In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import math
import re
import os
```

In [2]:
```
df = pd.read_csv(r"C:\Users\saiku\seaborn-data\diamonds.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	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

In [4]:
```
df.shape
```

Out[4]: (53940, 10)

In [5]:
```
df.describe()
```

Out[5]:

	carat	depth	table	price	x	y	
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31

In [6]:
```
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
```

In [7]:
```
for i in df.index:
    if df.loc[i, "x"] == 0:
        df.drop(i, inplace = True)
```

In [8]:
```
df.describe()
```

Out[8]:

	carat	depth	table	price	x	y	
count	53932.000000	53932.000000	53932.000000	53932.000000	53932.000000	53932.000000	53932
mean	0.797879	61.749336	57.457029	3932.136079	5.732007	5.735254	3
std	0.473986	1.432514	2.234064	3988.734835	1.119670	1.140343	0
min	0.200000	43.000000	43.000000	326.000000	3.730000	3.680000	0
25%	0.400000	61.000000	56.000000	949.750000	4.710000	4.720000	2
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3
75%	1.040000	62.500000	59.000000	5324.000000	6.540000	6.540000	4
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31

In [9]:
```
for i in df.index:
    if df.loc[i, "z"] == 0:
        df.drop(i, inplace = True)
```

In [10]:
```
df.describe()
```

Out[10]:

	carat	depth	table	price	x	y	
count	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920
mean	0.797698	61.749514	57.456834	3930.993231	5.731627	5.734887	3
std	0.473795	1.432331	2.234064	3987.280446	1.119423	1.140126	0
min	0.200000	43.000000	43.000000	326.000000	3.730000	3.680000	1
25%	0.400000	61.000000	56.000000	949.000000	4.710000	4.720000	2
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3
75%	1.040000	62.500000	59.000000	5323.250000	6.540000	6.540000	4
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31

In [11]:
```
df.shape
```

Out[11]:
```
(53920, 10)
```

In [12]:
```
y=df['price']
X=df[['carat','cut','color','clarity','depth','table','x','y','z']]
```

In [13]:
```
df['carat'].value_counts()
```

Out[13]:
```
0.30    2604
0.31    2249
1.01    2240
0.70    1981
0.32    1840
        ...
3.02       1
3.65       1
3.50       1
3.22       1
3.11       1
Name: carat, Length: 273, dtype: int64
```

In [14]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, randon
X_train.head()
```

Out[14]:

	carat	cut	color	clarity	depth	table	x	y	z
19321	1.20	Good	G	VS2	57.8	59.0	7.06	7.00	4.06
9132	1.08	Ideal	H	SI2	60.4	57.0	6.68	6.63	4.02
38015	0.39	Premium	I	VVS2	63.0	56.0	4.68	4.62	2.93
35917	0.43	Premium	H	SI1	60.2	57.0	4.93	4.91	2.96
50428	0.73	Ideal	H	SI2	61.6	57.0	5.79	5.81	3.57

In [15]:
```
X_train.head()
```

Out[15]:

	carat	cut	color	clarity	depth	table	x	y	z
19321	1.20	Good	G	VS2	57.8	59.0	7.06	7.00	4.06
9132	1.08	Ideal	H	SI2	60.4	57.0	6.68	6.63	4.02
38015	0.39	Premium	I	VVS2	63.0	56.0	4.68	4.62	2.93
35917	0.43	Premium	H	SI1	60.2	57.0	4.93	4.91	2.96
50428	0.73	Ideal	H	SI2	61.6	57.0	5.79	5.81	3.57

In [16]:
```
X_train.dtypes
```

Out[16]:
```
carat        float64
cut           object
color         object
clarity       object
depth        float64
table        float64
x            float64
y            float64
z            float64
dtype: object
```

In [17]:
```
X_train_cat = X_train.select_dtypes(include = 'object')
X_train_cat.head()
```

Out[17]:

	cut	color	clarity
19321	Good	G	VS2
9132	Ideal	H	SI2
38015	Premium	I	VVS2
35917	Premium	H	SI1
50428	Ideal	H	SI2

In [18]:
```
X_train_cat.shape
```

Out[18]: (40440, 3)

In [19]:
```
X_train_num = X_train.select_dtypes(include = 'float64')
X_train_num.head()
```

Out[19]:

	carat	depth	table	x	y	z
19321	1.20	57.8	59.0	7.06	7.00	4.06
9132	1.08	60.4	57.0	6.68	6.63	4.02
38015	0.39	63.0	56.0	4.68	4.62	2.93
35917	0.43	60.2	57.0	4.93	4.91	2.96
50428	0.73	61.6	57.0	5.79	5.81	3.57

In [20]:
```
X_train_num.shape
```

Out[20]: (40440, 6)

In [21]:
```
X_train_cat_le = pd.DataFrame(index=X_train_cat.index)
```

In [22]:
```
X_train_cat_le.head()
```

Out[22]:

19321
9132
38015
35917
50428

In [23]:
```
X_train_cat.cut.unique()
```

Out[23]: array(['Good', 'Ideal', 'Premium', 'Very Good', 'Fair'], dtype=object)

In [24]:
```
cut_encoder = {'Fair' : 1,'Good' : 2,'Very Good' : 3,'Ideal' : 4,'Premium' : 5 }
X_train_cat_le['cut'] = X_train_cat['cut'].apply(lambda x:cut_encoder[x])
X_train_cat_le.head()
```

Out[24]:

	cut
19321	2
9132	4
38015	5
35917	5
50428	4

In [25]:
```
X_train_cat.color.unique()
```

Out[25]:
```
array(['G', 'H', 'I', 'J', 'E', 'D', 'F'], dtype=object)
```

In [26]:
```
color_encoder = {'D' : 1, 'E' : 2, 'F' : 3, 'G' : 4, 'H' : 5, 'I' : 6, 'J' : 7}
X_train_cat_le['color'] = X_train_cat['color'].apply(lambda x:color_encoder[x])
X_train_cat_le.head()
```

Out[26]:

	cut	color
19321	2	4
9132	4	5
38015	5	6
35917	5	5
50428	4	5

In [27]:
```
X_train_cat_le.shape
```

Out[27]:
```
(40440, 2)
```

In [28]:
```
X_train_cat.clarity.unique()
```

Out[28]:
```
array(['VS2', 'SI2', 'VVS2', 'SI1', 'VS1', 'VVS1', 'I1', 'IF'],
      dtype=object)
```

In [29]:
```
clarity_encoder = {'I1':1, 'SI2':2,'SI1':3,'VS2':4,'VS1':5,'VVS2':6,'VVS1':7,'IF'
X_train_cat_le['clarity'] = X_train_cat['clarity'].apply(lambda x:clarity_encoder
X_train_cat_le.head()
```

Out[29]:

	cut	color	clarity
19321	2	4	4
9132	4	5	2
38015	5	6	6
35917	5	5	3
50428	4	5	2

In [30]:
```
X_train_num.head()
```

Out[30]:

	carat	depth	table	x	y	z
19321	1.20	57.8	59.0	7.06	7.00	4.06
9132	1.08	60.4	57.0	6.68	6.63	4.02
38015	0.39	63.0	56.0	4.68	4.62	2.93
35917	0.43	60.2	57.0	4.93	4.91	2.96
50428	0.73	61.6	57.0	5.79	5.81	3.57

In [31]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_num_rescaled = pd.DataFrame(scaler.fit_transform(X_train_num),
                                    columns = X_train_num.columns,
                                    index = X_train_num.index)
X_train_num_rescaled.head()
```

Out[31]:

	carat	depth	table	x	y	z
19321	0.845610	-2.749623	0.689470	1.183523	1.097416	0.749997
9132	0.592901	-0.939977	-0.205331	0.844689	0.776067	0.692282
38015	-0.860177	0.869669	-0.652731	-0.938646	-0.969639	-0.880470
35917	-0.775940	-1.079180	-0.205331	-0.715729	-0.717771	-0.837183
50428	-0.144167	-0.104756	-0.205331	0.051105	0.063889	0.042981

In [32]:
```
X_train_transformed = pd.concat([X_train_num_rescaled,X_train_cat_le], axis=1)

X_train_transformed
```

Out[32]:

	carat	depth	table	x	y	z	cut	color	clarity
19321	0.845610	-2.749623	0.689470	1.183523	1.097416	0.749997	2	4	4
9132	0.592901	-0.939977	-0.205331	0.844689	0.776067	0.692282	4	5	2
38015	-0.860177	0.869669	-0.652731	-0.938646	-0.969639	-0.880470	5	6	6
35917	-0.775940	-1.079180	-0.205331	-0.715729	-0.717771	-0.837183	5	5	3
50428	-0.144167	-0.104756	-0.205331	0.051105	0.063889	0.042981	4	5	2
...	...	...	...	...	...	...	...	...	...
16313	0.445487	1.217678	0.689470	0.559356	0.498144	0.692282	3	4	5
79	-1.133945	0.591262	0.689470	-1.491480	-1.429950	-1.428768	3	2	7
12126	1.056201	0.104050	-0.205331	1.138940	1.053991	1.125149	4	4	2
14155	0.782433	0.591262	-1.100131	0.898189	0.906344	0.995289	4	4	2
38425	-0.607467	-2.192809	0.689470	-0.385812	-0.422477	-0.649607	5	4	2

In [33]:
```
X_test.head()
```

Out[33]:

	carat	cut	color	clarity	depth	table	x	y	z
6797	1.01	Good	H	SI1	64.0	58.0	6.31	6.37	4.06
30232	0.33	Ideal	E	VS2	60.5	56.0	4.48	4.51	2.72
7429	0.91	Premium	D	SI1	62.8	57.0	6.17	6.12	3.86
35524	0.43	Premium	F	VS2	59.8	58.0	4.89	4.94	2.94
38052	0.40	Very Good	F	VVS2	60.5	57.0	4.76	4.79	2.89

In [34]:
```
X_test.dtypes
```

Out[34]:
```
carat       float64
cut          object
color        object
clarity      object
depth       float64
table       float64
x           float64
y           float64
z           float64
dtype: object
```

In [35]:
```
X_test_cat = X_test.select_dtypes(include=['object'])
X_test_cat.head()
```

Out[35]:

	cut	color	clarity
6797	Good	H	SI1
30232	Ideal	E	VS2
7429	Premium	D	SI1
35524	Premium	F	VS2
38052	Very Good	F	VVS2

In [36]:
```
X_test_num = X_test.select_dtypes(include=['float64'])
X_test_num.head()
```

Out[36]:

	carat	depth	table	x	y	z
6797	1.01	64.0	58.0	6.31	6.37	4.06
30232	0.33	60.5	56.0	4.48	4.51	2.72
7429	0.91	62.8	57.0	6.17	6.12	3.86
35524	0.43	59.8	58.0	4.89	4.94	2.94
38052	0.40	60.5	57.0	4.76	4.79	2.89

In [37]:
```
X_test_cat_le = pd.DataFrame(index = X_test_cat.index)
X_test_cat_le.head()
```

Out[37]:

6797
30232
7429
35524
38052

In [38]:
```
X_test_cat.cut.unique()
```

Out[38]:
```
array(['Good', 'Ideal', 'Premium', 'Very Good', 'Fair'], dtype=object)
```

In [39]:
```
cut_encoder = {'Fair' : 1,'Good' : 2,'Very Good' : 3,'Ideal' : 4,'Premium' : 5 }
X_test_cat_le['cut'] = X_test_cat['cut'].apply(lambda x:cut_encoder[x])
X_test_cat_le.head()
```

Out[39]:

	cut
6797	2
30232	4
7429	5
35524	5
38052	3

In [40]:
```
X_test_cat.color.unique()
```

Out[40]:
```
array(['H', 'E', 'D', 'F', 'G', 'J', 'I'], dtype=object)
```

In [41]:
```
color_encoder = {'D' : 1, 'E' : 2, 'F' : 3, 'G' : 4, 'H' : 5, 'I' : 6, 'J' : 7}
X_test_cat_le['color'] = X_test_cat['color'].apply(lambda x:color_encoder[x])
X_test_cat_le.head()
```

Out[41]:

	cut	color
6797	2	5
30232	4	2
7429	5	1
35524	5	3
38052	3	3

In [42]:
```
X_test_cat.clarity.unique()
```

Out[42]:
```
array(['SI1', 'VS2', 'VVS2', 'SI2', 'IF', 'I1', 'VVS1', 'VS1'],
      dtype=object)
```

In [43]:
```
clarity_encoder = {'I1':1, 'SI2':2,'SI1':3,'VS2':4,'VS1':5,'VVS2':6,'VVS1':7,'IF'
X_test_cat_le['clarity'] = X_test_cat['clarity'].apply(lambda x:clarity_encoder[x
X_test_cat_le.head()
```

Out[43]:

	cut	color	clarity
6797	2	5	3
30232	4	2	4
7429	5	1	3
35524	5	3	4
38052	3	3	6

In [44]:
```
X_test_num_rescaled = pd.DataFrame(scaler.transform(X_test_num),
                                   columns = X_test_num.columns,
                                   index = X_test_num.index)
X_test_num_rescaled.head()
```

Out[44]:

	carat	depth	table	x	y	z
6797	0.445487	1.565687	0.242070	0.514772	0.550255	0.749997
30232	-0.986531	-0.870375	-0.652731	-1.116980	-1.065175	-1.183477
7429	0.234896	0.730466	-0.205331	0.389939	0.333127	0.461419
35524	-0.775940	-1.357588	0.242070	-0.751396	-0.691716	-0.866041
38052	-0.839118	-0.870375	-0.205331	-0.867313	-0.821992	-0.938185

In [45]:
```
X_test_transformed = pd.concat([X_test_num_rescaled,X_test_cat_le],axis = 1)
X_test_transformed.head()
```

Out[45]:

	carat	depth	table	x	y	z	cut	color	clarity
6797	0.445487	1.565687	0.242070	0.514772	0.550255	0.749997	2	5	3
30232	-0.986531	-0.870375	-0.652731	-1.116980	-1.065175	-1.183477	4	2	4
7429	0.234896	0.730466	-0.205331	0.389939	0.333127	0.461419	5	1	3
35524	-0.775940	-1.357588	0.242070	-0.751396	-0.691716	-0.866041	5	3	4
38052	-0.839118	-0.870375	-0.205331	-0.867313	-0.821992	-0.938185	3	3	6

In [46]:
```
euc_dist = []
y_indices = np.array(X_train_transformed.index)
k = 5
y_predict = []
dist_list = []
for i in range(len(X_test_transformed)):
    dist = np.sqrt(((X_test_transformed.values[i]-X_train_transformed.values)**2)
    dist_list.append(dist)
    sort_index=np.argsort(dist_list[i])
    sort_y=y_indices[sort_index]
    y_index=sort_y[:k]
    y_pred=y_train[y_index]
    y_predict.append(y_pred.values.mean())
```

In [47]:
```
temp_df = pd.DataFrame({'Actual':y_test,'predicted':y_predict})
```

In [48]:
```
temp_df.head()
```

Out[48]:

	Actual	predicted
6797	4116	4248.4
30232	723	869.8
7429	4228	4399.6
35524	905	967.8
38052	1012	1295.6

In [49]:
```
from sklearn import metrics
from sklearn.metrics import r2_score
```

In [50]:
```
print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,y_predict))
```

```
Mean Absolute Error:  382.09099406528185
```

In [51]:
```
print('Mean Squared Error: ',metrics.mean_squared_error(y_test,y_predict))
```

```
Mean Squared Error:  510864.3279436202
```

In [52]:
```
print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,y_pre
```

```
Root Mean Squared Error:  714.7477372777197
```

In [53]:
```
print('Accuracy of algorithm: ',r2_score(y_test,y_predict))
```

```
Accuracy of algorithm:  0.9675594228434207
```

APPLYING KNN ALGORITHM

In [54]:
```
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor()
regressor.fit(X_train_transformed,y_train)
```

Out[54]:
```
KNeighborsRegressor()
```

In [55]:
```
y_test_predict_knn = regressor.predict(X_test_transformed)
```

In [56]:
```
y_test_predict_knn
```

Out[56]:
```
array([4248.4,  869.8, 4399.6, ..., 1910. , 1116. , 7369.6])
```

In [57]:
```
algorithm_df = pd.DataFrame({'Actual':y_test,'predicted':y_predict})
algorithm_df.head()
```

Out[57]:

	Actual	predicted
6797	4116	4248.4
30232	723	869.8
7429	4228	4399.6
35524	905	967.8
38052	1012	1295.6

In [58]:
```
from sklearn import metrics
from sklearn.metrics import r2_score
```

In [59]:
```
print('Mean Absolute Error: ',metrics.mean_absolute_error(y_test,y_predict))
```

```
Mean Absolute Error:  382.09099406528185
```

In [60]:
```
print('Mean Squared Error: ',metrics.mean_squared_error(y_test,y_predict))
```

```
Mean Squared Error:  510864.3279436202
```

In [61]:
```
print('Root Mean Squared Error: ',np.sqrt(metrics.mean_squared_error(y_test,y_pre
```

```
Root Mean Squared Error:  714.7477372777197
```

In [62]:
```
print('Accuracy of algorithm: ',r2_score(y_test,y_predict))
```

```
Accuracy of algorithm:  0.9675594228434207
```

In [ ]: