Filippo Riva & Fernando Pascacio Cruz
MT 4420: Statistical Learning
Dr. Brandt
04/29/24

1) INTRODUCTION:

This paper aims to guide readers through our team's development of a model to predict whether a given car attains high or low gas mileage, utilizing the Auto dataset. The project endeavors to demonstrate a comprehensive grasp of various model classification and regression techniques. To achieve this, we have employed Quadratic Discriminant Analysis (QDA), Linear Discriminant Analysis (LDA), logistic regression, K-Nearest Neighbors (KNN), and graph analysis to construct diverse models.

2) DATA SET:

The Auto dataset comprises various variables, including mpg, acceleration, cylinders, year, weight, horsepower, displacement, origin, and name. All the cars in the dataset are from the period spanning 1970 to 1982 and weigh less than 5000 pounds. The cars in the dataset originate from the USA, Europe, and Japan.

3) THE QUESTION:

The task at hand is to develop a predictive model that can determine whether a given car achieves high or low gas mileage using the Auto dataset. This dataset serves as the foundation for training and evaluating various classification algorithms, including Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), logistic regression, and K-Nearest Neighbors (KNN). In classification, the goal is to assign categorical labels to input data points, effectively distinguishing them into distinct classes based on learned patterns from the training data. Unlike regression models, which are better suited for modeling quantitative response variables, classification methods excel in analyzing qualitative response variables. Given the objective of predicting whether a car would exhibit higher or lower miles per gallon (mpg), the team opted for different classification algorithms. These algorithms are well-equipped to handle the qualitative nature of the response variable, aligning with the task requirements.

4) THE SOLUTION:

To prepare the data for classification, a new binary variable named "mpg01" is introduced. This variable assigns a value of 1 to cars surpassing the median mileage and 0 to those falling below it. Subsequently, the dataset is divided into a training set, consisting of 292 cars, and a test set, comprising the remaining 100 cars out of the total 392. This partitioning allows for the efficient training of models on a subset of data and facilitates the evaluation of their performance on unseen instances. The training / test was set up with 3 different seeds.

After training the model, our next step was to identify the most influential variables for predicting whether a car would have higher or lower mileage. To discern which variables were likely to be the most informative for predicting mpg01, we conducted a multiple regression analysis incorporating all the variables from the dataset along with mpg01. Additionally, we visually explored the relationships between each variable and mpg01 by plotting scatterplots. Following a thorough examination of the results from the multiple linear regression, particularly scrutinizing the p-values, conducting hypothesis tests, and carefully analyzing the scatterplots, we arrived at a decision. It became evident that two variables stood out as significant predictors for mpg01: "year" and "weight". These variables demonstrated notable associations with the target variable and were deemed most suitable for predicting whether a car would exhibit higher or lower mileage.

Following up, it was time to test our data set by using different types of classifications:
LDA classification (Part D):
LDA classification has two important assumptions:

1) Each predictor has a normal distribution
2) All predictors have the same variance (Standard Deviation)

LDA is a special case of QDA and Naïve Bayes, it is mostly used when the true decision boundaries are linear. The table below highlights the results that were produced from the LDA model.

Results from the analysis:

Seed	Confusion Matrix	Success Rate
0	mpg01.test ldaMod.class 0 1 0 45 2 1 8 45	0.90 = 90%
3	mpg01.test ldaMod.class 0 1 0 47 2 1 6 45	0.92 = 92%

4	mpg01.test ldaMod.class 0 1 0 44 2 1 4 50	0.94 = 94%

Overall, the success rate of every seed is above 90 % showing that the model develop is good at predicting whether a car has an High or low milage

After LDA classification, was time for QDA classification method (Part E). QDA makes different assumptions from LDA, it assumes that within each class, the predictors have a normal and the covariance matrix is allowed to be different for each class. QDA works better when boundaries between classes are non linear and it is recommended if the training set is very large. The table below highlights the results from the QDA model.

Result from the analysis:

Seed	Confusion Matrix	Success Rate
0	mpg01.test qdaMod.class 0 1 0 45 3 1 8 44	0.89 = 89%
3	mpg01.test qdaMod.class 0 1 0 46 3 1 7 44	0.90 = 90%
4	mpg01.test qdaMod.class 0 1 0 44 2 1 4 50	0.94 = 94%

Overall, the success rate in every seed is above 85 % showing that the model that we have developed is strong enough to predict whether a car has a high milage or low milage.

After we had done the QDA classification model, we then realized that we could also use a logistic regression model (Part F) to help us see if the variables that we had chosen are actually important in the determination of seeing if cars attain high or low gas mileage. This is because of how logistic regression differs from the rest of the classification models by still producing an R output where we can see the statistical significance of each predictor. It is a close relative of the LDA model, but LDA tends to outperform the logistic regression model when the predictors are normally distributed within each class or

the logistic regression model tends to outperform the LDA model when the predictors are not normally distributed within each class. The table below highlights the results that were produced from the logistic regression model.

Results from the analysis:

Seed	Confusion Matrix	Success Rate
0	mpg01.test glm.pred1 0 1 0 46 3 1 7 44 0.9	0.90 = 90%
3	mpg01.test glm.pred1 0 1 0 47 3 1 6 44 0.91	0.91 = 91%
4	mpg01.test glm.pred1 0 1 0 45 4 1 3 48 0.93	0.93 = 93%

Overall, the success rate in every seed is above 90 % showing that the model that we have developed is strong enough to predict whether a car has a high milage or low milage.

Finally, after we have finished with our logistic regression model, we are now left with the KNN model (Part H). The KNN model is a fantastic model in the way that it can be used for both regression and classification problems. It works really well when the decision boundary is highly non-linear. It is important to note that the performance can also be affected by the choice of our K value because a smaller K value can give high variance and low bias, while on the other hand having a large K value gives low variance and a high bias. Additionally, for the Auto Data set, we also had to standardize the data set because of how it has several variables that are measured in different units. This makes it a lot harder to identify a point's nearest neighbor and can alter the success rate of the KNN model, which is why we must scale the data set to that all the variables can contribute equal to any of the models that we are trying to develop. The table below highlights the results that were produced from the KNN model.

Results from the analysis:

Seed	K	Confusion Matrix	Success Rate
0	2	test.mpg01 KNN.pred1 0 1 0 46 5 1 7 42 0.88	0.88 = 88%
0	10	test.mpg01 KNN.pred1 0 1 0 48 4 1 5 43 0.91	0.91 = 91%
3	2	test.mpg01 KNN.pred1 0 1 0 44 6 1 9 41 0.85	0.85 = 85%
3	10	test.mpg01 KNN.pred1 0 1 0 49 3 1 4 44 0.93	0.93 = 93%

Overall, the success rate in every seed is above 85 % showing that the model that we have developed is strong enough to predict whether a car has a high milage or low milage.

Ultimately, we can now compare all the four different classification models to see which one is the overall best representation of predicting whether cars have a high milage or low milage. In order to determine which model is the best, we have to compare the confusion matrix of each individual model and as well as their success rates. From what we can gather, we were able to determine that the LDA model is the best model for the representation of whether or not cars have a high milage or low milage. The second best model that would follow suit would be our logistic regression model, the third best model would then be our QDA model, and then lastly the KNN model.

5) CONCLUSION

By running these 4 different classification models, we were able to determine which one was ultimately the best a predicting whether or not a car has a high milage or low milage. The classification model that was the best fit for the problem that we were assigned was the LDA model because of how it had a higher success rate and had predicted correctly the amount of cars that had high milage or low milage in its confusion matrix. As mentioned previously, we decided not to use a regression model because of the fact that the problem is more attributed to a

classification setting. This is because of the fact that the regression models are primarily used for quantitative response variables while our problem is more qualitative, which is what classification models investigates and analyzes qualitative response variables. There was one additional question that we had in mind in terms of if we had the time to investigate this problem even further and it was that would this best model change if we introduced a Naive Bayes Model into the equation and compared it to all the other classification models. Another question that we had in mind was how would this affect our models if we had a more updated/modern Auto data set that included super cars or even muscle cars.

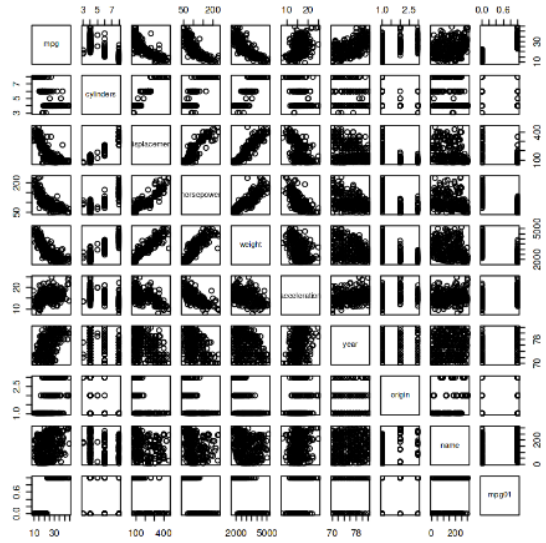
6) APPENDIX

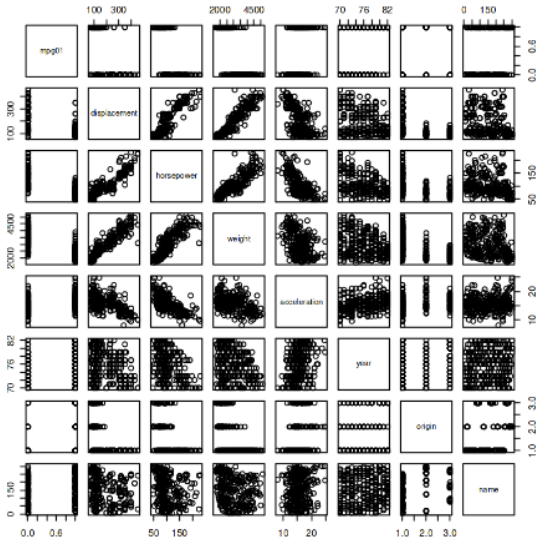
#PART A / #PART B / #PART C (Create new variable mpg01 / evaluate most important variables with regression and scatterplots method / train and test the set)

```
Auto=read.csv(file="Auto.csv",head=TRUE,na.strings="?",stringsAsFactors = T)
Auto = na.omit(Auto)
dim(Auto)
mpg=Auto$mpg
#summary(Auto)
mpg01=rep(0,392) #Set up a vector of 392 zeroes
mpg01[Auto$mpg > 22.75] = 1 # Switch value for car with over 22.75 mpg to 1
Auto=data.frame(Auto,mpg01) #Add new variable to Auto
set.seed(0) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars.
glm.fits=glm(mpg01 ~ displacement + horsepower + weight + acceleration + year + origin ,
data=Auto.train, family=binomial)
#summary(glm.fits)  #Try if train and test works with random variables
#seed(0)
glm.fit.seed0 =glm(mpg01 ~ displacement + horsepower + weight + acceleration + year + origin ,
data=Auto.train, family=binomial)
summary(glm.fit.seed0)
glm.probs.seed0 = predict(glm.fit.seed0, Auto.test, type = "response")
glm.pred.seed0 = rep(0,100)
glm.pred.seed0[glm.probs.seed0 > 0.5] = 1
confusion.matrix.seed0 = table(glm.pred.seed0, mpg01.test)
confusion.matrix.seed0
pairs(Auto.train)   #create scatterplots
pairs(~ mpg01 + displacement + horsepower + weight + acceleration + year + origin + name, data =
Auto.train)
```

```
Call:
glm(formula = mpg01 ~ displacement + horsepower + weight + acceleration +
    year + origin, family = binomial, data = Auto.train)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.629e+01  6.450e+00  -2.526  0.01155 *
displacement  7.795e-04  8.790e-03   0.089  0.92933
horsepower   -3.934e-02  2.729e-02  -1.442  0.14942
weight       -4.177e-03  1.296e-03  -3.222  0.00127 **
acceleration -1.952e-02  1.581e-01  -0.124  0.90171
year          4.153e-01  8.525e-02   4.871 1.11e-06 ***
origin        3.700e-01  3.864e-01   0.958  0.33831
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.67  on 291  degrees of freedom
Residual deviance: 124.80  on 285  degrees of freedom
AIC: 138.8

Number of Fisher Scoring iterations: 7
```

```
                mpg01.test
  glm.pred.seed0  0  1
               0 46  2
               1  7 45
```





#PART D (LDA analysis)

```
Auto=read.csv(file="Auto.csv",head=TRUE,na.strings="?",stringsAsFactor = T)
Auto = na.omit(Auto)
dim(Auto)
```

```
mpg=Auto$mpg
year=Auto$year
horsepower=Auto$horsepower
weight=Auto$weight
#summary(Auto)
mpg01=rep(0,392) #Set up a vector of 392 zeroes
mpg01[Auto$mpg > 22.75] = 1 # Switch value for car with over 22.75 mpg to 1
Auto=data.frame(Auto,mpg01) #Add new variable to Auto
set.seed(0) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars

library(MASS)
ldaMod = lda(mpg01~year+weight)
ldaMod
ldaMod.pred=predict(ldaMod,Auto.test)
ldaMod.class=ldaMod.pred$class
table(ldaMod.class,mpg01.test)
(45+45)/100

set.seed(3) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars

library(MASS)
ldaMod = lda(mpg01~year+weight)
ldaMod
ldaMod.pred=predict(ldaMod,Auto.test)
ldaMod.class=ldaMod.pred$class
table(ldaMod.class,mpg01.test)
(47+45)/100

set.seed(4) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars

library(MASS)
ldaMod = lda(mpg01~year+weight)
```

ldaMod
ldaMod.pred=predict(ldaMod,Auto.test)
ldaMod.class=ldaMod.pred$class
table(ldaMod.class,mpg01.test)
(44+50)/100

```
  Call:
  lda(mpg01 ~ year + weight)

  Prior probabilities of groups:
    0   1
  0.5 0.5

  Group means:
        year   weight
  0 74.39796 3620.403
  1 77.56122 2334.765

  Coefficients of linear discriminants:
                  LD1
  year    0.120627452
  weight -0.001670669

              mpg01.test
  ldaMod.class  0  1
             0 45  2
             1  8 45
```

0.9

```
Call:
lda(mpg01 ~ year + weight)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
      year   weight
0 74.39796 3620.403
1 77.56122 2334.765

Coefficients of linear discriminants:
                LD1
year    0.120627452
weight -0.001670669

            mpg01.test
ldaMod.class  0  1
           0 47  2
           1  6 45
```

0.92

```
Call:
lda(mpg01 ~ year + weight)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
      year   weight
0 74.39796 3620.403
1 77.56122 2334.765

Coefficients of linear discriminants:
               LD1
year    0.120627452
weight -0.001670669

            mpg01.test
ldaMod.class  0  1
           0 44  2
           1  4 50
```

0.94

#PART E (QDA ANALYSIS)

```
Auto=read.csv(file="Auto.csv",head=TRUE,na.strings="?",stringsAsFactor = T)
Auto = na.omit(Auto)
dim(Auto)
mpg=Auto$mpg
#summary(Auto)
mpg01=rep(0,392) #Set up a vector of 392 zeroes
mpg01[Auto$mpg > 22.75] = 1 # Switch value for car with over 22.75 mpg to 1
Auto=data.frame(Auto,mpg01) #Add new variable to Auto
set.seed(0) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars

library(MASS)
qdaMod = qda(mpg01~year+weight)
qdaMod
qdaMod.pred=predict(qdaMod,Auto.test)
qdaMod.class=qdaMod.pred$class
table(qdaMod.class,mpg01.test)
(45+44)/100

set.seed(3) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
```

```
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars

library(MASS)
qdaMod = qda(mpg01~year+weight)
qdaMod
qdaMod.pred=predict(qdaMod,Auto.test)
qdaMod.class=qdaMod.pred$class
table(qdaMod.class,mpg01.test)
(46+44)/100

set.seed(4) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars


library(MASS)
qdaMod = qda(mpg01~year+weight)
qdaMod
qdaMod.pred=predict(qdaMod,Auto.test)
qdaMod.class=ldaMod.pred$class
table(qdaMod.class,mpg01.test)
(44+50)/100
```

```
Call:
qda(mpg01 ~ year + weight)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
      year   weight
0 74.39796 3620.403
1 77.56122 2334.765

            mpg01.test
qdaMod.class  0  1
           0 45  3
           1  8 44

0.89
```

```
Call:
qda(mpg01 ~ year + weight)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
      year   weight
0 74.39796 3620.403
1 77.56122 2334.765

            mpg01.test
qdaMod.class  0  1
           0 46  3
           1  7 44

0.9
```

```
Call:
qda(mpg01 ~ year + weight)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
      year   weight
0 74.39796 3620.403
1 77.56122 2334.765

            mpg01.test
qdaMod.class  0  1
           0 44  2
           1  4 50

0.94
```

#PART F (LOGISTIC REGRESSION)

```
Auto = read.csv(file="Auto.csv",head=TRUE,na.strings="?",stringsAsFactors = TRUE)
Auto = na.omit(Auto)
dim(Auto)
mpg=Auto$mpg
weight=Auto$weight
horsepower=Auto$horsepower
year=Auto$year
#summary(Auto)
mpg01=rep(0,392) #Set up a vector of 392 zeroes
mpg01[Auto$mpg > 22.75] = 1 # Switch value for car with over 22.75 mpg to 1
Auto=data.frame(Auto,mpg01) #Add new variable to Auto

#set.seed=0
set.seed(0) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
```

```
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars
glm.fits1=glm(mpg01~weight+year,data=Auto.train,family=binomial)
summary(glm.fits1)
glm.probs1=predict(glm.fits1,Auto.test,type="response")
glm.pred1=rep(0,100)
glm.pred1[glm.probs1>.5]=1
table(glm.pred1,mpg01.test)
(46+44)/100

#set.seed=3
set.seed(3) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars
glm.fits1=glm(mpg01~weight+year,data=Auto.train,family=binomial)
summary(glm.fits1)
glm.probs1=predict(glm.fits1,Auto.test,type="response")
glm.pred1=rep(0,100)
glm.pred1[glm.probs1>.5]=1
table(glm.pred1,mpg01.test)
(47+44)/100

#set.seed=4
set.seed(4) # Choose a seed
train=sample.int(392,292) # Randomly choose 292 of the cars.
Auto.train=Auto[train,] # The rows of Auto for the training cars.
Auto.test=Auto[-train,] # The rows of Auto for the test cars.
mpg01.test=Auto$mpg01[-train] # mpg01 values for the test cars
glm.fits1=glm(mpg01~weight+year,data=Auto.train,family=binomial)
summary(glm.fits1)
glm.probs1=predict(glm.fits1,Auto.test,type="response")
glm.pred1=rep(0,100)
glm.pred1[glm.probs1>.5]=1
table(glm.pred1,mpg01.test)
(45+48)/100
```

```
Call:
glm(formula = mpg01 ~ weight + year, family = binomial, data = Auto.train)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.894e+01  5.329e+00  -3.554  0.00038 ***
weight      -5.030e-03  6.595e-04  -7.627 2.40e-14 ***
year         4.379e-01  8.253e-02   5.306 1.12e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.67  on 291  degrees of freedom
Residual deviance: 129.73  on 289  degrees of freedom
AIC: 135.73

Number of Fisher Scoring iterations: 7

         mpg01.test
glm.pred1  0  1
        0 46  3
        1  7 44
```

0.9

```
Call:
glm(formula = mpg01 ~ weight + year, family = binomial, data = Auto.train)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.579e+01  5.136e+00  -3.074  0.00211 **
weight      -4.928e-03  6.396e-04  -7.705 1.31e-14 ***
year         3.917e-01  7.813e-02   5.013 5.34e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.67  on 291  degrees of freedom
Residual deviance: 133.64  on 289  degrees of freedom
AIC: 139.64

Number of Fisher Scoring iterations: 7

         mpg01.test
glm.pred1  0  1
        0 47  3
        1  6 44
```

0.91

```
Call:
glm(formula = mpg01 ~ weight + year, family = binomial, data = Auto.train)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.035e+01  5.261e+00  -3.868  0.00011 ***
weight      -5.392e-03  7.077e-04  -7.619 2.56e-14 ***
year         4.660e-01  8.250e-02   5.648 1.62e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.74  on 291  degrees of freedom
Residual deviance: 127.30  on 289  degrees of freedom
AIC: 133.3

Number of Fisher Scoring iterations: 7

         mpg01.test
glm.pred1  0  1
        0 45  4
        1  3 48

0.93
```

#PART H (KNN)

```
Auto = read.csv(file="Auto.csv",head=TRUE,na.strings="?",stringsAsFactors = TRUE)
Auto = na.omit(Auto)
dim(Auto)
mpg=Auto$mpg
#summary(Auto)
mpg01=rep(0,392) #Set up a vector of 392 zeroes
mpg01[Auto$mpg > 22.75] = 1 # Switch value for car with over 22.75 mpg to 1
Auto=data.frame(Auto,mpg01) #Add new variable to Auto

#set.seed=0, K=2
set.seed(0) # Choose a seed
scram=sample.int(392, 392) # Randomly scramble 1 through 392
newAuto=Auto[scram,] # Scramble the data set

library(class)

# Define the "scaled" newAuto set.
# Remove 9th column (name) and 10th column (mpg01)
newAutoS=scale(newAuto[,-c(9,10)])
trn=rep(TRUE,292) # Use first 292 rows of newAuto as training set
tst=rep(FALSE,100) # Use last 100 rows of newAuto as test set.
train=c(trn,tst) # Put trn and tst together to form train.

train.X=newAutoS[train,c(5,7)]
```

```
test.X=newAutoS[!train,c(5,7)]

# NewAuto (without the S), since newAutoS does not have mpg01 in it!
train.mpg01=newAuto[train,10]
test.mpg01=newAuto[!train,10]
KNN.pred1=knn(train.X, test.X, train.mpg01, k = 2)
table(KNN.pred1,test.mpg01)
(46+42)/100

#set.seed=0, K=10
set.seed(0)
scram=sample.int(392, 392) # Randomly scramble 1 through 392
newAuto=Auto[scram,] # Scramble the data set
library(class)

# Define the "scaled" newAuto set.
# Remove 9th column (name) and 10th column (mpg01)
newAutoS=scale(newAuto[,-c(9,10)])
trn=rep(TRUE,292) # Use first 292 rows of newAuto as training set
tst=rep(FALSE,100) # Use last 100 rows of newAuto as test set.
train=c(trn,tst) # Put trn and tst together to form train.

train.X=newAutoS[train,c(5,7)]
test.X=newAutoS[!train,c(5,7)]

# NewAuto (without the S), since newAutoS does not have mpg01 in it!
train.mpg01=newAuto[train,10]
test.mpg01=newAuto[!train,10]
KNN.pred1=knn(train.X, test.X, train.mpg01, k = 10)
table(KNN.pred1,test.mpg01)
(48+43)/100

#set.seed=3, K=2
set.seed(3) # Choose a seed
scram=sample.int(392, 392) # Randomly scramble 1 through 392
newAuto=Auto[scram,] # Scramble the data set

library(class)

# Define the "scaled" newAuto set.
# Remove 9th column (name) and 10th column (mpg01)
newAutoS=scale(newAuto[,-c(9,10)])
trn=rep(TRUE,292) # Use first 292 rows of newAuto as training set
tst=rep(FALSE,100) # Use last 100 rows of newAuto as test set.
```

```
train=c(trn,tst) # Put trn and tst together to form train.

train.X=newAutoS[train,c(5,7)]
test.X=newAutoS[!train,c(5,7)]

# NewAuto (without the S), since newAutoS does not have mpg01 in it!
train.mpg01=newAuto[train,10]
test.mpg01=newAuto[!train,10]
KNN.pred1=knn(train.X, test.X, train.mpg01, k = 2)
table(KNN.pred1,test.mpg01)
(44+41)/100

#set.seed=3, K=10
set.seed(3) # Choose a seed
scram=sample.int(392, 392) # Randomly scramble 1 through 392
newAuto=Auto[scram,] # Scramble the data set

library(class)

# Define the "scaled" newAuto set.
# Remove 9th column (name) and 10th column (mpg01)
newAutoS=scale(newAuto[,-c(9,10)])
trn=rep(TRUE,292) # Use first 292 rows of newAuto as training set
tst=rep(FALSE,100) # Use last 100 rows of newAuto as test set.
train=c(trn,tst) # Put trn and tst together to form train.

train.X=newAutoS[train,c(5,7)]
test.X=newAutoS[!train,c(5,7)]

# NewAuto (without the S), since newAutoS does not have mpg01 in it!
train.mpg01=newAuto[train,10]
test.mpg01=newAuto[!train,10]
KNN.pred1=knn(train.X, test.X, train.mpg01, k = 10)
table(KNN.pred1,test.mpg01)
(49+44)/100
```

```
          test.mpg01
KNN.pred1  0  1
        0 46  5
        1  7 42
```

```
0.88
```

```
         test.mpg01
KNN.pred1  0  1
        0 48  4
        1  5 43
```

0.91

```
         test.mpg01
KNN.pred1  0  1
        0 44  6
        1  9 41
```

0.85

```
         test.mpg01
KNN.pred1  0  1
        0 49  3
        1  4 44
```

0.93

7) Bibliography

James, Gareth, et al. *An Introduction to Statistical Learning: With Applications in R*. Springer, 2022.

Brandt, Keith. *Section 4.5: Comparison of Classification Methods*, 2024

Brandt, Keith. *Review off Classification Methods*, 2024