Elliott Sperin

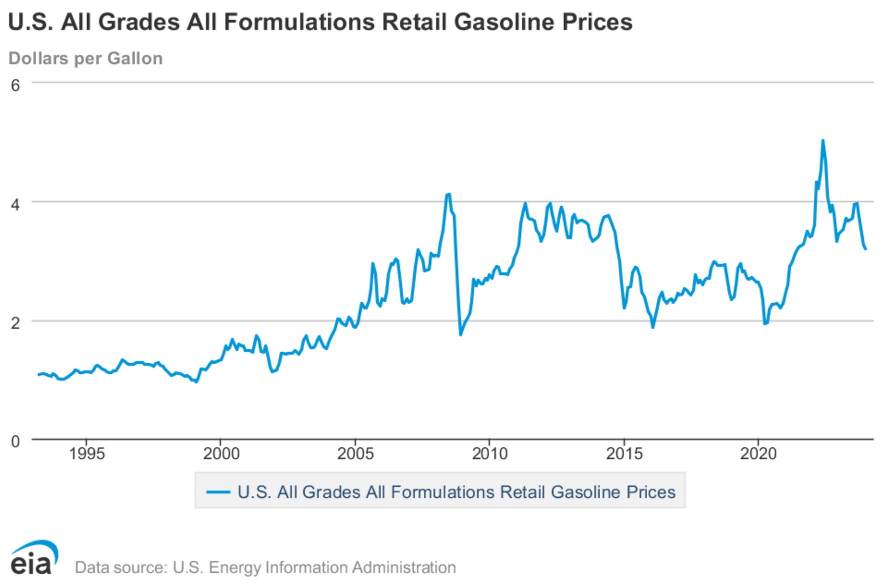
High or Low MPG Predictive Analysis

(i) Introduction:

Program Statement: Use the “Auto” dataset to predict whether a car get low or high gas mileage based on seven car attributes such as cylinders, displacement, horsepower, weight, acceleration, model year and origin, by selecting the best variables and using classification models to implement the prediction.

Gas is crucial for the modern world and when gas prices fluctuate it has a tremendous impact on the day-to-day lives of people. The cost of a person’s daily commute and the price of goods will increase with increasing gas prices. Since gas prices are increasing (figure 1), finding ways to mitigate the impact of rising gas prices is important for the government of the US and for the private sector. A practical way to mitigate high fuel prices is by making vehicles more fuel efficient. In 1993 a dataset was created by R. Quinlan that captured miles per gallon, number of cylinders, displacement, horsepower, weight, acceleration, model year and origin. This dataset has been analyzed numerous times to see the impact that each variable has on miles per gallon. In our analysis we will create predictive models that predict MPG based on key variables. Our analysis will be useful in determining which variables have the greatest impact on MPG. The results of this analysis will be particularly useful for passing legislation or engineering fuel efficient vehicles.

Figure 1: US gas prices from April 1993 to January 2024



(ii) Exploratory Data Analysis.

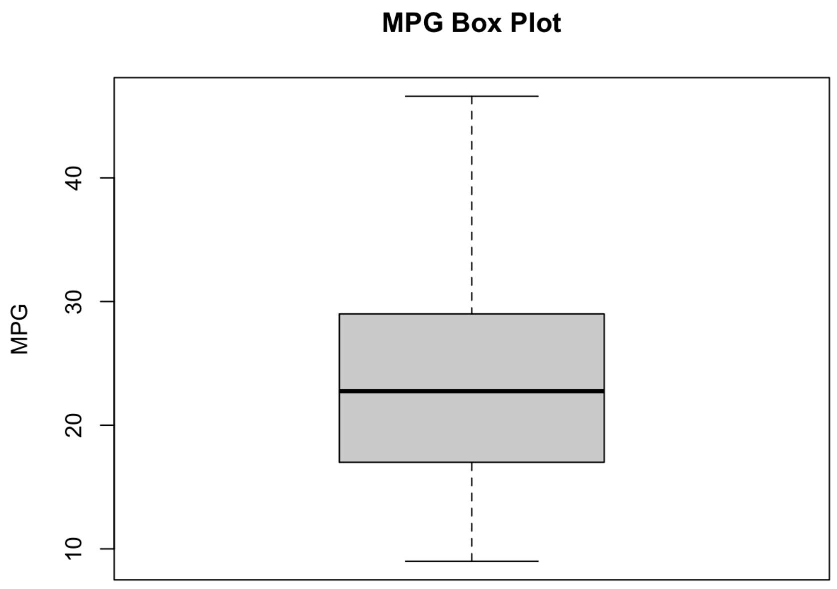
After removing rows with NA values our dataset includes 392 observations among 8 variables. The dataset includes the following variables:

MPG	**Cylinders**	**Displacement**	**Horsepower**
Miles per Gallon	Number of Cylinders	Volume of Cylinders	How quickly force is produced from engine
Weight	**Acceleration**	**Year**	**Origin**
Weight in pounds	Capacity to gain speed in a short time	Year the vehicle was produced	Place of vehicle origin. Represented as discrete numbers.

For this analysis MPG is the dependent/response variable and all other factors will be the independent/predictor variables.

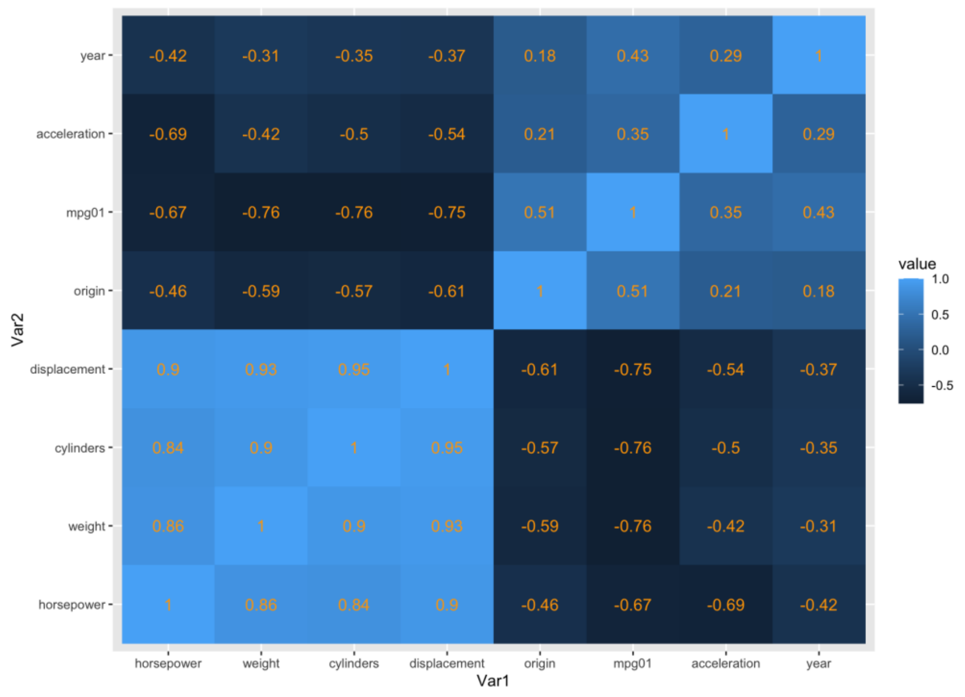
Outliers: Considering the box and whisker plot (figure 2), there are no outliers in the response variable.

Figure 2: MPG Box and Whisker Plot



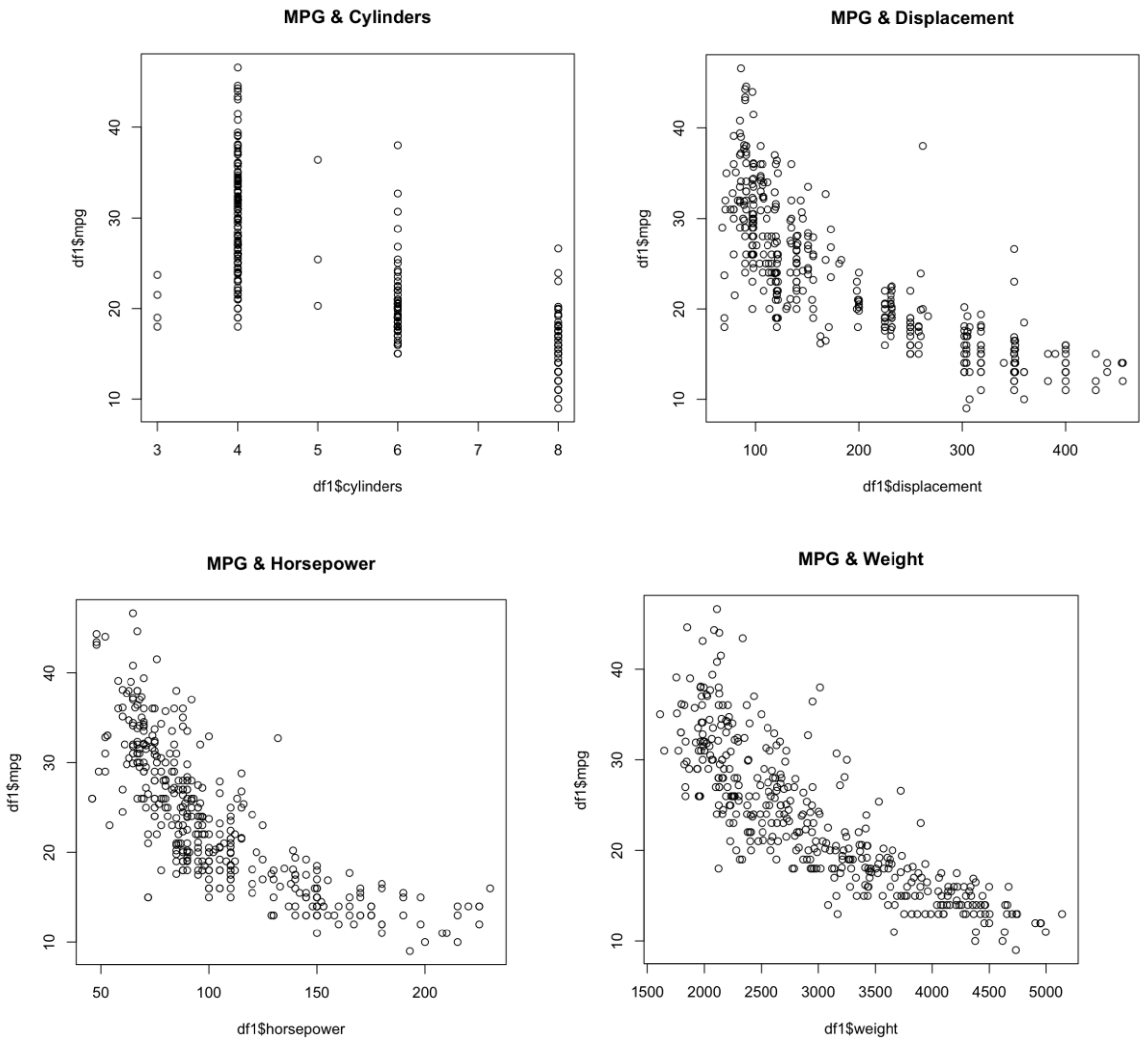
Correlation: MPG is least correlated with Origin, Acceleration, and Year. MPG is most correlated with Weight, Horsepower, Cylinders, and displacement, while these variables are strongly correlated with MPG, they are also correlated with each other, and may be removed in the modeling process due to multicollinearity.

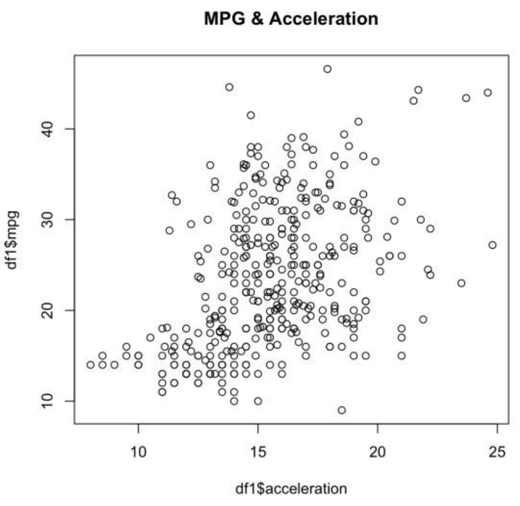
Figure 3: Heatmap showing correlation of data.

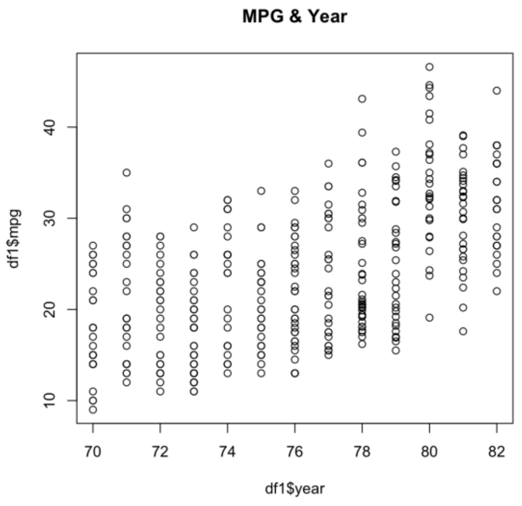


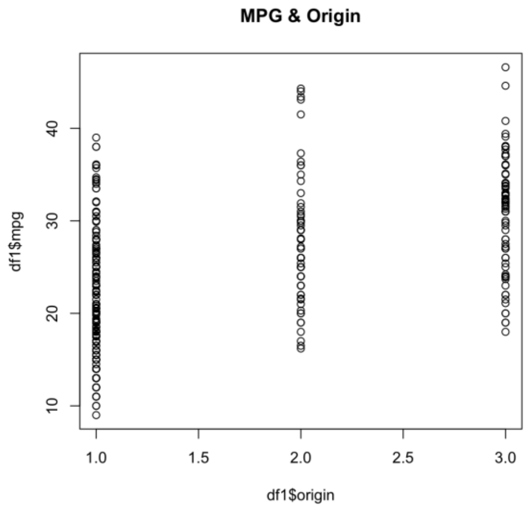
	horsepower	weight	cylinders	displacement	origin	mpg01	acceleration	year
year	-0.42	-0.31	-0.35	-0.37	0.18	0.43	0.29	1
acceleration	-0.69	-0.42	-0.5	-0.54	0.21	0.35	1	0.29
mpg01	-0.67	-0.76	-0.76	-0.75	0.51	1	0.35	0.43
origin	-0.46	-0.59	-0.57	-0.61	1	0.51	0.21	0.18
displacement	0.9	0.93	0.95	1	-0.61	-0.75	-0.54	-0.37
cylinders	0.84	0.9	1	0.95	-0.57	-0.76	-0.5	-0.35
weight	0.86	1	0.9	0.93	-0.59	-0.76	-0.42	-0.31
horsepower	1	0.86	0.84	0.9	-0.46	-0.67	-0.69	-0.42

Figure 4: Scatterplots showing MPG and response variables.









The scatterplots in *figure 4* showcase the relationship between MPG and the independent variables. The scatterplots confirm the weak correlation between MPG and Acceleration, Origin, and year.

Creating binary variable: The purpose of this analysis includes predicting whether a car has high or low MPG. To create this binary outcome, I created a new binary response variable that separates MPG below or equal to 22.75 as low mileage (0) and MPG above 22.75 as high mileage (1).

Variable selection: The variable selection process is necessary to remove variables not needed in the analysis and may cause multicollinearity. To select the significant variables, I used backward logistic regression with MPG01 as the response variable. This method selects variables by starting with all variables and removing variables until the AIC is the lowest. The selected variables include horsepower, weight, year, and origin. The combination of these variables produced the lowest AIC.

(iii) Method

The modeling process uses split data to test models and cross validation.

Split data – train & test: To begin I randomly split the data and assigned 75% to the training dataset and 25% to the testing dataset. I then created the following models using the training data and then tested using the training and testing dataset: LDA, QDA, Naive Bayes, Logistic Regression, KNN at k values of 1, 3, 5, 7, 9, 11,13, & 15, and SVM model.

I used the training dataset predictor variables to make a prediction with the models and compared the predicted response to the actual training dataset response to get the training error. I then used the testing dataset predictor variables to make a prediction using the models and then compared the predicted response to the actual test dataset response to get the testing error. I created a "confusion matrix" that compares the actual test response to the model's predicted response using the test dataset.

An analysis of the training error, testing error, and confusion matrixes for the models is used to choose the best performing model.

Cross Validation: Considering there are so few observations, this analysis would benefit from cross validation to fully utilize each observation in the model creation process. I chose to perform Monte Carlo cross validation by randomly creating training and testing data subsets over a chosen number of iterations to ensure each piece of data is uesd to train and test each model. I chose 100 iterations of splitting the data into different random training and testing subsets. I then created the following models on each iteration: LDA, QDA, Naive Bayes, Logistic Regression, KNN at k values of 1, 3, 5, 7, 9, 11,13, & 15, and SVM model.

I assessed the performance of each model by averaging each model's testing error across all the iterations.

(iv) Results

Split data – train & test subsets Training and Testing Error.

Name	Train Error	Test Error
LDA	0.08163265	0.1122449
QDA	0.07823129	0.07142857
Bayes	0.0952381	0.10204082
Log	0.08843537	0.09183673
KNN	0	0.091837
SVM	0.08843537	0.1122449

Confusion Matrixes:

LDA	0	1		QDA	0	1
0	43	3		0	47	3
1	8	44		1	4	44

Bayes	0	1		Log	0	1
0	44	7		0	47	4
1	3	44		1	5	42

KNN	0	1		SVM	0	1
0	43	8		0	44	7
1	7	40		1	4	43

Cross Validation – Mean and variability of Testing Error

Name	Avg. Testing Error	Var. Testing Error
LDA	0.0968	4.00E-04
QDA	0.0968	4.00E-04
Bayes	0.0987	5.00E-04
Log	0.0966	4.00E-04
KNN	0.0980	5.00E-04
SVM	0.0972	5.00E-04

(v) Findings

When using the split train and test data the Quadratic Discriminant Analysis was the best preforming model with the lowest test error of 0.07142857. These findings are reinforced by the confusion matrix as the QDA model has the highest accuracy among all the models. The combination of false negatives and false positives is the lowest. It's important to note that a different model might seem optimal under the condition that false negatives are more costly than false positives (or vice verses). In this scenario, the Bayes model could be the better model as it has a lower number of false negatives. For our analysis the QDA model is the best performing.

When using cross validation, the Logistic regression model achieved the best results with a testing error of .0966. LDA & QDA models are rounded to the same average testing error of .0968. Overall, the testing error for all models are very close to each other. For this situation, it is necessary to consider other characteristics of the model to make a final selection for our analysis. I would choose the logistic regression model as the final model for this analysis as it not only had the lowest average testing error, but it gives a lot of helpful information about each predictor variable, including AIC, beta values, and more.

Appendix:

```
library(lattice)

library(ggplot2)

library(dplyr)

library(melt)

library(reshape2)

library(naivebayes)

library(psych)

library(MASS)

library(e1071)




df1 <- read.table("Auto.csv", head=T, sep=",")

dim(df1)

head(df1)

summary(df1)


#MPG median is 22.75


boxplot(df1$mpg, main='MPG Box Plot', ylab='MPG')


#No outliers - no need to remove points to reassess median.


#assign new Variables (b)
```

```
df1['mpg01'] <- ifelse((df1$mpg > 22.75),1,0)

dim(df1[df1$mpg01 ==1,])[1]

dim(df1[df1$mpg01 ==0,])[1]


df <- df1[,-1]

dim(df[df$mpg01 ==1,])[1]

dim(df[df$mpg01 ==0,])[1]


#Visually analyze variables (c)

plot(df1$cylinders,df1$mpg, main='MPG & Cylinders')

plot(df1$displacement,df1$mpg,main='MPG & Displacement')

plot(df1$horsepower,df1$mpg,main='MPG & Horsepower')

plot(df1$weight,df1$mpg,main='MPG & Weight')

plot(df1$acceleration,df1$mpg,main='MPG & Acceleration')

plot(df1$year,df1$mpg,main='MPG & Year')

plot(df1$origin,df1$mpg,main='MPG & Origin')


#Correlation Heatmap

corr_mat <- round(cor(df1[-1]),2)


# reduce the size of correlation matrix

melted_corr_mat <- melt(corr_mat)

dist <- as.dist((1-corr_mat)/2)

# hierarchical clustering the dist matrix

hc <- hclust(dist)

corr_mat <-corr_mat[hc$order, hc$order]
```

```
# reduce the size of correlation matrix

melted_corr_mat <- melt(corr_mat)

#head(melted_corr_mat)

ggplot(data = melted_corr_mat,aes(x=Var1, y=Var2, fill=value)) + geom_tile()
+geom_text(aes(Var2, Var1, label = value), color = "orange",

                                   size = 4)


pairs.panels(df)




#Variable selection

glm1 <- glm(mpg01~.,df,family= 'binomial')

glm2 <- step(glm1, direction ='backward')

summary(glm2)

summary(glm1)







#Split the data into training and test sets

smp_size <- floor(0.75 * nrow(df))


## set the seed to make your partition reproducible

set.seed(123)

train_ind <- sample(seq_len(nrow(df)), size = smp_size)
```

```
train <- df[train_ind, ]

test <- df[-train_ind, ]


dim(train)

dim(test)









#modeling

TrainErr <- NULL;

TestErr  <- NULL;

#formula <- mpg01 ~ cylinders + displacement + horsepower + weight +acceleration + year
+ origin

formula <- mpg01 ~ horsepower + weight + year + origin

#formula <- mpg01 ~ cylinders + displacement + horsepower+weight+year+origin

#model 1: LDA

model_1 <- lda(train[,1:7], train$mpg01);

## training error

## we provide a detailed code here

pred1 <- predict(model_1,train[,1:7])$class

TrainErr <- c(TrainErr, mean( pred1  != train$mpg01));

TrainErr;
```

```
## 0.08163265 for miss.class.train.error

## testing error

pred1test <- predict(model_1,test[,1:7])$class;

TestErr <- c(TestErr,mean(pred1test != test$mpg01));

TestErr;

## 0.1122449 for miss.class.test.error



## You can also see the details of Testing Error

##    by the confusion table, which shows how the errors occur

cm1 <- table(pred1test,  test$mpg01)

cm1


#model 2: QDA

model_2 <- qda(train[,1:7], train$mpg01)

model_2$means

plot(pred2)

## Training Error

pred2 <- predict(model_2,train[,1:7])$class

TrainErr <- c(TrainErr, mean( pred2!= train$mpg01))

TrainErr

## 0.01136364 for miss.class.train.error of QDA,

##  which is much smaller than LDA

##  Testing Error

pred2test <- predict(model_2,test[,1:7])$class
```

```
TestErr <- c(TestErr, mean(pred2test != test$mpg01))

TestErr

cm2 <- table(pred2test,  test$mpg01)

cm2


#model 3:Naive Bayes

model_3<-naiveBayes(formula, data = train)


pred3 <- predict(model_3, train[,1:7]);

TrainErr <- c(TrainErr, mean( pred3 != train$mpg01))

TrainErr

## Testing Error

pred3test <- predict(model_3,test[,1:7])

TestErr <- c(TestErr,  mean( predict(model_3,test[,1:7]) != test$mpg01))

TestErr


cm3 <- table(test$mpg01, pred3test)





#model 4: Logistic Regression

model_4 <- glm(formula,train, family =binomial(link="logit"))


summary(model_4)
```

```
plot(train$weight, train$mpg01)

lines(train$weight,fitted.values(model_4), col="red")


confint(model_4, level=0.95)

coefficients(model_4)


y_pred <- predict(model_4,test[,1:7], type = 'response')



## Training Error  of (multinomial) logisitic regression

#TrainErr <- c(TrainErr, mean( round(predict(model_4, train[,1:7]))  != train$mpg01))

#TrainErr

##  0.08843537 for miss.class.train.error

## Testing Error of (multinomial) logisitic regression

#TestErr <- c(TestErr, mean( round(predict(model_4,test[,1:7])) != test$mpg01) )

#TestErr



#cm <- table(data = test$mpg01, round(y_pred))

#cm



library(nnet)

mod4 <- multinom(formula, data=train)

summary(mod4);

## Training Error  of (multinomial) logisitic regression
```

```
TrainErr <- c(TrainErr, mean( predict(mod4, train[,1:7])  != train$mpg01))

TrainErr


##  0.08843537 for miss.class.train.error

## Testing Error of (multinomial) logisitic regression

TestErr <- c(TestErr, mean( predict(mod4,test[,1:7]) != test$mpg01) )

TestErr


pred4test <- predict(mod4,test[,1:7])



cm4 <- table(test$mpg01, pred4test)

cm4




#model 5: KNN

library(class);

kkk <- c(1,3,5,7, 9, 11, 13, 15);

xnew <- train[,1:7];

res1train <- NULL;

for (i in 1: 8){

  kk <- kkk[i];

  ypred2.train <- knn(train[,1:7], xnew, train[,8], k=kk);

  res1train <- rbind(res1train, cbind(kk, mean(ypred2.train != train[,8])));
}


min(res1train)


TrainErr <- c(TrainErr, min(res1train))

TrainErr


testerror = NULL

## 4B: 8 KNN methods

xnew2 <- test[,-8];

for (i in 1: 8){

  kk <- kkk[i];

  ypred2.test <- knn(train[,1:7], xnew2, train[,8], k=kk);

  testerror <- cbind( testerror, mean( ypred2.test != test[,8]));

}

colnames(testerror) <- c("KNN1", "KNN3", "KNN5", "KNN7", "KNN9", "KNN11", "KNN13",
"KNN15")


round(testerror,6)


min(round(testerror,6))


TestErr <- c(TestErr, min(round(testerror,6)) )

TestErr
```

```
ypred2.test <- knn(train[,1:7], xnew2, train[,8], k=3)

ypred2.test

cm5 <- table(test$mpg01,ypred2.test)

cm5

#model 6: SVM

model_6 = svm(formula = formula, data = train, type = 'C-classification', kernel = 'linear')


svm_pred_train = predict(model_6, newdata = train[,1:7])


TrainErr <- c(TrainErr, mean( svm_pred_train  != train$mpg01))

TrainErr


svm_pred_test = predict(model_6, newdata = test[,1:7])

TestErr <- c(TestErr, mean(svm_pred_test != test$mpg01) )

TestErr

svm_pred_test

cm6 = table(test[, 8],svm_pred_test)



cm1

cm2

cm3

cm4

cm5

cm6
```

```
name <- c("LDA","QDA","Bayes","Log", 'KNN','SVM')

Results <- data.frame(name,TrainErr,TestErr)

Results


n= dim(df)[1];

n1= round(n/10);


#Cross Validation

B = 100;

TEALL=NULL;

for (b in 1:B){

 flag = sort(sample(1:n, n1));

 train = df[-flag,];

 test = df[flag, ];


 #model 1: LDA

 model_1 <- lda(train[,1:7], train$mpg01);


 pred1test <- predict(model_1,test[,1:7])$class;

 TestErr1 <- c(TestErr,mean(pred1test != test$mpg01));


 #model 2: QDA

 model_2 <- qda(train[,1:7], train$mpg01)

 pred2test <- predict(model_2,test[,1:7])$class

 TestErr2<- c(TestErr, mean(pred2test != test$mpg01))
 #}



 #model 3:Naive Bayes

 model_3<-naiveBayes(formula, data = train)


 pred3test <- predict(model_3,test[,1:7])

 TestErr3 <- c(TestErr,  mean( predict(model_3,test[,1:7]) != test$mpg01))


 #model 4: Logistic Regression


 library(nnet)

 model4 <- multinom(formula, data=train)


 TestErr4 <- c(TestErr, mean( predict(mod4,test[,1:7]) != test$mpg01) )


 #model 5: KNN

 library(class);

 kkk <- c(1,3,5,7, 9, 11, 13, 15);

 xnew <- train[,1:7];

 testerror = NULL

 ## 4B: 8 KNN methods

 xnew2 <- test[,-8];

 #for (i in 1: 8){

  # kk <- kkk[i];

   #ypred2.test <- knn(train[,1:7], xnew2, train[,8], k=kk);

 #  testerror <- cbind( testerror, mean( ypred2.test != test[,8]));

 #}
```

```
  #colnames(testerror) <- c("KNN1", "KNN3", "KNN5", "KNN7", "KNN9", "KNN11", "KNN13",
"KNN15")


  #round(testerror,6)


  #min(round(testerror,6))


  #TestErr5 <- c(TestErr, min(round(testerror,6)) )
  ypred2.test <- knn(train[,1:7], xnew2, train[,8], k=3);
  TestErr5 <- c(TestErr, mean( ypred2.test != test[,8]))



  #model 6: SVM
  model_6 = svm(formula = formula, data = train, type = 'C-classification', kernel = 'linear')


  svm_pred_test = predict(model_6, newdata = test[,1:7])
  TestErr6 <- c(TestErr, mean(svm_pred_test != test$mpg01) )


  #name <- c("LDA","QDA","Bayes","Log", 'KNN','SVM')
  #Results <- data.frame(name,TrainErr,TestErr)
  #Results


  TEALL = rbind(TEALL, cbind(TestErr1,TestErr2,TestErr3,TestErr4,TestErr5,TestErr6))


}
dim(TEALL);
```

```
round(apply(TEALL, 2, mean),6);



round(apply(TEALL, 2, var),6);

name <- c("LDA","QDA","Bayes","Log", 'KNN','SVM')

Results2 <- data.frame(name,round(apply(TEALL, 2, mean),6),round(apply(TEALL, 2,
var),6))

Results2
```