Fall Term 2022
Adv Stat Learning I (MATH-7635)

Team:
Rahul Marru
Manivardhan reddy Pindi

Project Title: Diamond Price Prediction Using R.

Abstract:

Gemstones like diamonds are always in demand because of their value in the investment market. This makes it very important for diamond dealers to predict its accurate price. However, the prediction process is difficult due to the wide variation in the diamond stones sizes and characteristics [1]. We are solved the problem of predicting diamond prices in the United States by Subset Selection and Shrinkage Methods like Forward selection, Backward selection, LASSO and Ridge Regression models.

Dataset Source: The data is collected from the data.world, It consists of the number of observations are 53940(almost 54000) and the number of predictors is 11.

Link: https://data.world/nahrin/diamonds

Dataset Description:

Attribute	Description
price	in US dollars ($326 - $18,823)
carat	weight of the diamond (0.2 - 5.01)
cut	quality of the cut (Fair, Good, Very Good, Premium, Ideal)
color	diamond color, from J (worst) to D (best)
clarity	a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))
X	length in mm (0 - 10.74)
y	width in mm (0 - 58.9)
z	depth in mm (0 - 31.8)
depth	total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43 - 79)
table	width of top of diamond relative to widest point (43 - 95)

Introduction:

That Diamond is one of the strongest and the most valuable substances produced naturally as a form of carbon. However, unlike gold and silver, determining the price of a diamond is very complex because many features are to be considered for its price. How Linear Model Selection and Regularization methods can predict the price of the diamond you desire to buy and the basic measurement metrics to measure

the quality of Models [2]. Diamond is a solid form of carbon element that present in crystal structure that known as diamond cubic making it unique. Diamond is known with their hardness, good thermal conductivity, high index of refraction, high dispersion, and adamantine luster. The high luster gives diamond the ability to reflect lights that strikes on their surface thus giving them the 'sparkle'. Color and clarity determine the price of diamond to be selected as jewelry gems. Jewelry diamonds have the lowest number of specific gravity with it happens to be very close to 3.52 with minimal impurities and defects. Quality of diamonds that are made into jewelry gems are determined by color, cut, clarity and carat weight. Another category of diamonds that are currently becoming a trend among diamond jewelry lovers are colored diamonds that occur in variety of hues such as red, pink, yellow, orange, purple, blue, green, and brown. The quality of this diamond's type is determined by intensity, purity, and quality of their color, which, the most saturated and vivid color hold a greater price [3].

Literation Review:

The authors, Abirami R and Agniswar P (2021) [4] made a study that Diamonds are one of the most valuable gems in the world. It is also one of the most expensive gems, and therefore has an extremely volatile price. The value of diamonds depends upon their structure, cut, inclusions (impurities), carats, and many other features. Diamond prices are usually set for the day and traded in US Dollars. To better predict the price of diamonds, the Kaggle diamond dataset is used and a scatterplot of metrics such as carats, price, and the color is used to understand the nature of their relationships. They solve the problem of forecasting diamond prices in the United States. They use four prediction algorithms based on this understanding: linear regression, lasso regression, support vector machines, and random forest. From that Random Forest has the most accurate results since it makes use of multiple trees, with 98% accuracy. Secondly, lasso regression has an accurate result almost equal to linear regression. Support Vector Regression (SVR) has the worst result of 68% accuracy. The shows the r2 score is 90% which means the score is good and the prediction is very accurate. The mean Square error is 0.151 which means that we have a very small margin of error since 0 is no error and1 is a high margin of error. Mean Absolute Error is around $805 which is a very small error since the values of diamonds are in hundreds and thousands of dollars. Explained Variance Score is 0.905 which is almost 1, which is the best score.

Step-By-Step's in work approach:

- Import required libraries.
- Data Preprocessing/Feature Engineering and EDA.
- FEATURE SELECTION {Train-Test-Split}
- Model Building And
- Evaluation.
- Import required libraries.

Packages Imported:

```
> library(tidyverse)
  library(ggplot2)
  library(GGally)
  library(MASS)
  library(gridExtra)
  library(caret)
  library(vcd)
  library(scales)
  library(repr)
```

Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a Statistical methods.

- Steps generally do in data preprocessing are Getting Data, Find Missing Values, Encoding Categorical Data, and Splitting dataset into training and test set.

Loading Data:

```
> data <- read_excel("C:/Users/admin/Downloads/rahul/data.xlsx")
New names:
• `` -> `...1`
> View(data)
> data <- subset(data, select = -c(1))
> head(data)
# A tibble: 6 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <chr>     <chr> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
>
```

Dimensions of Data:

```
> ncol(data)
[1] 10
> nrow(data)
[1] 53940
> dim(data)
[1] 53940    10
```

Removing the Duplicate Values from the data:

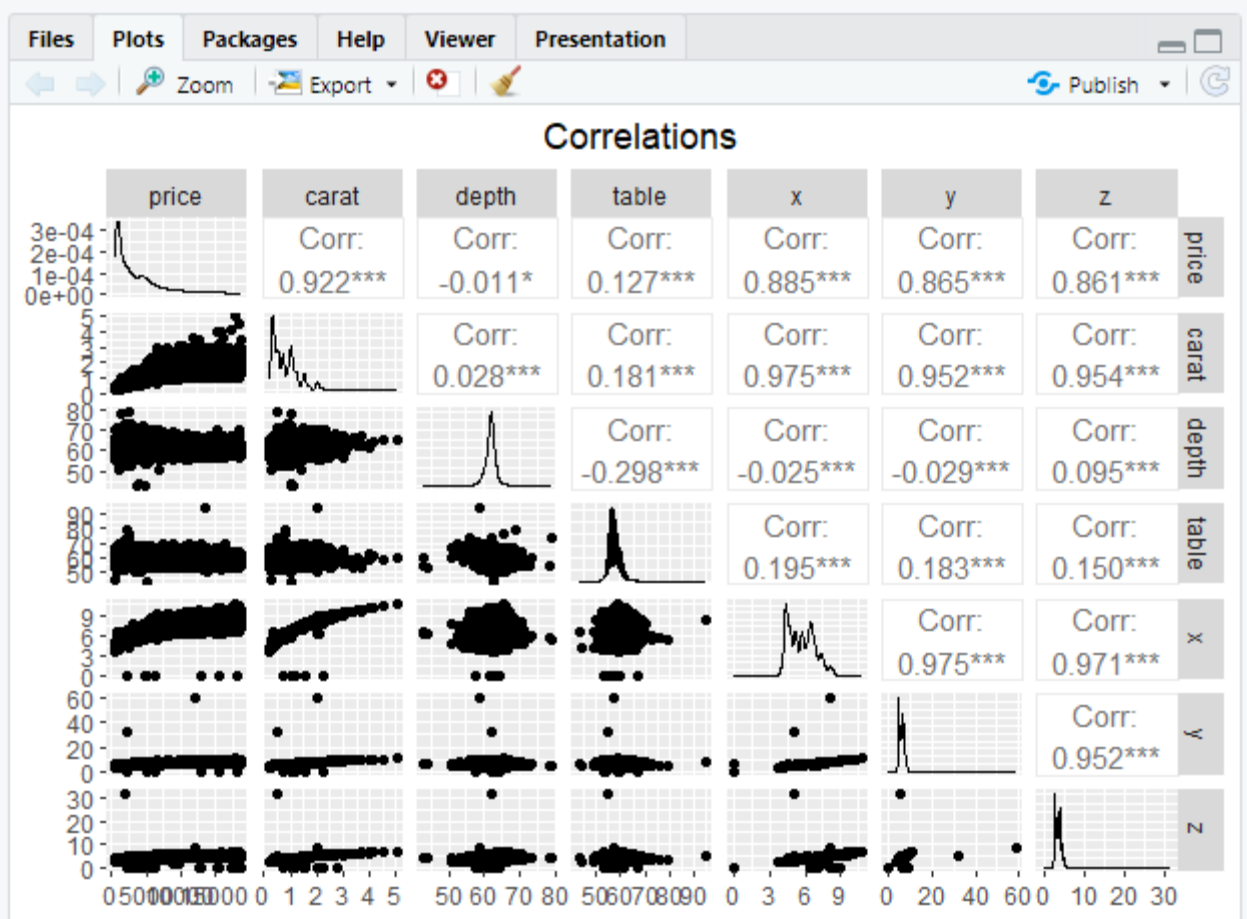
```
> data <- distinct(data)
> dim(data)
[1] 53794    10
```

Summary of Dataset: Describes the statically values of each predictor.

```
> summary(data)
     carat               cut               color             clarity          depth           table      
 Min.   :0.2000   Length:53794       Length:53794       Length:53794       Min.   :43.00   Min.   :43.00  
 1st Qu.:0.4000   Class :character   Class :character   Class :character   1st Qu.:61.00   1st Qu.:56.00  
 Median :0.7000   Mode  :character   Mode  :character   Mode  :character   Median :61.80   Median :57.00  
 Mean   :0.7978                                                           Mean   :61.75   Mean   :57.46  
 3rd Qu.:1.0400                                                           3rd Qu.:62.50   3rd Qu.:59.00  
 Max.   :5.0100                                                           Max.   :79.00   Max.   :95.00  
     price             x                y                z          
 Min.   :  326   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
 1st Qu.:  951   1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
 Median : 2401   Median : 5.700   Median : 5.710   Median : 3.530  
 Mean   : 3933   Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
 3rd Qu.: 5327   3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.030  
 Max.   :18823   Max.   :10.740   Max.   :58.900   Max.   :31.800  
```

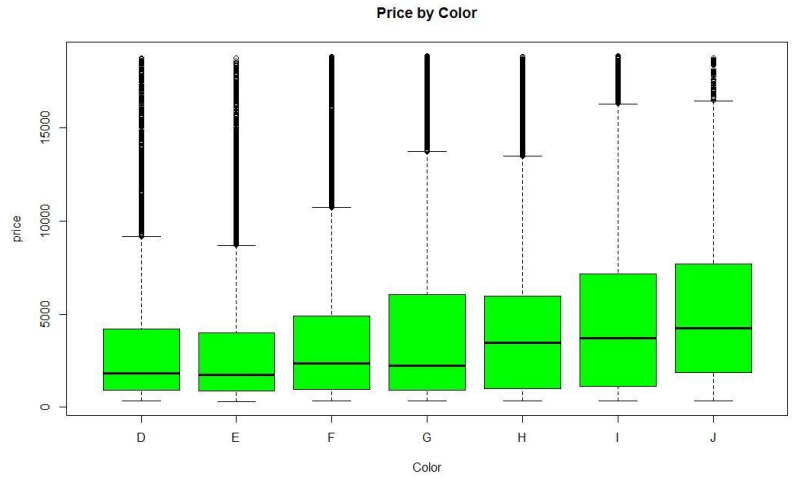
Visualization Plot for Correlations between pairs of numerical variables:

The plot describes the relationship between the each predictor with remain predictor.



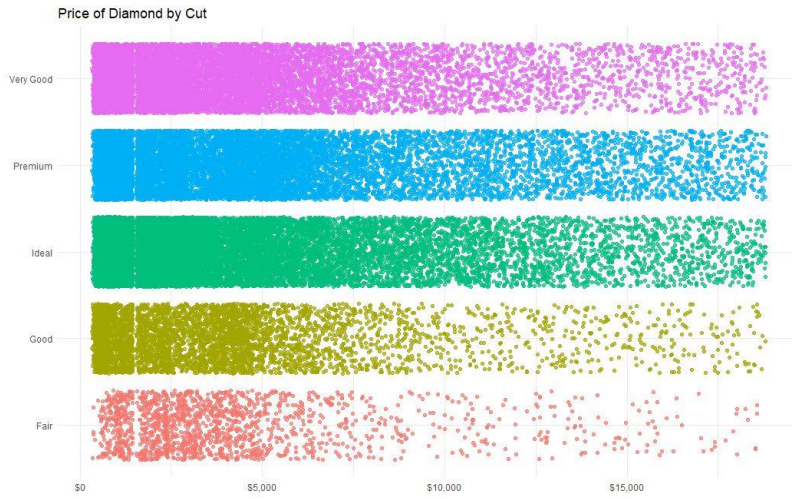
Visualization of Box Plots between price and categorical variable {Color} to check any outliers are present.

```
> boxplot(price~color, data=data,
+         col=("green"),
+         main="Price by Color", xlab="Color")
```



Visually exploring if there are any correlation between the cut and the price of a diamond

```
> library(scales)
> ggplot(data,
+        aes(y = factor(cut,
+                       labels = c("Fair",
+                                  "Good",
+                                  "Ideal",
+                                  "Premium",
+                                  "Very Good")),
+            x = price,
+            color = cut)) +
+     geom_jitter(alpha = 0.7,
+                 size = 1.5) +
+     scale_x_continuous(label = dollar) +
+     labs(title = "Price of Diamond by Cut",
+          x = "",
+          y = "") +
+     theme_minimal() +
+     theme(legend.position = "none")
```



Perform One-Hot Encoding on Data:

We would perform one-hot encoding to convert a categorical variable that contains in clarity, color and cut columns into new variables that contain only 0 and 1 values.

Clarity Column:

```
> data$clarity_ord[data$clarity=="I1"]<-1
Warning message:
Unknown or uninitialised column: `clarity_ord`.
> data$clarity_ord[data$clarity=="SI2"]<-2
> data$clarity_ord[data$clarity=="SI1"]<-3
> data$clarity_ord[data$clarity=="VS2"]<-4
> data$clarity_ord[data$clarity=="VS1"]<-5
> data$clarity_ord[data$clarity=="VVS2"]<-6
> data$clarity_ord[data$clarity=="VVS1"]<-7
> data$clarity_ord[data$clarity=="IF"]<-8
```

Color Column:

```
> data$cut_ord[data$cut=="Fair"]<-1
Warning message:
Unknown or uninitialised column: `cut_ord`.
>
> data$cut_ord[data$cut=="Good"]<-2
> data$cut_ord[data$cut=="Very Good"]<-3
> data$cut_ord[data$cut=="Premium"]<-4
> data$cut_ord[data$cut=="Ideal"]<-5
```

Cut Column:

```
> data$cut_ord[data$cut=="Fair"]<-1
Warning message:
Unknown or uninitialised column: `cut_ord`.
>
> data$cut_ord[data$cut=="Good"]<-2
> data$cut_ord[data$cut=="Very Good"]<-3
> data$cut_ord[data$cut=="Premium"]<-4
> data$cut_ord[data$cut=="Ideal"]<-5
```

One-Hot Encoded Data after Applying:

New predictors will be add to data, such that categorical variable with its corresponding observations.

```
> dummy <- dummyVars(" ~ .", data = data)
> newdata <- data.frame(predict(dummy, newdata = data))
> head(newdata)
  carat cutFair cutGood cutIdeal cutPremium cutVery.Good colorD colorE colorF colorG colorH colorI colorJ
1  0.23       0       0        1          0            0      0      1      0      0      0      0      0
2  0.21       0       0        0          1            0      0      1      0      0      0      0      0
3  0.23       0       1        0          0            0      0      1      0      0      0      0      0
4  0.29       0       0        0          1            0      0      0      0      0      0      1      0
5  0.31       0       1        0          0            0      0      0      0      0      0      0      1
6  0.24       0       0        0          0            1      0      0      0      0      0      0      1
  clarityI1 clarityIF claritySI1 claritySI2 clarityVS1 clarityVS2 clarityVVS1 clarityVVS2 depth table
1         0         0          0          1          0          0           0           0  61.5    55
2         0         0          1          0          0          0           0           0  59.8    61
3         0         0          0          0          1          0           0           0  56.9    65
4         0         0          0          0          0          1           0           0  62.4    58
5         0         0          0          1          0          0           0           0  63.3    58
6         0         0          0          0          0          0           0           1  62.8    57
  price    x    y    z clarity_ord color_ord cut_ord
1   326 3.95 3.98 2.43           2         6       5
2   326 3.89 3.84 2.31           3         6       4
3   327 4.05 4.07 2.31           5         6       2
4   334 4.20 4.23 2.63           4         2       4
5   335 4.34 4.35 2.75           2         1       2
6   336 3.94 3.96 2.48           6         1       3
```

Drop the Original Categorical Variables { clarity_ord, color_ord, cut_ord} in data:

```
> newdata <- subset(newdata, select = -c(clarity_ord, color_ord, cut_ord))
> head(newdata)
  carat cutFair cutGood cutIdeal cutPremium cutVery.Good colorD colorE colorF colorG colorH colorI colorJ
1  0.23       0       0        1          0            0      0      1      0      0      0      0      0
2  0.21       0       0        0          1            0      0      1      0      0      0      0      0
3  0.23       0       1        0          0            0      0      1      0      0      0      0      0
4  0.29       0       0        0          1            0      0      0      0      0      0      1      0
5  0.31       0       1        0          0            0      0      0      0      0      0      0      1
6  0.24       0       0        0          0            1      0      0      0      0      0      0      1
  clarityI1 clarityIF claritySI1 claritySI2 clarityVS1 clarityVS2 clarityVVS1 clarityVVS2 depth table
1         0         0          0          1          0          0           0           0  61.5    55
2         0         0          1          0          0          0           0           0  59.8    61
3         0         0          0          0          1          0           0           0  56.9    65
4         0         0          0          0          0          1           0           0  62.4    58
5         0         0          0          1          0          0           0           0  63.3    58
6         0         0          0          0          0          0           0           1  62.8    57
  price    x    y    z
1   326 3.95 3.98 2.43
2   326 3.89 3.84 2.31
3   327 4.05 4.07 2.31
4   334 4.20 4.23 2.63
5   335 4.34 4.35 2.75
6   336 3.94 3.96 2.48
```

Train-Test Split Procedure:

The procedure involves taking a dataset and dividing it into two subsets.

- **Train Dataset:** Used to fit the machine learning model.
- **Test Dataset:** Used to evaluate the fit machine learning model.

Splitting the data into two individual parts, 70% of training data and 30% of test data

```
> set.seed(11)
> data.split <- initial_split(newdata, prop = 3/4)
> train <- training(data.split)
> test <- testing(data.split)
```

Modeling Techniques:

Stepwise Selection:

Stepwise regression is a technique that iteratively evaluates each independent variable's statistical significance in a linear regression model.

- **Forward selection:**

  Forward Selection chooses a subset of the predictor variables for the final model. We can do forward stepwise in context of linear regression whether n is less than p or n is greater than p. Forward selection is a very attractive approach, because it's both tractable and it gives a good sequence of models.

```
> fwd.pred.train = predict(linear.fwd, train)
Warning message:
In predict.lm(linear.fwd, train) :
  prediction from a rank-deficient fit may be misleading
> mse.fwd.train = MSE(fwd.pred.train)
> r2.fwd.train = rsquared(fwd.pred.train)
> fwd.pred.test = predict(linear.fwd, test)
Warning message:
In predict.lm(linear.fwd, test) :
  prediction from a rank-deficient fit may be misleading
>
> fwd.pred.test = predict(linear.fwd, test)
Warning message:
In predict.lm(linear.fwd, test) :
  prediction from a rank-deficient fit may be misleading
> mse.fwd.test = MSE(fwd.pred.test)
> r2.fwd.test = rsquared(fwd.pred.test)
> summary(linear.fwd)
```

- **Backward selection:**

  It starts with the full least squares model with all p predictors, unlike forward stepwise selection, and iteratively eliminates the least valuable predictor, one at a time. Because we can perform least squares regression when n is more than p, we must be in a position

where we have more observations than variables in order to execute backward selection. We are unable to fit a least squares model if p is higher than n.

```
> bwd.pred.train = predict(linear.bwd, train)
> mse.bwd.train = MSE(bwd.pred.train)
> r2.bwd.train = rsquared(bwd.pred.train)
>
> bwd.pred.test = predict(linear.bwd, test)
> mse.bwd.test = MSE(bwd.pred.test)
> r2.bwd.test = rsquared(bwd.pred.test)
> summary(linear.bwd)
```

- **Lasso Regression:**

  This is a regularization technique used in feature selection using a Shrinkage method also referred to as the penalized regression method. Lasso is short for Least Absolute Shrinkage and Selection Operator, which is used both for regularization and model selection. If a model uses the L1 regularization technique, then it is called lasso regression.

```
> cv.lasso.out = cv.glmnet(xtrain, ytrain, alpha=1)
> bestlam.lasso = cv.lasso.out$lambda.min
> i <- which(cv.lasso.out$lambda == cv.lasso.out$lambda.min)
> mse.min.lasso <- cv.lasso.out$cvm[i]
> lasso.model = glmnet(xtrain, ytrain, alpha=1, lambda=bestlam.lasso)
> lasso.pred.train = predict(lasso.model, newx=xtrain)
> mse.lasso.train = MSE(lasso.pred.train)
> r2.lasso.train = rsquared(lasso.pred.train)
> lasso.pred.test = predict(lasso.model, newx=xtest)
> mse.lasso.test = MSE(lasso.pred.test)
> r2.lasso.test = rsquared(lasso.pred.test)
```

- **Ridge Regression:**

  Similar to the lasso regression, ridge regression puts a similar constraint on the coefficients by introducing a penalty factor. However, while lasso regression takes the magnitude of the coefficients, ridge regression takes the square. Ridge regression is also known as L2 Regularization.

```
> cv.ridge.out = cv.glmnet(xtrain, ytrain, alpha=0)
> bestlam.ridge = cv.ridge.out$lambda.min
> i <- which(cv.ridge.out$lambda == cv.ridge.out$lambda.min)
> mse.min.ridge <- cv.ridge.out$cvm[i]
> ridge.model = glmnet(xtrain, ytrain, alpha=0, lambda=bestlam.ridge)
> ridge.pred.train = predict(ridge.model, newx=xtrain)
> mse.ridge.train = MSE(ridge.pred.train)
> r2.ridge.train = rsquared(ridge.pred.train)
> ridge.pred.test = predict(ridge.model, newx=xtest)
> mse.ridge.test = MSE(ridge.pred.test)
> r2.ridge.test = rsquared(ridge.pred.test)
```

Metrics for Evaluating Regression Model Performance:

- The Root Mean Square Error is measured by taking the square root of the average of the squared difference between the prediction and the actual value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

```
> MSE <- function(pred){
+     if (length(pred)==length(test$price))
+     {mse = sum((test$price-pred)^2)/length(test$price)}
+     if (length(pred)==length(train$price))
+     {mse = sum((train$price-pred)^2)/length(train$price)}
+     return (mse)}
```

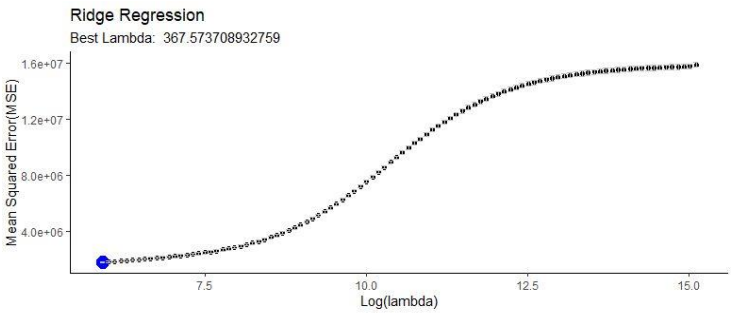
- **Coefficient of Determination or R^2:**

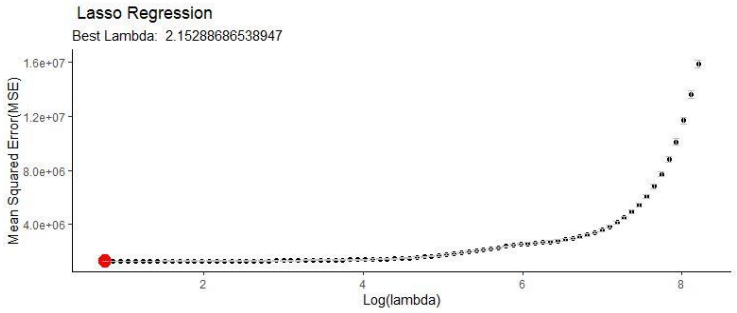
  It measures how well the actual outcomes are replicated by the regression line. It helps you to understand how well the independent variable adjusted with the variance in your model. That means how good is your model for a dataset.

$$R^2 = \frac{SSR}{SST} = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2}$$

```
> rsquared <- function(pred){
+     if (length(pred)==length(test$price)){
+         r2 = 1 - (sum((test$price-pred)^2)/sum((test$price-mean(test$price))^2))}
+     if (length(pred)==length(train$price)){
+         r2 = 1 - (sum((train$price-pred)^2)/sum((train$price-mean(train$price))^2))}
+     return (r2)}
```

Mean Square Error Comparison on Lasso and Ridge Regression:





Result:

- **MSE Score of all four models for both train and test data:**

mse.bwd.test	1275793.41981197
mse.bwd.tra…	1273648.51420872
mse.fwd.test	1275421.37534264
mse.fwd.tra…	1273570.63994261

mse.lasso.t…	1276686.90208875
mse.lasso.t…	1274080.35022849

mse.ridge.t…	1710548.20165803
mse.ridge.t…	1746915.90235272

- **R Squared Score of all four models for both train and test data:**

r2.bwd.test	0.919822734626103
r2.bwd.train	0.919907393364586
r2.fwd.test	0.919846115768917
r2.fwd.train	0.919912290440108

r2.lasso.te…	0.919766583712876
r2.lasso.tr…	0.919880237621006
r2.ridge.te…	0.892500560851466
r2.ridge.tr…	0.890146263563765

Conclusion:

By comparing Lasso regression or L1 Regularization got better than Ridge Regression models when we compared in Plot. The amount of shrinkage for each method is determined by lambda, a tuning parameter. Lambda was chosen for each method by 10-fold cross validation using the mean squared error on the training set to measure performance. The optimal lambda for ridge regression is 367.57 and optimal lambda for lasso model is 2.1588. Lasso is a more accurate model compared to ridge regression.

When comparing all the four techniques such as Forward Selection, Backward Selection, Lasso and Ridge Regression. Among these both forward selection and backward selection got best R-squared score – 0.9198 than other models.

We can reduce the score of MSE of each by normalize the data. Due the response values are much long range in between other predictors.

Reference:

[1] Alsuraihi W, Al-hazmi E, Bawazeer K, & Alghamdi H (2020, March 1). Machine learning algorithms for diamond price prediction: Proceedings of the 2020 2nd International Conference on Image, video and Signal Processing. ACM other conferences. https://dl.acm.org/doi/10.1145/3388818.3393715

[2] Mihir H, Patel M, Jani S, & Gajjar R. (2021). Diamond price prediction using machine learning. 2021 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4). https://doi.org/10.1109/c2i454156.2021.9689412

[3] https://rpubs.com/yousefhosny1/diamond-price-prediction

[4] Abirami R., & Agniswar (2022, December 1). Automated diamond price prediction using machine learning. SRM University AP. https://srmap.edu.in/wp-content/uploads/2021/12/Automated-Diamond-Price-Prediction-Using-Machine-Learning.pdf?x81859

Code:

Libraries:

```
library(tidyverse)
library(dplyr)
library(rsample)
library(randomForest)
library(janitor)
library(rpart)
library(rpart.plot)
library(gbm)
library(glmnet)
library(tree)
library(RColorBrewer)
library(gridExtra)
library(ggplot2)
```

```
library(pls)
library(jtools)
library(magrittr)
library(tidyverse)
library(ggplot2)
library(GGally)
library(MASS)
library(gridExtra)
library(caret)
library(vcd)
library(scales)
library(repr)

dataset:
data <- read_excel("C:/Users/admin/Downloads/rahul/data.xlsx")
View(data)
head(data)

data <- subset(data, select = -c(1))
head(data)

dim(data)
data <- distinct(data)

data <- distinct(data)
dim(data)

data$clarity_ord[data$clarity=="I1"]<-1
data$clarity_ord[data$clarity=="SI2"]<-2
data$clarity_ord[data$clarity=="SI1"]<-3
data$clarity_ord[data$clarity=="VS2"]<-4
data$clarity_ord[data$clarity=="VS1"]<-5
data$clarity_ord[data$clarity=="VVS2"]<-6
data$clarity_ord[data$clarity=="VVS1"]<-7
data$clarity_ord[data$clarity=="IF"]<-8
data$color_ord[data$color=="J"]<-1

data$color_ord[data$color=="I"]<-2
data$color_ord[data$color=="H"]<-3
data$color_ord[data$color=="G"]<-4
data$color_ord[data$color=="F"]<-5
data$color_ord[data$color=="E"]<-6
data$color_ord[data$color=="D"]<-7
data$cut_ord[data$cut=="Fair"]<-1

data$cut_ord[data$cut=="Good"]<-2
data$cut_ord[data$cut=="Very Good"]<-3
data$cut_ord[data$cut=="Premium"]<-4
data$cut_ord[data$cut=="Ideal"]<-5

 dummy <- dummyVars(" ~ .", data = data)
newdata <- data.frame(predict(dummy, newdata = data))
head(newdata)

newdata <- subset(newdata, select = -c(clarity_ord, color_ord, cut_ord))
head(newdata)
```

```
colnames(newdata)

Split data into training and testing sets:
set.seed(11)
data.split <- initial_split(newdata, prop = 3/4)
train <- training(data.split)
test <- testing(data.split)

 rsquared <- function(pred){
   if (length(pred)==length(test$price))
{
       r2 = 1 - (sum((test$price-pred)^2)/sum((test$price-mean(test$price))^2))
}
    if (length(pred)==length(train$price))
{
       r2 = 1 - (sum((train$price-pred)^2)/sum((train$price-mean(train$price))^2))}
    return (r2)
}


MSE <- function(pred)
{
    if (length(pred)==length(test$price))
        {mse = sum((test$price-pred)^2)/length(test$price)
}
    if (length(pred)==length(train$price))
        {
mse = sum((train$price-pred)^2)/length(train$price)}
    return (mse)
}

Forward Selection:
 linear.fwd <- step(lm(price ~., data=train), direction = c("forward"))
fwd.pred.train = predict(linear.fwd, train)
mse.fwd.train = MSE(fwd.pred.train)
r2.fwd.train = rsquared(fwd.pred.train)
fwd.pred.test = predict(linear.fwd, test)

fwd.pred.test = predict(linear.fwd, test)
mse.fwd.test = MSE(fwd.pred.test)
r2.fwd.test = rsquared(fwd.pred.test)
summary(linear.fwd)


Backward Selection:
linear.bwd = step(lm(price ~., data=train), direction = c("backward"))
bwd.pred.train = predict(linear.bwd, train)
mse.bwd.train = MSE(bwd.pred.train)
r2.bwd.train = rsquared(bwd.pred.train)

bwd.pred.test = predict(linear.bwd, test)
mse.bwd.test = MSE(bwd.pred.test)
r2.bwd.test = rsquared(bwd.pred.test)
summary(linear.bwd)
```

```
 grid = 10^seq(10, -2, length=100)
xtrain = model.matrix(price ~.,train)[,-1]
ytrain = train$price
xtest = model.matrix(price ~.,test)[,-1]
ytest = test$price

cv.ridge.out = cv.glmnet(xtrain, ytrain, alpha=0)
bestlam.ridge = cv.ridge.out$lambda.min
i <- which(cv.ridge.out$lambda == cv.ridge.out$lambda.min)
mse.min.ridge <- cv.ridge.out$cvm[i]
ridge.model = glmnet(xtrain, ytrain, alpha=0, lambda=bestlam.ridge)
ridge.pred.train = predict(ridge.model, newx=xtrain)
mse.ridge.train = MSE(ridge.pred.train)
r2.ridge.train = rsquared(ridge.pred.train)
ridge.pred.test = predict(ridge.model, newx=xtest)
mse.ridge.test = MSE(ridge.pred.test)
r2.ridge.test = rsquared(ridge.pred.test)

cv.lasso.out = cv.glmnet(xtrain, ytrain, alpha=1)
bestlam.lasso = cv.lasso.out$lambda.min
i <- which(cv.lasso.out$lambda == cv.lasso.out$lambda.min)
mse.min.lasso <- cv.lasso.out$cvm[i]
lasso.model = glmnet(xtrain, ytrain, alpha=1, lambda=bestlam.lasso)
lasso.pred.train = predict(lasso.model, newx=xtrain)
mse.lasso.train = MSE(lasso.pred.train)
r2.lasso.train = rsquared(lasso.pred.train)
lasso.pred.test = predict(lasso.model, newx=xtest)
mse.lasso.test = MSE(lasso.pred.test)
r2.lasso.test = rsquared(lasso.pred.test)

s1 <- ggplot(mapping = aes(x=log(cv.ridge.out$lambda), y=cv.ridge.out$cvm)) +
  geom_point() +
   geom_point(aes(x=log(bestlam.ridge), y=mse.min.ridge, color="blue", size = 2), show.legend = FALSE,
color="blue") +
    geom_errorbar(aes(ymin=cv.ridge.out$cvm-cv.ridge.out$cvsd, ymax=cv.ridge.out$cvm+cv.ridge.out$cvsd),
color="gray") +
    xlab("Log(lambda)") +
   ylab("Mean Squared Error") +
    labs(title = "Optimal Lambda for Ridge Regression", subtitle = paste("Best Lambda: ", bestlam.ridge)) +
     theme_classic()
s2 <- ggplot(mapping = aes(x=log(cv.lasso.out$lambda), y=cv.lasso.out$cvm)) +
     geom_point() +
     geom_point(aes(x=log(bestlam.lasso), y=mse.min.lasso, size = 2), show.legend = FALSE, color="red") +
     geom_errorbar(aes(ymin=cv.lasso.out$cvm-cv.lasso.out$cvsd, ymax=cv.lasso.out$cvm+cv.lasso.out$cvsd),
color="gray") +
   xlab("Log(lambda)") +
   ylab("Mean Squared Error") +
    labs(title = "Optimal Lambda for Lasso Regression", subtitle = paste("Best Lambda: ", bestlam.lasso)) +
     theme_classic()

 grid.arrange(s1, s2, nrow=2)

ridge.coef <- rownames_to_column(data.frame(coef(ridge.model)[,1]), var = "Variable") %>%
 rename(Coefficient = coef.ridge.model....1.)
```

```
ridge.coef <- ridge.coef %>%
    filter(Variable != "(Intercept)") %>%
    arrange(desc(abs(Coefficient)))
 lasso.coef <- rownames_to_column(data.frame(coef(lasso.model)[,1]), var = "Variable") %>%
    rename(Coefficient = coef.lasso.model....1.)

 lasso.coef <- lasso.coef %>%
    filter(Variable != "(Intercept)") %>%
   arrange(desc(abs(Coefficient)))

coef.compare <- lasso.coef %>%
   left_join(ridge.coef, by = "Variable") %>%
   rename("Lasso Coefficient" = Coefficient.x) %>%
    rename("Ridge Coefficient" = Coefficient.y)

 coef.compare
```