Predicting Car Fuel Efficiency

By Nickolas Hirata

Georgia Institute of Technology | Masters in Analytics

Why I chose this project

I originally wanted to showcase a PowerBI dashboard on examining Sample Return Lander's workforce budget but realized that it lacked Data Science practices & procedures which I wanted to better demonstrate in this interview.

I mainly chose this project because it was one of the first assignments from my favorite class "Data Mining & Statistical Learning" taught by Dr. Yajun Mei, that helped me understand Data Science storytelling.

The problem or challenge

How can we identify what is considered a fuel efficient car?

The Dataset

- Miles per Gallon (MPG)
- # of Cylinders
- Displacement
- Horsepower
- Weight
- Acceleration
- Year (‘70-’82)
- Origin (1-USA, 2-Europe, 3-Japan)
- *~~Car Name~~

I derived the response Y variable (mpg01) based on the median value of MPG (22.75) where greater than or equal to the median equals TRUE (Fuel Efficient) and then replaced the original mpg attribute.

mpg01	cylinders	displacement	horsepower	weight
Mode :logical	Min. :3.000	Min. : 68.0	Min. : 46.0	Min. :1613
FALSE:196	1st Qu.:4.000	1st Qu.:105.0	1st Qu.: 75.0	1st Qu.:2225
TRUE :196	Median :4.000	Median :151.0	Median : 93.5	Median :2804
	Mean :5.472	Mean :194.4	Mean :104.5	Mean :2978
	3rd Qu.:8.000	3rd Qu.:275.8	3rd Qu.:126.0	3rd Qu.:3615
	Max. :8.000	Max. :455.0	Max. :230.0	Max. :5140

acceleration	year	origin
Min. : 8.00	Min. :70.00	Min. :1.000
1st Qu.:13.78	1st Qu.:73.00	1st Qu.:1.000
Median :15.50	Median :76.00	Median :1.000
Mean :15.54	Mean :75.98	Mean :1.577
3rd Qu.:17.02	3rd Qu.:79.00	3rd Qu.:2.000
Max. :24.80	Max. :82.00	Max. :3.000

392 rows, 8 attributes - 6 rows were deleted due to missing data and Car Name was dropped based on instruction by the professor.

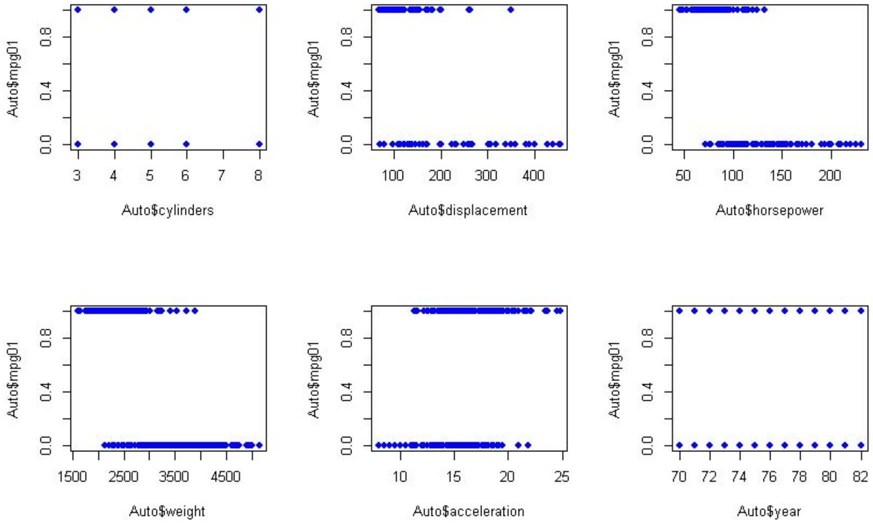
Auto MPG Dataset - UCI Machine Learning Repository
https://archive.ics.uci.edu/ml/datasets/auto+mpg

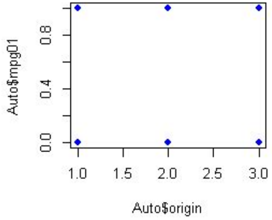
The hypothesis (or prediction)

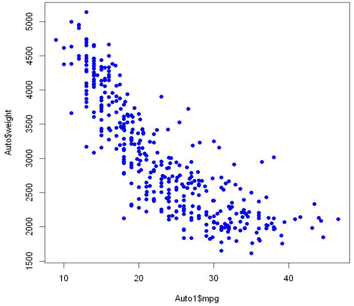
What do you think will happen?

Most factors will have a negative correlation with Miles Per gallon.

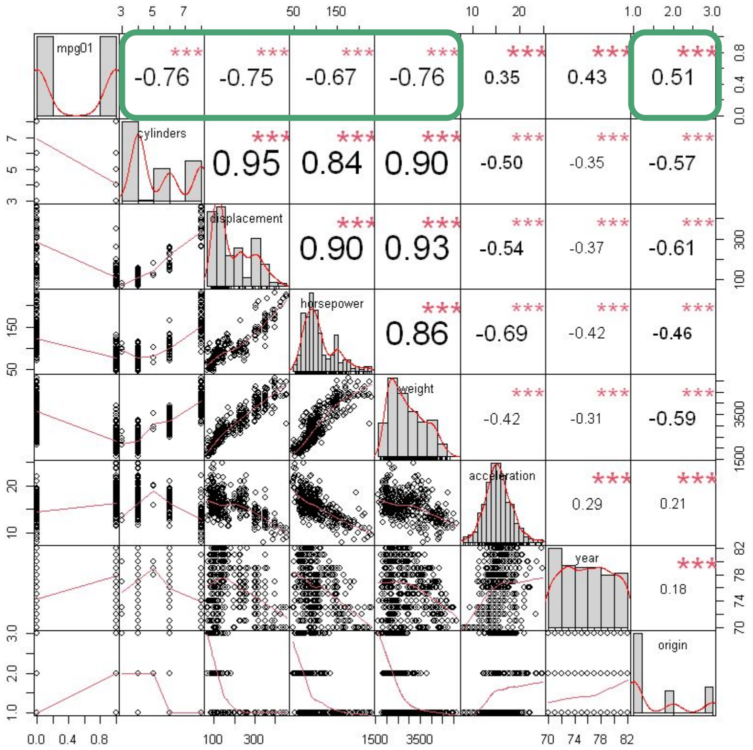
Exploratory Data Analysis







High Correlation - Magnitude greater than +/- (0.5)

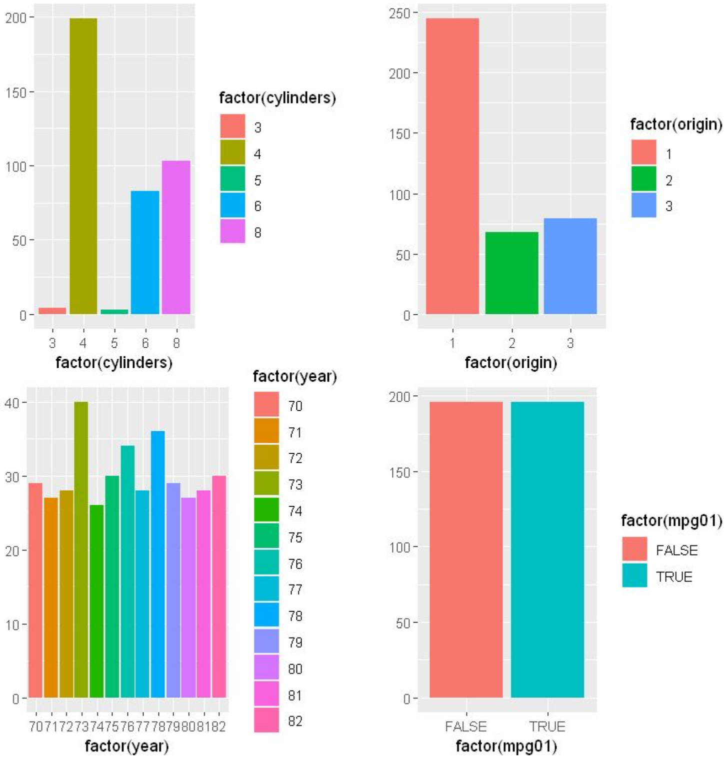


Research

Explanation of all the research done about this problem/challenge.

Goal: Identify trends, outliers, and better understand the data.

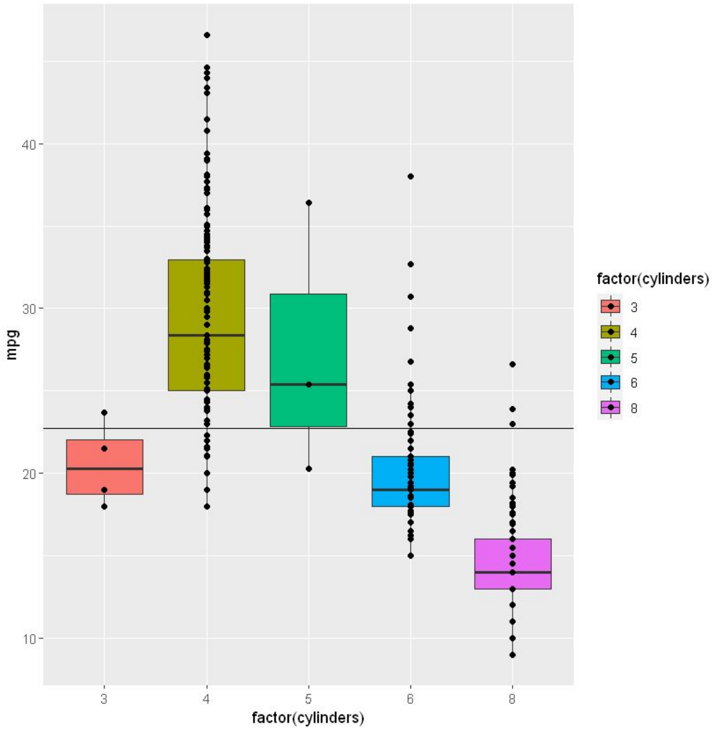
Research



Count Distribution for Categorical Data

- 4 cylinder cars account for 50% of the total.
- USA (Origin 1) accounts for 62.5% of the total.
- Year is well balanced.
- MPG01 (response variable) split at the median 22.75.

Research

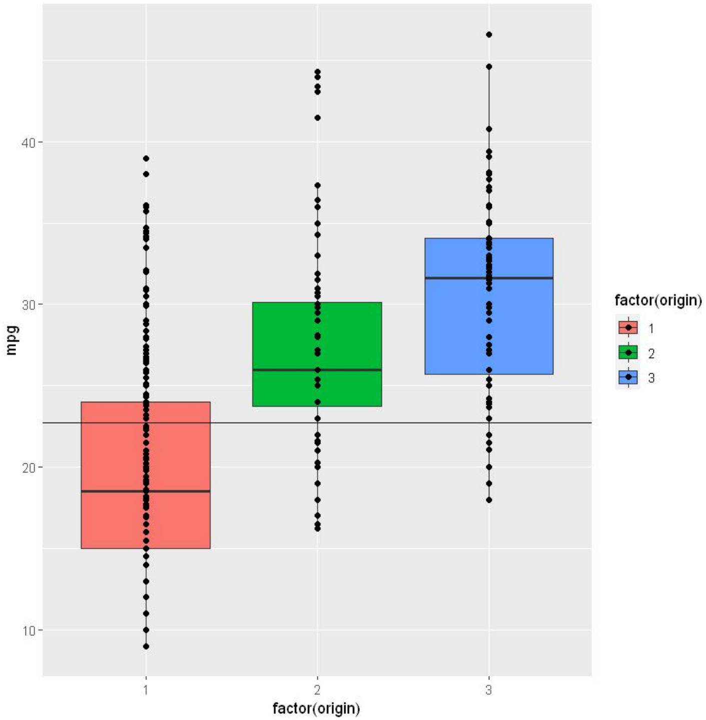


Horizontal black line - *Median of the Y-axis*

Cylinders to MPG

- 3 cylinder cars should generally outperform higher cylinders but didn't for this dataset.
  - Mazda Rx-2 Coupe, Rx-3, Rx-4, Rx-7
- Other than the 3 cylinder cars, we can see that higher cylinders take more miles per gallon to operate.

Research

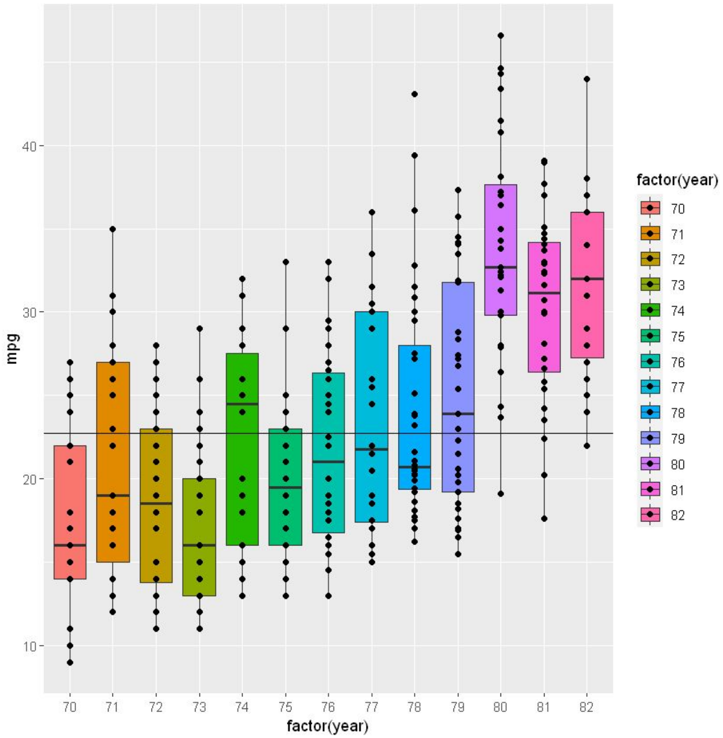


Horizontal black line *- Median of the Y-axis*

Origin to MPG

- Although, USA (Origin 1) accounts for most of the cars in the dataset, we can see that Europe (Origin 2) and Japan (Origin 3) have higher fuel efficiency on average.

Research



Horizontal black line *- Median of the Y-axis*

Year to MPG

- The correlation between Year and MPG was moderately positive at 0.43 which can be visually seen here.
- It's fair to say that fuel mileage continues to improve over time as technology advances.

Each Data Scientist uses different methods of experimentation

My testing methods

It’s always good practice to compare the results of different analytic techniques; this can either help to confirm results or highlight how different modeling assumptions and characteristics uncover new insights.

Training & Test Data split 90/10

What methods did you use in your experiment?

- Linear Discriminant Analysis
- Naive Bayes
- Logistic Regression
- K Nearest Neighbors

Linear Discriminant Analysis

Why did you choose this model?

LDA provides a dimensionality reduction technique for classification and uses distance to the class mean which is easier to interpret. Only handles continues data.

More about Linear Discriminant Analysis (LDA)

https://analyticsindiamag.com/a-hands-on-guide-to-linear-discriminant-analysis-for-binary-classification/

Prior Probabilities

50.4% of the training observations are Not-Fuel efficient cars while 49.6% represent Fuel efficient cars.

Group Means

Represent the average attribute values based on Fuel/Non-Fuel Efficient Cars.

Confusion Matrix

High Accuracy, Sensitivity but Low Specificity

```
Prior probabilities of groups:
    FALSE      TRUE
0.5042493 0.4957507

Group means:
      cylinders horsepower   weight displacement   origin
FALSE  6.769663  130.21910 3624.781     273.1517 1.168539
TRUE   4.177143   78.74857 2340.234     115.8829 1.965714

Coefficients of linear discriminants:
                       LD1
cylinders    -0.4898149289
horsepower    0.0026227391
weight       -0.0009761428
displacement  0.0001766958
origin        0.1607607744
```

```
               Accuracy : 0.8462
                 95% CI : (0.6947, 0.9414)
    No Information Rate : 0.5385
    P-Value [Acc > NIR] : 5.274e-05

                  Kappa : 0.688

 Mcnemar's Test P-Value : 0.6831

            Sensitivity : 0.9048
            Specificity : 0.7778
         Pos Pred Value : 0.8261
         Neg Pred Value : 0.8750
             Prevalence : 0.5385
         Detection Rate : 0.4872
   Detection Prevalence : 0.5897
      Balanced Accuracy : 0.8413
```

Naive Bayes

Why did you choose this model?

Naive Bayes doesn’t require as much training data and handles both continuous and discrete data.

More about Naive Bayes

https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf

Conditional probabilities performed similarly to LDA but the categorical variables were correctly transformed into factors.

```
Conditional probabilities:
               cylinders
auto.train[, 1]           3           4           5           6           8
          FALSE 0.016759777 0.100558659 0.000000000 0.374301676 0.508379888
          TRUE  0.005747126 0.902298851 0.011494253 0.063218391 0.017241379

               horsepower
auto.train[, 1]     [,1]     [,2]
          FALSE 129.5587 37.27232
          TRUE   79.3046 15.85390

               weight
auto.train[, 1]     [,1]     [,2]
          FALSE 3614.972 674.9411
          TRUE  2336.690 410.9347

               displacement
auto.train[, 1]     [,1]     [,2]
          FALSE 272.7989 88.66313
          TRUE  116.5833 40.20360

               origin
auto.train[, 1]          1          2          3
          FALSE 0.88826816 0.06703911 0.04469274
          TRUE  0.37356322 0.25862069 0.36781609
```

<u>Confusion Matrix</u>

High Accuracy, Sensitivity and Specificity

```
               Accuracy : 0.8718
                 95% CI : (0.7257, 0.957)
    No Information Rate : 0.5385
    P-Value [Acc > NIR] : 1.038e-05

                  Kappa : 0.741

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9048
            Specificity : 0.8333
         Pos Pred Value : 0.8636
         Neg Pred Value : 0.8824
             Prevalence : 0.5385
         Detection Rate : 0.4872
   Detection Prevalence : 0.5641
      Balanced Accuracy : 0.8690
```

Logistic Regression

Why did you choose this model?

Similar to Naive Bayes except it directly models the probability of P(y|x) by learning the input to output mapping by minimising the error. Works reasonably well even when some of the variables are correlated.

More about Logistic Regression
https://www.ibm.com/topics/logistic-regression

Insignificant Variables

Removing insignificant variable (displacement) didn't change the results of the model.

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	7.638e+00	2.432e+00	3.140	0.00169	**
cylinders4	4.260e+00	1.410e+00	3.021	0.00252	**
cylinders5	1.970e+01	9.713e+02	0.020	0.98382	
cylinders6	2.872e+00	1.733e+00	1.657	0.09751	.
cylinders8	5.811e+00	2.394e+00	2.428	0.01519	*
horsepower	-4.580e-02	1.737e-02	-2.637	0.00836	**
weight	-1.776e-03	8.649e-04	-2.054	0.04002	*
displacement	-1.184e-02	1.200e-02	-0.987	0.32385	
origin2	-5.103e-01	6.749e-01	-0.756	0.44955	
origin3	7.889e-01	8.222e-01	0.959	0.33731	

Confusion Matrix

High Accuracy, Sensitivity and Specificity

```
               Accuracy : 0.8718
                 95% CI : (0.7257, 0.957)
    No Information Rate : 0.5385
    P-Value [Acc > NIR] : 1.038e-05

                  Kappa : 0.741

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9048
            Specificity : 0.8333
         Pos Pred Value : 0.8636
         Neg Pred Value : 0.8824
             Prevalence : 0.5385
         Detection Rate : 0.4872
   Detection Prevalence : 0.5641
      Balanced Accuracy : 0.8690
```

K Nearest Neighbor

Why did you choose this model?

K-Nearest Neighbor (KNN) is very intuitive and easy to use since it's a non-parametric algorithm (no assumptions).

More about K Nearest Neighbors (KNN)
https://www.fromthegenesis.com/pros-and-cons-of-k-nearest-neighbors/

<u>Choosing "k"</u>

k (# of neighbors) is identified by running multiple k's and picking the best outcome (k=5).

k_value	value
<chr>	<dbl>
1	0.10256410
3	0.07692308
5	0.05128205
7	0.10256410
9	0.12820513
11	0.10256410
13	0.12820513
15	0.12820513

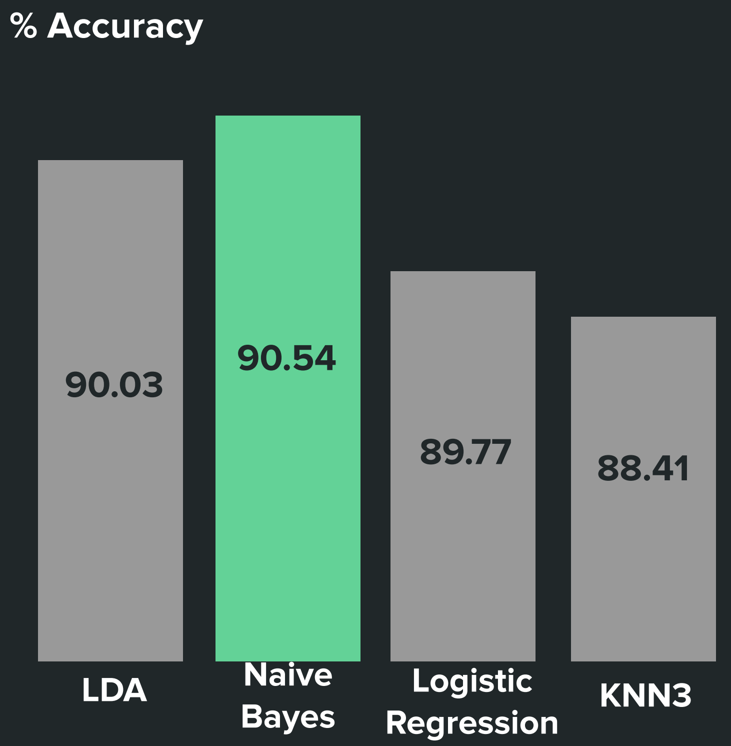
<u>Confusion Matrix</u>

High Accuracy, Sensitivity but Low Specificity

```
               Accuracy : 0.8718
                 95% CI : (0.7257, 0.957)
    No Information Rate : 0.5385
    P-Value [Acc > NIR] : 1.038e-05

                  Kappa : 0.739

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9524
            Specificity : 0.7778
         Pos Pred Value : 0.8333
         Neg Pred Value : 0.9333
             Prevalence : 0.5385
         Detection Rate : 0.5128
   Detection Prevalence : 0.6154
      Balanced Accuracy : 0.8651
```

Cross Validation

Running one variation of the split data is not enough to make a clear decision so multiple splits are required for the overall conclusion.



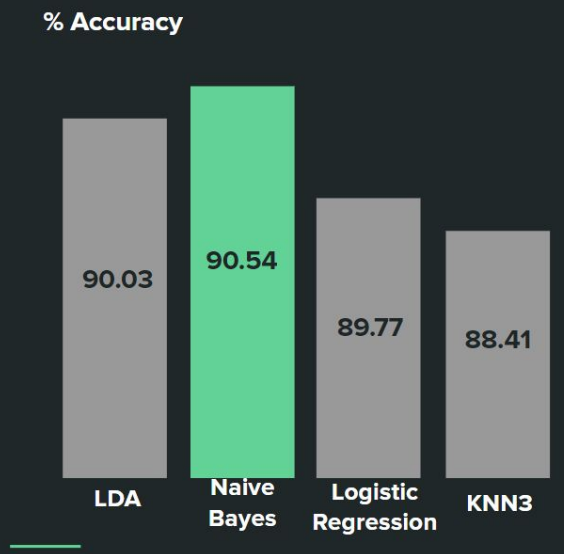
Monte Carlo Cross Validation - 100 run average

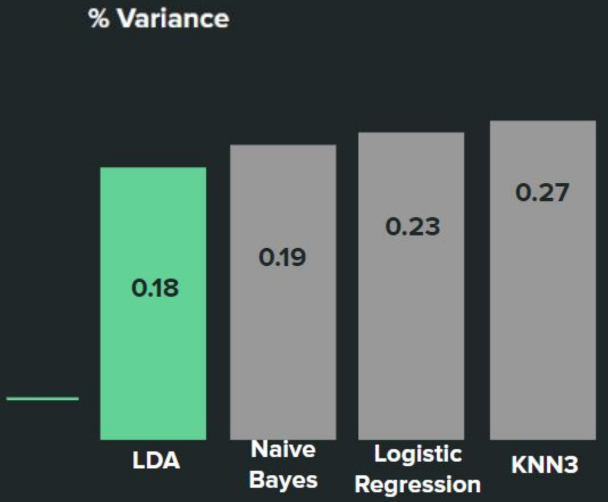
Aha!

My discoveries

What did you learn after testing?

1. Naive Bayes performed the best with a 90.54% accuracy.
2. LDA had the least amount of variability in the cross validation.
3. Results were very close to one another even though certain assumptions weren't met.





Conclusion

Naive Bayes and LDA performed better than the other models based on cross validation percentage accuracy and variance. However, if assumptions are not met, the model may inaccurately reflect the data and will likely result in inaccurate predictions. I would suggest further testing models that are not affected by multicollinearity such as the Random Forest model.

What will I do next?

What will you do with your findings next? How will you further your research/findings?

- Try Random Forest and other Ensemble methods.
- Use outside sources to find more meaningful ways to use hyperparameters.

Appendix

Assignment

ISyE 7406: Data Mining & Statistical Learning
HW#3

Classification in R. In this problem, you are asked to write a report to summarize your analysis of the popular "Auto MPG" data set in the literature. Much research has been done to analyze this data set, and here the objective of our analysis is to predict whether a given car gets high or low gas mileage based 7 car attributes such as cylinders, displacement, horsepower, weight, acceleration, model year and origin.

(a) The "Auto MPG" data set is available at UCI Machine Learning (ML) Repository:

```
https://archive.ics.uci.edu/ml/datasets/Auto+MPG
```

Download the data file "auto-mpg.data" from UCI ML Repository or from Canvas, and use Excel or Notepad to see the data (this is a .txt file).

There are 398 rows (i.e., 398 different kinds of cars), and 9 columns (the car attributes and name). Before we do any analysis, we need to clean the raw data. In particular, some values are missing for this dataset. Many statistical methods have been proposed to deal with missing values, and please conduct literature research by yourself. For the purpose of simplicity in this homework, here we adopt a simple though inefficient method to remove those rows with missing values. Also we remove the last column of car names, which is text/string and may cause trouble in our numerical analysis. These two deletions lead to a new cleaned data set of 392 observations and 8 columns.

To save your time, you can access the cleaned data from the file "Auto.csv" from Canvas and the R code below if you save it in the local folder of your computer, say, "C:/Temp":

```
Auto1 <- read.table(file = "C:/Temp/Auto.csv", sep = ",", header=T);
```

(b) Create a binary variable, `mpg01`, that contains a 1 if `mpg` contains a value above its median, and a 0 if `mpg` contains a value below its median. You can compute the median using the `median()` function. Note you may find it helpful to use the `data.frame()` function to create a single data set containing both `mpg01` and the other `Auto` variables.

```
mpg01 = I(Auto1$mpg >= median(Auto1$mpg))
Auto  = data.frame(mpg01, Auto1[,-1]);  ## replace column "mpg" by "mpg01".
```

(c) Explore the data graphically in order to investigate the association between `mpg01` and the other features. Which of the other features seem most likely to be useful in predicting `mpg01`? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

(d) Split the data into a training set and a test set. Any reasonable splitting is acceptable, as long as you clearly explain how you split and why you think it is reasonable. For your convenience, you can either randomly split, or save every fifth (or tenth) observations as testing data.

(e) Perform the following classification methods on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (c). What is the test error of the model obtained?

(1) *LDA* (2) *QDA* (3) *Naive Bayes* (4) *Logistic Regression*

(5) *KNN* with **several** values of K. Use only the variables that seemed most associated with `mpg01` in (c). Which value of K seems to perform the best on this data set?

Appendix

Library Packages

```
library(MASS)
library(PerformanceAnalytics)
library(dplyr)
library(tidyverse)
library(caret)
library(gridExtra)
```

Slide 4 & 6

```
summary(Auto)
par(mfrow=c(3,3))
plot(Auto$mpg01~Auto$cylinders,pch=19,col="blue")
plot(Auto$mpg01~Auto$displacement,pch=19,col="blue") #sig
plot(Auto$mpg01~Auto$horsepower,pch=19,col="blue") #sig
plot(Auto$mpg01~Auto$weight,pch=19,col="blue") #sig
plot(Auto$mpg01~Auto$acceleration,pch=19,col="blue") #sig
plot(Auto$mpg01~Auto$year,pch=19,col="blue")
plot(Auto$mpg01~Auto$origin,pch=19,col="blue")
par(mfrow=c(1,1))
plot(Auto$weight~Auto$horsepower,pch=19,col="blue")
plot(Auto$weight~Auto1$mpg,pch=19,col="blue")
```

```
##Correlation Plot
chart.Correlation(Auto1, histogram=TRUE, pch=19)
```

Slide 8

```
plot1=qplot(factor(cylinders), data=Auto, geom="bar", fill=factor(cylinders))
plot2=qplot(factor(year), data=Auto, geom="bar", fill=factor(year))
plot3=qplot(factor(origin), data=Auto, geom="bar", fill=factor(origin))
plot4=qplot(factor(mpg01), data=Auto, geom="bar", fill=factor(mpg01))
grid.arrange(plot1,plot3,plot2,plot4,ncol=2)
```

Slide 9-11

```
qplot(factor(cylinders), mpg, data = Auto1, geom = c("boxplot"),fill=factor(cylinders))+
    geom_point() +
    geom_hline(yintercept = median(Auto1$mpg, na.rm=TRUE))
qplot(factor(year), mpg, data = Auto1, geom = c("boxplot"),fill=factor(year)) +
    geom_point() +
    geom_hline(yintercept = median(Auto1$mpg, na.rm=TRUE))
qplot(factor(origin), mpg, data = Auto1, geom = c("boxplot"),fill=factor(origin)) +
    geom_point() +
    geom_hline(yintercept = median(Auto1$mpg, na.rm=TRUE))
```

Appendix

Data Splitting Technique

```
#SPLIT DATA INTO TRAINING AND TEST SETS#
n = dim(Auto)[1]; ### total number of observations
n1 = round(n/10); ### number of observations randomly selected for testing data; ~10% of my data
RNGkind(sample.kind = "Rounding")
set.seed(888); ### set the seed for randomization
flag = sort(sample(1:n, n1));
auto.train = Auto[-flag,]
auto.test = Auto[flag,]
auto.train$mpg01 <- as.factor(auto.train$mpg01);
auto.test$mpg01 <- as.factor(auto.test$mpg01);

#USE ONLY VARIABLES THAT SEEMED MOST ASSOCIATED WITH MPG01
auto.train = auto.train[c("mpg01","cylinders","horsepower","weight","displacement","origin")]
auto.test = auto.test[c("mpg01","cylinders", "horsepower","weight","displacement","origin")]
```

Linear Discriminant Analsyis (LDA)

```
set.seed(77)

TrainErr <- NULL;
TestErr  <- NULL;
### Method 1: LDA
# fit1 <- lda( y ~ ., data= auto.train, CV= TRUE)
mod1 <- lda(auto.train[,2:6], auto.train[,1]);
## training error
## we provide a detailed code here
pred1 <- predict(mod1,auto.train[,2:6])$class;
TrainErr <- c(TrainErr, mean( pred1  != auto.train$mpg01));
TrainErr;
## 0.09348442 for miss.class.train.error
confusionMatrix(as.factor(pred1),auto.train$mpg01,positive="TRUE")
## testing error
pred1test <- predict(mod1,auto.test[,2:6])$class;
TestErr <- c(TestErr,mean(pred1test != auto.test$mpg01));
TestErr;
## 0.1538462 for miss.class.test.error
## You can also see the details of Testing Error
##     by the confusion table, which shows how the errors occur
mod1
ldahist(predict(mod1,auto.train[,2:6])$x[,1], g= pred1)
confusionMatrix(as.factor(pred1test),auto.test$mpg01,positive="TRUE")
```

Appendix

Data Splitting Technique

```
#SPLIT DATA INTO TRAINING AND TEST SETS#
n = dim(Auto)[1]; ### total number of observations
n1 = round(n/10); ### number of observations randomly selected for testing data; ~10% of my data
RNGkind(sample.kind = "Rounding")
set.seed(888); ### set the seed for randomization
flag = sort(sample(1:n, n1));
auto.train = Auto[-flag,]
auto.test = Auto[flag,]
auto.train$mpg01 <- as.factor(auto.train$mpg01);
auto.test$mpg01 <- as.factor(auto.test$mpg01);

#USE ONLY VARIABLES THAT SEEMED MOST ASSOCIATED WITH MPG01
auto.train = auto.train[c("mpg01","cylinders","horsepower","weight","displacement","origin")]
auto.test = auto.test[c("mpg01","cylinders", "horsepower","weight","displacement","origin")]
```

Linear Discriminant Analysis (LDA)

```
set.seed(77)

TrainErr <- NULL;
TestErr  <- NULL;
### Method 1: LDA
# fit1 <- lda( y ~ ., data= auto.train, CV= TRUE)
mod1 <- lda(auto.train[,2:6], auto.train[,1]);
## training error
## we provide a detailed code here
pred1 <- predict(mod1,auto.train[,2:6])$class;
TrainErr <- c(TrainErr, mean( pred1  != auto.train$mpg01));
TrainErr;
## 0.09348442 for miss.class.train.error
confusionMatrix(as.factor(pred1),auto.train$mpg01,positive="TRUE")
## testing error
pred1test <- predict(mod1,auto.test[,2:6])$class;
TestErr <- c(TestErr,mean(pred1test != auto.test$mpg01));
TestErr;
## 0.1538462 for miss.class.test.error
## You can also see the details of Testing Error
##     by the confusion table, which shows how the errors occur
mod1
ldahist(predict(mod1,auto.train[,2:6])$x[,1], g= pred1)
confusionMatrix(as.factor(pred1test),auto.test$mpg01,positive="TRUE")
```

Appendix

Naive Bayes

```
auto.train$origin=as.factor(auto.train$origin)
auto.test$origin=as.factor(auto.test$origin)

auto.train$cylinders=as.factor(auto.train$cylinders)
auto.test$cylinders=as.factor(auto.test$cylinders)
## Method 3: Naive Bayes
##  This has been implemented in the R library "e1071"
##  You need to first install this library
##
library(e1071)
mod3 <- naiveBayes( auto.train[,2:6], auto.train[,1])
## Training Error
pred3 <- predict(mod3, auto.train[,2:6]);
TrainErr <- c(TrainErr, mean( pred3 != auto.train$mpg01))
TrainErr
##  0.09631728 for miss.class.train.error of Naive Bayes
## Testing Error
pred3test<-predict(mod3,auto.test[,2:6]);
TestErr <- c(TestErr, mean( predict(mod3,auto.test[,2:6]) != auto.test$mpg01))
TestErr
##  0.1282051 for miss.class.test.error of Naive Bayes
# mod3
confusionMatrix(as.factor(pred3test),auto.test$mpg01,positive="TRUE")
```

Logistic Regression

```
## Method 4: Logistic Regression
## Both R code lead to the same results, the default link for binomial is "logit"

glm_model <- glm(mpg01 ~ ., data = auto.train, family = binomial(link = "logit"))
# summary(glm_model)
probs <- predict(glm_model, auto.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs >= 0.5] <- 1
pred.glm= as.logical(pred.glm)
summary(glm_model)
confusionMatrix(as.factor(pred.glm),auto.test$mpg01, positive='TRUE')
## 0.1538462 if we used cylinders, weight, horsepower, displacement, and origin as the only predictors.
```

K-Nearest Neighbor

```
## Method 5: KNN model with several K values using only variables that most associated with mpg01.
library(class)
TEALL<-NULL
xnew <- auto.test[,-1];
for(i in seq(1,16,2)){
  ypred.test <- knn(auto.train[,-1], xnew, auto.train[,1],k=i);
  temptesterror <- mean(ypred.test != auto.test[,1]);
  TEALL <- c(TEALL, temptesterror);
}
output<-data.frame(k_value=c("1","3","5","7","9","11","13","15"),value=TEALL,stringsAsFactors = F)
output
##Optimal K Value
output$k_value[which(output$value==min(output$value))]

ypred1.test=knn(auto.test[,-1], xnew, auto.test[,1],k=5)
confusionMatrix(as.factor(ypred1.test),auto.test$mpg01,positive="TRUE")
```

Appendix

Cross Validation

```r
## Cross Validation for all Models ##
set.seed(888)
B= 100; ### number of loops
TEALL = NULL; ### Final TE values
KNNALL= NULL; ### Final Knn values
te1<-NULL;te2<-NULL;te3<-NULL; te4<-NULL;
for (b in 1:B){
  flag = sort(sample(1:n, n1));
  auto.train = Auto[-flag,]
  auto.test = Auto[flag,]
  auto.train = auto.train[c("mpg01","cylinders","horsepower","weight","displacement","origin",'year')]
  auto.test = auto.test[c("mpg01","cylinders", "horsepower","weight","displacement","origin",'year')]

  ### Method 1: LDA
  # fit1 <- lda( y ~ ., data= auto.train, CV= TRUE)
  mod1 <- lda(auto.train[,2:6], auto.train[,1]);
  pred1test <- predict(mod1,auto.test[,2:6])$class;
  te1 <- c(te1,mean(pred1test != auto.test$mpg01));


  ## Method 2: QDA
  mod2 <- qda(auto.train[,2:6], auto.train[,1])
  te2 <- c(te2, mean( predict(mod2,auto.test[,2:6])$class != auto.test$mpg01))


  ## Method 3: Naive Bayes
  mod3 <- naiveBayes( auto.train[,2:6], auto.train[,1])
  te3 <- c(te3,  mean( predict(mod3,auto.test[,2:6]) != auto.test$mpg01))


  ## Method 4: Logistic Regression
  ## Both R code lead to the same results, the default link for binomial is "logit"
  glm_model <- glm(mpg01 ~ ., data = auto.train, family = binomial(link = "logit"))
  probs <- predict(glm_model, auto.test, type = "response")
  pred.glm <- rep(0, length(probs))
  pred.glm[probs >= 0.5] <- 1
  # table(pred.glm, auto.test$mpg01)
  te4<-c(te4,mean(pred.glm != auto.test$mpg01))


  ## Method 5: KNN model with several K values using only variables that most associated with mpg01.
  cverror<-NULL
  xnew <- auto.test[,-1];
  for(i in seq(1,16,2)){
    ypred.test <- knn(auto.train[,-1], xnew, auto.train[,1],k=i);
    temptesterror <- mean(ypred.test != auto.test[,1]);
    cverror <- c(cverror, temptesterror);

  }

  KNNALL<-rbind(KNNALL,cbind(cverror[1],cverror[2],cverror[3],cverror[4],cverror[5],cverror[6],cverror[7],cverror[8]))
}
```

Results

```r
TEALL<-rbind(TEALL,cbind(te1,te2,te3,te4,KNNALL))
TEALL<-data.frame(TEALL)
colnames(TEALL) <- c("LDA", "QDA", "Naive Bayes", "Log Reg.", "KNN1",
                     "KNN3", "KNN5", "KNN7", "KNN9", "KNN11", "KNN13", "KNN15");
apply(TEALL,2,mean)
apply(TEALL,2,var)
```