STP 598 Machine Learning Final Project

Jennifer Liu, Anik Jha, Himanshu Aggarwal

May 4, 2017

Question:

Try different methods of fitting a model to usedcars.csv, which contains the selling price of used cars dependent on the qualities of each car. The independent variables are year of the make, mileage, type of trim, type of engine displacement, fuel type, wheel type, whether there was only one owner, color, region, and sound system.

Reading the Data

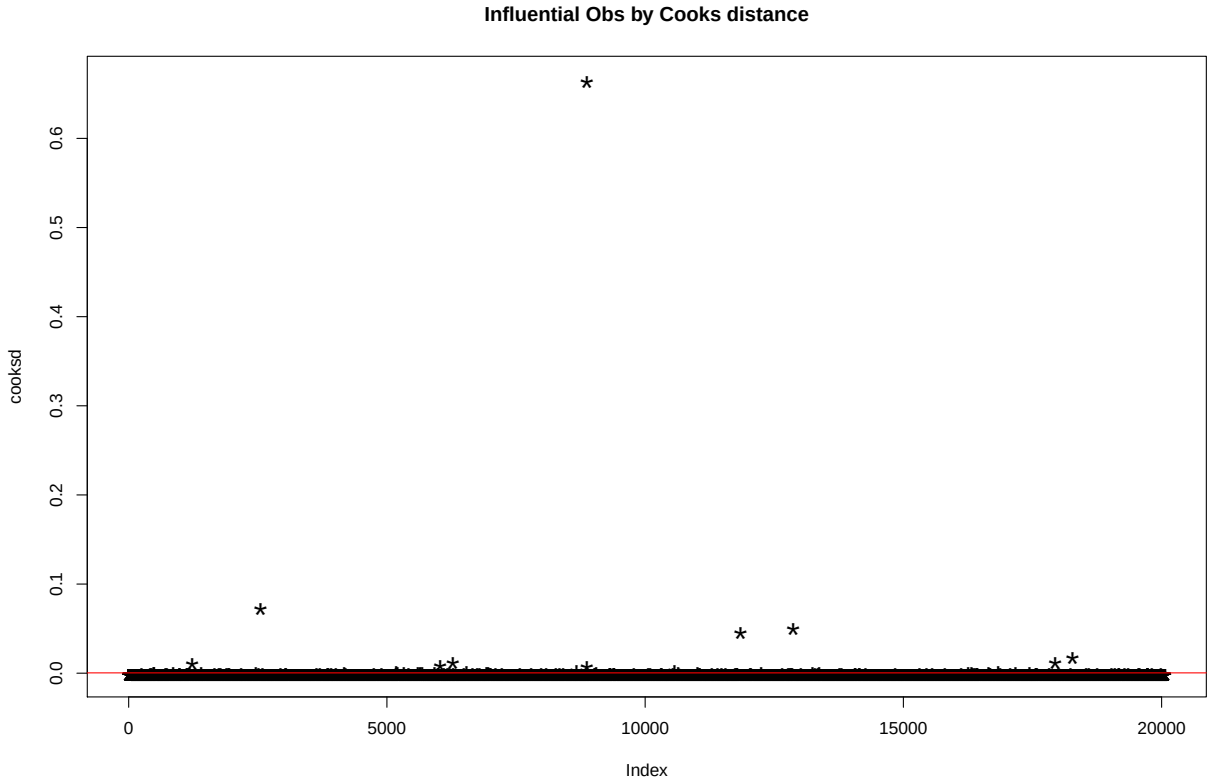
Outlier's detection and treatment

Outliers are the observations that is generally deviated from the expected value by huge margins, which in turn set off any learning algorithm. This makes the outlier's detections and treatment an important step in any machine learning algorithm. Because of the presence of both numerical and categorical variables in the dataset, we decided to run a multivariate outlier detection module based on Cook's distance to overcome the outliers.

Cook's Distance Module for outlier detection

The main objective of the module is to compute the influence each observation creates on the predicted outcome. It works on the basic principle of predicting the outcome with and without the observation for which the influence is to be evaluated. This way we calculate the impact of any given observation on the fitted values. Generally, the observations with cook's distance 4 times greater than the mean can be classified as influential.

```
mod <- lm(price ~ ., data=usedcars)
cooksd <- cooks.distance(mod)
# plot cook's distance
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")

abline(h = 4*mean(cooksd, na.rm=T), col="red")  # add cutoff line
```




```
  #text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>20*mean(cooksd, na.rm=T),
     #names(cooksd),""), col="red")  # add labels
  influential <- as.numeric(names(cooksd)[(cooksd > 4*mean(cooksd, na.rm=T))])
  # influential row numbers
  head(usedcars[influential, ],3)
```

```
##     price   trim isOneOwner mileage year  color displacement     fuel
## 167 72900 63 AMG          f   38512 2010  White          6.3 Gasoline
## 199 39555    350          f   56595 2012 Silver            3   Diesel
## 200 37995    350          t   66689 2012  Black            3   Diesel
##     region   soundSystem wheelType
## 167    Pac          unsp      unsp
## 199    ENC Harman Kardon     Alloy
## 200    Mid Harman Kardon     Alloy
```

```
  usedcars_1=usedcars[-c(influential),]
  rows_deleted=nrow(usedcars)-nrow(usedcars_1)
  summary(usedcars_1)
```

```
##      price            trim       isOneOwner    mileage
##  Min.   :  699   550    :11708   f:16248    Min.   :    15
##  1st Qu.:13476   430    : 2774   t: 3380    1st Qu.: 40389
##  Median :28995   500    : 2634              Median : 67711
##  Mean   :30391   600    :  524              Mean   : 73127
##  3rd Qu.:43501   63 AMG :  522              3rd Qu.: 98411
##  Max.   :79999   55 AMG :  353              Max.   :315340
##                  (Other): 1113
##       year         color       displacement        fuel
```

```
##  2007   :3569   Black :8007   5.5    :9418   Diesel  :  173
##  2008   :2313   Blue  : 889   4.3    :2774   Gasoline:19248
##  2012   :1802   Gray  :2128   4.6    :2745   Hybrid  :  207
##  2010   :1577   other : 940   5      :2634
##  2011   :1567   Silver:4291   6.3    : 455
##  2013   :1138   unsp  : 800   5.4    : 353
##  (Other):7662   White :2573   (Other):1249
##      region            soundSystem     wheelType
##  SoA    :6371   Bang Olufsen :  68   Alloy  :10885
##  Pac    :3181   Bose         :1245   other  :  112
##  Mid    :2608   Harman Kardon:4205   Premium:  413
##  WSC    :2440   Premium      :5198   unsp   : 8218
##  ENC    :1948   unsp         :8912
##  Mtn    : 986
##  (Other):2094
```

Analysis of outliers detected

Running the module on the "used cars" data we observed 435 outliers. Analyzing the top 3 outliers we observed that when we compared the prices for a used car for the years 2010 and 2012, it came out to be around $40,000-$50,000 and $60,000-$70,000 respectively, which differ significantly from the prices mentioned for these entries. Hence, these are considered as outliers.

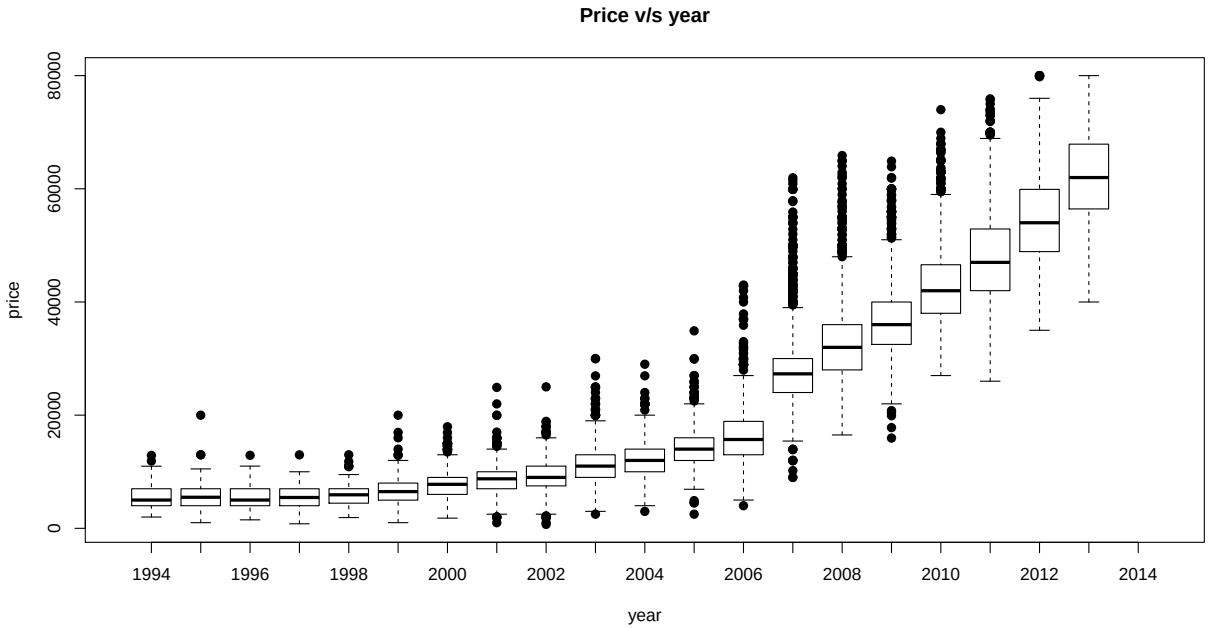
Treatment of outliers

As we are predicting the price of a used cars for any given input of the significant variables and we have significantly smaller number of outliers when compared to the original data thus we decided to eliminate them.

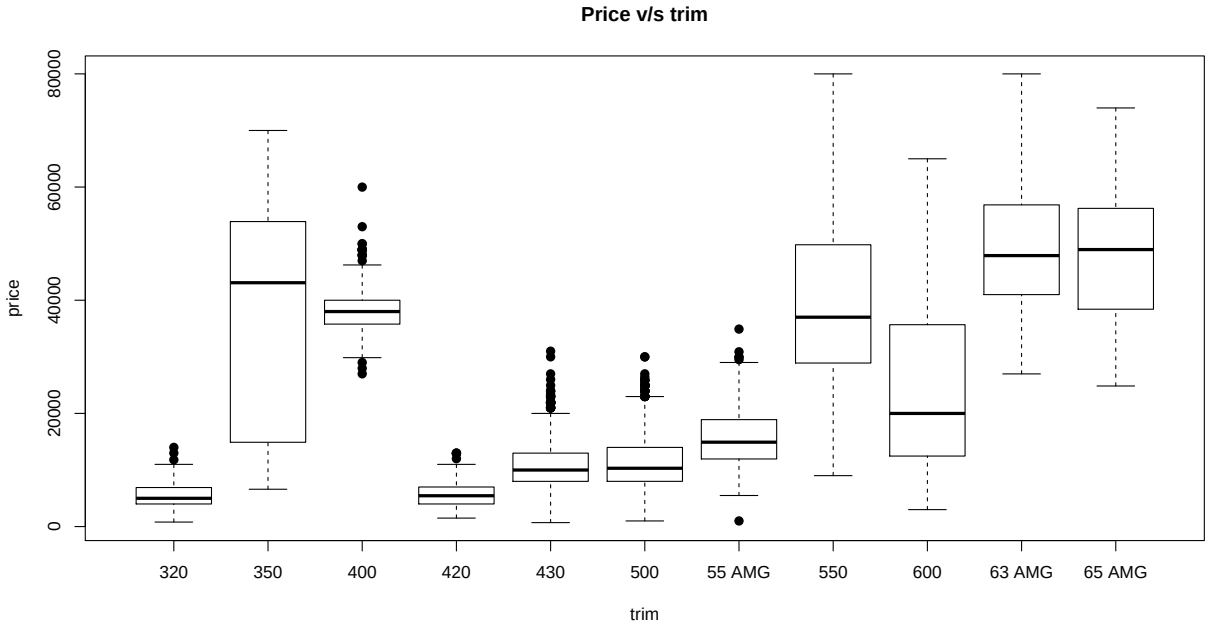
Creating boxplots of the data

Most of the independent variables given are categorical. Earlier in the code, year and displacement were also changed to categorical data. To get an idea of the significant variables we also plotted each independent variable against the price of the used cars.

```r
plot(usedcars_1$year,usedcars_1$price,xlab="year", ylab="price",pch=16,cex=1.2,
     main = "Price v/s year")
```

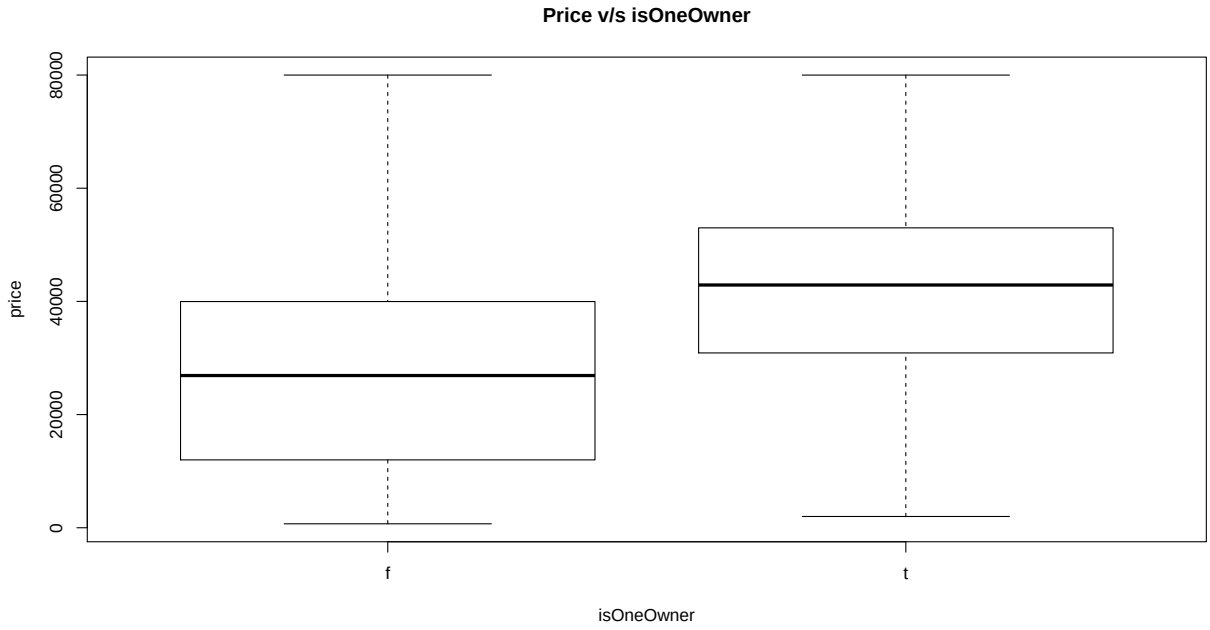



```r
# A clear trend is visible depicting that year is directly related with price
# i.e. newer cars are generally costly
plot(usedcars_1$trim,usedcars_1$price,xlab="trim", ylab="price",pch=16,cex=1.2,
     main ="Price v/s trim")
```

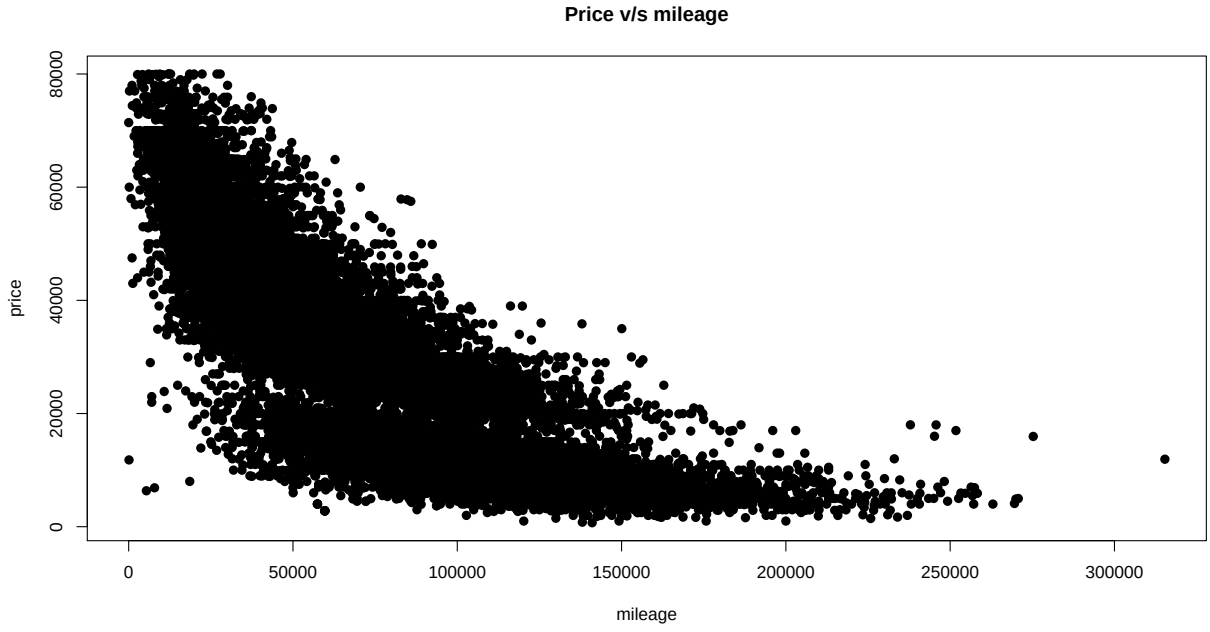



```r
# A clear trend is not visible from the plot hence direct relationship can't be
# established
```

```
plot(usedcars_1$isOneOwner,usedcars_1$price,xlab="isOneOwner",ylab="price",
     pch=16,cex=1.2,main ="Price v/s isOneOwner")
```

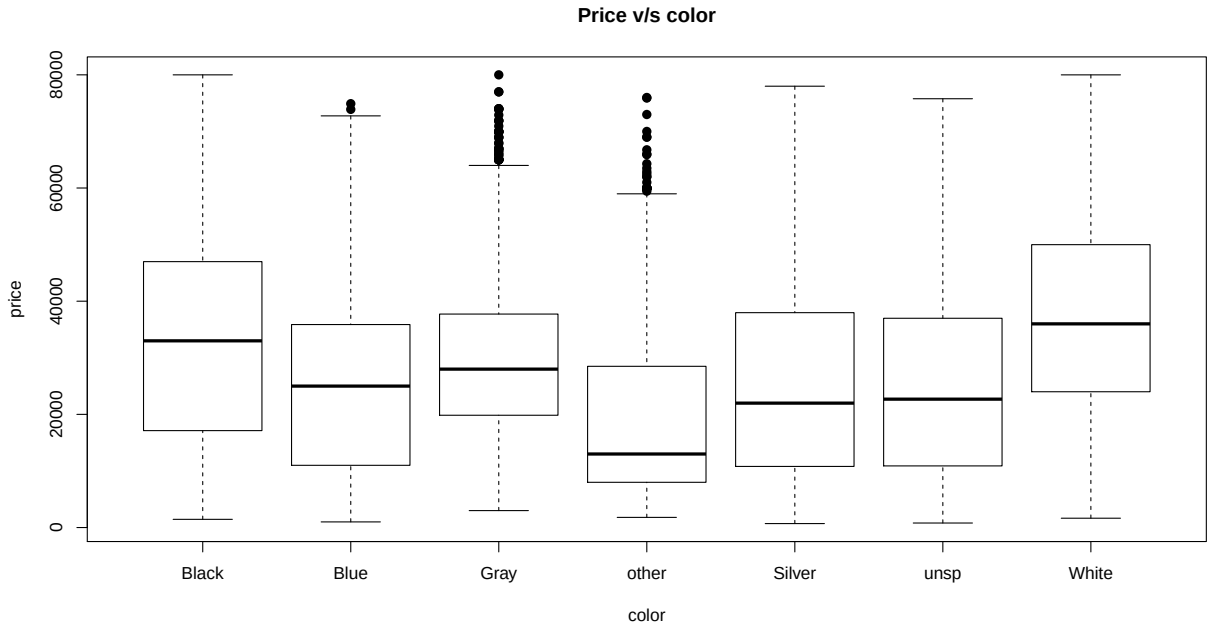



```
# Clearly, used cars with single owners tend to be more expensive than the ones
# with multiple users
plot(usedcars_1$mileage,usedcars_1$price,xlab="mileage", ylab="price",pch=16,
     cex=1.2,main ="Price v/s mileage")
```

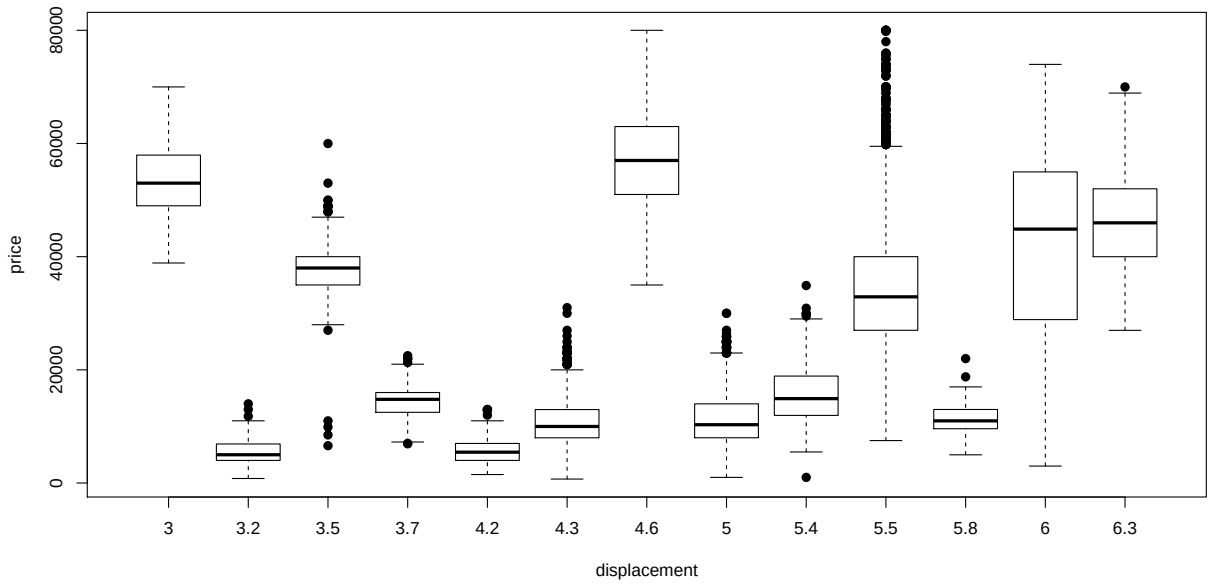



```
# Used cars with lesser mileage are generally more expensive than their
# counterparts
```

```
plot(usedcars_1$color,usedcars_1$price,xlab="color",ylab="price",pch=16,
     cex=1.2,main ="Price v/s color")
```

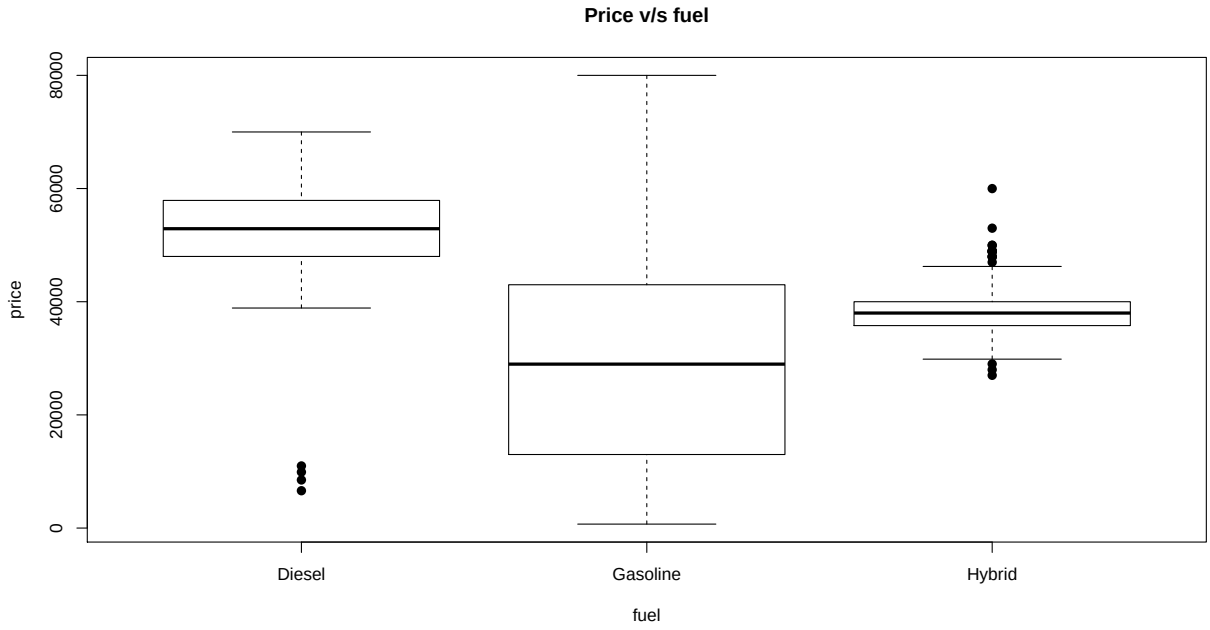



```
# Colors don't tend to impact the prices of cars by much, however black/white
# cars are slightly more on the expensive side when compared to others
plot(usedcars_1$displacement,usedcars_1$price,xlab="displacement",ylab="price",
     pch=16,cex=1.2,title("Price v/s displacement"))
```

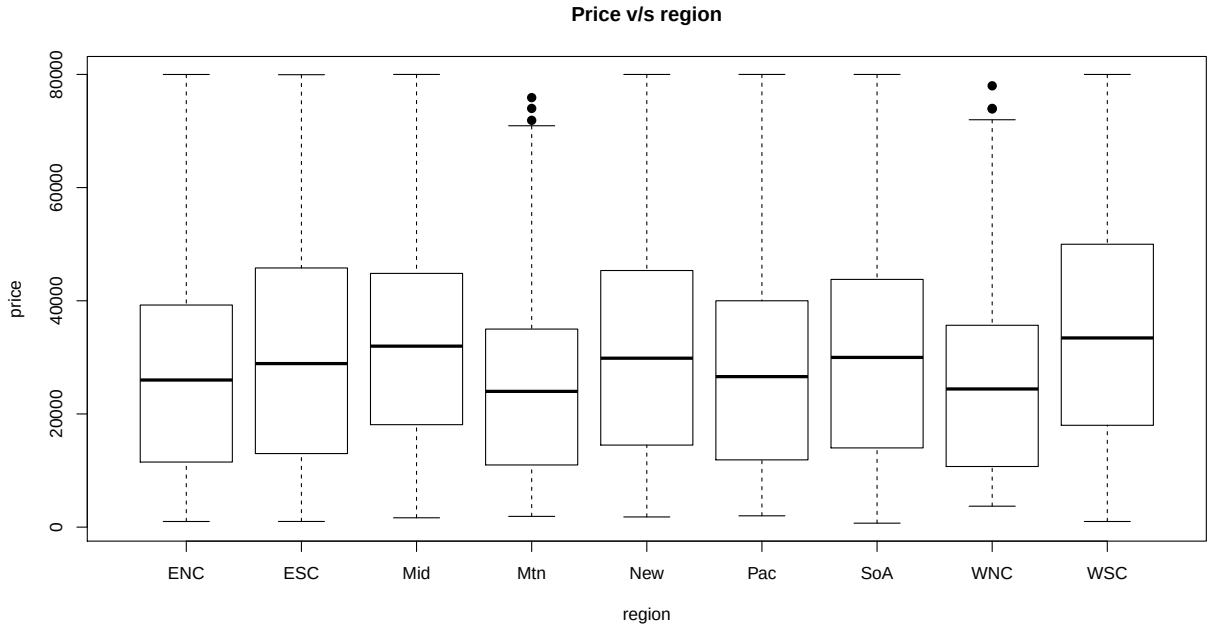



```
# Cars with displacement engines in certain group tend to be more expensive
# than others. Since, displacement is considered as categorical variable
```

```
plot(usedcars_1$fuel,usedcars_1$price,xlab="fuel", ylab="price",pch=16,cex=1.2,
     main ="Price v/s fuel")
```

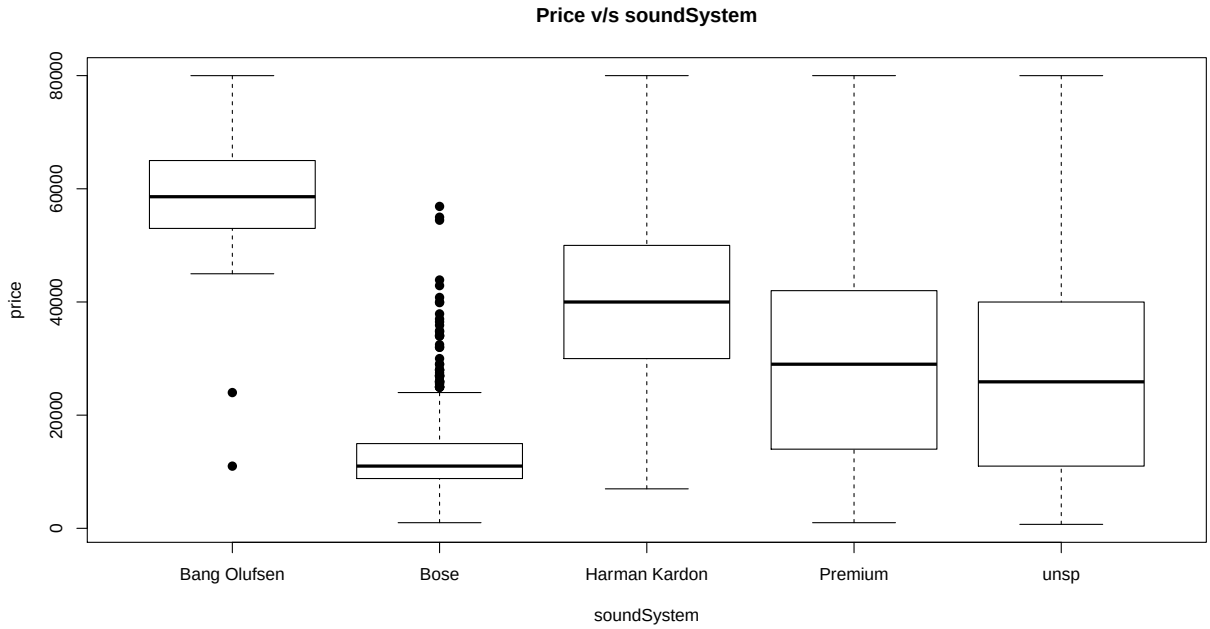



```
# Cars with diesel models are fairly expensive than gasoline and hybrid models
  plot(usedcars_1$region,usedcars_1$price,xlab="region", ylab="price", pch=16,
     cex=1.2,main ="Price v/s region")
```

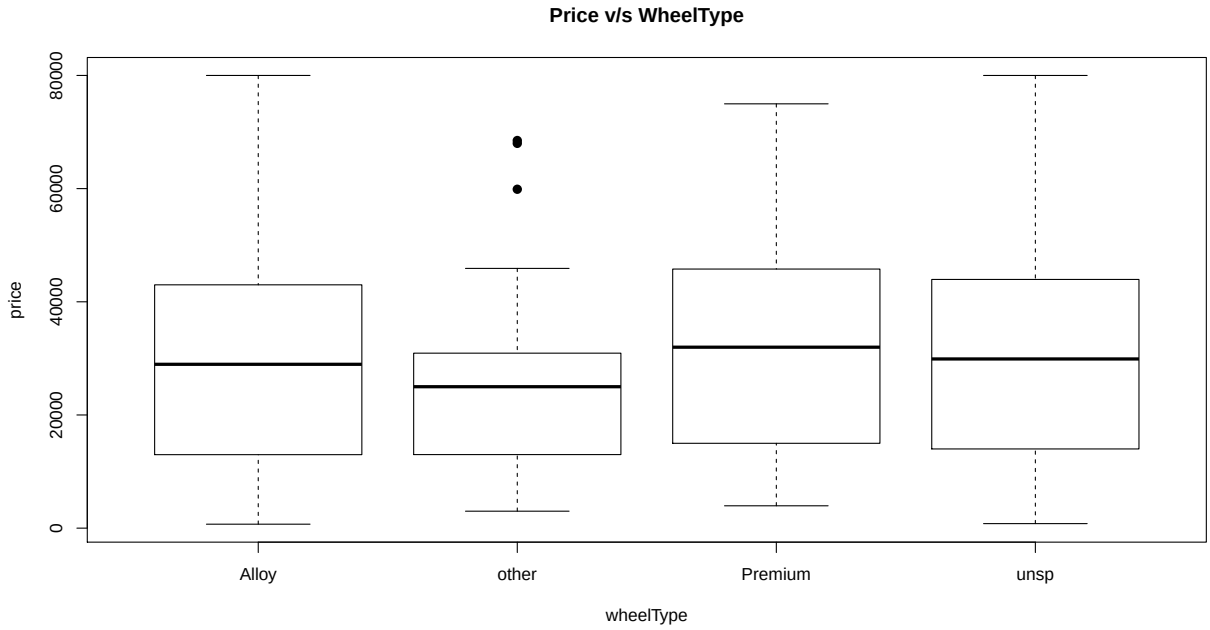



```
# Region doesn't seem to have any impact on the prices of the used cars
```

```r
plot(usedcars_1$soundSystem,usedcars_1$price,xlab="soundSystem",ylab="price",
     pch=16,cex=1.2,main ="Price v/s soundSystem")
```




```r
# A clear relationship can be established between the model of sound system and
# the price i.e. cars with certain price range tend to have a particular choice
# of sound systems
plot(usedcars_1$wheelType,usedcars_1$price,xlab="wheelType",ylab="price",
     pch=16,cex=1.2, main ="Price v/s WheelType")
```



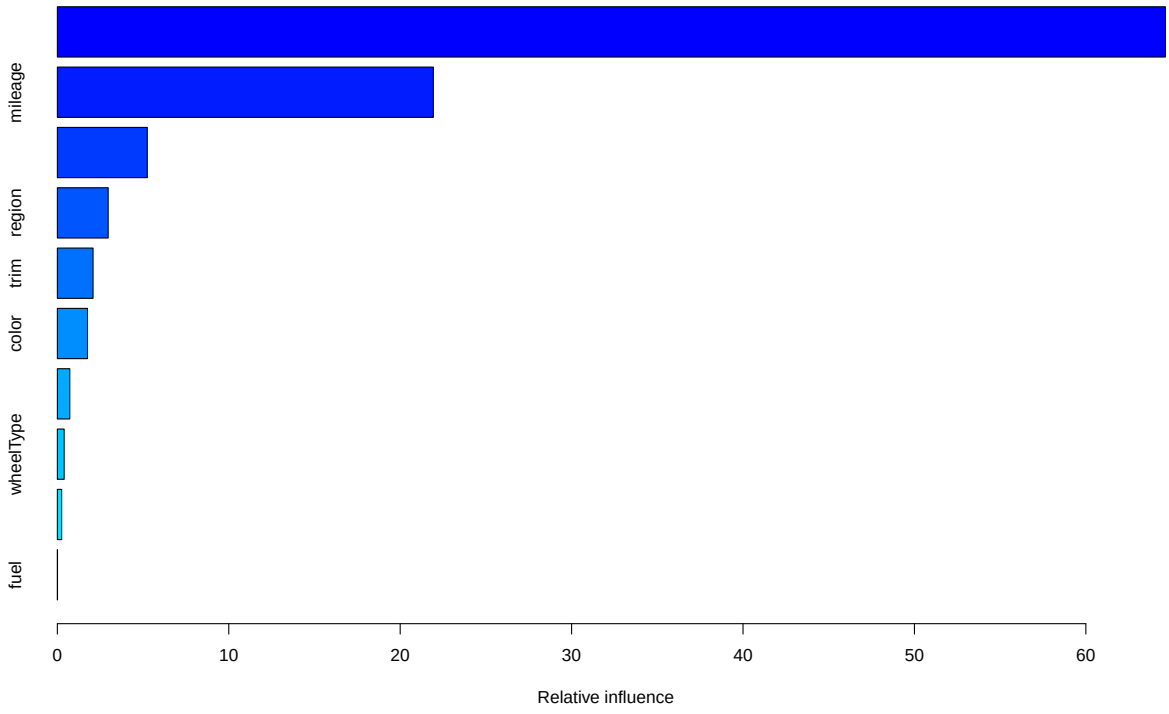
```
  # Wheel types don't tend to impact significantly with the predicted price of
  # the used cars
```

From the above plots, using the natural instincts one can conclude that Year, Mileage, isOneOwner, SoundSystem, Fuel, Displacement and Color can be considered as significant variables when compared to others.

Use boosting to check significance

However, in order to confirm it the significant scores are computed using boosting and compared it with the above results.

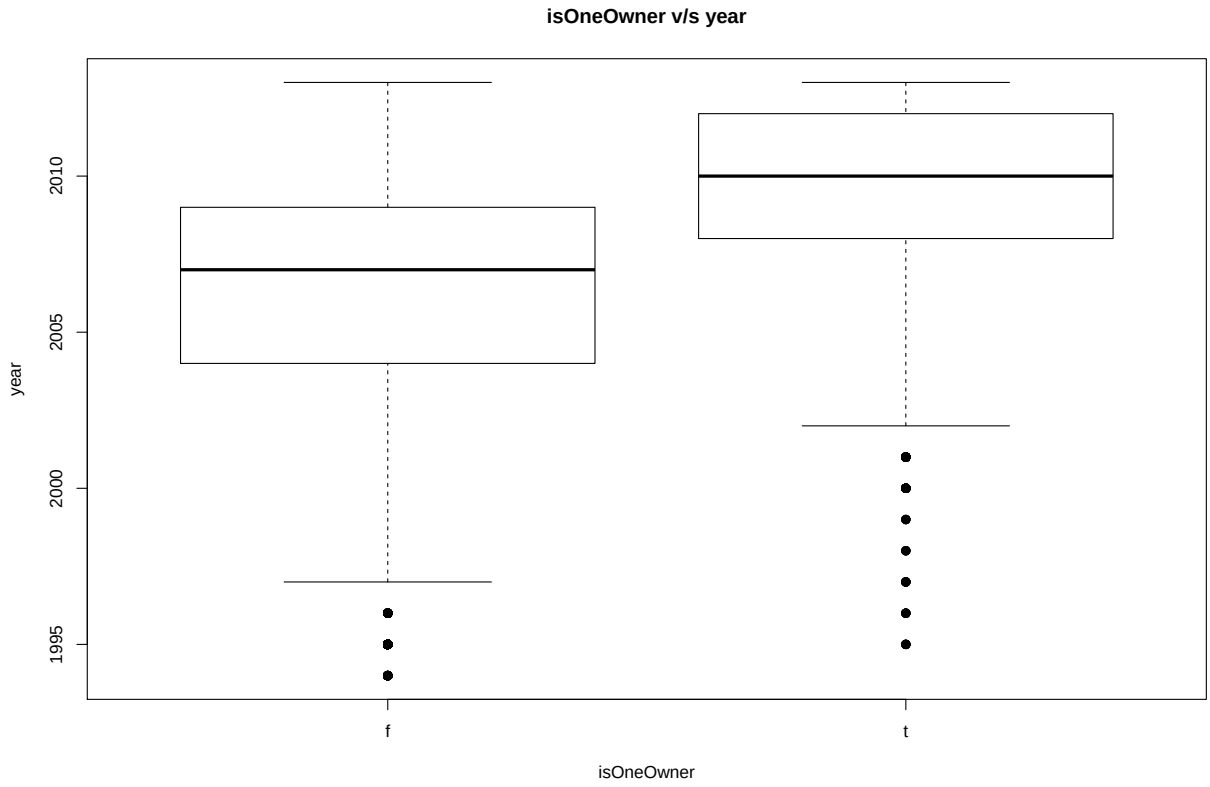
```
set.seed(14)
n=nrow(usedcars_1)
n1=floor(n/2)
n2=floor(n/4)
n3=n-n1-n2
ii = sample(1:n,n)
train=usedcars_1[ii[1:n1],]
val = usedcars_1[ii[n1+1:n2],]
test = usedcars_1[ii[n1+n2+1:n3],]

boostfit = gbm(price~.,data=train,distribution="gaussian",interaction.depth=4,
               n.trees=5000,shrinkage=.2)
boostvalpred=predict(boostfit,newdata=val,n.trees=5000)
summary(boostfit)
```



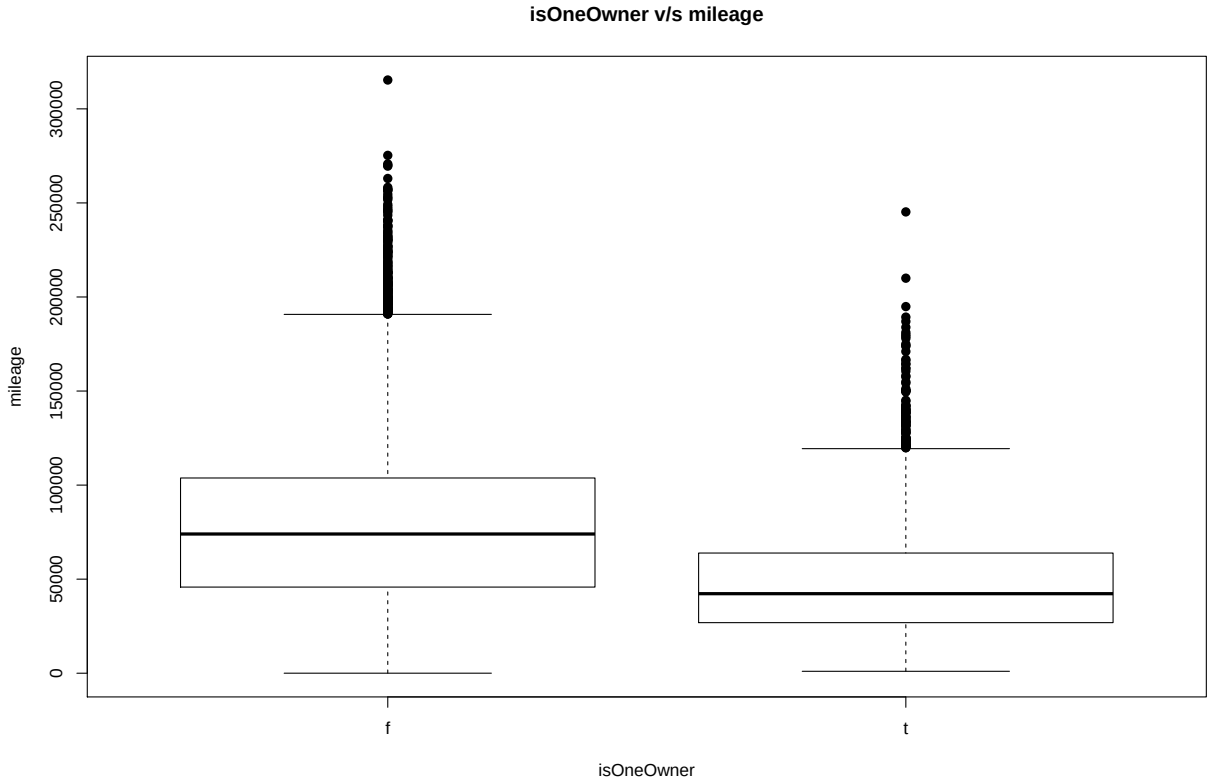
```
##                         var      rel.inf
## year                   year 64.636545388
## mileage             mileage 21.933079433
## displacement   displacement  5.245551148
## region               region  2.964134534
## trim                   trim  2.082440346
## color                 color  1.757482618
## soundSystem   soundSystem  0.725877161
## wheelType       wheelType  0.395228998
## isOneOwner     isOneOwner  0.250867098
## fuel                   fuel  0.008793277
```

Running the short boost fit with training and test data, we find year and mileage as our most important variables. From the results one can notice that most of the results match with the above. However, lower significance scores of isOneOwner and Fuel variables is certainly a surprise.Hence, we deep-dive into the analysis to confirm the significance scores of these variables.

```
plot(usedcars_1$isOneOwner,as.numeric(as.character(usedcars_1$year)),
       xlab="isOneOwner", ylab="year",pch=16,cex=1.2,
     main ="isOneOwner v/s year")
```




```
# Clearly most cars with single owners are new
  plot(usedcars_1$isOneOwner,usedcars_1$mileage,xlab="isOneOwner",ylab="mileage",
       pch=16, cex=1.2,main ="isOneOwner v/s mileage")
```




```
# Also, most cars with single owners have lesser mileage on them
```

Thus further analysis revealed the possible reasons for their lower significance scores: **For isOneOwner:** 1. Uneven distribution of the data for this variable with over 80% of cars had 'isOneOwner' flag set to be false hence reducing its influence in predicting the price. 2. Most of the used cars with single owners were relatively new and are with lower mileage making it highly correlated with them. And since both 'Year' and 'Mileage' are relatively more significant than 'isOneOwner' thus resulting in lower significance score. Similarly, for the variable **FuelType** we observed for most of the used cars (98%) fuel type is gasoline. Thus other fueltypes do not contribute in predicting the price. Hence reducing the significance score for the variable.

Add interactions

Next we want to add interactions into the data, most of our data points are categorical so we generated interactions manually using interaction(). Also, because mileage is considered significant and it is difficult to interact it with the other categorical variables and resulting in a different significant variable, is slightly misleading hence it was excluded from the interactions. Since there was no proper means to group the categories in trim we believed it could be excluded from the interactions.

```
usedcars10 = usedcars_1
# Change variables with "Unspecified" to have unsp as first factor
# The first factor is removed when anayzing and makes more sense to remove unsp from model
usedcars_1$wheelType <- relevel(usedcars_1$wheelType, ref = "unsp")
usedcars_1$soundSystem <- relevel(usedcars_1$soundSystem, ref = "unsp")
usedcars_1$color <- relevel(usedcars_1$color, ref = "unsp")

# Add Interactions
usedcars_1$isOneOwner.year = interaction(usedcars_1$isOneOwner,usedcars_1$year)
```

```
usedcars_1$isOneOwner.color = interaction(usedcars_1$isOneOwner,usedcars_1$color)
usedcars_1$isOneOwner.displacement = interaction(usedcars_1$isOneOwner,
                                                 usedcars_1$displacement)
usedcars_1$isOneOwner.region = interaction(usedcars_1$isOneOwner,usedcars_1$region)
usedcars_1$isOneOwner.soundSystem = interaction(usedcars_1$isOneOwner,
                                                usedcars_1$soundSystem)
usedcars_1$isOneOwner.wheelType = interaction(usedcars_1$isOneOwner,usedcars_1$wheelType)
usedcars_1$year.color = interaction(usedcars_1$year,usedcars_1$color)
usedcars_1$year.displacement = interaction(usedcars_1$year,usedcars_1$displacement)
usedcars_1$year.fuel = interaction(usedcars_1$year,usedcars_1$fuel)
usedcars_1$year.region = interaction(usedcars_1$year,usedcars_1$region)
usedcars_1$year.soundSystem = interaction(usedcars_1$year,usedcars_1$soundSystem)
usedcars_1$year.wheelType = interaction(usedcars_1$year,usedcars_1$wheelType)
usedcars_1$color.displacement = interaction(usedcars_1$color,usedcars_1$displacement)
usedcars_1$color.fuel = interaction(usedcars_1$color,usedcars_1$fuel)
usedcars_1$color.region = interaction(usedcars_1$color,usedcars_1$region)
usedcars_1$color.soundSystem = interaction(usedcars_1$color,usedcars_1$soundSystem)
usedcars_1$color.wheelType = interaction(usedcars_1$color,usedcars_1$wheelType)
usedcars_1$displacement.fuel = interaction(usedcars_1$displacement,usedcars_1$fuel)
usedcars_1$displacement.region = interaction(usedcars_1$displacement,usedcars_1$region)
usedcars_1$displacement.soundSystem = interaction(usedcars_1$displacement,
                                                  usedcars_1$soundSystem)
usedcars_1$displacement.wheelType = interaction(usedcars_1$displacement,
                                                usedcars_1$wheelType)
usedcars_1$fuel.region = interaction(usedcars_1$fuel,usedcars_1$region)
usedcars_1$fuel.soundSystem = interaction(usedcars_1$fuel,usedcars_1$soundSystem)
usedcars_1$fuel.wheelType = interaction(usedcars_1$fuel,usedcars_1$wheelType)
usedcars_1$region.soundSystem = interaction(usedcars_1$region,usedcars_1$soundSystem)
usedcars_1$region.wheelType = interaction(usedcars_1$region,usedcars_1$wheelType)
usedcars_1$wheelType.soundSystem = interaction(usedcars_1$wheelType,usedcars_1$soundSystem)
usedcars_1$mileage.2 = (usedcars_1$mileage)^2

x = model.matrix(usedcars10$price~.,usedcars10)[,-1]
x = scale(x)
x<-x[, colSums(is.na(x)) != nrow(x)]
y = usedcars_1$price

x1 = model.matrix(usedcars_1$price~.,usedcars_1)[,-1]
x1 = scale(x1)
x1<-x1[, colSums(is.na(x1)) != nrow(x1)]
```

Forward Selection

When adding interactions and scaling the data, all of the dummy variables for the categorical variables and interactions were added. Any combinations that did not occur and created NA columns were removed. The resulting matrix had 1347 variables. A linear regression model was created using all of the interaction variables and the forward selection model was implemented. Given the size of the data matrix, the forward selection method took between 1 and 2 hours to generate. Due to processing power and time constraints it was determined that using cross validation to evaluate out of sample fits would not be possible.

```
source("do-stepcv.R")
attach(usedcars_1)
data1 = data.frame(price,x1)
```

```
 data2 = data.frame(price,x)

 # Use Forward Step to determine significant variables
 n1 = nrow(data1)
lmf = lm(data1$price~.,data1)
 bictran = extractAIC(lmf,k=log(n1))

#print(summary(lmf)) -->

 nullmod = lm(data1$price~1,data1)
 fullmod = lmf
 fwd = stepAIC(nullmod,scope=formula(lmf),direction="forward",k=log(n1),
trace=0)
 bicfwd = extractAIC(fwd,k=log(n1))
 cat("bic for forward model is: ",bicfwd,"\n")
```

```
## bic for forward model is:  71 324683.2
```

```
 print(summary(fwd)$sigma)
```

```
## [1] 3846.25
```

```
 keeps <- c(variable.names(fwd))[-1]
 x2  <- subset(data1, select = keeps)
 x2 = as.matrix(x2)
```

Lasso, Ridge, Elastic Net

Besides forward selection, we can improve the performance of a linear model by adding and shrinking parameters. In this section the Lasso, Ridge, and Elastic Net methods were tested.

```
## Fit methods on grid of lambda values.
set.seed(14)
gsize=100
source("do-stepcv.R")
n=length(y)
nfold=10
fid1=getfolds(nfold,n, dorand = TRUE)
#grid=10^seq(10,-15,l ength=gsize)
grid=5^seq(10,-2,length=gsize)
#print(grid)
# The values were already scaled so they are considered standardized, but the y was not set
# to mean of 0, so intercept cannot be set to FALSE

# First choose lasso
cvdgnL = cv.glmnet(x1,y,nfolds=10,foldid=fid1,family="gaussian",standardize=FALSE,alpha=1,
                   lambda=grid)
cvdgnR = cv.glmnet(x1,y,nfolds=10,foldid=fid1,family="gaussian",standardize=FALSE,alpha=0,
                   lambda=grid)
cvdgnE = cv.glmnet(x1,y,nfolds=10,foldid=fid1,family="gaussian",standardize=FALSE,
                   alpha=0.5,lambda=grid)

bestlamL = cvdgnL$lambda.min
print(paste('best lambda for lasso is: ',round(bestlamL,2)),col='red',cex=1.5)
```

```
## [1] "best lambda for lasso is:  16.92"
```

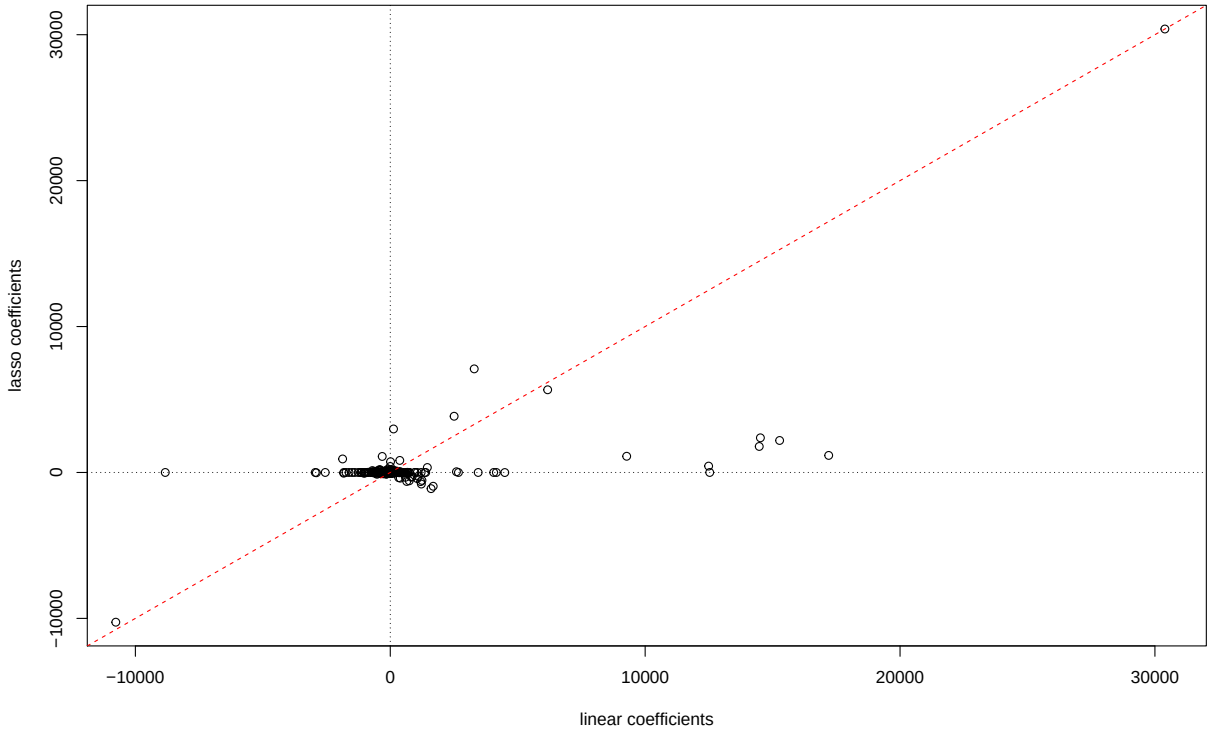
```
bestlamR = cvdgnR$lambda.min
print(paste('best lambda for ridge is: ',round(bestlamR,2)),col='red',cex=1.5)
```

```
## [1] "best lambda for ridge is:  4.32"
```

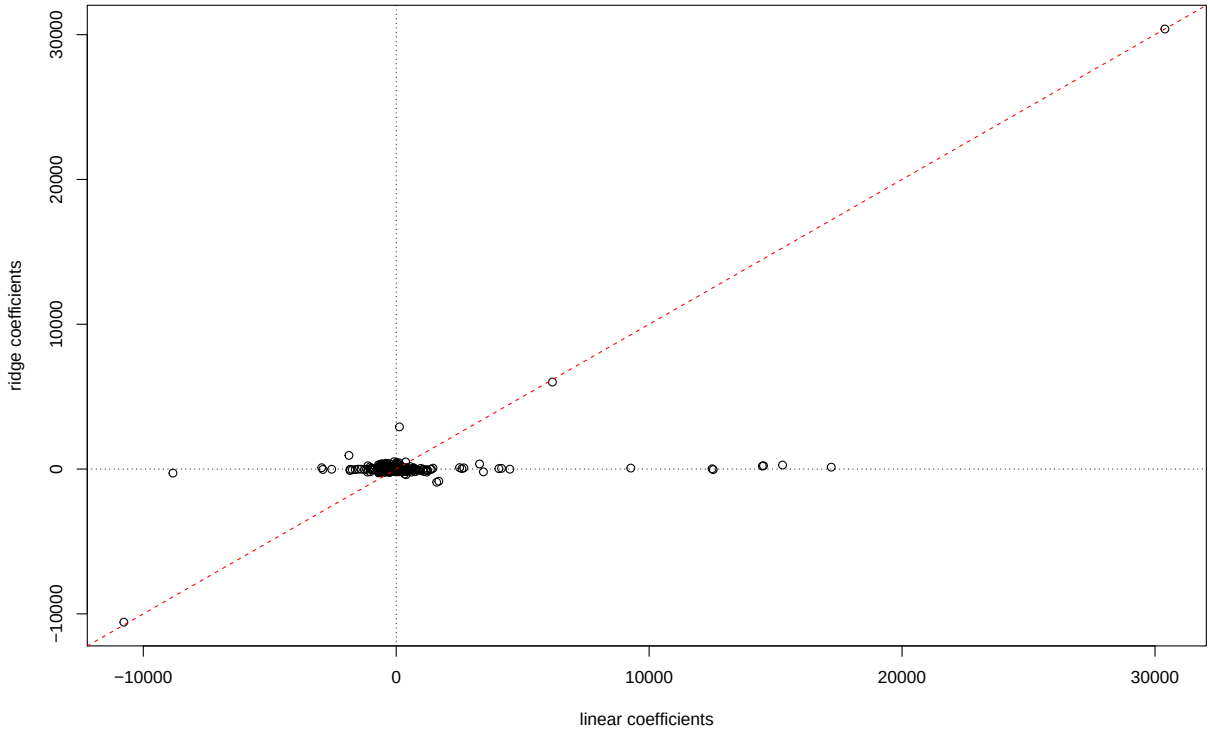
```
bestlamE = cvdgnE$lambda.min
print(paste('best lambda for elastic net is: ',round(bestlamE,10)),col='red',cex=1.5)
```

```
## [1] "best lambda for elastic net is:  30.3853081847"
```

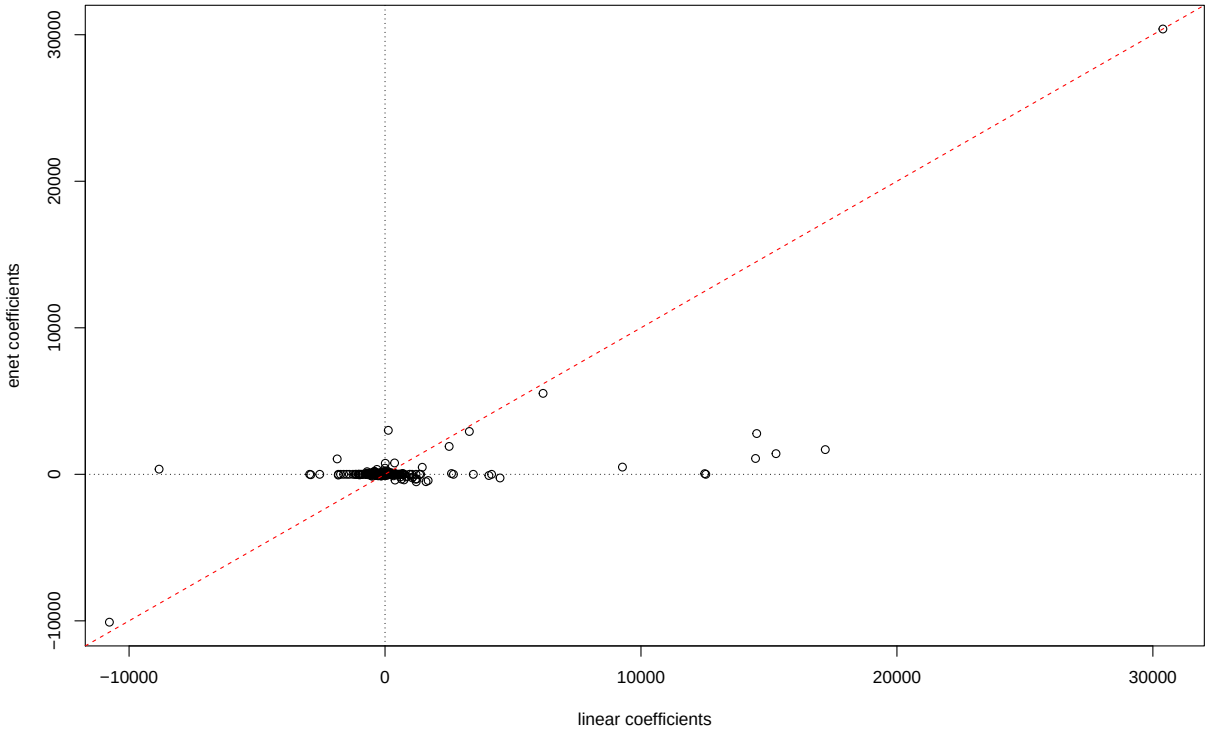
```
##get coefficients for best lambda
bestlassocoef = predict(cvdgnL,s=bestlamL,type='coefficients',exact=TRUE)[,1]
ddf = data.frame(x1,y)
lm.mod = lm(y~.,ddf)
rgxy = range(c(lm.mod$coeficients,bestlassocoef))
plot(lm.mod$coef,bestlassocoef,xlim=rgxy,ylim=rgxy,xlab='linear coefficients',
     ylab='lasso coefficients')
abline(0,1,col='red',lty=2)
abline(v=0,lty=3)
abline(h=0,lty=3)
```



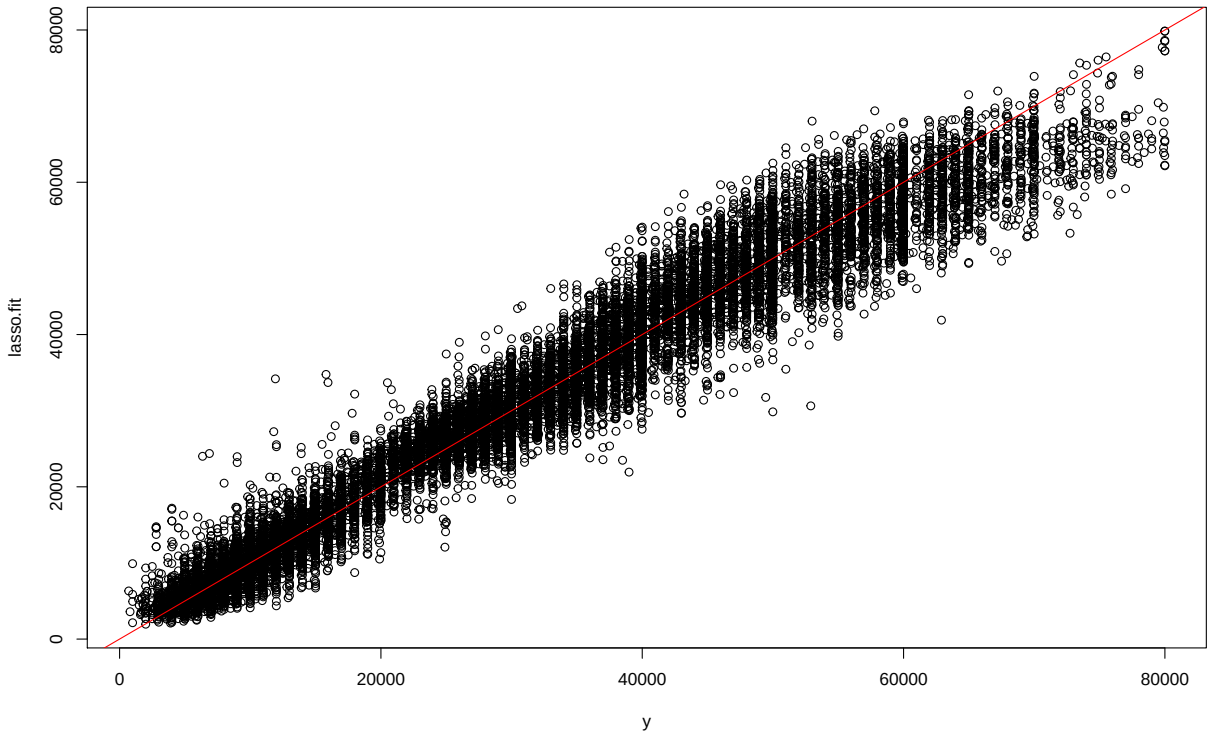

```
bestridgecoef = predict(cvdgnR,s=bestlamR,type='coefficients',exact=TRUE)[,1]
rgxy = range(c(lm.mod$coeficients,bestridgecoef))
plot(lm.mod$coef,bestridgecoef,xlim=rgxy,ylim=rgxy,xlab='linear coefficients',
     ylab='ridge coefficients')
abline(0,1,col='red',lty=2)
abline(v=0,lty=3)
abline(h=0,lty=3)
```




```
bestenetcoef = predict(cvdgnE,s=bestlamE,type='coefficients',exact=TRUE)[,1]
rgxy = range(c(lm.mod$coeficients,bestenetcoef))
plot(lm.mod$coef,bestenetcoef,xlim=rgxy,ylim=rgxy,xlab='linear coefficients',
     ylab='enet coefficients')
abline(0,1,col='red',lty=2)
abline(v=0,lty=3)
abline(h=0,lty=3)
```



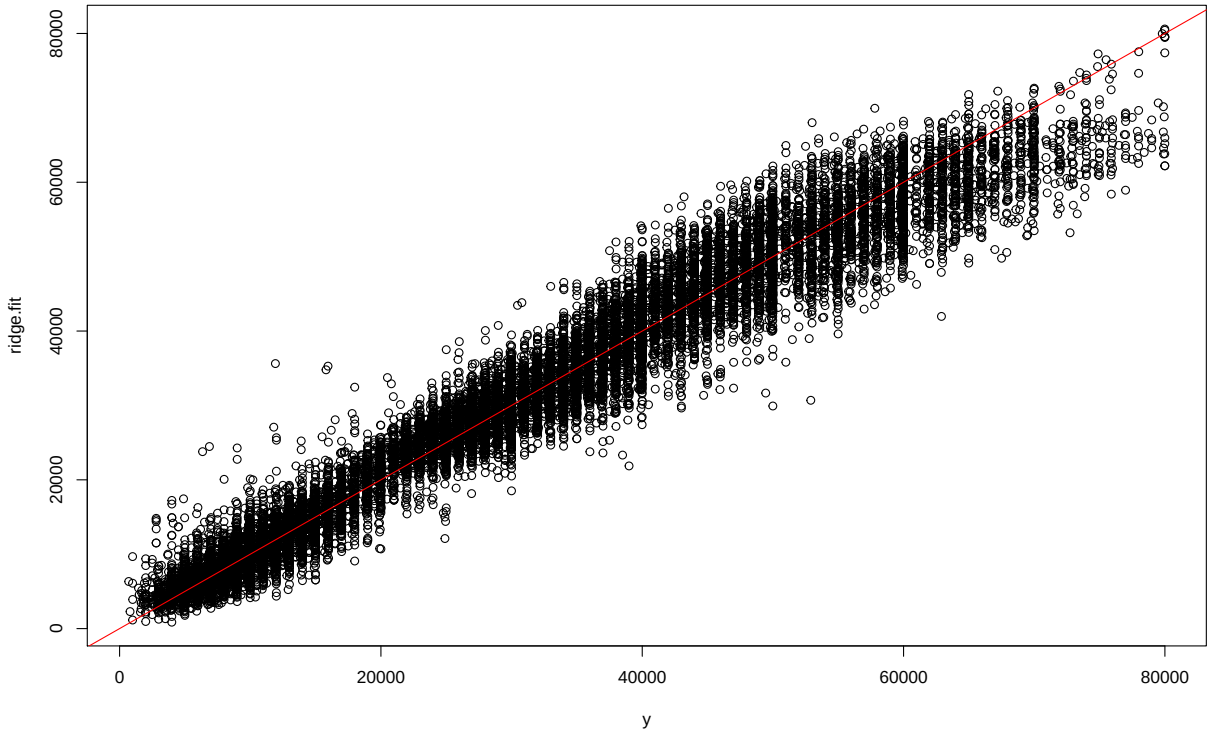

```
##get fits
lasso.fit = predict(cvdgnL,s=bestlamL,newx=x1)
lm.fit = lm.mod$fitted
fmat = cbind(y,lm.fit,lasso.fit)
colnames(fmat) = c('y','linear','lasso')
plot(y,lasso.fit)
abline(0,1,col='red')
```




```
rmse <- sqrt(mean((y - lasso.fit)^2))
print(rmse)
```

```
## [1] 3784.938
```

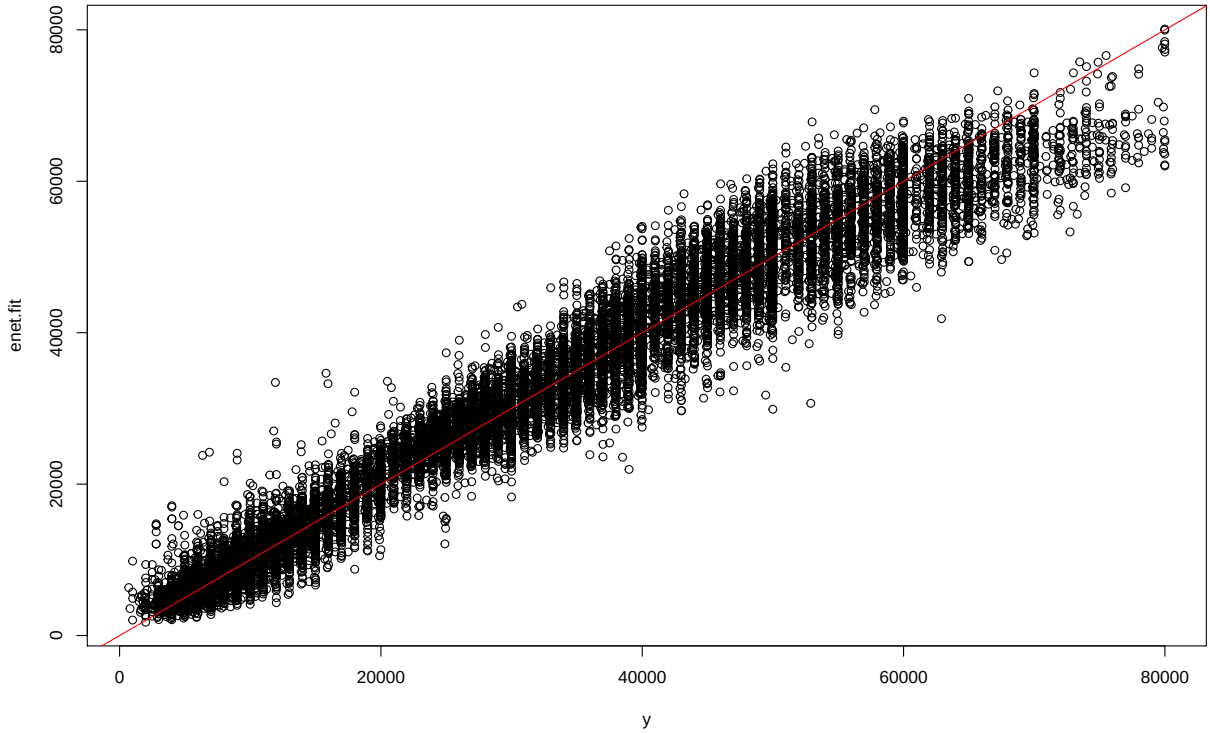
```
ridge.fit = predict(cvdgnR,s=bestlamR,newx=x1)
lm.fit = lm.mod$fitted
fmat = cbind(y,lm.fit,lasso.fit,ridge.fit)
colnames(fmat) = c('y','linear','lasso','ridge')
plot(y,ridge.fit)
abline(0,1,col='red')
```




```
rmse1 <- sqrt(mean((y - ridge.fit)^2))
print(rmse1)
```

```
## [1] 3759.036
```

```
enet.fit = predict(cvdgnE,s=bestlamE,newx=x1)
lm.fit = lm.mod$fitted
fmat = cbind(y,lm.fit,lasso.fit,ridge.fit,enet.fit)
colnames(fmat) = c('y','linear','lasso','ridge','enet')
plot(y,enet.fit)
abline(0,1,col='red')
```




```
rmse1 <- sqrt(mean((y - enet.fit)^2))
print(rmse1)
```

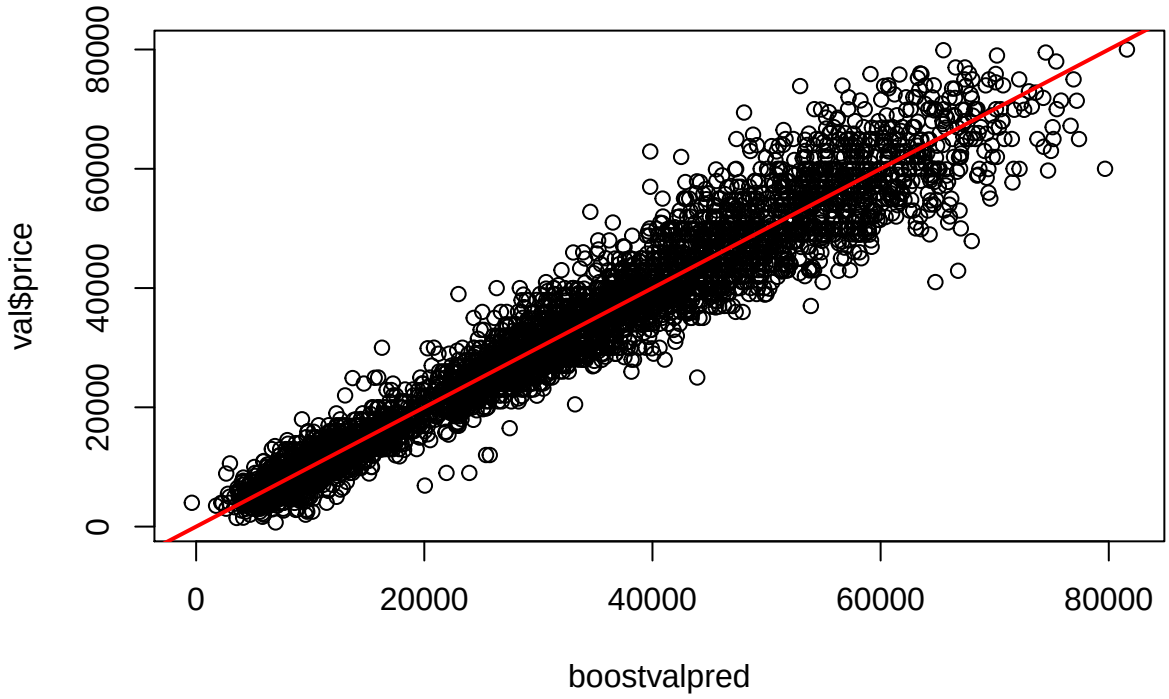
```
## [1] 3783.185
```

Random Forests and Boosting

Tree based methods are nonlinear and already can utilize interactions. While the error is not as low as the linear regression, the speed of the computations and having no need to predefine and add dummy variables or interactions gives it some advantages, and the error is still reasonable compared to regression

```
library(gbm)
library(randomForest)
set.seed(14)
n=nrow(usedcars10)
n1=floor(n/2)
n2=floor(n/4)
n3=n-n1-n2
ii = sample(1:n,n)
train=usedcars10[ii[1:n1],]
val = usedcars10[ii[n1+1:n2],]
test = usedcars10[ii[n1+n2+1:n3],]
data2 = rbind(train,val)

boostfit = gbm(train$price~.,data=train,distribution="gaussian",interaction.depth=4,
               n.trees=5000,shrinkage=.2)
boostvalpred=predict(boostfit,newdata=val,n.trees=5000)
```

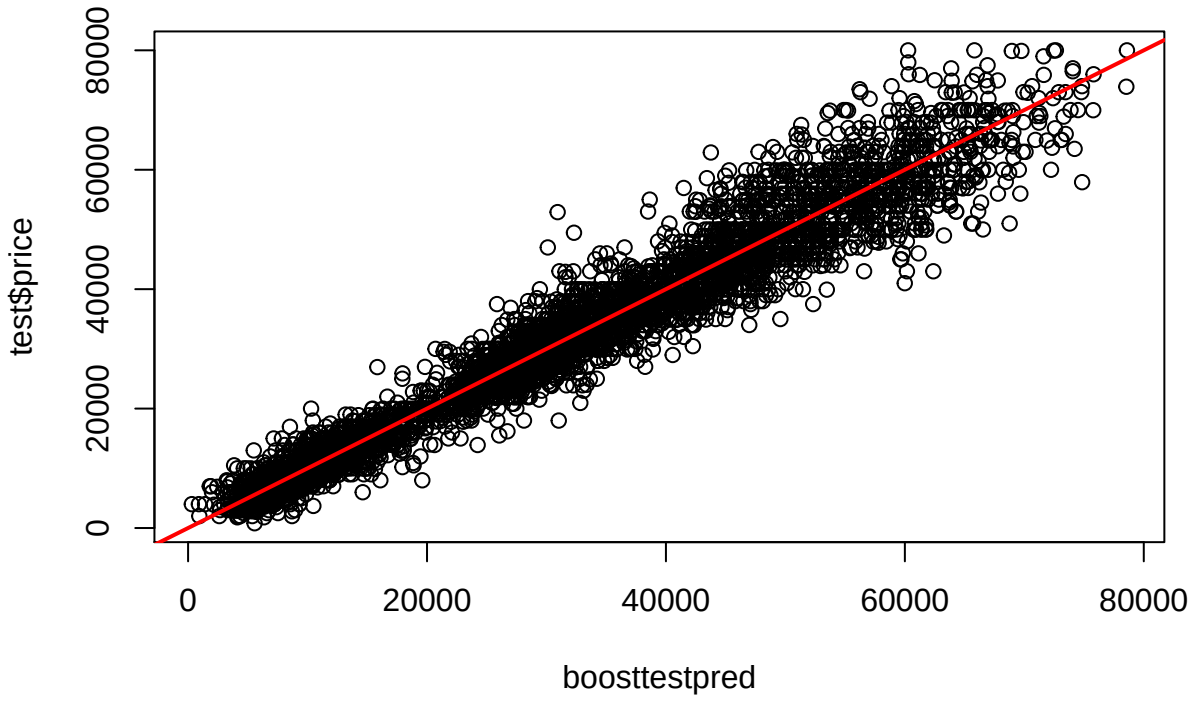
```
plot(boostvalpred,val$price)
abline(0,1,col="red",lwd=2)
```




```
rmse4 <-sqrt(mean((val$price - boostvalpred)^2))
print(rmse4)
```

```
## [1] 4224.475
```

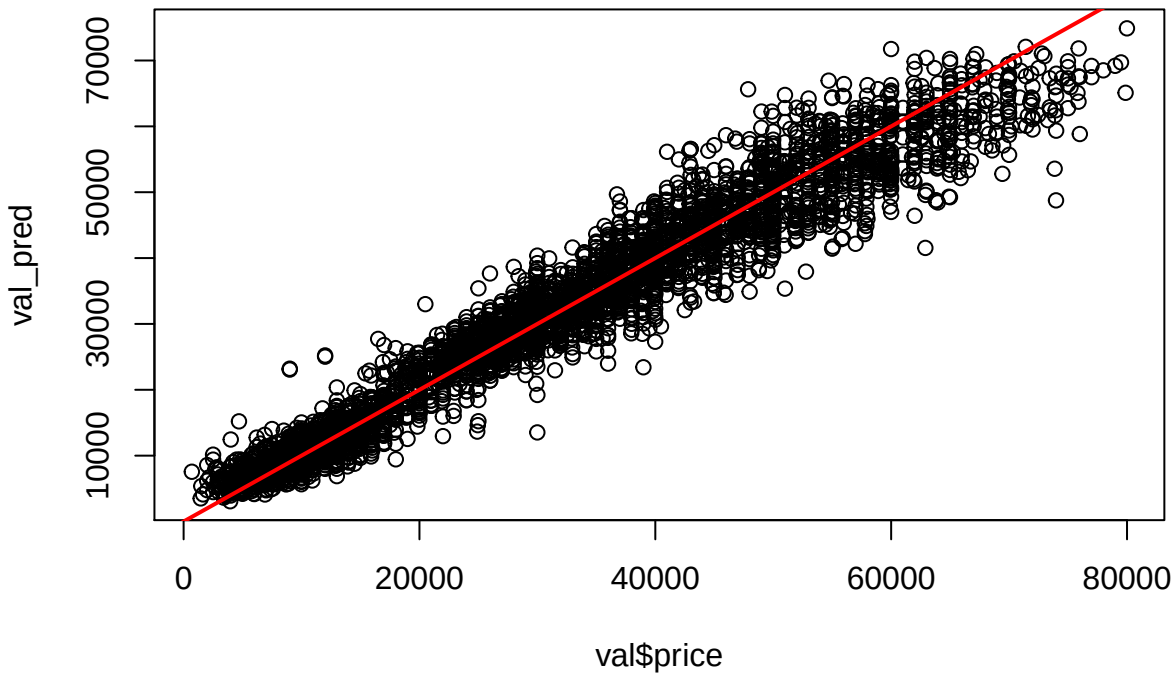
```
boostfit2 = gbm(data2$price~.,data=data2,distribution="gaussian",
                interaction.depth=4, n.trees=5000,shrinkage=.2)
boosttestpred=predict(boostfit2,newdata=test,n.trees=5000)
plot(boosttestpred,test$price)
abline(0,1,col="red",lwd=2)
```




```
rmse5 <-sqrt(mean((test$price - boosttestpred)^2))
print(rmse5)
```

```
## [1] 4030.771
```

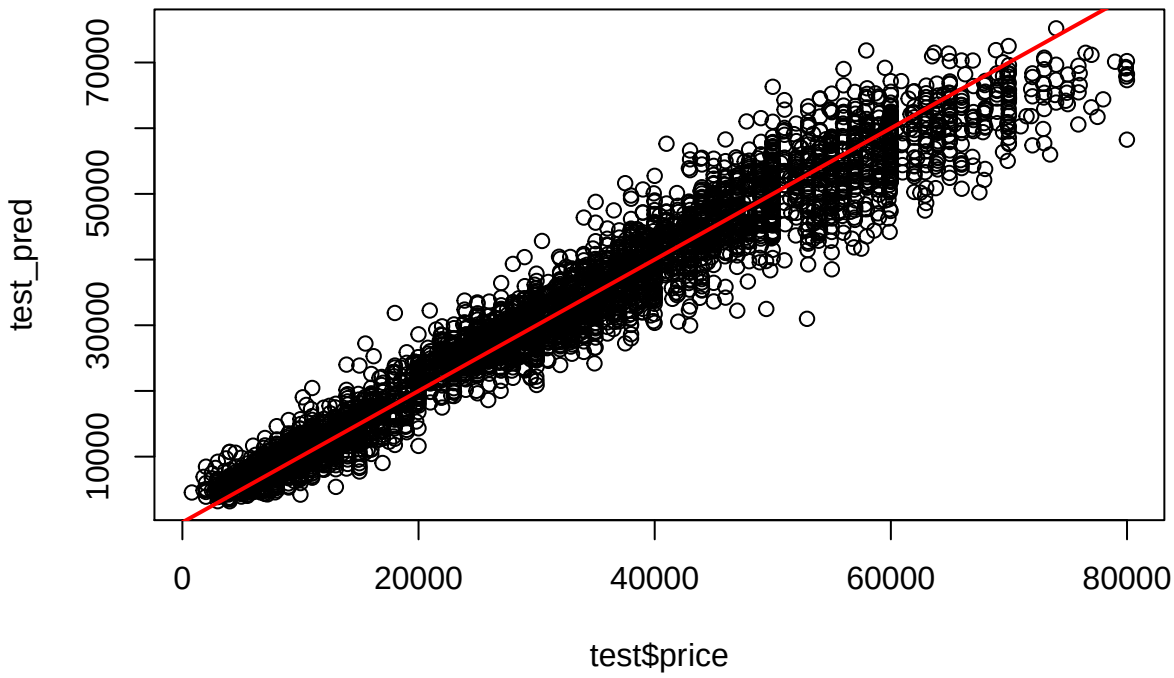
```
forest1 = randomForest(train$price~.,data=train,mtry=4,ntree=500)
val_pred = predict(forest1,newdata=val)
plot(val$price,val_pred)
abline(0,1,col="red",lwd=2)
```




```
rmse6 <- sqrt(mean((val$price-val_pred)^2))
print(rmse6)
```

```
## [1] 3907.173
```

```
forest2 = randomForest(data2$price~.,data=data2,mtry=4,ntree=500)
test_pred = predict(forest2,newdata=test)
plot(test$price,test_pred)
abline(0,1,col="red",lwd=2)
```




```
rmse7 <- sqrt(mean((test$price-test_pred)^2))
print(rmse7)
```

```
## [1] 3817.134
```

Conclusions

Final values obtained from the above methodologies are tabulated below

		Without Interactions	With Interactions
Tree-based Models	Boosting	Test Data: 4224 Validation Data: 4031	N/A
	Random Forest	Test Data: 3907 Validation Data: 3807	N/A
Generalized Linear Models	Lasso	4269.5	3784
	Ridge	4269.1	3759
	Elastic Net	4269.5	3783
	Forward stepwise Selection	4282	3846

Figure 1: RMSE values

Conclusions were drawn based on the hypothesis Comparing Linear based methods with Tree based methods:

Hypothesis 1:

Plotting numerical variable 'mileage' against 'price' we observe that the relation is not linear hence, if interactions are not included for the variables we could expect tree based methods to be significantly better than that of the linear methods.

Result

The RMSE associated with Random forest without any interactions between the variables is around 3907 and for the Linear methods including Lasso, Ridge and Elastic Net it came to be around 4300. The difference is substantial enough to conclude that the tree based method (i.e. Random Forest) performed significantly better than Lasso, Ridge or E-net, hence proving the hypothesis.

Hypothesis 2:

Introducing interactions between variables as the input for generalized linear models significantly reduces the error in prediction values generated from these models

Result

The reduction of RMSE by about 10%, between before and after the introduction of interactions for all Generalized linear models is an evident proof to conclude that interactions improve the error margin between actual and predicted values.

Hypothesis 3:

Errors induced in the prediction values when evaluated through Forward stepwise selection and Lasso (separately) are comparable

Result

RMSE value for Forward stepwise selection was obtained to be 3846 whereas for Lasso it was 3784, which more or less can be considered as same. Thus proving the hypothesis. This is also expected as both of them work on a similar principle of eliminating not so significant variables from the model. However, Also, Lasso is noticeably faster with a run time of around 5-10 min when compared to 1-2 hours for forward stepwise selection model

Hypothesis 4

Ridge comparatively performs better than Lasso in reducing the errors in the predicted values

Result

RMSE values for Ridge and Lasso methods post inducing interactions are 3759 and 3784 respectively. Although the difference is insignificant, Ridge can still be considered an effective model for this data. This can be explained as Ridge uses all of the variables as their coefficients cannot be shrunk to 0 unlike Lasso, so prediction error tends to be lower when there are a lot of variables that are highly correlated