In [8]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score, mean_absolute_error, mean_squared_error, r2_score, confusion_matrix, classification_report
```

In [10]:
```python
df = pd.read_csv("resturant.csv")
df.head()
```

Out[10]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Price range
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...	121.027535	14.565443	French, Japanese, Desserts	...	Botswana Pula(P)	Yes	No	No	No	3
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...	121.014101	14.553708	Japanese	...	Botswana Pula(P)	Yes	No	No	No	3
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...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	...	Botswana Pula(P)	Yes	No	No	No	4
3	6318506	Ooma	162	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	...	Botswana Pula(P)	No	No	No	No	4
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	...	Botswana Pula(P)	Yes	No	No	No	4

5 rows × 21 columns

In [12]:
```python
df.columns
```

Out[12]:
```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes'],
      dtype='object')
```

In [14]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Restaurant ID         9551 non-null   int64  
 1   Restaurant Name       9551 non-null   object 
 2   Country Code          9551 non-null   int64  
 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
```

In [16]:
```python
df.dtypes
```

Out[16]:
```
Restaurant ID             int64
Restaurant Name          object
Country Code              int64
City                     object
Address                  object
Locality                 object
Locality Verbose         object
Longitude               float64
Latitude                float64
Cuisines                 object
Average Cost for two      int64
Currency                 object
Has Table booking        object
Has Online delivery      object
Is delivering now        object
Switch to order menu     object
Price range               int64
Aggregate rating        float64
Rating color             object
Rating text              object
Votes                     int64
dtype: object
```

In [18]:
```
df.describe()
```

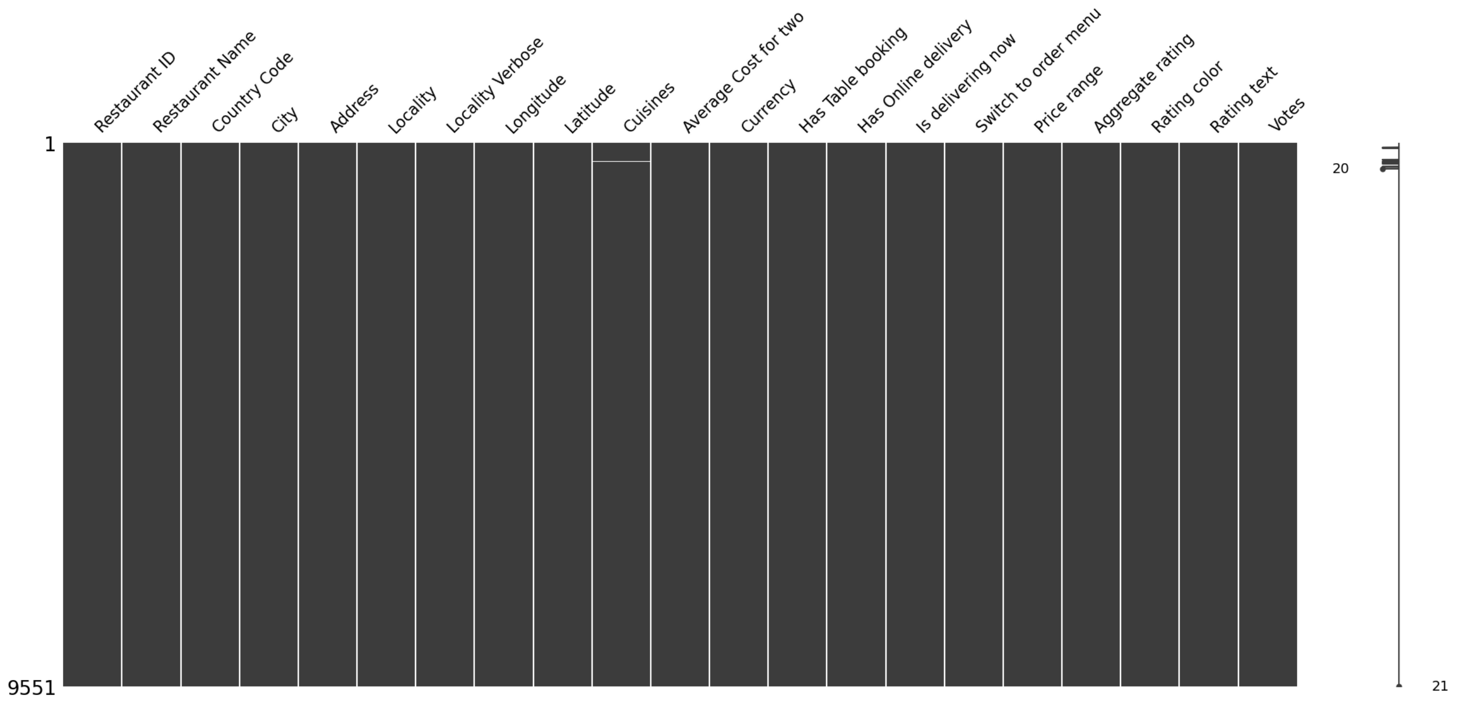
Out[18]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.666370	156.909748
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.516378	430.169145
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.000000	0.000000
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2.500000	5.000000
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3.200000	31.000000
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3.700000	131.000000
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	10934.000000

In [23]:
```
!pip install missingno
```

```
Requirement already satisfied: missingno in c:\users\sguha\anaconda3\lib\site-packages (0.5.2)
Requirement already satisfied: numpy in c:\users\sguha\anaconda3\lib\site-packages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in c:\users\sguha\anaconda3\lib\site-packages (from missingno) (3.9.2)
Requirement already satisfied: scipy in c:\users\sguha\anaconda3\lib\site-packages (from missingno) (1.13.1)
Requirement already satisfied: seaborn in c:\users\sguha\anaconda3\lib\site-packages (from missingno) (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\sguha\anaconda3\lib\site-packages (from matplotlib->missingno) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in c:\users\sguha\anaconda3\lib\site-packages (from seaborn->missingno) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sguha\anaconda3\lib\site-packages (from pandas>=1.2->seaborn->missingno) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\sguha\anaconda3\lib\site-packages (from pandas>=1.2->seaborn->missingno) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\sguha\anaconda3\lib\site-packages (from python-dateutil>=2.7->missingno) (1.16.0)
```

In [25]:
```
import missingno as msno
msno.matrix(df)
plt.show()
```



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

Out[27]:
```
(9551, 21)
```

In [29]:
```
df.isnull().sum()
```

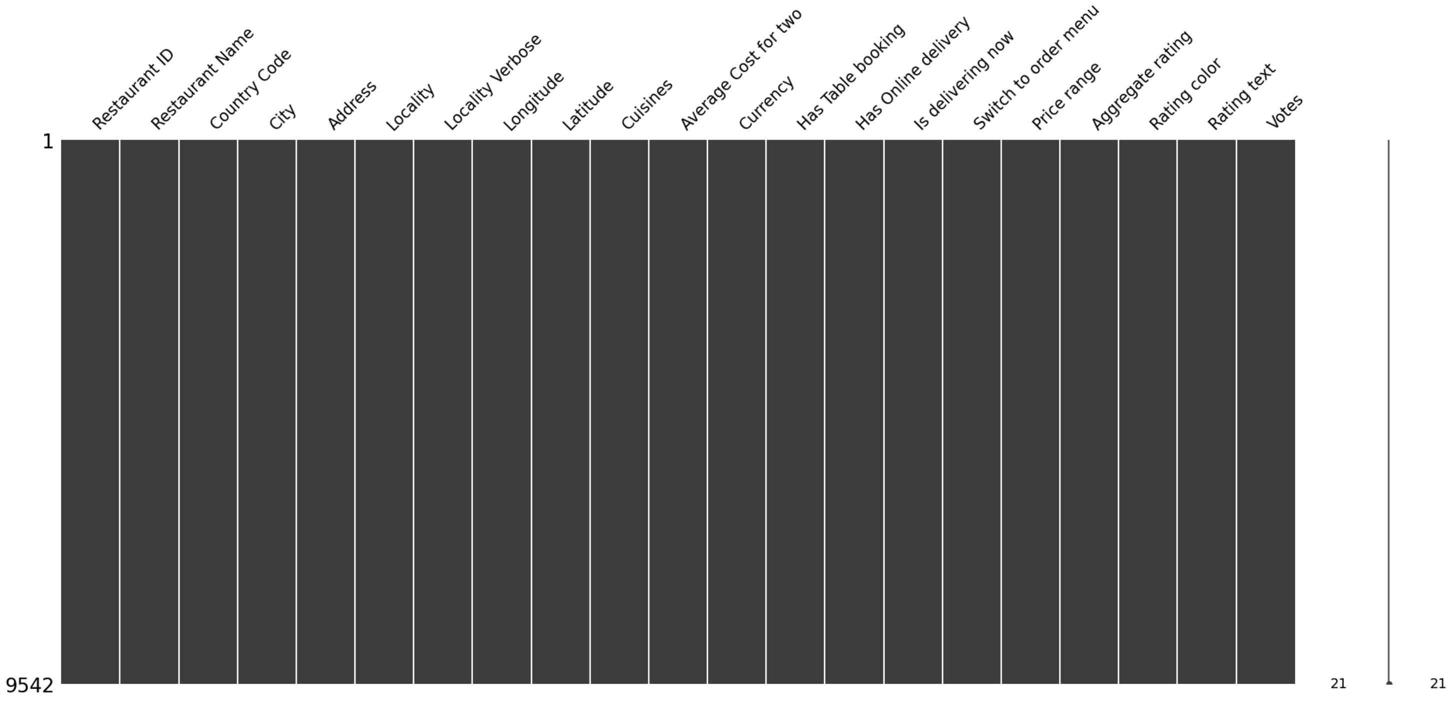
```
Out[29]: Restaurant ID           0
         Restaurant Name         0
         Country Code            0
         City                    0
         Address                 0
         Locality                0
         Locality Verbose        0
         Longitude               0
         Latitude                0
         Cuisines                9
         Average Cost for two    0
         Currency                0
         Has Table booking       0
         Has Online delivery     0
         Is delivering now       0
         Switch to order menu    0
         Price range             0
         Aggregate rating        0
         Rating color            0
         Rating text             0
         Votes                   0
         dtype: int64
```

```
In [31]: df = df.dropna()
```

```
In [33]: df.shape
```

```
Out[33]: (9542, 21)
```

```
In [37]: msno.matrix(df)
         plt.show()
```



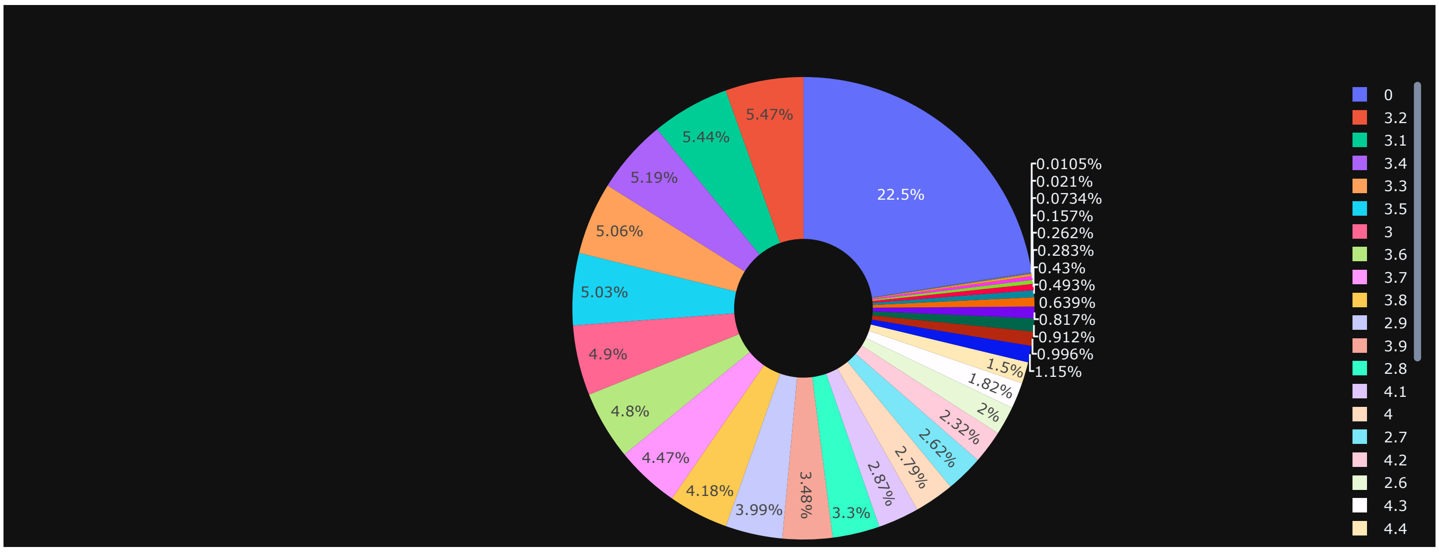
```
In [39]: df['Aggregate rating'].describe()
```

```
Out[39]: count    9542.000000
         mean        2.665238
         std         1.516588
         min         0.000000
         25%         2.500000
         50%         3.200000
         75%         3.700000
         max         4.900000
         Name: Aggregate rating, dtype: float64
```

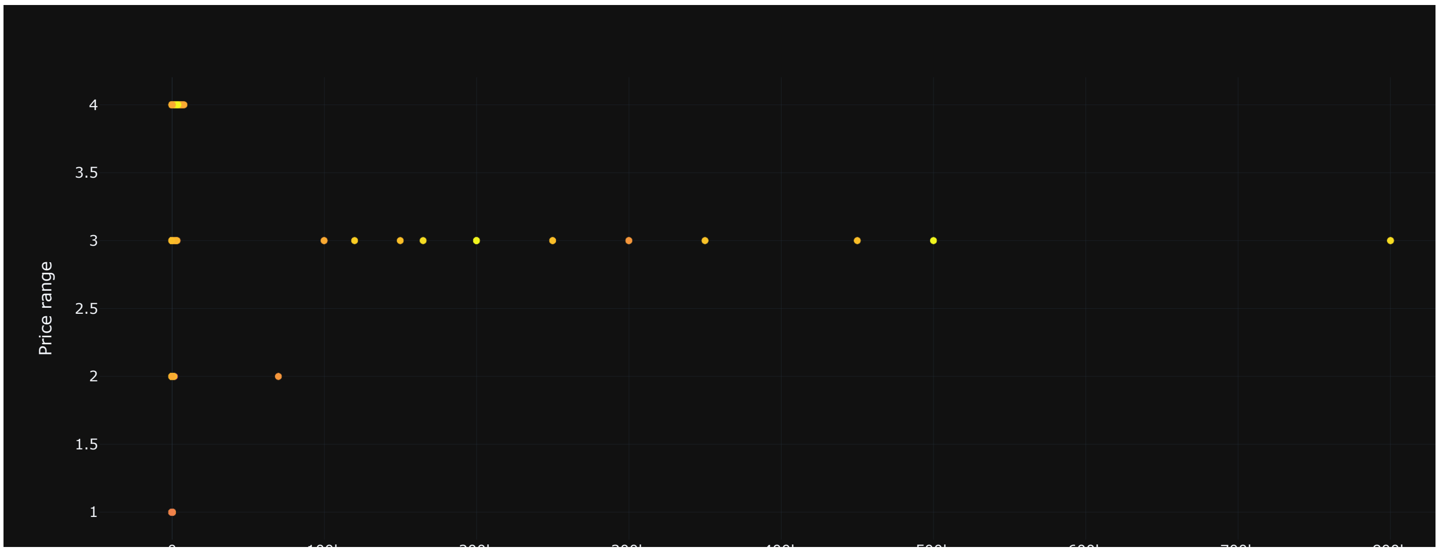
```
In [41]: df['Aggregate rating'].value_counts()
```

```
Out[41]: Aggregate rating
         0.0    2148
         3.2     522
         3.1     519
         3.4     495
         3.3     483
         3.5     480
         3.0     468
         3.6     458
         3.7     427
         3.8     399
         2.9     381
         3.9     332
         2.8     315
         4.1     274
         4.0     266
         2.7     250
         4.2     221
         2.6     191
         4.3     174
         4.4     143
         2.5     110
         4.5      95
         2.4      87
         4.6      78
         4.9      61
         2.3      47
         4.7      41
         2.2      27
         4.8      25
         2.1      15
         2.0       7
         1.9       2
         1.8       1
         Name: count, dtype: int64
```

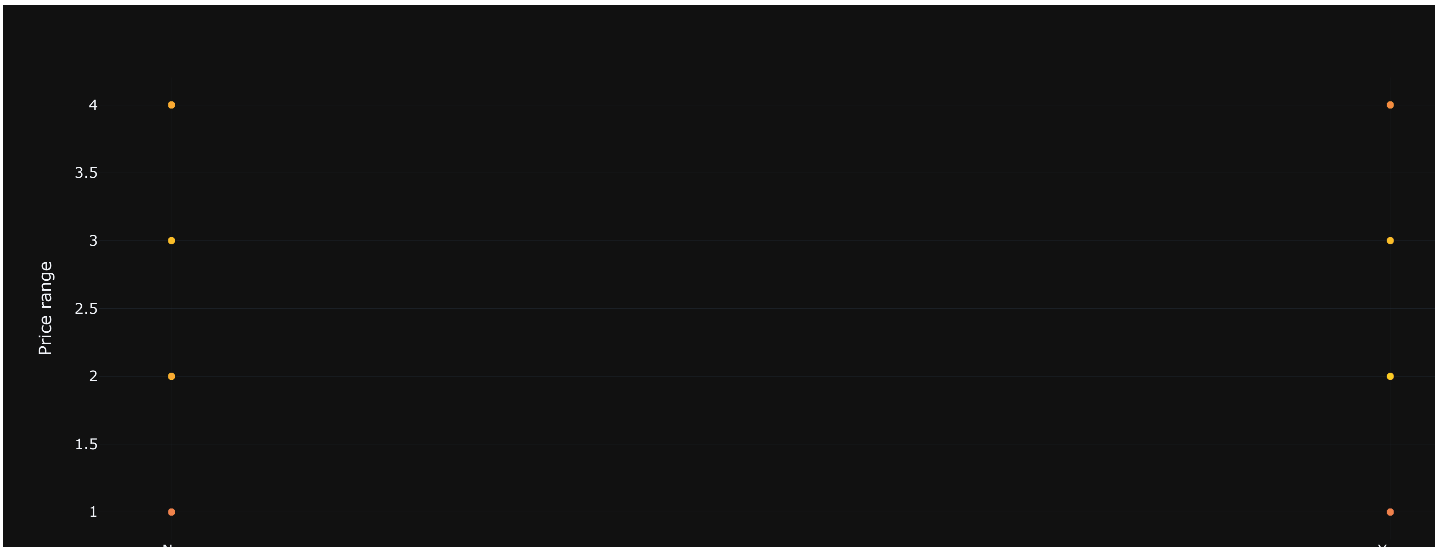
```
In [43]: import plotly.express as px
         fig = px.pie(df,names ="Aggregate rating",hole = 0.3,template ="plotly_dark")
         fig.show()
```



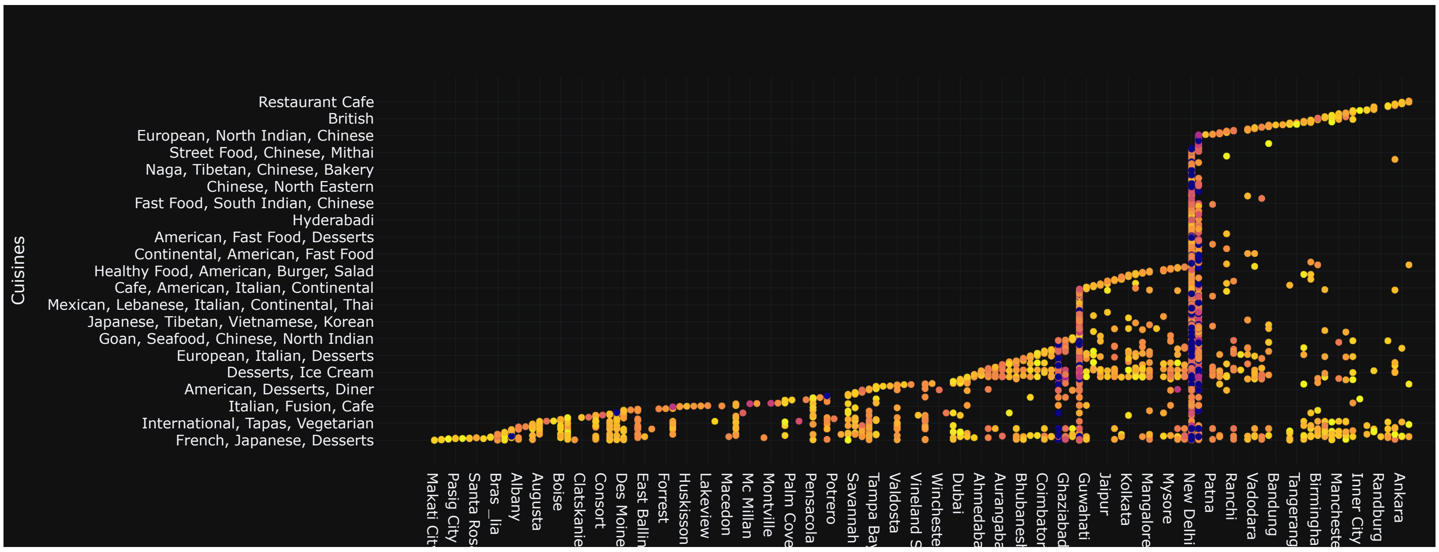
```
In [45]: fig = px.scatter(df,x ="Average Cost for two",y="Price range",color= "Aggregate rating",template="plotly_dark")
         fig.show()
```



In [47]:
```python
fig = px.scatter(df,x ="Has Online delivery",y="Price range",color= "Aggregate rating",template="plotly_dark")
fig.show()
```



In [49]:
```python
fig = px.scatter(df,x ="City",y="Cuisines",color= "Aggregate rating",template="plotly_dark")
fig.show()
```



In [51]:
```python
from sklearn.preprocessing import LabelEncoder


#encoded_df = df.copy()

label_encoder = LabelEncoder()

columns_to_encode = ['Restaurant Name', 'City', 'Address', 'Locality', 'Locality Verbose', 'Cuisines', 'Currency', 'Rating color', 'Rating text']

# Encode categorical columns
for column in columns_to_encode:
    df[column] = label_encoder.fit_transform(df[column])

# Map 'Yes' and 'No' to numerical values for binary categorical columns
binary_columns = ['Has Table booking', 'Has Online delivery', 'Is delivering now', 'Switch to order menu']
binary_mapping = {'No': 0, 'Yes': 1}

# Encode binary categorical columns
for column in binary_columns:
    df[column] = df[column].map(binary_mapping)

# Print first few rows of encoded DataFrame
print(df.head())
```

```
   Restaurant ID  Restaurant Name  Country Code  City  Address  Locality  \
0        6317637             3743           162    74     8680       175
1        6304287             3168           162    74     6049       596
2        6300002             2894           162    76     4678       312
3        6318506             4701           162    76     8685       864
4        6314302             5516           162    76     8684       864

   Locality Verbose   Longitude   Latitude  Cuisines  ...  Currency  \
0               176  121.027535  14.565443       920  ...         0
1               604  121.014101  14.553708      1111  ...         0
2               318  121.056831  14.581404      1671  ...         0
3               877  121.056475  14.585318      1126  ...         0
4               877  121.057508  14.584450      1122  ...         0

   Has Table booking  Has Online delivery  Is delivering now  \
0                  1                    0                  0
1                  1                    0                  0
2                  1                    0                  0
3                  0                    0                  0
4                  1                    0                  0

   Switch to order menu  Price range  Aggregate rating  Rating color  \
0                     0            3               4.8             0
1                     0            3               4.5             0
2                     0            4               4.4             1
3                     0            4               4.9             0
4                     0            4               4.8             0

   Rating text  Votes
0            1    314
1            1    591
2            5    270
3            1    365
4            1    229

[5 rows x 21 columns]
```

In [53]:
```python
# drop features that inhibit model building
df = df.drop('Restaurant ID', axis=1)
df = df.drop('Restaurant Name', axis=1)
df = df.drop('Country Code', axis=1)
df = df.drop('City', axis=1)
df = df.drop('Address', axis=1)
df = df.drop('Locality', axis=1)
df = df.drop('Locality Verbose', axis=1)
df = df.drop('Longitude', axis=1)
df = df.drop('Latitude', axis=1)
df = df.drop('Cuisines', axis=1)
df = df.drop('Currency', axis=1)
```

In [55]:
```python
print(df.describe())
```

```
       Average Cost for two  Has Table booking  Has Online delivery  \
count           9542.000000        9542.000000          9542.000000
mean            1200.326137           0.121358             0.256864
std            16128.743876           0.326560             0.436927
min                0.000000           0.000000             0.000000
25%              250.000000           0.000000             0.000000
50%              400.000000           0.000000             0.000000
75%              700.000000           0.000000             1.000000
max           800000.000000           1.000000             1.000000

       Is delivering now  Switch to order menu  Price range  Aggregate rating  \
count        9542.000000                9542.0  9542.000000       9542.000000
mean            0.003563                   0.0     1.804968          2.665238
std             0.059589                   0.0     0.905563          1.516588
min             0.000000                   0.0     1.000000          0.000000
25%             0.000000                   0.0     1.000000          2.500000
50%             0.000000                   0.0     2.000000          3.200000
75%             0.000000                   0.0     2.000000          3.700000
max             1.000000                   0.0     4.000000          4.900000

       Rating color  Rating text         Votes
count   9542.000000  9542.000000   9542.000000
mean       2.952840     1.788933    156.772060
std        1.492629     1.694795    430.203324
min        0.000000     0.000000      0.000000
25%        2.000000     0.000000      5.000000
50%        2.000000     2.000000     31.000000
75%        4.000000     3.000000    130.000000
max        5.000000     5.000000  10934.000000
```

In [57]:
```python
df
```

Out[57]:

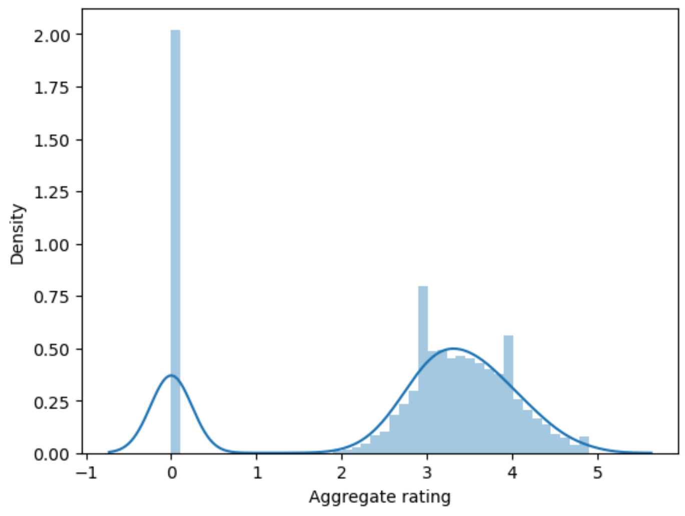
	Average Cost for two	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Price range	Aggregate rating	Rating color	Rating text	Votes
0	1100	1	0	0	0	3	4.8	0	1	314
1	1200	1	0	0	0	3	4.5	0	1	591
2	4000	1	0	0	0	4	4.4	1	5	270
3	1500	0	0	0	0	4	4.9	0	1	365
4	1500	1	0	0	0	4	4.8	0	1	229
...	...	...	...	...	...	...	...	...	...	...
9546	80	0	0	0	0	3	4.1	1	5	788
9547	105	0	0	0	0	3	4.2	1	5	1034
9548	170	0	0	0	0	4	3.7	5	2	661
9549	120	0	0	0	0	4	4.0	1	5	901
9550	55	0	0	0	0	2	4.0	1	5	591

9542 rows × 10 columns

In [59]:
```python
sns.distplot(df['Aggregate rating'])
```

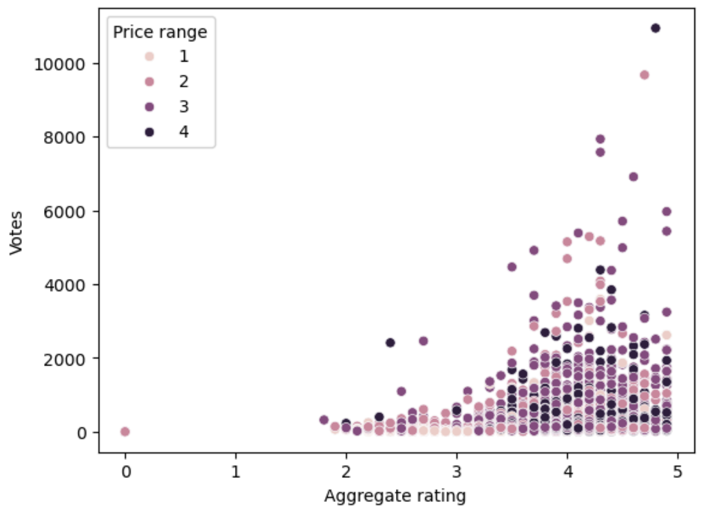
```
C:\Users\sguha\AppData\Local\Temp\ipykernel_7664\95774152.py:1: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
Out[59]: <Axes: xlabel='Aggregate rating', ylabel='Density'>
```

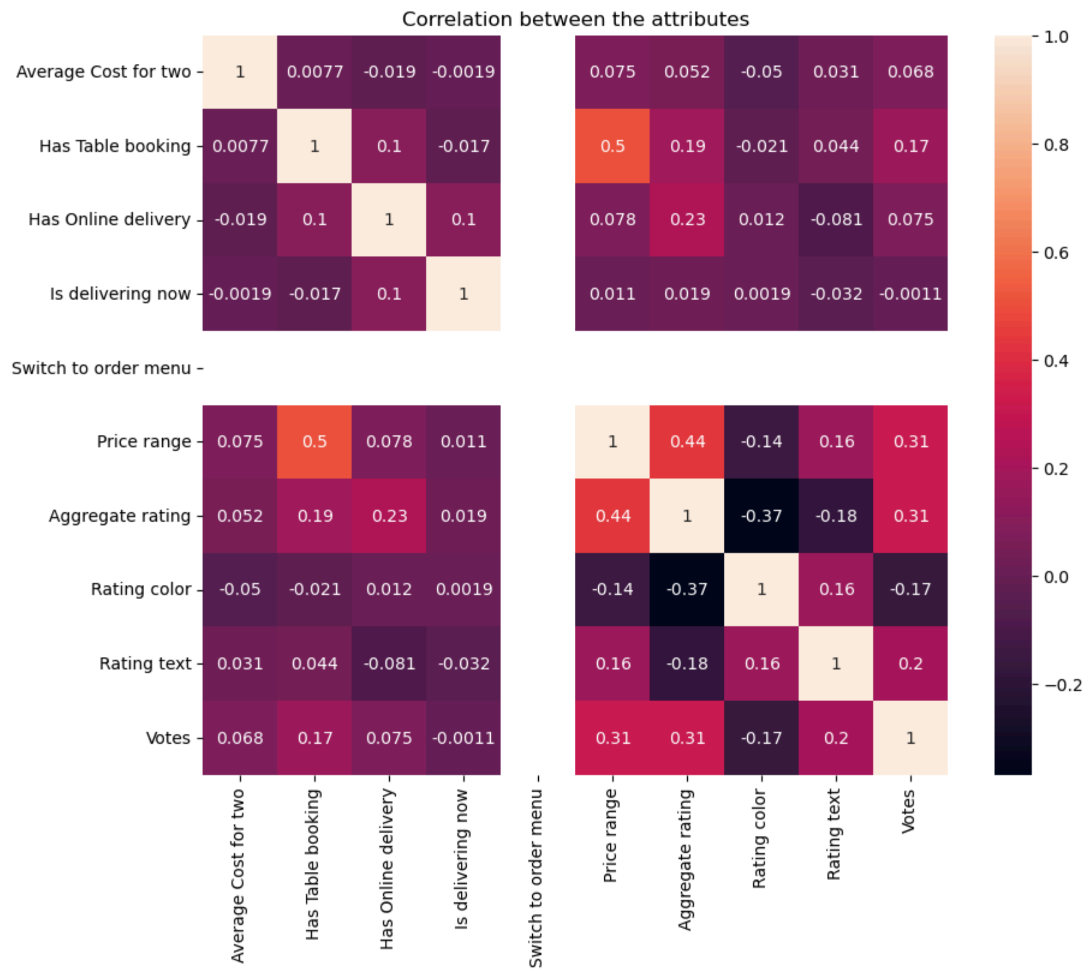


```
In [61]: sns.scatterplot(x=df["Aggregate rating"],y=df["Votes"],hue=df["Price range"])
```

```
Out[61]: <Axes: xlabel='Aggregate rating', ylabel='Votes'>
```



```
In [63]: plt.figure(figsize=(10,8))
         sns.heatmap(df.corr(),annot=True)
         plt.title("Correlation between the attributes")
         plt.show()
```



```
In [65]: x = df.drop('Aggregate rating', axis=1)
         y = df['Aggregate rating']
```

```
In [67]: #Data Splitting
```

```
In [69]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=250)
         x_train.head()
         y_train.head()
```

```
Out[69]: 9453    3.8
         3230    3.9
         6529    4.2
         2576    2.8
         1907    3.0
         Name: Aggregate rating, dtype: float64
```

```
In [71]: print("x_train: ", x_train.shape)
         print("x_test: ",x_test.shape)
         print("y_train: ",y_train.shape)
         print("y_test: ",y_test.shape)
```

```
x_train:  (8587, 9)
x_test:  (955, 9)
y_train:  (8587,)
y_test:  (955,)
```

```
In [73]: #training by linear regression algorithm
         linreg = LinearRegression()
         linreg.fit(x_train,y_train)
         linreg_pred=linreg.predict(x_test)
```

```
In [75]: #evaluating performance metrics of linear regression
         linreg_mae = mean_absolute_error(y_test, linreg_pred)
         linreg_mse = mean_squared_error(y_test, linreg_pred)
         linreg_r2 = r2_score(y_test, linreg_pred)
         print(f"Mean Absolute Error of the linear regression model is: {linreg_mae:.2f}")
         print(f"Mean Squared Error of the linear regression model is: {linreg_mse:.2f}")
         print(f"R2 score of the linear regression model is: {linreg_r2:.2f}")
```

```
Mean Absolute Error of the linear regression model is: 0.99
Mean Squared Error of the linear regression model is: 1.43
R2 score of the linear regression model is: 0.38
```

```
In [77]: # training by decision tree regressor algorithm
         dtree = DecisionTreeRegressor()
         dtree.fit(x_train, y_train)
         dtree_pred = dtree.predict(x_test)
```

```
In [79]: #evaluating performance metrics of decision tree
         dtree_mae = mean_absolute_error(y_test, dtree_pred)
         dtree_mse = mean_squared_error(y_test, dtree_pred)
         dtree_r2 = r2_score(y_test, dtree_pred)
         print(f"Mean Absolute Error of the decision tree model is: {dtree_mae:.2f}")
         print(f"Mean Squared Error of the decision tree model is: {dtree_mse:.2f}")
         print(f"R2 score of the decision tree model is: {dtree_r2:.2f}")
```

```
Mean Absolute Error of the decision tree model is: 0.14
Mean Squared Error of the decision tree model is: 0.05
R2 score of the decision tree model is: 0.98
```

```
In [ ]:
```