Analysis diamonds by their cut,color,clarity ,price and other attributes

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn .preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression,Ridge,Lasso
```

In [2]:

```
data=pd.read_csv('diamonds.csv')
data
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
53935	53936	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

53940 rows × 11 columns

In [3]:

```
data.drop(data.columns[0],axis=1,inplace=True)
```

In [4]:

```
y=data['price']
X=data.drop('price',axis=1)
```

In [14]:

```
data.nunique(axis = 0, dropna = True)
#or
#print(f"Cuts:{len(data['cut'].unique())}")
#print(f"Clarity:{len(data['clarity'].unique())})
#print(f"Color : {len(data['color'].unique())})
```

Out[14]:

```
carat         273
cut             5
color           7
clarity         8
depth         184
table         127
price       11602
x             554
y             552
z             375
dtype: int64
```

In [20]:

```
encoder =LabelEncoder()
X['cut']=encoder.fit_transform(X['cut'])
cut_mapping={index: label for index , label in enumerate(encoder.classes_)}
cut_mapping

X['color']=encoder.fit_transform(X['color'])
color_mapping={index: label for index , label in enumerate(encoder.classes_)}
color_mapping

X['clarity']=encoder.fit_transform(X['clarity'])
clarity_mapping={index: label for index , label in enumerate(encoder.classes_)}
clarity_mapping
```

Out[20]:

```
{0: 'I1',
 1: 'IF',
 2: 'SI1',
 3: 'SI2',
 4: 'VS1',
 5: 'VS2',
 6: 'VVS1',
 7: 'VVS2'}
```

In [21]:

```
print(cut_mapping)
print(color_mapping)
print(clarity_mapping)
```

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

In [22]:

```
X
```

Out[22]:

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

53940 rows × 9 columns

In [23]:

```
scaler =MinMaxScaler()
X=scaler.fit_transform(X)
```

In [25]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,train_size=0.8)
```

In [29]:

```
std_model=LinearRegression()
l1_model=Lasso(alpha=1)
```

```
l2_model=Ridge(alpha=1)

std_model.fit(X_train,y_train)
l1_model.fit(X_train,y_train)
l2_model.fit(X_train,y_train)
```

Out[29]:

```
Ridge(alpha=1)
```

In [30]:

```
print(f"---Without regularization: {std_model.score(X_test,y_test)}")
print(f"Lass0(l1) regularization: {l1_model.score(X_test,y_test)}")
print(f"Ridge(l2) regularization: {l2_model.score(X_test,y_test)}")
```

```
---Without regularization: 0.8885588236488259
---Without regularization: 0.8879923610225717
---Without regularization: 0.8884321168000842
```

In [35]:

```
l2_model=Ridge(alpha=0.001)
l2_model.fit(X_train,y_train)
print(f"Ridge(l2) regularization: {l2_model.score(X_test,y_test)}")
```

```
Ridge(l2) regularization: 0.8885590417355508
```

In [ ]: