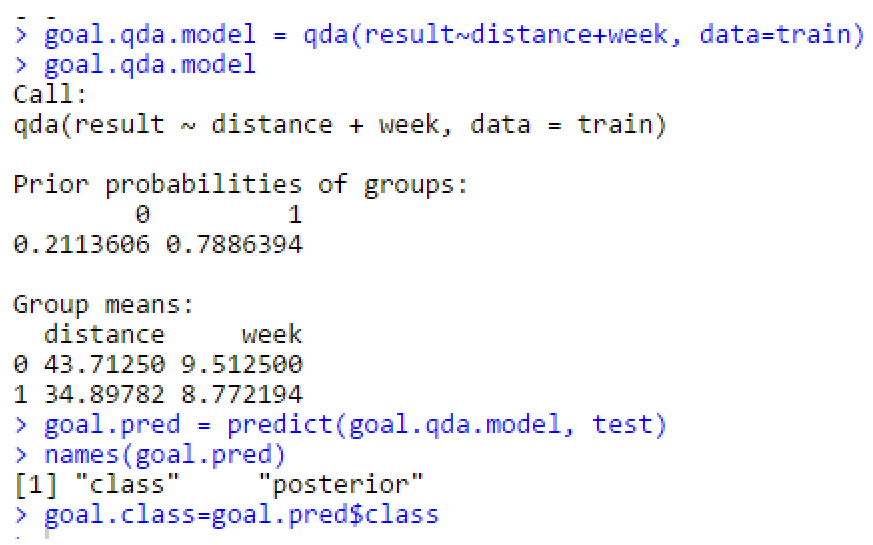
And then we created a table which summarized the prediction in comparison with the actual class label in the testing set. From the table we will see that this model predict 11 samples to be failure however it turns out only 8 of them were right (72.73%) and the other 3 were misclassified and it predicts 179 samples to be success while only 155 of them turned out to be right (86.59%). It looks like that the accuracy for this model to make the right prediction for both the failure and success class got increased in comparison with LR model.



```
> mean(goal.class==test$result)
[1] 0.8578947
```

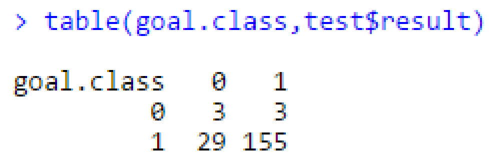
Then we calculated the overall accuracy of this model to be 85.80% (misclassification rate: 14.20%). And we can see the overall accuracy of this model indeed got increased with the increased accuracies of both classes.

2.4. QDA



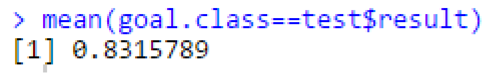
```
> goal.qda.model = qda(result~distance+week, data=train)
> goal.qda.model
Call:
qda(result ~ distance + week, data = train)

Prior probabilities of groups:
        0         1 
0.2113606 0.7886394 

Group means:
  distance     week
0 43.71250 9.512500
1 34.89782 8.772194
> goal.pred = predict(goal.qda.model, test)
> names(goal.pred)
[1] "class"     "posterior"
> goal.class=goal.pred$class
```

Shown above are steps that we used to build QDA model with the training set and then apply this model on the testing set.



```
> table(goal.class,test$result)

goal.class   0   1
         0   3   3
         1  29 155
```

And then we created a table which summarized the prediction in comparison with the actual class label in the testing set. From the table we will see that this model predict 6 samples to be failure however it turns out only 3 of them were right (50%) and the other 3 were misclassified and it predicts 184 samples to be success while only 155 of them turned out to be right (84.24%). It seems like that the accuracy for this model to make the right prediction for both the failure and success class got decreased in comparison of both LR and LDA model.



```
> mean(goal.class==test$result)
[1] 0.8315789
```

Then we calculated the overall accuracy of this model to be 83.16% (misclassification rate: 16.84%). And we can see the overall accuracy of this model indeed got decreased with the decreased accuracies of both classes.

2.5. Naïve Bayes (NB)

```
> library(e1071)
> goal.nb.model = naiveBayes(result~distance+week, data=train, laplace = 3)
> goal.nb.model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.2113606 0.7886394 

Conditional probabilities:
   distance
Y       [,1]     [,2]
  0 43.71250 7.868724
  1 34.89782 9.360488

   week
Y       [,1]     [,2]
  0 9.512500 4.584617
  1 8.772194 4.916930

> goal.pred=predict(goal.nb.model, test)
```

Shown above are steps that we used to build NB model with the training set and then apply this model on the testing set.

```
> table(goal.pred,test$result)

goal.pred   0   1
        0   7   3
        1  25 155
```

And then we created a table which summarized the prediction in comparison with the actual class label in the testing set. From the table we will see that this model predict 10 samples to be failure however it turns out only 7 of them were right (70%) and the other 3 were misclassified and it predicts 180 samples to be success while only 155 of them turned out to be right (86.11%). We can see that the ability of this model to make right prediction for failure class got increased in comparison of the LR model while its ability to right predict the success class got decreased a little compared with the LR model.

```
> mean(goal.pred==test$result)
[1] 0.8526316
```

Then we calculated the overall accuracy of this model to be 85.26% (misclassification rate: 14.74%) which is the same as the LR model. And we can foresee that the overall accuracy of this model to be equal to or around the accuracy of LR model with its increased accuracy for failure class and decreased accuracy for success class.

2.6. Conclusion

To summarize our analysis above, we would like to state that with these LR, LDA, QDA, NB model approaches to this NFL field goal dataset. We found the QDA might be the least favorable one we would choose to apply to this data since it has the lowest (83.16%) accuracy or highest misclassification rate (16.84%) and the LDA model will be the best model to fit this dataset with highest (85.80%) accuracy or lowest misclassification rate (14.20%). Both LR and NB models give the same accuracy (85.26%) which is slightly worse than the LDA model. And we could also see the LR model makes better prediction for the success (1) class while the NB model makes better predictions for the failure (0) class.