In [41]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression as LR
from sklearn.model_selection import train_test_split as TTS
from sklearn.tree import DecisionTreeRegressor as DR , plot_tree as PT
from sklearn.ensemble import RandomForestRegressor as RF
from sklearn.metrics import mean_squared_error as MSE, mean_absolute_error as
import numpy as np
```

In [4]:
```
df=pd.read_csv('D:\Intern\Cognifyz Intern\Dataset .csv')
```

In [7]:
```
df.head(3)
```

Out[7]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitud
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.0275
1	6304287	Izakaya Kikufuji	162	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.0141
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.0568

3 rows × 21 columns

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

Out[5]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggr
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.0
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.6
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.5
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.0
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2.5
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3.2
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3.7
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.9

In [6]:
```
df.info()
```

```
 3   City                  9551 non-null   object
 4   Address               9551 non-null   object
 5   Locality              9551 non-null   object
 6   Locality Verbose      9551 non-null   object
 7   Longitude             9551 non-null   float64
 8   Latitude              9551 non-null   float64
 9   Cuisines              9542 non-null   object
 10  Average Cost for two  9551 non-null   int64
 11  Currency              9551 non-null   object
 12  Has Table booking     9551 non-null   object
 13  Has Online delivery   9551 non-null   object
 14  Is delivering now     9551 non-null   object
 15  Switch to order menu  9551 non-null   object
 16  Price range           9551 non-null   int64
 17  Aggregate rating      9551 non-null   float64
 18  Rating color          9551 non-null   object
 19  Rating text           9551 non-null   object
 20  Votes                 9551 non-null   int64
dtypes: float64(3), int64(5), object(13)
memory usage: 1.5+ MB
```

Split into Feature and Target

In [8]:
```
x=df[['Country Code','Average Cost for two','Price range','Votes']]
y=df['Aggregate rating']
```

Split into train Test data

In [9]:
```
x_train,x_test,y_train,y_test=TTS(x,y,test_size=0.3,random_state=203)
```

Fit in the Model-(Linear Regression)

In [11]:
```
lm=LR()
lm.fit(x_train,y_train)
```

Out[11]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

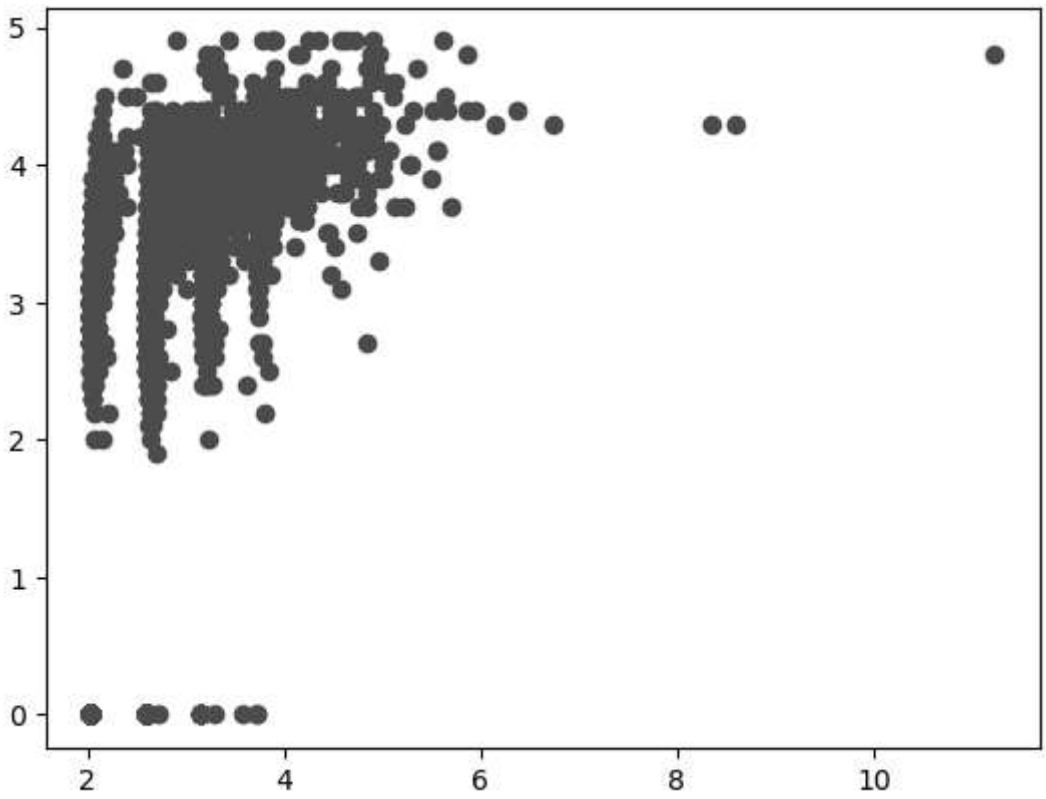
In [19]:
```
y_pred=lm.predict(x_test)
```

In [20]:
```
mse=MSE(y_pred,y_test)
print(f"Mean Squared Error is : {mse}")
mae=MAE(y_pred,y_test)
print(f"Mean Absoulte Error is : {mae}")
r2=RS(y_pred,y_test)
print(f"R2 Score is : {r2}")
```

```
Mean Squared Error is : 1.7492919932102107
Mean Absoulte Error is : 1.0953288581603926
R2 Score is : -1.825475037117438
```

In [59]:
```
plt.scatter(y_pred,y_test,c='red')
```

Out[59]:
```
<matplotlib.collections.PathCollection at 0x29b0f70a6d0>
```



In [23]:
```
print(y_pred)
```

```
[2.61310323 2.06504124 2.15843283 ... 2.66425065 2.03803603 2.63226449]
```

In [26]:
```
new_data=np.array([[162,16000,2,3500]])
predict=lm.predict(new_data)
print(f"Predicted Aggregate rating of the Restaurant is :{predict}")
```

```
Predicted Aggregate rating of the Restaurant is :[5.75168948]

C:\Users\nikhil\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning:
X does not have valid feature names, but LinearRegression was fitted with fea
ture names
  warnings.warn(
```

Predict Using Decision Tree

In [29]:
```
dm=DR(max_depth=5)
dm.fit(x_train,y_train)
```

Out[29]:
```
DecisionTreeRegressor(max_depth=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:
```
dr_y_pred=dm.predict(x_test)
```

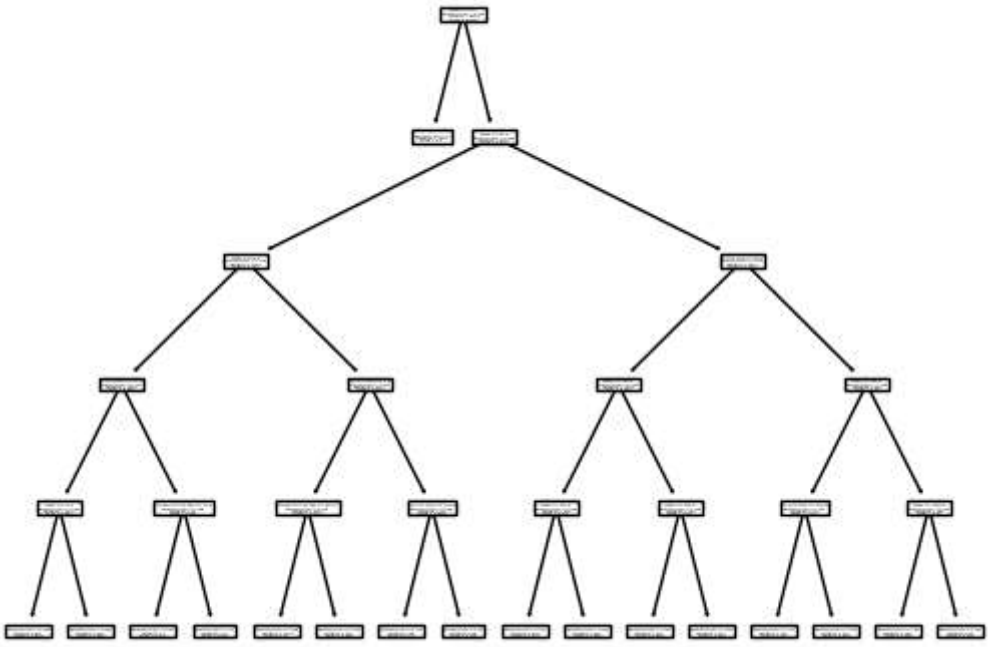
In [31]:
```
dr_mse=MSE(dr_y_pred,y_test)
print(f"Mean Squared Error is :{dr_mse}")
```

```
Mean Squared Error is :0.11432877538614103
```

In [34]:
```
PT(dm, feature_names=list(x))
```

Out[34]:
```
[Text(0.46875, 0.9166666666666666, 'Votes <= 3.5\nsquared_error = 2.288\nsamp
les = 6685\nvalue = 2.678'),
 Text(0.4375, 0.75, 'squared_error = 0.0\nsamples = 1485\nvalue = 0.0'),
 Text(0.5, 0.75, 'Votes <= 94.5\nsquared_error = 0.308\nsamples = 5200\nvalue
= 3.443'),
 Text(0.25, 0.5833333333333334, 'Votes <= 20.5\nsquared_error = 0.174\nsample
s = 3135\nvalue = 3.187'),
 Text(0.125, 0.4166666666666667, 'Country Code <= 22.0\nsquared_error = 0.067
\nsamples = 1381\nvalue = 3.028'),
 Text(0.0625, 0.25, 'Votes <= 11.5\nsquared_error = 0.052\nsamples = 1347\nva
lue = 3.012'),
 Text(0.03125, 0.08333333333333333, 'squared_error = 0.028\nsamples = 821\nva
lue = 2.964'),
 Text(0.09375, 0.08333333333333333, 'squared_error = 0.081\nsamples = 526\nva
lue = 3.086'),
 Text(0.1875, 0.25, 'Average Cost for two <= 27.5\nsquared_error = 0.255\nsam
ples = 34\nvalue = 3.659'),
 Text(0.15625, 0.08333333333333333, 'squared_error = 0.093\nsamples = 7\nvalu
e = 3.086'),
 Text(0.21875, 0.08333333333333333, 'squared_error = 0.19\nsamples = 27\nvalu
e = 3.807'),
 Text(0.375, 0.4166666666666667, 'Country Code <= 7.5\nsquared_error = 0.222
\nsamples = 1754\nvalue = 3.313'),
 Text(0.3125, 0.25, 'Average Cost for two <= 825.0\nsquared_error = 0.203\nsa
mples = 1630\nvalue = 3.275'),
 Text(0.28125, 0.08333333333333333, 'squared_error = 0.188\nsamples = 1370\nv
alue = 3.24'),
 Text(0.34375, 0.08333333333333333, 'squared_error = 0.243\nsamples = 260\nva
lue = 3.456'),
 Text(0.4375, 0.25, 'Country Code <= 175.0\nsquared_error = 0.195\nsamples =
124\nvalue = 3.818'),
 Text(0.40625, 0.08333333333333333, 'squared_error = 0.173\nsamples = 29\nval
ue = 4.241'),
 Text(0.46875, 0.08333333333333333, 'squared_error = 0.13\nsamples = 95\nvalu
e = 3.688'),
 Text(0.75, 0.5833333333333334, 'Country Code <= 25.5\nsquared_error = 0.261
\nsamples = 2065\nvalue = 3.832'),
 Text(0.625, 0.4166666666666667, 'Votes <= 263.5\nsquared_error = 0.238\nsamp
les = 1603\nvalue = 3.723'),
 Text(0.5625, 0.25, 'Votes <= 142.5\nsquared_error = 0.248\nsamples = 900\nva
lue = 3.571'),
 Text(0.53125, 0.08333333333333333, 'squared_error = 0.257\nsamples = 399\nva
lue = 3.471'),
 Text(0.59375, 0.08333333333333333, 'squared_error = 0.226\nsamples = 501\nva
lue = 3.651'),
 Text(0.6875, 0.25, 'Votes <= 626.5\nsquared_error = 0.159\nsamples = 703\nva
lue = 3.917'),
 Text(0.65625, 0.08333333333333333, 'squared_error = 0.144\nsamples = 431\nva
lue = 3.828'),
 Text(0.71875, 0.08333333333333333, 'squared_error = 0.151\nsamples = 272\nva
lue = 4.057'),
 Text(0.875, 0.4166666666666667, 'Votes <= 402.5\nsquared_error = 0.156\nsamp
les = 462\nvalue = 4.21'),
 Text(0.8125, 0.25, 'Country Code <= 215.5\nsquared_error = 0.151\nsamples =
277\nvalue = 4.077'),
 Text(0.78125, 0.08333333333333333, 'squared_error = 0.173\nsamples = 135\nva
lue = 4.217'),
```

```
 Text(0.84375, 0.08333333333333333, 'squared_error = 0.093\nsamples = 142\nva
lue = 3.944'),
 Text(0.9375, 0.25, 'Votes <= 1229.5\nsquared_error = 0.096\nsamples = 185\nv
alue = 4.41'),
 Text(0.90625, 0.08333333333333333, 'squared_error = 0.091\nsamples = 152\nva
lue = 4.359'),
 Text(0.96875, 0.08333333333333333, 'squared_error = 0.052\nsamples = 33\nval
ue = 4.645')]
```



Predict Using Random Forest

In [40]:
```
rf=RF(n_estimators=100)
rf.fit(x_train,y_train)
```

Out[40]:
```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [42]:
```
rf_y_pred=rf.predict(x_test)
```

In [43]:
```
rf_mse=MSE(rf_y_pred,y_test)
print(f"Mean squared error is : {rf_mse}")
```

```
Mean squared error is : 0.13753745563138678
```

In [47]:
```
importances = rf.feature_importances_
```

In [48]:
```
sorted_indices = np.argsort(importances)[::-1]
```

In [58]:
```
for f in range(x_train.shape[1]):
    if f <= len(sorted_indices) - 1:
        print("%d. %s (%f)" % (f + 1, x_train.columns[sorted_indices[f]], impo
    else:
        print(f"Index {f} is out of bounds (less data than expected)")
```

```
1. Votes (0.972883)
2. Average Cost for two (0.014861)
3. Country Code (0.009859)
4. Price range (0.002397)
```

In [ ]:
```

```