Team_3_Group_Project

Hanyu Chen,Song Lin,Rohan Gupta,Deniz Ozakyol,Yesol Lee, Shi Wang

10/7/2021

##Install Packages##

```
#install.packages(c("rpart.plot", "rpart"))
#install.packages(c("randomForest"))
#install.packages(c("gbm"))
#install.packages('data.table')
#install.packages('ggplot2')
#install.packages('fastDummies')
library('fastDummies')
library(scales)
library(rpart)
library(rpart.plot)
library(ggplot2)
library(data.table)
library(ggthemes)
library(scales)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(fastDummies)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-2
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
theme_set(theme_bw())
```

##Loading the Dataset##

```r
diamond=fread("~/Desktop/BU/BA810/diamonds_dataset.csv")
```

##Data Cleaning##

```r
## Check if there is any missing value
any(is.na(diamond))
```

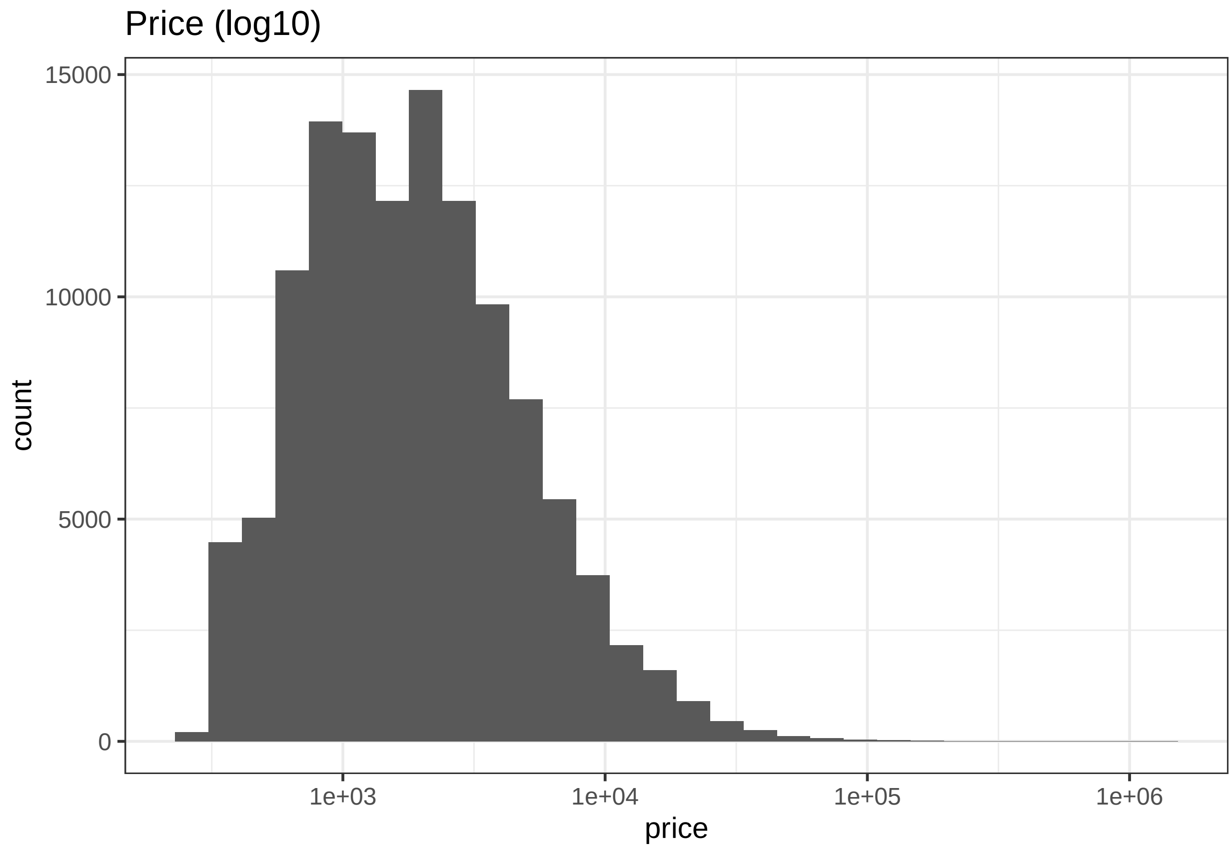
```
## [1] FALSE
```

```r
## Drop Unnecessary columns
diamond <- diamond[,url:=NULL]
diamond <- diamond[,date_fetched:=NULL]
diamond <- diamond[,id:=NULL]
## Change our predictor price to log
diamond$logprice=log(diamond$price + 1)
```

Why using log price

```r
ggplot(aes(price), data=diamond) +
  geom_histogram() +
  ggtitle('Price (log10)') +
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

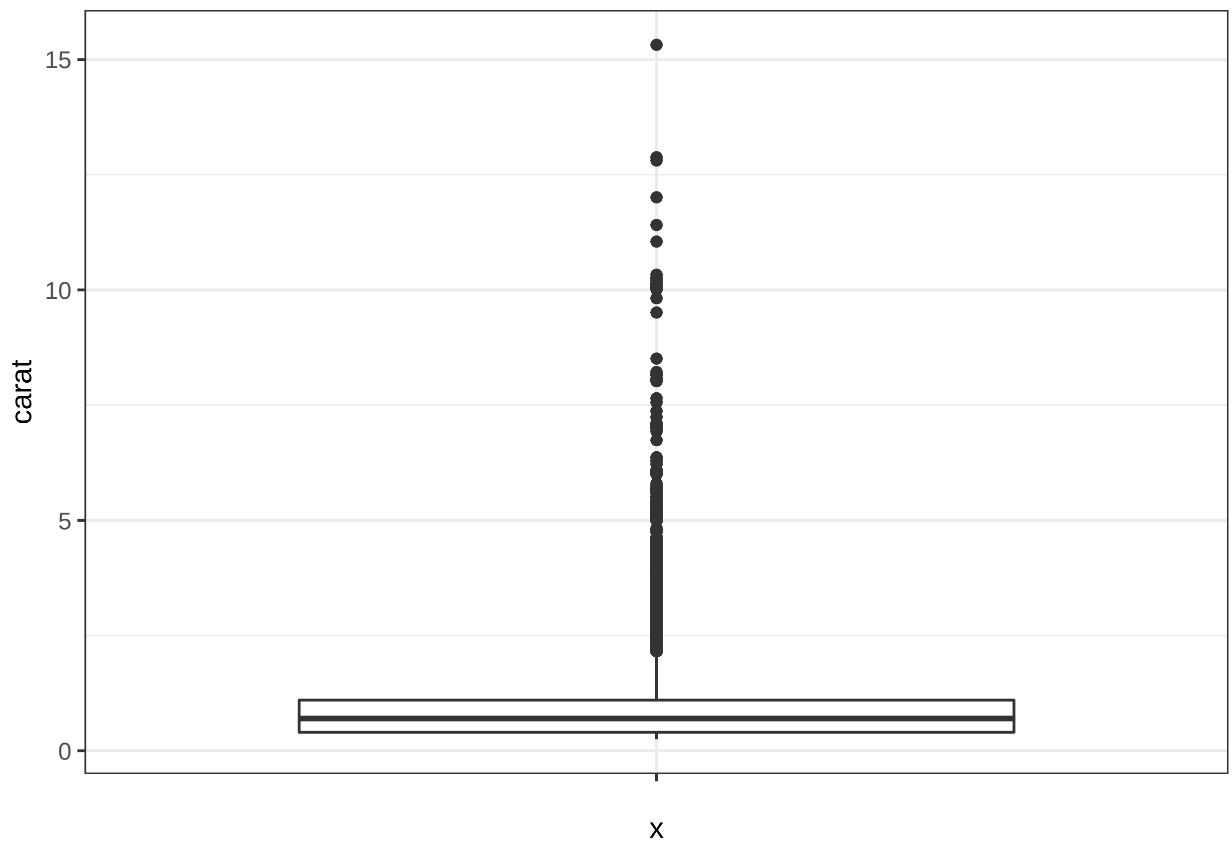



```
#Since it appears to be a normal distribution
```

Data Introduction

1. Carat: The boxplot indicates there are many outliers greater than the 3Q + 1.5IQR.

```
ggplot(diamond, aes(x="", y=carat)) +
  geom_boxplot()
```

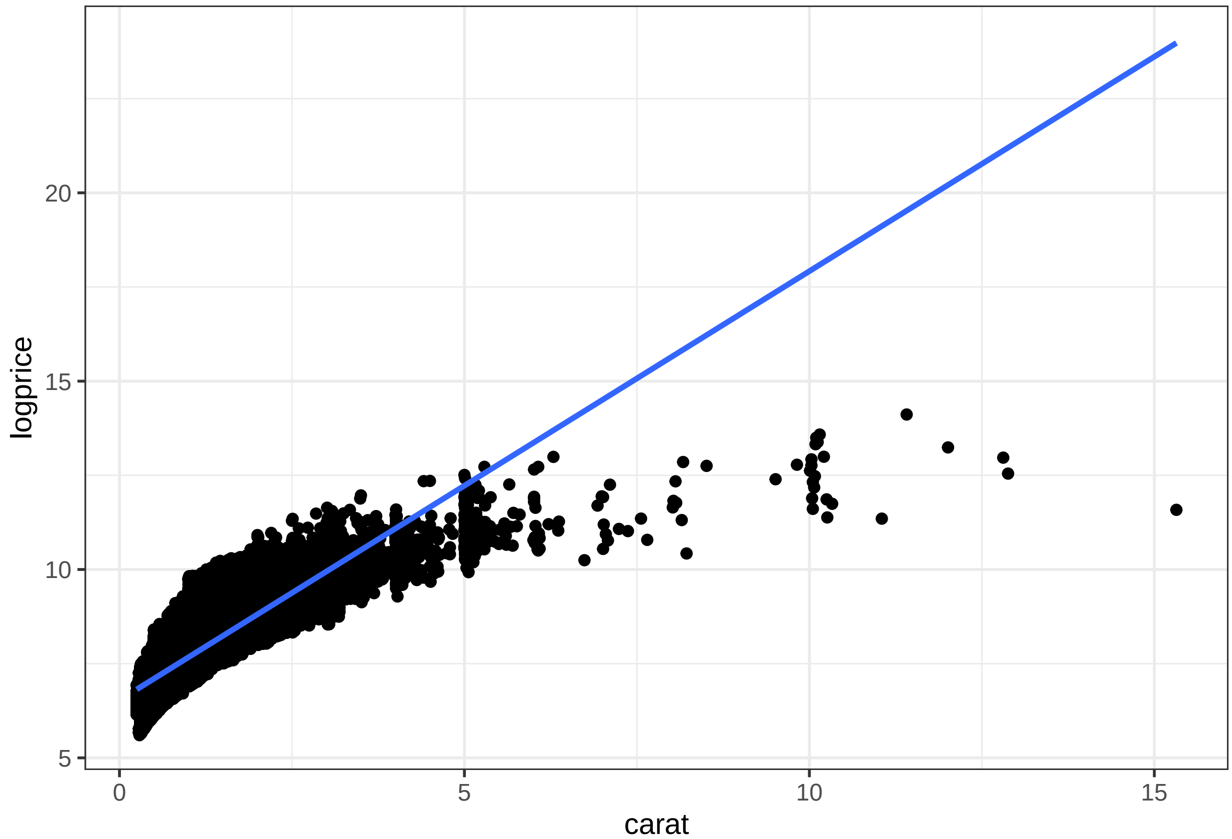


2. The relationship between logprice and carat

When we computed scatter plot between carat and price, we found that most of our datapoints are concentrated in the carat range of 0 to 8. We also found that there are positive relationship between carat and price.

```
ggplot(diamond, aes(x=carat, y=logprice)) +
  geom_point() +
  geom_smooth(method=lm)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
cor(diamond$carat,diamond$price)
```

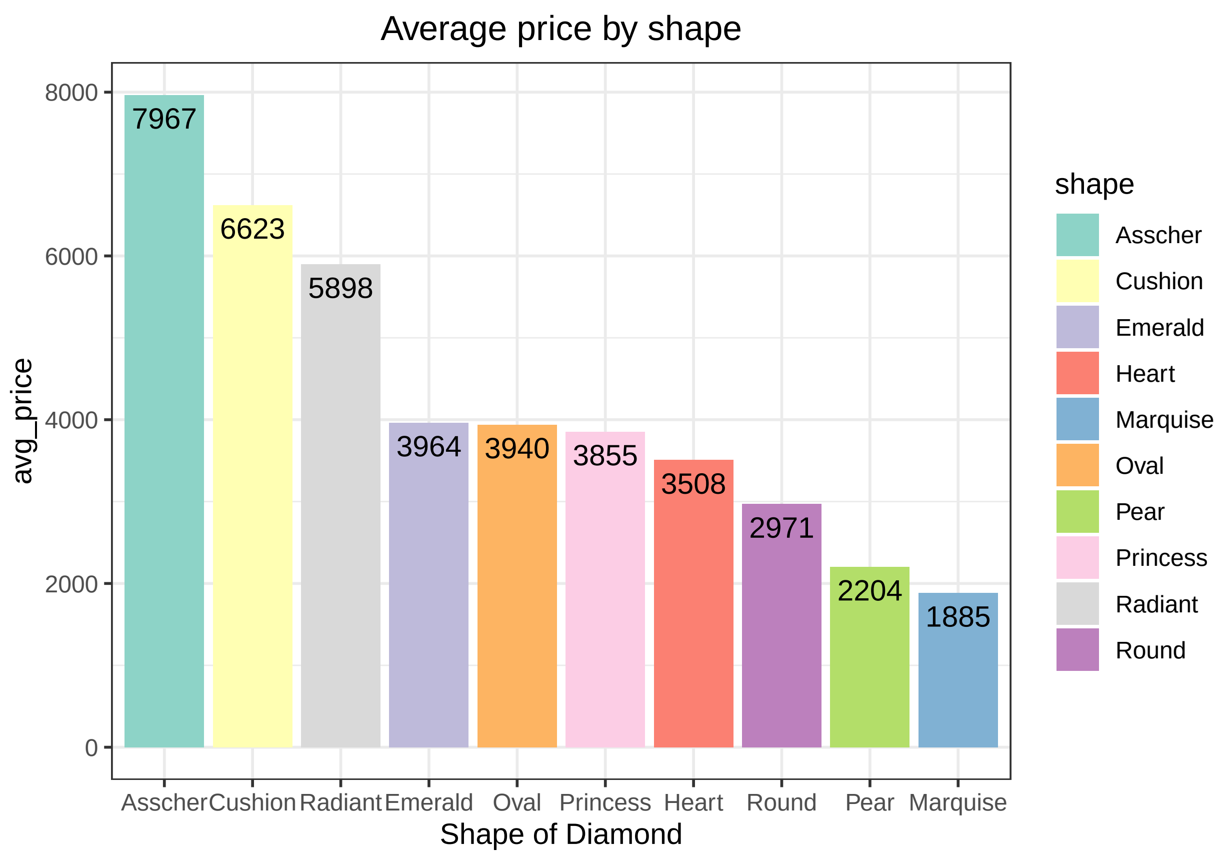
```
## [1] 0.5555784
```

```
##The correlation between carat and price is 0.555784
```

3. Average price of each shape

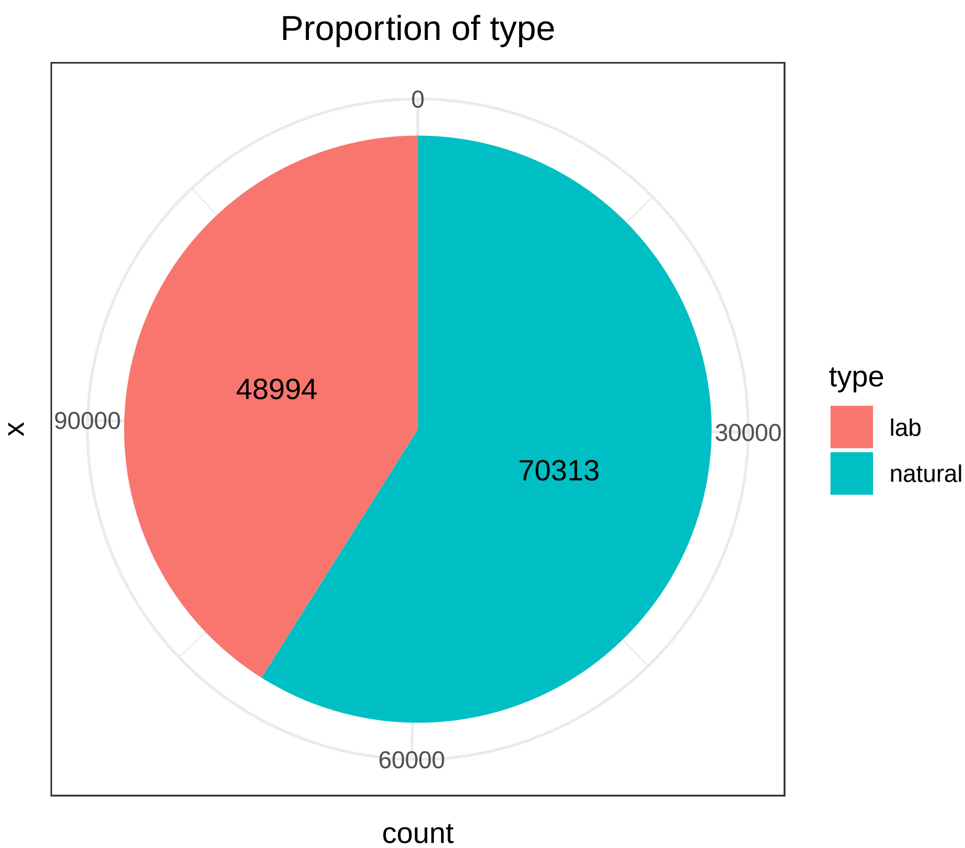
Shape Asscher has the highest averabe price. We thought this is very interesting because according to the report from "The Diamond Regitry" the round shape diamond are the most expensive.

```
avg_p_shape <- diamond[, mean(price), by=shape]
setnames(avg_p_shape, "V1", "avg_price")
avg_p_shape <- avg_p_shape[order(avg_price, decreasing = TRUE)]

ggplot(avg_p_shape, aes(x=reorder(shape, -avg_price), y=avg_price, fill=shape)) +
  geom_bar(stat="identity") +
  scale_fill_brewer(palette = "Set3") +
  geom_text(aes(label=round(avg_price,0)), color="black", vjust=1.6) +
  scale_x_discrete(name ="Shape of Diamond") +
  ggtitle("Average price by shape") +
  theme(plot.title = element_text(hjust = 0.5))
```



4. Proportion of each type

```
type_count <- diamond[,.N, by=type]
setnames(type_count, "N", "count")

ggplot(type_count, aes(x = "", y = count, fill = type)) +
     geom_bar(width = 1, stat = "identity") +
     coord_polar(theta = "y", start = 0) +
     geom_text(aes(label = count), position = position_stack(vjust = 0.5)) +
     ggtitle("Proportion of type") +
     theme(plot.title = element_text(hjust = 0.5),
           axis.ticks = element_blank())
```

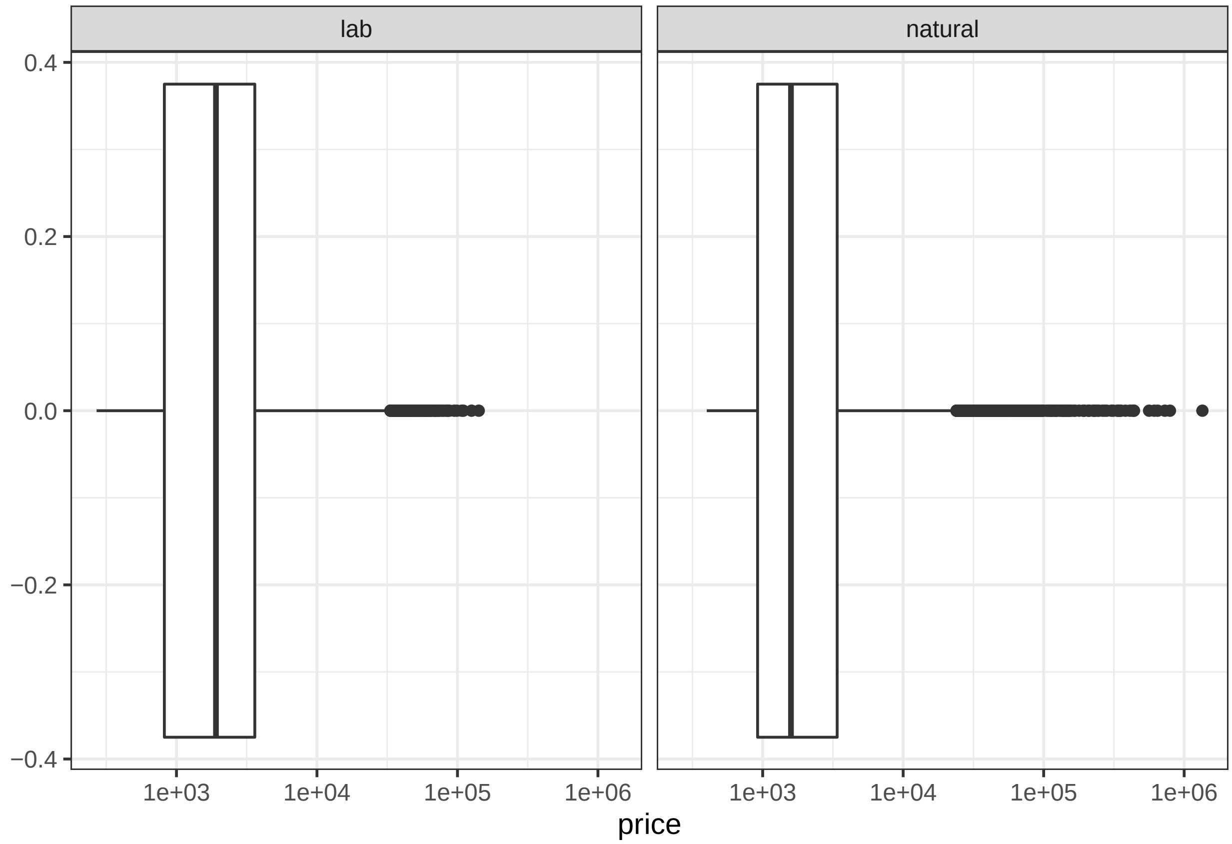


5. Average price for each type In terms of median, lab-produced diamonds has higher price. Natural diamonds, however, are more dispersed and it is more likely to have higher price compared to lab-produced diamonds

```
avg_p_type <- diamond[, mean(price), by=type]
setnames(avg_p_type, "V1", "avg_price")
avg_p_type <- avg_p_type[order(avg_price, decreasing = TRUE)]
avg_p_type
```

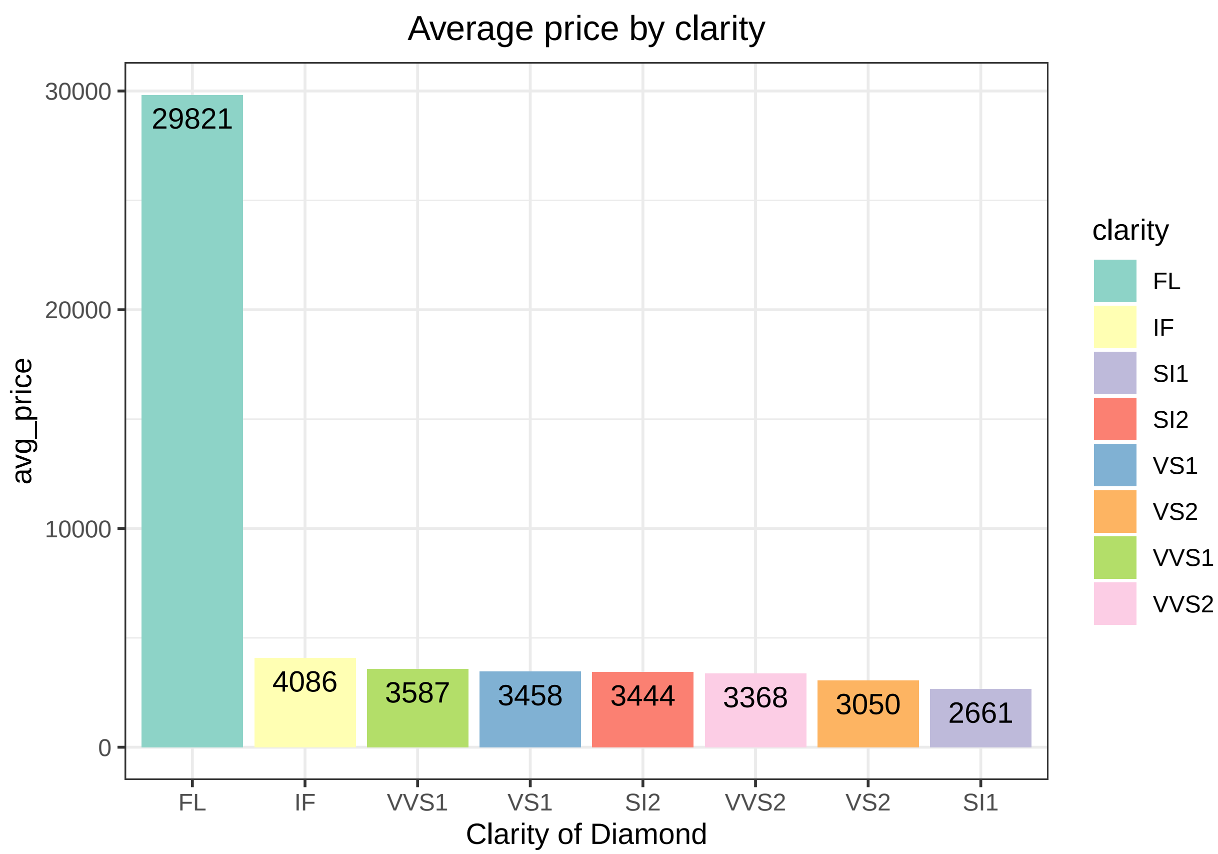
```
##       type avg_price
## 1: natural  3358.127
## 2:     lab  3184.541
```

```
##       type avg_price
## 1: natural  3358.127
## 2:     lab  3184.541
# price distribution by type
ggplot(diamond, aes(x=price)) +
  geom_boxplot() +
  scale_x_log10() +
  facet_wrap(~type)
```



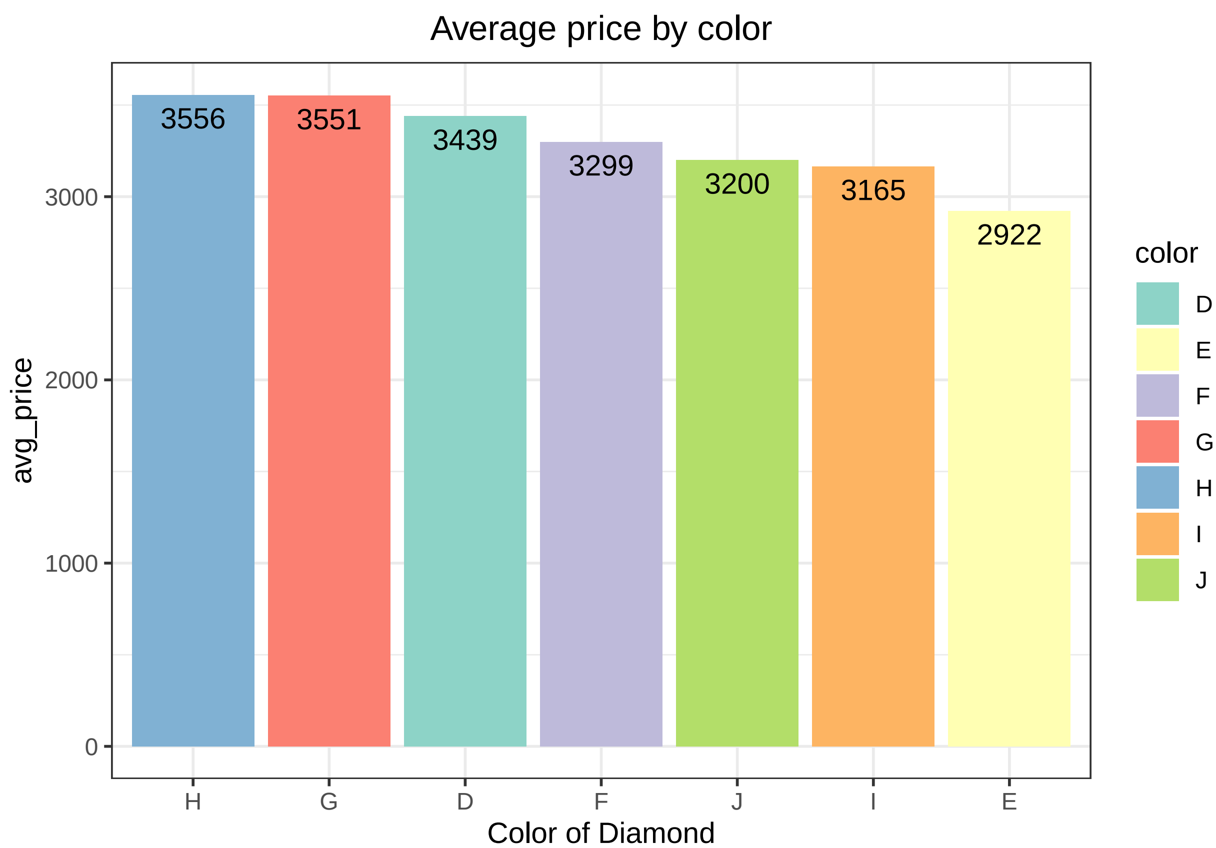
6. Average price of each clarity The FL clarity level shows an exceptionally high price. The price is quite similar throughout the rest of the clarity level.

```
avg_p_clarity <- diamond[, mean(price), by=clarity]
setnames(avg_p_clarity, "V1", "avg_price")
avg_p_clarity <- avg_p_clarity[order(avg_price, decreasing = TRUE)]

ggplot(avg_p_clarity, aes(x=reorder(clarity, -avg_price), y=avg_price,
                          fill=clarity)) +
  geom_bar(stat="identity") +
  scale_fill_brewer(palette = "Set3") +
  geom_text(aes(label=round(avg_price,0)), color="black", vjust=1.6) +
  scale_x_discrete(name ="Clarity of Diamond") +
  ggtitle("Average price by clarity") +
  theme(plot.title = element_text(hjust = 0.5))
```



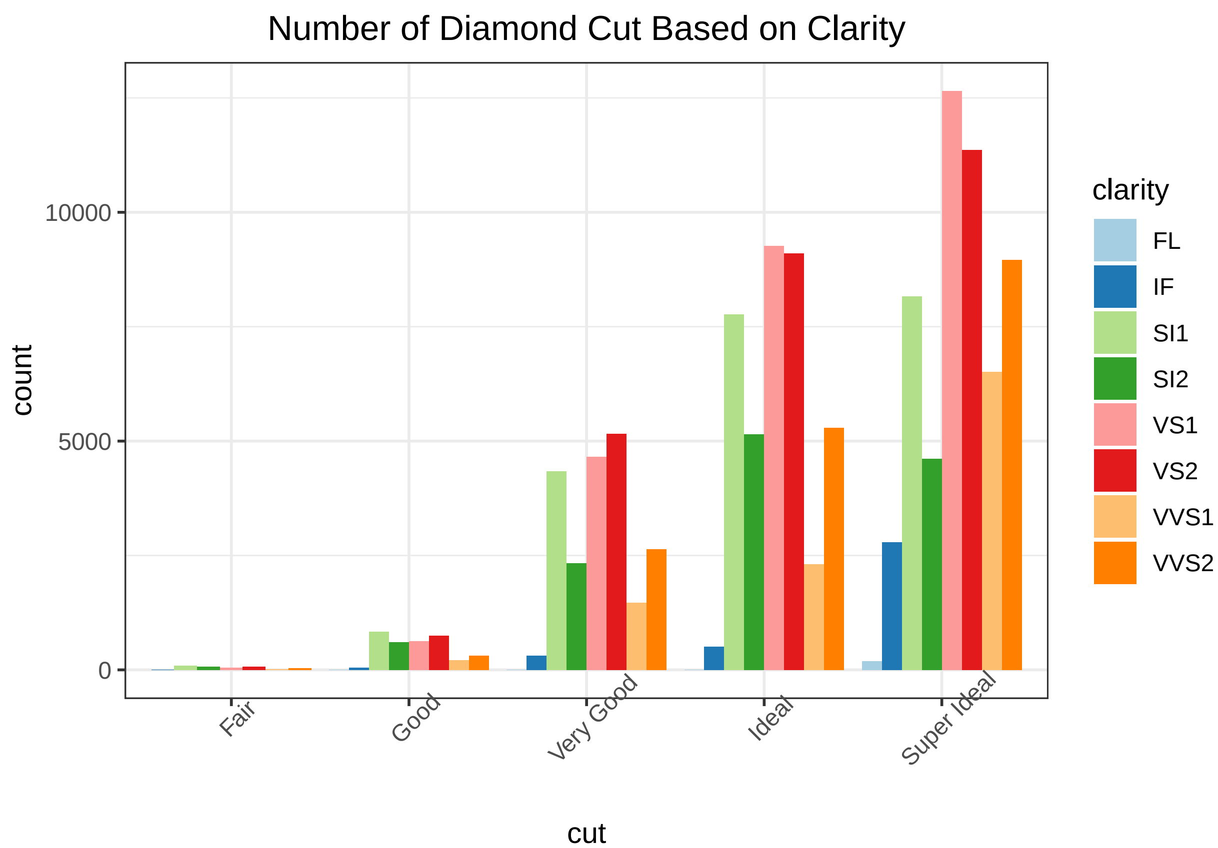
7. Average price of each color Price only slightly varies by the color of the diamonds. Customers are not very sensitive about color

```
avg_p_color <- diamond[, mean(price), by=color]
setnames(avg_p_color, "V1", "avg_price")
avg_p_color <- avg_p_color[order(avg_price, decreasing = TRUE)]

ggplot(avg_p_color, aes(x=reorder(color, -avg_price), y=avg_price, fill=color))+
  geom_bar(stat="identity") +
  scale_fill_brewer(palette = "Set3") +
  geom_text(aes(label=round(avg_price,0)), color="black", vjust=1.6) +
  scale_x_discrete(name ="Color of Diamond") +
  ggtitle("Average price by color") +
  theme(plot.title = element_text(hjust = 0.5))
```



8. The relationship between clarity and cut, and super ideal cut with VS1 clear is the most popular one

```
order_level<-c("Fair","Good","Very Good","Ideal","Super Ideal")
diamond$cut<-factor(diamond$cut,levels=order_level)
ggplot(diamond, aes(x = cut, fill = clarity)) +
  geom_bar(position = "dodge") +
  theme(axis.text.x = element_text(angle = 45))+
  labs(title="Number of Diamond Cut Based on Clarity")+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_fill_brewer(palette="Paired")
```



Machine Learning

##Dummy variables Conversion##

```
dd_dummies <- dummy_cols(diamond,select_columns = NULL)
colnames(dd_dummies)[colnames(dd_dummies) == 'cut_Super Ideal']<-'cut_SuperIdeal'
colnames(dd_dummies)[colnames(dd_dummies) == 'cut_Very Good'] <- 'cut_VeryGood'
```

##Train/Test Split##

```
set.seed(810)
# 70% train, 30% test
data1 = sort(sample(nrow(dd_dummies), nrow(dd_dummies)*.7))
train<-dd_dummies[data1,]
test<-dd_dummies[-data1,]
```

1. Linear Regression Model ## Fit the linear regression model

```
model <- lm(logprice ~ carat + shape_Cushion + shape_Emerald + shape_Heart +
              shape_Marquise + shape_Oval + shape_Pear + shape_Princess +
              shape_Radiant + shape_Round + cut_Good + cut_Ideal +
              cut_SuperIdeal + cut_VeryGood + color_E + color_F + color_G +
              color_H + color_I + color_J + clarity_IF + clarity_SI1 +
```

```
                clarity_SI2 + clarity_VS1 + clarity_VS2 + clarity_VVS1 +
                clarity_VVS2 + report_GIA + report_HRD + report_IGI +
                type_natural, data=train)
```

Predict the diamond price with test set

```
preds <- predict(model, test)
modelEval <- cbind(test$logprice, preds)
colnames(modelEval) <- c('Actual', 'Predicted')
modelEval <- as.data.frame(modelEval)
```

Calculate the MSE test, and the MSE of linear Regression is 0.1782691

```
mse <- mean((modelEval$Actual - modelEval$Predicted)^2)
mse
```

```
## [1] 0.1592207
```

```
#0.1782691
```

2. Lasso Regression

```
f1_L <- as.formula( logprice ~ +cut_Fair+cut_Good+cut_Ideal+cut_SuperIdeal+
                      cut_VeryGood+
                      color_D+color_E+color_F+color_G+color_H+color_I+color_J+
                      report_GCAL+report_GIA+report_HRD+report_IGI+
                      clarity_FL+clarity_IF+clarity_SI1+clarity_SI2+
                      clarity_VS1+clarity_VS2+clarity_VVS1+clarity_VVS2+
                      type_lab+type_natural+
                      shape_Asscher+shape_Cushion+
                      shape_Emerald+shape_Heart+shape_Marquise+shape_Oval+
                      shape_Pear+shape_Princess+shape_Radiant+shape_Round+
                      carat)

#Training the model
x.train_L <- model.matrix(f1_L, train)[, -1]
y.train_L <- train$logprice
x.test_L <- model.matrix(f1_L, test)[, -1]
y2.test_L <- test$logprice
f1.L <- cv.glmnet(x.train_L, y.train_L, alpha = 1, nfolds = 10)


# Finding Test and Train MSEs
yhat.train.L <- predict(f1.L, x.train_L, s = f1.L$lambda.1se)
mse.train.L <- mean((y.train_L - yhat.train.L)^2)
yhat.test.L <- predict(f1.L, x.test_L, s = f1.L$lambda.1se)
mse.test.L <- mean((y2.test_L - yhat.test.L)^2)
L.coef <- predict(f1.L, type = "coefficients",s = f1.L$lambda.1se)
```

```r
#Coefficients
L.coef
```

```
## 38 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)     6.582366e+00
## cut_Fair        .
## cut_Good        .
## cut_Ideal       .
## cut_SuperIdeal  8.294424e-02
## cut_VeryGood    .
## color_D         .
## color_E         .
## color_F         .
## color_G         5.671835e-03
## color_H         .
## color_I        -2.742259e-02
## color_J        -1.707647e-01
## report_GCAL     .
## report_GIA      .
## report_HRD      .
## report_IGI     -6.476407e-03
## clarity_FL      1.100471e-01
## clarity_IF      2.553536e-02
## clarity_SI1    -7.705931e-02
## clarity_SI2    -4.684686e-02
## clarity_VS1     .
## clarity_VS2     .
## clarity_VVS1    2.594173e-02
## clarity_VVS2    1.740235e-03
## type_lab       -7.097128e-01
## type_natural    7.122101e-11
## shape_Asscher   .
## shape_Cushion   .
## shape_Emerald  -1.050063e-02
## shape_Heart     .
## shape_Marquise  .
## shape_Oval      .
## shape_Pear     -7.494775e-03
## shape_Princess -9.169590e-03
## shape_Radiant   .
## shape_Round     7.610044e-02
## carat           1.352302e+00
```

```r
mse.test.L
```

```
## [1] 0.1666735
```

3. Ridge Regression

```r
f1_R <- as.formula( logprice ~ cut_Fair+cut_Good+cut_Ideal+cut_SuperIdeal+
                      cut_VeryGood+
                      color_D+color_E+color_F+color_G+color_H+color_I+color_J+
                      report_GCAL+report_GIA+report_HRD+report_IGI+
                      clarity_FL+clarity_IF+clarity_SI1+clarity_SI2+
                      clarity_VS1+clarity_VS2+clarity_VVS1+clarity_VVS2+
                      type_lab+type_natural+
                      shape_Asscher+shape_Cushion+shape_Emerald+
                      shape_Heart+shape_Marquise+shape_Oval+shape_Pear+
                      shape_Princess+shape_Radiant+shape_Round +carat)

#Training the Model
x.train_R <- model.matrix(f1_R, train)[, -1]
y.train_R <- train$logprice
x.test_R <- model.matrix(f1_R, test)[, -1]
y.test_R <- test$logprice
f1.R <- cv.glmnet(x.train_R, y.train_R, alpha = 0, nfolds = 10)

#Finding MSes

yhat.train.R <- predict(f1.R, x.train_R, s = f1.R$lambda.1se)
mse.train.R <- mean((y.train_R - yhat.train.R)^2)
yhat.test.R <- predict(f1.R, x.test_R, s = f1.R$lambda.1se)
mse.test.R <- mean((y.test_R - yhat.test.R)^2)
R.coef <- predict(f1.R, type = "coefficients",s = f1.R$lambda.1se)

#Coefficients
R.coef
```

```
## 38 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)     6.364935071
## cut_Fair       -0.037896116
## cut_Good       -0.044373886
## cut_Ideal      -0.028277802
## cut_SuperIdeal  0.059642943
## cut_VeryGood   -0.049869662
## color_D        -0.001129659
## color_E        -0.023694685
## color_F         0.044099252
## color_G         0.069277501
## color_H         0.038972454
## color_I        -0.028923681
## color_J        -0.159029399
## report_GCAL    -0.032859855
## report_GIA      0.087040171
## report_HRD      0.011753352
## report_IGI     -0.088834039
## clarity_FL      0.574965510
## clarity_IF      0.115878507
## clarity_SI1    -0.094875560
## clarity_SI2    -0.049634144
## clarity_VS1     0.030878151
```

```
## clarity_VS2    -0.021358368
## clarity_VVS1    0.079862807
## clarity_VVS2    0.047431259
## type_lab       -0.249323369
## type_natural    0.243853065
## shape_Asscher   0.154273944
## shape_Cushion   0.093994964
## shape_Emerald  -0.090050108
## shape_Heart    -0.035740516
## shape_Marquise -0.126062173
## shape_Oval      0.021101344
## shape_Pear     -0.108843444
## shape_Princess -0.091458255
## shape_Radiant   0.137353991
## shape_Round     0.047562653
## carat           1.233000146
```

```
mse.test.R
```

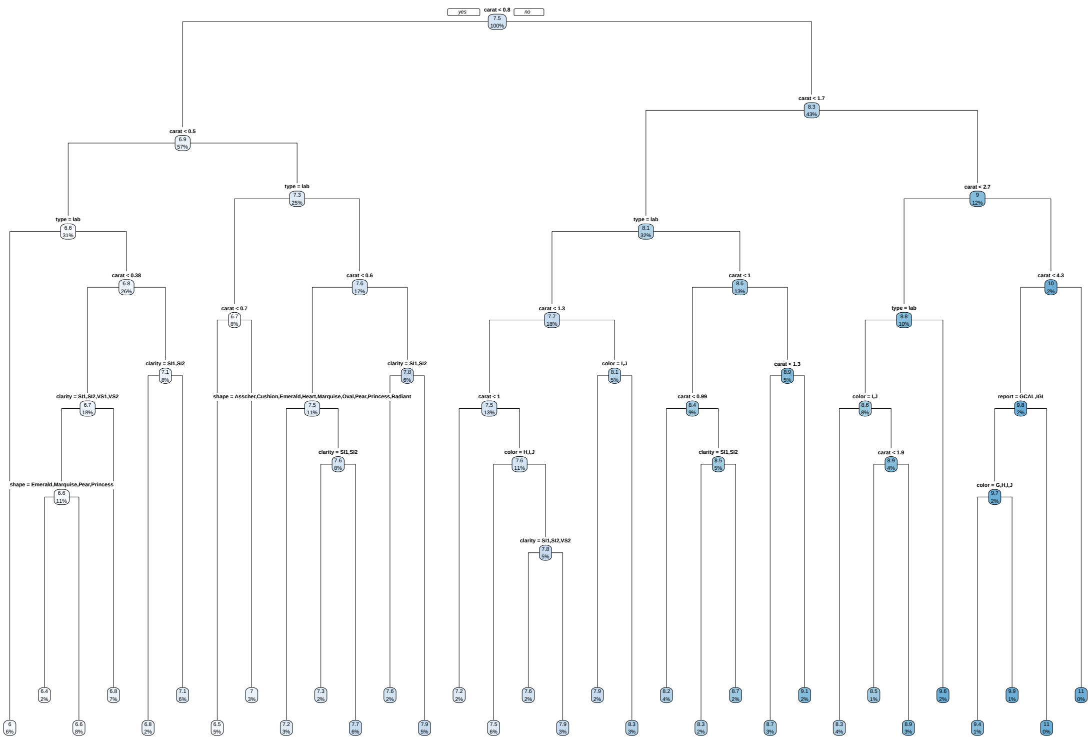
```
## [1] 0.1735329
```

4. Regression Tree

```
x.train_RT<- train[,.(cut,color,report,clarity,type,shape,carat,logprice)]
x.test_RT <- test[,.(cut,color,report,clarity,type,shape,carat,logprice)]

f1_RT <- as.formula( logprice ~ carat + cut + clarity + color+
                    report+type+shape)

#Model Training
x.train.RT <- model.matrix(f1_RT, x.train_RT)[, -1]
y.train.RT <- x.train_RT$logprice
x.test.RT <- model.matrix(f1_RT, x.test_RT)[, -1]
y.test.RT <- x.test_RT$logprice
fit.tree <- rpart(f1_RT,
                  x.train_RT,
                  control = rpart.control(cp = 0.001))

#Plotting and Computing MSEs
par(xpd = TRUE)
yhat.tree <- predict(fit.tree,x.train_RT)
mse.tree <- mean((yhat.tree - y.train.RT) ^ 2)
mse.tree
```

```
## [1] 0.05245066
```

```
yhat.tree.test <- predict(fit.tree,x.test_RT)
mse.tree.test <- mean((yhat.tree.test - y.test.RT) ^ 2)

rpart.plot(fit.tree, type = 1)
```

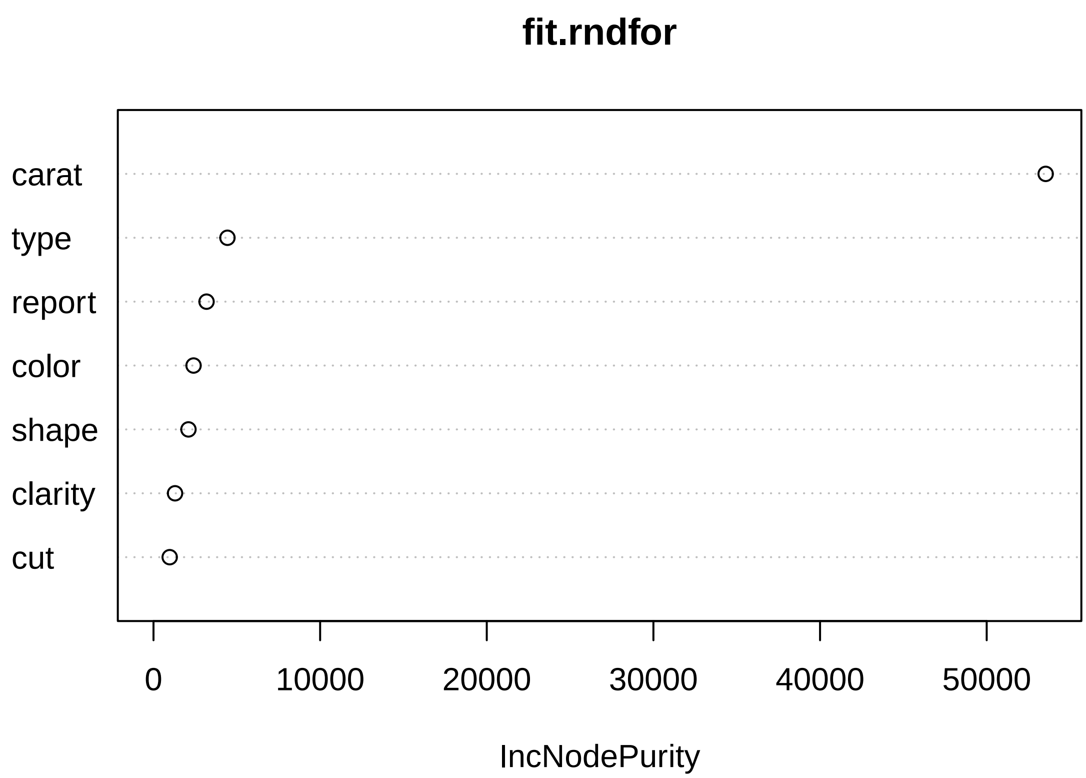


```
mse.tree
```

```
## [1] 0.05245066
```

5.random Forest

```
f1_RFC <- as.formula( logprice ~ carat + cut + clarity + color+
                    report+type+shape)


fit.rndfor <- randomForest(f1_RFC, x.train_RT,
                           ntree=500,
                           do.trace=F)

#Calculating the MSEs
yhat.rndfor <- predict(fit.rndfor, x.train_RT)
mse.tree_RFC <- mean((yhat.rndfor - y.train.RT) ^ 2)


yhat.rndfor.test <- predict(fit.rndfor, x.test_RT)
mse.tree.test_RFC <- mean((yhat.rndfor.test - y.test.RT) ^ 2)


varImpPlot(fit.rndfor)
```




```
mse.tree.test_RFC
```

```
## [1] 0.02585908
```

6. Boosted Forest

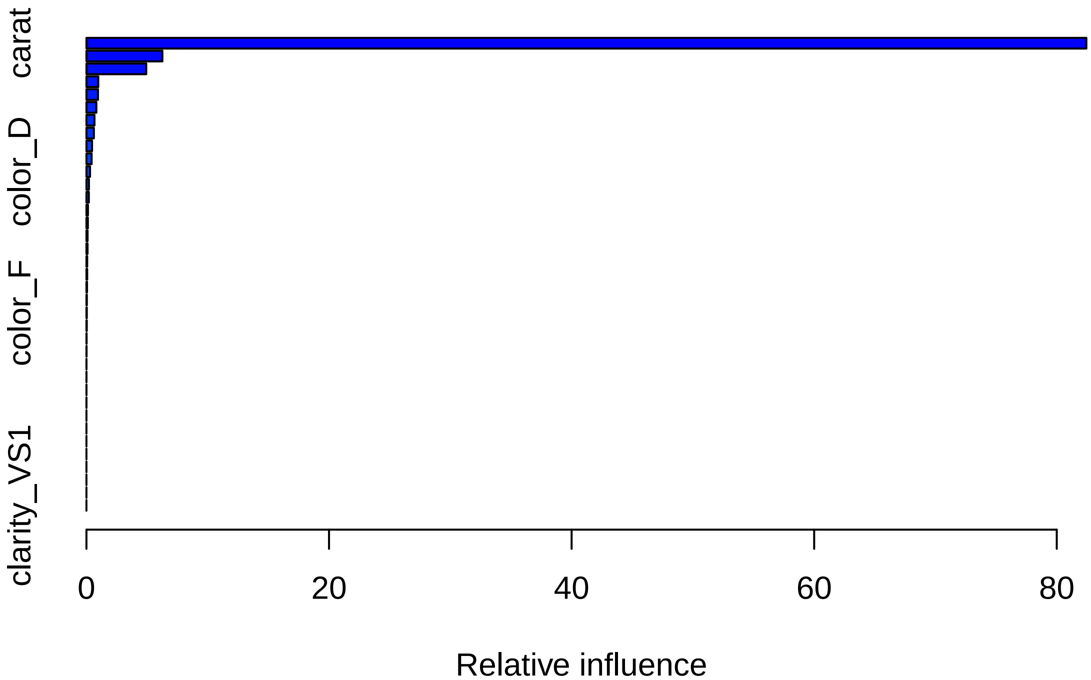
```
f1_BF <- as.formula( logprice ~ cut_Fair+cut_Good+cut_Ideal+cut_SuperIdeal+
                       cut_VeryGood+
                      color_D+color_E+color_F+color_G+color_H+color_I+color_J+
                      report_GCAL+report_GIA+report_HRD+report_IGI+
                      clarity_FL+clarity_IF+clarity_SI1+clarity_SI2+
                       clarity_VS1+clarity_VS2+
                       clarity_VVS1+clarity_VVS2+
                      type_lab+type_natural+
                      shape_Asscher+shape_Cushion+shape_Emerald+
                       shape_Heart+shape_Marquise+shape_Oval+shape_Pear+
                       shape_Princess+shape_Radiant+shape_Round+
                      carat)


#Training the model
x.train_BF <- model.matrix(f1_BF, train)[, -1]
y.train_BF <- train$logprice
x.test_BF <- model.matrix(f1_BF, test)[, -1]
y.test_BF <- test$logprice
```

```r
fit.btree <- gbm(f1_BF,train,
      distribution = "gaussian",
      n.trees = 5000,
      interaction.depth = 1,
      shrinkage = 0.1, cv.folds=5)

#CV gives best iteration to be 4982
#Calculating MSEs
relative.influence(fit.btree)
```

```
## n.trees not given. Using 4998 trees.
```

```
##       cut_Fair       cut_Good      cut_Ideal cut_SuperIdeal   cut_VeryGood
##   1.030329e+01   6.874942e+00   2.049293e+00   1.608612e+03   8.152047e-01
##        color_D        color_E        color_F        color_G        color_H
##   5.833208e+02   1.797247e+02   6.010120e+01   2.088094e+00   2.310160e+02
##        color_I        color_J    report_GCAL     report_GIA     report_HRD
##   1.301261e+03   1.916647e+03   3.028206e+00   1.868797e+03   5.575315e+01
##     report_IGI    clarity_FL     clarity_IF    clarity_SI1    clarity_SI2
##   2.451266e+02   1.844975e+02   3.928644e+02   8.427884e+02   1.205447e+03
##    clarity_VS1    clarity_VS2   clarity_VVS1   clarity_VVS2       type_lab
##   6.207928e-01   7.043650e+01   4.038732e+02   1.222937e+02   9.678883e+03
##   type_natural  shape_Asscher  shape_Cushion  shape_Emerald    shape_Heart
##   1.229112e+04   8.579215e+00   2.210102e+01   1.228319e+02   1.011160e+00
## shape_Marquise     shape_Oval     shape_Pear shape_Princess  shape_Radiant
##   1.092635e+00   3.031033e+01   8.237342e-01   9.507161e+01   1.206811e+00
##    shape_Round          carat
##   9.100689e+02   1.617876e+05
```

```r
yhat.btree <- predict(fit.btree, train, n.trees = 4982)
mse.btree <- mean((yhat.btree - y.train_BF) ^ 2)


yhat.btree.test <- predict(fit.btree, test, n.trees = 4982)
mse.btree.test <- mean((yhat.btree.test - y.test_BF) ^ 2)

print(summary.gbm(fit.btree))
```




```
##                             var      rel.inf
## carat                     carat 8.243985e+01
## type_natural       type_natural 6.263017e+00
## type_lab               type_lab 4.931934e+00
## color_J                 color_J 9.766393e-01
## report_GIA           report_GIA 9.522570e-01
## cut_SuperIdeal cut_SuperIdeal 8.196778e-01
## color_I                 color_I 6.630652e-01
## clarity_SI2         clarity_SI2 6.142430e-01
## shape_Round         shape_Round 4.637312e-01
## clarity_SI1         clarity_SI1 4.294480e-01
## color_D                 color_D 2.972347e-01
## clarity_VVS1       clarity_VVS1 2.057961e-01
## clarity_IF           clarity_IF 2.001865e-01
## report_IGI           report_IGI 1.249058e-01
## color_H                 color_H 1.177156e-01
## clarity_FL           clarity_FL 9.401181e-02
## color_E                 color_E 9.157981e-02
## shape_Emerald     shape_Emerald 6.258973e-02
## clarity_VVS2       clarity_VVS2 6.231552e-02
## shape_Princess shape_Princess 4.844432e-02
## clarity_VS2         clarity_VS2 3.594015e-02
## color_F                 color_F 3.062493e-02
## report_HRD           report_HRD 2.840936e-02
## shape_Oval           shape_Oval 1.544481e-02
## shape_Cushion     shape_Cushion 1.132613e-02
```

```
## cut_Fair                cut_Fair 5.250103e-03
## shape_Asscher   shape_Asscher 4.371592e-03
## cut_Good                cut_Good 3.503169e-03
## report_GCAL       report_GCAL 1.543041e-03
## color_G                  color_G 1.064001e-03
## cut_Ideal              cut_Ideal 1.044230e-03
## shape_Radiant   shape_Radiant 6.149380e-04
## shape_Marquise shape_Marquise 5.567587e-04
## shape_Heart       shape_Heart 5.152427e-04
## shape_Pear          shape_Pear 4.197388e-04
## cut_VeryGood     cut_VeryGood 4.153925e-04
## clarity_VS1       clarity_VS1 3.163288e-04
```

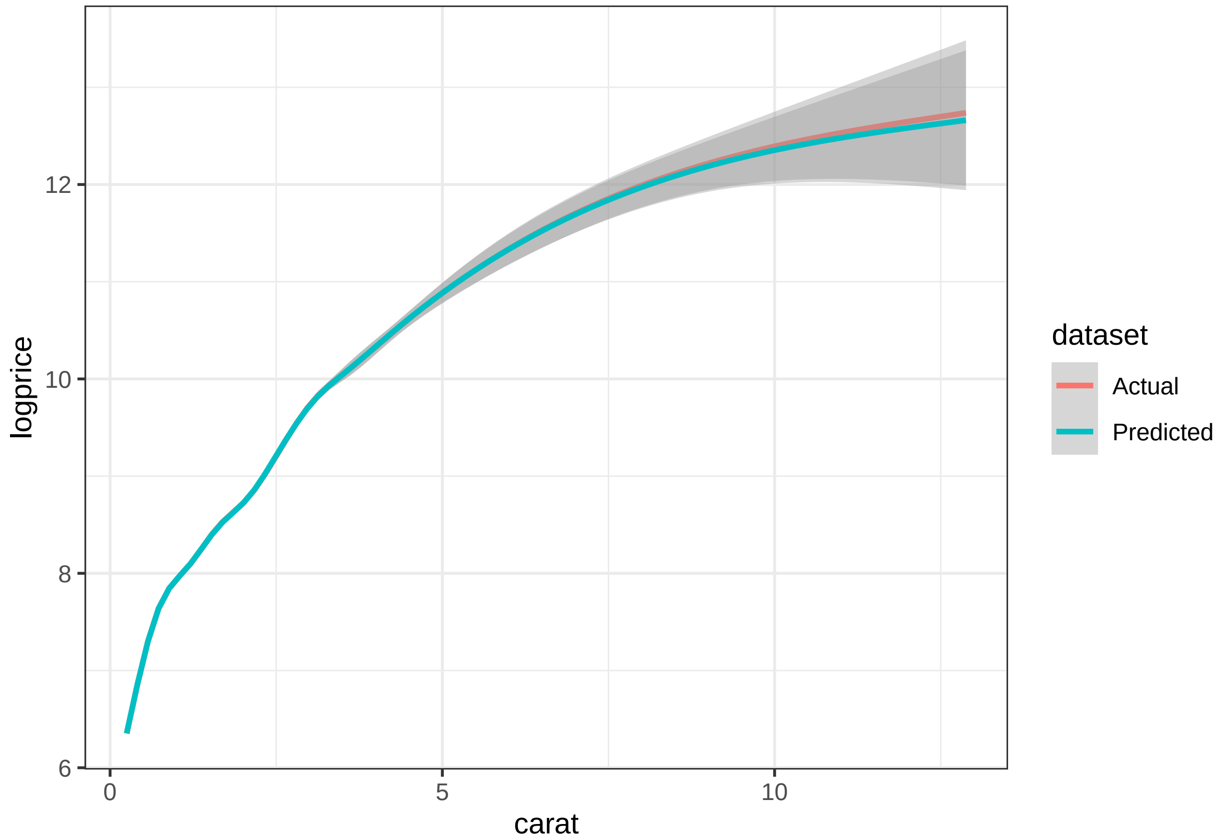
```
mse.btree.test
```

```
## [1] 0.01824531
```

7. Conclusion

```
A <- data.table(
  carat = test$carat,
   logprice = y.test_BF,
  dataset = "Actual"
)
P <- rbind(A, data.table(
  carat = test$carat,
  logprice = yhat.btree.test,
  dataset = "Predicted"))

ggplot(P, aes(carat, logprice, color=dataset)) + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
 # Checking Points where dimaond is overpriced and where it is underpriced
#against the most important variable using Boosted Trees, which comes out to
#be the best model!
```