BUS193 Data Mining Case 1

Section 1

Group 8: Caitlyn Blair, Shruti Hardasani, Rainna Sena

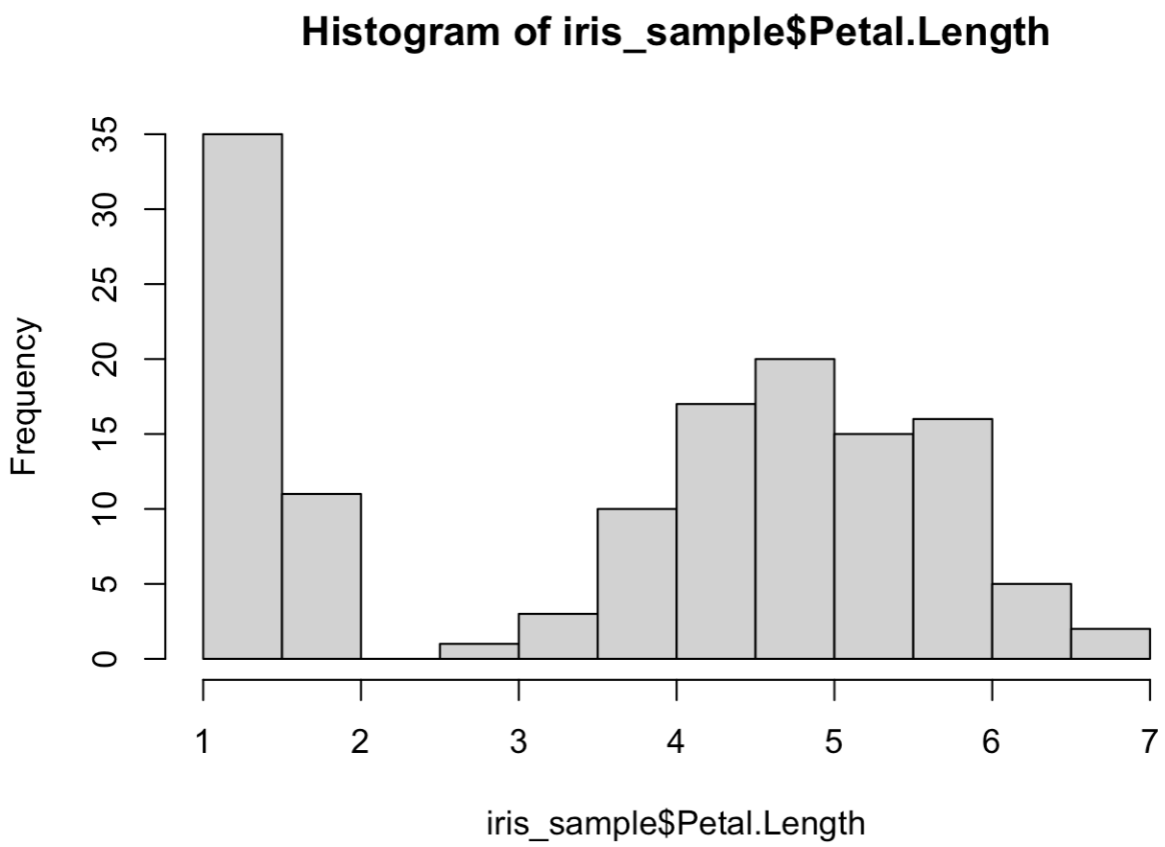
10 September 2024

Figure 1: Summary of the 90% Data Sample

```
> summary(iris_sample)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :46
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.500   1st Qu.:0.300   versicolor:42
 Median :5.800   Median :3.000   Median :4.400   Median :1.300   virginica :47
 Mean   :5.821   Mean   :3.056   Mean   :3.736   Mean   :1.204
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

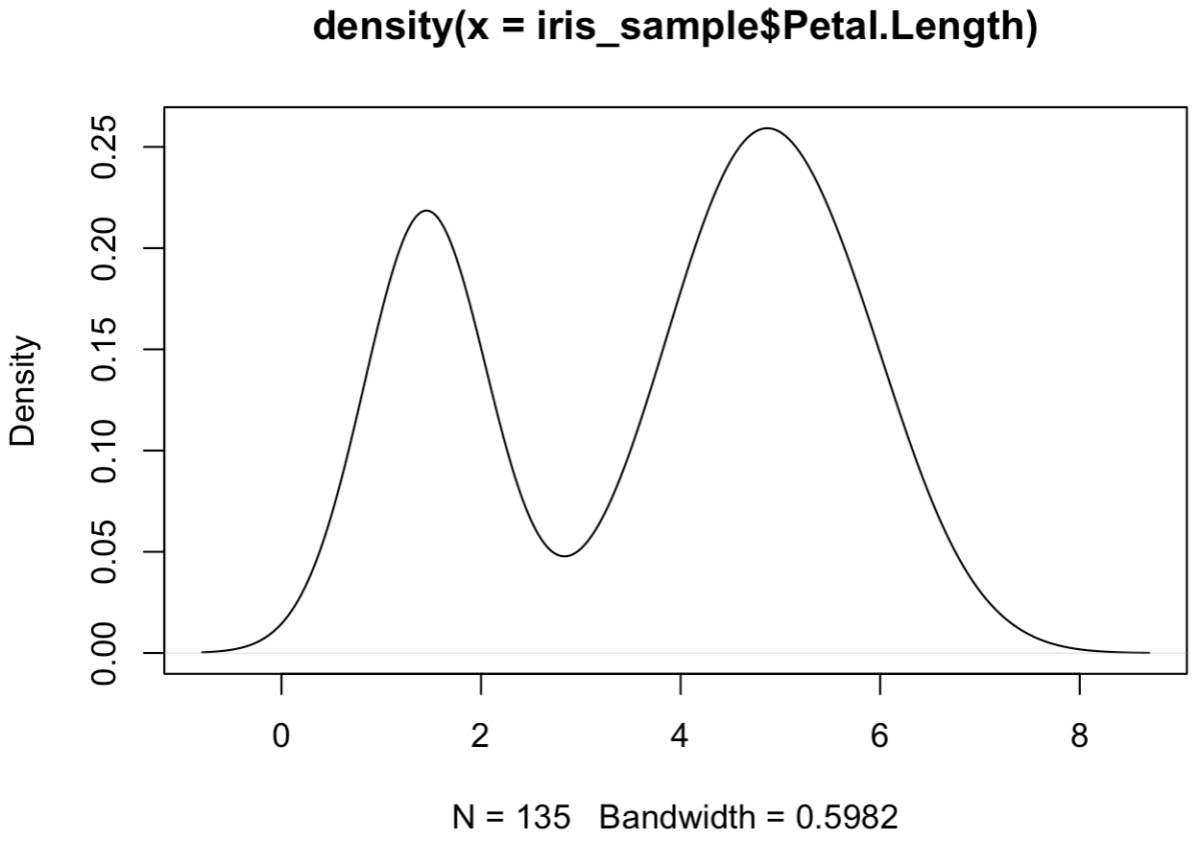
The variables are all quantitative, with the exception of species, so we were able to pull out the summary statistics for sepal length, sepal width, petal length and petal width. The values for sepal length were the highest, as it has a mean of 5.821. For the histogram and scatter plot we decided to stick to the petal length and petal width as our variables. The mean for petal length is 3.736, with a minimum of 1, and maximum of 6.9. The mean for petal width is 1.204, with a minimum of 0.1, and maximum of 2.5

Figure 2: Histogram of Petal Length



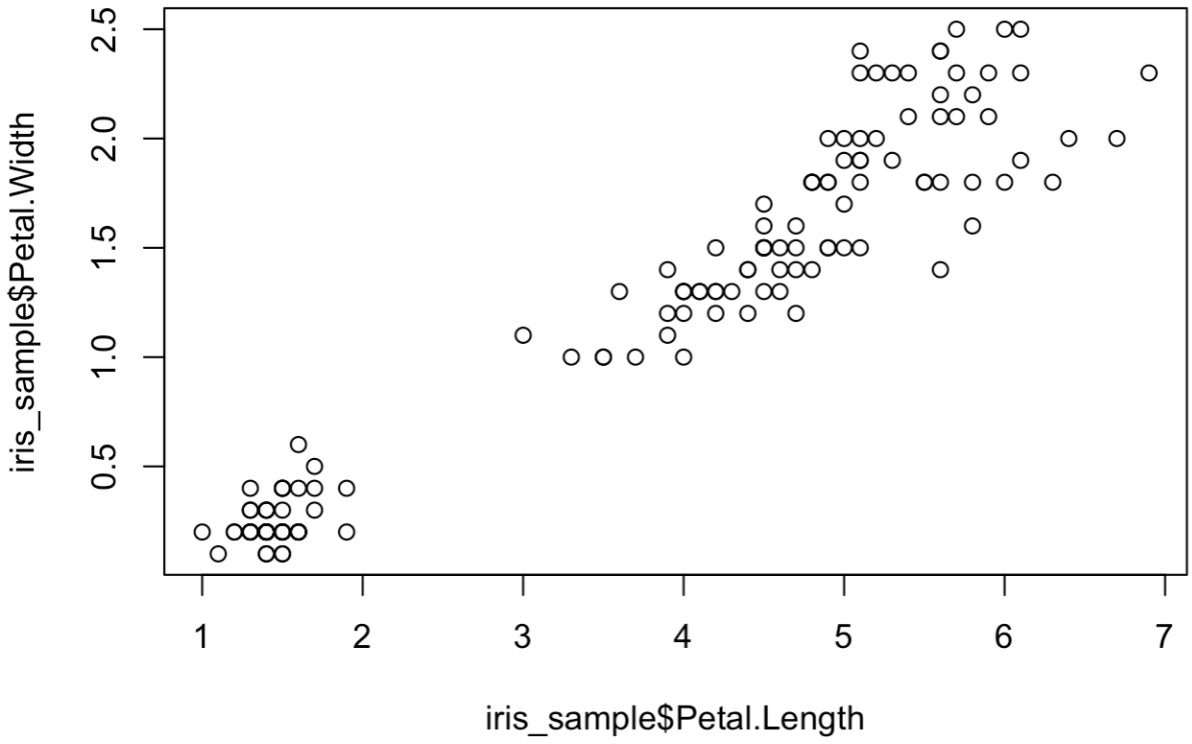
This histogram illustrates a bimodal distribution. This means there are two peaks in the probability frequency. Between 1 centimeter and 2 centimeters consists of the first peak which also includes the first maxima as 35. While the second peak occurs between 2.5 centimeters to 7 centimeters with the maxima being 20.

Figure 3: Density Plot Using Petal Length



This density plot uses the petal length. Similar to the histogram the distribution is bimodal because there are two peaks. The area under the curve is equal to 1. This shows where the majority of the data is concentrated. The higher the curve, the higher the amount of data in that area. We can see that the peaks occur at 2 different petal length values; around 1.7 centimeters and 5 centimeters.

Figure 4: Scatter Plot Using Petal Length



We created a scatter plot using length and width. The output is shown above. There seems to be a positive linear correlation between the petal length and petal width, which means that as petal length increases, petal width increases as well. Most of the data seems to occur as having more than a 4 centimeter petal length and 1 centimeter width. The random sample gave very few outputs of data with petal length between 2-3 centimeters and petal width between 0.5-1 centimeter. There does not seem to be any outliers in the data sample.

CODE:

```
data(iris)
dim(iris)

iris_sample = iris[sample(nrow(iris), nrow(iris) * 0.9), ]
summary(iris_sample)

hist(iris_sample$Petal.Length)

plot(density(iris_sample$Petal.Length))
plot(iris_sample$Petal.Length, iris_sample$Petal.Width)
```

BUS193 Data Mining Case 2

Section 1

Group 8: Caitlyn Blair, Shruti Hardasani, Rainna Sena

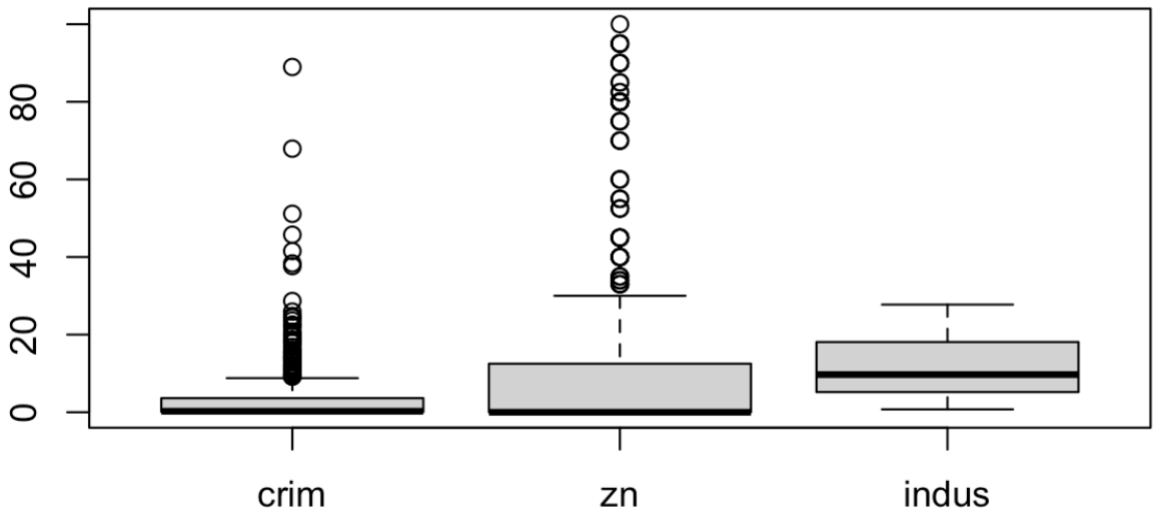
26 September 2024

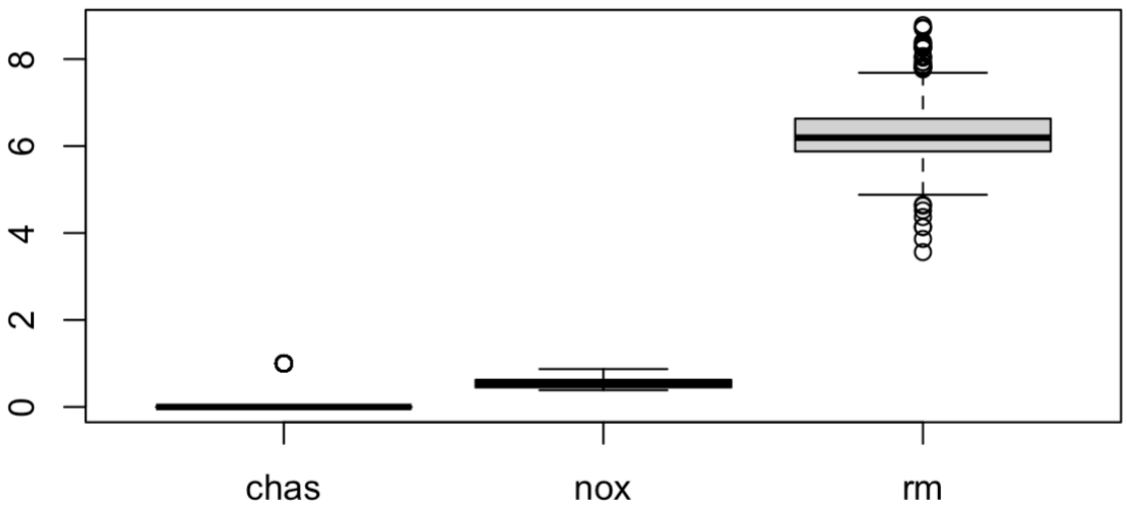
Figure 1: Summary of 90% Training Data

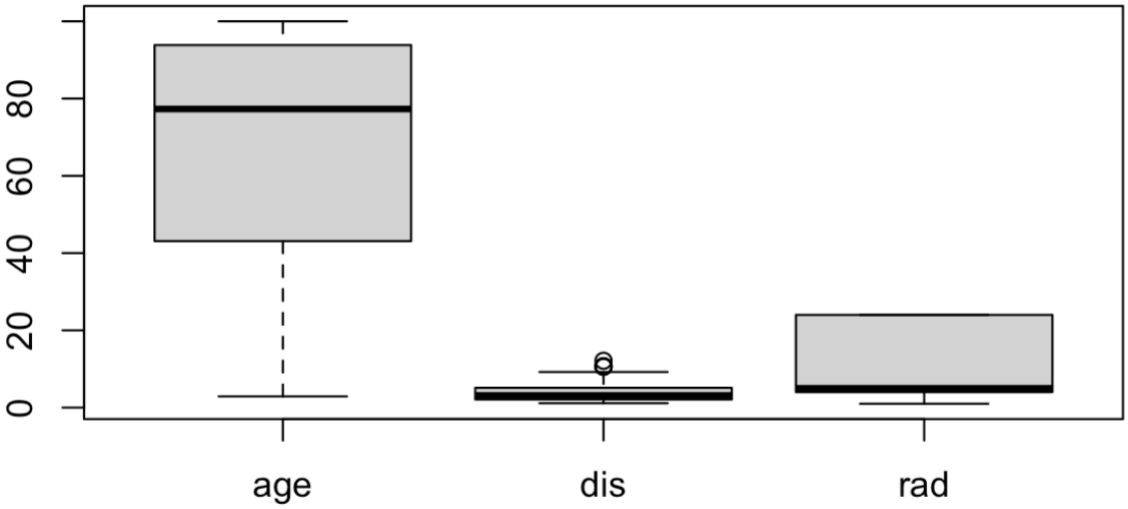
```
> summary(Boston_train)
      crim                zn              indus            chas              nox               rm             age       
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.74   Min.   :0.00000   Min.   :0.3850   Min.   :3.561   Min.   :  2.90  
 1st Qu.: 0.08254   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000   1st Qu.:0.4490   1st Qu.:5.877   1st Qu.: 43.10  
 Median : 0.25915   Median :  0.00   Median : 9.69   Median :0.00000   Median :0.5380   Median :6.193   Median : 77.30  
 Mean   : 3.53938   Mean   : 11.56   Mean   :11.18   Mean   :0.06813   Mean   :0.5557   Mean   :6.277   Mean   : 68.20  
 3rd Qu.: 3.62118   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000   3rd Qu.:0.6240   3rd Qu.:6.630   3rd Qu.: 93.85  
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000   Max.   :0.8710   Max.   :8.780   Max.   :100.00  
      dis              rad              tax             ptratio          black            lstat            medv      
 Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.920   Min.   : 5.00  
 1st Qu.: 2.093   1st Qu.: 4.000   1st Qu.:280.5   1st Qu.:17.40   1st Qu.:375.43   1st Qu.: 6.885   1st Qu.:17.10  
 Median : 3.152   Median : 5.000   Median :330.0   Median :19.00   Median :391.93   Median :11.380   Median :21.40  
 Mean   : 3.784   Mean   : 9.514   Mean   :408.2   Mean   :18.45   Mean   :356.66   Mean   :12.654   Mean   :22.66  
 3rd Qu.: 5.117   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.26   3rd Qu.:16.920   3rd Qu.:25.15  
 Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.970   Max.   :50.00  
```

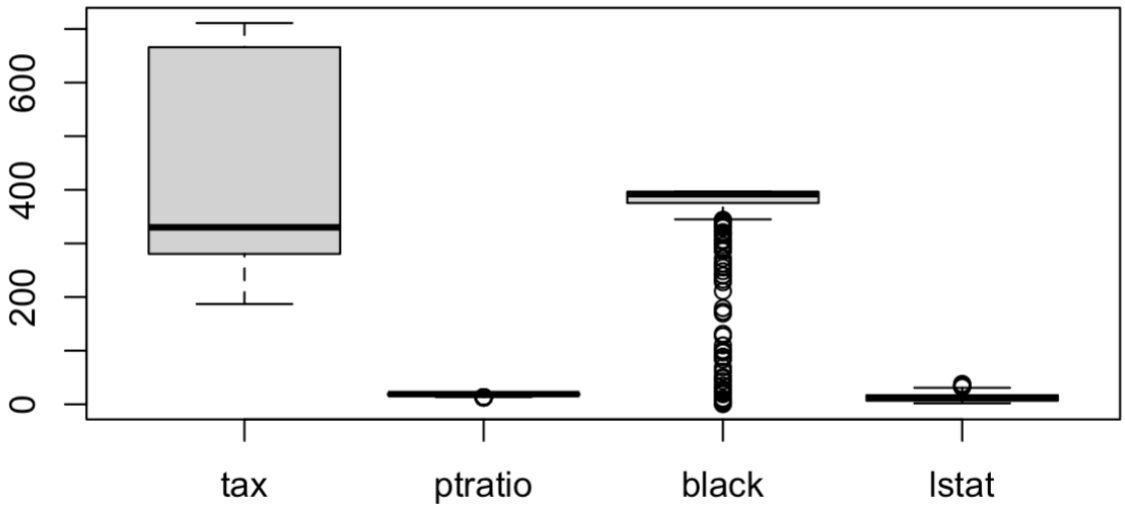
Our summary demonstrates all 14 variables that are quantitative with a 90% training sample. Black has the highest median with 391.93 with tax a close second of 330.0. On the other hand, chas has the lowest with 0 with a close second crim having 0.26. Black also has the highest mean of 356.66

Figure 2: Boxplots of Variables




0
2
4
6
8
chas
nox
rm


0
20
40
60
80
age
dis
rad


0
200
400
600
tax
ptratio
black
lstat

Since there were a lot of variables we decided to clump a few to have the boxplot graphs together to make it easier to read. Indus, age, rad, nox, and tax all don't have any outliers. While crim, zn, chas, rm, ptratio, black, and istat all have outliers. Crim, rm, and black seem to have the most outliers.

Figure 3: Linear Regression of All Independent Variables

```
Coefficients:
(Intercept)         crim           zn        indus         chas
  3.612e+01   -1.130e-01    4.559e-02    3.870e-03    2.592e+00
        nox           rm          age          dis          rad
 -1.788e+01    3.784e+00    5.069e-04   -1.561e+00    2.968e-01
        tax      ptratio        black        lstat
 -1.137e-02   -9.050e-01    9.681e-03   -5.283e-01
```

When we run the linear regression model with every variable in the 90% test set, we get an output as shown above. These variables can be used in a linear regression equation to predict the median home prices with all 13 independent variables. However, this equation is not the best linear model, so we can conduct variable selection to get our optimal model.

Figure 4: Optimal Model

```
medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn +
    rad + tax + crim
```

In order to conduct variable selection to get the optimal model, we ran a forward selection using R. It starts with adding only one variable to the model and continues this process until it gives us an output for the optimal model. This optimal model output is shown above and uses each independent variable except for age and indus. These variables are not necessary in the final optimal model.

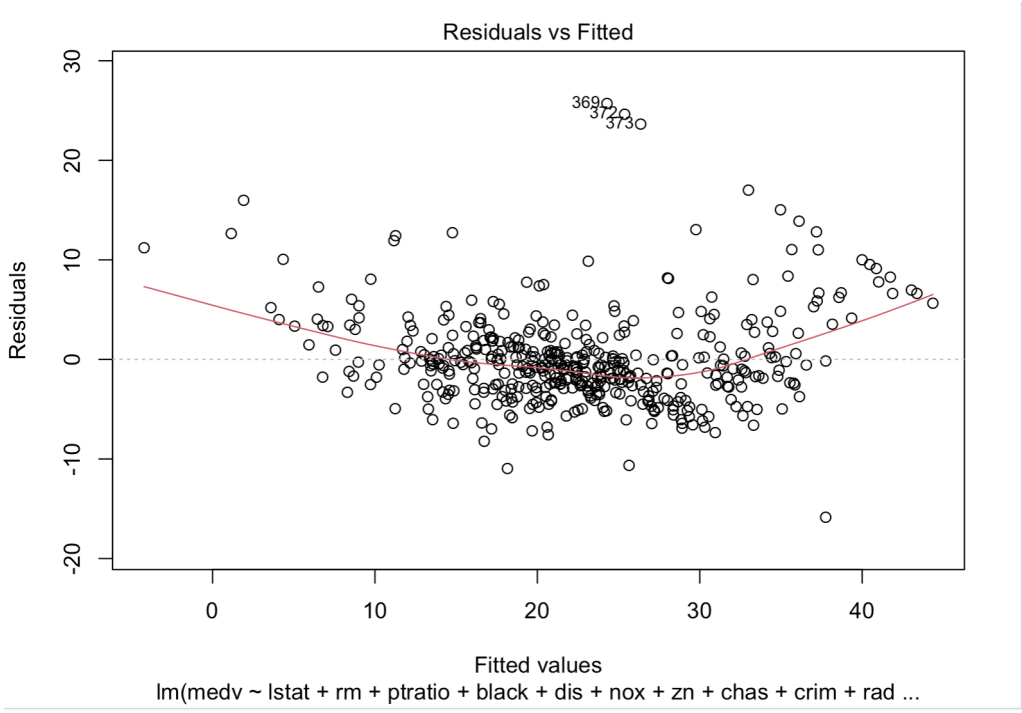
Figure 5: AIC Outputs

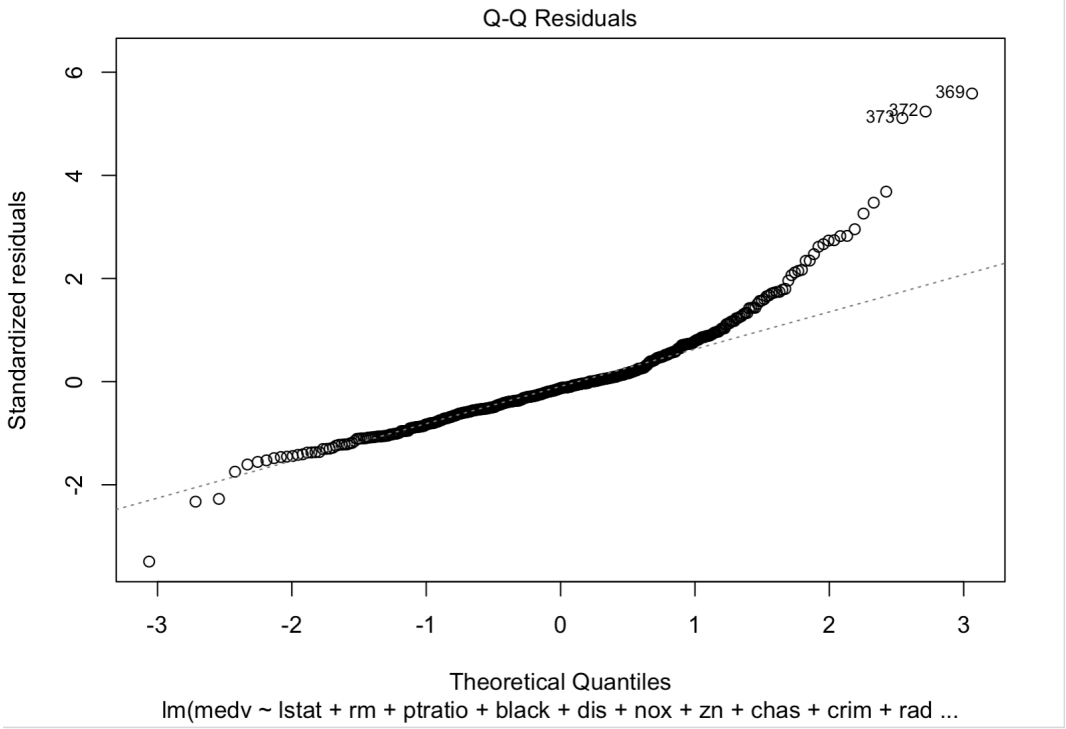
```
> AIC(model_1)
[1] 2726.583
```

```
> AIC(model_opt)
[1] 2722.588
```

To make sure that the optimal model output was better than the original model, we conducted the AIC test. Both outputs are shown above, and we can see that the model 1 AIC is 2726.583 and the optimal model output is 2722.588, which is lower. This means that the forward selection worked and the optimal mode is better than the model 1.

Figure 6: Residual Diagnosis





Both graphs illustrate residual findings and are randomly scattered around zero. While in both graphs there seem to be a few points skewing away from the optimal line. We assume this is normally distributed due to the majority of the points near zero.

Figure 7: Out of Sample Model MSE

```
> mean((pi - Boston_test$medv)^2)
[1] 22.00395
```

The out of sample MSE gives us the mean value of the squared deviations of the Boston test so that we can compare it to the true values, which can be found in the exploratory data analysis. The value of 22.00395 is close to 22.66, which tells us that the sample of 10% of the data does a good job of representing all the values in the Boston dataset .

Figure 8: 10 fold Cross Validation

```
> cv.glm(data = Boston, glmfit = model_2, K = 10)$delta[2]
[1] 23.53149
```

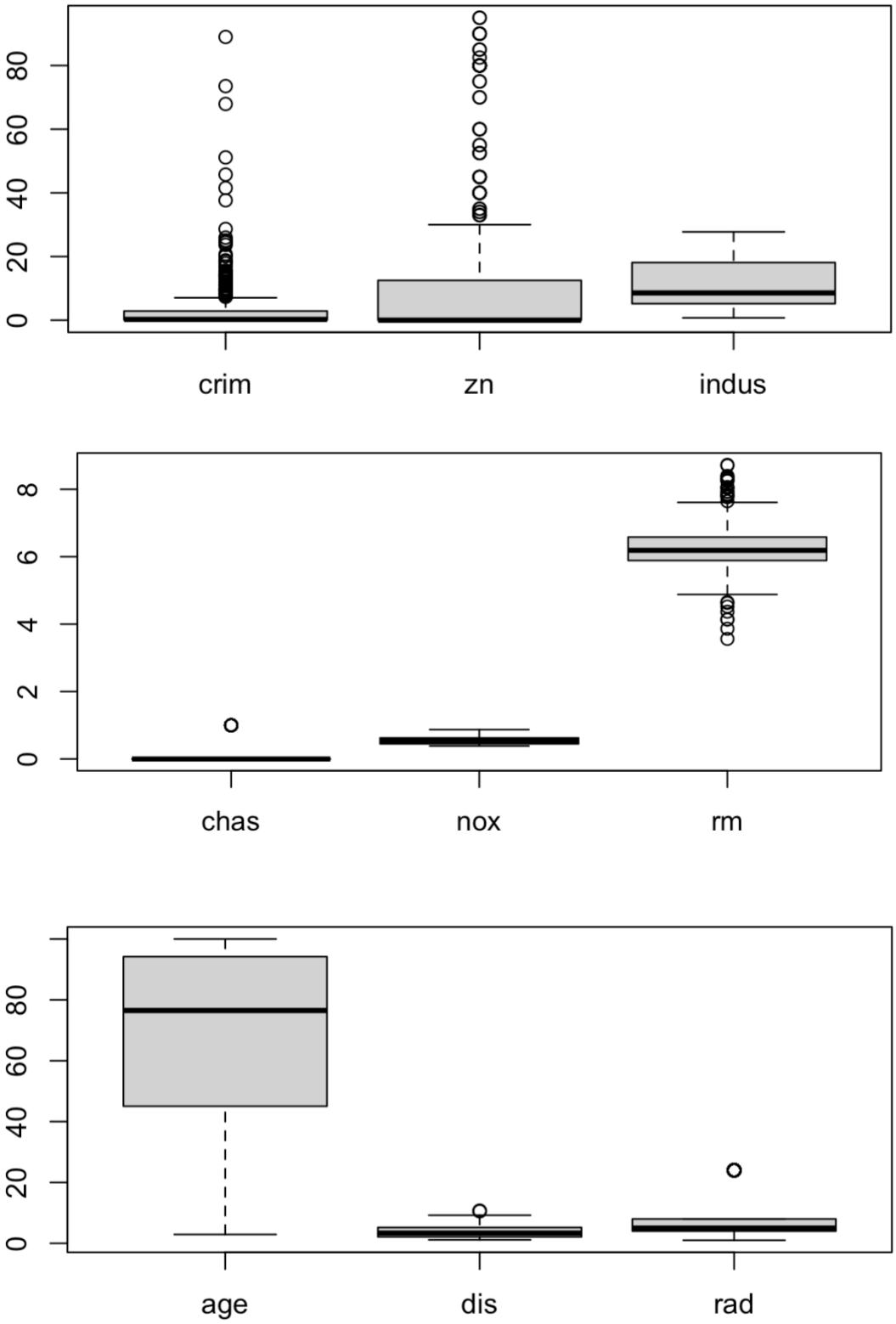
On average, the squared difference between the predicted values and the actual values in the boston dataset is approximately 23.53. This value is close to the out of sample MSE which shows that the model is good, and fits the dataset well.

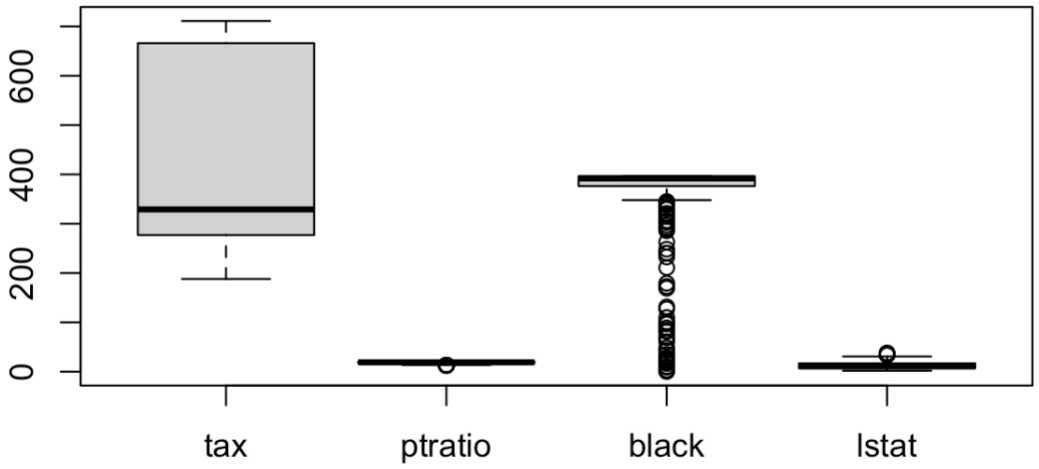
Figure 9: Summary of 80% Training Data

```
> summary(Boston_train2)
      crim                zn              indus             chas              nox               rm             age       
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.74   Min.   :0.00000   Min.   :0.3850   Min.   :3.561   Min.   :  2.90  
 1st Qu.: 0.07958   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000   1st Qu.:0.4480   1st Qu.:5.886   1st Qu.: 45.08  
 Median : 0.24751   Median :  0.00   Median : 8.56   Median :0.00000   Median :0.5350   Median :6.189   Median : 76.50  
 Mean   : 3.56976   Mean   : 11.03   Mean   :11.02   Mean   :0.07178   Mean   :0.5524   Mean   :6.266   Mean   : 68.48  
 3rd Qu.: 2.84478   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000   3rd Qu.:0.6240   3rd Qu.:6.579   3rd Qu.: 94.15  
 Max.   :88.97620   Max.   : 95.00   Max.   :27.74   Max.   :1.00000   Max.   :0.8710   Max.   :8.725   Max.   :100.00  
      dis              rad              tax            ptratio          black            lstat            medv      
 Min.   : 1.130   Min.   : 1.000   Min.   :188.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.920   Min.   : 5.00  
 1st Qu.: 2.111   1st Qu.: 4.000   1st Qu.:277.0   1st Qu.:17.40   1st Qu.:376.46   1st Qu.: 7.093   1st Qu.:17.18  
 Median : 3.373   Median : 5.000   Median :329.0   Median :19.05   Median :391.88   Median :10.925   Median :21.40  
 Mean   : 3.827   Mean   : 9.302   Mean   :402.3   Mean   :18.46   Mean   :357.60   Mean   :12.609   Mean   :22.46  
 3rd Qu.: 5.213   3rd Qu.: 8.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.31   3rd Qu.:16.780   3rd Qu.:25.00  
 Max.   :10.710   Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.970   Max.   :50.00  
```

In this summary we used an 80% training sample with the same 14 variables that are quantitative. Once again black has the highest median of 391.88 with tax still coming in second at 329.0. This time tax has the highest mean of 402.3 and black coming in second at 357.60.

Figure 10: Summary of 90% Training Data





As we did in the first testing we bunched up a few variables at a time for our boxplots so they are easier to see and analyze. Indus, nox, age, and tax still don't have any outliers. While on the other hand, crim, zn, chas, rm dis, rad, ptratio, black, and istat have outliers. In our first testing, rad didn't have any outliers but now it does. Crim, rm, and black seem to have the most outliers in these boxplots.

Figure 11: Summary of 90% Training Data

```
Coefficients:
(Intercept)         crim           zn        indus         chas
  3.594e+01   -1.008e-01    5.190e-02    3.054e-02    3.010e+00
        nox           rm          age          dis          rad
 -1.817e+01    3.707e+00    7.052e-04   -1.506e+00    3.214e-01
        tax      ptratio        black        lstat
 -1.171e-02   -9.294e-01    1.033e-02   -5.181e-01
```

When running a linear regression model with each independent variable of the 80% training sample, we can see that the outputs are fairly similar to the original 90% training sample we conducted. The biggest difference in value is that the age was 5.069e-04 in the 90% sample, and it is 7.052e-04 in this sample. Other than that, the values are very similar.

Figure 12: Optimal Model of 80% Training Sample

```
medv ~ lstat + rm + ptratio + chas + dis + nox + black + zn +
    rad + crim + tax
```

To get our optimal model for the 80% training sample, we used the forward selection again. This gave us the exact same output as the 90% training sample, using all independent variables except age and indus.

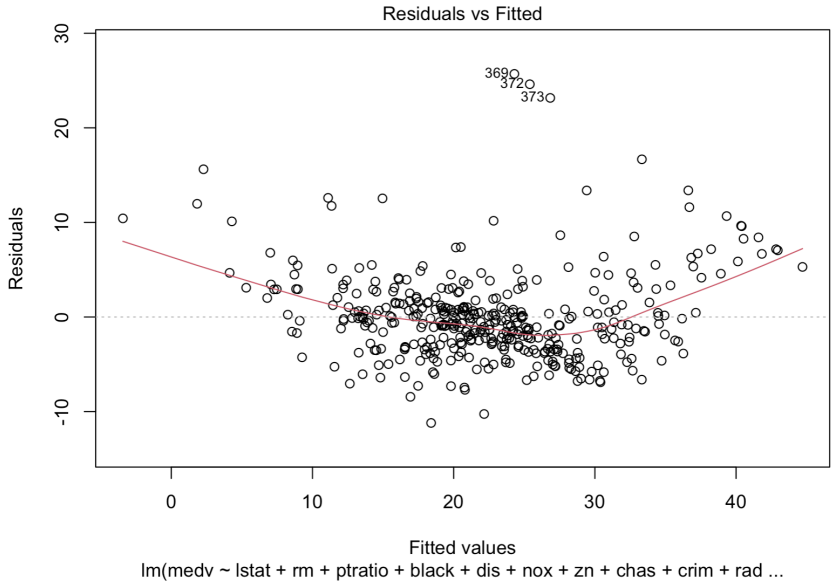
Figure 13: AIC of 80% Training Sample

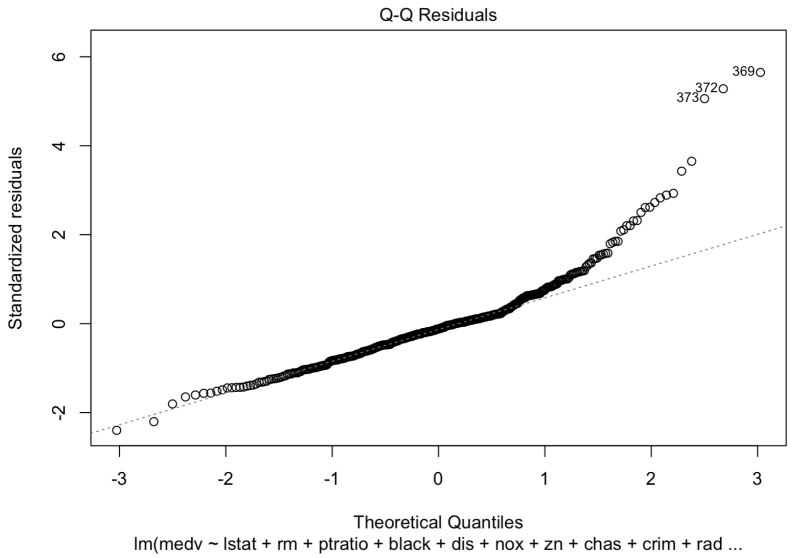
```
> AIC(model_3)
[1] 2416.984
```

```
> AIC(model_opt2)
[1] 2413.195
```

In order to make sure the optimal model given is truly better than the model that contained all independent variables, we ran the AIC again. The output of the original model was 2416.984, while the optimal model was 2413.195. This shows that our optimal model is the best fit model for a linear regression model.

Figure 14: Residual Analysis of 80% Training Sample





As illustrated in our first run of the case these graphs show very similar results. With the majority of the points on or near the zero line. So we assume this is normally distributed.

Figure 15: Out of Sample MSE on 80% Testing Sample

```
> mean((pi - Boston_test2$medv)^2)
[1] 23.8328
```

This test tells us that 80% of the testing sample provides us a good insight, and ensures that the model generalized all of the data well.

Figure 16: 10 Fold Cross Validation on All Data

```
> cv.glm(data = Boston, glmfit = model_4, K = 10)$delta[2]
[1] 23.2882
```

The value of 23.2882 is very close to the value we got from the out-of-sample testing, which leads us to conclude that using 80% data gives us a good representation of all of the data.

In conclusion, we have found that the optimal model for both samples gives the same results, which exclude age and indus in the linear regression model. The out of sample MSE and 10 fold cross validation for both sets are very close which show that our models are very good and fit the dataset well. All other outputs were similar which is to be expected since both the 90% set and 80% set were taken from the same Boston dataset.

Boston Housing data.
Random sample a training data set that contains 90% of the original data points.
(i) Start with exploratory data analysis. Are there outliers?
(ii) Conduct linear regression on the training data.
(iii) Conduct variable selection. Find the best linear model. Show residual diagnosis.
(iv) Test the out-of-sample performance. Using final linear model built from (iii) on the 90% of original data, test with the remaining 10% testing data. Report out-of-sample model MSE etc.
(v) Cross validation on the original data. Use 10-fold cross validation. Does (v) yield similar answer as (iv)?
(vi) Now repeat previous steps for another random sample (that is, to draw another training data set with 90% of original data, and the rest 10% as testing; or you can try 80% training vs. 20% left as testing). Do you get similar results? What's your conclusion?
Write a brief report including all labeled figures and tables.

```
library(MASS)
data(Boston);

#for the training and test sets
subset <- sample(nrow(Boston), nrow(Boston) * 0.9)
Boston_train = Boston[subset, ]
Boston_test = Boston[-subset, ]



#to make box plot and test for outliers
boxplot(Boston_train[,1:3])
boxplot(Boston_train[,4:6])
boxplot(Boston_train[,7:9])
boxplot(Boston_train[,10:13])



#to run linear regression with all variables
model_1 <- lm(medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +
              black + lstat, data = Boston_train)
model_1
```

```
#to find best model (using forward)
nullmodel = lm(medv ~ 1, data = Boston_train)
fullmodel = lm(medv ~ ., data = Boston_train)
model.step <- step(nullmodel, scope = list(lower = nullmodel,
                                   upper = fullmodel),
                   direction = "forward")

#optimal model
model_opt <- lm(medv ~ lstat + rm + ptratio + black + dis + nox + zn + chas +
                  crim + rad + tax, data = Boston_train)

model_1 <- lm(medv ~ ., data = Boston_train)
AIC(model_1)
AIC(model_opt) #optimal model AIC is better

BIC(model_1)
BIC(model_opt) #optimal model BIC is better too

#residual diagnosis, only use Residuals vs Fitted and Q-Q Residuals
plot(model_opt)


# pi is a vector that contains predicted values for test set.
pi <- predict(object = model_opt, newdata = Boston_test)
pi

mean((pi - Boston_test$medv)^2) #MSE of test set

library(boot)
model_2 = glm(medv ~ . - age - indus, data = Boston) #optimal model
cv.glm(data = Boston, glmfit = model_2, K = 10)$delta[2]


#for the second training and test set (using 0.8)
subset2 <- sample(nrow(Boston), nrow(Boston) * 0.8)
Boston_train2 = Boston[subset2, ]
Boston_test2 = Boston[-subset2, ]

#to make box plot and test for outliers
```

```
boxplot(Boston_train2[,1:3])
boxplot(Boston_train2[,4:6])
boxplot(Boston_train2[,7:9])
boxplot(Boston_train2[,10:13])


#to run linear regression with all variables
model_3 <- lm(medv ~ ., data = Boston_train2)
model_3

#to find best model (using forward)
nullmodel2 = lm(medv ~ 1, data = Boston_train2)
fullmodel2 = lm(medv ~ ., data = Boston_train2)
model.step2 <- step(nullmodel2, scope = list(lower = nullmodel2,
                                    upper = fullmodel2),
                    direction = "forward")

#optimal model
model_opt2 <- lm(medv ~ lstat + rm + ptratio + black + dis + nox + zn + chas +
                   crim + rad + tax, data = Boston_train2)

AIC(model_3)
AIC(model_opt2)

BIC(model_3)
BIC(model_opt2)

#residual diagnosis, only use Residuals vs Fitted and Q-Q Residuals
plot(model_opt2)


# pi is a vector that contains predicted values for test set.
pi <- predict(object = model_opt2, newdata = Boston_test2)
pi

mean((pi - Boston_test2$medv)^2) #MSE of test set

library(boot) #for 10 fold cross validation
model_4 = glm(medv ~ . - age - indus, data = Boston) #optimal model
cv.glm(data = Boston, glmfit = model_4, K = 10)$delta[2]
```

BUS193 Data Mining Case 3

Section 1

Group 8: Caitlyn Blair, Shruti Hardasani, Rainna Sena

15 October 2024

Figure 1: EDA of all Variables

```
> summary(credit_train) #for EDA
   chk_acct            duration      credit_his          purpose             amount      saving_acct
 Length:800         Min.   : 4.00   Length:800         Length:800         Min.   :  250   Length:800
 Class :character   1st Qu.:12.00   Class :character   Class :character   1st Qu.: 1374   Class :character
 Mode  :character   Median :18.00   Mode  :character   Mode  :character   Median : 2309   Mode  :character
                    Mean   :20.93                                         Mean   : 3293
                    3rd Qu.:24.00                                         3rd Qu.: 4010
                    Max.   :72.00                                         Max.   :18424
 present_emp        installment_rate     sex             other_debtor       present_resid    property
 Length:800         Min.   :1.000    Length:800         Length:800         Min.   :1.000   Length:800
 Class :character   1st Qu.:2.000    Class :character   Class :character   1st Qu.:2.000   Class :character
 Mode  :character   Median :3.000    Mode  :character   Mode  :character   Median :3.000   Mode  :character
                    Mean   :2.987                                         Mean   :2.825
                    3rd Qu.:4.000                                         3rd Qu.:4.000
                    Max.   :4.000                                         Max.   :4.000
      age         other_install        housing            n_credits         job              n_people
 Min.   :19.00   Length:800         Length:800         Min.   :1.000   Length:800         Min.   :1.000
 1st Qu.:27.00   Class :character   Class :character   1st Qu.:1.000   Class :character   1st Qu.:1.000
 Median :33.00   Mode  :character   Mode  :character   Median :1.000   Mode  :character   Median :1.000
 Mean   :35.54                                         Mean   :1.414                      Mean   :1.157
 3rd Qu.:42.00                                         3rd Qu.:2.000                      3rd Qu.:1.000
 Max.   :75.00                                         Max.   :4.000                      Max.   :2.000
  telephone           foreign             response
 Length:800         Length:800         Min.   :0.0000
 Class :character   Class :character   1st Qu.:0.0000
 Mode  :character   Mode  :character   Median :0.0000
                                       Mean   :0.3013
                                       3rd Qu.:1.0000
                                       Max.   :1.0000
```

Illustrated above are the EDAs for the variables. One can see that age has the highest median with 33 and response has the lowest median of 0. Age seems to stick out the most due to it having the highest mean of 35.54 while once again response has the lowest mean of 0.30. While on average present_resid and installament_rate seem to have very similar outputs.

Figure 2: Boxplot of Duration

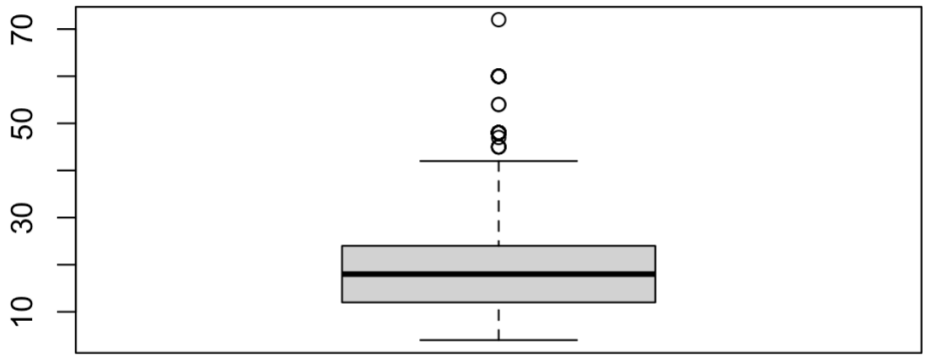


Figure 3: Boxplot of Amount

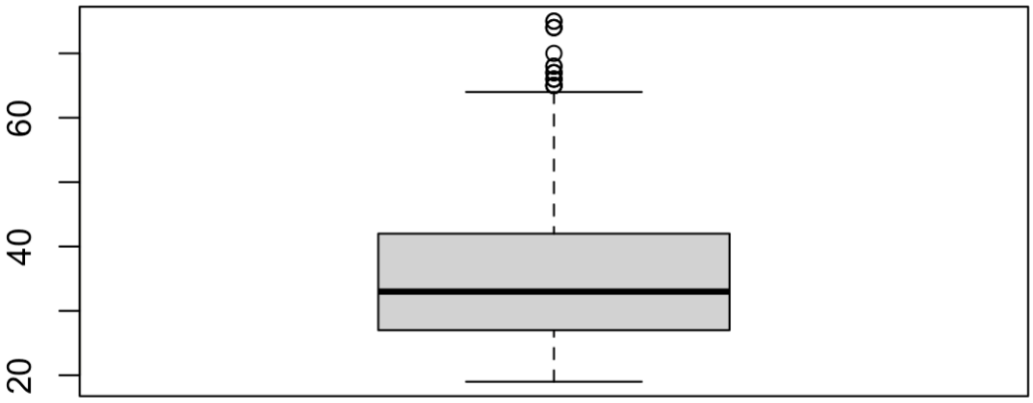


Figure 4: Boxplot of Installment Rate

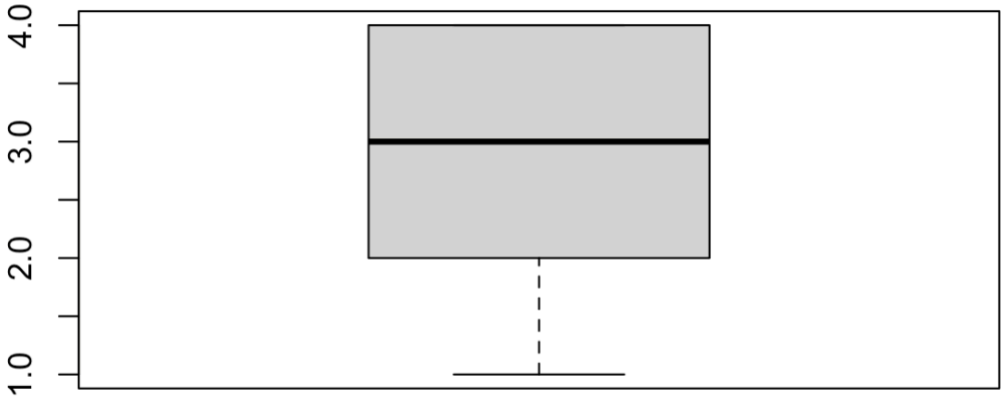


Figure 5: Boxplot of Present Resid

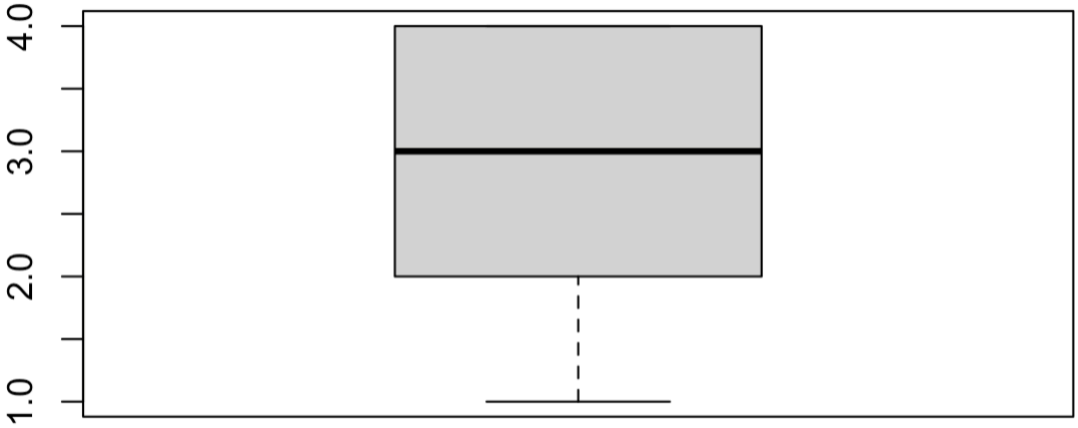


Figure 6: Boxplot of Age

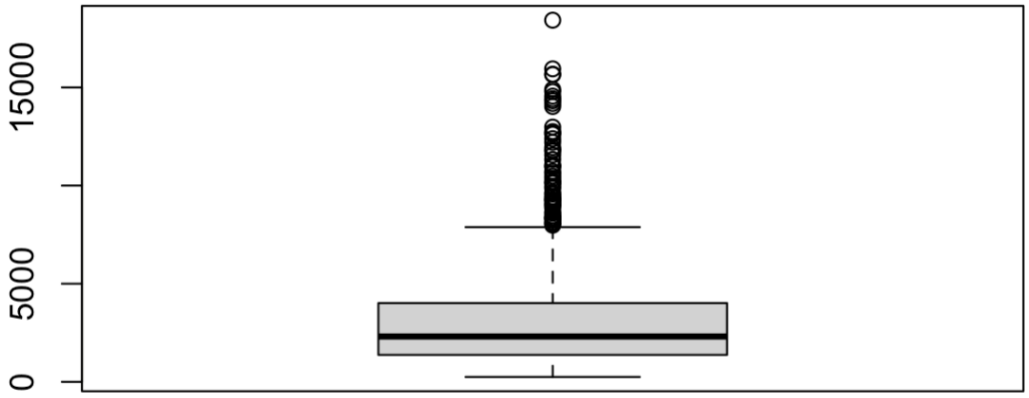


Figure 7: Boxplot of Number of Credits

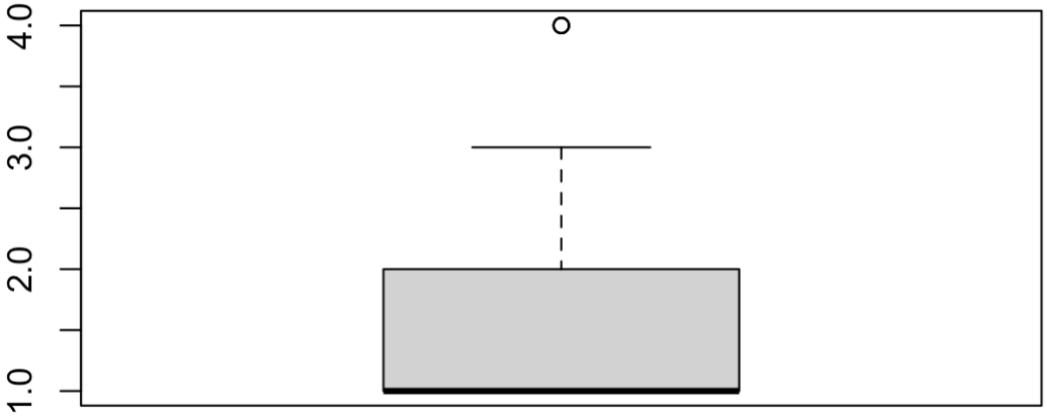
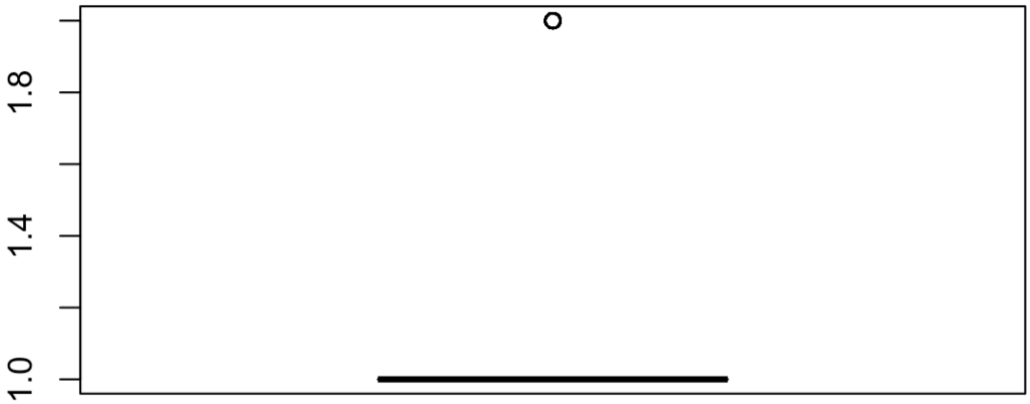


Figure 8: Boxplot of Number of People



These are our 7 box plots on the numerical values and these help illustrate any outliers that occur in our data. Duration, amount, and age seem to have many outliers. While, the number of credits and number of people seem to only have one outlier each. Furthermore, installment rate and present resid don't have any outliers.

Figure 9: Logistic Regression with all Variables

```
Call:  glm(formula = response ~ ., family = binomial, data = credit_train)

Coefficients:
      (Intercept)        chk_acctA12        chk_acctA13        chk_acctA14
        0.2279867         -0.3610481         -0.6988782         -1.6918778
         duration      credit_hisA31      credit_hisA32      credit_hisA33
        0.0282812          0.3997571         -0.4850622         -0.6445315
    credit_hisA34        purposeA41        purposeA410        purposeA42
       -1.4061109         -1.7596391         -0.7105172         -0.9162560
       purposeA43        purposeA44        purposeA45        purposeA46
       -0.9395669         -0.7353854         -0.1076252          0.2286839
       purposeA48        purposeA49            amount     saving_acctA62
       -1.7409732         -0.6465588          0.0001601         -0.3297304
   saving_acctA63     saving_acctA64     saving_acctA65     present_empA72
       -0.1863021         -1.0216132         -1.0761290         -0.3601515
   present_empA73     present_empA74     present_empA75   installment_rate
       -0.4932166         -1.0948069         -0.6853224          0.3947359
           sexA92             sexA93             sexA94   other_debtorA102
       -0.5616370         -0.8809145         -0.5556147          0.5266139
 other_debtorA103      present_resid       propertyA122       propertyA123
       -0.9957885          0.0842577         -0.0287477         -0.0164821
     propertyA124                age  other_installA142  other_installA143
        0.6181106         -0.0182687          0.0366844         -0.5640281
      housingA152        housingA153          n_credits            jobA172
       -0.5280988         -0.8446922          0.2428216          0.6451809
          jobA173            jobA174           n_people      telephoneA192
        0.8078773          0.5955008          0.3792197         -0.3664514
      foreignA202
       -1.4543769
```

When we run logistic regression on all the variables, the coefficients of the model are as shown above. This includes all independent variables so it is not likely to be the best model. We can run a variable selection to see which of these independent variables are significant enough to be used in an optimal model.

Figure 10: Histogram of In Sample Prediction

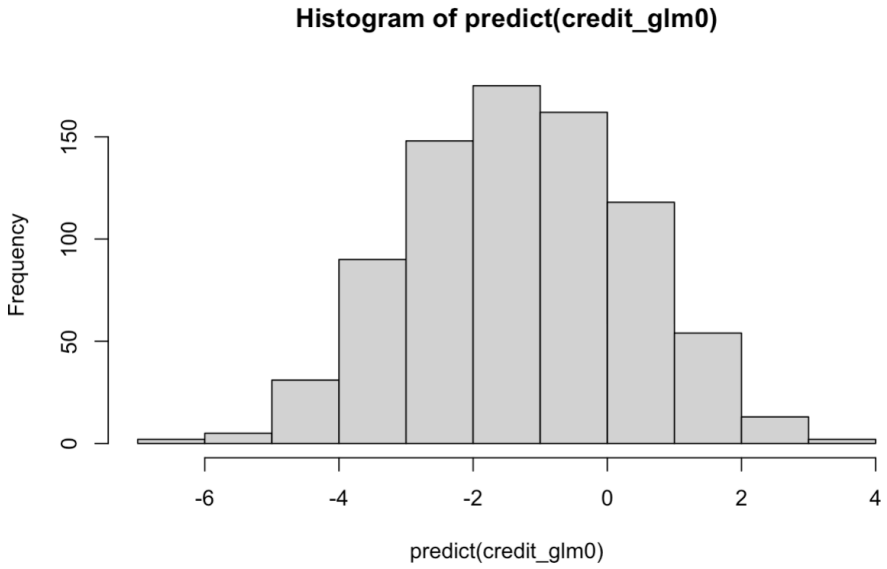
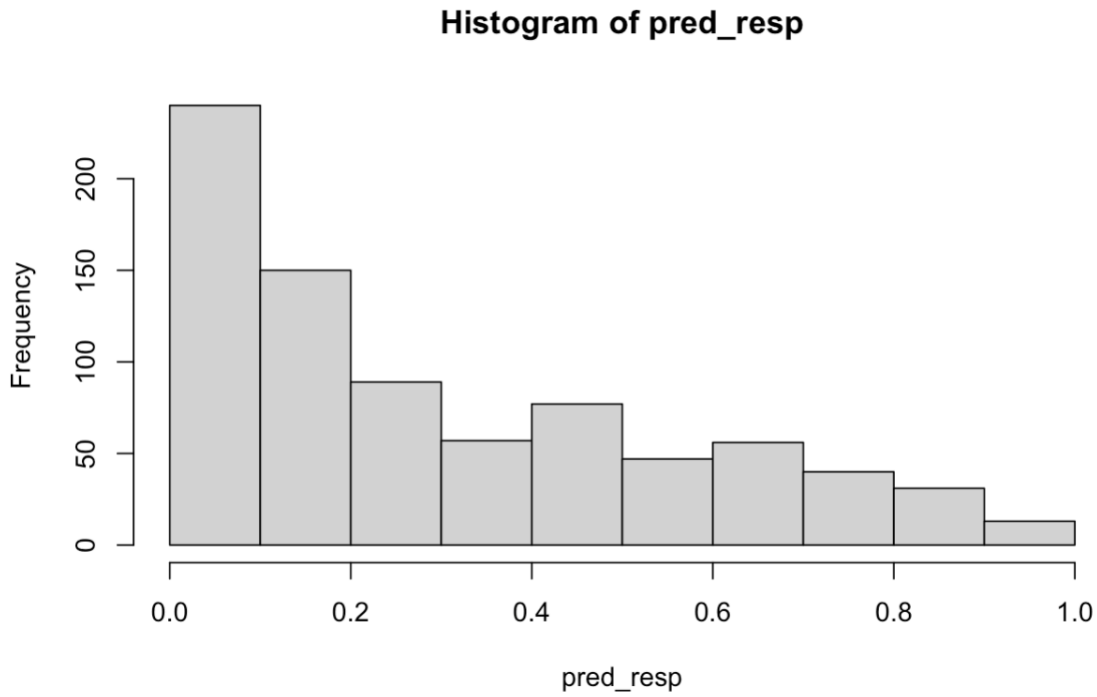


Figure 11: Histogram of In Sample Predicted Probability



The first histogram up above illustrates our in-sample prediction. This graph seems to be symmetrical with it peaking way above 150. Our second histogram seems to show a right skew. The highest data is peaked on the right side of the graph with it declining to the left as it moves on.

Figure 12: Model Fitting and Coefficients

```
Step:  AIC=-1443.56
response ~ chk_acct + duration + credit_his + purpose + other_debtor +
    saving_acct + installment_rate + housing + amount + foreign +
    telephone + age

Coefficients:
     (Intercept)       chk_acctA12       chk_acctA13       chk_acctA14          duration      credit_hisA31
       0.5706583        -0.2921744        -0.7419796        -1.6312558         0.0278560         0.3763343
   credit_hisA32     credit_hisA33     credit_hisA34       purposeA41       purposeA410       purposeA42
      -0.7825334        -0.8061121        -1.5438256        -1.7105474        -0.4164949        -0.8350548
      purposeA43        purposeA44        purposeA45        purposeA46        purposeA48        purposeA49
      -0.9815471        -0.7243905        -0.1562888         0.2937496        -1.9083756        -0.6274456
other_debtorA102  other_debtorA103   saving_acctA62    saving_acctA63    saving_acctA64    saving_acctA65
       0.5798777        -1.0240365        -0.3196016        -0.3014763        -0.8789751        -1.0956361
installment_rate       housingA152       housingA153            amount       foreignA202    telephoneA192
       0.3556103        -0.6118211        -0.3304688         0.0001428        -1.4232208        -0.3896352
               age
      -0.0143291
```

When we run a forward selection on the logistic regression model, the output is as shown above including 12 of the independent variables with their coefficients. The model omits 8 variables which are other_install, present_emp, n_credits, n_people, present_resid, sex, property, and job.

Figure 13: AIC of Both Models

```
> AIC(credit_glm0)
[1] 801.2242
```

```
> AIC(model_opt)
[1] 788.8455
```

In order to compare which model is better, we decided to use AIC as our criteria. The AIC of the model including all independent variables is 801.22. The AIC of the optimal model is 788.84. This means that the optimal model is a better fit because the AIC is lower than the original model.

Figure 14: ROC Curve for In Sample

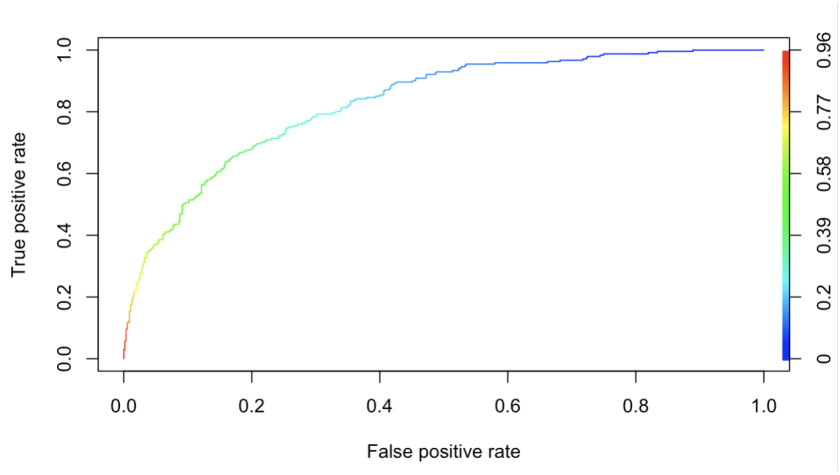


Figure 15: AUC of In Sample

```
> unlist(slot(performance(pred, "auc"), "y.values"))
[1] 0.8283909
```

The model has an AUC which is higher than 0.7, letting us know that the model can fairly distinguish between positive and negative cases.

Figure 16: Misclassification Table of Optimal Model (In Sample)

	Predicted	
Truth	0	1
0	502	57
1	119	122

For every misclassification table shown, we applied a cutoff value of 0.5. Given these values, we were able to calculate the miscalculation rate, MR = (57 + 119) / (502 + 57 + 119 + 122) = 0.22.

Figure 17: AUC of Out of Sample

```
> unlist(slot(performance(pred, "auc"), "y.values"))
[1] 0.7889169
```

The out of sample has an AUC higher than 0.7, making it a good model but not the optimal one since the in sample has a larger value.

Figure 18: ROC Curve of Out of Sample

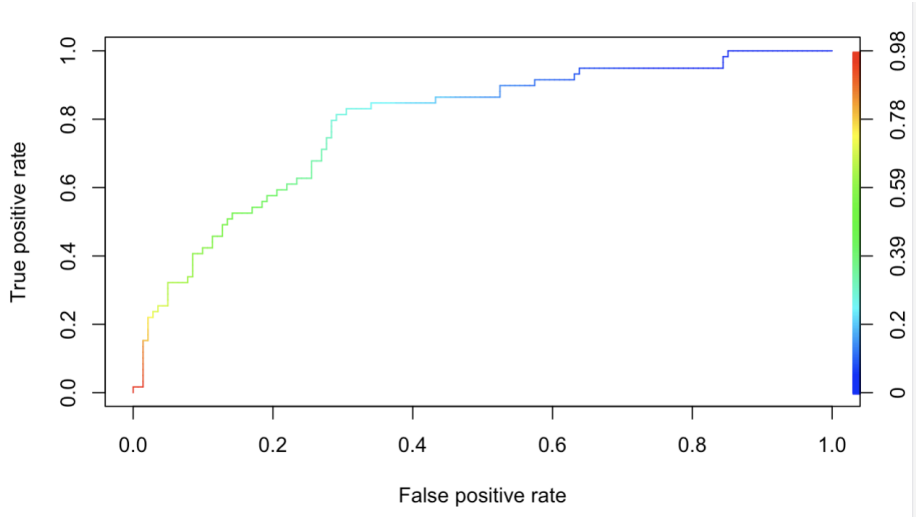


Figure 19: Misclassification Table of Out of Sample

Truth	Predicted 0	1
0	122	19
1	29	30

Once again, using a cutoff value of 0.5, above is the results for the misclassification table, MR = (19 + 29) / (122 + 19 + 29 + 30) = 0.24 are higher than the in sample so we get to a conclusion that this model makes more incorrect predictions leading to poor model performance.

Figure 20: Result of 5 Fold Cross Validation

```
> cv_result$delta[2]
[1] 0.867
```

On average, the squared difference between the predicted values and the actual values in the German credit score data is 0.867. It gives us an indication of the model's predictive accuracy, and since it is low we can say that the model is good, and fits the dataset well.

Figure 21: EDA of New 80% Training Set

```
> summary(credit_train2) #for EDA
   chk_acct            duration     credit_his          purpose             amount      saving_acct
 Length:800         Min.   : 4.0   Length:800         Length:800         Min.   :  250   Length:800
 Class :character   1st Qu.:12.0   Class :character   Class :character   1st Qu.: 1380   Class :character
 Mode  :character   Median :18.0   Mode  :character   Mode  :character   Median : 2330   Mode  :character
                    Mean   :21.1                                         Mean   : 3290
                    3rd Qu.:24.0                                         3rd Qu.: 3974
                    Max.   :60.0                                         Max.   :18424
 present_emp        installment_rate     sex            other_debtor       present_resid     property
 Length:800         Min.   :1.000    Length:800         Length:800         Min.   :1.000   Length:800
 Class :character   1st Qu.:2.000    Class :character   Class :character   1st Qu.:2.000   Class :character
 Mode  :character   Median :3.000    Mode  :character   Mode  :character   Median :3.000   Mode  :character
                    Mean   :2.985                                         Mean   :2.836
                    3rd Qu.:4.000                                         3rd Qu.:4.000
                    Max.   :4.000                                         Max.   :4.000
      age        other_install        housing           n_credits         job              n_people
 Min.   :19.0   Length:800         Length:800         Min.   :1.000   Length:800         Min.   :1.000
 1st Qu.:27.0   Class :character   Class :character   1st Qu.:1.000   Class :character   1st Qu.:1.000
 Median :33.0   Mode  :character   Mode  :character   Median :1.000   Mode  :character   Median :1.000
 Mean   :35.5                                         Mean   :1.389                      Mean   :1.153
 3rd Qu.:42.0                                         3rd Qu.:2.000                      3rd Qu.:1.000
 Max.   :75.0                                         Max.   :4.000                      Max.   :2.000
  telephone          foreign            response
 Length:800         Length:800         Min.   :0.0000
 Class :character   Class :character   1st Qu.:0.0000
 Mode  :character   Mode  :character   Median :0.0000
                                       Mean   :0.3137
                                       3rd Qu.:1.0000
                                       Max.   :1.0000
```

Here is our second run of EDA's for the variables. Age seems to still be producing the highest median (33) and mean (35.5), with response still having the lowest median (0) and mean (0.31). In addition, installmen_rate and present_resid have similar outputs with the exact same medians (3) and means (2.9).

In our first run we included images of 7 boxplots. For this second run our boxplots were exactly the same so the images aren't included. What we found was duration, amount, and age seem to have many outliers. While, the number of credits and number of people seem to only have one outlier each. Furthermore, installment rate and present resid don't have any outliers.

Figure 22: Logistic Regression with All Variables

```
Call:  glm(formula = response ~ ., family = binomial, data = credit_train2)

Coefficients:
      (Intercept)       chk_acctA12       chk_acctA13       chk_acctA14          duration      credit_hisA31
        -0.9919203        -0.3246686        -1.1436151        -1.6837537         0.0377897         0.6763997
     credit_hisA32     credit_hisA33     credit_hisA34       purposeA41       purposeA410       purposeA42
        -0.1580219        -0.5131069        -1.3409246        -1.5268123        -1.2401881        -0.8040594
       purposeA43        purposeA44        purposeA45        purposeA46        purposeA48       purposeA49
        -0.9067831        -1.2662810        -0.4805119        -0.1747028        -2.0546883        -0.6276495
           amount    saving_acctA62    saving_acctA63    saving_acctA64    saving_acctA65    present_empA72
         0.0001127        -0.3044531        -0.3298246        -1.7124298        -1.1554768         0.3848369
    present_empA73    present_empA74    present_empA75  installment_rate             sexA92            sexA93
         0.1673888        -0.5956828         0.0644450         0.3402430        -0.1671574        -0.7444538
           sexA94  other_debtorA102  other_debtorA103      present_resid      propertyA122      propertyA123
        -0.2263970         0.7170219        -1.0559456        -0.0087191         0.1526831         0.1012915
     propertyA124               age other_installA142 other_installA143       housingA152       housingA153
         0.7081333        -0.0049953        -0.2228220        -0.5498318        -0.3967049        -0.9482995
         n_credits           jobA172           jobA173           jobA174          n_people    telephoneA192
         0.3754104         0.7166112         0.7267168         0.6088312         0.1500298        -0.2217752
      foreignA202
        -1.1055243
```

When we rerun the logistic regression model using all independent variables in the new 80% test set, the coefficients of the variables are as shown above. We will run forward selection again to find a better fitting model.

Figure 23: Histogram In Sample Prediction

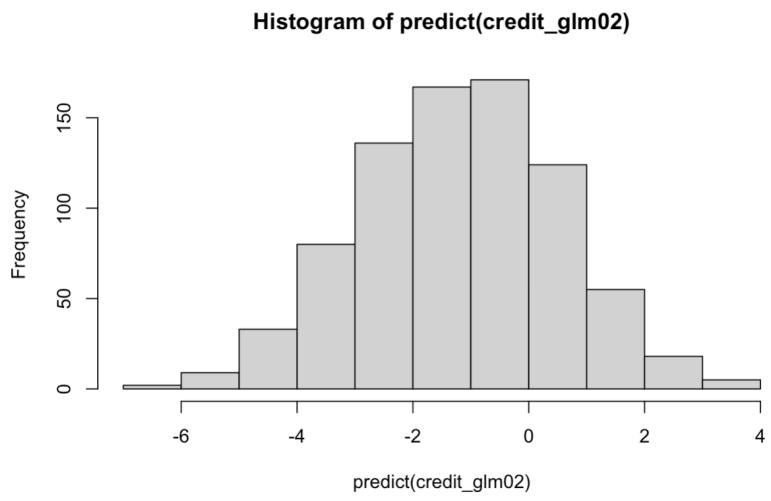
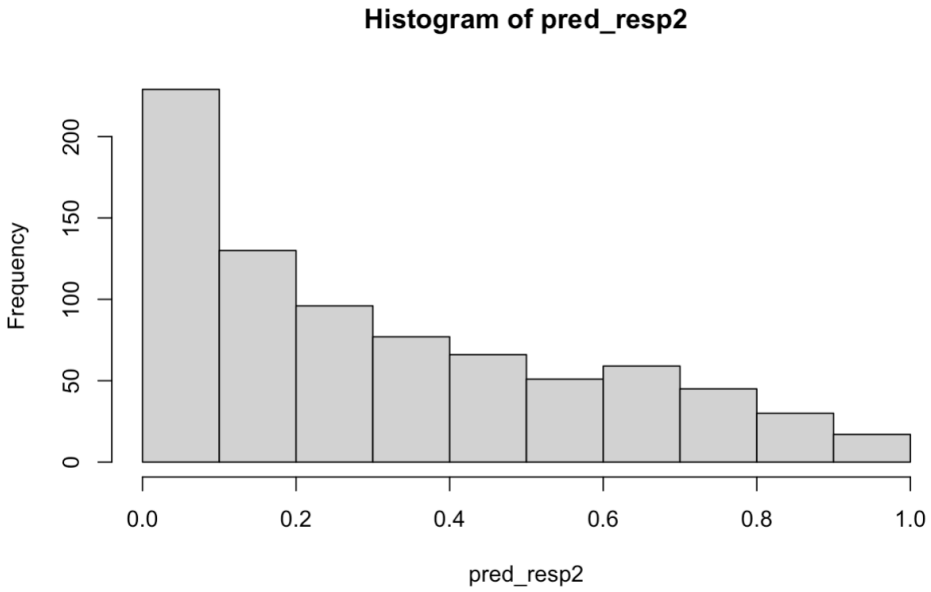


Figure 24: Histogram of In Sample Predicted Probability



Our second run through once again shows very similar results to our first run. Once again our fist histogram shown above is symmetrical with its highest peak right in the middle which reaches up to 150. The second histogram shows a right skew with the maximum on the left side of the graph while the values decline down towards the right.

Figure 25: Optimal Model Using Forward Selection

```
Step:  AIC=-1428.9
response ~ chk_acct + duration + credit_his + other_debtor +
    saving_acct + present_emp + purpose + installment_rate +
    amount + sex
```

```
Coefficients:
    (Intercept)      chk_acctA12      chk_acctA13      chk_acctA14         duration     credit_hisA31
     -6.632e-01       -3.573e-01       -1.318e+00       -1.712e+00        3.987e-02        5.698e-01
  credit_hisA32    credit_hisA33    credit_hisA34  other_debtorA102  other_debtorA103    saving_acctA62
     -4.703e-01       -5.996e-01       -1.374e+00        8.332e-01       -1.042e+00       -2.334e-01
 saving_acctA63   saving_acctA64   saving_acctA65   present_empA72    present_empA73   present_empA74
     -3.731e-01       -1.525e+00       -1.153e+00        6.290e-01        3.875e-01       -3.273e-01
 present_empA75       purposeA41      purposeA410       purposeA42       purposeA43       purposeA44
      2.952e-01       -1.454e+00       -1.358e+00       -7.608e-01       -8.751e-01       -1.346e+00
     purposeA45       purposeA46       purposeA48       purposeA49  installment_rate           amount
     -4.547e-01       -5.778e-02       -1.799e+00       -5.574e-01        3.145e-01        9.937e-05
         sexA92           sexA93           sexA94
     -8.631e-03       -5.922e-01       -1.856e-01
```

We used forward selection again to fit a better model as shown above. The model used 10 independent variables and excluded n_credits, foreign, other install, age, telephone, housing,

n_people, present_resid, property, and job. This was different from the optimal model of the original training set. Present_emp, amount, and sex were included in this optimal model while they were excluded in the original one. Housing, telephone, foreign, and age were excluded here whereas they were included in the original model. This is likely due to the fact that the training sample of both were different and had differences.

Figure 26: AIC of Optimal Model

```
> AIC(credit_glm02)
[1] 814.1679
```

```
> AIC(model_opt2)
[1] 799.7981
```

To compare the new optimal model to the new set of 80% training data, we use the AIC as our criteria again. The new optimal model proves to be better than the other because of the lower AIC.

Figure 27: ROC Curve of In Sample

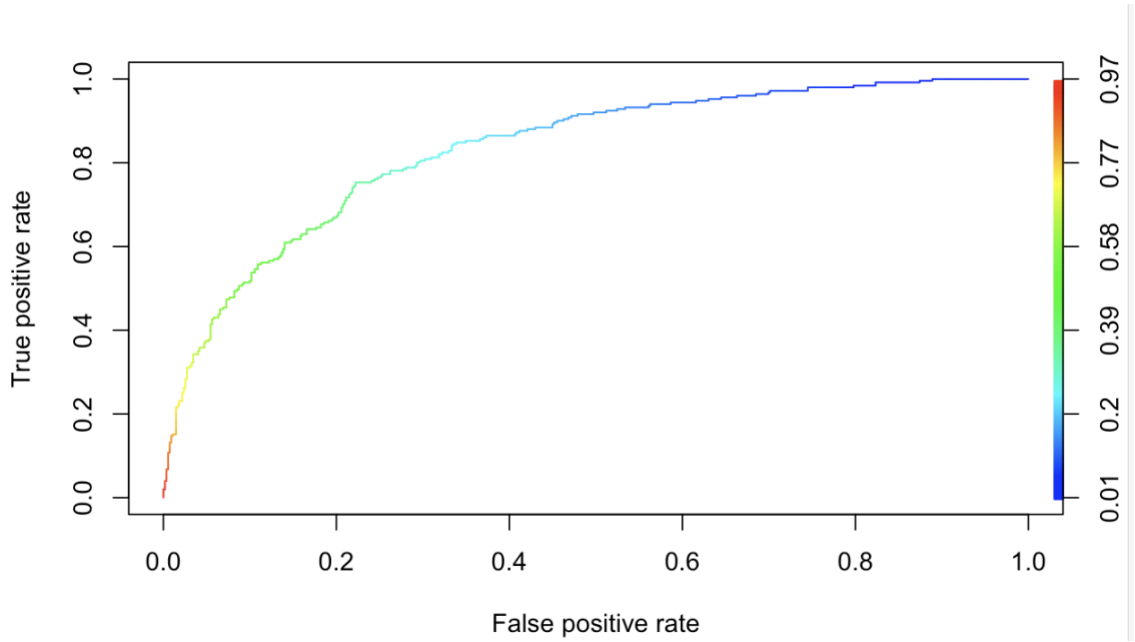


Figure 28: AUC of In Sample

```
> unlist(slot(performance(pred2, "auc"), "y.values"))
[1] 0.8314211
```

Comparing this value to the AUC for both In sample(0.828), and Out of sample(0.789), it is clear that this model is better at distinguishing between positive and negative cases because it is higher.

Figure 29: Misclassification Table

Truth	Predicted 0	1
0	489	60
1	113	138

Using the same equation as last time, MR = (60 + 113) / (489 + 60 + 113 + 138) = 0.216, we can see that it is slightly lower than the value we got from the optimal model from earlier(0.22). This model has better performance with less errors.

Figure 30: ROC Curve of Out of Sample

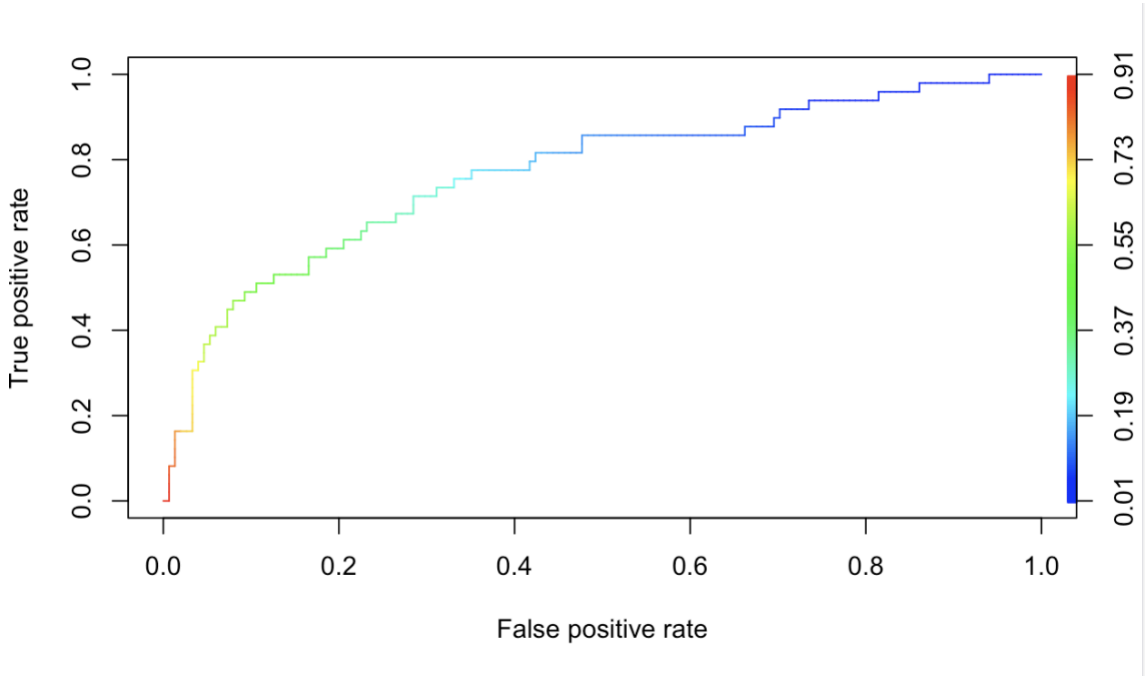


Figure 31: Misclassification Table of Out of Sample

Truth	Predicted 0	1
0	135	16
1	25	24

This table presents us with MR = (16 + 25) / (135 + 16 + 25 + 24) = 0.205, which is the lowest rate we have gotten, meaning that this creates the least amount of errors.

Figure 32: AUC of Out of Sample

```
> unlist(slot(performance(pred2, "auc"), "y.values"))
[1] 0.7690228
```

The area under the curve lets us evaluate the model's performance in context of the ROC curve, and this has been the lowest indicating that it does not perform as well as the other models that we have tested.

Figure 33: 5 Fold Cross Validation

```
> cv_result2$delta[2]
[1] 0.8736
```

This value is slightly higher than the the 5 cross validation we got from the first training set letting us know that the doesn't fit the data set as well.

Looking at both of the training sets and comparing the logistic regression, AIC/BIC, AUC, out-of-sample, and 5-fold cross validation we noticed that we get very similar results. The boxplots and histograms were almost identical. Furthermore, there were only a small difference in the values we got when testing the model's performance on the dataset. The main difference we saw was during the forward selection, when we were presented with a different number of variables.

```
#############Start of the Code ###############

german_credit =
read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data
")

colnames(german_credit) = c("chk_acct", "duration", "credit_his", "purpose", "amount",
"saving_acct", "present_emp", "installment_rate", "sex", "other_debtor", "present_resid",
"property", "age", "other_install", "housing", "n_credits", "job", "n_people", "telephone",
"foreign", "response")

# orginal response coding 1= good, 2 = bad we need 0 = good, 1 = bad

german_credit$response = german_credit$response - 1

#############End of the Code ###############


#for 80% training set
index <- sample(nrow(german_credit),nrow(german_credit)*0.80)
credit_train = german_credit[index,]
credit_test = german_credit[-index,]

credit_train
summary(credit_train) #for EDA

library(ggplot2) #for boxplots
boxplot(credit_train[,2])
boxplot(credit_train[,5])
boxplot(credit_train[,8])
boxplot(credit_train[,11])
boxplot(credit_train[,13])
boxplot(credit_train[,16])
boxplot(credit_train[,18])

credit_glm0 <- glm(response~., family=binomial, data=credit_train)
credit_glm0 #for the logistic regression model
summary(credit_glm0)

AIC(credit_glm0) #to evaluate model fitting
```

```
hist(predict(credit_glm0)) #histogram
pred_resp <- predict(credit_glm0,type="response")
hist(pred_resp)

#to find best model
nullmodel = lm(response ~ 1, data = credit_train)
fullmodel = lm(response ~ ., data = credit_train)
model.step <- step(nullmodel, scope = list(lower = nullmodel,
                                  upper = fullmodel),
                   direction = "forward")

#best model
model_opt <- glm(formula = response ~ chk_acct + duration + credit_his + purpose +
other_debtor +
                   saving_acct + installment_rate + housing + amount + foreign +
                   telephone + age, family = binomial, data = credit_train)
summary(model_opt)
AIC(model_opt) #aic for optimal model

install.packages('ROCR')
library(ROCR) #In sample prediction
pred_glm0_train <- predict(model_opt, type = "response")
pred <- prediction(pred_glm0_train, credit_train$response)
perf <- performance(pred, "tpr", "fpr")
plot(perf, colorize=TRUE) #ROC curve for training set

#Get the AUC
unlist(slot(performance(pred, "auc"), "y.values"))

#for the misclassification table
table(credit_train$response, (pred_glm0_train > 0.5)*1, dnn=c("Truth","Predicted"))

costfunc <- function(obs, pred.p){
  weight1 <- 5 # define the weight for "true=1 but pred=0" (FN)
  weight0 <- 1 # define the weight for "true=0 but pred=1" (FP)
  pcut <- 0.5
  c1 <- (obs==1)&(pred.p < pcut) # count for "true=1 but pred=0" (FN)
  c0 <- (obs==0)&(pred.p >= pcut) # count for "true=0 but pred=1" (FP)
  cost <- mean(weight1*c1 + weight0*c0) # misclassification with weight
```

```
  return(cost) # you have to return to a value when you write R functions
}

#out of sample prediction
pred_glm0_test<- predict(credit_glm0, newdata = credit_test, type="response")
pred <- prediction(pred_glm0_test, credit_test$response)
perf <- performance(pred, "tpr", "fpr")
plot(perf, colorize=TRUE)

#Get the AUC for out of sample
unlist(slot(performance(pred, "auc"), "y.values"))

#for 5 fold cross validation
library(boot)
credit_glm1<- glm(response~. , family=binomial, data=german_credit);
cv_result <- cv.glm(data=german_credit, glmfit=credit_glm1, cost=costfunc, K=
                    5)
cv_result$delta[2]
1



### for repeating above with new 80% training data
#for 80% training set
index2 <- sample(nrow(german_credit),nrow(german_credit)*0.80)
credit_train2 = german_credit[index2,]
credit_test2 = german_credit[-index2,]

credit_train2
summary(credit_train2) #for EDA

library(ggplot2) #for boxplots
boxplot(credit_train2[,2])
boxplot(credit_train2[,5])
boxplot(credit_train2[,8])
boxplot(credit_train2[,11])
boxplot(credit_train2[,13])
boxplot(credit_train2[,16])
boxplot(credit_train2[,18])

credit_glm02 <- glm(response~., family=binomial, data=credit_train2)
```

```
credit_glm02 #for the logistic regression model
summary(credit_glm02)

AIC(credit_glm02) #to evaluate model fitting

hist(predict(credit_glm02)) #histogram
pred_resp2 <- predict(credit_glm02,type="response")
hist(pred_resp2)

#to find best model
nullmodel2 = lm(response ~ 1, data = credit_train2)
fullmodel2 = lm(response ~ ., data = credit_train2)
model.step <- step(nullmodel2, scope = list(lower = nullmodel2,
                                  upper = fullmodel2),
                   direction = "forward")

#best model
model_opt2 <- glm(formula = response ~ chk_acct + duration + credit_his + purpose +
present_emp +
                    foreign + saving_acct + other_debtor + age + housing + other_install +
                    amount + installment_rate, family = binomial, data = credit_train2)
summary(model_opt2)
AIC(model_opt2) #aic for optimal model

install.packages('ROCR')
library(ROCR) #In sample prediction
pred_glm0_train2 <- predict(model_opt2, type = "response")
pred2 <- prediction(pred_glm0_train2, credit_train2$response)
perf2 <- performance(pred2, "tpr", "fpr")
plot(perf2, colorize=TRUE) #ROC curve for training set

#Get the AUC
unlist(slot(performance(pred2, "auc"), "y.values"))

#for the misclassification table
table(credit_train2$response, (pred_glm0_train2 > 0.5)*1, dnn=c("Truth","Predicted"))

costfunc2 <- function(obs, pred.p){
  weight1 <- 5 # define the weight for "true=1 but pred=0" (FN)
  weight0 <- 1 # define the weight for "true=0 but pred=1" (FP)
```

```
  pcut <- 0.5
  c12 <- (obs==1)&(pred.p < pcut) # count for "true=1 but pred=0" (FN)
  c02 <- (obs==0)&(pred.p >= pcut) # count for "true=0 but pred=1" (FP)
  cost2 <- mean(weight1*c12 + weight0*c02) # misclassification with weight
  return(cost2) # you have to return to a value when you write R functions
}

#out of sample prediction
pred_glm0_test2<- predict(credit_glm02, newdata = credit_test2, type="response")
pred2 <- prediction(pred_glm0_test2, credit_test2$response)
perf2 <- performance(pred2, "tpr", "fpr")
plot(perf2, colorize=TRUE)

#Get the AUC for out of sample
unlist(slot(performance(pred2, "auc"), "y.values"))

#for 5 fold cross validation
library(boot)
credit_glm12<- glm(response~. , family=binomial, data=german_credit);
cv_result2 <- cv.glm(data=german_credit, glmfit=credit_glm12, cost=costfunc2, K=
                     5)
cv_result2$delta[2]
```

BUS193 Data Mining Case 4

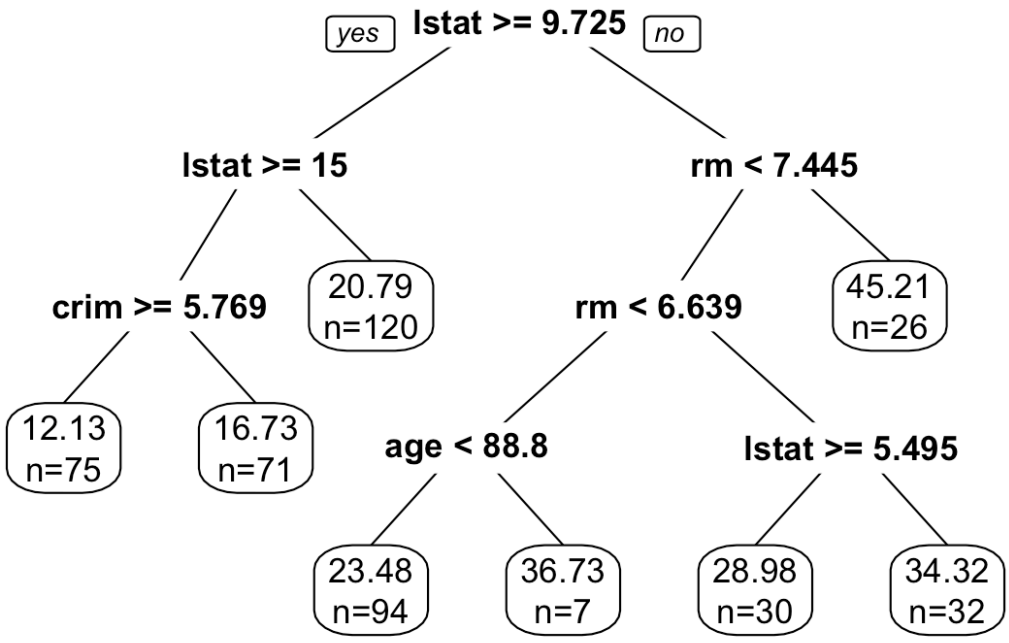
Section 1

Group 8: Caitlyn Blair, Shruti Hardasani, Rainna Sena

5 November 2024

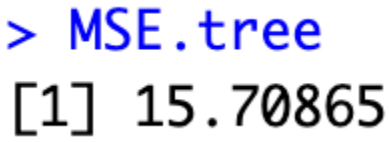
Figure 1: Regression Tree on Training Data

```
n= 455

node), split, n, deviance, yval
      * denotes terminal node

 1) root 455 38564.5500 22.41582
   2) lstat>=9.725 266  6486.3510 17.26429
     4) lstat>=15 146  2606.5860 14.36644
       8) crim>=5.76921 75  1040.8830 12.13200 *
       9) crim< 5.76921 71   795.6992 16.72676 *
     5) lstat< 15 120  1162.0480 20.79000 *
   3) lstat< 9.725 189 15083.8200 29.66614
     6) rm< 7.445 163  6836.9830 27.18712
      12) rm< 6.6385 101  3004.9980 24.39604
        24) age< 88.8 94   885.3631 23.47766 *
        25) age>=88.8 7   975.7143 36.72857 *
      13) rm>=6.6385 62  1763.4590 31.73387
        26) lstat>=5.495 30   488.6680 28.98000 *
        27) lstat< 5.495 32   833.9822 34.31562 *
     7) rm>=7.445 26   965.0785 45.20769 *
```



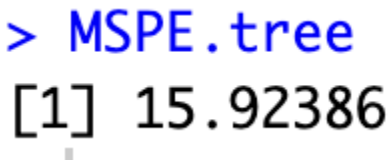
The top output is the printed version of the regression tree, which is difficult to interpret. This is why we showed the model visually below. For the regression tree on the Boston housing dataset, we took a random sample of 90% for our training set and used the other 10% as the test set. The output for the regression tree is as shown above. Going to the left side means the criteria is true while the right side means no.

Figure 2: In-Sample MSE Performance



The MSE helps us to get an idea about how good the model performance is. To get this, we use the predict function to get the in-sample predicted values. The in-sample MSE is 15.71, which is a relatively low value so this is good.

Figure 2: Out-of-Sample MSE Performance



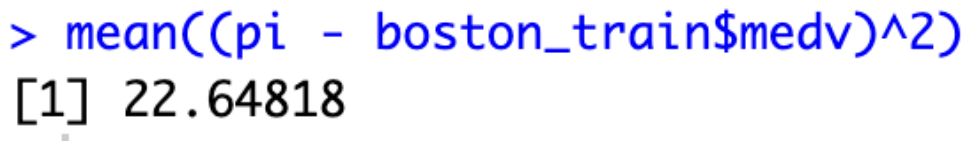
Using the predict function once again to get our predicted values for the out-of-sample values, the MSE is 15.92 which is very close to the in-sample MSE.

Figure 3: Linear Regression Model on Boston Training Set

```
Coefficients:
(Intercept)        crim          zn        chas         nox          rm
  39.559371   -0.114504    0.044809    3.132874  -19.657220    3.524630
        dis         rad         tax     ptratio       black       lstat
  -1.539422    0.287534   -0.009829   -0.963041    0.008518   -0.539330
```

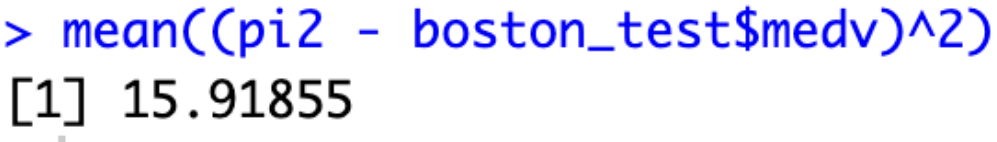
We fit a linear regression model on the same Boston training set using every variable except for indus and age, and above are the coefficients for that equation.

Figure 4: Linear Regression In-Sample MSE Performance



Using the predict function again, we found the MSE of the predicted values for the in-sample values. The result was 22.65.

Figure 5: Linear Regression Out-of-Sample MSE Performance



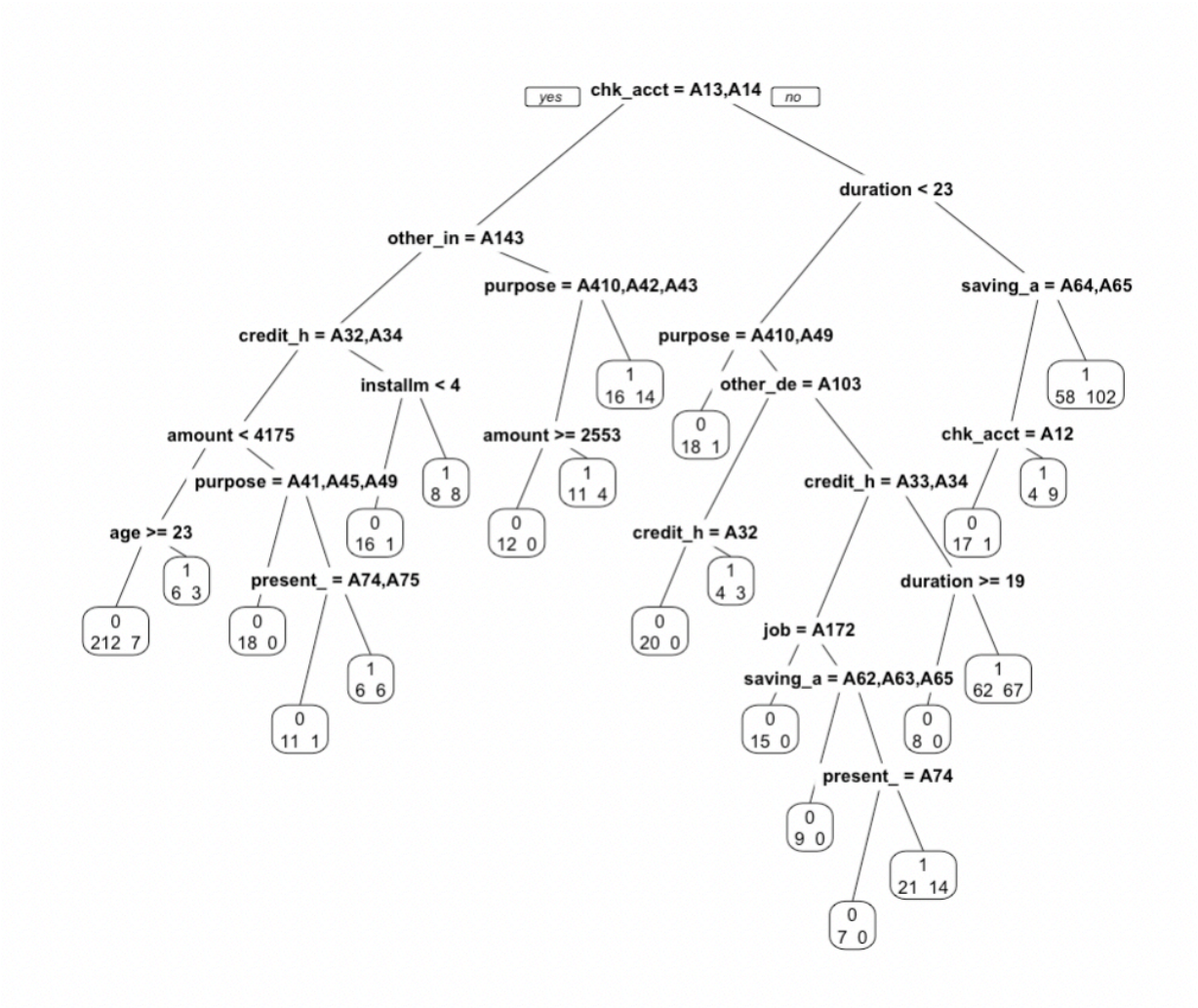
We tested the out-of-sample predicted values against the actual values for the out-of-sample MSE and the result was 15.92.

When we compare the MSE values from the CART regression tree model versus the linear regression model, we can see that the CART model is better in predicting the median housing prices. This is because the MSE values from the CART model (15.71 in-sample and 15.92 out-of-sample) were lower than the MSE values from the linear regression model (22.65 in-sample and 15.92 out-of-sample). Although the out-of-sample MSE from the linear regression model was very slightly smaller, the overall values from the CART model was overall smaller which leads us to conclude that the CART model is better.

German Credit Score Data

Figure 1: Classification Tree on Training Data

```
n= 800

node), split, n, loss, yval, (yprob)
      * denotes terminal node

   1) root 800 241 0 (0.69875000 0.30125000)
     2) chk_acct=A13,A14 360  44 0 (0.87777778 0.12222222) *
     3) chk_acct=A11,A12 440 197 0 (0.55227273 0.44772727)
       6) duration< 22.5 249  85 0 (0.65863454 0.34136546)
        12) credit_his=A32,A33,A34 228  69 0 (0.69736842 0.30263158)
          24) purpose=A41,A410,A43,A45,A49 96  19 0 (0.80208333 0.19791667) *
          25) purpose=A40,A42,A44,A46,A48 132  50 0 (0.62121212 0.37878788)
            50) other_install=A142,A143 118  40 0 (0.66101695 0.33898305)
             100) credit_his=A34 40   7 0 (0.82500000 0.17500000) *
             101) credit_his=A32,A33 78  33 0 (0.57692308 0.42307692)
               202) amount>=1485.5 40  10 0 (0.75000000 0.25000000) *
               203) amount< 1485.5 38  15 1 (0.39473684 0.60526316)
                 406) sex=A93,A94 17   6 0 (0.64705882 0.35294118) *
                 407) sex=A91,A92 21   4 1 (0.19047619 0.80952381) *
            51) other_install=A141 14   4 1 (0.28571429 0.71428571) *
        13) credit_his=A30,A31 21   5 1 (0.23809524 0.76190476) *
       7) duration>=22.5 191  79 1 (0.41361257 0.58638743)
        14) saving_acct=A64,A65 31  10 0 (0.67741935 0.32258065)
          28) chk_acct=A12 18   1 0 (0.94444444 0.05555556) *
          29) chk_acct=A11 13   4 1 (0.30769231 0.69230769) *
        15) saving_acct=A61,A62,A63 160  58 1 (0.36250000 0.63750000)
          30) duration< 47.5 131  55 1 (0.41984733 0.58015267)
            60) amount>=1549.5 120  55 1 (0.45833333 0.54166667)
             120) purpose=A41 18   5 0 (0.72222222 0.27777778) *
             121) purpose=A40,A410,A42,A43,A45,A46,A49 102  42 1 (0.41176471 0.58823529)
               242) other_debtor=A103 8   2 0 (0.75000000 0.25000000) *
               243) other_debtor=A101,A102 94  36 1 (0.38297872 0.61702128)
                 486) amount< 4231 64  29 1 (0.45312500 0.54687500)
                   972) amount>=2313 43  20 0 (0.53488372 0.46511628)
                    1944) sex=A93,A94 28  10 0 (0.64285714 0.35714286) *
                    1945) sex=A91,A92 15   5 1 (0.33333333 0.66666667) *
                   973) amount< 2313 21   6 1 (0.28571429 0.71428571) *
                 487) amount>=4231 30   7 1 (0.23333333 0.76666667) *
            61) amount< 1549.5 11   0 1 (0.00000000 1.00000000) *
          31) duration>=47.5 29   3 1 (0.10344828 0.89655172) *
```



This is our classification tree. The second image is easier to read because it visually represents the outputs. For the classification tree we used the German data set. We did a random sample of 80% for our training set and used the other 20% as the test set. To the left shows if the criteria is true while going to the right shows if the criteria is not true.

Figure 2: In sample Misclassification Table

Truth	Predicted 0	1
0	363	196
1	11	230

Our misclassification rate is computed by (196+11) / (363+196+11+230) =0.259. This means that about 25.9% of our predictions are incorrect.

Figure 3: Out of Sample Misclassification Table

Truth	Predicted 0	1
0	73	68
1	14	45

Our misclassification rate is computed by (68+14) / (73+68+14+45)=0.41. This means that about 41% of our predictions are incorrect.

Figure 4: In Sample Misclassification Cost (Asymmetric)

```
[1] 0.50625
```

The in-sample asymmetric misclassification cost tells us the cost of classification errors within the dataset used to train our model. There is a occurrence of 0.50624.

Figure 5: Out of Sample Misclassification Cost (Asymmetric)

```
[1] 0.6975
```

A lower value indicates a better classification model, so we can say there are more errors in the out-of-sample, also referred to as the testing data set. There are added weights of 5, so a false negative costs 1 while a false positive costs 5.

Figure 6: Out of sample ROC curve

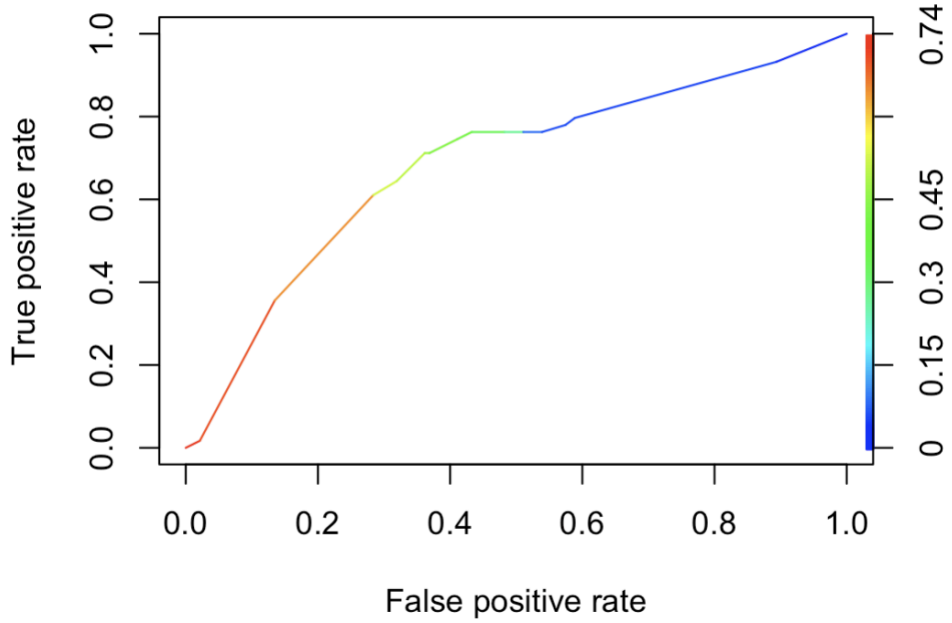
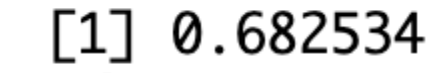


Figure 7: Out of Sample AUC



The area under the curve lets us evaluate the model's performance in context of the ROC curve, and the AUC is 0.683 which isn't over 0.7, so we can't say that this is a good model, but it is close.

CODE:

```
library(MASS)
data(Boston)

#for the training and test sets
subset <- sample(nrow(Boston), nrow(Boston) * 0.9)
boston_train = Boston[subset, ]
boston_test = Boston[-subset, ]

install.packages('rpart')
install.packages("rpart.plot")
library(rpart)
library(rpart.plot)

#for decision tree on training set
boston_rpart <- rpart(formula = medv ~ ., data = boston_train)
boston_rpart
prp(boston_rpart, digits = 4, extra = 1) #to visualize the tree

#prediction using regression trees
boston_train_pred_tree <- predict(boston_rpart, data = boston_train) #in sample prediction
values
boston_test_pred_tree <- predict(boston_rpart, boston_test) #out of sample prediction values

#for MSE
MSE.tree <- mean((boston_train_pred_tree - boston_train$medv)^2) #in sample prediction error
MSE.tree
MSPE.tree <- mean((boston_test_pred_tree - boston_test$medv)^2) #out of sample prediction
error
MSPE.tree

model_1 <- lm(medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +
              black + lstat, data = boston_train)


#predicted values for MSE (in sample)
pi <- predict(object = model_1, newdata = boston_train)
pi
```

```
#MSE (in sample)
mean((pi - boston_train$medv)^2)

#predicted for MSE (out of sample)
pi2 <- predict(object = model_1, newdata = boston_test)
pi2

#MSE (out of sample)
mean((pi2 - boston_test$medv)^2)
```

```
#for the german part
german_credit =
read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data
")

colnames(german_credit) = c("chk_acct", "duration", "credit_his", "purpose", "amount",
"saving_acct", "present_emp", "installment_rate", "sex", "other_debtor", "present_resid",
"property", "age", "other_install", "housing", "n_credits", "job", "n_people", "telephone",
"foreign", "response")

# original response coding 1= good, 2 = bad we need 0 = good, 1 = bad

german_credit$response = german_credit$response - 1

#for 80% training set
index <- sample(nrow(german_credit),nrow(german_credit)*0.80)
credit_train = german_credit[index,]
credit_test = german_credit[-index,]

credit_rpart <- rpart(formula = response ~ .,
                      data = credit_train,
                      method = "class",
                      parms = list(loss=matrix(c(0,5,1,0), nrow = 2)))
```

```
#using all variables (training)
credit_rpart0 <- rpart(formula = response ~ ., data = credit_train, method = "class")
credit_rpart0

#for the classification trees
prp(credit_rpart, extra = 1) #with the costs
prp(credit_rpart0, extra = 1) #without the costs

#for misclassification table on ALL data
pred0 <- predict(credit_rpart0, type="class")
table(credit_train$response, pred0, dnn = c("True", "Pred"))

#for in sample prediction and misclass table (need to report after)
credit_train.pred.tree1<- predict(credit_rpart, credit_train, type="class")
table(credit_train$response, credit_train.pred.tree1, dnn=c("Truth","Predicted"))
#i think here we do the (B+C) / (A+B+C+D)

#for out of sample prediction and misclass table (need to report after)
credit_test.pred.tree1<- predict(credit_rpart, credit_test, type="class")
table(credit_test$response, credit_test.pred.tree1, dnn=c("Truth","Predicted"))

#cost function for expected loss with weights, can change pcut values by changing
#weight1 and weight0
cost <- function(r, phat){
  weight1 <- 5
  weight0 <- 1
  pcut <- weight0/(weight1+weight0)
  c1 <- (r==1)&(phat<pcut) #logical vector - true if actual 1 but predict 0
  c0 <-(r==0)&(phat>pcut) #logical vector - true if actual 0 but predict 1
  return(mean(weight1*c1+weight0*c0))
}
#type = prob to get the probability
#in sample misclassification cost
cost(credit_train$response, predict(credit_rpart, credit_train, type="prob"))



#For ROC curve out of sample
#predicted prob on test set
credit_test_prob_rpart = predict(credit_rpart, credit_test, type="prob")
install.packages('ROCR')
```

```
library(ROCR)
#out of sample ROC curve
pred = prediction(credit_test_prob_rpart[,2], credit_test$response)
perf = performance(pred, "tpr", "fpr")
plot(perf, colorize=TRUE)
#for AUC
slot(performance(pred,"auc"), "y.values")[[1]]

#idk if we need this part
#the credit_test_prob_rpart saved the predicted value for test set
#the 1/5+1 sets the cutoff value
credit_test_pred_rpart = as.numeric(credit_test_prob_rpart[,2] > 1/(5+1))
table(credit_test$response, credit_test_pred_rpart, dnn=c("Truth","Predicted"))
```

BUS193 Data Mining Case 5

Section 1

Group 8: Caitlyn Blair, Shruti Hardasani, Rainna Sena

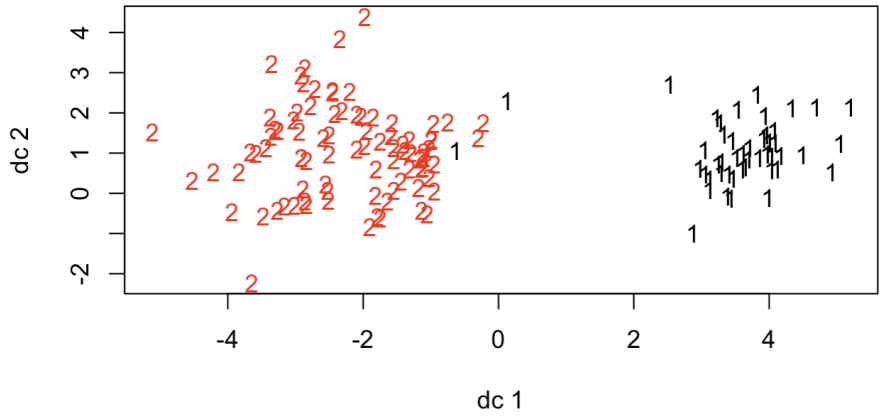
19 November 2024

Figure 1: K Means Clustering (2 Clusters)

1	2
47	88

Here we have 2 clusters: the first one has 47 points while the other has 88 points.

Figure 2: Plotted Clusters



Here is our cluster diagram illustrating 2 clusters. There is a pretty distinct difference between the two clusters apart from the few points from group 1 that seem close to group 2.

Figure 3: Items in Each Cluster

```
> iris_train$Species[fit$cluster == 1]
 [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [8] setosa     setosa     setosa     setosa     setosa     setosa     setosa
[15] setosa     setosa     setosa     versicolor setosa     setosa     setosa
[22] setosa     setosa     setosa     setosa     setosa     versicolor setosa
[29] setosa     setosa     setosa     setosa     setosa     setosa     setosa
[36] setosa     setosa     setosa     setosa     setosa     setosa     setosa
[43] setosa     setosa     setosa     setosa     setosa
Levels: setosa versicolor virginica
> iris_train$Species[fit$cluster == 2] #2nd group
 [1] versicolor virginica  versicolor virginica  versicolor virginica  virginica
 [8] versicolor versicolor virginica  versicolor virginica  versicolor virginica
[15] virginica  versicolor versicolor virginica  virginica  virginica  versicolor
[22] versicolor virginica  virginica  virginica  versicolor virginica  virginica
[29] virginica  virginica  versicolor virginica  virginica  virginica  versicolor
[36] versicolor versicolor virginica  virginica  virginica  versicolor virginica
[43] versicolor versicolor virginica  virginica  versicolor virginica  versicolor
[50] versicolor versicolor virginica  versicolor versicolor versicolor versicolor
[57] versicolor versicolor versicolor versicolor virginica  versicolor virginica
[64] versicolor versicolor versicolor virginica  virginica  virginica  versicolor
[71] virginica  versicolor virginica  versicolor virginica  virginica  versicolor
[78] versicolor virginica  virginica  virginica  virginica  virginica  virginica
[85] versicolor virginica  virginica  virginica
```

This illustrates what is exactly in the two different groups. This was done on our 90% random sample data set. Group one consists mainly of setosa and a few versicolor, while group two consists of all versicolor and virginia.

Figure 4: Cluster Means

	Group.1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	1	5.012766	3.372340	1.536170	0.2893617
2	2	6.300000	2.886364	4.972727	1.7068182

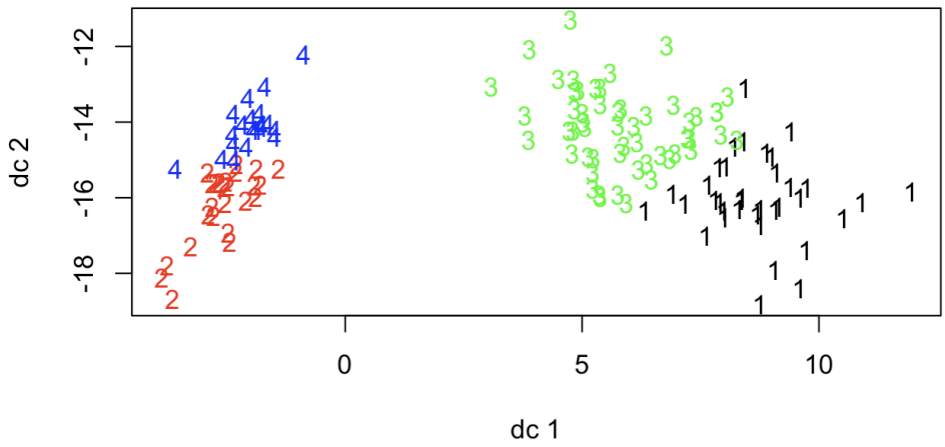
The means for the two groups are relatively close. What's interesting is most of the means for group 1 are lower than the means in group 2. They are all similar except for the petal length.

Figure 5: K Means Clustering (4 Clusters)

1	2	3	4
36	25	54	20

We decided to do 4 clusters and the code set each group with a different amount of points: group 1 has 36 points, group 2 has 25 points, group 3 has 54 points, and group 4 has 20 points.

Figure 6: Plotted Clusters



Once again here is a diagram showing what the points clustered into 4 groups like on a graph. Groups one and three are close in proximity to each other while groups 2 and 4 are close to one another.

Figure 7: Items in Each Cluster

```
> iris_train$Species[fit$cluster == 1]
 [1] virginica  virginica  virginica  virginica  versicolor virginica  virginica
 [8] virginica  virginica  virginica  virginica  virginica  virginica  virginica
[15] virginica  virginica  virginica  virginica  virginica  virginica  virginica
[22] virginica  virginica  virginica  virginica  virginica  virginica  virginica
[29] virginica  virginica  versicolor virginica  virginica  virginica  virginica
[36] virginica
Levels: setosa versicolor virginica
> iris_train$Species[fit$cluster == 2] #2nd group
 [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[12] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[23] setosa setosa setosa
Levels: setosa versicolor virginica
> iris_train$Species[fit$cluster == 3]
 [1] versicolor versicolor versicolor virginica  versicolor versicolor versicolor
 [8] versicolor versicolor versicolor versicolor versicolor virginica  versicolor
[15] virginica  virginica  versicolor virginica  versicolor versicolor versicolor
[22] versicolor virginica  versicolor versicolor versicolor versicolor virginica
[29] versicolor versicolor versicolor virginica  versicolor versicolor versicolor
[36] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[43] versicolor versicolor versicolor versicolor versicolor virginica  versicolor
[50] virginica  virginica  versicolor virginica  virginica
Levels: setosa versicolor virginica
> iris_train$Species[fit$cluster == 4]
 [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[12] setosa setosa setosa setosa setosa setosa setosa setosa setosa
```

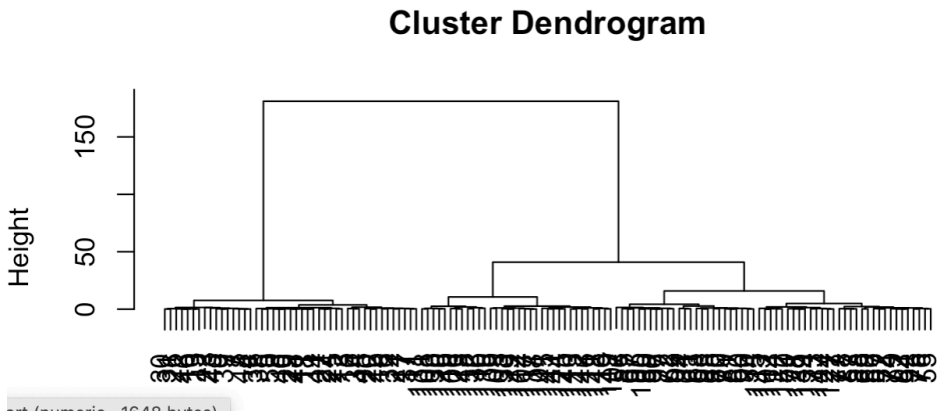
Here we laid out what exactly was put into all the 4 different clusters. Group 1 has mostly virginica and a few versicolor, group 2 has all setosa, group 3 has a mix of versicolor and virginica, and group 4 has all setosa.

Figure 8: Cluster Means

	Group.1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	1	6.863889	3.061111	5.747222	2.055556
2	2	5.248000	3.644000	1.496000	0.296000
3	3	5.877778	2.751852	4.388889	1.450000
4	4	4.715000	3.130000	1.425000	0.205000

The means for the 4 different clusters are all similar to one another. Unlike our 2 clusters above here there isn't a distinct group who has the highest or lowest means. But group 1 seems to have the highest mean of 6.86 for sepal length while group 4 has one of the lowest means of 0.21 for petal width. There is the highest variation in petal length while the rest are relatively similar.

Figure 9: Hierarchical Clustering



We can see the hierarchical relationship for the whole set, there are many instances where there is overlap. There seems to be one that stands out the most and surpasses a height of 150 and almost touches 200.

Figure 10: 2 Clusters

```
groupIris.2
 1  2
90 45
```

This shows that for the two groups, group 1 had 90 points while group 2 had only 45 points.

Figure 11: Items in Each Cluster

```
> iris_train$Species[groupIris.2 == 1]
 [1] versicolor virginica  versicolor virginica  versicolor virginica  virginica  versicolor
 [9] versicolor virginica  versicolor virginica  versicolor virginica  virginica  versicolor
[17] versicolor virginica  virginica  virginica  versicolor versicolor virginica  versicolor
[25] virginica  virginica  versicolor virginica  virginica  virginica  virginica  versicolor
[33] virginica  virginica  virginica  versicolor versicolor versicolor virginica  virginica
[41] virginica  versicolor virginica  versicolor versicolor versicolor virginica  virginica
[49] versicolor virginica  versicolor versicolor versicolor virginica  versicolor versicolor
[57] versicolor versicolor versicolor versicolor versicolor versicolor virginica  versicolor
[65] virginica  versicolor versicolor versicolor virginica  virginica  virginica  versicolor
[73] virginica  versicolor virginica  versicolor virginica  virginica  versicolor versicolor
[81] virginica  virginica  virginica  virginica  virginica  virginica  versicolor virginica
[89] virginica  virginica

> iris_train$Species[groupIris.2 == 2]
 [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[12] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[23] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[34] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[45] setosa
```

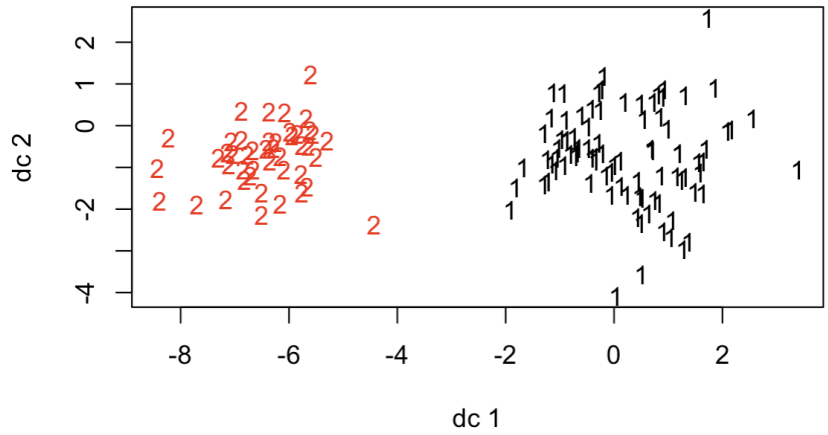
The above shows which points were put into each set. Group 1 consisted of all virginia and versicolor while group 2 had all setosa.

Figure 12: Cluster Means

	Group.1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	1	6.272222	2.875556	4.932222	1.6922222
2	2	5.011111	3.415556	1.464444	0.2555556

Group 1 had the highest mean of 6.27 for sepal length and group 2 had the lowest mean of 0.26 for petal width. Once again, the means are similar to each other except in petal length.

Figure 13: Plotted Clusters



Here is a diagram showing what the points clustered into 2 groups like on a graph. There is no overlap between the two.

Figure 14: 4 Clusters

```
groupIris.4
 1  2  3  4
31 34 25 45
```

This shows that for the 4 groups, group 1 had 31 points, group 2 had 34 points, group 3 had 25 points and group 4 had 45 points.

Figure 15: Items in Each Cluster

```
> iris_train$Species[groupIris.4 == 1]
 [1] versicolor virginica  versicolor versicolor versicolor versicolor virginica  virginica
 [9] virginica  versicolor virginica  versicolor versicolor virginica  versicolor virginica
[17] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[25] virginica  versicolor virginica  virginica  versicolor virginica  virginica
Levels: setosa versicolor virginica
> iris_train$Species[groupIris.4 == 2]
 [1] virginica virginica virginica virginica virginica virginica virginica virginica virginica
[10] virginica virginica virginica virginica virginica virginica virginica virginica virginica
[19] virginica virginica virginica virginica virginica virginica virginica virginica virginica
[28] virginica virginica virginica virginica virginica virginica virginica
Levels: setosa versicolor virginica
> iris_train$Species[groupIris.4 == 3]
 [1] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [9] versicolor versicolor virginica  versicolor versicolor versicolor versicolor versicolor
[17] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[25] versicolor
Levels: setosa versicolor virginica
> iris_train$Species[groupIris.4 == 4]
 [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[14] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[27] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[40] setosa setosa setosa setosa setosa setosa
```

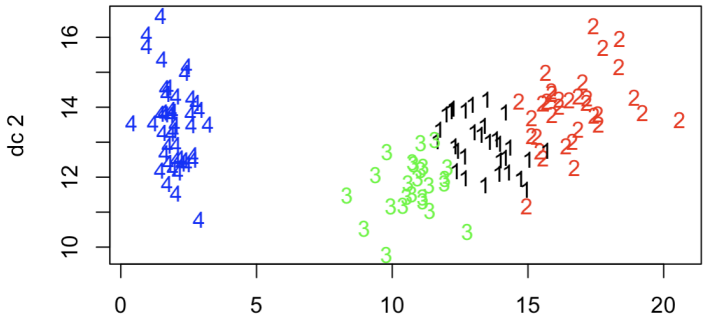
This shows which points were put into each set. Set 1 was a combination for both versicolor and virginica, while set 2 was just virginica, set 3 was versicolor and set 4 was setosa.

Figure 16: Cluster Means

	Group.1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	1	6.203226	2.858065	4.761290	1.6290323
2	2	6.867647	3.061765	5.794118	2.0823529
3	3	5.548000	2.644000	3.972000	1.2400000
4	4	5.011111	3.415556	1.464444	0.2555556

The means for the 4 different clusters are all similar to one another. Group 2 sees to have the highest mean of 6.87 for sepal length and group 4 has the lowest mean of 0.26 for petal width. The cluster means are relatively similar except in petal length, which was similar to the other cluster means.

Figure 17: Plot of all Clusters



Here is a diagram showing what the points clustered into 4 groups like on a graph. Groups four is separated from the rest of the groups. Group 1,2,3 are in close proximity of each other.

Figure 18: Groceries Dataset

```
> Groceries
transactions in sparse format with
 9835 transactions (rows) and
 169 items (columns)
```

The dimensions of the Grocery dataset is shown above with 9835 transactions with 169 items.

Figure 19: First 10 Transactions

```
      items
[1]  {citrus fruit, semi-finished bread, margarine, ready soups}
[2]  {tropical fruit, yogurt, coffee}
[3]  {whole milk}
[4]  {pip fruit, yogurt, cream cheese , meat spreads}
[5]  {other vegetables, whole milk, condensed milk, long life bakery product}
[6]  {whole milk, butter, yogurt, rice, abrasive cleaner}
[7]  {rolls/buns}
[8]  {other vegetables, UHT-milk, rolls/buns, bottled beer, liquor (appetizer)}
[9]  {pot plants}
[10] {whole milk, cereals}
```

The first 10 transactions are shown above.

Figure 20: Most Frequent Purchase

```
most frequent items:
      whole milk other vegetables
            2513             1903
```

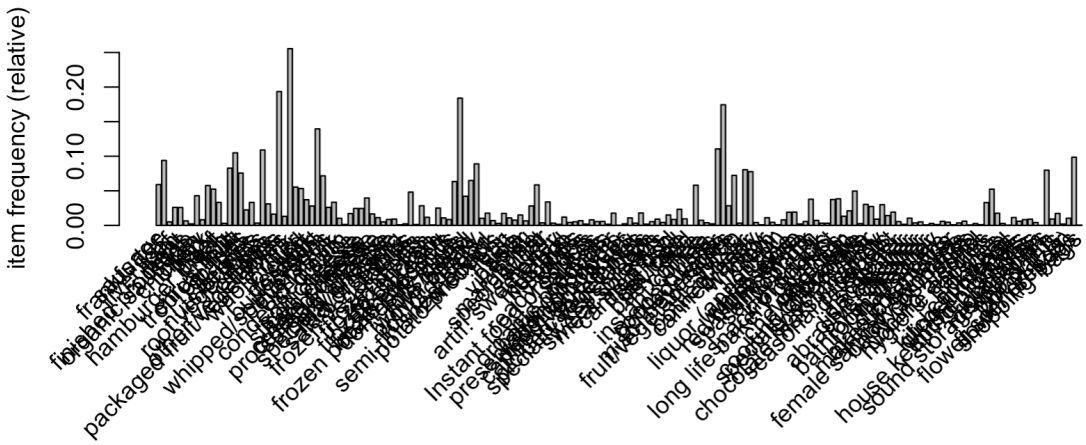
The most frequent transaction is the purchase of milk, which was bought 2513 times with other vegetables following at 1903 purchases.

Figure 21: Average & Largest Items in Transaction

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000
```

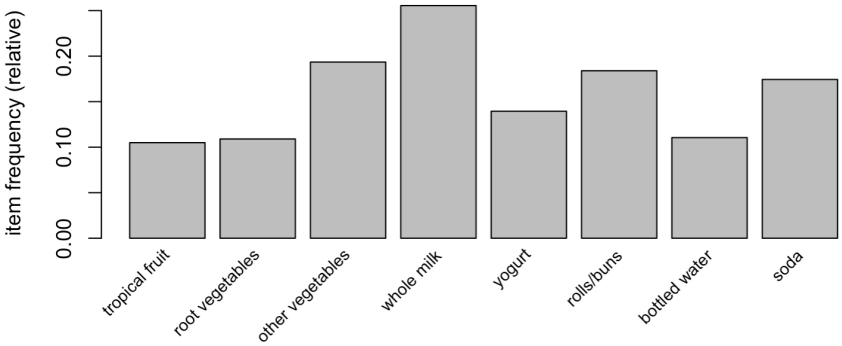
The average transaction had about 4.409 items, while the largest purchase in the dataset included 32 items.

Figure 22: Item Frequency Plot



The item frequency plot with all 169 items is shown above. It is difficult to read because it includes everything and it is so close together.

Figure 23: Item Frequency Plot (Support > 10%)



The item frequency plot above shows only the items that have a support of at least 10%. This includes tropical fruit, root vegetables, other vegetables, whole milk, yogurt, rolls/buns, bottled water, and soda. This plot is far easier to interpret because it only includes 8 items out of the 169.

Figure 24: Generating Association Rules Input

```
# support = 0.01 and confidence = 0.4
rules <- apriori(Groceries, parameter = list(support = 0.01,
                                             confidence = 0.4))
```

To generate our association rules with the Groceries dataset, we chose a support of 0.1 (10%) and a confidence of 0.4 (40%).

Figure 25: Number of Rules

```
> rules
set of 62 rules
```

Our criteria generated 62 rules.

Figure 26: Distribution of Number of Rules

```
rule length distribution
 2  3
25 37
```

The distribution of the number of items in each rule is as shown above. This means that there are 25 rules with items and 37 rules with 3 items included.

Figure 27: Average Lift

```
summary of quality measures:
    support         confidence        coverage            lift           count
 Min.   :0.01007   Min.   :0.4016   Min.   :0.01729   Min.   :1.572   Min.   : 99.0
 1st Qu.:0.01149   1st Qu.:0.4178   1st Qu.:0.02450   1st Qu.:1.729   1st Qu.:113.0
 Median :0.01388   Median :0.4502   Median :0.03005   Median :1.947   Median :136.5
 Mean   :0.01752   Mean   :0.4643   Mean   :0.03882   Mean   :1.993   Mean   :172.4
 3rd Qu.:0.01790   3rd Qu.:0.4975   3rd Qu.:0.04291   3rd Qu.:2.195   3rd Qu.:176.0
 Max.   :0.05602   Max.   :0.5862   Max.   :0.13950   Max.   :3.030   Max.   :551.0
```

The average lift of the rules we generated is 1.9993.

Figure 28: 5 Rules with Highest Lift

```
    lhs                                rhs                support    confidence coverage   lift     count
[1] {citrus fruit, root vegetables}   => {other vegetables} 0.01037112 0.5862069  0.01769192 3.029608 102
[2] {tropical fruit, root vegetables} => {other vegetables} 0.01230300 0.5845411  0.02104728 3.020999 121
[3] {root vegetables, rolls/buns}     => {other vegetables} 0.01220132 0.5020921  0.02430097 2.594890 120
[4] {root vegetables, yogurt}         => {other vegetables} 0.01291307 0.5000000  0.02582613 2.584078 127
[5] {yogurt, whipped/sour cream}      => {other vegetables} 0.01016777 0.4901961  0.02074225 2.533410 100
```

The rules with the 5 highest lift are shown above. A lift greater than 1 means that the rule is strong. The 5 highest lift range from 3.03 to 2.53 which means these rules are pretty strong.

The 62 rules that we generated with our criteria of 0.1 support and 0.4 confidence were interesting to see because the antecedents were a wide range of items such as hard cheese, ham, fruits, etc. while the consequent items for all 62 rules were either whole milk or other vegetables. This seems to make sense because the 2 most frequent purchases were whole milk and other vegetables. Since we set our confidence level pretty high and these were already frequently purchased items, it was not a surprise to see that either whole milk and other vegetables were in every single one of these association rules.

```
#Case 5

data("iris")
#for the training and test sets
index <- sample(nrow(iris), nrow(iris) * 0.9)
iris_train = iris[index,]
iris_test = iris[-index,]

# K-Means Cluster Analysis
fit <- kmeans(iris_train[,1:4], 2) #2 cluster solution
# Display number of clusters in each cluster
table(fit$cluster)

# Plot cluster in kmeans
plotcluster(iris_train[, 1:4], fit$cluster)

# See exactly which items are in 1st group
iris_train$Species[fit$cluster == 1]
iris_train$Species[fit$cluster == 2] #2nd group


# get cluster means
aggregate(iris_train[,1:4], by = list(fit$cluster), FUN = mean)

# ~~~~~~~~~~~~~~~ for the 4 clusters
# K-Means Cluster Analysis
fit <- kmeans(iris_train[,1:4], 4) #4 cluster solution
# Display number of clusters in each cluster
table(fit$cluster)

# Plot cluster in kmeans
plotcluster(iris_train[, 1:4], fit$cluster)

# See exactly which items are in 1st group
iris_train$Species[fit$cluster == 1]
iris_train$Species[fit$cluster == 2] #2nd group
iris_train$Species[fit$cluster == 3]
iris_train$Species[fit$cluster == 4]
```

```
# get cluster means
aggregate(iris_train[,1:4], by = list(fit$cluster), FUN = mean)
```

```
#hierarchical
# Hierarchical clustering Calculate the distance matrix
Iris.dist = dist(iris_train[,1:4])
# Obtain clusters using the Wards method
Iris.hclust = hclust(Iris.dist, method = "ward.D")
plot(Iris.hclust)

# Cut dendrogram at the 3 clusters level and obtain cluster membership
groupIris.2 = cutree(Iris.hclust, k = 2)
table(groupIris.2)

# See exactly which item are in third group
iris_train$Species[groupIris.2 == 2]

# get cluster means for raw data
aggregate(iris_train[,1:4], by = list(groupIris.2), FUN = mean)

# Centroid Plot against 1st 2 discriminant functions Load the fpc library
# needed for plotcluster function
library(fpc)
plotcluster(iris_train[,1:4], groupIris.2)

# Hierarchical clustering Calculate the distance matrix
Iris.dist = dist(iris_train[,1:4])
# Obtain clusters using the Wards method
Iris.hclust = hclust(Iris.dist, method = "ward.D")
plot(Iris.hclust)

# Cut dendrogram at the 3 clusters level and obtain cluster membership
groupIris.4 = cutree(Iris.hclust, k = 4)
table(groupIris.4)
```

```
# See exactly which item are in third group
iris_train$Species[groupIris.4 == 4]

# get cluster means for raw data
aggregate(iris_train[,1:4], by = list(groupIris.4), FUN = mean)

# Centroid Plot against 1st 2 discriminant functions Load the fpc library
# needed for plotcluster function
library(fpc)
plotcluster(iris_train[,1:4], groupIris.4)



#~~~~~~~~~~~~~~~~~~~~~~ 2) data exploration
## The library "arules" implements association rules
install.packages("arules")
library ("arules")
data(Groceries)

Groceries
summary(Groceries)
# a) dimensions: 9835 transactions & 169 items or products

## b) Look at the first 10 transactions
inspect(Groceries[1:10])

# c) most frequent purchased item is in the summary, its whole milk w 2513 purchases
summary(Groceries)

# d) is also in the summary, and the mean is 4.409 items

# e) also in summary, the max is 32

# f) item frequency plot of all items
itemFrequencyPlot(Groceries)

# g) item frequency plot showing only those items with a support of at least 10%
itemFrequencyPlot(Groceries, support = 0.1, cex.names = 0.8)
```

```
# ~~~~~~~~~~~~~~~~~~~~ association rules
# a)
## The function for generating association rules is apriori
## Generates rules that have a single item as their consequent
# support = 0.01 and confidence = 0.4
rules <- apriori(Groceries, parameter = list(support = 0.01,
                                 confidence = 0.4))

## b) Print out the number of rules = 62
rules

## c) distribution A summary of the rules generated
# 25 rules with 2 items and 37 rules with 3 items
summary(rules)

# d) the average lift is 1.993
inspect(rules)

# e) 5 highest lift
inspect(head(sort(rules, by = "lift"), n = 5)) # to show 5 highest lift
```