Diamond price prediction

Eng. Ahmed Saeed

Data Extraction 1

Data Cleaning 2

Domain Knowledge 3

Data Analysis 4

Machine Learning 5

Model Deployments 6

Data Extraction 1

```python
import pandas as pd
import seaborn as sns
import joblib
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```

Data Extraction 1

Using API class python
Load dataset using pandas

pandas

```python
class Diamond:
    """Diamon price prediction class"""
    def __init__(self, path):
        """Load DataFrame"""
        self.df = pd.read_csv(path)
```

Data Extraction 1

```
df = pd.read_csv("diamonds.csv")
df.head(15)
```

pandas

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39
10	0.30	Good	J	SI1	64.0	55.0	339	4.25	4.28	2.73
11	0.23	Ideal	J	VS1	62.8	56.0	340	3.93	3.90	2.46
12	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33
13	0.31	Ideal	J	SI2	62.2	54.0	344	4.35	4.37	2.71
14	0.20	Premium	E	SI2	60.2	62.0	345	3.79	3.75	2.27

Data Extraction 1

How much unique categorical data?

```
df["cut"].unique()
```

```
array(['Ideal', 'Premium', 'Good', 'Very Good', 'Fair'], dtype=object)
```

```
df["clarity"].unique()
```

```
array(['SI2', 'SI1', 'VS1', 'VS2', 'VVS2', 'VVS1', 'I1', 'IF'],
      dtype=object)
```

```
df["color"].unique()
```

```
array(['E', 'I', 'J', 'H', 'F', 'G', 'D'], dtype=object)
```

Data Extraction 1

Dataset size

```
print("Original shape is 53940 X 10")
df.shape
```

```
Original shape is 53940 X 10

(53940, 10)
```

Data Cleaning 2

Dataset information

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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: 4.1+ MB
```

Data Cleaning

2

Is there null data?

```
df.isnull().sum()
```

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

Data Cleaning

2

Is there null data?

```
df.describe()
```

	carat	depth	table	price	x	\
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	
std	0.474011	1.432621	2.234491	3989.439738	1.121761	
min	0.200000	43.000000	43.000000	326.000000	0.000000	
25%	0.400000	61.000000	56.000000	950.000000	4.710000	
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	
max	5.010000	79.000000	95.000000	18823.000000	10.740000	

	y	z
count	53940.000000	53940.000000
mean	5.734526	3.538734
std	1.142135	0.705699
min	0.000000	0.000000
25%	4.720000	2.910000
50%	5.710000	3.530000
75%	6.540000	4.040000
max	58.900000	31.800000

Domain Knowledge 3

How to calculate the price of the diamonds?

Diamond the 4C`s.

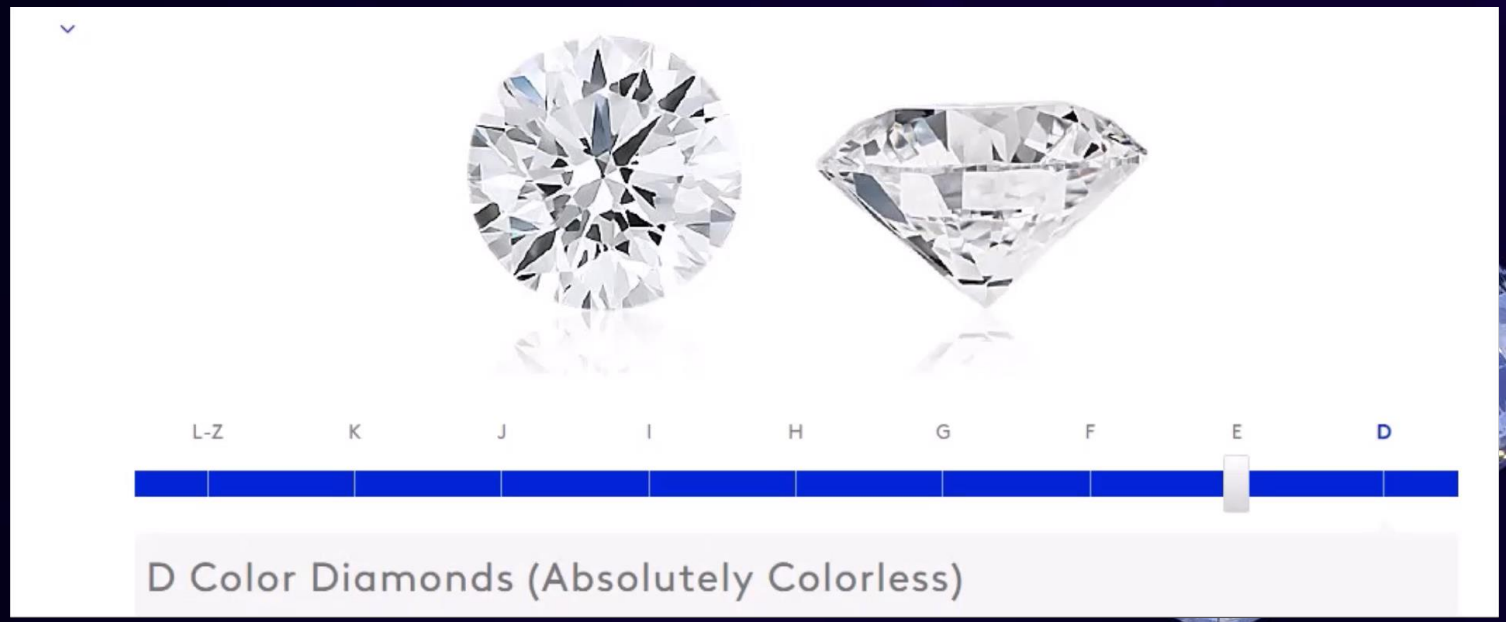
GIA
Gemological Institute of America

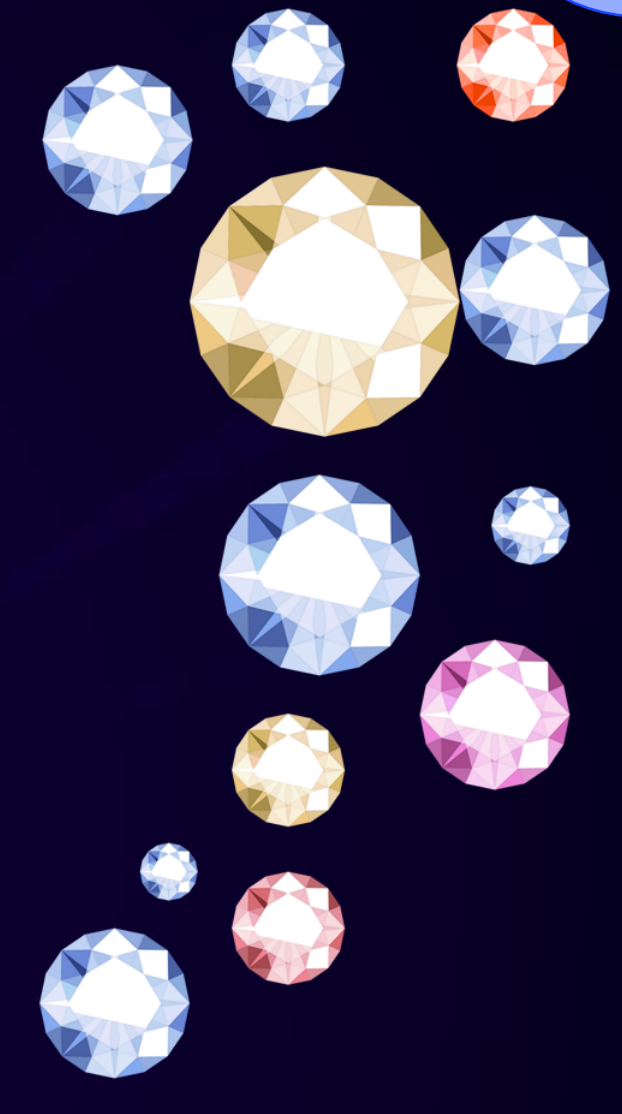
1- Color
2- Clarity
3- Cut grade
4- Carat weight

These are the most important features.

1- Color

GIA’s D-to-Z diamond color-grading system measures the degree of colorlessness, many of these diamond color distinctions are so subtle that they are invisible to the untrained eye and make a very big difference in diamond quality and price.

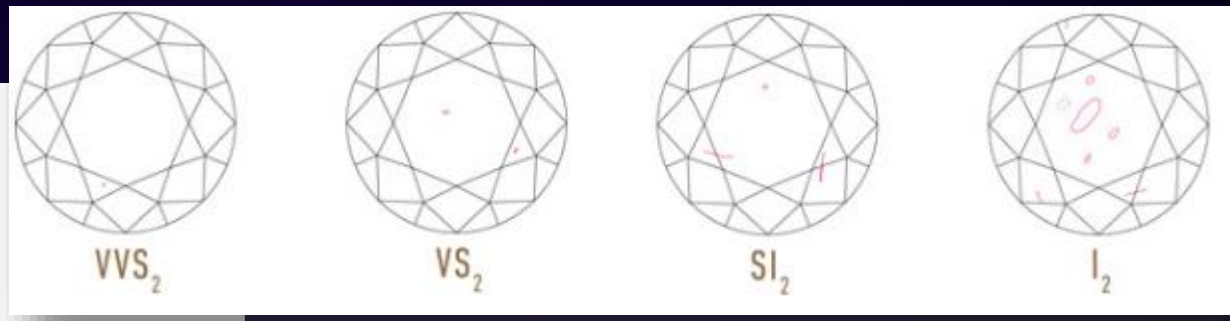


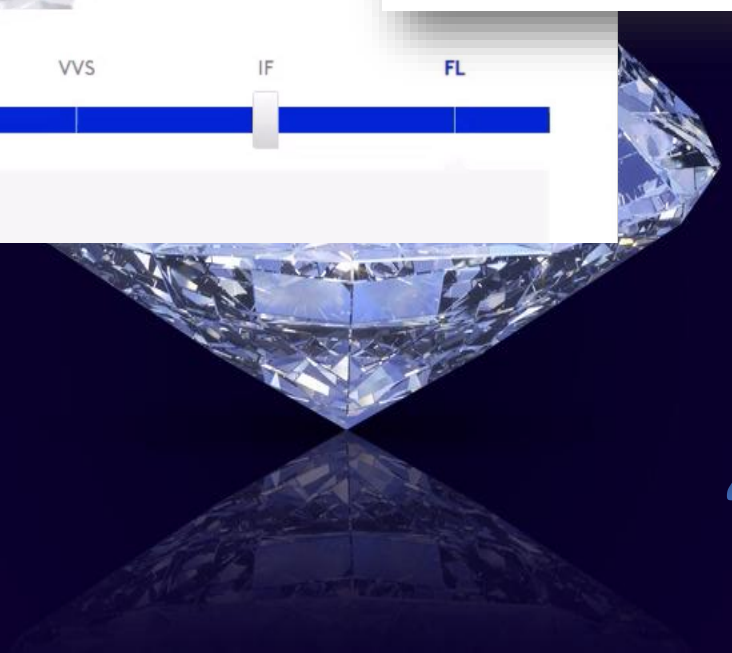


2- Clarity

Domain Knowledge 3

Diamond Clarity Refers to the Absence of internal Inclusions and external Blemishes.
Knowing what diamond clarity truly means helps you understand the factors that contribute to diamond quality and price.



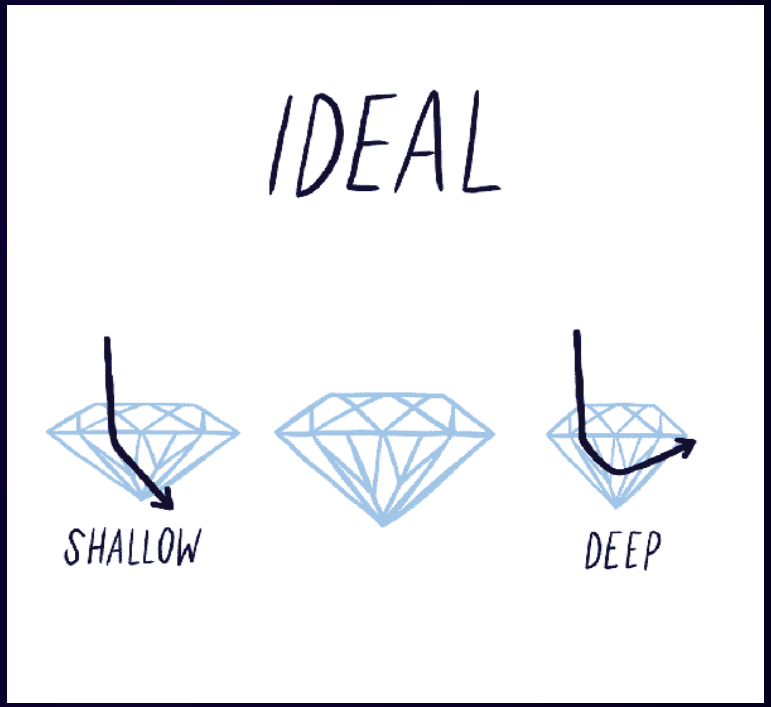


"IF" is rarely found

3- Cut grade

Domain Knowledge 3

Diamond Cut does not refer to a diamond's shape, but what diamond cut actually does mean how well a diamond's facets interact with light and the quality of workmanship.



A well-cut diamond will direct more light through the crown.

4- Carat

Diamond carat weight measures how much a diamond weighs, and two diamonds of equal carat weight can have very different values (and prices) depending on three other factors of the diamond Color, Clarity, and Cut.

Data analysis

4

Correlations

```
print(df.corr())
sns.heatmap(df.corr(), annot=True)
```

	carat	depth	table	price	x	y	z
carat	1.000000	0.028224	0.181618	0.921591	0.975094	0.951722	0.953387
depth	0.028224	1.000000	-0.295779	-0.010647	-0.025289	-0.029341	0.094924
table	0.181618	-0.295779	1.000000	0.127134	0.195344	0.183760	0.150929
price	0.921591	-0.010647	0.127134	1.000000	0.884435	0.865421	0.861249
x	0.975094	-0.025289	0.195344	0.884435	1.000000	0.974701	0.970772
y	0.951722	-0.029341	0.183760	0.865421	0.974701	1.000000	0.952006
z	0.953387	0.094924	0.150929	0.861249	0.970772	0.952006	1.000000

```
<matplotlib.axes._subplots.AxesSubplot at 0x14d6e005208>
```

Missing 3 important features because they are categorical

Data analysis

4

Correlations using heat-map

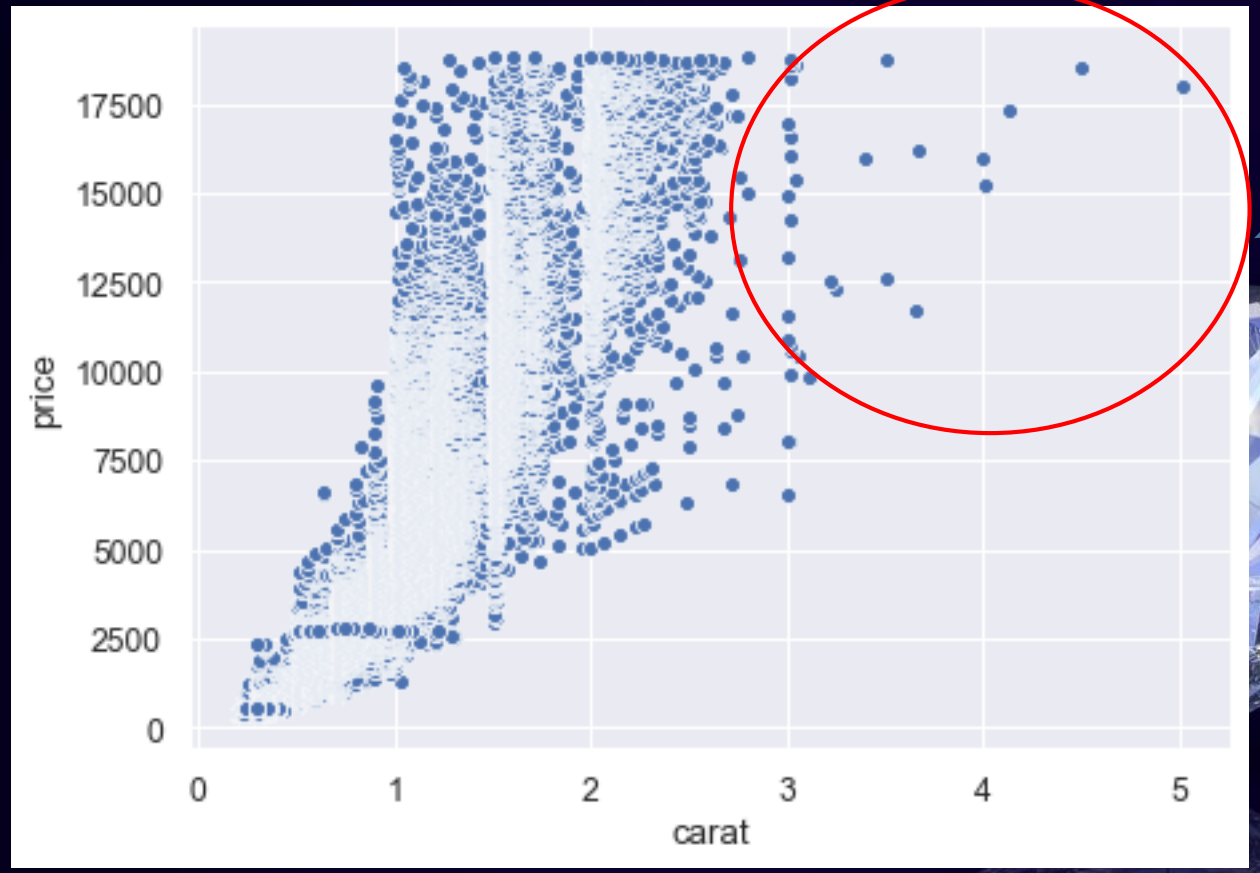
Visualize nominal data

Data analysis

4

Relation between carat and price.

```
sns.scatterplot(x=self.df["carat"], y = self.df["price"])
```



Outliers?

Data analysis

4

Decide to get rid of outlier(s).

```
df[df["carat"] > 3]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
19339	3.01	Premium	I	I1	62.7	58.0	8040	9.10	8.97	5.67
21758	3.11	Fair	J	I1	65.9	57.0	9823	9.15	9.02	5.98
21862	3.01	Premium	F	I1	62.2	56.0	9925	9.24	9.13	5.73
22428	3.05	Premium	E	I1	60.9	58.0	10453	9.26	9.25	5.66
22540	3.02	Fair	I	I1	65.2	56.0	10577	9.11	9.02	5.91
22741	3.01	Fair	H	I1	56.1	62.0	10761	9.54	9.38	5.31
23644	3.65	Fair	H	I1	67.1	53.0	11668	9.53	9.48	6.38
24131	3.24	Premium	H	I1	62.1	58.0	12300	9.44	9.40	5.85
24297	3.22	Ideal	I	I1	62.6	55.0	12545	9.49	9.42	5.92
24328	3.50	Ideal	H	I1	62.8	57.0	12587	9.65	9.59	6.03
25460	3.01	Premium	G	SI2	59.8	58.0	14220	9.44	9.37	5.62
25998	4.01	Premium	I	I1	61.0	61.0	15223	10.14	10.10	6.17
25999	4.01	Premium	J	I1	62.5	62.0	15223	10.02	9.94	6.24
26100	3.04	Very Good	I	SI2	63.2	59.0	15354	9.14	9.07	5.75
26431	3.40	Fair	D	I1	66.8	52.0	15964	9.42	9.34	6.27
26444	4.00	Very Good	I	I1	63.3	58.0	15984	10.01	9.94	6.31
26467	3.01	Ideal	J	SI2	61.7	58.0	16037	9.25	9.20	5.69

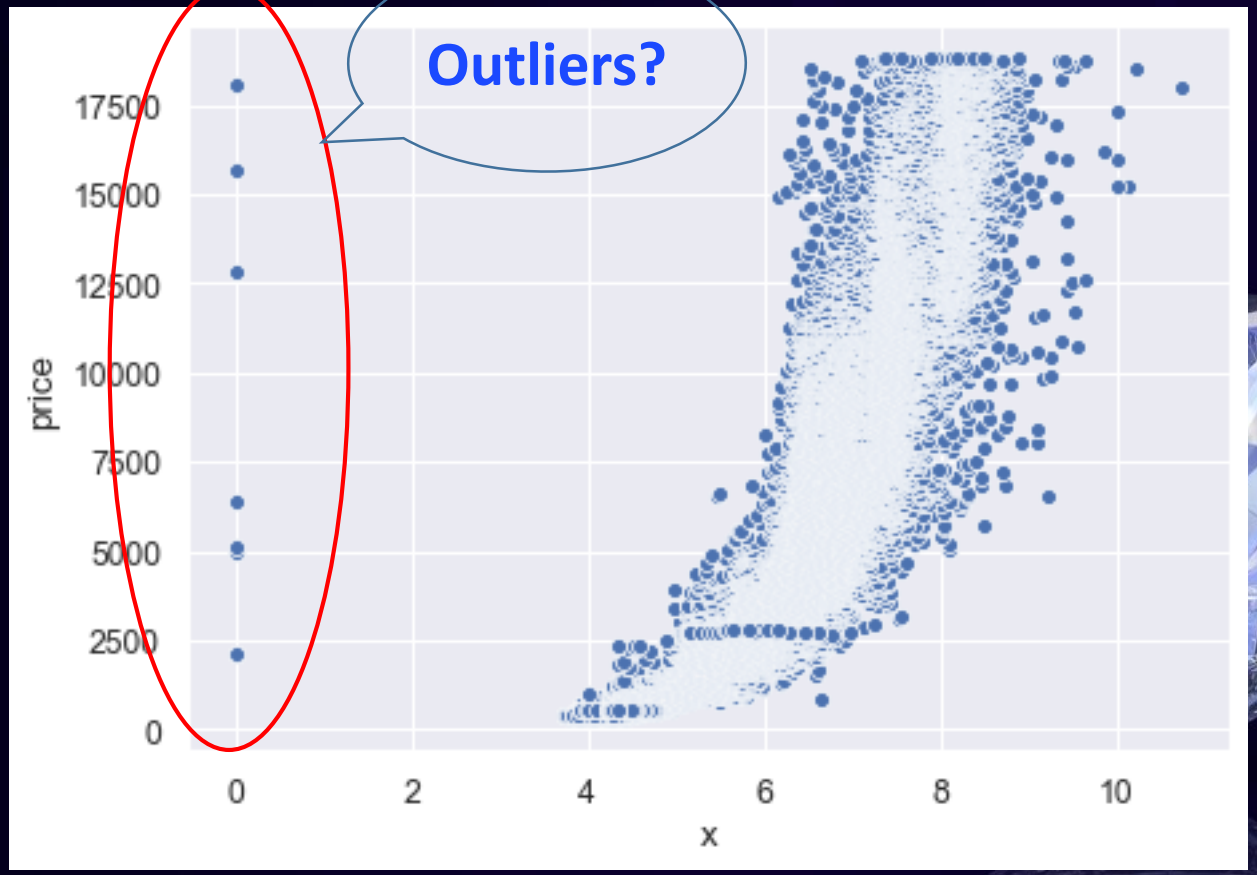
Visualize nominal data

Data analysis

4

Relation between X and price.

```
sns.scatterplot(x=self.df["x"], y = self.df["price"])
```



Data analysis

4

Decide to get rid of outlier(s).

```
df[df["x"] < 2 ]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
11182	1.07	Ideal	F	SI2	61.6	56.0	4954	0.0	6.62	0.0
11963	1.00	Very Good	H	VS2	63.3	53.0	5139	0.0	0.00	0.0
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.0	0.00	0.0
24520	1.56	Ideal	G	VS2	62.2	54.0	12800	0.0	0.00	0.0
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.0	0.00	0.0
27429	2.25	Premium	H	SI2	62.8	59.0	18034	0.0	0.00	0.0
49556	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.00	0.0
49557	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.00	0.0

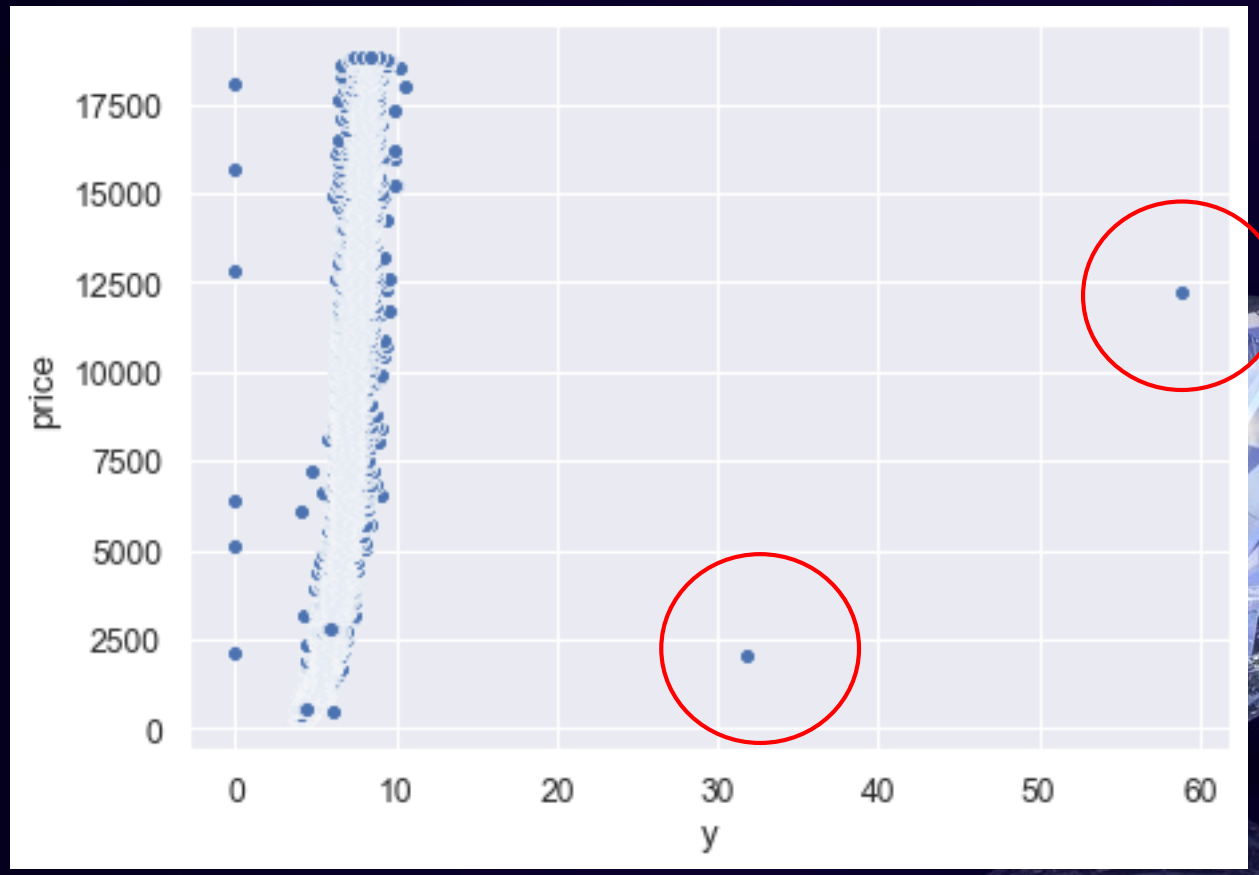
Visualize nominal data

Data analysis

4

Relation between Y and price.

```
sns.scatterplot(x=self.df["y"], y = self.df["price"])
```



Outliers?

Data analysis

4

Decide to get rid of outlier(s).

```
df[df["y"] > 30 ]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
24067	2.00	Premium	H	SI2	58.9	57.0	12210	8.09	58.9	8.06
49189	0.51	Ideal	E	VS1	61.8	55.0	2075	5.15	31.8	5.12

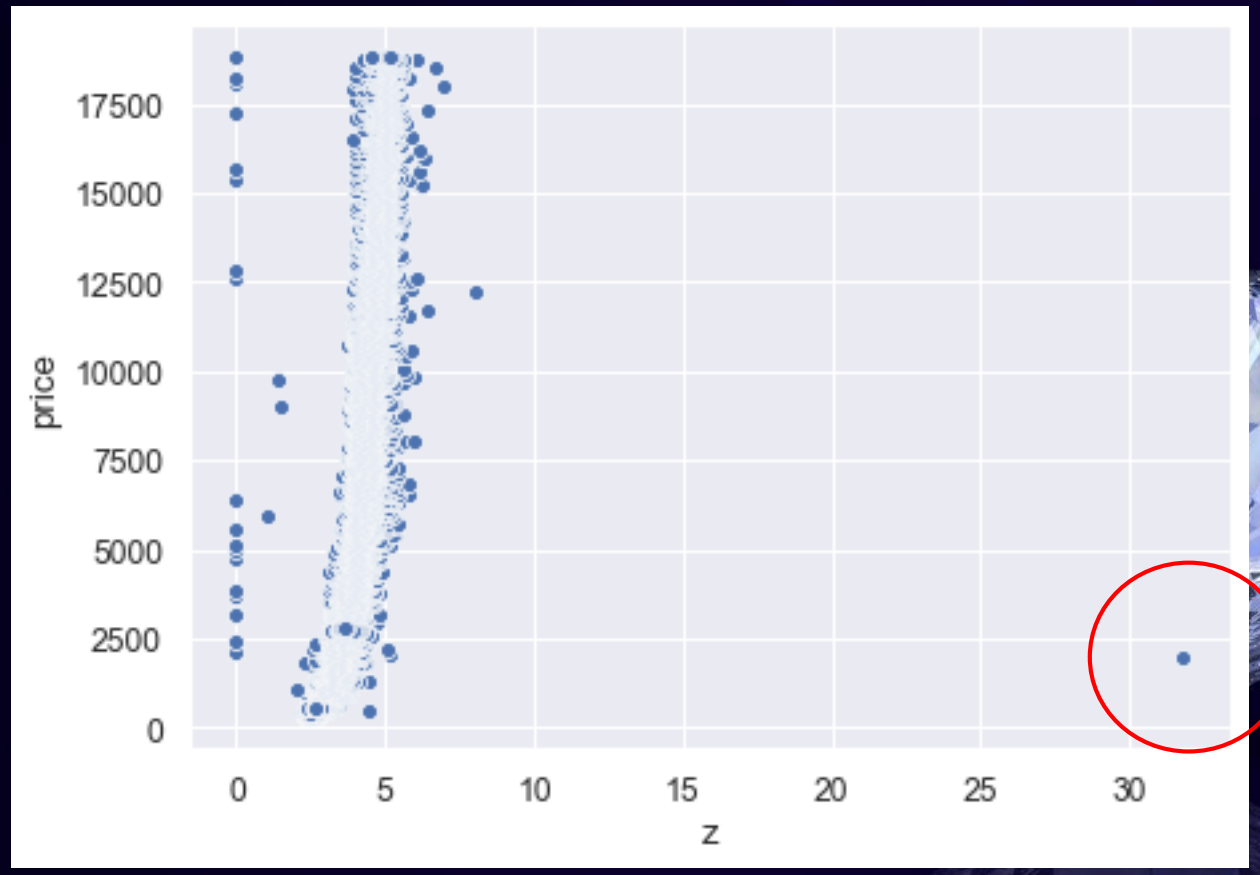
Visualize nominal data

Data analysis

4

Relation between Z and price.

```
sns.scatterplot(x=self.df["z"], y = self.df["price"])
```



Outliers?

Data analysis

4

Decide to get rid of outlier(s).

```
df[df["z"] > 30 ]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
48410	0.51	Very Good	E	VS1	61.8	54.7	1970	5.12	5.15	31.8

Data analysis

4

Get outliers indexes.

```
carat_index = list(df[df["carat"] >= 3].index)
x_index = list(df[df["x"] < 2 ].index)
y_index = list(df[df["y"] > 30 ].index)
z_index = list(df[df["z"] > 30 ].index)
depth_index = list(df[(df["depth"] > 75) | (df["depth"] < 45 )].index)
table_index = list(df[df["table"] > 90].index)

indexes = carat_index + x_index + y_index + z_index + depth_index + table_index
len(indexes)
```

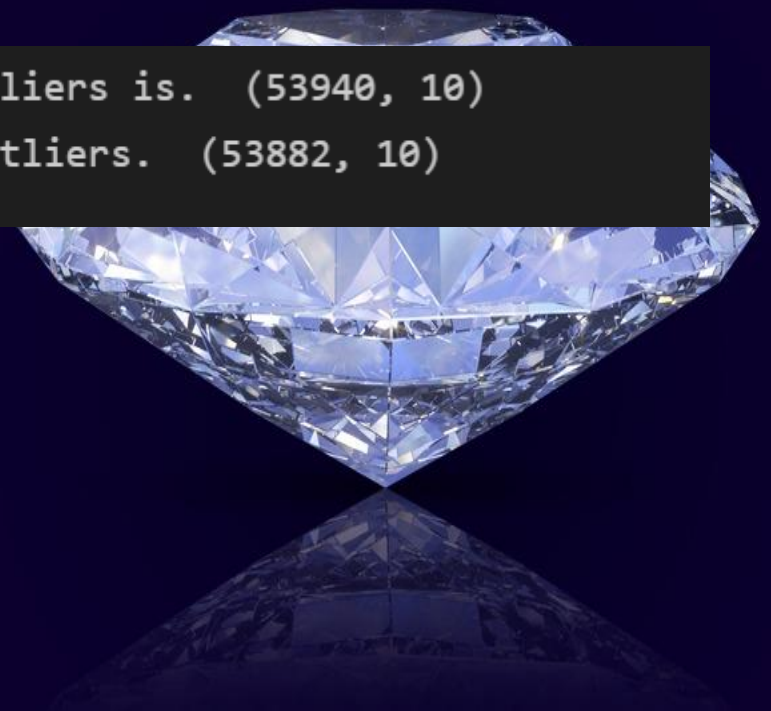
```
58
```

Data analysis 4

Delete outliers using their indexes

```
def delete_outliers(self,indexes):
    """Delete outliers by knowing their indexes"""
    for outlier in indexes:
        self.df.drop(outlier, inplace=True)
```

```
DataFrame shape before delete outliers is.  (53940, 10)
DataFrame shape after deleting outliers.  (53882, 10)
```

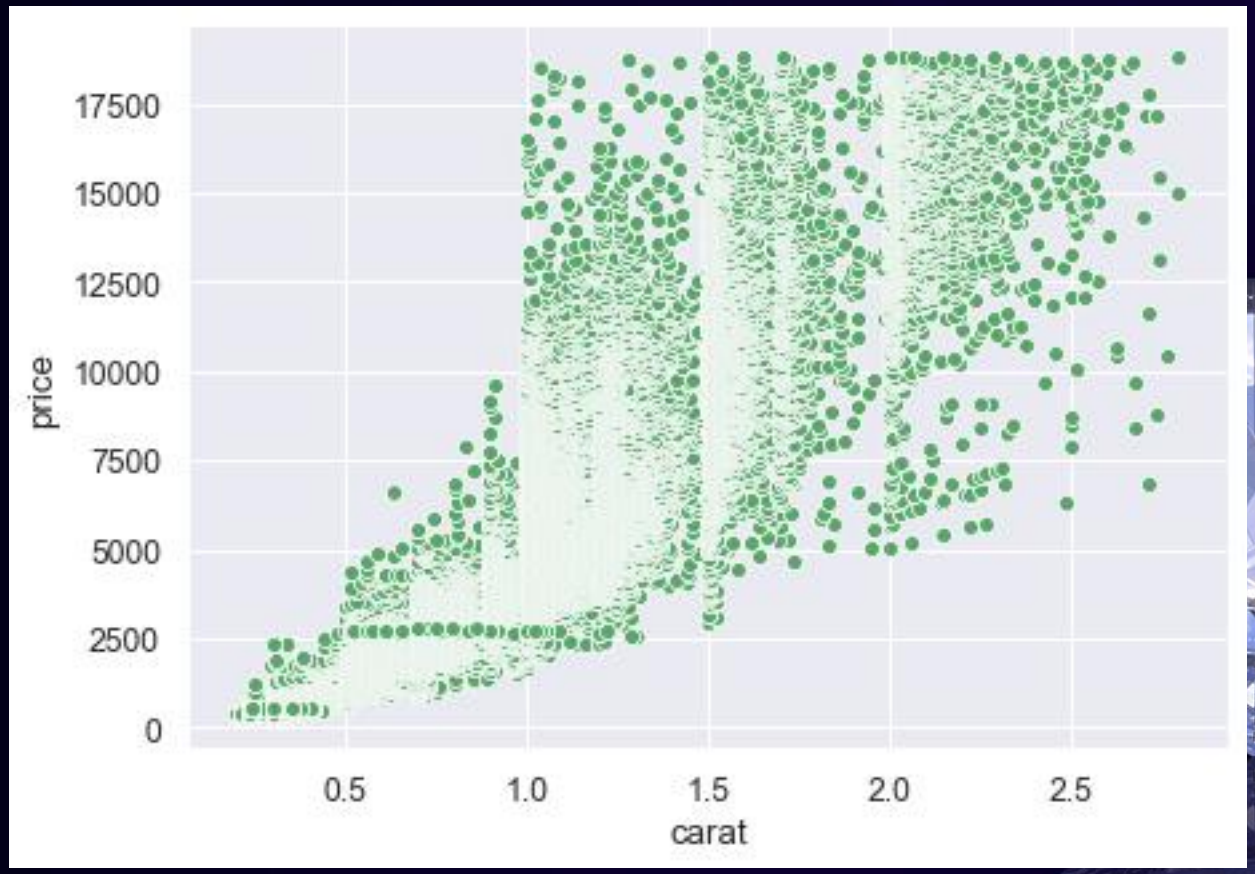


Visualize nominal data

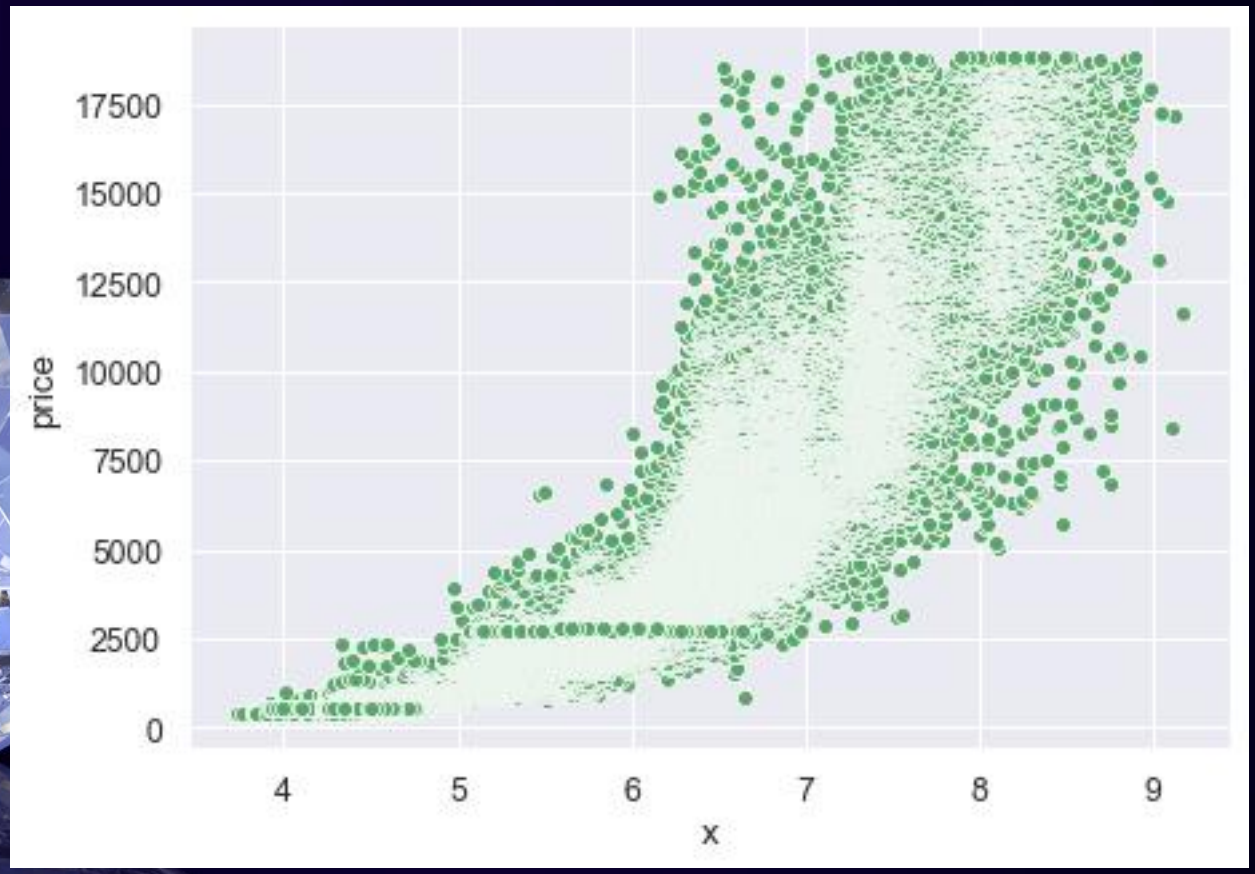
Data analysis

4

```
sns.scatterplot(x=self.df["carat"], y = self.df["price"])
```




```
sns.scatterplot(x=self.df["x"], y = self.df["price"])
```

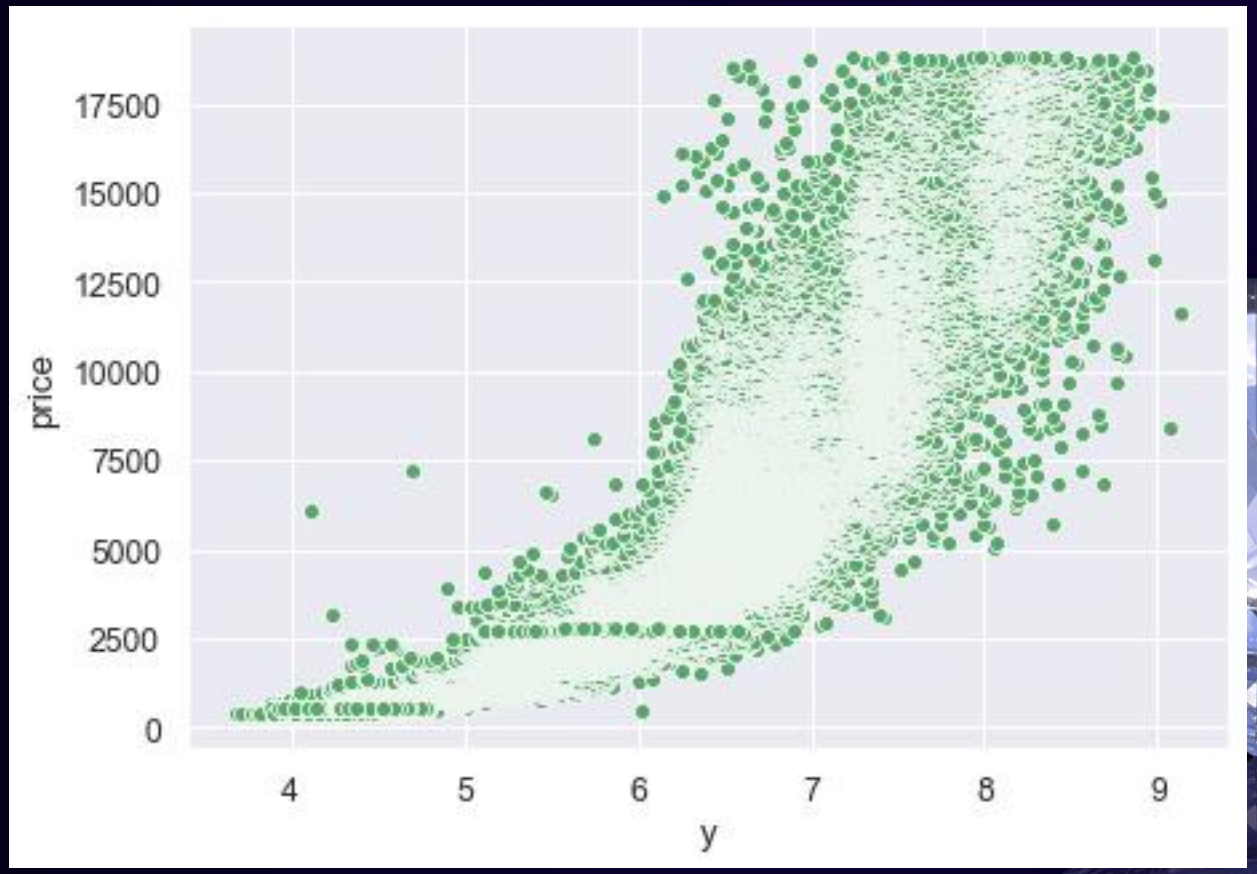


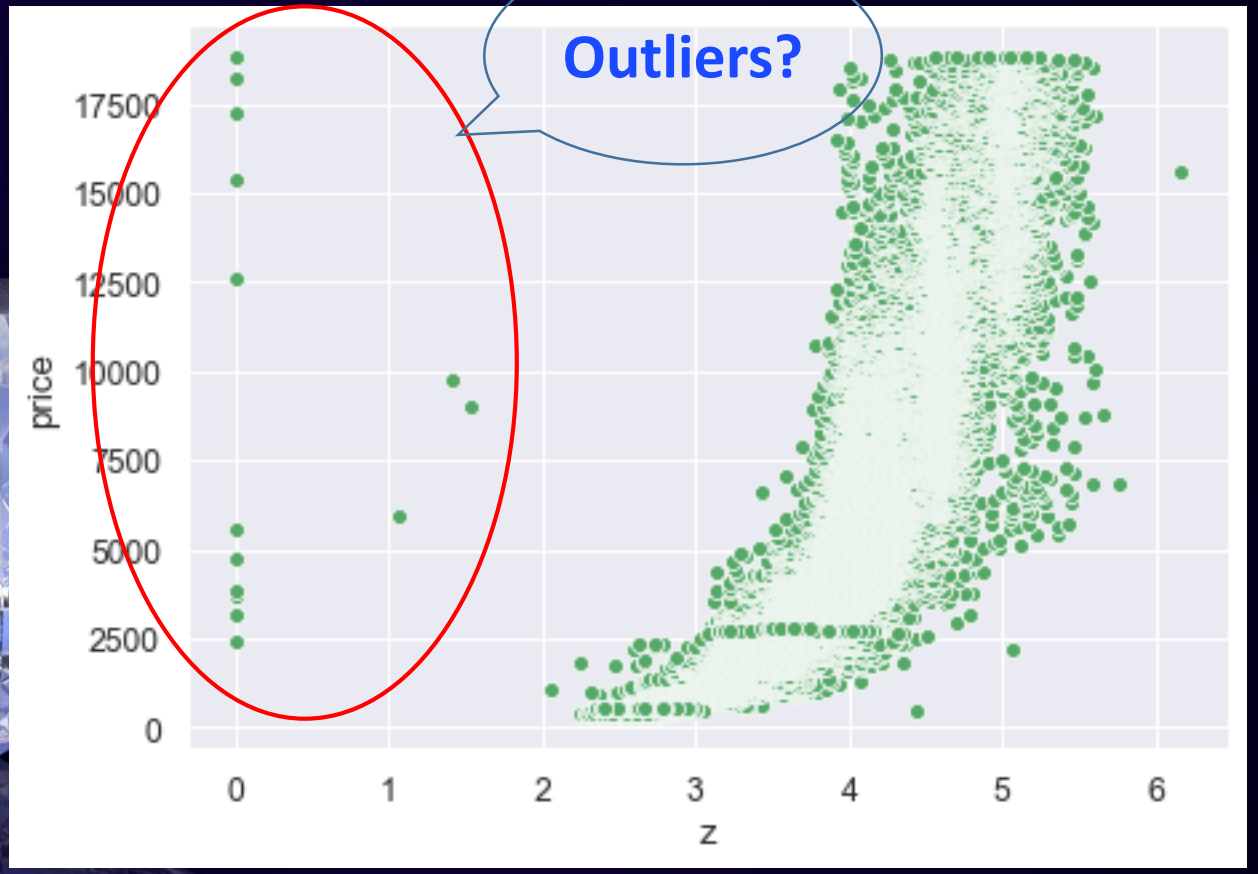
Visualize nominal data

Data analysis

4

```
sns.scatterplot(x=self.df["y"], y = self.df["price"])
sns.scatterplot(x=self.df["z"], y = self.df["price"])
```





Data analysis

4

Decide to get rid of outlier(s).

```
df[df["z"] < 2]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
2207	1.00	Premium	G	SI2	59.1	59.0	3142	6.55	6.48	0.00
2314	1.01	Premium	H	I1	58.1	59.0	3167	6.66	6.60	0.00
4791	1.10	Premium	G	SI2	63.0	59.0	3696	6.50	6.47	0.00
5471	1.01	Premium	F	SI2	59.2	58.0	3837	6.50	6.47	0.00
10167	1.50	Good	G	I1	64.0	61.0	4731	7.15	7.04	0.00
13601	1.15	Ideal	G	VS2	59.2	56.0	5564	6.88	6.83	0.00
14635	1.07	Ideal	F	SI1	60.6	57.0	5909	6.62	6.67	1.07
20694	1.53	Ideal	I	SI1	61.9	54.0	8971	7.43	7.50	1.53
21654	1.41	Ideal	H	VS1	60.7	56.0	9752	7.31	7.22	1.41
24394	2.18	Premium	H	SI2	59.4	61.0	12631	8.49	8.45	0.00
26123	2.25	Premium	I	SI1	61.3	58.0	15397	8.52	8.42	0.00
27112	2.20	Premium	H	SI1	61.2	59.0	17265	8.42	8.37	0.00
27503	2.02	Premium	H	VS2	62.7	53.0	18207	8.02	7.95	0.00
27739	2.80	Good	G	SI2	63.8	58.0	18788	8.90	8.85	0.00
51506	1.12	Premium	G	I1	60.4	59.0	2383	6.71	6.67	0.00

```
z_index2 = list(df[df["z"] < 2].index)
for i in z_index2:
    df.drop(i, inplace=True)
```

Final scatterplot between z and price

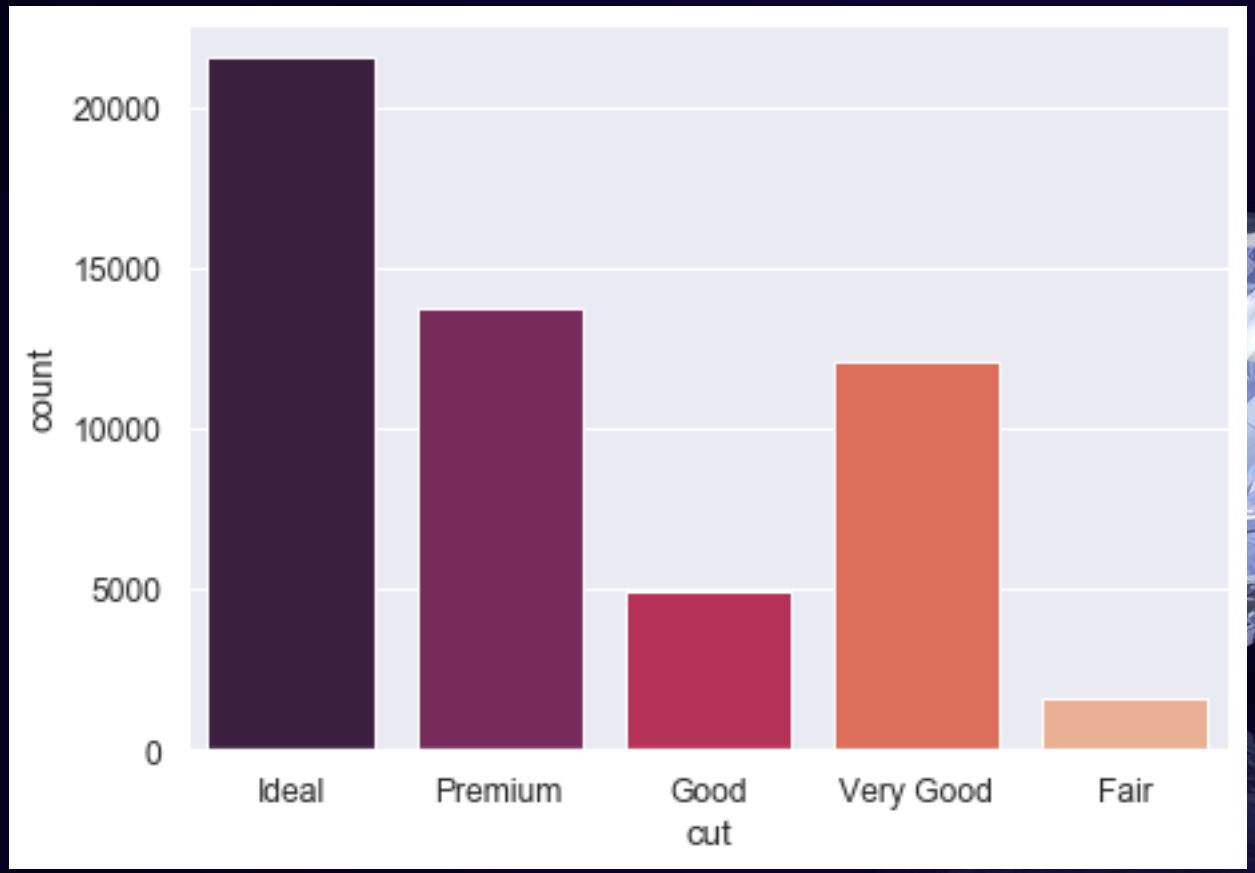
After removing outliers

Visualize categorical data

Data analysis 4

Count-plot: Cut

```
sns.countplot(df["cut"], palette='rocket')
```



```
df["cut"].value_counts()
```

```
Ideal        21551
Premium      13791
Very Good    12082
Good          4906
Fair          1610
Name: cut, dtype: int64
```

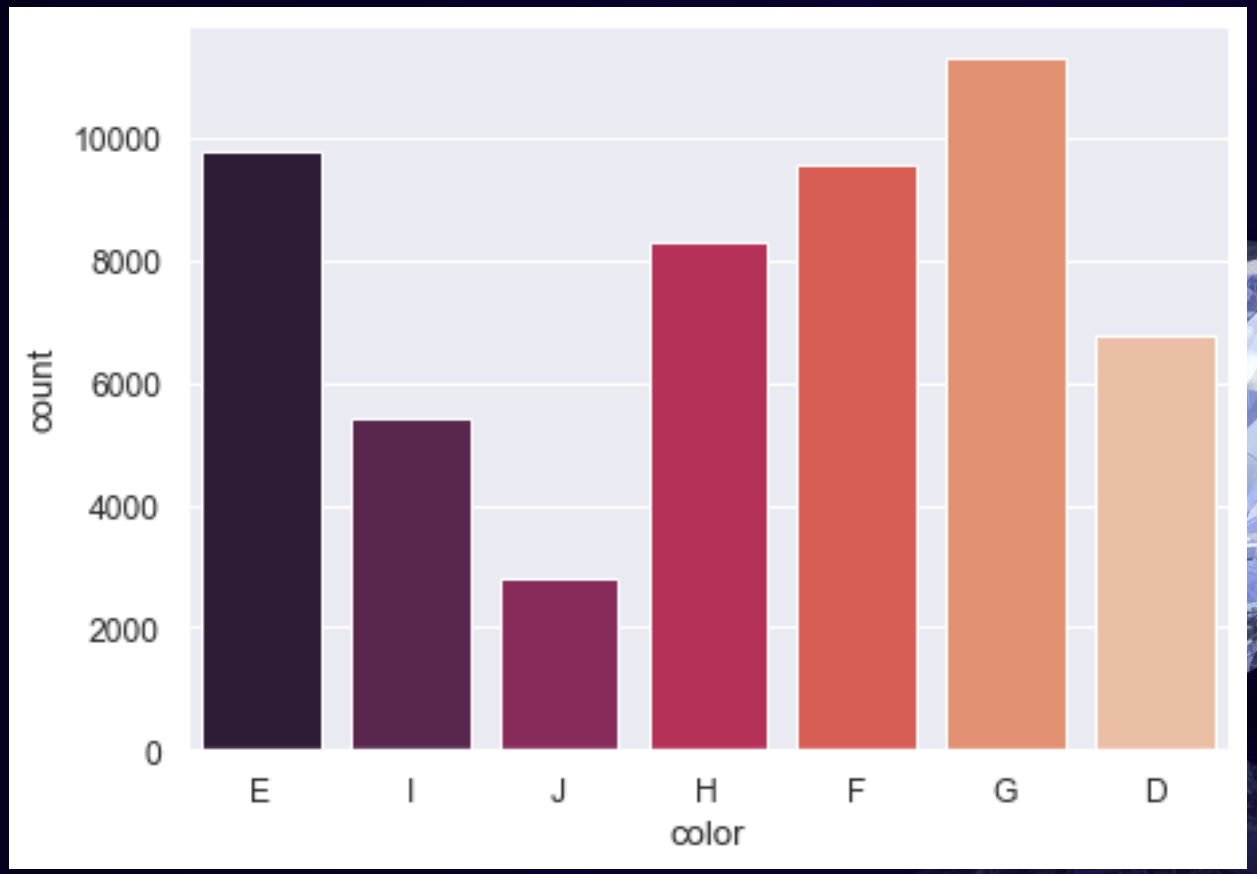
Visualize categorical data

Data analysis

4

Count-plot: Color

```
sns.countplot(df["color"], palette='rocket')
```



```
df["color"].value_counts()
```

```
G    11292
E     9797
F     9542
H     8304
D     6775
I     5422
J     2808
Name: color, dtype: int64
```

Visualize categorical data

Data analysis

4

Count-plot: Clarity

```
sns.countplot(df["clarity"], palette='rocket')
```

```
df["clarity"].value_counts()
```

```
SI1     13065
VS2     12258
SI2      9194
VS1      8171
VVS2     5066
VVS1     3655
IF       1790
I1        741
Name: clarity, dtype: int64
```

Data analysis

4

Convert categorical data.

```
df.head(2)
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31

Cut: **Ideal > Premium > Very good > Good > Fair**

Color: **D > E > F > G > H > I > J**

Clarity: **IF > VVS1 > VVS2 > VS1 > VS2 > SI1 > SI2 > I1**

Data analysis

4

Cut: Ideal > Premium > Very good > Good > Fair

```
def grade(self,x):
    """convert cut grade string feature into numerical value"""
    x = str(x)
    if x == "Fair":
        return 1
    elif x == "Good":
        return 2
    elif x == "Very Good":
        return 3
    elif x == "Premium":
        return 4
    elif x == "Ideal":
        return 5
```

Data analysis

4

Color: D > E > F > G > H > I > J

```
def color_grade(self,x):
    """convert color grade feature into numerical value"""
    x = str(x)
    if x == "J":
        return 1
    elif x == "I":
        return 1
    elif x == "H":
        return 2
    elif x == "G":
        return 3
    elif x == "F":
        return 4
    elif x == "E":
        return 5
    elif x == "D":
        return 6
```

Data analysis

4

Clarity: IF > VVS1 > VVS2 > VS1 > VS2 > SI1 > SI2 > I1

```
def clarity_quality(self,x):
    """convert clarity quality feature into numerical value"""
    x = str(x)
    if x == "I1":
        return 1
    elif x == "SI2":
        return 2
    elif x == "SI1":
        return 3
    elif x == "VS2":
        return 4
    elif x == "VS1":
        return 5
    elif x == "VVS2":
        return 6
    elif x == "VVS1":
        return 7
    elif x == "IF":
        return 8
```

Data analysis

4

Convert columns.

```python
def convert_columns(self):
    """Recall previous functions to convert the three features 1-cut_grade 2-color_grade and 3-clar_quality"""
    self.df["cut_grade"] = self.df["cut"].apply(self.grade)
    self.df["color_grade"] = self.df["color"].apply(self.color_grade)
    self.df["clar_quality"] = self.df["clarity"].apply(self.clarity_quality)
```

	carat	cut	color	clarity	depth	table	price	x	y	z	cut_grade	color_grade	clear_quality
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	5	5	2
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	4	5	3
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	2	5	5
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	4	1	4
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	2	1	2

Data analysis

4

Knowing about depth feature.

```
depth = df["z"] / ((df["x"] + df["y"])/2) * 100
pd.DataFrame(depth, columns=["Depth size in percent"]).head(15)
```

	Depth size in percent
0	61.286255
1	59.767141
2	56.896552
3	62.396204
4	63.291139
5	62.784810
6	62.295082
7	61.858191
8	65.098039
9	59.378882
10	64.009379
11	62.835249
12	60.362694

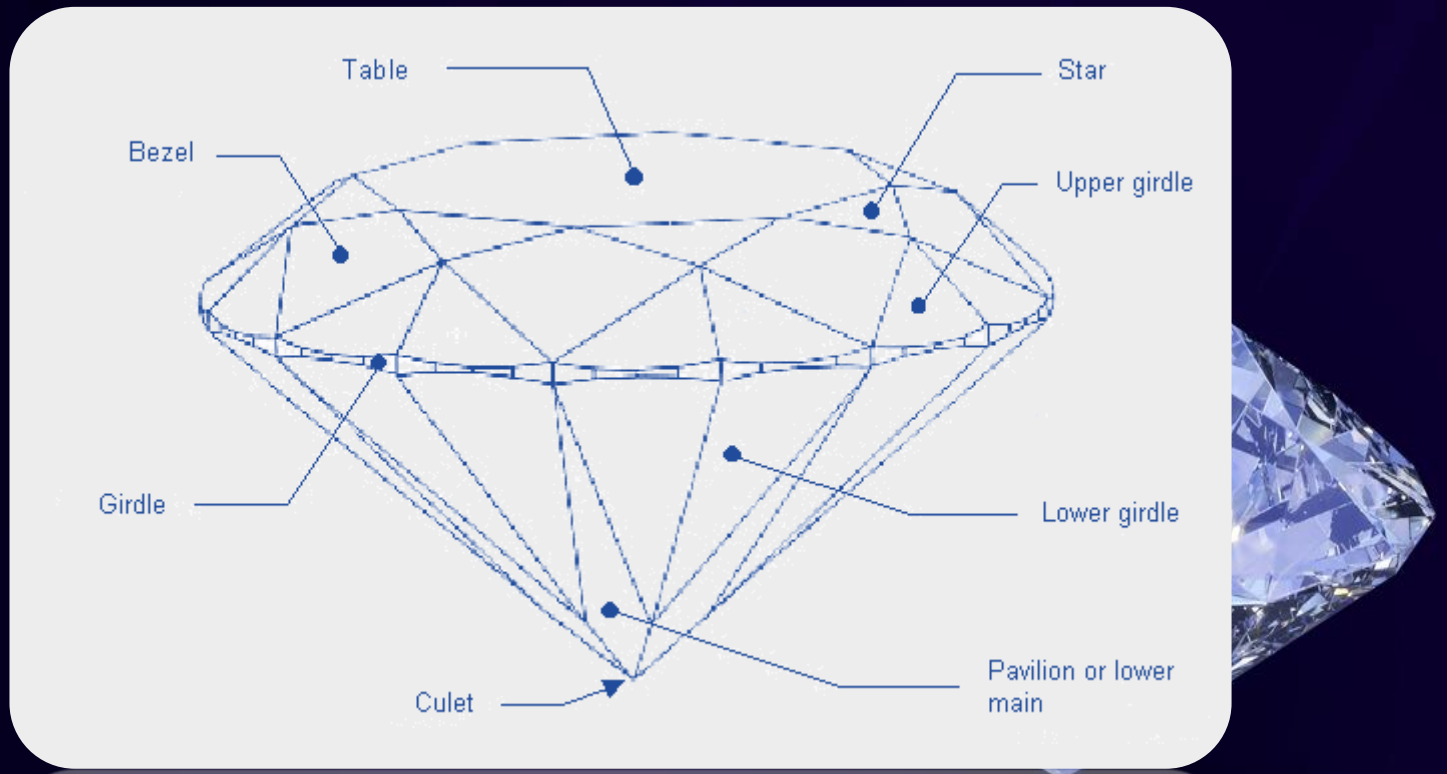
Original dataset

depth
61.5
59.8
56.9
62.4
63.3
62.8
62.3
61.9
65.1
59.4
64.0
62.8
60.4
62.2
60.2

Data analysis

4

Knowing about table feature.



Delete this feature doesn't affect on the predictions.

Data analysis 4

Delete those unnecessary columns.

```
def delete_columns(self):
    """Delete unncessary columns"""
    self.df.drop(["cut","color","clarity","depth","table"], inplace=True, axis=1)
```

Final Dataset

	carat	price	x	y	z	cut_grade	color_grade	clear_quality
0	0.23	326	3.95	3.98	2.43	5	5	2
1	0.21	326	3.89	3.84	2.31	4	5	3

Machine Learning 5

```python
import pandas as pd
import seaborn as sns
import joblib
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```


```python
def split_data(self):
    """Split DataFrame into X -> features and y -> the target
    the function returns X_train, X_test, y_train, y_test"""
    X = self.df.drop("price", axis = 1)
    y = self.df["price"]
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=30)
    return X_train, X_test, y_train, y_test
```

Machine Learning 5

```python
def split_display_shapes(self, X_train, X_test, y_train, y_test):
    """Display the shapes of training and testing"""
    print(X_train.shape)
    print(X_test.shape)
    print(y_train.shape)
    print(y_test.shape)
```

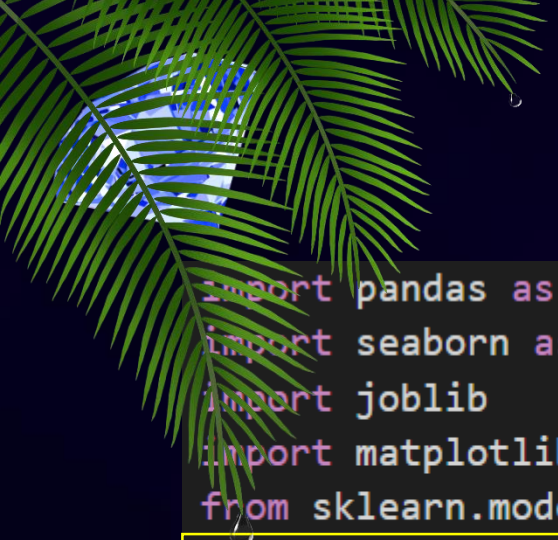
```
(40437, 7)
(13480, 7)
(40437,)
(13480,)
```

X train
X test
y train
y test

Machine Learning 5

```
import pandas as pd
import seaborn as sns
import joblib
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```

```
def regression_model(self,X_train, y_train):
    """Evaluate and fit the DecisionTreeRegressor model by x_train, y_train
    the function returns the fitted model variable that can be used for predictions later"""
    model = DecisionTreeRegressor()
    fitted_model = model.fit(X_train, y_train)
    return fitted_model
```




Machine Learning 5

Model performance on Training and Testing.

```
def model_accuracy_score(self,model, X_train, X_test, y_train, y_test):
    """How accurate is DecisionTreeRegressor model on training and testing"""
    print("Model train accuracy score :",model.score(X_train, y_train))
    print("Model test accuracy score :",model.score(X_test, y_test))
```

```
Model train accuracy score is : 0.999997685088036
Model test accuracy score is : 0.9623301284501552
```

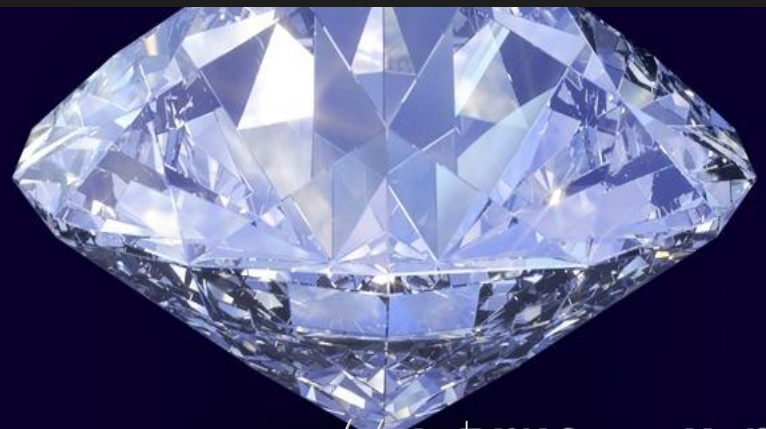
Machine Learning 5

R2 Score: with best possible score is 1.0

```
from sklearn.metrics import r2_score

print("R2_Score is: ", r2_score(y_test, y_pred).round(4))
```

```
R2_Score is:  0.9632
```



Using formula

$$R^2 = 1 - \frac{RSS}{TSS}$$

Where →

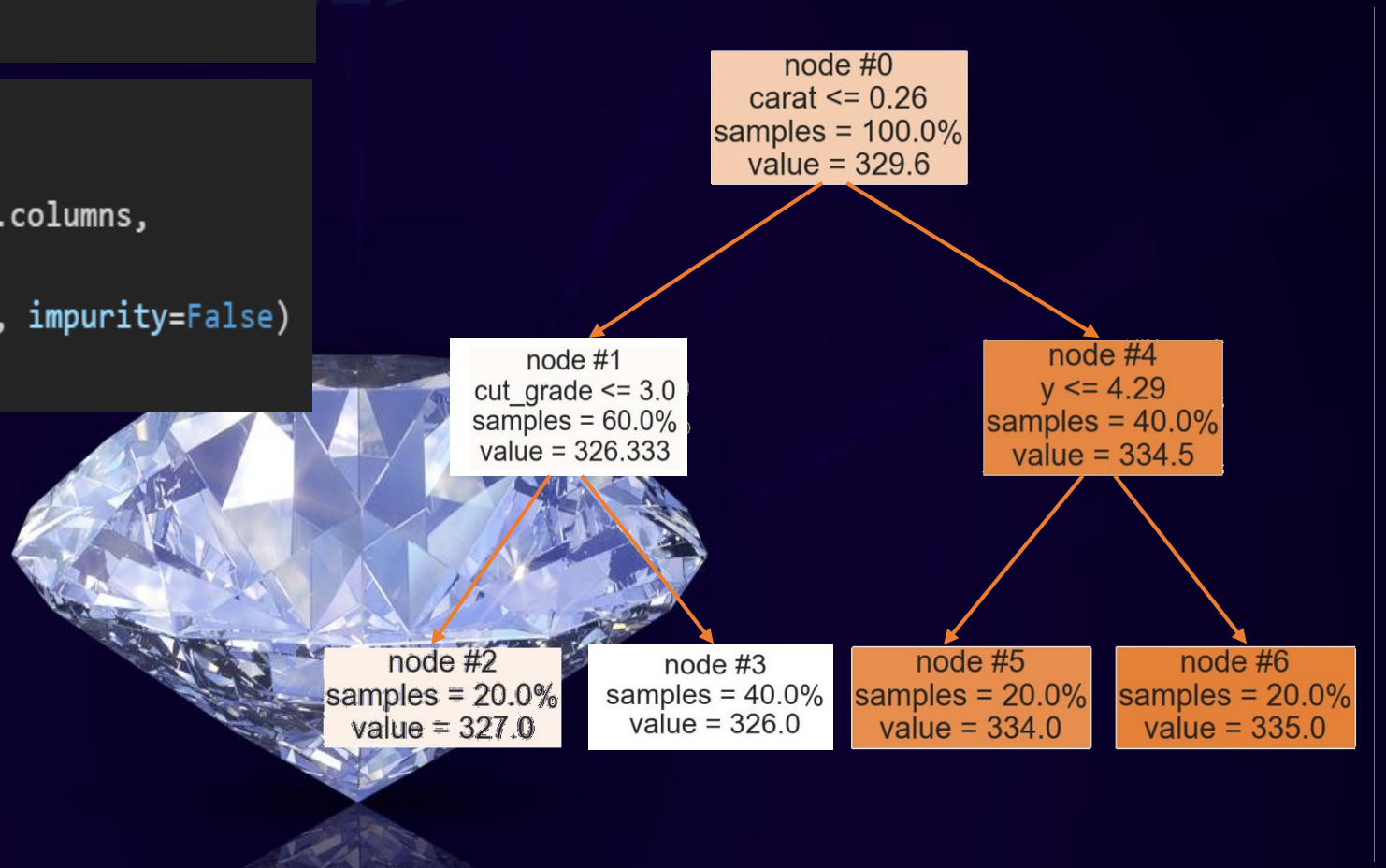
$$1 - \frac{((y\_true - y\_pred) ** 2).sum()}{((y\_true - y\_true.mean()) ** 2).sum()}$$

Machine Learning 5

Plot decision tree of 5 samples

```
from sklearn import tree
```

```
model.fit(X_train[:5], y_train[:5])
fig = plt.figure(figsize=(25,20))
tree.plot_tree(model, feature_names=X_train.columns,
               filled=True,node_ids=True,
              proportion=True, fontsize=40, impurity=False)
fig.savefig("DecisionTree_5_samples.jpg")
```



Model Deployment 6

Save the model to used it in any application.

```
import joblib
```

```
def save_model(self, model, name):
    """Save the fitted model to deploy it and use it with it`s weight later"""
    joblib.dump(model,name + ".h5")
    print("Model saved sucessfuully")

def load_model(self, name):
    """Load the model that has been saved alrady"""
    file = joblib.load(name + ".h5")
    print(f"The model {file} has been loaded sucessfully")
    return file
```

Time to lunch an application Deploy model to ‘website’

predict my diamond price.

Please fill in your diamond details.

carat weight (0.5 - 7.17 CT)

length (0 - 10.74 in mm)

width (0 - 58.9 in mm)

depth (0 - 31.8 in mm)

Color grade

D

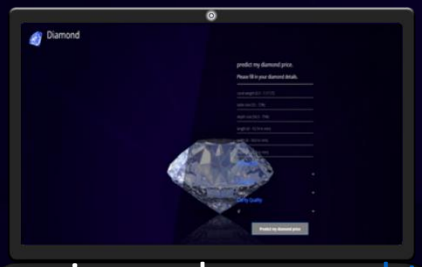
Cut grade

Ideal

Clarity Quality

IF

Predict my diamond price



Type in any browser: http://localhost:5000/


THANKS
FOR WATCHING