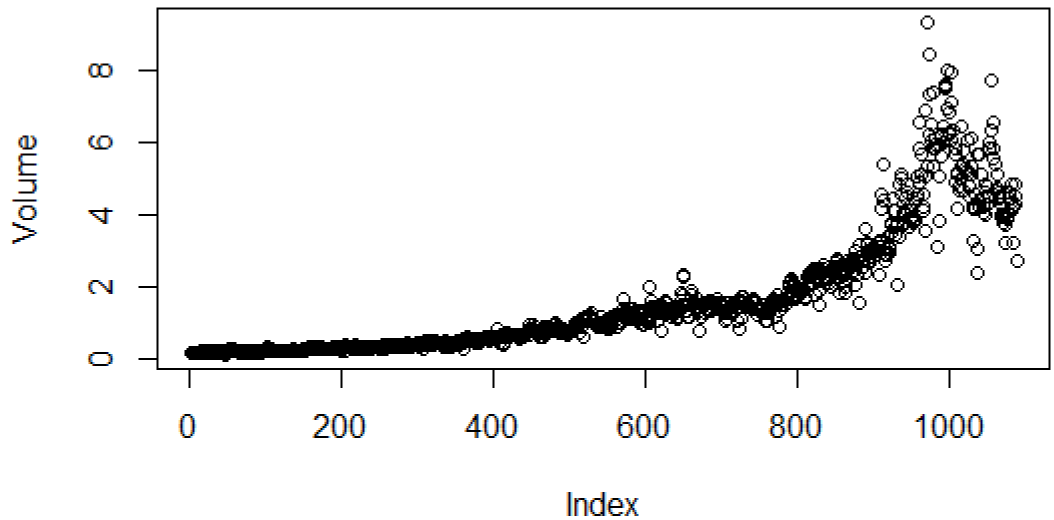
1 a)

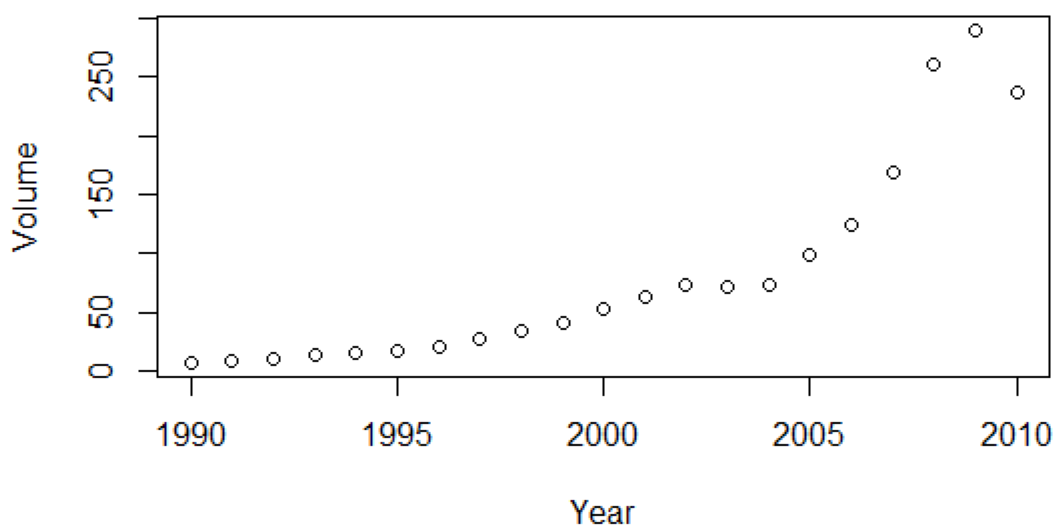
```
> str(weekly)
'data.frame':	1089 obs. of  9 variables:
 $ Year     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
 $ Lag1     : num  0.816 -0.27 -2.576 3.514 0.712 ...
 $ Lag2     : num  1.572 0.816 -0.27 -2.576 3.514 ...
 $ Lag3     : num  -3.936 1.572 0.816 -0.27 -2.576 ...
 $ Lag4     : num  -0.229 -3.936 1.572 0.816 -0.27 ...
 $ Lag5     : num  -3.484 -0.229 -3.936 1.572 0.816 ...
 $ Volume   : num  0.155 0.149 0.16 0.162 0.154 ...
 $ Today    : num  -0.27 -2.576 3.514 0.712 1.178 ...
 $ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
> dim(weekly)
[1] 1089    9
> summary(weekly)
      Year           Lag1               Lag2               Lag3               Lag4         
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580   1st Qu.: -1.1580  
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410   Median :  0.2380  
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472   Mean   :  0.1458  
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
      Lag5              Volume            Today          Direction 
 Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950   Down:484  
 1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540   Up  :605  
 Median :  0.2340   Median :1.00268   Median :  0.2410             
 Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499             
 3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050             
 Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260             
> 
```

```
> cor(weekly[,-9])
              Year         Lag1        Lag2        Lag3         Lag4         Lag5      Volume
Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923 -0.030519101  0.84194162
Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096 -0.06495131
Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482 -0.08551314
Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175 -0.06928771
Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027 -0.06107462
Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000 -0.05851741
Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414  1.00000000
Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698 -0.03307778
              Today
Year   -0.032459894
Lag1   -0.075031842
Lag2    0.059166717
Lag3   -0.071243639
Lag4   -0.007825873
Lag5    0.011012698
Volume -0.033077783
Today   1.000000000
> 
```

The correlation matrix suggests that there is no significant relationship/correlation between Lag1,2,3,4,5 variables, but there is a 0.842 correlation between 'Year' and 'Volume' variable. Hence plot between year and volume is plotted.



PLOT(VOLUME)



YEAR WISE MEAN OF VOLUME

1 b)

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

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
```

Predictor Lag 2 seems to be significant with a p value less than 0.05.

1c)

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

log.prob=predict(logistic.fit,type="response")
log.prob

logistic.pred=rep("Down",1089)
logistic.pred[log.prob>0.5]="Up"

table(logistic.pred,Direction)
```

```
             Direction
logistic.pred Down  Up
         Down   54  48
         Up    430 557
> 
```

Accuracy= (True Positive+True Negative)/(Negative+Positive) = (557+54)/(430+557+48+54) =0.561=56.1%

Error Rate= 1- Accuracy = 0.4389=43.89%

Sensitivity =True Positive / Positive = 557/(557+48)= 0.92=92%

Specificity=True Negative/Negative=54/(54+430)=0.1115=11.15%

False Positive Rate= 1-Specificity= 88.84% {The confusion matrix suggests that the model predicts the Direction to be Up 88.84% when it actually is Down according to the given data.]

The model accurately predicts 92% of the times when the market goes Up and only 11.15% when the market goes Down. The overall accuracy though is 56.1%

The error rate does not represent the performance of logistic regression in prediction as the error rate calculated here is the training error rate as the data being compared for accuracy , error and other metrics is between the model predicted using the training data using which the logistic model was made and not the test data.

1)d)

```
Console ~/
> train=(Year<2009)
> Week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
>
> log1.fit=glm(Direction~Lag2,data=Weekly,family=binomial,subset=train)
> summary(log1.fit)

Call:
glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
    subset = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.536  -1.264   1.021   1.091   1.368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1350.5  on 983  degrees of freedom
AIC: 1354.5

Number of Fisher Scoring iterations: 4

> log1.prob=predict(log1.fit,Week.200910,type="response")
>
>
> log1.pred=rep("Down",length(log1.prob))
> log1.pred[log1.prob>0.5]="Up"
>
> table(log1.pred,Direction.200910)
         Direction.200910
log1.pred Down Up
```

```
> table(log1.pred,Direction.200910)
         Direction.200910
log1.pred Down Up
     Down    9  5
     Up     34 56
> mean(log1.pred==Direction.200910)
[1] 0.625
>
```

Accuracy= (True Positive+True Negative)/(Negative+Positive) = (56+9)/(56+9+5+34) =0.625=62.5%

Error Rate= 1- Accuracy = 0.375=37.5%

Sensitivity =True Positive / Positive = 56/(56+5)= 0.918=91.8%

Specificity=True Negative/Negative=9/(9+34)=0.209=20.93%

1)e) LDA

```
> train=(Year<2009)
> Week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
>
> lda1.fit=lda(Direction~Lag2,data=Weekly,subset=train)
> #summary(lda1.fit)
> lda1.prob=predict(lda1.fit,Week.200910)
> lda1.class=lda1.prob$class
>
> table(lda1.class,Direction.200910)
          Direction.200910
lda1.class Down Up
      Down    9  5
      Up     34 56
> mean(lda1.class==Direction.200910)
[1] 0.625
>
```

Accuracy= (True Positive+True Negative)/(Negative+Positive) = (56+9)/(56+9+5+34) =0.625=62.5%

Error Rate= 1- Accuracy = 0.375=37.5%

Sensitivity =True Positive / Positive = 56/(56+5)= 0.918=91.8%

Specificity=True Negative/Negative=9/(9+34)=0.209=20.93%

1)f) QDA

```
> train=(Year<2009)
> week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
> lda1.fit=qda(Direction~Lag2,data=Weekly,subset=train)
> lda1.prob=predict(lda1.fit,week.200910)
> lda1.class=lda1.prob$class
> table(lda1.class,Direction.200910)
          Direction.200910
lda1.class Down Up
      Down    0  0
      Up     43 61
> mean(lda1.class==Direction.200910)
[1] 0.5865385
>
```

Accuracy= (True Positive+True Negative)/(Negative+Positive) = (61)/(61+43) =0.5865=58.65%

Error Rate= 1- Accuracy = 0.4134=41.34%

Sensitivity =True Positive / Positive = 61/(61+0)= 1=100%

Specificity=True Negative/Negative=0/(0+43)=0=0%

1)g) KNN

```
> library(class)
> train=(Year<2009)
> Direction.200910=Direction[!train]
> train.X=cbind(Lag2)[train,]
> test.X=cbind(Lag2)[!train,]
> train.Direction=Direction[train]
> set.seed(1)
> knn.pred=knn(data.frame(train.X),data.frame(test.X),train.Direction,k=1)
>
> table(knn.pred,Direction.200910)
        Direction.200910
knn.pred Down Up
    Down   21 30
    Up     22 31
> mean(knn.pred==Direction.200910)
[1] 0.5
>
```

Accuracy= (True Positive+True Negative)/(Negative+Positive) = (31+21)/(31+21+30+22) =0.5=50%

Error Rate= 1- Accuracy = 0.5=50%

Sensitivity =True Positive / Positive = 31/(31+30)= 0.5082=50.82%

Specificity=True Negative/Negative=21/(21+22)=0.4883=48.83%

1)h)

Model	Accuracy(%)	Error Rate(%)
Logistic	62.5	37.5
LDA	62.5	37.5
QDA	58.65	41.34
KNN	50	50

According to the above table, Logistic and the LDA models have the highest accuracy and the lowest error rate and seem to be the methods with the best results for this data.

1) i)

#LOGISTIC1

```
Console ~/
[1] 0.625
> train=(Year<2009)
> Week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
>
> log1.fit=glm(Direction~Lag2+I(Lag2^2),data=Weekly,family=binomial,subset=train)
> summary(log1.fit)

Call:
glm(formula = Direction ~ Lag2 + I(Lag2^2), family = binomial,
    data = Weekly, subset = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.791  -1.253   1.005   1.100   1.196

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.179006   0.068357   2.619  0.00883 **
Lag2        0.064920   0.029734   2.183  0.02901 *
I(Lag2^2)   0.004713   0.004569   1.031  0.30236
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1349.4  on 982  degrees of freedom
AIC: 1355.4

Number of Fisher Scoring iterations: 4

> log1.prob=predict(log1.fit,Week.200910,type="response")
>
>
> log1.pred=rep("Down",length(log1.prob))
> log1.pred[log1.prob>0.5]="Up"
>
> table(log1.pred,Direction.200910)
         Direction.200910
```

```
Console ~/
>
> table(log1.pred,Direction.200910)
         Direction.200910
log1.pred Down Up
     Down    8  4
     Up     35 57
> mean(log1.pred==Direction.200910)
[1] 0.625
>
```

#LOGISTIC2

```
Console ~/
[1] 0.5
> #LOGISTIC2
> train=(Year<2009)
> Week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
> log1.fit=glm(Direction~Lag2:Lag1,data=Weekly,family=binomial,subset=train)
> summary(log1.fit)

Call:
glm(formula = Direction ~ Lag2:Lag1, family = binomial, data = Weekly,
    subset = train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.368  -1.269   1.077   1.089   1.353

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.21333    0.06421   3.322 0.000893 ***
Lag2:Lag1    0.00717    0.00697   1.029 0.303649
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1353.6  on 983  degrees of freedom
AIC: 1357.6

Number of Fisher Scoring iterations: 4

> log1.prob=predict(log1.fit,Week.200910,type="response")
> log1.pred=rep("Down",length(log1.prob))
> log1.pred[log1.prob>0.5]="Up"
> table(log1.pred,Direction.200910)

         Direction.200910
log1.pred Down Up
     Down    1  1
     Up     42 60
> mean(log1.pred==Direction.200910)
[1] 0.5865385
>
```

Of the above 2,
The basic model with Direction calculated using

1) **Direction~Lag2 (62.5%)**
2) Lag2+(Lag^2) (62.5%) **–** Cannot use this as (Lag2^2) does not have significance wrt p values
3) Lag2:Lag1 (58.65%)

As the basic model (1) itself is having the higher Accuracy and hence a high error rate, it can be chosen over model(2) as the accuracy is the same for additional terms and effort. With respect to interaction Lag2:Lag1 interaction gives the maximum accuracy of all the interaction possibilities.

Lag 2 is the central terms in all the models considered as it is the most significant variable in the basic model.

#LDA

```
Console ~/
> week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
>
> lda1.fit=lda(Direction~Lag2:Lag5,Data=Weekly,subset=train)
> #summary(lda1.fit)
> lda1.prob=predict(lda1.fit,week.200910)
>
>
> lda1.class=lda1.prob$class
>
> table(lda1.class,Direction.200910)
          Direction.200910
lda1.class Down Up
      Down    0  0
      Up     43 61
> mean(lda1.class==Direction.200910)
[1] 0.5865385
> train=(Year<2009)
> week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
>
> lda1.fit=lda(Direction~I(Lag2^2),Data=Weekly,subset=train)
> #summary(lda1.fit)
> lda1.prob=predict(lda1.fit,week.200910)
>
>
> lda1.class=lda1.prob$class
>
> table(lda1.class,Direction.200910)
          Direction.200910
lda1.class Down Up
      Down    0  0
      Up     43 61
> mean(lda1.class==Direction.200910)
[1] 0.5865385
>
```

Adding interaction terms and transformations only makes the basic model worse(at best 58.65%). Hence,

Direction ~ Lag2 is the best model for the given data with an error rate of 37.5 and an accuracy of 62.5%.

#QDA

```
Console ~/
> #QDA
> train=(Year<2009)
> Week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
>
> lda1.fit=qda(Direction~Lag2+I(Lag2^2),data=Weekly,subset=train)
> #summary(lda1.fit)
> lda1.prob=predict(lda1.fit,Week.200910)
>
>
> lda1.class=lda1.prob$class
>
> table(lda1.class,Direction.200910)
          Direction.200910
lda1.class Down Up
      Down    7  3
      Up     36 58
> mean(lda1.class==Direction.200910)
[1] 0.625
>
```

Taking a square transformation increases the accuracy from 58.65%(basic model) to 62.5%. Hence

Direction~Lag2+(Lag2^2)

Is a good model for QDA

#KNN

```
Console ~/
> library(class)
> train=(Year<2009)
> Direction.200910=Direction[!train]
> train.X=cbind(Lag2)[train,]
> test.X=cbind(Lag2)[!train,]
> train.Direction=Direction[train]
> maxk=0
> max_K=1
> mean_K=0
> for (i in 1:length(train.X))
+   {
+ set.seed(1)
+ knn.pred=knn(data.frame(train.X),data.frame(test.X),train.Direction,k=i)
+ mean_K=mean(knn.pred==Direction.200910)
+ if(mean_K>maxk)
+  {    maxk=mean_K
+       max_K=i
+ }
+ }
Error in knn(data.frame(train.X), data.frame(test.X), train.Direction,  :
  too many ties in knn
> print(maxk)
[1] 0.6442308
> print(max_K)
[1] 152
> knn.pred=knn(data.frame(train.X),data.frame(test.X),train.Direction,k=max_K)
>
> table(knn.pred,Direction.200910)
        Direction.200910
knn.pred Down Up
    Down    9  3
    Up     34 58
> mean(knn.pred==Direction.200910)
[1] 0.6442308
>
```

It can be seen on running a for loop for all values of K possible till the length of the train data set, it can be seen that with a K value of 152, accuracy obtained which is the maximum is 0.644=64.4%

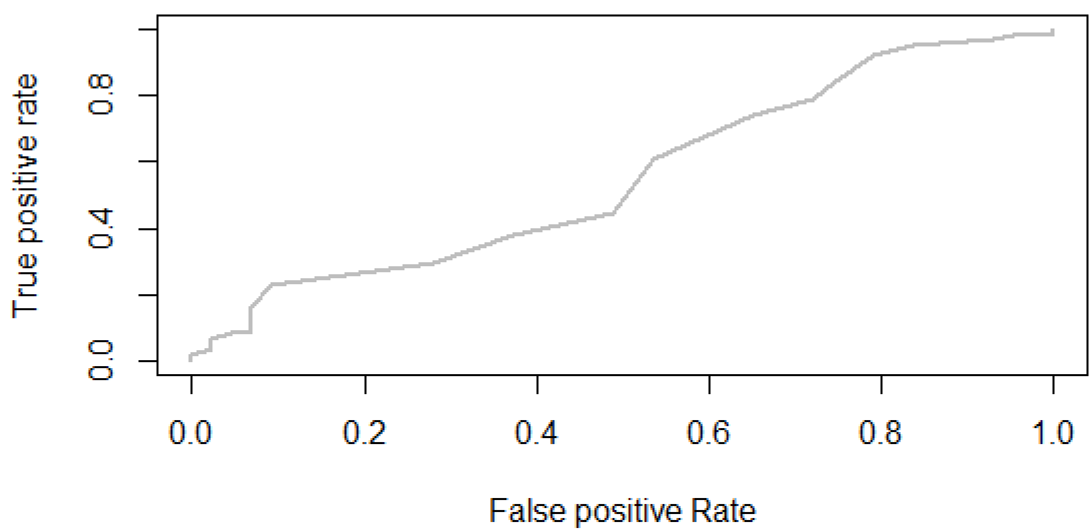
Though K values of around 10 are preferred because, the test error rate might be really high at such high values(152) of K in general, though here it turns out better here.

---

2)

The best model chosen in Question 1(i) is Direction~Lag2 .

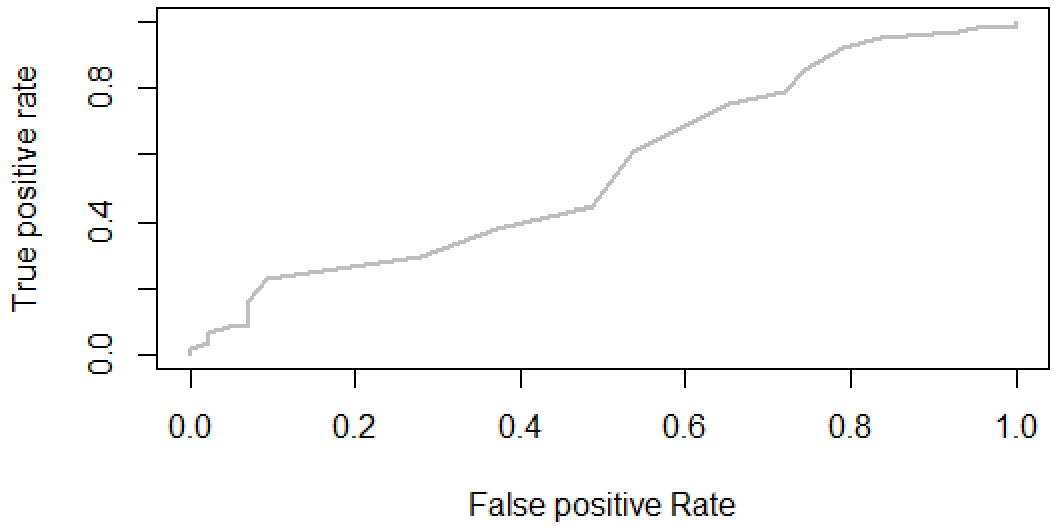
```
Console ~/
> train=(Year<2009)
> Week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
> LR.fit=glm(Direction~Lag2,data=Weekly,family=binomial,subset=train)
> #summary(log1.fit)
> LR.pred=predict(LR.fit,Week.200910,type="response")
>
> roc.curve=function(s,print=FALSE)
+ {
+   Ps=(LR.pred>s)*1
+   FP=sum((Ps==1)*(Direction.200910=="Down"))/sum(Direction.200910=="Down")
+   TP=sum((Ps==1)*(Direction.200910=="Up"))/sum(Direction.200910=="Up")
+   if(print==TRUE){
+     print(table(Observed=Direction.200910,Predicted=Ps))
+   }
+   vect=c(FP,TP)
+   names(vect)=c("FPR","TPR")
+   return(vect)
+ }
> threshold=0.5
> roc.curve(threshold,print=TRUE)
        Predicted
Observed  0  1
    Down  9 34
    Up    5 56
      FPR       TPR
0.7906977 0.9180328
>
>
> ROC.curve=Vectorize(roc.curve)
> M.ROC=ROC.curve(seq(0,1,by=0.01))
> plot(M.ROC[1,],M.ROC[2,],col="grey",lwd=2,type="l",xlab="False positive Rate",ylab="True
positive rate")
>
```



The ROC Curve graces along the 45degree line(AUC around 0.5) and slightly above , hence the model can be assumed to be more or less like a random guessing model. The model is not a good one.

Console ~/

```
> train=(Year<2009)
> Week.200910=Weekly[!train,]
> Direction.200910=Direction[!train]
> LR.fit=lda(Direction~Lag2,Data=Weekly,subset=train)
> #summary(lda1.fit)
> lda1.prob=predict(LR.fit,Week.200910)
> LR.pred=lda1.prob$posterior[,2]
>
>
>
> roc.curve=function(s,print=FALSE)
+ {
+   Ps=(LR.pred>s)*1
+   FP=sum((Ps==1)*(Direction.200910=="Down"))/sum(Direction.200910=="Down")
+   TP=sum((Ps==1)*(Direction.200910=="Up"))/sum(Direction.200910=="Up")
+   if(print==TRUE){
+     print(table(Observed=Direction.200910,Predicted=Ps))
+   }
+   vect=c(FP,TP)
+   names(vect)=c("FPR","TPR")
+   return(vect)
+ }
> threshold=0.5
> roc.curve(threshold,print=TRUE)
        Predicted
Observed  0  1
    Down  9 34
    Up    5 56
      FPR       TPR
0.7906977 0.9180328
>
>
> ROC.curve=Vectorize(roc.curve)
> M.ROC=ROC.curve(seq(0,1,by=0.01))
> plot(x, y, ...)C[1,],M.ROC[2,],col="grey",lwd=2,type="l",xlab="False positive Rate",ylab="True po
sitive rate")
> plot(M.ROC[1,],M.ROC[2,],col="grey",lwd=2,type="l",xlab="False positive Rate",ylab="True
  positive rate")
```



The ROC Curve graces along the 45degree line and slightly above , hence the model can be assumed to be more or less like a random guessing model. The model is not a good one.

---

3)

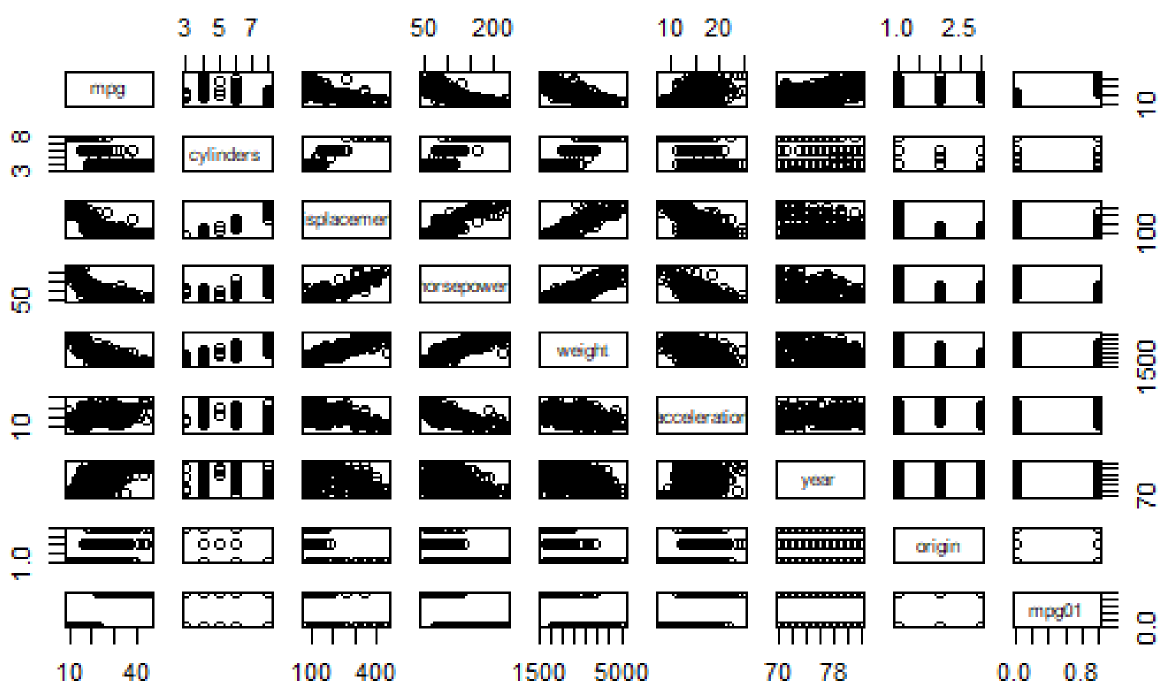
a)

```
Console ~/
> attach(Auto)
> mpg01=rep(0,length(mpg))
> mpg01[mpg>median(mpg)]=1
> df=data.frame(Auto,mpg01)
>
```

b)

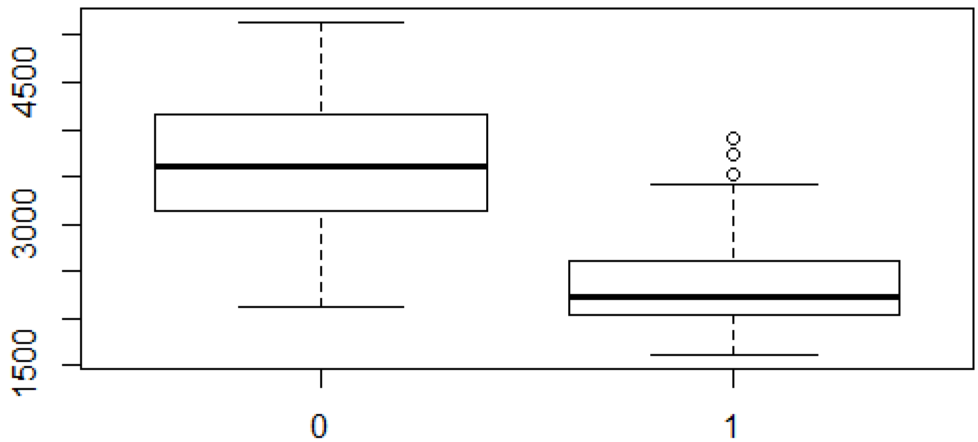
```
Console ~/
> cor(df[,-9])
                    mpg  cylinders displacement horsepower     weight acceleration
mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442    0.4233285
cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273   -0.5046834
displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944   -0.5438005
horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377   -0.6891955
weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000   -0.4168392
acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392    1.0000000
year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199    0.2903161
origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054    0.2127458
mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566    0.3468215
                   year     origin      mpg01
mpg           0.5805410  0.5652088  0.8369392
cylinders    -0.3456474 -0.5689316 -0.7591939
displacement -0.3698552 -0.6145351 -0.7534766
horsepower   -0.4163615 -0.4551715 -0.6670526
weight       -0.3091199 -0.5850054 -0.7577566
acceleration  0.2903161  0.2127458  0.3468215
year          1.0000000  0.1815277  0.4299042
origin        0.1815277  1.0000000  0.5136984
mpg01         0.4299042  0.5136984  1.0000000
> 
```

pairs(df)

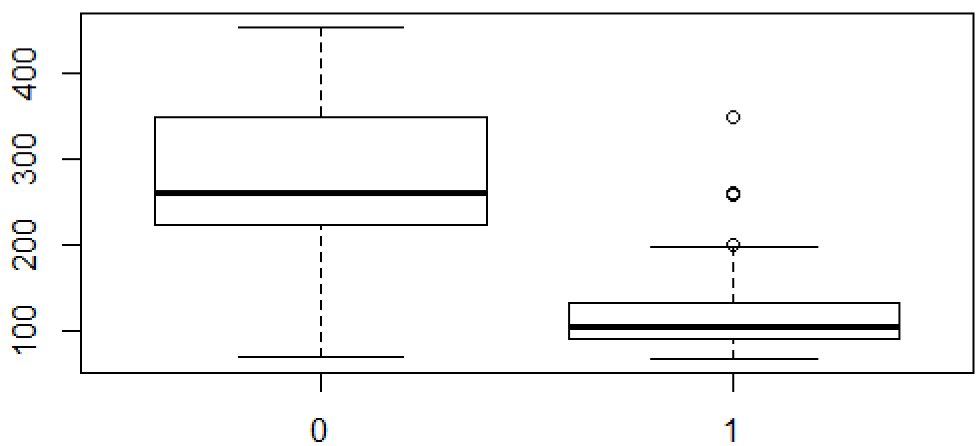


Mpg01 has some relation with displacement, horsepower, weight and acceleration according to the above boxplot.

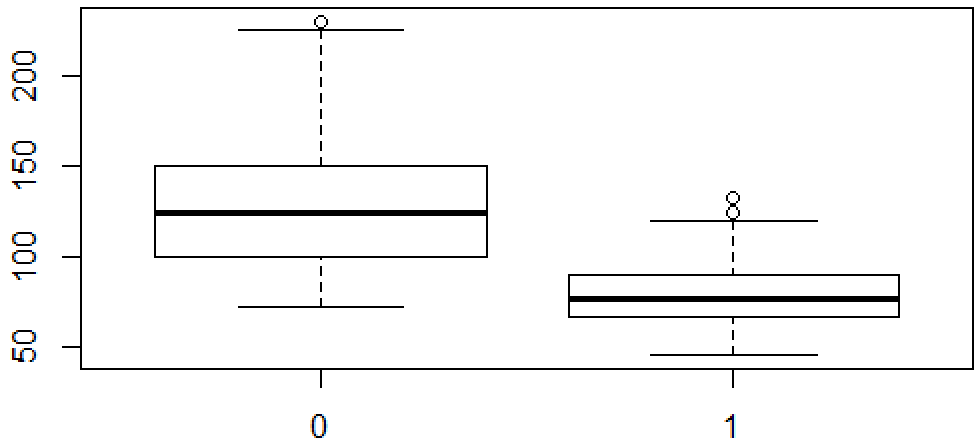
Weight vs mpg01



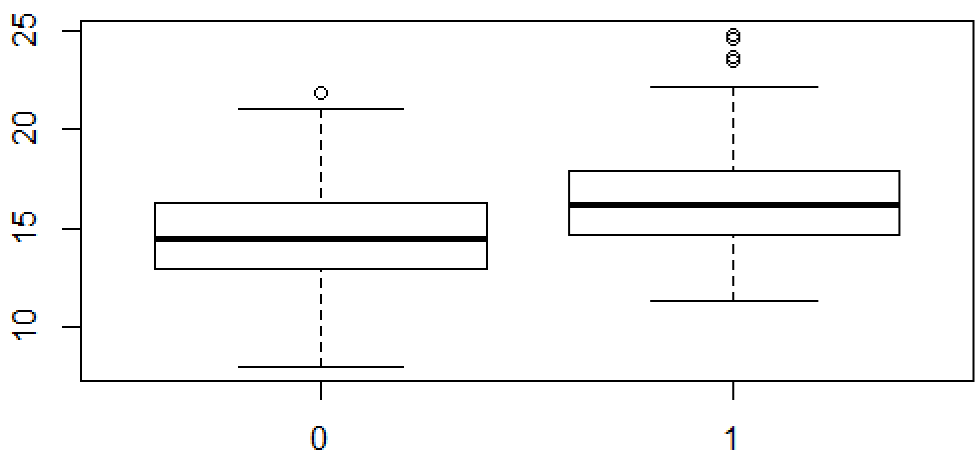
Displacement vs mpg01

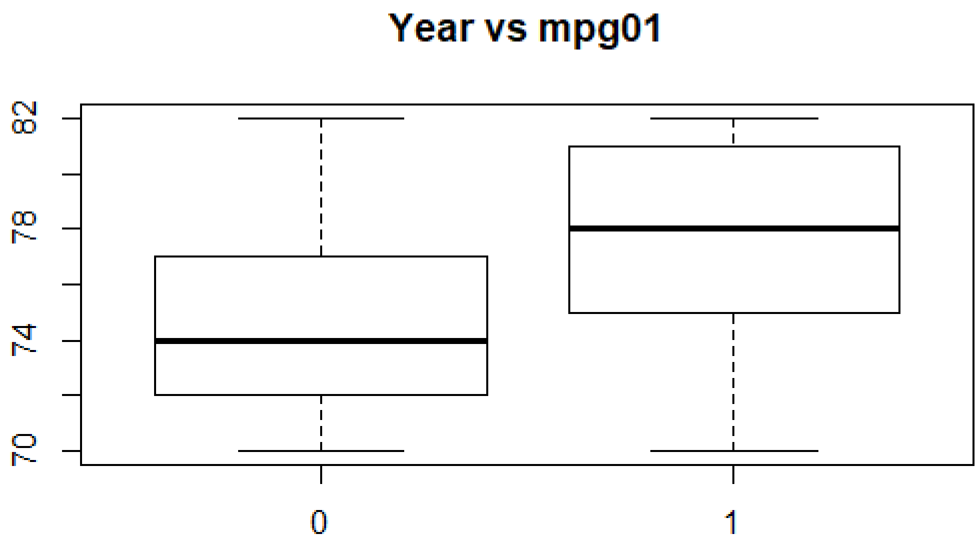


Hosepower vs mpg01

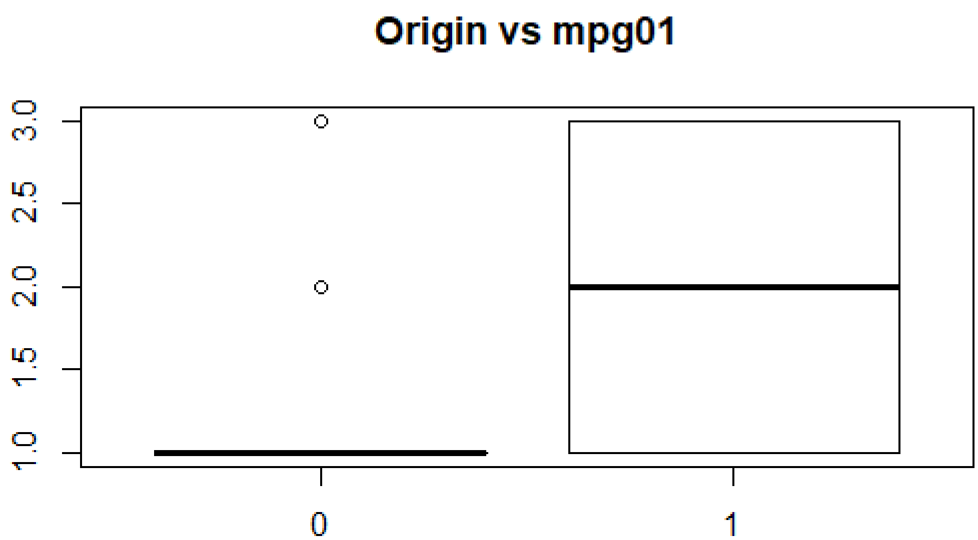


Acceleration vs mpg01

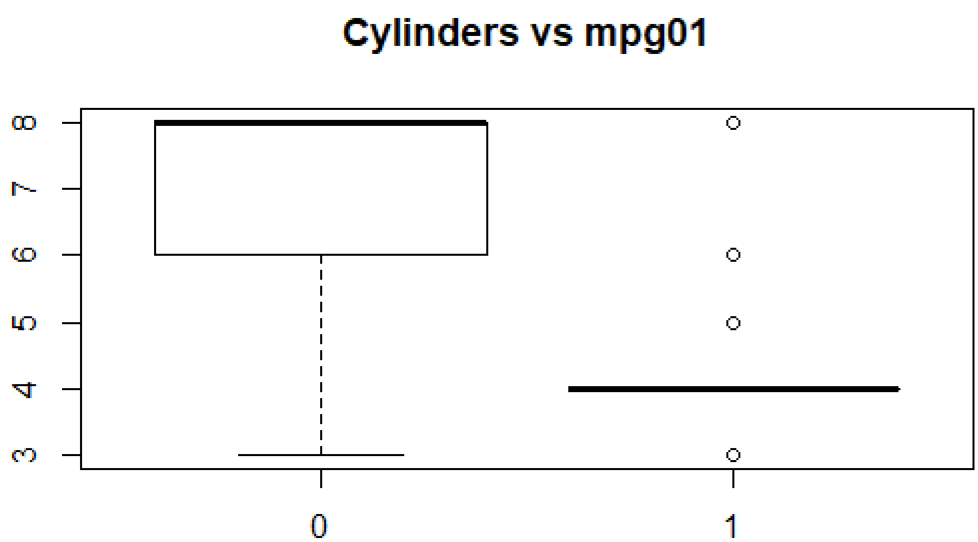




Year vs mpg01
82
78
74
70
0
1



Origin vs mpg01
3.0
2.5
2.0
1.5
1.0
0
1



It can be concluded that there is some relationship between mpg01 and weight, displacement, horsepower, cylinders.

c)

#According to the summary, 75 percentile of data is above year=79, hence assuming 75 % of

data to bet rain data and 25% data as test data, data is split accordingly.

```
Console ~/
> train.dat=year<79
> test.dat=year>=79
>
```

train.dat is considered as the train data

test.dat is considered as the test data

d)

LDA

```
> Auto_test=df[test.dat,]
> mpg01_test=mpg01[test.dat]
> lda.fit=lda(mpg01~weight+displacement+horsepower+cylinders,Data=df,subset=train.dat)
> lda1.prob=predict(lda.fit,Auto_test)
> lda1.class=lda1.prob$class
> table(lda1.class,mpg01_test)
          mpg01_test
lda1.class  0  1
         0 17 16
         1  1 80
> mean(lda1.class==mpg01_test)
[1] 0.8508772
>
```

Error Rate: 14.92%

e)

QDA

```
> Auto_test=df[test.dat,]
> mpg01_test=mpg01[test.dat]
> qda.fit=qda(mpg01~weight+displacement+horsepower+cylinders,Data=df,subset=train.dat)
> qda1.prob=predict(qda.fit,Auto_test)
> qda1.class=lda1.prob$class
> table(qda1.class,mpg01_test)
          mpg01_test
qda1.class  0  1
         0 17 19
         1  1 77
> mean(qda1.class==mpg01_test)
[1] 0.8245614
>
```

Error Rate: 17.56%

f)

Logistic

```
Console ~/
[1] 0.4473684
> library(MASS)
> Auto_test=df[test.dat,]
> mpg01_test=mpg01[test.dat]
> log.fit=glm(mpg01~weight+displacement+horsepower+cylinders,data=df,family=binomial,subset=t
rain.dat)
> summary(log.fit)

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

Deviance Residuals: 
     Min        1Q    Median        3Q       Max  
-2.46868  -0.12598  -0.00411   0.24423   2.17907  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) 15.854208   2.879739   5.505 3.68e-08 ***
weight      -0.002937   0.001146  -2.563   0.0104 *  
displacement -0.006911   0.014889  -0.464   0.6425    
horsepower  -0.047461   0.022930  -2.070   0.0385 *  
cylinders   -0.620741   0.643450  -0.965   0.3347    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 363.21  on 277  degrees of freedom
Residual deviance: 106.58  on 273  degrees of freedom
AIC: 116.58

Number of Fisher Scoring iterations: 8

> log1.prob=predict(log.fit,Auto_test,type="response")
> log1.pred=rep(0,length(log1.prob))
> log1.pred[log1.prob>0.5]=1
> table(log1.pred,mpg01_test)
```

```
Console ~/
> table(log1.pred,mpg01_test)
         mpg01_test
log1.pred  0  1
        0 18 22
        1  0 74
> mean(log1.pred==mpg01_test)
[1] 0.8070175
> 
```

Error Rate=19.3%

g)

KNN

Console ~/

```
> library(class)
> Auto_test=df[test.dat,]
> mpg01_test=mpg01[test.dat]
> train.X=cbind(weight,displacement,horsepower,cylinders)[train.dat,]
> test.X=cbind(weight,displacement,horsepower,cylinders)[test.dat,]
> train.mpg01=mpg01[train.dat]
>
>   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=1)
>   table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 16 24
       1  2 72
>   mean(knn.pred==mpg01_test)
[1] 0.7719298
>   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=10)
>   table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 24
       1  0 72
>   mean(knn.pred==mpg01_test)
[1] 0.7894737
>   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=20)
>   table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 21
       1  0 75
>   mean(knn.pred==mpg01_test)
[1] 0.8157895
```

Console ~/

```
>   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=30)
>   table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 20
       1  0 76
>   mean(knn.pred==mpg01_test)
[1] 0.8245614
>   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=40)
>   table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 24
       1  0 72
>   mean(knn.pred==mpg01_test)
[1] 0.7894737
>   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=50)
>   table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 29
       1  0 67
>   mean(knn.pred==mpg01_test)
[1] 0.745614
>   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=100)
>   table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 25
       1  0 71
>   mean(knn.pred==mpg01_test)
[1] 0.7807018
```

```
Console ~/
> knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=150)
> table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 22
       1  0 74
> mean(knn.pred==mpg01_test)
[1] 0.8070175
> knn.pred=knn(data.frame(train.X),data.frame(test.X),train.mpg01,k=200)
> table(knn.pred,mpg01_test)
        mpg01_test
knn.pred  0  1
       0 18 63
       1  0 33
> mean(knn.pred==mpg01_test)
[1] 0.4473684
>
```

K Value	Error Rate
1	22.8%
10	19.3%
20	18.42%
30	17.54%
40	21.05%
50	25.44%
100	21.93%
150	19.3%
200	55.3%

K Value of 30 seems to perform better for this KNN model.