IT 461 Practical Machine Learning
Fall 2020

Project Part II

Group Member Names: Nada AlMutairi , Sara AlFayze
Main Topic of Problem: Diamond price prediction

I. Introduction

a) Introducing the topic and why is it useful to use ML algorithms on it.

A Diamond is the hardest rock known to human. Its name means unbreakable or invincible. Also, it is one of the most precious jewelry. Diamond price affected by many factors like weight and color. Using ML algorithms, we can predict the price of a diamond by knowing its features.

b) What is the problem you want to solve and what are you proposing to solve it with?

We want to predict a diamond price. Since the price is continues variable, we will use Multiple Linear Regression model and Random Forest model.

Notice:

- In phase1 we said that we will use neural network but after trying we discover that our problem is simple so no need for complex model. Blues, the dataset contains more than 50,000 rows, our laptop computer cannot process NN to this dataset.

II. Data Preprocessing

a) Show what preprocessing techniques you will be using

1. Delete 'Unnamed: 0' this is just an index counter
2. Convert 'cut, color and clarity' variables are categorical data to a numerical form

b) Why did you choose to apply these techniques?

For point a.1 as we said, it just index counter it is not useful in predicting the target. For point a.2, many machine learning algorithms cannot work on label data directly. They require all input variables and output variables to be numeric. This means that categorical variables in our dataset must be converted to a numerical form.

c) Provide examples of preprocessing steps

- Delete ‘Unnamed: 0’ this is just an index counter

```
In [8]: diamonds.drop('Unnamed: 0',axis=1,inplace=True)
```

```
In [9]: diamonds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53940 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   carat    53940 non-null  float64
 1   cut      53940 non-null  object
 2   color    53940 non-null  object
 3   clarity  53940 non-null  object
 4   depth    53940 non-null  float64
 5   table    53940 non-null  float64
 6   price    53940 non-null  int64
 7   x        53940 non-null  float64
 8   y        53940 non-null  float64
 9   z        53940 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 3.9+ MB
```

- Convert ‘cut, color and clarity’ variable are categorical data to a numerical form

```
In [10]: diamonds['cut'].replace(('Fair','Good','Very Good','Premium','Ideal'),(1,2,3,4,5), inplace = True)
```

```
In [11]: diamonds['color'].replace(('D','E','F','G','H','I','J'),(7,6,5,4,3,2,1), inplace = True)
```

```
In [12]: diamonds['clarity'].replace(('I1','SI2','SI1','VS2','VS1','VVS2','VVS1','IF'),(1,2,3,4,5,6,7,8),inplace = True)
```

```
In [13]: diamonds
```

Out[13]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	5	6	2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	4	6	3	59.8	61.0	326	3.89	3.84	2.31
2	0.23	2	6	5	56.9	65.0	327	4.05	4.07	2.31
3	0.29	4	2	4	62.4	58.0	334	4.20	4.23	2.63
4	0.31	2	1	2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
53935	0.72	5	7	3	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	2	7	3	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	3	7	3	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	4	3	2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	5	7	2	62.2	55.0	2757	5.83	5.87	3.64

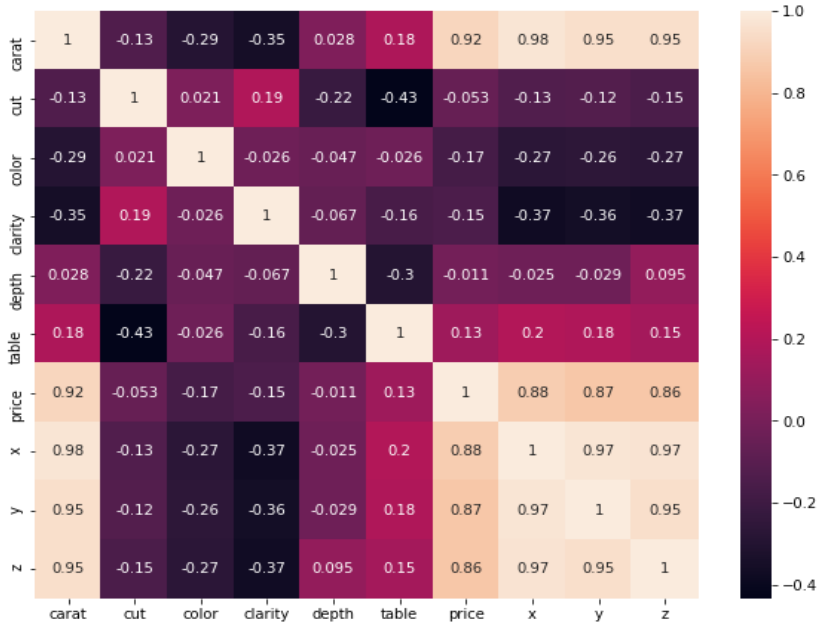
53940 rows × 10 columns

III. Methodology

a) Explain the ML algorithm used

- **Multiple Linear Regression:**

Linear Regression establish the relationships between independent and dependent variables. The dependent variable is always a continuous variable. In our case, dependent variable is the 'Price' and all other variables are consider as Independent variables which they 'cut, color, clarity, depth, table, x, y and z'.
As we can see from correlation matrix most of these variables have a strong impact on price, that is why we use multiple linear regression.



- **Random Forest Regression:**

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be used for both classification and regression problems. Since predicting the price of a diamond from dataset is a regression task, we will use Random Forest as another algorithm.

b) Implementation (with screenshots)

- Spilt the data into X which has all independent variables and y which is the price variable that we want to predict based on independent values. Then we spilt them to training and testing set.

```
In [51]:
# spilt the data
x = diamonds.drop(['price'],1)
y = diamonds['price']
# traning size = 70% and testing size = 30%
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_state=142)
```

First, we make instance of each model and we fit our data on it, then we predict unseen data.

- Linear Regression Model

```
In [62]:
# build a regression model
regr = LinearRegression()
# fit the model to our data
model = regr.fit(X_train, y_train)
# predcit test data
y_reg_pred = model.predict(X_test)
```

- Random Forest Model

```
In [63]:
# build a random forest model
rr = RandomForestRegressor()
# fit the model to our data
rr.fit(X_train,y_train)
# predcit test data
y_rand_pred = rr.predict(X_test)
```

c) What are the initial parameters of the model if any

In random forest model we use default value for the number of trees in the forest which is 100

IV. Evaluation and Results

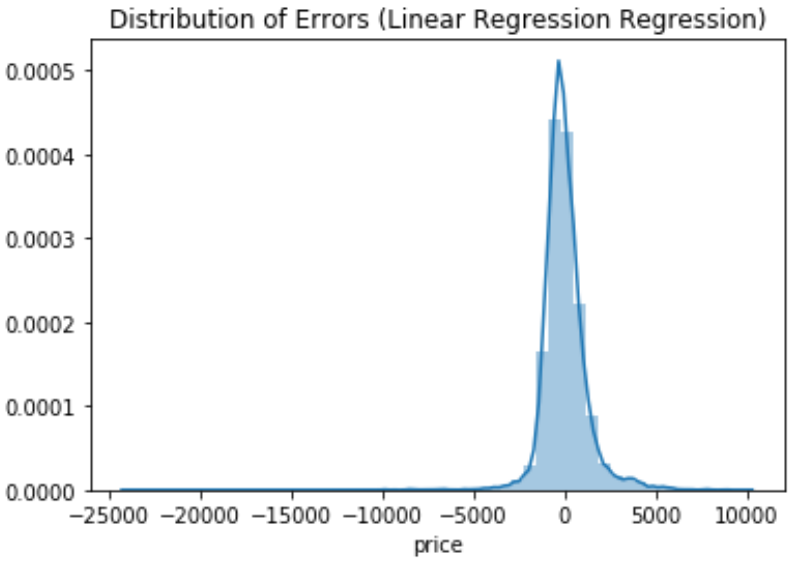
a) Show your results (tables, figure, confusion matrices …etc)

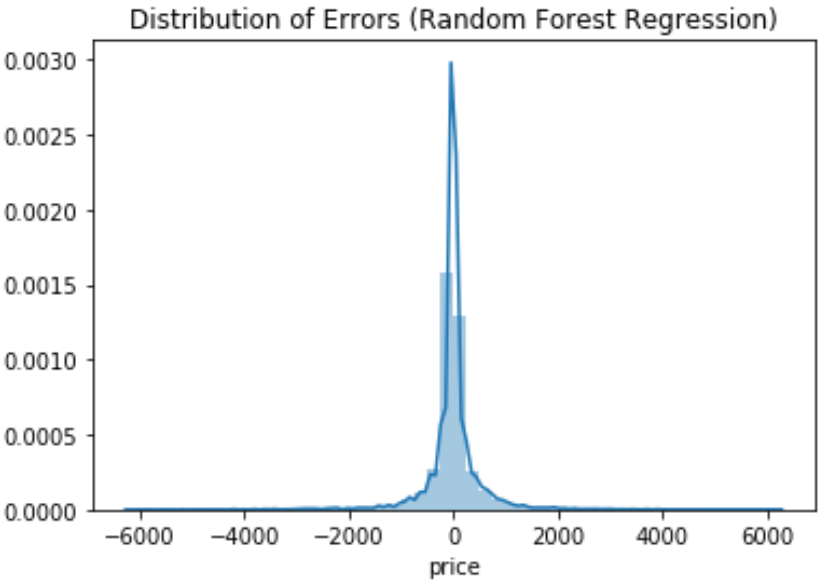
We assess our model performance using Root Mean Squared Error and R Squared.

```
In [20]:
# cheak linear regression model performance
print("RMSE: {} for Linear Regression model ".format(np.sqrt(mean_squared_error((y_test),(y_reg_pred)))))
print("R2  : {} for Linear Regression model".format(np.sqrt(r2_score((y_test),(y_reg_pred)))))
# cheak random forest model performance
print("RMSE: {} for random forest regressor".format(np.sqrt(mean_squared_error((y_test),(y_rand_pred)))))
print("R2  : {} for random forest regressor".format(np.sqrt(r2_score((y_test),(y_rand_pred)))))
```

```
RMSE: 1231.8661058153214 for Linear Regression model
R2  : 0.9516579826277319 for Linear Regression model
RMSE: 565.8279656769693 for random forest regressor
R2  : 0.9899972871229271 for random forest regressor
```

b) Plot your output (visualization of error)





c) Which evaluation techniques is better and why?

As we can see from performance measure, random forest has lowest value for RMES, and highest value of R squared. So, in our case, random forest works better than linear regression in predicting diamond price prediction.

V. Analysis and conclusion

a) Explain your results in analytical way instead of numbers. Meaning, why do you think one algorithm performed that way and the other didn't?

In conclude, we think that random forest work well than linear regression that is because we have a large dataset. Also, a linear model can obtain values outside the training set during prediction, as we can see from distribution of Errors (Linear Regression) plot the range of value is too large than training data. While there are no values outside the range in random forest model for the price variable. Moreover, A prediction from the random forest regression is an average of the predictions produced by the trees in the forest which makes it more accurate. In other hand, if we care about time we would accept linear regression model since random forest take a little bit more time to processing than linear regression.