In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
import statsmodels.api as sm
```

In [2]:
```
df=pd.read_csv("Downloads/DiamondsNew.csv")
```

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

Out[3]:

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

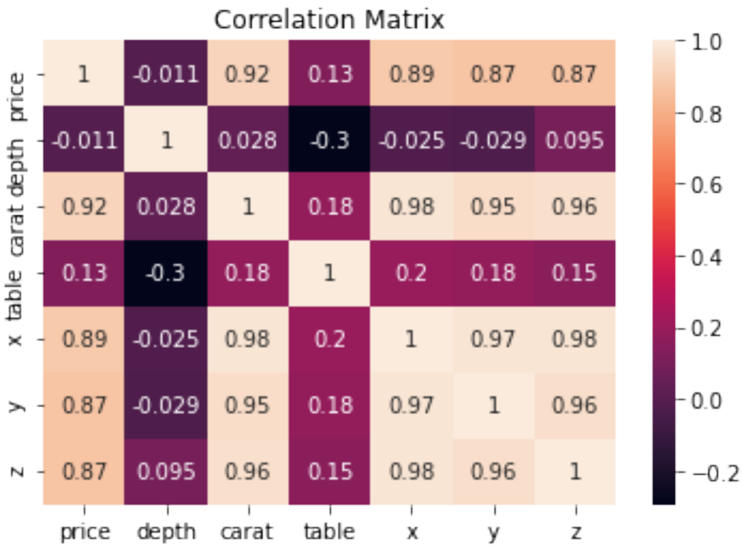
In [4]:
```
#categorical variables
cat_columns_data=df.loc[:,'Unnamed: 0':'color']
```

In [5]:
```
#dropping the category variable
df.drop(['Unnamed: 0','cut','clarity','color'],axis=1,inplace=True)
```

In [6]:
```
zero_indexs=df[(df['x']==0) | (df['y']==0) | (df['z']==0)].index
```

In [7]:
```
#taking indexes of zero values x,y,z rows and dropping them
df.drop(zero_indexs,axis=0,inplace=True)
```

In [8]:
```
#to find correlation matrix
sns.heatmap(df.corr(),annot=True)
plt.title('Correlation Matrix')
plt.show()
```



In [9]:
```
#dropping x y and z because of high multicollinearity as x y and z
df.drop(['x','y','z'],axis=1,inplace=True)
```

In [10]:
```
from scipy import stats
```

In [11]:
```
z = np.abs(stats.zscore(np.array(df.table)))
print(z)
```

```
[1.09972532 1.58598783 3.37646327 ... 1.13836898 0.24313126 1.0997
2532]
```

In [12]:
```
#finding outliers using IQR

percentile25Depth = df['depth'].quantile(0.25)
percentile75Depth = df['depth'].quantile(0.75)
IQR= percentile75Depth-percentile25Depth
upper_bound=percentile75Depth+1.5*IQR
lower_bound=percentile25Depth-1.5*IQR

#For table column
Q3=df.table.quantile(0.75)
Q1=df.table.quantile(0.25)
IQR=Q3-Q1
upper_tabel=Q3+1.5*IQR
lower_table=(Q1-1.5*IQR)

#For carat column
Q3=df.carat.quantile(0.75)
Q1=df.carat.quantile(0.25)
IQR=Q3-Q1
upper_carat=Q3+1.5*IQR
lower_carat=(Q1-1.5*IQR)
```

In [13]:
```
df=df[((df['depth']<upper_bound) & (df['depth']>lower_bound)) & ((df
```

In [14]:
```
from sklearn.preprocessing import MinMaxScaler
for i in ['carat','table','depth']:
    scaler=MinMaxScaler().fit(df[[i]])
    df[i]=scaler.transform(df[[i]])
```

In [15]:
```
X=df.drop('price',axis=1)
Y=df.price
```

In [16]:
```
df
```

Out[16]:

	price	depth	carat	table
0	326	0.457627	0.016760	0.263158
1	326	0.169492	0.005587	0.789474
3	334	0.610169	0.050279	0.526316
4	335	0.762712	0.061453	0.526316
5	336	0.677966	0.022346	0.438596
...	...	...	...	...
53935	2757	0.338983	0.290503	0.438596
53936	2757	0.728814	0.290503	0.263158
53937	2757	0.677966	0.279330	0.701754
53938	2757	0.372881	0.368715	0.526316
53939	2757	0.576271	0.307263	0.263158

49100 rows × 4 columns

In [17]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,ra
```

In [18]:
```
linear_model=LinearRegression()
linear_model.fit(X_train,Y_train)
```

Out[18]: LinearRegression()

In [19]:
```
y_pred=linear_model.predict(X_test)
```

In [20]:
```
linear_model.score(X_train,Y_train)
```

Out[20]: 0.8259078865390754

In [21]:
```
new_linear_model=sm.OLS(Y, sm.add_constant(X)).fit()
```

In [22]:
```
new_linear_model.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.825
Model:	OLS	**Adj. R-squared:**	0.825
Method:	Least Squares	**F-statistic:**	7.733e+04
Date:	Sat, 15 Jan 2022	**Prob (F-statistic):**	0.00
Time:	23:58:01	**Log-Likelihood:**	-4.2551e+05
No. Observations:	49100	**AIC:**	8.510e+05
Df Residuals:	49096	**BIC:**	8.511e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	166.5470	28.361	5.872	0.000	110.960	222.134
depth	-783.3942	34.965	-22.405	0.000	-851.927	-714.862
carat	1.402e+04	29.366	477.402	0.000	1.4e+04	1.41e+04
table	-1064.1285	37.598	-28.302	0.000	-1137.822	-990.435

Omnibus:	19406.306	**Durbin-Watson:**	0.672
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	151500.037
Skew:	1.705	**Prob(JB):**	0.00
Kurtosis:	10.901	**Cond. No.**	9.65

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [23]:
```
from sklearn.metrics import r2_score,mean_squared_error,mean_square
```

In [24]:
```
r2_score(Y_test,y_pred)
```

Out[24]: 0.822990892006697

In [25]:
```
RMSE=np.sqrt(mean_squared_error(Y_test,y_pred))
```

In [26]:
```
RMSE
```

Out[26]: 1404.1348014173188

In [27]:
```
SSR=np.sum((y_pred-np.mean(Y_test))**2)
```

In [28]:
```
SSE=np.sum((Y_test-y_pred)**2)
```

In [29]:
```
SST=SSR+SSE
```

In [30]:
```
# IQR
#Q1 = np.percentile(df.table, 25,            interpolation = 'mi
#
#Q3 = np.percentile(df.table, 75,
#              interpolation = 'midpoint')
#IQR = Q3 - Q1
```

Using categorical variables

In [31]:
```
new_df=pd.read_csv("Downloads/DiamondsNew.csv")
```

In [32]:
```
new_df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [33]:
```
#encoding the categorical column value according to the choice
new_df=new_df.replace({'cut': {'Fair':0,'Good':1,'Very Good':2,'Pre
```

In [34]:
```
new_df=new_df.replace({'clarity': {"IF": 8, 'VVS1' :7, 'VVS2': 6, 'V
```

In [35]:
```
new_df=new_df.replace({'color' : { 'D' : 6, 'E' : 5, 'F' : 4, 'G' :
```

In [36]:
```
new_df.head()
```

Out[36]:

	cut	clarity	color	price	depth	carat	table	x	y	z
0	4	2	5	326	61.5	0.23	55.0	3.95	3.98	2.43
1	3	3	5	326	59.8	0.21	61.0	3.89	3.84	2.31
2	1	5	5	327	56.9	0.23	65.0	4.05	4.07	2.31
3	3	4	1	334	62.4	0.29	58.0	4.20	4.23	2.63
4	1	2	0	335	63.3	0.31	58.0	4.34	4.35	2.75

In [37]:
```
new_df.drop(new_df[(new_df.x==0)|(new_df.y==0)|(new_df.z==0)].index
```

In [38]:
```
new_df.drop(['x','y','z'],axis=1,inplace=True)
```

In [39]:
```
percentile25Depth = new_df['depth'].quantile(0.25)
percentile75Depth = new_df['depth'].quantile(0.75)
IQR= percentile75Depth-percentile25Depth
upper_bound=percentile75Depth+1.5*IQR
lower_bound=percentile25Depth-1.5*IQR

#For table column
Q3=new_df.table.quantile(0.75)
Q1=new_df.table.quantile(0.25)
IQR=Q3-Q1
upper_tabel=Q3+1.5*IQR
lower_table=(Q1-1.5*IQR)

#For carat column
Q3=new_df.carat.quantile(0.75)
Q1=new_df.carat.quantile(0.25)
IQR=Q3-Q1
upper_carat=Q3+1.5*IQR
lower_carat=(Q1-1.5*IQR)
```

In [40]:
```
new_df=new_df[((new_df['depth']<upper_bound) & (new_df['depth']>low
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
cols=['carat', 'depth','table'] #identifying the columns to be stan
for i in cols:
    scale = StandardScaler().fit(df[[i]])
    df[i] = scale.transform(df[[i]])
```

In [42]:
```
x=new_df.drop('price',axis=1)
y=new_df.price
```

In [43]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,ra
```

In [44]:
```
new_model=LinearRegression()
new_model.fit(x_train,y_train)
```

Out[44]:
```
LinearRegression()
```

In [45]:
```
y_pred=new_model.predict(x_test)
```

In [46]:
```
new_model.score(x_train,y_train)
```

Out[46]:
```
0.8928951082031753
```

In [47]:
```
new_fitted_model=sm.OLS(y, sm.add_constant(x)).fit()
```

In [48]:
```
new_fitted_model.summary()
```

Out[48]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.893
Model:	OLS	**Adj. R-squared:**	0.893
Method:	Least Squares	**F-statistic:**	6.820e+04
Date:	Sat, 15 Jan 2022	**Prob (F-statistic):**	0.00
Time:	23:58:02	**Log-Likelihood:**	-4.1351e+05
No. Observations:	49100	**AIC:**	8.270e+05
Df Residuals:	49093	**BIC:**	8.271e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2899.6305	426.756	-6.795	0.000	-3736.077	-2063.184
cut	62.9910	5.907	10.663	0.000	51.412	74.570
clarity	504.7449	3.256	155.032	0.000	498.364	511.126
color	312.8014	3.079	101.584	0.000	306.766	318.837
depth	-26.8626	5.115	-5.251	0.000	-36.889	-16.836
carat	8741.2666	13.871	630.170	0.000	8714.079	8768.454
table	-30.8440	2.985	-10.332	0.000	-36.695	-24.993

Omnibus:	16489.248	**Durbin-Watson:**	0.565
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	96018.304
Skew:	1.499	**Prob(JB):**	0.00
Kurtosis:	9.160	**Cond. No.**	7.27e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.27e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

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

In [50]:
```
r2_score(y_test,y_pred)
```

Out[50]: 0.8927916059469776

In [51]:
```
RMSE=np.sqrt(mean_squared_error(y_test,y_pred))
```

In [52]:
```
RMSE
```

Out[52]: 1092.7615158662854

In [53]:
```
SSR=np.sum((y_pred-np.mean(y_test))**2)
```

In [54]:
```
SSE=np.sum((y_test-y_pred)**2)
```

In [55]:
```
SST=SSR+SSE
```

In [56]:
```
SST
```

Out[56]: 110169614239.6313

In [57]:
```
from statsmodels.formula.api import ols
# Ordinary Least Squares (OLS) model
model = ols('price ~depth+carat+table', data=new_df).fit()
anova_table = sm.stats.anova_lm(model)
anova_table
```

Out[57]:

	df	sum_sq	mean_sq	F	PR(>F)
depth	1.0	1.271498e+07	1.271498e+07	6.447938	1.111118e-02
carat	1.0	4.558855e+11	4.558855e+11	231185.654069	0.000000e+00
table	1.0	1.579588e+09	1.579588e+09	801.030253	8.228284e-175
Residual	49096.0	9.681463e+10	1.971945e+06	NaN	NaN

In [58]:
```
#RMSE WAS 1404======>WITHOUT CATEGORICAL VARIABLES
#RMSE IS 1092=======>WITH CATEGORICAL VARIABLES
```

In [ ]: