



Diamonds Price Prediction

- *To predict the diamond price within 15% difference**
- *Predicting price based on the required features of a diamond, for the purpose of budgeting or price references*

** the difference (in percentage) between the actual and the predicted price*



Contents

1. Data Characteristics
2. Exploratory Data Analysis (EDA) & Data Preparation
3. Features Selection
4. Models
5. Results
6. Feature Importance
7. Conclusion



Data Characteristics



Data Source

Kaggle

- More than 53,000 data on the measurements and price of diamonds
- Diamond data:
 - ❖ Price
 - ❖ 4Cs
 - ❖ Proportion and Measurements
- Assumption:
 - ❖ Price = Sale price



4Cs

- Carat
- Color
- Cut
- Clarity



Carat

- Weight of the diamond

Color

- Color of the diamond with D (best) and Z (worst)
- In the data, D to J.



Source: Petragems

Cut

- Cut quality of the diamond
- GIA classify as Excellent, Very Good, Good, Fair and Poor
- In the data, Ideal (best), Premium, Very Good, Good and Fair (worst)



Clarity

- Visible inclusions under 10X magnification

Symbol	Name
FL	Flawless
IF	Internally Flawless
VVS1/VVS2	Very, Very Slightly Included
VS1/VS2	Very Slightly Included
SI1/SI2	Slightly Included
I1/I2/I3	Included



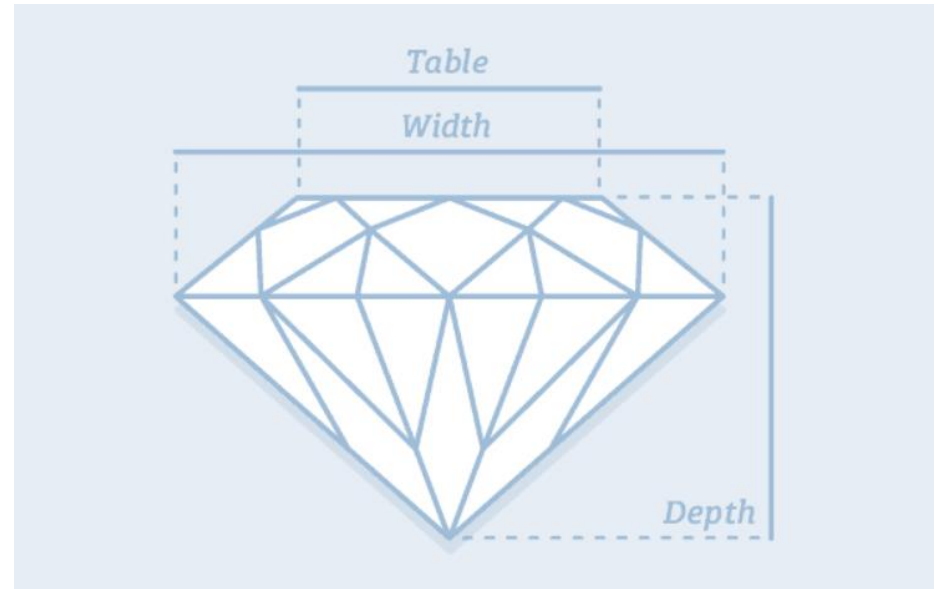
Proportion

Depth %

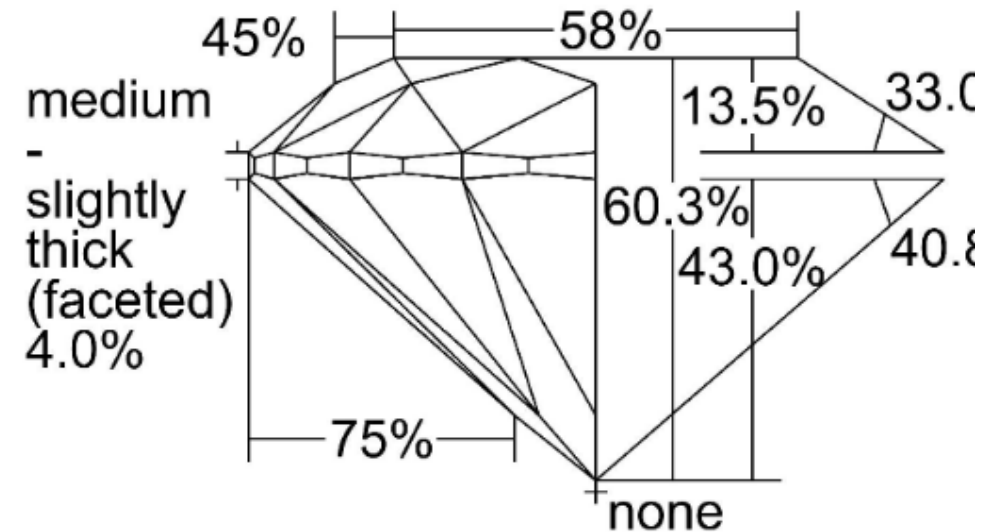
- Measurement from top to bottom
- Total height divided by total width (e.g. 60.3%)

Table %

- Flat facet on diamond's surface
- Table width as a percentage of the total width (e.g. 58%)



Source: diamonds.pro



Source: GIA

Measurements

“X”

➤ Length in mm


“Y”

➤ Width in mm

“Z”

➤ Depth in mm





Exploratory Data Analysis (EDA) & Data Preparation

Looking at the data....

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 53940 entries, 0 to 53939  
Data columns (total 11 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   Unnamed: 0   53940 non-null  int64  
1   carat        53940 non-null  float64  
2   cut          53940 non-null  object  
3   color        53940 non-null  object  
4   clarity      53940 non-null  object  
5   depth        53940 non-null  float64  
6   table        53940 non-null  float64  
7   price        53940 non-null  int64  
8   x            53940 non-null  float64  
9   y            53940 non-null  float64  
10  z            53940 non-null  float64  
dtypes: float64(6), int64(2), object(3)  
memory usage: 4.5+ MB
```

```
df.head(10)
```

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

```
df_na = df.isna()  
df_na.sum()
```

```
carat    0  
cut       0  
color    0  
clarity  0  
depth    0  
table    0  
price    0  
x         0  
y         0  
z         0  
dtype: int64
```

```
df_cat = df.select_dtypes(include=['object'])
```

```
df_cat.nunique()
```

```
cut      5  
color    7  
clarity  8  
dtype: int64
```

```
print(set(df['cut']))  
print(set(df['color']))  
print(set(df['clarity']))
```

```
{'Fair', 'Very Good', 'Ideal', 'Good', 'Premium'}  
{'E', 'G', 'F', 'I', 'D', 'J', 'H'}  
{'VVS2', 'SI2', 'VS2', 'SI1', 'IF', 'I1', 'VVS1', 'VS1'}
```


- Dropping columns
- Replacing the labels/objects into numerical form

```
df = df.drop(columns='Unnamed: 0')

replace = {
    'cut':{
        'Fair': 0,
        'Good': 1,
        'Very Good': 2,
        'Ideal': 3,
        'Premium': 4},
    'color':{
        'J': 0,
        'I': 1,
        'H': 2,
        'G': 3,
        'F': 4,
        'E': 5,
        'D': 6},
    'clarity':{
        'I1':0,
        'SI2':1,
        'SI1':2,
        'VS2':3,
        'VS1':4,
        'VVS2':5,
        'VVS1':6,
        'IF':7}
}

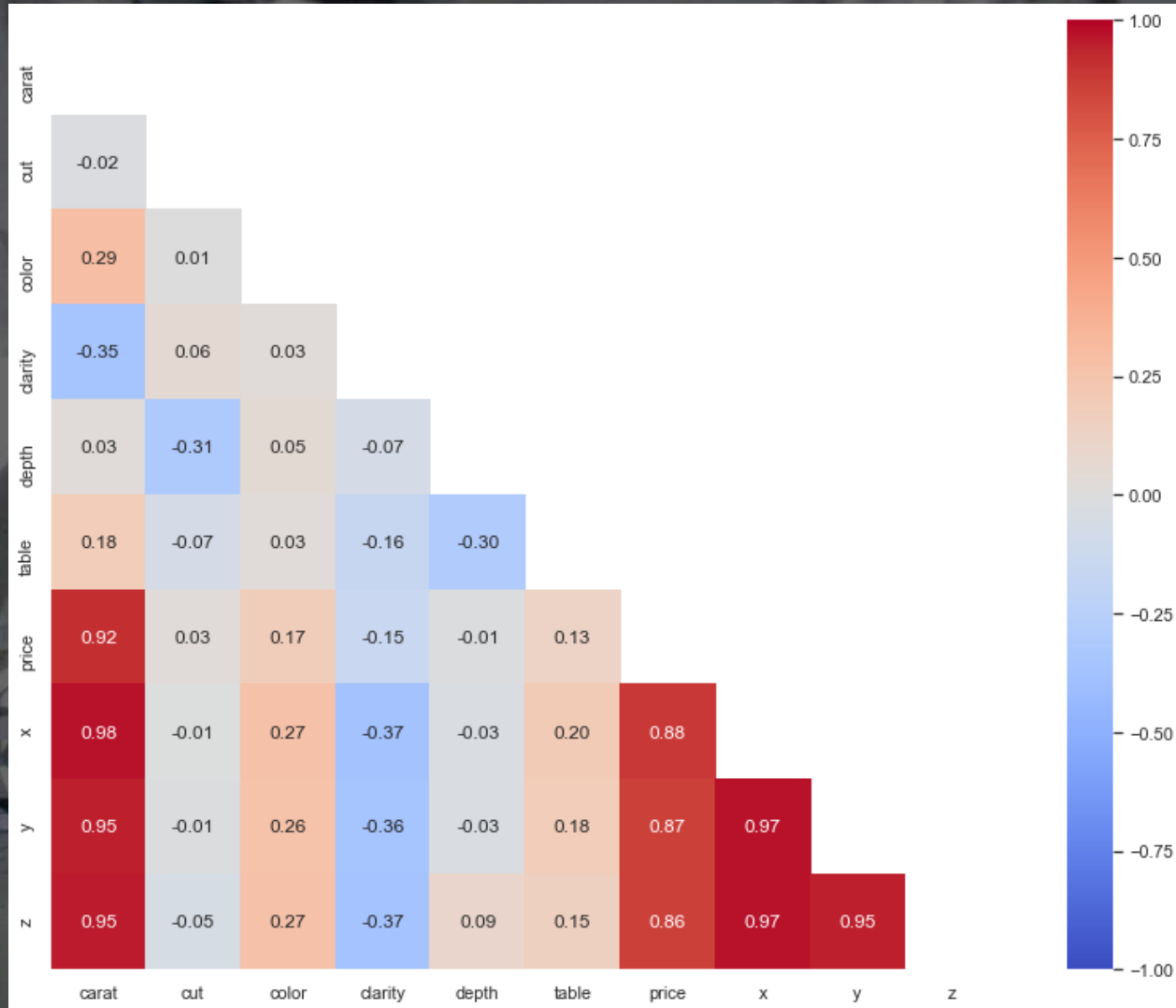
df.replace(replace,inplace=True)
```



Features Selection



Correlation Map



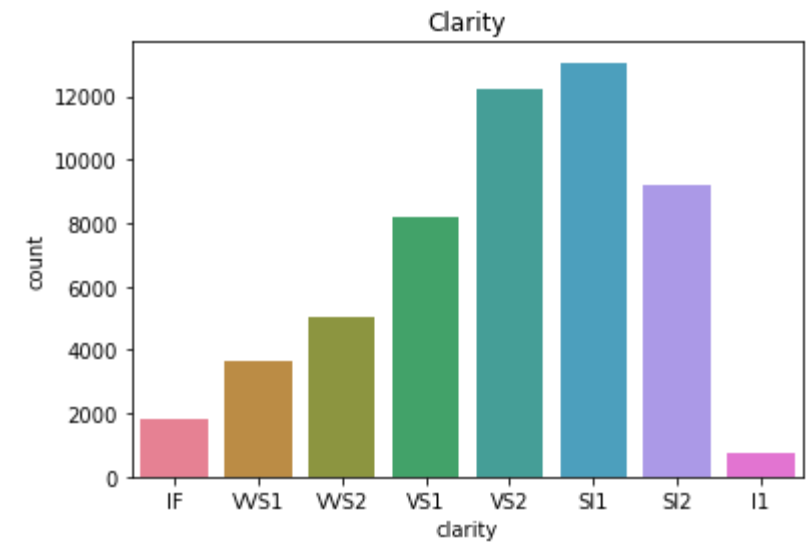
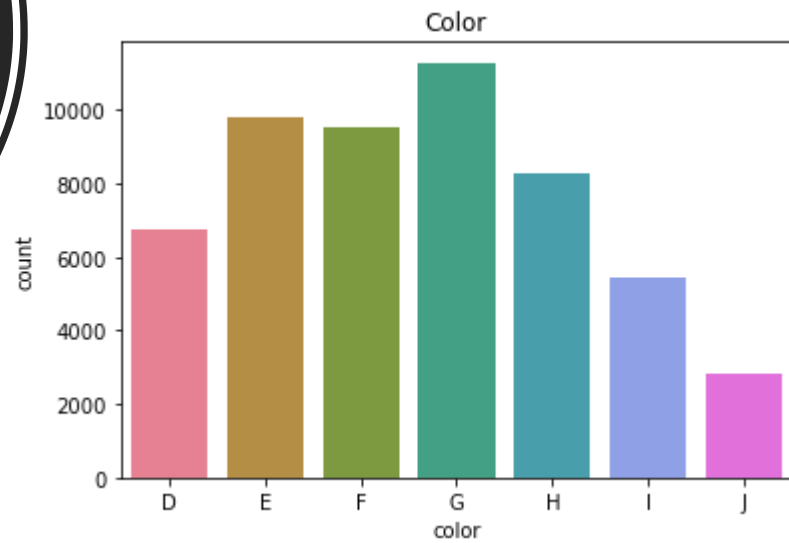
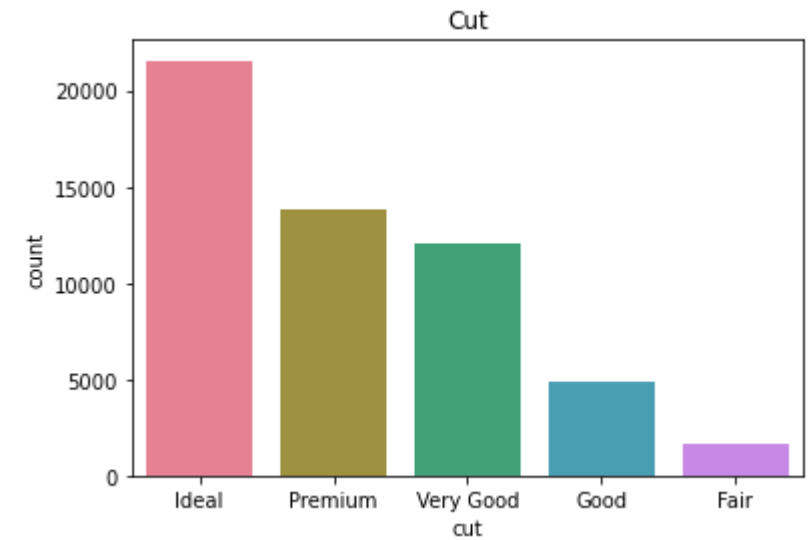
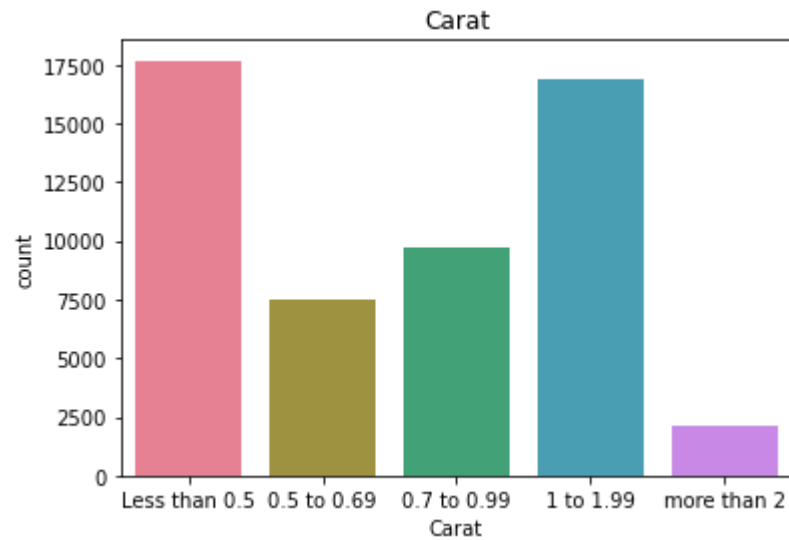
Features Selected

1. Carat
2. Cut
3. Color
4. Clarity
5. Price

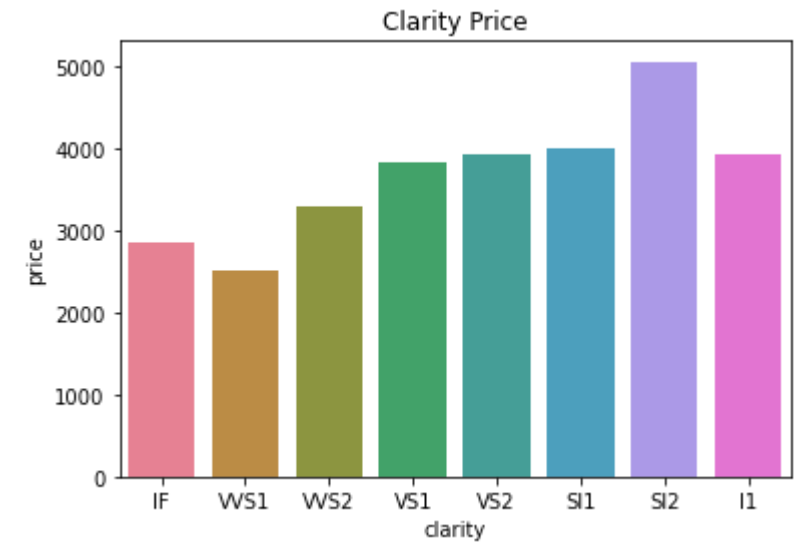
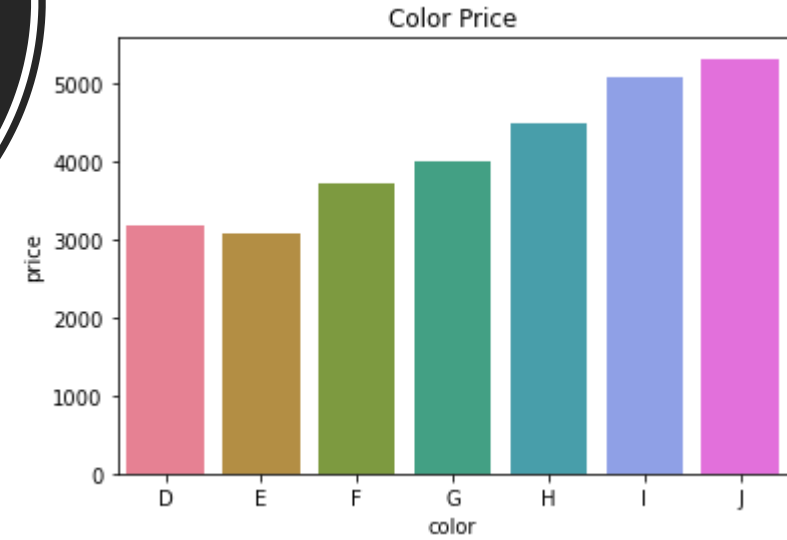
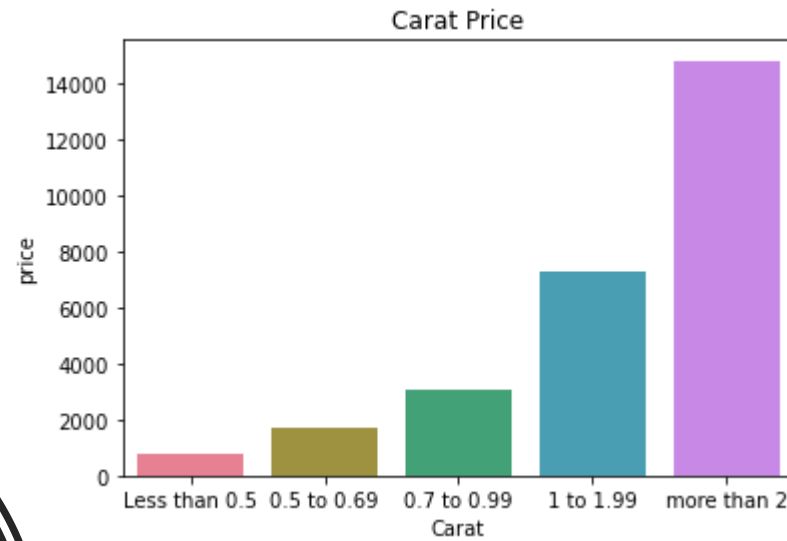
Depth, table, x, y and z are all measurements of a diamond. Thus, it is more relevant to drop these features and use only carat as a measurement feature.

Furthermore, the 5 features are more commonly used when selecting for a diamond.

Visualizing the features



Visualizing the features



Models

- Simple Linear Regression
- Multiple Linear Regression
- Random Forest Regressor



Simple Linear Regression

```
x = df['carat'].values.reshape(-1,1)  
y = df['price']
```

	Desired Output (Actuals)	Predicted Output	Difference in %
1388	559	-397.373228	171.086445
50052	2201	2244.056339	-1.956217
41645	1238	845.652450	31.692048
42377	1304	1078.719765	17.276092
17244	6901	9779.899514	-41.717135
...
44081	1554	1622.543500	-4.410779
23713	633	-319.684123	150.503021
31375	761	146.450506	80.755518
21772	9836	7293.848157	25.845383
4998	3742	6128.511584	-63.776365

10788 rows × 3 columns

Multiple Linear Regression

```
x = df.drop(['price', 'depth', 'table', 'x', 'y', 'z'], axis = 1)
y = df['price']
```

Independent features (X):

- Carat
- Cut
- Color
- Clarity

	Desired Output (Actuals)	Predicted Output	Difference in %
1388	559	527.926318	5.558798
50052	2201	3153.126187	-43.258800
41645	1238	2038.421855	-64.654431
42377	1304	2443.338768	-87.372605
17244	6901	9976.726450	-44.569286
...
44081	1554	2236.840080	-43.940803
23713	633	259.097402	59.068341
31375	761	858.164438	-12.767995
21772	9836	8458.572494	14.003940
4998	3742	3961.113921	-5.855530

10788 rows × 3 columns

Random Forest Regressor

```
x = df.drop(['price', 'depth', 'table', 'x', 'y', 'z'], axis = 1)
y = df['price']
```

Independent features (X):

- Carat
- Cut
- Color
- Clarity

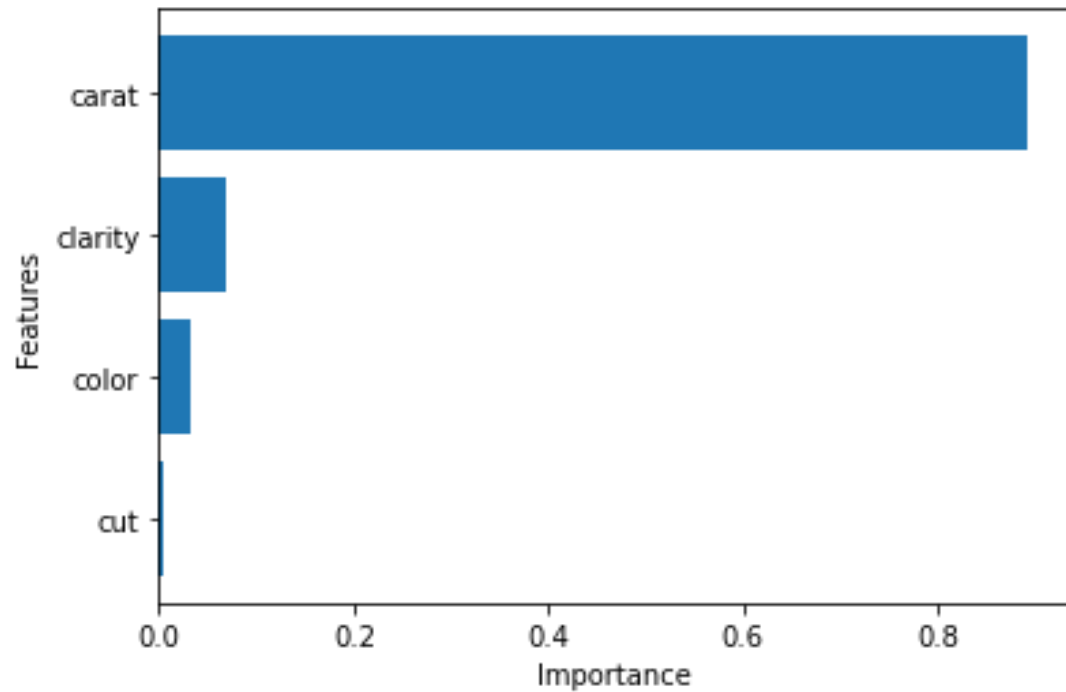
	Desired Output (Actuals)	Predicted Output	Difference in %
1388	559	560.190000	-0.212880
50052	2201	2181.063169	0.905808
41645	1238	1131.364077	8.613564
42377	1304	1201.474634	7.862375
17244	6901	9366.556500	-35.727525
...
44081	1554	1714.377580	-10.320308
23713	633	584.439897	7.671422
31375	761	810.796186	-6.543520
21772	9836	9189.585286	6.571927
4998	3742	3584.870000	4.199091
10788 rows × 3 columns			

Results

	Simple Linear Regression	Multiple Linear Regression	Random Forest Regressor
Mean Absolute Error (MAE)	1009.50	855.90	297.20
Training score	0.85	0.90	0.99
Test score	0.85	0.90	0.98
Percentage of difference below 15%	28.05%	34.96%	85.31%
Predicted Price (SGD) <ul style="list-style-type: none">▪ Carat: 1▪ Cut: Premium▪ Color: E▪ Clarity: VVS2	7,324.31	9,898.01	11,905.13

Feature Importance

Diamond Price Prediction
Feature Importance



Conclusion

- *Random Forest Regressor provides the highest test score and highest percentage of test data having difference of less than 15%*
- *Although carat is the most important feature of the diamond price, we cannot solely based on carat to predict the diamond price (as evident in simple linear regression model)*





References

Data Source:

- <https://www.kaggle.com/shivam2503/diamonds>

Diamond information:

- <https://www.gia.edu/report-check?reportno=6245697081>
- <https://4cs.gia.edu/en-us/diamond-clarity/>
- <https://www.diamonds.pro/education/diamond-depth-and-table/>
- <https://www.petrageoms.com/education/diamond-color/>