CS 5565

LAB3(Classificatio)

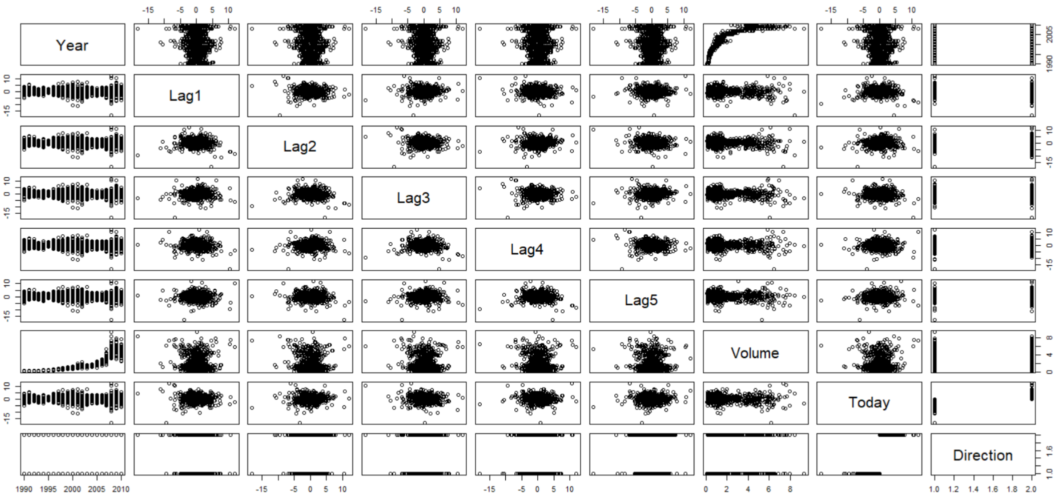
Shubhabrata Mukherjee

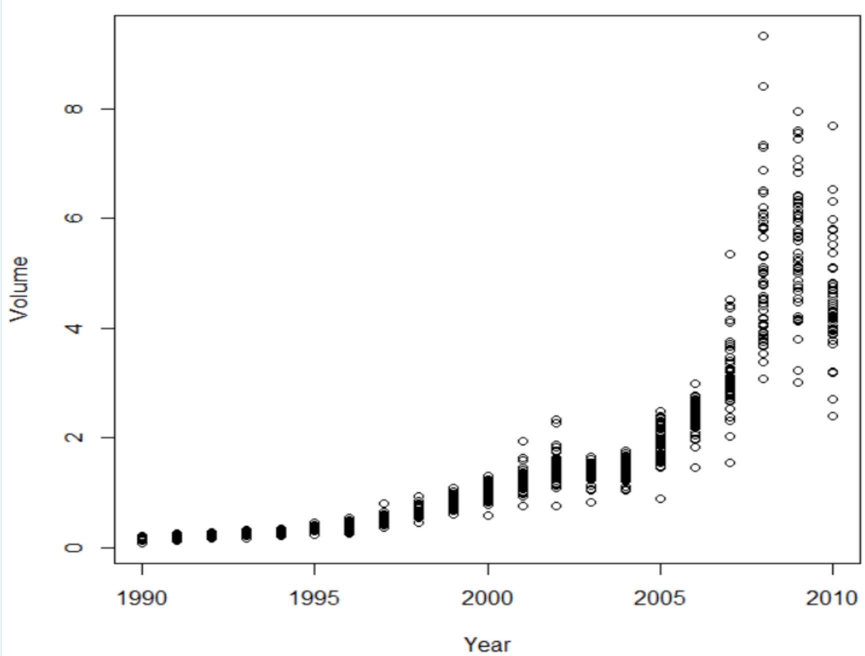
Id: 16201097

2) a)

```
> pairs(Weekly)
> summary(Weekly)
      Year          Lag1               Lag2               Lag3               Lag4               Lag5              Volume            Today          Direction
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950   Down:484
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580   1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540   Up  :605
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410   Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472   Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
```

```
> attach(Weekly)
> plot(Year, Volume)
```





From pairplot and Volume Vs Year plot, we can see there is a pattern. The volume increases exponentially with the year.

b)

```
> glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Weekly ,family =binomial)
> summary(glm.fits)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6949  -1.2565   0.9913   1.0849   1.4579

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

<u>Observation:</u>

The smallest p-value here is associated with Lag2. The coefficient for this predictor is positive so, the market had a positive return last week, then it is more likely to go up today.

c)

```
> glm.probs=predict(glm.fits,type ="response")
> glm.pred = rep ("Down" ,nrow(Weekly))
> glm.pred[glm.probs >.5]="Up"
> table(glm.pred,Direction)
        Direction
glm.pred Down  Up
    Down   54  48
    Up    430 557
> (54+557) /nrow(Weekly)
[1] 0.5610652
> mean (glm.pred == Direction )
[1] 0.5610652
```

From confusion matrix we can see that, The overall fraction of correct prediction is = 0.561. So logistic regression correctly predicted the movement of the market 56.1% of the time. It would appear that the logistic regression model is working a little better than random guessing.

However, this result is misleading because we trained and tested the model on the same set of 1, 250 observations. In other words, 100 − 56.1 = 43.9 % is the training error rate. As we have seen previously, the training error rate is often overly optimistic—it ends to underestimate the test error rate.

d)

Firstly we select the training data and test data part,

```
Console   Terminal ×   Jobs ×
~/Downloads/myr/
> train =(Year <2009)
> Weekly.train = Weekly [train,]
> dim (Weekly.train)
[1] 985   9
> Weekly.test = Weekly [!train,]
> dim (Weekly.test)
[1] 104   9
```

Then, we perform logistic regression on training data and we use the test data for prediction.

```
> glm.newfit= glm(Direction~Lag2,data = Weekly.train , family = binomial,subset=train)
> glm.probs=predict(glm.newfit,Weekly.test,type ="response")
> row2 = nrow(Weekly.test)
> glm.pred = rep ("Down",row2)
> glm.pred[glm.probs >.5]="Up"
```

And, finally calculate the prediction accuracy :

```
> Direction.test = Direction[!train]
> table(glm.pred,Direction.test)
        Direction.test
glm.pred Down Up
    Down    9  5
    Up     34 56
> mean (glm.pred == Direction.test)
[1] 0.625
```

e)

The same way as logistic regression, using LDA we get below results:

```
> mylda = lda(Direction~Lag2,data = Weekly.train)
> mylda
Call:
lda(Direction ~ Lag2, data = Weekly.train)

Prior probabilities of groups:
     Down        Up 
0.4477157 0.5522843 

Group means:
            Lag2
Down -0.03568254
Up    0.26036581

Coefficients of linear discriminants:
           LD1
Lag2 0.4414162
```

```
> Direction.test = Direction[!train]
> mylda.pred = predict(mylda,Weekly.test)
> mylda.class = mylda.pred$class
> table(mylda.class,Direction.test)
           Direction.test
mylda.class Down Up
       Down    9  5
       Up     34 56
> mean (mylda.class == Direction.test)
[1] 0.625
```

So, we get same accuracy level like logistic regression in LDA as well.

f)

The same way as LDA, using QDA we get below results:

```
> myqda = qda(Direction~Lag2,data = Weekly.train)
> myqda
Call:
qda(Direction ~ Lag2, data = Weekly.train)

Prior probabilities of groups:
     Down        Up 
0.4477157 0.5522843 

Group means:
            Lag2
Down -0.03568254
Up    0.26036581
```

```
> Direction.test = Direction[!train]
> myqda.pred = predict(myqda,Weekly.test)
> myqda.class = myqda.pred$class
> table(myqda.class,Direction.test)
           Direction.test
myqda.class Down Up
       Down    0  0
       Up     43 61
> mean (myqda.class == Direction.test)
[1] 0.5865385
```

So, we get accuracy level in QDA as 58.65%.

g)

Now we repeat the process using KNN where, K = 1 .

Firstly we select the training data and test data part,

```
> train.data = as.matrix(Lag2[ train])
> test.data = as.matrix(Lag2[!train])
> train.Direction = Direction [train ]
```

Then, we perform KNN on training data we use the test data for prediction.

```
> set.seed (1)
> myknn.pred = knn(train.data,test.data,train.Direction , k =1)
```

and finally, we calculate the accuracy of the model:

```
> table (myknn.pred , Direction.test)
          Direction.test
myknn.pred Down Up
      Down   21 30
      Up     22 31
> mean(myknn.pred == Direction.test)
[1] 0.5
```

So, we saw that KNN (K = 1) performance is poor compared to the Logistic regression, LDA and QDA.

h) So, from the accuracy calculated for above processes we get:

1) Accuracy for logistic regression for all variables: 56.1 %

2) Accuracy for logistic regression for Log2 variable: 62.5 %

3) Accuracy for LDA for Log2 variable: 62.5 %

4) Accuracy for QDA for Log2 variable: 58.65 %

5) Accuracy for KNN (K= 1) for Log2 variable: 50 %

From the above comparison we see that Logistic regression and LDA both with Lag2 variable produce maximum correct prediction probability.

I) Now , Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

2)a)

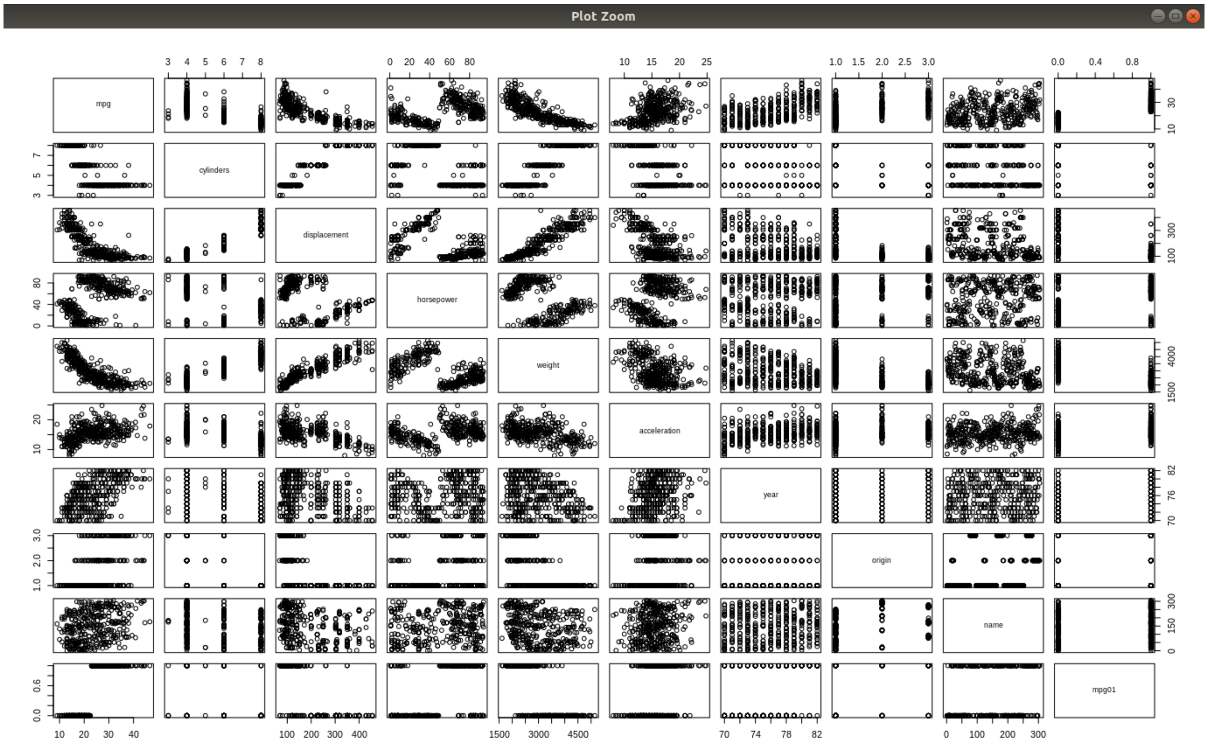
Following the instructions we get:

```
> myauto = read.csv("Auto.csv",header=T,na.strings='?')
> myauto = na.omit(myauto)
> #fix(myauto)
> mpg01 = rep(1,nrow(myauto))
> mpgmed = median(myauto$mpg)
> mpg01[myauto$mpg < mpgmed] = 0
> newauto = cbind(myauto,mpg01)
> fix(newauto)
```

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	mpg01
18	8	307	130	3504	12	70	1	chevrolet chevelle malibu	0
15	8	350	165	3693	11.5	70	1	buick skylark 320	0
18	8	318	150	3436	11	70	1	plymouth satellite	0
16	8	304	150	3433	12	70	1	amc rebel sst	0
17	8	302	140	3449	10.5	70	1	ford torino	0
15	8	429	198	4341	10	70	1	ford galaxie 500	0
14	8	454	220	4354	9	70	1	chevrolet impala	0
14	8	440	215	4312	8.5	70	1	plymouth fury iii	0
14	8	455	225	4425	10	70	1	pontiac catalina	0
15	8	390	190	3850	8.5	70	1	amc ambassador dpl	0
15	8	383	170	3563	10	70	1	dodge challenger se	0
14	8	340	160	3609	8	70	1	plymouth 'cuda 340	0
15	8	400	150	3761	9.5	70	1	chevrolet monte carlo	0
14	8	455	225	3086	10	70	1	buick estate wagon (sw)	0
24	4	113	95	2372	15	70	3	toyota corona mark ii	1
22	6	198	95	2833	15.5	70	1	plymouth duster	0
18	6	199	97	2774	15.5	70	1	amc hornet	0
21	6	200	85	2587	16	70	1	ford maverick	0
27	4	97	88	2130	14.5	70	3	datsun pl510	1

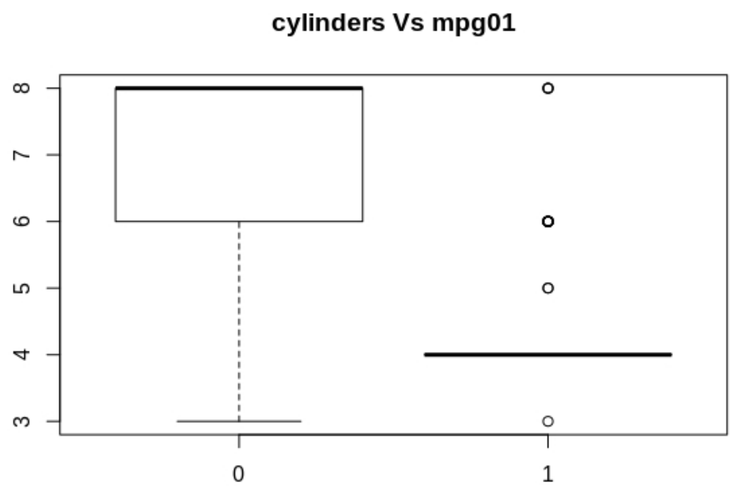
b)

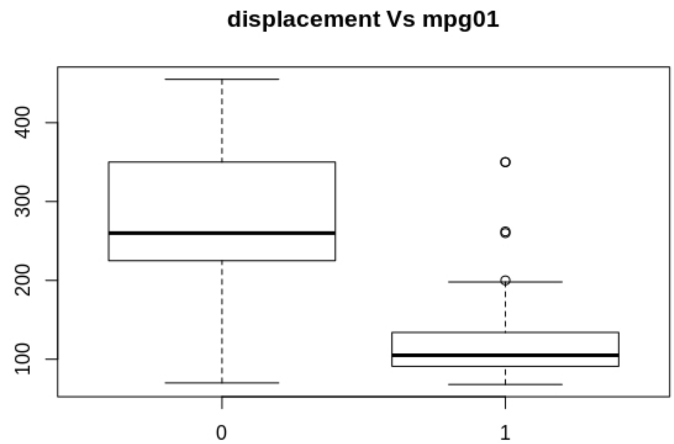
Using scatter-plot we get:

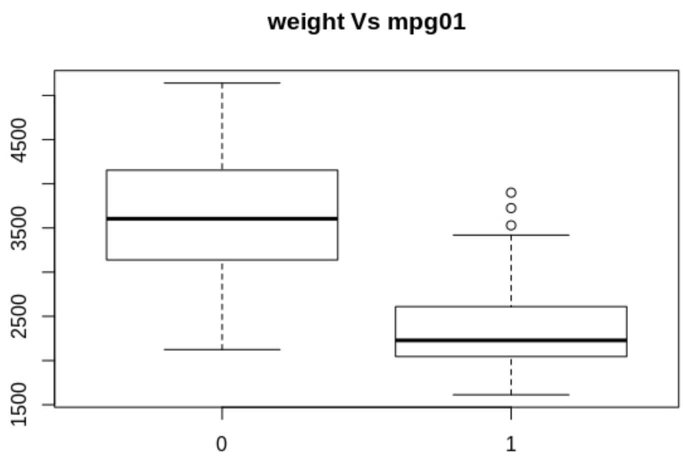


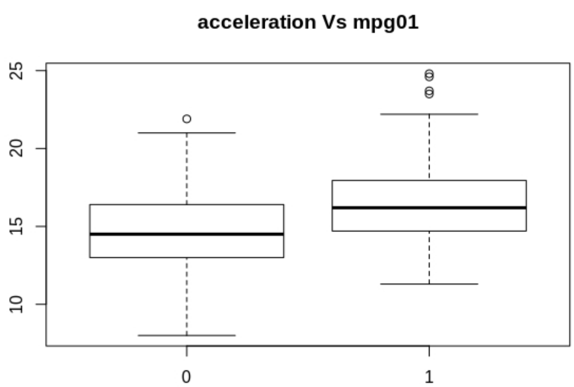
Using box-plots we get:

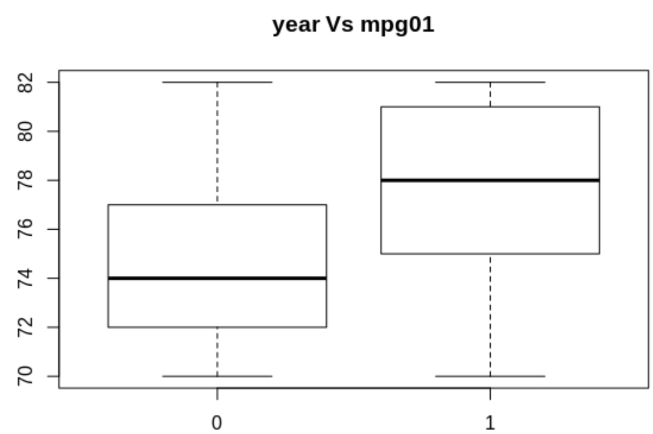
```
> boxplot(cylinders~mpg01,data=newauto,main = "cylinders Vs mpg01")
> boxplot(displacement~mpg01,data=newauto,main = "displacement Vs mpg01")
> boxplot(weight~mpg01,data=newauto,main = "weight Vs mpg01")
> boxplot(acceleration~mpg01,data=newauto,main = "acceleration Vs mpg01")
> boxplot(year~mpg01,data=newauto,main = "year Vs mpg01")
> boxplot(origin~mpg01,data=newauto,main = "origin Vs mpg01")
> boxplot(horsepower~mpg01,data=newauto,main = "horsepower Vs mpg01")
```

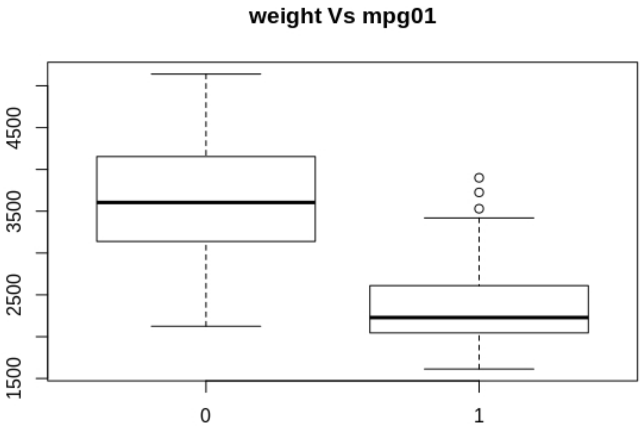

cylinders Vs mpg01
8
7
6
5
4
3
0
1

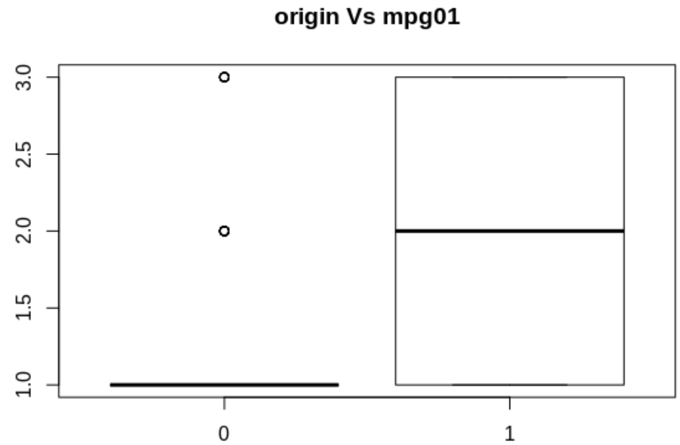

displacement Vs mpg01
400
300
200
100
0
1


weight Vs mpg01
4500
3500
2500
1500
0
1


acceleration Vs mpg01
25
20
15
10
0
1

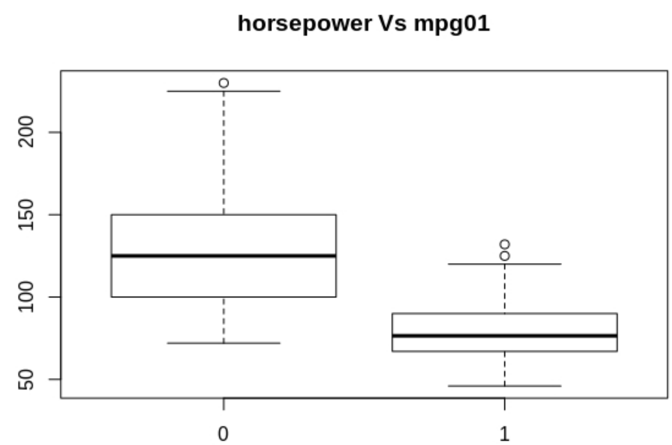

year Vs mpg01
82
80
78
76
74
72
70
0
1





So, in from box plots we see that, cylinders, displacement, weight and horsepower plots are not showing overlaps, so they have roles in predicting mpg01.

Other variables are overlapped, so they unlikely have significant roles in predicting mpg01.



c)

Dividing the data to training and test set with 70% & 30% ratio we get:

```
> (nrow(newauto))*(.7)
[1] 274.4
> newauto.train = newauto[276:nrow(newauto),]
> dim (newauto.train)
[1] 117  10
> newauto.test = newauto [1:275,]
> dim (newauto.test)
[1] 275  10
> mpg01.test = mpg01[1:275]
```

d)

Then applying LDA in cylinders, displacement, weight and horsepower we get:

```
> library(MASS)
> autolda = lda(mpg01~cylinders+displacement+weight+horsepower,data = newauto.train)
> autolda
Call:
lda(mpg01 ~ cylinders + displacement + weight + horsepower, data = newauto.train)

Prior probabilities of groups:
        0         1 
0.1623932 0.8376068 

Group means:
  cylinders displacement   weight horsepower
0  6.736842     258.3158 3506.579  116.31579
1  4.316327     124.8776 2441.918   79.61224

Coefficients of linear discriminants:
                       LD1
cylinders    -0.4841986832
displacement -0.0052953226
weight       -0.0006729959
horsepower   -0.0041634657
```

```
> autolda.pred = predict(autolda,newauto.test)
> autolda.class = autolda.pred$class
> table(autolda.class,mpg01.test)
             mpg01.test
autolda.class   0   1
            0 126   0
            1  51  98
> mean (autolda.class == mpg01.test)
[1] 0.8145455
> ldaerror = (1-mean (autolda.class == mpg01.test))*100
> ldaerror
[1] 18.54545
```

So for LDA we saw that, test error for the model obtained is 18.5%.

e)

Now, performing QDA in the same chunk of training and test data we get:

```
> autoqda = qda(mpg01~cylinders+displacement+weight+horsepower,data = newauto.train)
> autoqda
Call:
qda(mpg01 ~ cylinders + displacement + weight + horsepower, data = newauto.train)

Prior probabilities of groups:
        0         1 
0.1623932 0.8376068 

Group means:
  cylinders displacement   weight horsepower
0  6.736842     258.3158 3506.579  116.31579
1  4.316327     124.8776 2441.918   79.61224
```

```
> autoqda.pred = predict(autoqda,newauto.test)
> autoqda.class = autoqda.pred$class
> table(autoqda.class,mpg01.test)
             mpg01.test
autoqda.class   0   1
            0 128   0
            1  49  98
> mean (autoqda.class == mpg01.test)
[1] 0.8218182
> qdaerror = (1-mean (autoqda.class == mpg01.test))*100
> qdaerror
[1] 17.81818
```

So for LDA we saw that, test error for the model obtained is 17.8%.

f)

Now, performing logistic regression in the same training and test data we get:

```
> autoglm= glm(mpg01~cylinders+displacement+weight+horsepower,data = newauto.train , family = binomial)
> autoglm

Call:  glm(formula = mpg01 ~ cylinders + displacement + weight + horsepower,
    family = binomial, data = newauto.train)

Coefficients:
 (Intercept)     cylinders  displacement        weight    horsepower
   15.291871      0.057354     -0.002994     -0.003515     -0.031887

Degrees of Freedom: 116 Total (i.e. Null);  112 Residual
Null Deviance:      103.8
Residual Deviance: 46.5        AIC: 56.5
```

```
> autoglm.probs=predict(autoglm,newauto.test,type ="response")
> autoglm.pred = rep (0,nrow(newauto.test))
> autoglm.pred[autoglm.probs >.5]=1
> table(autoglm.pred,mpg01.test)
            mpg01.test
autoglm.pred   0   1
           0 117   0
           1  60  98
> mean (autoglm.pred == mpg01.test)
[1] 0.7818182
> lregerror = (1 - mean (autoglm.pred == mpg01.test))*100
> lregerror
[1] 21.81818
```

So for logistic regression we saw that, test error for the model obtained is 21.82%.

g)

Now, performing KNN(K=1) in the same training and test data we get:

```
> library (class )
> train.X = cbind (cylinders,displacement,weight,horsepower) [276:nrow(newauto),]
> test.X = cbind (cylinders,displacement,weight,horsepower) [1:275,]
> train.mpg01 = mpg01[276:nrow(newauto)]
> test.mpg01 = mpg01[1:275]
> set.seed (1)
> autoknn.pred = knn (train.X, test.X,train.mpg01,k =1)
```

```
> table(autoknn.pred, test.mpg01)
            test.mpg01
autoknn.pred   0   1
           0 126   1
           1  51  97
> mean(autoknn.pred == test.mpg01)
[1] 0.8109091
> knnerror = (1 - mean(autoknn.pred == test.mpg01))*100
> knnerror
[1] 18.90909
```

So for KNN(K=1) we saw that, test error for the model obtained is 18.91%.

Now changing the value of K we get:

```
> library (class )
> train.X = cbind (cylinders,displacement,weight,horsepower) [276:nrow(newauto),]
> test.X = cbind (cylinders,displacement,weight,horsepower) [1:275,]
> train.mpg01 = mpg01[276:nrow(newauto)]
> test.mpg01 = mpg01[1:275]
> set.seed (1)
> autoknn.pred = knn (train.X, test.X,train.mpg01,k=5)
> table(autoknn.pred, test.mpg01)
            test.mpg01
autoknn.pred   0   1
           0 123   0
           1  54  98
> mean(autoknn.pred == test.mpg01)
[1] 0.8036364
> knnerror = (1 - mean(autoknn.pred == test.mpg01))*100
> knnerror
[1] 19.63636
```

So for KNN(K=5) we saw that, test error for the model obtained is 19.64%.

```
> library (class )
> train.X = cbind (cylinders,displacement,weight,horsepower) [276:nrow(newauto),]
> test.X = cbind (cylinders,displacement,weight,horsepower) [1:275,]
> train.mpg01 = mpg01[276:nrow(newauto)]
> test.mpg01 = mpg01[1:275]
> set.seed (1)
> autoknn.pred = knn (train.X, test.X,train.mpg01,k=10)
> table(autoknn.pred, test.mpg01)
            test.mpg01
autoknn.pred   0   1
           0 124   0
           1  53  98
> mean(autoknn.pred == test.mpg01)
[1] 0.8072727
> knnerror = (1 - mean(autoknn.pred == test.mpg01))*100
> knnerror
[1] 19.27273
```

So for KNN(K=10) we saw that, test error for the model obtained is 19.27%.

```
> library (class )
> train.X = cbind (cylinders,displacement,weight,horsepower) [276:nrow(newauto),]
> test.X = cbind (cylinders,displacement,weight,horsepower) [1:275,]
> train.mpg01 = mpg01[276:nrow(newauto)]
> test.mpg01 = mpg01[1:275]
> set.seed (1)
> autoknn.pred = knn (train.X, test.X,train.mpg01,k=20)
> table(autoknn.pred, test.mpg01)
            test.mpg01
autoknn.pred   0   1
           0 116   0
           1  61  98
> mean(autoknn.pred == test.mpg01)
[1] 0.7781818
> knnerror = (1 - mean(autoknn.pred == test.mpg01))*100
> knnerror
[1] 22.18182
```

So for KNN(K=20) we saw that, test error for the model obtained is 22.18%.

```
> library (class )
> train.X = cbind (cylinders,displacement,weight,horsepower) [276:nrow(newauto),]
> test.X = cbind (cylinders,displacement,weight,horsepower) [1:275,]
> train.mpg01 = mpg01[276:nrow(newauto)]
> test.mpg01 = mpg01[1:275]
> set.seed (1)
> autoknn.pred = knn (train.X, test.X,train.mpg01,k=15)
> table(autoknn.pred, test.mpg01)
            test.mpg01
autoknn.pred   0   1
           0 127   0
           1  50  98
> mean(autoknn.pred == test.mpg01)
[1] 0.8181818
> knnerror = (1 - mean(autoknn.pred == test.mpg01))*100
> knnerror
[1] 18.18182
```

So for KNN(K=15) we saw that, test error for the model obtained is 18.18%.

We experimented with multiple values and saw that, K = 15 perform best on this data set.