```
import pandas as pd
```

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

```
data.tail()
```

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
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

```
data.shape
```

```
(53940, 11)
```

```
data.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
```

```
data = data.drop(["Unnamed: 0"], axis=1)
```

```
data.head()
```

	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

```
data.describe()
```

	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

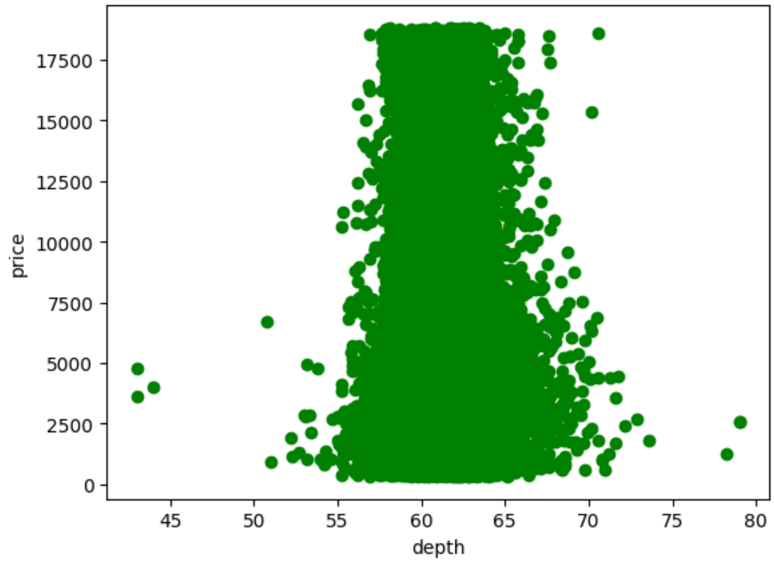
```
#Dropping dimentionless diamonds
data = data.drop(data[data["x"]==0].index)
```

```
data = data.drop(data[data["y"]==0].index)
data = data.drop(data[data["z"]==0].index)
data.shape
```

```
(53920, 10)
```

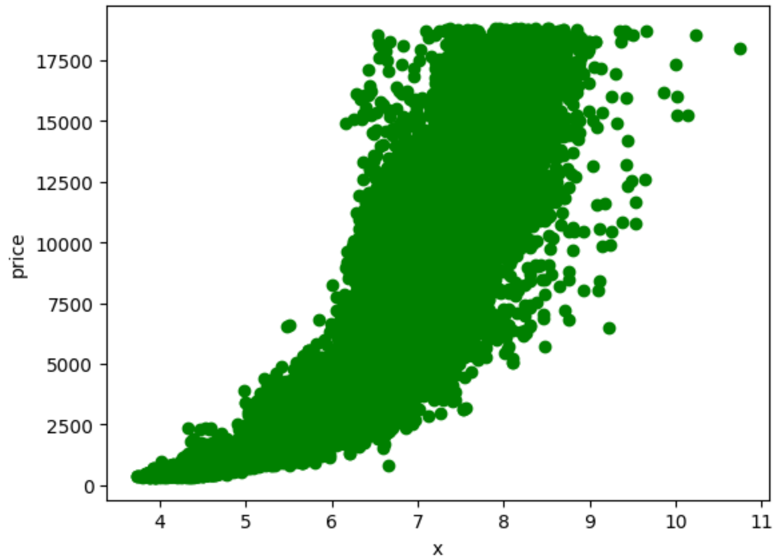
```
import matplotlib.pyplot as plt
x=data["depth"].values
y=data["price"].values
plt.scatter(x,y,color='green')
plt.xlabel("depth")
plt.ylabel("price")
```

```
Text(0, 0.5, 'price')
```




```
x=data["x"].values
y=data["price"].values
plt.scatter(x,y,color='green')
plt.xlabel("x")
plt.ylabel("price")
```

```
Text(0, 0.5, 'price')
```

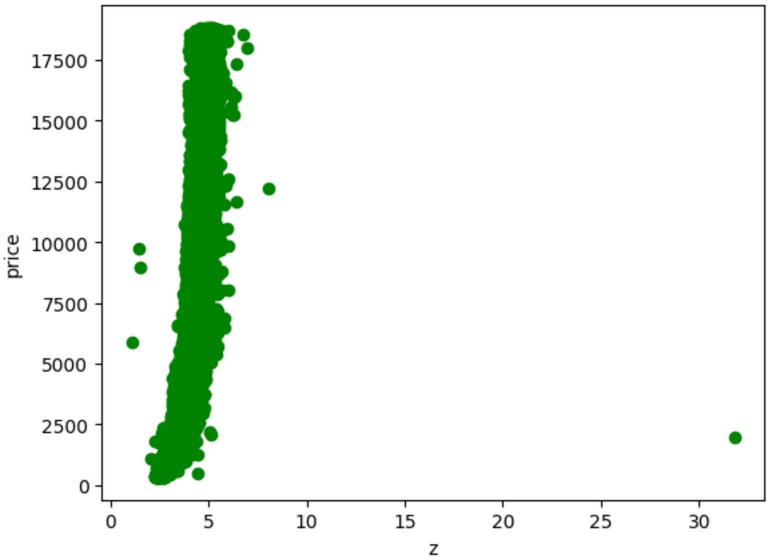



```
x=data["y"].values
y=data["price"].values
plt.scatter(x,y,color='green')
plt.xlabel("y")
plt.ylabel("price")
```

```
Text(0, 0.5, 'price')
```


```
x=data["z"].values
y=data["price"].values
plt.scatter(x,y,color='green')
plt.xlabel("z")
plt.ylabel("price")
```

```
Text(0, 0.5, 'price')
```




```
#Dropping the outliers.
data = data[(data["depth"]<75)&(data["depth"]>45)]
data = data[(data["table"]<80)&(data["table"]>40)]
data = data[(data["x"]<30)]
data = data[(data["y"]<30)]
data = data[(data["z"]<30)&(data["z"]>2)]
data.shape
```

```
(53907, 10)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53907 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   carat    53907 non-null  float64
 1   cut      53907 non-null  object 
 2   color    53907 non-null  object 
 3   clarity  53907 non-null  object 
 4   depth    53907 non-null  float64
 5   table    53907 non-null  float64
 6   price    53907 non-null  int64  
 7   x        53907 non-null  float64
 8   y        53907 non-null  float64
 9   z        53907 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 4.5+ MB
```

```
data.describe()
```

	carat	depth	table	price	x	y	z
count	53907.000000	53907.000000	53907.000000	53907.000000	53907.000000	53907.000000	53907.000000
mean	0.797628	61.749741	57.455948	3930.584470	5.731463	5.733292	3.539441
std	0.473765	1.420119	2.226153	3987.202815	1.119384	1.111252	0.691434
min	0.200000	50.800000	43.000000	326.000000	3.730000	3.680000	2.060000
25%	0.400000	61.000000	56.000000	949.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5322.000000	6.540000	6.540000	4.040000
max	5.010000	73.600000	79.000000	18823.000000	10.740000	10.540000	6.980000

```
s = (data.dtypes =="object")
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
```

```
Categorical variables:
['cut', 'color', 'clarity']
```

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
```

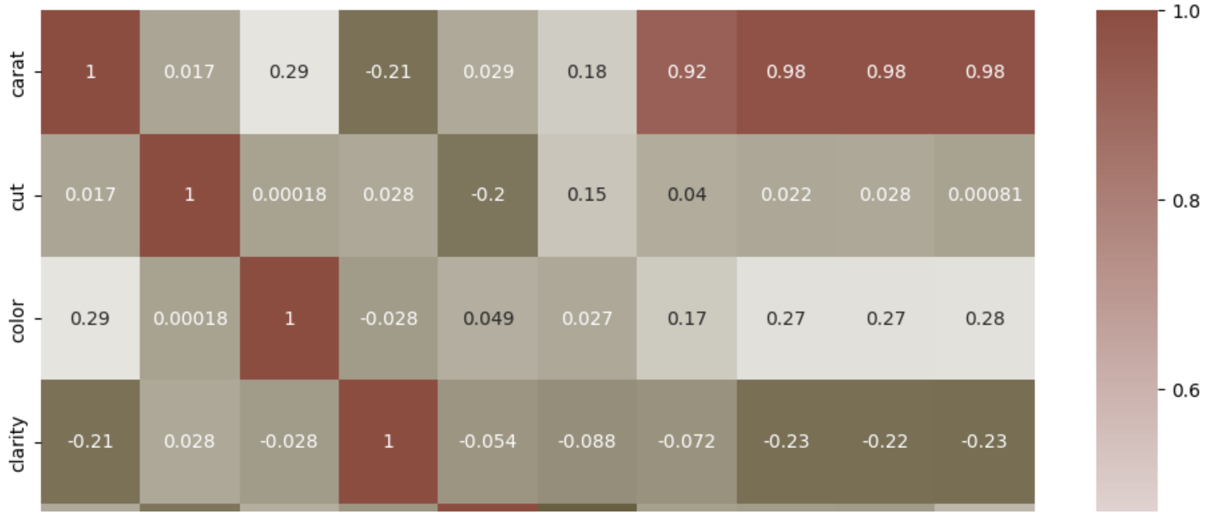
```
label_data = data.copy()

label_encoder = LabelEncoder()
for col in object_cols:
    label_data[col] = label_encoder.fit_transform(label_data[col])
label_data.head()
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	2	1	3	61.5	55.0	326	3.95	3.98	2.43
1	0.21	3	1	2	59.8	61.0	326	3.89	3.84	2.31
2	0.23	1	1	4	56.9	65.0	327	4.05	4.07	2.31
3	0.29	3	5	5	62.4	58.0	334	4.20	4.23	2.63
4	0.31	1	6	3	63.3	58.0	335	4.34	4.35	2.75

```
cmap = sns.diverging_palette(70,20,s=50, l=40, n=6,as_cmap=True)
corrmat= label_data.corr()
f, ax = plt.subplots(figsize=(12,12))
sns.heatmap(corrmat,cmap=cmap,annot=True, )
```

```
<Axes: >
```




```
X= label_data.drop(["price"],axis =1) #features 9features
y= label_data["price"] #target value
```

```
print(X)
```

```
       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

[53907 rows x 9 columns]
```

```
print(y)
```

```
0         326
1         326
2         327
3         334
4         335
         ... 
53935    2757
53936    2757
53937    2757
53938    2757
53939    2757
Name: price, Length: 53907, dtype: int64
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.25, random_state=7)
```

```
print(X_train)
```

```
       carat  cut  color  clarity  depth  table     x     y     z
8778    0.90    1      0        2   58.5   63.0  6.21  6.30  3.66
47001   0.53    2      3        7   62.5   54.0  5.19  5.21  3.25
40421   0.50    0      4        4   66.6   58.0  4.95  4.84  3.26
44023   0.51    2      1        2   62.2   57.0  5.12  5.08  3.17
13402   0.32    3      4        7   62.2   58.0  4.33  4.38  2.71
...      ...  ...    ...      ...    ...    ...   ...   ...   ...
919     0.72    0      2        4   56.9   69.0  5.93  5.77  3.33
53492   0.75    3      2        2   61.5   58.0  5.83  5.87  3.60
38492   0.42    3      2        4   61.2   58.0  4.82  4.86  2.96
10750   1.12    4      3        3   60.2   59.0  6.70  6.85  4.08
49719   0.72    2      6        7   61.5   57.0  5.74  5.77  3.54

[40430 rows x 9 columns]
```

```
print(y_train)
```

```
8778     4469
47001    1818
40421    1134
44023    1546
13402     602
         ...
919      2879
53492    2683
38492    1031
10750    4852
49719    2150
Name: price, Length: 40430, dtype: int64
```

```
from sklearn.linear_model import LinearRegression
```

```
mymodel = LinearRegression()
```

```
mymodel.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

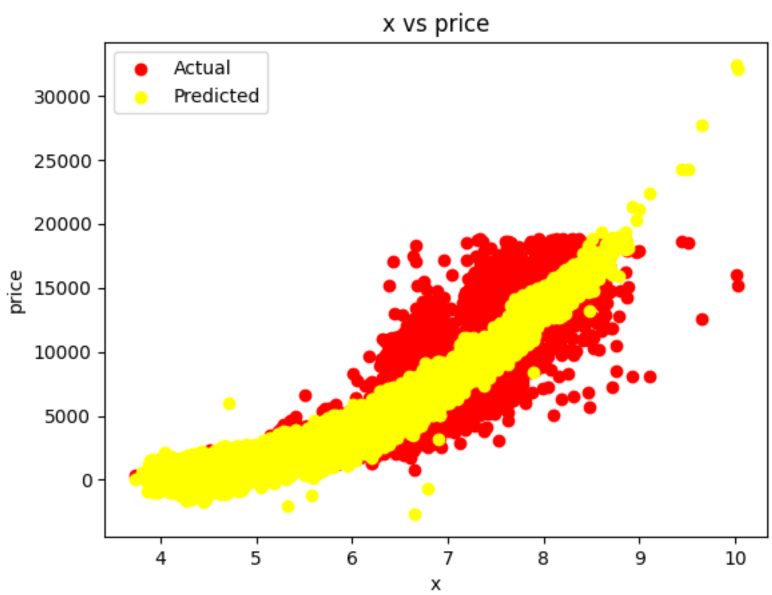
```
pred = mymodel.predict(X_test)
```

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
import numpy as np
```

```
# Model Evaluation
print("R^2:",metrics.r2_score(y_test, pred))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, pred))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
print("MAE:",metrics.mean_absolute_error(y_test, pred))
print("MSE:",metrics.mean_squared_error(y_test, pred))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
R^2: 0.8890105065854332
Adjusted R^2: 0.888936332274842
MAE: 849.3507396470712
MSE: 1741183.667805709
RMSE: 1319.5391876733745
```

```
import matplotlib.pyplot as plt
plt.scatter(X_test['x'],y_test,color="red")
plt.scatter(X_test['x'],output,color="yellow")
plt.legend(["Actual" , "Predicted"])
plt.xlabel('x')
plt.ylabel('price')
plt.title('x vs price')
plt.show()
```



```
from sklearn.tree import DecisionTreeRegressor
```

```
decisiontreemodel = DecisionTreeRegressor()
```

```
decisiontreemodel.fit(X_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
treeoutput = decisiontreemodel.predict(X_test)
```

```
print(treeoutput[:5])
```

```
[6055. 2564. 4339. 3655.  700.]
```

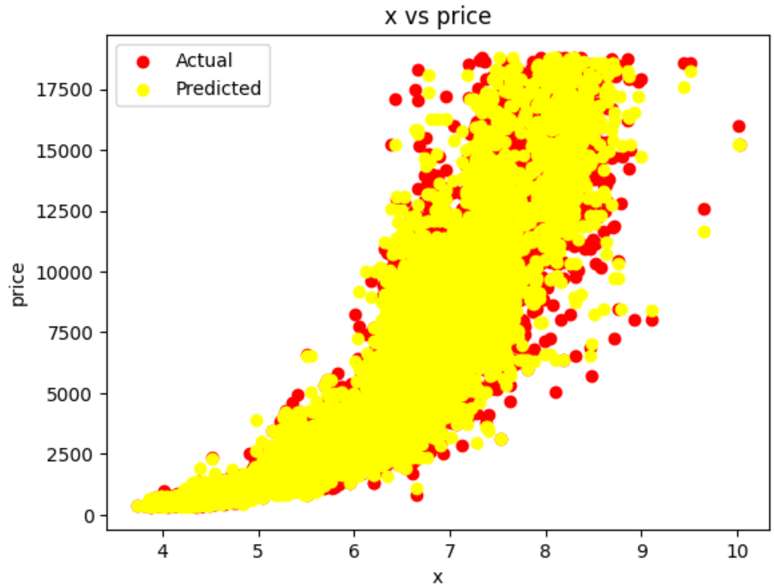
```
print(y_test[:5])
```

```
16087    6426
164      2771
5785     3903
4703     3678
28059     660
Name: price, dtype: int64
```

```
# Model Evaluation
print("R^2:",metrics.r2_score(y_test, treeoutput))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, treeoutput))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
print("MAE:",metrics.mean_absolute_error(y_test, treeoutput))
print("MSE:",metrics.mean_squared_error(y_test, treeoutput))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, treeoutput)))
```

```
R^2: 0.9647179178635343
Adjusted R^2: 0.9646943388837825
MAE: 359.7847443793129
MSE: 553499.1042145878
RMSE: 743.9752040320885
```

```
import matplotlib.pyplot as plt
plt.scatter(X_test['x'],y_test,color="red")
plt.scatter(X_test['x'],treeoutput,color="yellow")
plt.legend(["Actual" , "Predicted"])
plt.xlabel('x')
plt.ylabel('price')
plt.title('x vs price')
plt.show()
```




```
from sklearn.ensemble import RandomForestRegressor
```

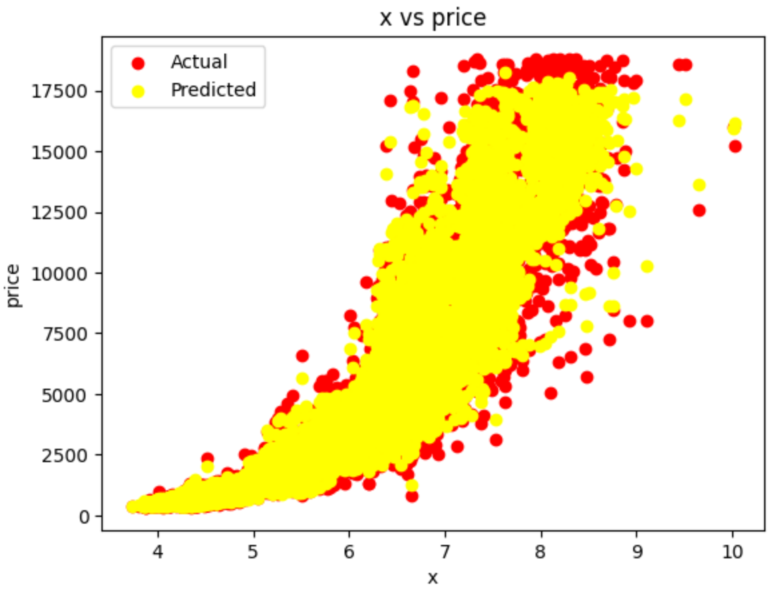
```
forestmodel = RandomForestRegressor()
```

```
forestmodel.fit(X_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
forestoutput = forestmodel.predict(X_test)
```

```
plt.scatter(X_test['x'],y_test,color="red")
plt.scatter(X_test['x'],forestoutput,color="yellow")
plt.legend(["Actual" , "Predicted"])
plt.xlabel('x')
plt.ylabel('price')
plt.title('x vs price')
plt.show()
```




```
# Model Evaluation
print("R^2:",metrics.r2_score(y_test, forestoutput))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, forestoutput))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
print("MAE:",metrics.mean_absolute_error(y_test, forestoutput))
print("MSE:",metrics.mean_squared_error(y_test, forestoutput))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, forestoutput)))
```

```
R^2: 0.9807999016737576
Adjusted R^2: 0.9807870702424859
MAE: 270.960836614055
MSE: 301207.7683880054
RMSE: 548.8239867097697
```

```
from sklearn.neighbors import KNeighborsRegressor
```

```
neighbormodel = KNeighborsRegressor()
```

```
neighbormodel.fit(X_train,y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

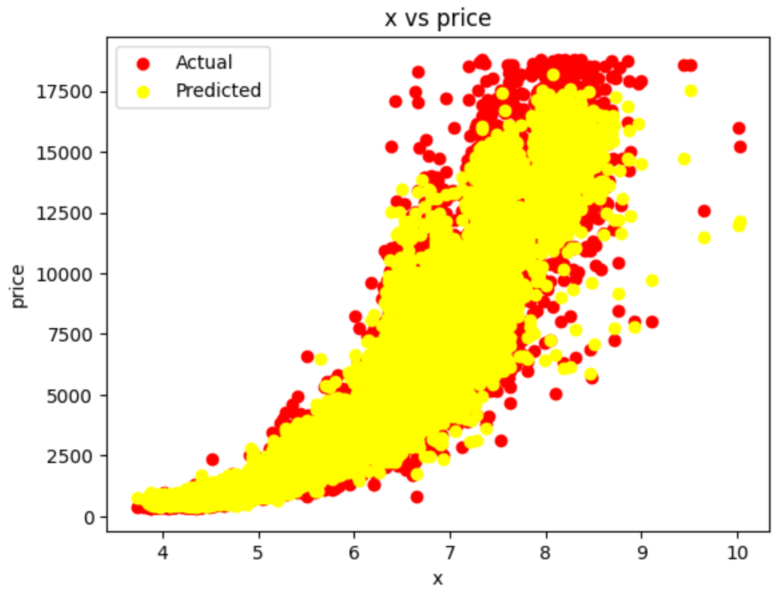
```
neighboroutput = neighbormodel.predict(X_test)
```

```
# Model Evaluation
print("R^2:",metrics.r2_score(y_test, neighboroutput))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, neighboroutput))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
print("MAE:",metrics.mean_absolute_error(y_test, neighboroutput))
print("MSE:",metrics.mean_squared_error(y_test, neighboroutput))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, neighboroutput)))
```

```
R^2: 0.9488465711208338
Adjusted R^2: 0.9488123852695
```

```
MAE: 477.70471173109746
MSE: 802486.00274245
RMSE: 895.8158308170547
```

```
plt.scatter(X_test['x'],y_test,color="red")
plt.scatter(X_test['x'],neighboroutput,color="yellow")
plt.legend(["Actual" , "Predicted"])
plt.xlabel('x')
plt.ylabel('price')
plt.title('x vs price')
plt.show()
```




```
print(X_train)
```

```
       carat  cut  color  clarity  depth  table     x     y     z
8778    0.90    1      0        2   58.5   63.0  6.21  6.30  3.66
47001   0.53    2      3        7   62.5   54.0  5.19  5.21  3.25
40421   0.50    0      4        4   66.6   58.0  4.95  4.84  3.26
44023   0.51    2      1        2   62.2   57.0  5.12  5.08  3.17
13402   0.32    3      4        7   62.2   58.0  4.33  4.38  2.71
...      ...  ...    ...      ...    ...    ...   ...   ...   ...
919     0.72    0      2        4   56.9   69.0  5.93  5.77  3.33
53492   0.75    3      2        2   61.5   58.0  5.83  5.87  3.60
38492   0.42    3      2        4   61.2   58.0  4.82  4.86  2.96
10750   1.12    4      3        3   60.2   59.0  6.70  6.85  4.08
49719   0.72    2      6        7   61.5   57.0  5.74  5.77  3.54

[40430 rows x 9 columns]
```

```
forestmodel.predict([[0.90,1,0,2,58.5,63.0,6.21,6.30,3.66]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegr
  warnings.warn(
array([4354.75])
```

Colab paid products - Cancel contracts here