**INTRODUCTION**

Identifying the key predictors of mpg is relevant for the customers and dealers since fuel efficiency is an important consideration in the automobile market. With the cost of oil and gas prices rising in recent years, maximizing a vehicles miles per gallon means a lower day to day operating cost of the vehicle. Evidence that oil prices will rise again (Fournier, J., et al. (2013)) means that maximizing a vehicles miles per gallon will continue to become more essential information to consumers and auto manufacturers alike. We have analyzed the AutoMPG dataset obtained from the UCI Machine Learning Repository (Dua, D. and Karra Taniskidou, E. (2017)). The dataset contains 398 rows and 8 attributes. The response variable is mpg (continuous) and other variables are displacement (continuous), horsepower (continuous), weight (continuous), acceleration (continuous), cylinders (multi-valued discrete), origin (multi-valued discrete), model year (multi-valued discrete), and car name (string). Our analysis considers displacement, horsepower, weight, acceleration, cylinders, origin and model year. We have conducted ordinary least square regression, stepwise selection using AIC and BIC, Ridge and Lasso regression with cross-validation and Principle Component Regression. Square transformation on displacement and square transformation on weight are conducted and this transformed data is used in when fitting Ordinary Least Squares, Lasso, and Ridge Regression.

**METHODS**

**Data Set**

The data set used in this project was the Miles Per Gallon Data Set from UCI Machine Learning Repository. This data set consisted of 398 observations, and our response variable was miles per gallon. We originally had eight predictor variables, but we removed the predictor “model name“ based on the fact there were almost no observations with the same model name and so the predictor functioned more as an index. Of the seven remaining predictors, origin, cylinders, and model year were categorical with the following integer level ranges: 1-3 (origin), 3-8 (cylinders) and 70-82 (model year). The other predictors weight, accel, displacement and horsepower were real valued continuous variables.

**Cleaning and Set Up**

During the initial summary statistics, we noticed that the cylinder variable had only a few observations in the 3 and 5 cylinder levels which made physical sense because it is pretty rare to see cars with 3 or 5 cylinders. As a result we decided to combine the 3 cylinder observation into the 4 cylinder level and the 5 cylinder observations into the 6 cylinder level. In addition, we decided to reduce the number of levels in the model year factor by combining the years 70-73, 74-77, 78-82. The predictor was then recoded with factor levels 1, 2 and 3, respectively. The data was then split into training and test data with 75% of the data used for training set and the remaining 25% used for the testing data set. For reproduction of the code in R, a seed was set as set.seed(123).

**Statistical Analysis**

We first used the Ordinary Least Squares method to fit a multiple linear regression model. In addition, two different transformation methods were employed on the predictors, and the resulting transformed predictors were used analogously to fit linear regression models.

Full Model:

transformation of displacement:

transformation of weight:

Model Selection was then performed through backward AIC/BIC (Witten et al. 2017) selection and as well as a composite forward/backward selection to see which predictors were needed. The following are the new models after model selection techniques were applied.

Full Model:

transformation of displacement:

transformation of weight:

Cross Validation with K=5, 10, and N (Leave One Out) folds was then used to compare models. The linear regression models were also compared with AIC, BIC, adjusted R2 and mean squared error (Witten et al. 2017).

Following linear regression, we ran Ridge and Lasso penalized linear regression models to see if these models were an improvement. Two different Ridge and two different Lasso regression models were run where the first Ridge and Lasso were on the full model with all predictors included. The second Ridge and Lasso model were run on the best transformation of the predictors that was determined during the ordinary least squares linear regression. Cross validation was used to find the best tuning parameter for each Ridge and Lasso regression model. Mean square error was calculated for each model so the models could be compared with each other and with the ordinary least squares linear regression models. The final approach to improve our prediction was a principal component regression. Similar to Lasso and Ridge, we used cross validation to determine the optimum number of components to be used and then calculated mean square error to compare with other models.

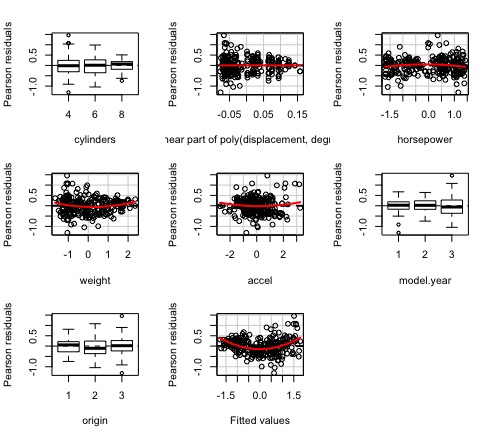
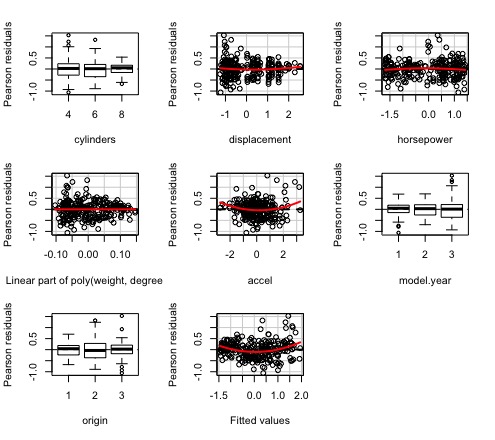
**RESULTS**

**Exploratory Data Analysis and Diagnostics**

We initially fit an ordinary least squares regression model on the training set of the full seven predictors. Prior to model building, several diagnostic plots were generated including residual plots, component residual plots and influence plots.

**FigureResidual Plot of Full Model
 1. Residual plots of the raw predictor variables.**

The residual plots in Fig. 1, as well as the other component residual and CERES plots (not shown), suggest some predictors may benefit from transformations. The predictors were standardized before transforming. The displacement and weight variables were then subjected to second order polynomial transformations.

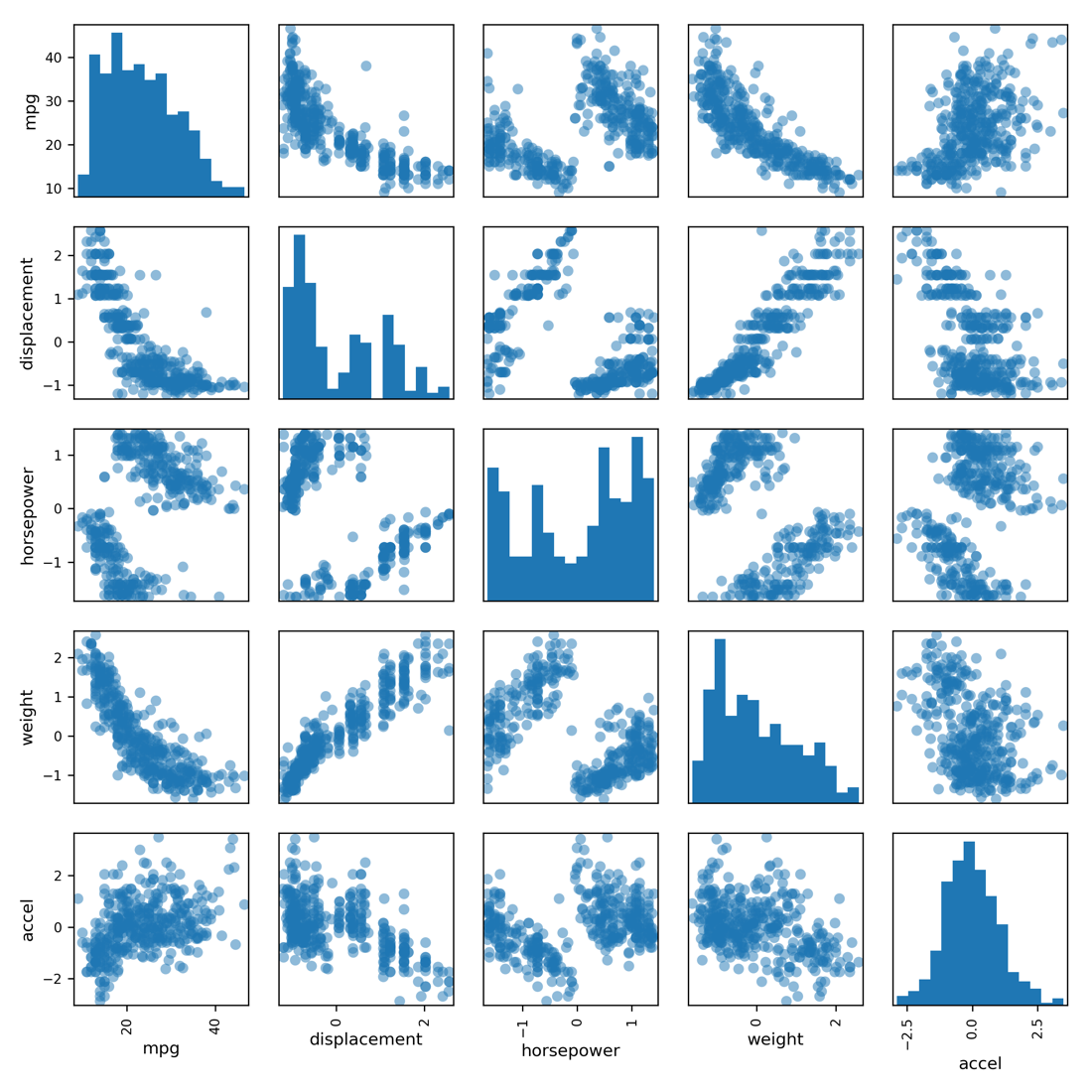
**Figure 2. Residual plots after second order transformations of the displacement predictor (left) and weight predictor (right).**

As seen in Fig. 2, both transformations helped correct some of the initial problems. Performing transformations weight and displacement, while not transforming the other continuous variables accel and horsepower, was based on the cumulative results that horsepower was shown not to be significant in the hypothesis testing of as well as it being removed from backward and both directional AIC stepwise model selection. For acceleration, we did try a few transformations and they seemed to have little to no benefit to the model.

We also checked for collinearity problems in our data using variance inflation factors. The variance inflation factors for each predictor is shown in the table below. None of the variance inflation factors were greater than five or ten (Belsley et. al), meaning multicollinearity did not appear to be an issue with this data set.

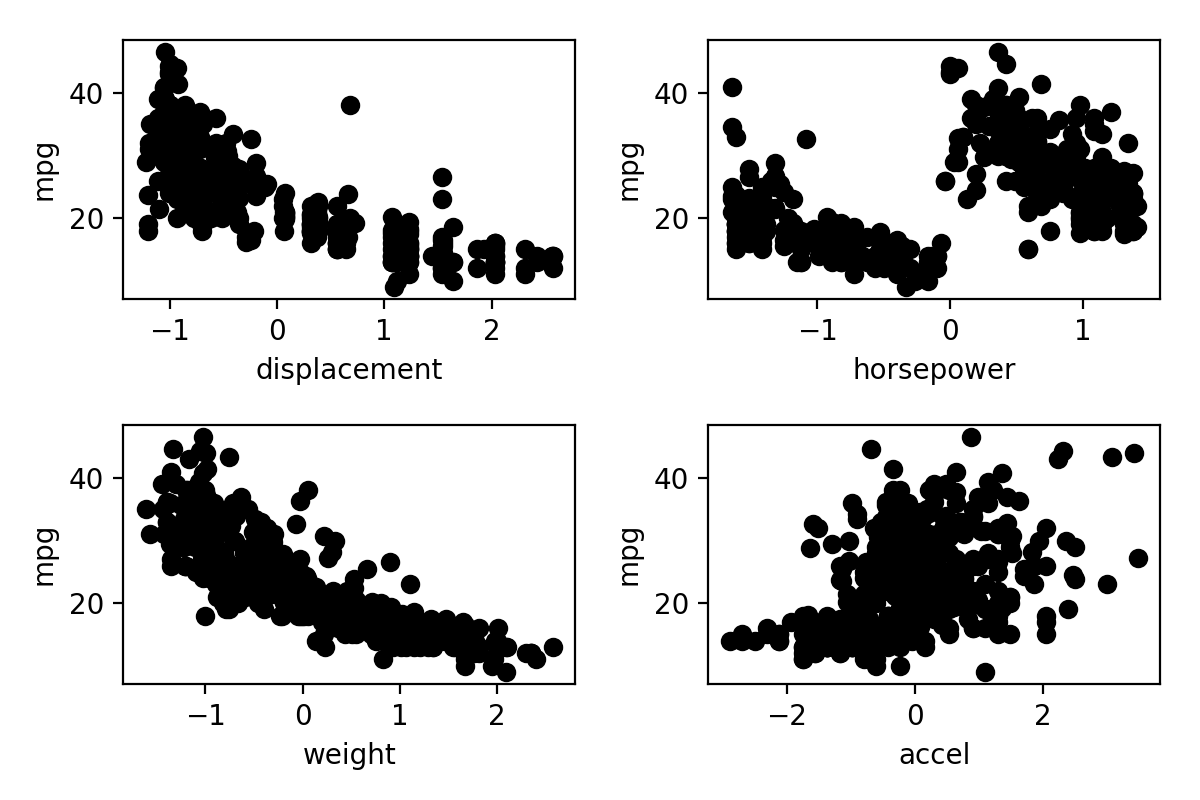
**Table 1. Variance inflation factors for the predictor variables.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Variance Inflation Factors | | | | | | |
| Cylinders | Displacement | Horsepower | Weight | Acceleration | Model.Year | Origin |
| 1.928 | 4.678 | 1.245 | 3.032 | 1.288 | 1.062 | 1.222 |

In addition to variance inflation factors, we also checked the correlation between predictors. Visually, the results can be seen in the correlation Fig. 3 below. Correlation between variables was deemed negligible; however, we did try fitting Ridge, Lasso, and principle component regression models to see if they would improve over the ordinary least squares model and help mitigate some of the problems that might arise from **collinearity being present in the data. One unexpected result from Fig. 3 is that**

**Figure 3. Correlation of predictors after standardization.**

horsepower has a non-linear relationship with every other predictor including the response variable (mpg). This adds further evidence, in addition to hypothesis testing and stepwise selection, that dropping horsepower from our model would improve it. Fig. 4 below shows the continuous predictors versus only the response where a linear relationship is desired based on the assumptions of ordinary least squares. Again, the non-linear relationship between mpg and horsepower supports its removal from any models. Furthermore, Fig. 4 provides supporting evidence for why a transformation on displacement and weight may improve the model. Both displacement and weight have a slight curvature to their relationship with mpg. Moreover, acceleration has a weak relationship with mpg, which is likely why a transformation on acceleration does not seem to have much of an effect.

 **Figure 4. Each continuous predictor after being standardized versus the response variable.**

**Ordinary Least Squares and Cross Validation**

After the exploratory data analysis, we have three different models to try and fit including the full model (1), x2 transformation of displacement model (2), and the x2 transformation of weight (3). As was mentioned briefly in previous sections, we also used backward stepwise selection and forward-backward stepwise selection to further refine our models. In all instances, backward stepwise selection and forward-backward stepwise selection yielded the same results and will be referred to as just stepwise selection from here on. For the full model, stepwise selection eliminated the horsepower predictor. Using stepwise selection on the x2 displacement model, we found that origin and cylinder predictors were removed. Stepwise selection on the x2 transformation of weight model removed horsepower, displacement and cylinders. The stepwise selection reduced models give us three new models to compare in addition to the original full and transformed model and are notated as (4), (5) and (6), respectively in the methods section.

We then fit the six models and used the adjusted R2, AIC, BIC and mean squared error metrics to compare. The results are in Table 2.

**Table 2. Results from the final six models.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ordinary Least Squares Results | | | | | | |
|  | Full Model | Reduced Full Model | X2 Displacement Model | Reduced X2 Displacement Model | X2 Weight Model | Reduced X2 Weight Model |
| AIC | 293.605 | 292.609 | 269.265 | 264.553 | 260.134 | 257.319 |
| BIC | 336.380 | 331.819 | 315.604 | 296.633 | 306.472 | 289.400 |
| Adjusted R2 | 0.828 | 0.828 | 0.844 | 0.845 | 0.849 | 0.849 |
| Mean Squared Error | 0.186 | 0.183 | 0.183 | 0.187 | 0.149 | 0.146 |

All metrics indicate the best model between the six is the X2 weight transformation with the stepwise selection-suggested predictors removed. This model has the lowest AIC, BIC and mean squared error of the six models as well as being tied for highest adjusted R2.

To further confirm our results, we also ran cross validation for the six models. We use K=5 and K=10 folds and compared using the root mean squared error (RMSE) and adjusted R2 metrics. The results are in the following table.

**Table 3. Results from the final six models after using cross validation.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cross Validation of Ordinary Least Squares Models | | | | |
|  | K=10 RMSE | K=10 R2 | K=5 RMSE | K=5 R2 |
| Full Model | 0.427 | 0.825 | 0.420 | 0.829 |
| Reduced Model | 0.419 | 0.839 | 0.425 | 0.824 |
| X2 Displacement Model | 0.407 | 0.840 | 0.406 | 0.839 |
| Reduced X2 Displacement Model | 0.391 | 0.848 | 0.397 | 0.844 |
| X2 Weight Model | 0.397 | 0.848 | 0.391 | 0.851 |
| Reduced X2 Weight Model | 0.390 | 0.847 | 0.397 | 0.842 |

The results of cross validation are similar but provide some alternative insights from before. The weight transformation again possesses the best improvement. However, in cross validation the full X2 transformed weight model has highest Adjusted R2 for K=10 and K=5 as well as lowest RMSE for K=5 folds. The reduced X2 transformation on weight model has the lowest RMSE for K=10 folds. This is different from the previous result because the reduced X2 transformation on weight model was supported as best by all metrics; however, in cross validation three of the four metrics support the full X2 transformation of weight model but one metric supports the reduced X2 transformation of weight.

**Ridge and Lasso Regression**

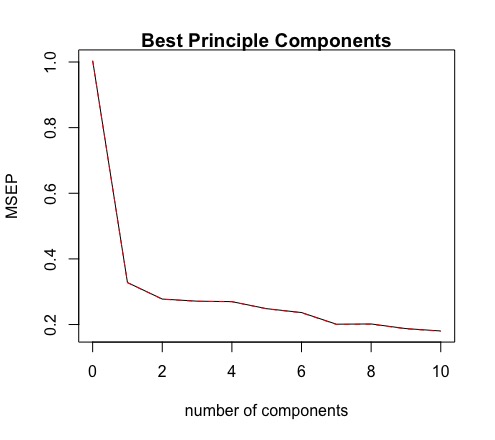
Following ordinary least squares, we fit both Ridge and Lasso regression models to the data. Since our previous results showed that the X2 transformation of weight as an improvement and as the best transformation on the data, we also fit a Ridge and Lasso regression model to the transformed data. This resulted in two Ridge regression models and two Lasso regression models. The tuning parameter for both Ridge and Lasso was determined using cross validation. For Ridge regression, the alpha value was fixed and set at zero. For Lasso, the alpha value was fixed at one. Our results from the four models are shown in below in Table 4.

**Table 4. Results from the four models subjected to Ridge and Lasso regression.**

|  |  |
| --- | --- |
| Ridge and LASSO Regression Results | |
| Type of Model | Mean Square Error |
| Ridge Full Model | 0.207 |
| Ridge Transformed Data Model | 0.169 |
| LASSO Full Model | 0.198 |
| LASSO Transformed Data Model | 0.150 |

Overall, Lasso performed better than Ridge regression. Fitting Lasso and Ridge regression models to the transformed data also helped performance. Interestingly, coefficients removed in the Lasso regression models were horsepower and displacement; in contrast, the full model and Lasso model fit to the transformed data only removed horsepower. These results agree with our findings from exploratory data analysis and ordinary least squares with respect to the lack of information provided by the horsepower variable.

**Principle Component Regression**

The last regression method evaluated was principle component regression. This method converts the data into a new X space and then searches for the best number of principle components that describe the variability in the data. We first fit a principle component regression model to the full data set. Fig. 5 below shows the error versus the number of principle components. We can see that the best number of principle components to include in order to minimize error for our principle component regression model is all 10 of the principle components. Selecting all possible principle components to use is equivalent to ordinary least squares regression.

**Figure 5. Error versus the number of principle components.**

Since our findings show the best number of principle components to use is all of them, we are essentially saying that ordinary least squares is better than any possible principle component regression we could perform on this data. In essence we stop here and conclude principle component regression will not give us a better model than any of our previous work.

**Conclusion**

**Table 5. Mean square error comparison between all models**

|  |  |
| --- | --- |
| Mean Square Error Final Results | |
| Model | MSE |
| OLS Full Model | 0.186 |
| OLS Reduced Model | 0.183 |
| OLS X2 Displacement Model | 0.183 |
| OLS Reduced X2 Displacement Model | 0.187 |
| OLS X2 Weight Model | 0.149 |
| OLS Reduced X2 Weight Model | 0.146 |
| Ridge Full Model | 0.207 |
| Ridge Transformed Model | 0.169 |
| LASSO Full Model | 0.198 |
| LASSO Transformed Model | 0.150 |

The research goal of our project was to minimize the squared error loss (mean squared error) when predicting miles per gallon given information on engine cylinders, engine displacement, horsepower, weight, acceleration, origin and model year. The above table shows our final mean squared error results for the different models we fit. The model with the lowest mean squared error was the ordinary least squares model fit to the weight squared transformation of the data with cylinders, displacement and horsepower removed based on stepwise model selection. However, a few of the models performed almost as well, including the ordinary least squares performed on the weight squared transformation with no predictors removed. Although the AIC, BIC and adjusted R2 metrics supported the reduced weight squared model, cross validation was more in favor of the weight squared model with no predictors removed. A case for either of those models could be made depending on how much importance having displacement, cylinders and horsepower information in your model was to your goals. The other model that was very close was the Lasso regression fit to the weight squared transformed data. Although it had the third lowest mean squared error, this would be the model I would argue for going forward. This model is a nice middle ground between ordinary least squares model on the weight squared data with no predictors removed and the ordinary least squares model on the weight squared data with stepwise selection predictors removed since the Lasso model removed some of the predictors that the stepwise did as well as penalized the coefficients. Going forward, some small changes to the Lasso regression method on the transformed data could see it outperform ordinary least squares such as cross validation to find the alpha tuning parameter where in our case we designated lasso equal to one. Another change could be to find a better way to include weight squared in the Lasso regression model. During the transformation of the data, we included both weight and weight squared in matrix to preserve hierarchical structure but we could also have tried orthogonal polynomials so that only weight squared was used for the LASSO regression.

**References**

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017). *An introduction to statistical learning with applications in R*. New York: Springer.

Belsley D.A, Kuh E. and Welsch R.E. (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. New York: John Wiley & Son.

Fournier, J., et al. (2013), "The Price of Oil – Will it Start Rising Again?", *OECD Economics Department Working Papers*, No. 1031, OECD Publishing, Paris, <https://doi.org/10.1787/5k49q186vxnp-en>.

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Hastie, T., Tibshirani, R.,, Friedman, J. (2001). *The Elements of Statistical Learning*. New York, NY, USA: Springer New York Inc..

**Appendix**

**R Code**

### Data Set Up

# load the library

library(RCurl)

# specify the URL for the MPG data CSV

urlfile <-'https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'

# download the file

downloaded <- getURL(urlfile, ssl.verifypeer=FALSE)

# treat the text data as a steam so we can read from it

connection <- textConnection(downloaded)

# parse the downloaded data as CSV

dataset <- read.table(connection ,header=FALSE)

# preview the first 5 rows

head(dataset)

colnames(dataset)<-c("mpg", "cylinders", "displacement","horsepower", "weight", "accel", "model.year","origin", "car.name")

MPGdat<-dataset

MPGdat$origin<-as.factor(MPGdat$origin)

## Some Horsepower Observations have empties

badRows<-vector(mode="numeric", length=0)

for (i in 1:length(dataset$horsepower)) {

if(dataset$horsepower[i]=="?"){

badRows<-c(badRows, i)

}

}

badRows

## Remove Rows with empty horsepower observations

for (i in 1:length(badRows)) {

MPGdat<-MPGdat[-c(badRows[i]),]

}

MPGdat$horsepower<-as.numeric(MPGdat$horsepower)

## Histograms and tables for MPG data

for (i in 1:8 ){

if(is.factor(MPGdat[,i])==FALSE){

hist(MPGdat[,i], main = colnames(MPGdat)[i])}

else{

if(i==2){print("cylinders:")}

if(i==7){print("model year:")}

if(i==8){print("origin:")}

print(table(MPGdat[,i]))

}

}

## Getting rid of 3 and 5 cyclinders

for (i in 1:length(MPGdat$cylinders)) {

if(MPGdat$cylinders[i]==3){

MPGdat$cylinders[i]<-4

}

if(MPGdat$cylinders[i]==5){

MPGdat$cylinders[i]<-6

}

}

table(MPGdat$cylinders)

MPGdat$cylinders<-as.factor(MPGdat$cylinders)

## Combining Model Years

for (i in 1:length(MPGdat$model.year)) {

if(MPGdat$model.year[i]==70 | MPGdat$model.year[i]==71 | MPGdat$model.year[i]==72 | MPGdat$model.year[i]==73){

MPGdat$model.year[i]<-1

}

if(MPGdat$model.year[i]==74 | MPGdat$model.year[i]==75 | MPGdat$model.year[i]==76 | MPGdat$model.year[i]==77){

MPGdat$model.year[i]<-2

}

if(MPGdat$model.year[i]>=78 ){

MPGdat$model.year[i]<-3

}

}

table(MPGdat$model.year)

MPGdat$model.year<-as.factor(MPGdat$model.year)

### Training and Test Data

require(caTools)

set.seed(123) # set seed to ensure you always have same random numbers generated

sample = sample.split(MPGdat,SplitRatio = 0.75) # splits the data in the ratio mentioned in SplitRatio. After splitting marks these rows as logical TRUE and the the remaining are marked as logical FALSE

train =subset(MPGdat,sample ==TRUE)

train$car.name=NULL

# creates a training dataset named train with rows which are marked as TRUE

test=subset(MPGdat, sample==FALSE)

test$car.name=NULL

# creates a test dataset named test with rows which are marked as FALSE

require(caret)

## standardizing test and training

preprocessParams <- preProcess(train[,1:8], method=c("center", "scale"))

print(preprocessParams)

train.std <- predict(preprocessParams, train[,1:8])

test.std<-predict(preprocessParams, test[,1:8])

train<-train.std

test<-test.std

## Histograms and tables of train data

for (i in 1:8 ){

if(is.factor(train[,i])==FALSE){

hist(train[,i], main = colnames(train)[i])}

else{

if(i==2){print("cylinders:")}

if(i==7){print("model year:")}

if(i==8){print("origin:")}

print(table(train[,i]))

}

}

##Histograms and tables of test

for (i in 1:8 ){

if(is.factor(test[,i])==FALSE){

hist(test[,i], main = colnames(test)[i])}

else{

if(i==2){print("cylinders:")}

if(i==7){print("model year:")}

if(i==8){print("origin:")}

print(table(test[,i]))

}

}

### Simple Linear Regression and Diagnostics

require(car)

require(MASS)

mpgMod<-lm(mpg~. , data=train)

summary(mpgMod)

vif(mpgMod)

## StepAIC

stepAIC(mpgMod, direction = "backward")

stepAIC(mpgMod, direction = "both")

## Throws out horsepower

## Diagnostic Plots

residualPlots(mpgMod)

crPlots(mpgMod)

ceresPlots(mpgMod)

influenceIndexPlot(model = mpgMod, id.n = 5)

## Predict on Test Set

pMod1<-predict(mpgMod, test)

actualsMod1<-data.frame(cbind(actuals=test$mpg, predicteds=pMod1))

head(actualsMod1)

EPEmod1<- mean( (test$mpg - pMod1)^2 )

EPEmod1

## Removing Horsepower based on StepAIC and pvalues

mpgMod2<-lm(mpg~ cylinders + displacement + weight + accel + model.year + origin, data=train)

summary(mpgMod2)

stepAIC(mpgMod2, direction = "backward")

stepAIC(mpgMod2, direction = "both")

## Diagnostic Plots

residualPlots(mpgMod2)

crPlots(mpgMod2)

ceresPlots(mpgMod2)

## Predict on Test Data

pMod2<-predict(mpgMod2, test)

actualsMod2<-data.frame(cbind(actuals=test$mpg, predicteds=pMod2))

head(actualsMod1)

EPEmod2<- mean( (test$mpg - pMod2)^2 )

EPEmod2

## X^2 transformation of displacement

mpgMod3<-lm(mpg~ cylinders + poly(displacement, degree = 2) + horsepower + weight + accel + model.year + origin, data=train)

summary(mpgMod3)

## StepAIC

stepAIC(mpgMod3, direction = "backward")

stepAIC(mpgMod3, direction = "both")

## Removes origin and cylinders

##Diagnostic Plots

residualPlots(mpgMod3)

crPlots(mpgMod3)

##Predict On Test Data

pMod3<-predict(mpgMod3, test)

actualsMod3<-data.frame(cbind(actuals=test$mpg, predicteds=pMod3))

head(actualsMod3)

EPEmod3<- mean( (test$mpg - pMod3)^2 )

EPEmod3

## x^2 displacement with origin and cylinders removed based on StepAIC

mpgMod4<-lm(mpg~ poly(displacement, degree = 2) + weight + horsepower + accel + model.year, data=train)

summary(mpgMod4)

## StepAIC

stepAIC(mpgMod4, direction = "backward")

stepAIC(mpgMod4, direction = "both")

## Diagnostic Plots

residualPlots(mpgMod4)

crPlots(mpgMod4)

## Predict on Test Data

pMod4<-predict(mpgMod4, test)

actualsMod4<-data.frame(cbind(actuals=test$mpg, predicteds=pMod4))

head(actualsMod4)

EPEmod4<- mean( (test$mpg - pMod4)^2 )

EPEmod4

## R^2, AIC, BIC, EPE Results Comparison

AdjR2<-c(summary(mpgMod)$adj.r.squared,summary(mpgMod2)$adj.r.squared, summary(mpgMod3)$adj.r.squared, summary(mpgMod4)$adj.r.squared)

AdjR2

LinRegAIC<-c(AIC(mpgMod), AIC(mpgMod2), AIC(mpgMod3), AIC(mpgMod4))

LinRegAIC

LinRegBIC<-c(BIC(mpgMod), BIC(mpgMod2), BIC(mpgMod3), BIC(mpgMod4))

LinRegBIC

LinRegEPE<-c(EPEmod1, EPEmod2, EPEmod3, EPEmod4)

LinRegEPE

## x^2 transformation on weight

mpgMod5<-lm(mpg~ cylinders + displacement + horsepower + poly(weight, degree =2) + accel + model.year + origin , data=train)

summary(mpgMod5)

##StepAIC

stepAIC(mpgMod5, direction = "backward")

stepAIC(mpgMod5, direction = "both")

## Removes displacement, hp, and cylinders

## Diagnostic Plots

residualPlots(mpgMod5)

crPlots(mpgMod5)

ceresPlots(mpgMod5)

## Predict on Test Set

pMod5<-predict(mpgMod5, test)

actualsMod5<-data.frame(cbind(actuals=test$mpg, predicteds=pMod5))

head(actualsMod5)

EPEmod5<- mean( (test$mpg - pMod5)^2 )

EPEmod5

## weight X^2 w/ hp, displacement and cylinders thrown out from stepAIC

mpgMod6<-lm(mpg~ accel+ poly(weight, degree = 2) + model.year + origin, data=train)

summary(mpgMod6)

stepAIC(mpgMod6, direction = "backward")

stepAIC(mpgMod6, direction = "both")

## Diagnostic Plots

residualPlots(mpgMod6)

crPlots(mpgMod6)

## Predict on Test Data

pMod6<-predict(mpgMod6, test)

actualsMod6<-data.frame(cbind(actuals=test$mpg, predicteds=pMod6))

head(actualsMod6)

EPEmod6<- mean( (test$mpg - pMod6)^2 )

EPEmod6

## Results comparison

AdjR2<-c(AdjR2, summary(mpgMod5)$adj.r.squared, summary(mpgMod6)$adj.r.squared)

round(AdjR2, 3)

LinRegAIC<-c(LinRegAIC, AIC(mpgMod5), AIC(mpgMod6))

round(LinRegAIC, 3)

LinRegBIC<-c(LinRegBIC, BIC(mpgMod5), BIC(mpgMod6))

round(LinRegBIC, 3)

LinRegEPE<-c(LinRegEPE,EPEmod5, EPEmod6)

round(LinRegEPE, 3)

### Cross Validation of the six models

library(caret)

cv\_10fold = trainControl(method = "cv", number = 10)

cv\_loo = trainControl(method = "LOOCV")

cv\_5fold = trainControl(method = "cv", number = 5)

mpgModf1 = mpg~.

mpgModf2 = mpg~ cylinders + displacement + weight + accel + model.year + origin

mpgModf3 = mpg~ cylinders + poly(displacement, degree = 2) + horsepower + weight + accel + model.year + origin

mpgModf4 = mpg~ poly(displacement, degree = 2) + weight + accel + model.year

mpgModf5 = mpg~ cylinders + displacement + horsepower + poly(weight,degree = 2) + accel + model.year + origin

mpgModf6 = mpg~ poly(weight, degree = 2) + model.year + origin

## K=10 Cross Validation

modela10 = train(mpgModf1, data = train, trControl = cv\_10fold,

method = "lm")

modelb10 = train(mpgModf2, data = train, trControl = cv\_10fold,

method = "lm")

modelc10 = train(mpgModf3, data = train, trControl = cv\_10fold,

method = "lm")

modeld10 = train(mpgModf4, data = train, trControl = cv\_10fold,

method = "lm")

modele10 = train(mpgModf5, data = train, trControl = cv\_10fold,

method = "lm")

modelf10 = train(mpgModf6, data = train, trControl = cv\_10fold,

method = "lm")

print(modela10)

print(modelb10)

print(modelc10)

print(modeld10)

print(modele10)

print(modelf10)

## K=N LOOCV

modelaLOO = train(mpgModf1, data = train, trControl = cv\_loo,

method = "lm")

modelbLOO = train(mpgModf2, data = train, trControl = cv\_loo,

method = "lm")

modelcLOO = train(mpgModf3, data = train, trControl = cv\_loo,

method = "lm")

modeldLOO = train(mpgModf4, data = train, trControl = cv\_loo,

method = "lm")

modeleLOO = train(mpgModf5, data = train, trControl = cv\_loo,

method = "lm")

modelfLOO = train(mpgModf6, data = train, trControl = cv\_loo,

method = "lm")

print(modelaLOO)

print(modelbLOO)

print(modelcLOO)

print(modeldLOO)

print(modeleLOO)

print(modelfLOO)

## K=5 Fold Cross Validation

modela5 = train(mpgModf1, data = train, trControl = cv\_5fold,

method = "lm")

modelb5 = train(mpgModf2, data = train, trControl = cv\_5fold,

method = "lm")

modelc5 = train(mpgModf3, data = train, trControl = cv\_5fold,

method = "lm")

modeld5 = train(mpgModf4, data = train, trControl = cv\_5fold,

method = "lm")

modele5 = train(mpgModf5, data = train, trControl = cv\_5fold,

method = "lm")

modelf5 = train(mpgModf6, data = train, trControl = cv\_5fold,

method = "lm")

print(modela5)

print(modelb5)

print(modelc5)

print(modeld5)

print(modele5)

print(modelf5)

## Results Comparison

tableColNames<-c("K10 RMSE", "K10 R^2", "K5 RMSE", "K5 R^2", "LOO RMSE", "LOO R^2")

ModelA\_CV<-c(modela10$results$RMSE, modela10$results$Rsquared, modela5$results$RMSE, modela5$results$Rsquared, modelaLOO$results$RMSE, modelaLOO$results$Rsquared)

names(ModelA\_CV)<-tableColNames

ModelB\_CV<-c(modelb10$results$RMSE, modelb10$results$Rsquared, modelb5$results$RMSE, modelb5$results$Rsquared, modelbLOO$results$RMSE, modelbLOO$results$Rsquared)

names(ModelB\_CV)<-tableColNames

ModelC\_CV<-c(modelc10$results$RMSE, modelc10$results$Rsquared, modelc5$results$RMSE, modelc5$results$Rsquared, modelcLOO$results$RMSE, modelcLOO$results$Rsquared)

names(ModelC\_CV)<-tableColNames

ModelD\_CV<-c(modeld10$results$RMSE, modeld10$results$Rsquared, modeld5$results$RMSE, modeld5$results$Rsquared, modeldLOO$results$RMSE, modeldLOO$results$Rsquared)

names(ModelD\_CV)<-tableColNames

ModelE\_CV<-c(modele10$results$RMSE, modele10$results$Rsquared, modele5$results$RMSE, modele5$results$Rsquared, modeleLOO$results$RMSE, modeleLOO$results$Rsquared)

names(ModelE\_CV)<-tableColNames

ModelF\_CV<-c(modelf10$results$RMSE, modelf10$results$Rsquared, modelf5$results$RMSE, modelf5$results$Rsquared, modelfLOO$results$RMSE, modelfLOO$results$Rsquared)

names(ModelF\_CV)<-tableColNames

## Results Comparison Table

CV\_Results<-rbind(ModelA\_CV, ModelB\_CV, ModelC\_CV, ModelD\_CV, ModelE\_CV, ModelF\_CV)

CV\_Results

### Ridge Regression

## Ridge Full Model

library(glmnet)

y<-train$mpg

x<-data.matrix(train[,-1])

lambda <- 10^seq(10, -2, length = 100)

ridge.mod <- glmnet(x, y, alpha = 0, lambda = lambda)

predict(ridge.mod, s = 0, exact = F, type = 'coefficients')[1:8,]

cv.out <- cv.glmnet(x, y, alpha = 0)

bestlam <- cv.out$lambda.min

xtest<-data.matrix(test[,-1])

ridge.pred <- predict(ridge.mod, s = bestlam, newx = xtest)

EPEridge=mean((ridge.pred-test$mpg)^2)

round(EPEridge, 3)

## weight^2 ridge model

y<-train$mpg

x<-data.matrix(train[,-1])

weight2<-(x[,4]\*\*2)

xTrans<-cbind(x,weight2)

lambda <- 10^seq(10, -2, length = 100)

ridge.mod.2 <- glmnet(xTrans, y, alpha = 0, lambda = lambda)

predict(ridge.mod.2, s = 0, exact = F, type = 'coefficients')[1:9,]

cv.out.2 <- cv.glmnet(xTrans, y, alpha = 0)

bestlam.2 <- cv.out.2$lambda.min

xtest<-data.matrix(test[,-1])

weight2Test<-(xtest[,4]\*\*2)

xtestTrans<-cbind(xtest,weight2Test)

ridge.pred.2 <- predict(ridge.mod.2, s = bestlam, newx = xtestTrans)

EPEridge.2=mean((ridge.pred.2-test$mpg)^2)

round(EPEridge.2, 3)

## Lasso Regression

## Lasso Full Model

lasso.mod <- glmnet(x, y, alpha = 1, lambda = lambda)

cv.out.l <- cv.glmnet(x, y, alpha = 1)

bestlam.l <- cv.out.l$lambda.min

lasso.pred <- predict(lasso.mod, s = bestlam.l, newx = xtest)

EPElasso=mean((lasso.pred-test$mpg)^2)

round(EPElasso, 3)

bestlam.l

lasso.coef=predict(lasso.mod, type="coefficients", s=bestlam.l)[1:8,]

lasso.coef[lasso.coef!=0]

## weight^2 LASSO

lasso.mod.2 <- glmnet(xTrans, y, alpha = 1, lambda = lambda)

cv.out.l.2 <- cv.glmnet(xTrans, y, alpha = 1)

bestlam.l.2 <- cv.out.l.2$lambda.min

lasso.pred.2 <- predict(lasso.mod.2, s = bestlam.l.2, newx = xtestTrans)

EPElasso.2=mean((lasso.pred.2-test$mpg)^2)

round(EPElasso.2, 3)

bestlam.l.2

lasso.coef.2=predict(lasso.mod.2, type="coefficients", s=bestlam.l.2)[1:9,]

lasso.coef.2[lasso.coef!=0]

## Principle Component Analysis

library(pls)

pcr.mod<-pcr(mpg~., data=train, scale=TRUE, validation="CV")

summary(pcr.mod)

par(mfrow=c(1,1))

validationplot(pcr.mod, val.type = "MSEP", main="Best Principle Components")

**Python Code:**

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import pandas as pd

from os.path import expanduser

import numpy as np

from os.path import expanduser

import matplotlib.pyplot as plt

from matplotlib import cm

#Load packages

hd = expanduser('~')

wd = hd+'/Dropbox/math6388/project\_2/'

#Set working directory to Dropbox folder (note - Dropbox must be in your home directory)

mpg\_dat = pd.read\_csv(wd+'mpg\_data.csv')

#Load the data

mpg\_dat.columns.values

list(mpg\_dat)

mpg\_dat.describe()

#Check out the data

mpg\_dat.drop('car.name', axis = 1, inplace = True)

#Delete the car name feature

mpg\_dat = pd.get\_dummies(mpg\_dat, columns = ['cylinders'])

mpg\_dat = pd.get\_dummies(mpg\_dat, columns = ['origin'])

mpg\_dat = pd.get\_dummies(mpg\_dat, columns = ['model.year'])

#Make cylinders, origin, and model year categorical features with dummy indicators

mpg\_dat['cylinders\_3'].value\_counts()

mpg\_dat['cylinders\_4'].value\_counts()

mpg\_dat['cylinders\_5'].value\_counts()

mpg\_dat['cylinders\_6'].value\_counts()

mpg\_dat['cylinders\_8'].value\_counts()

#Explore the cylinders categories

mpg\_dat.loc[mpg\_dat.loc[mpg\_dat['cylinders\_3'] == 1].index.tolist(), 'cylinders\_4'] = 1

mpg\_dat['cylinders\_4'].value\_counts()

mpg\_dat.rename(columns = {'cylinders\_4': 'cylinders\_=<4'}, inplace = True)

mpg\_dat.loc[mpg\_dat.loc[mpg\_dat['cylinders\_5'] == 1].index.tolist(), 'cylinders\_6'] = 1

mpg\_dat['cylinders\_6'].value\_counts()

mpg\_dat.rename(columns = {'cylinders\_6': 'cylinders\_5\_or\_6'}, inplace = True)

#Since there are only 4 vehicles with 3 cylinders and 3 with 5 cylinders, I combined the 3 cylinder vehicles with the 4 cylinder vehicles and renamed it less than or equal to 4 cylinders and combined the 5 cylinder vehicles with the 6 vehicle cylinders and renamed it 6 or 5 cylinder vehicles

mpg\_dat.drop('cylinders\_3', axis = 1, inplace = True)

mpg\_dat.drop('cylinders\_5', axis = 1, inplace = True)

#Deleted the 3 and 4 cylinder columns

mpg\_dat['cylinders\_=<4'].value\_counts()

mpg\_dat['cylinders\_5\_or\_6'].value\_counts()

mpg\_dat['cylinders\_8'].value\_counts()

mpg\_dat.columns.values

#See the new counts and column names

mpg\_dat['origin\_1'].value\_counts()

mpg\_dat['origin\_2'].value\_counts()

mpg\_dat['origin\_3'].value\_counts()

#All these counts look good

mpg\_dat['model.year\_72'].value\_counts()

mpg\_dat['model.year\_73'].value\_counts()

mpg\_dat['model.year\_74'].value\_counts()

mpg\_dat['model.year\_75'].value\_counts()

#Going to combine some of the categories to reduce the number of features

mpg\_dat.loc[mpg\_dat.loc[mpg\_dat['model.year\_70'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_71'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_72'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_73'] == 1].index.tolist(), 'model.year\_70-73'] = 1

mpg\_dat.loc[[x for x in mpg\_dat.index.values.tolist() if x not in mpg\_dat.loc[mpg\_dat['model.year\_70'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_71'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_72'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_73'] == 1].index.tolist()], 'model.year\_70-73'] = 0

mpg\_dat.drop('model.year\_70', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_71', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_72', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_73', axis = 1, inplace = True)

mpg\_dat.loc[mpg\_dat.loc[mpg\_dat['model.year\_74'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_75'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_76'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_77'] == 1].index.tolist(), 'model.year\_74-77'] = 1

mpg\_dat.loc[[x for x in mpg\_dat.index.values.tolist() if x not in mpg\_dat.loc[mpg\_dat['model.year\_74'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_75'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_76'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_77'] == 1].index.tolist()], 'model.year\_74-77'] = 0

mpg\_dat.drop('model.year\_74', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_75', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_76', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_77', axis = 1, inplace = True)

mpg\_dat.loc[mpg\_dat.loc[mpg\_dat['model.year\_78'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_79'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_80'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_81'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_82'] == 1].index.tolist(), 'model.year\_78-82'] = 1

mpg\_dat.loc[[x for x in mpg\_dat.index.values.tolist() if x not in mpg\_dat.loc[mpg\_dat['model.year\_78'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_79'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_80'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_81'] == 1].index.tolist() + mpg\_dat.loc[mpg\_dat['model.year\_82'] == 1].index.tolist()], 'model.year\_78-82'] = 0

mpg\_dat.drop('model.year\_78', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_79', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_80', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_81', axis = 1, inplace = True)

mpg\_dat.drop('model.year\_82', axis = 1, inplace = True)

mpg\_dat['model.year\_70-73'].value\_counts()

mpg\_dat['model.year\_74-77'].value\_counts()

mpg\_dat['model.year\_78-82'].value\_counts()

names = mpg\_dat.columns.values

#Removed the old model year categories and check the final column names

X\_train, X\_test, y\_train, y\_test = train\_test\_split(mpg\_dat.loc[:,mpg\_dat.columns.values[1:]], mpg\_dat.loc[:,['mpg']], random\_state = 0)

#Split the data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train.loc[:,X\_train.columns.values[:]])

#Standardize the training dataset features to a mean of 0 and std of 1

np.mean(X\_train, axis = 0)

np.std(X\_train, axis = 0)

#Make sure the standardization worked

X\_test = scaler.transform(X\_test)

#Use the standardization scaler from the training features dataset on the test features dataset

np.mean(X\_test, axis = 0)

np.std(X\_test, axis = 0)

#Make sure the standardization worked (note the mean and std will not be exactly 0 and 1, respectively, for the testing features dataset since the scaler from the training features dataset is used)

y\_all = pd.concat([y\_train, y\_test])

y\_all.reset\_index(drop = True, inplace = True)

X\_all = pd.DataFrame(np.concatenate((X\_train, X\_test), axis = 0), columns = names[1:14])

all = pd.concat([y\_all, X\_all], axis = 1, ignore\_index = True, sort = False)

all.columns = names

cmap = cm.get\_cmap('gnuplot')

scatter = pd.plotting.scatter\_matrix(all.loc[:,all.columns.values.tolist()[0:5]], marker = 'o', s = 40, hist\_kwds = {'bins':15}, figsize = (9, 9), cmap = cmap)

plt.tight\_layout()

plt.savefig(wd+'cor\_mat.png', dpi = 300)

#Plots the continues features pairwise vs one another with histograms of the distrubtions across the diagonal

cols = 2

rows = (X\_test.shape[1]-9)//2 + (X\_test.shape[1]-9)%2

plt.figure()

gs = plt.GridSpec(rows, cols)

for n in range(X\_test.shape[1]-9):

    i = n%2

    j = n//2

    plt.subplot(gs[j , i])

    plt.scatter(all.loc[:,all.columns.values.tolist()[n+1]], all.loc[:,all.columns.values.tolist()[0]], color = 'black')

    plt.xlabel(all.columns.values.tolist()[n+1])

    plt.ylabel(all.columns.values.tolist()[0])

plt.tight\_layout()

plt.savefig(wd+'mpg\_vs\_feature.png', dpi = 200)

plt.show()