restaurants.03

August 18, 2024

0.1 *Predictive Modeling*

0.1.1 Import Libraries

[ ]:
```
# Importing Libraries
import pandas as pd
import numpy as np

# Visualization Libraries
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# Ignore all warnings
import warnings
warnings.filterwarnings('ignore')
```

0.1.2 Dataset Loading

[ ]:
```
# Load Dataset
df = pd.read_csv("Dataset.csv")
```

0.1.3 Build a Regression Model

[ ]:
```
# Creating a regression model to predict the aggregate rating of a restaurant␣
 ↪based on available features
# Import necessary libraries for data splitting, regression, and evaluation
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

[ ]:
```
# Convert categorical variables to numeric using one-hot encoding
df = pd.get_dummies(df, columns=['Has Table booking', 'Has Online delivery'],␣
 ↪drop_first=True)
```

[ ]:
```
# Select features and target variable
features = ['Average Cost for two', 'Votes', 'Price range', 'Has Table␣
 ↪booking_Yes', 'Has Online delivery_Yes']
target = 'Aggregate rating'

X = df[features]
y = df[target]
```

Split the Dataset into Training and Testing Sets

[ ]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

Experiment with Different Algorithms

[ ]:
```
# Experimenting with different algorithms (e.g., linear regression, decision␣
 ↪trees, random forest)
# And compare their performance by evaluating the model's performance using␣
 ↪appropriate metrics

# Define a dictionary of regression models to experiment with
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor()
}

# Iterate through each model, train, evaluate, and print results
for model_name, model in models.items():
    # Train the model on the training data
    model.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = model.predict(X_test)

    # Calculate Mean Squared Error (MSE) to evaluate model accuracy
    mse = mean_squared_error(y_test, y_pred)

    # Calculate R-squared (R2) to assess the goodness of fit
    r2 = r2_score(y_test, y_pred)

    # Print model performance metrics
    print(f"Model: {model_name}")
    print(f"Mean Squared Error: {mse}")
    print(f"R-squared: {r2}")
    print("-----------------------------")
```

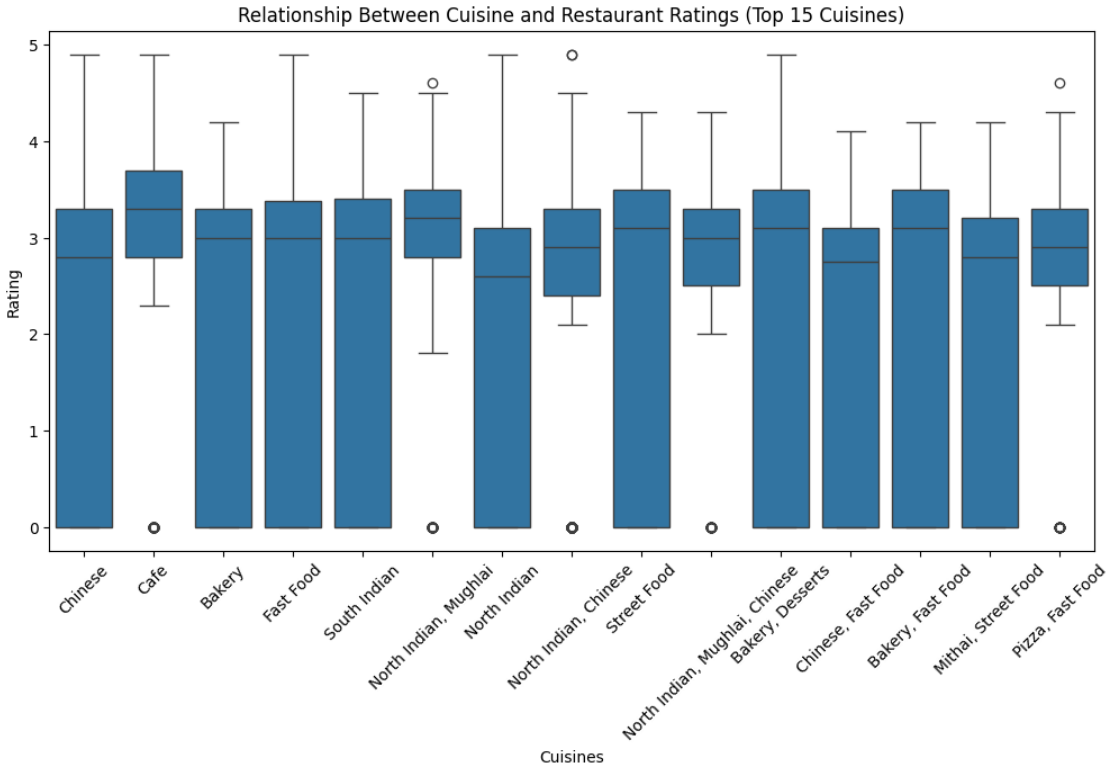
```
Model: Linear Regression
```

```
Mean Squared Error: 1.6764802747031442
R-squared: 0.2634446409021949
---------------------------
Model: Decision Tree
Mean Squared Error: 0.2065120259095257
R-squared: 0.9092697112533938
---------------------------
Model: Random Forest
Mean Squared Error: 0.13589278641425256
R-squared: 0.940296010870826
---------------------------
```

0.2 *Customer Preference Analysis*

0.2.1 Relationship Between the Type of Cuisine and the Restaurant's Rating

[ ]:
```
# Analyzing the relationship between the type of cuisine and the restaurant's␣
 ↪rating
# There are many cuisine names present in the data, so i select only the top 15␣
 ↪cuisines
top_n = 15
top_cuisines = df['Cuisines'].value_counts().nlargest(top_n).index

# Filter the dataset to include only the top 15 cuisines
df_filtered = df[df['Cuisines'].isin(top_cuisines)]

# Create a box plot
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_filtered, x='Cuisines', y='Aggregate rating')

# Set labels
plt.title(f'Relationship Between Cuisine and Restaurant Ratings (Top {top_n}␣
 ↪Cuisines)')
plt.xlabel('Cuisines')
plt.ylabel('Rating')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Display Chart
plt.show()
```



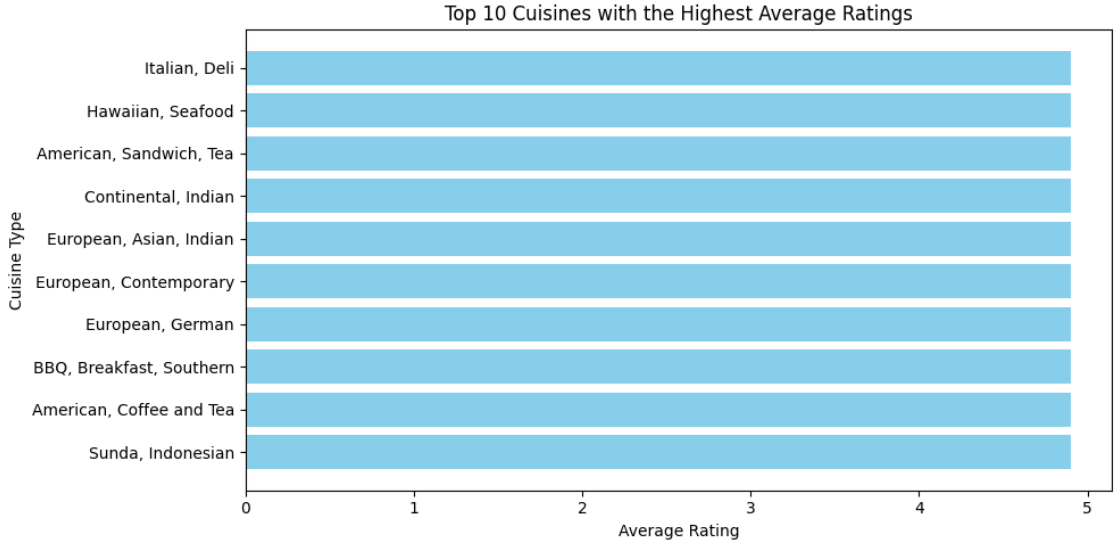
0.2.2 Most Popular Cuisines by Number of Votes

[ ]:
```
# Identifying most popular cuisines based on number of votes
top_cuisines = df.groupby('Cuisines')['Votes'].sum().nlargest(10)

# Display result
print("Top Cuisines by Number of Votes:")
print(top_cuisines)
```

```
Top Cuisines by Number of Votes:
Cuisines
North Indian, Mughlai                53747
North Indian                         46241
North Indian, Chinese                42012
Cafe                                 30657
Chinese                              21925
North Indian, Mughlai, Chinese       20115
Fast Food                            17852
South Indian                         16433
Mughlai, North Indian                15275
Italian                              14799
Name: Votes, dtype: int64
```

0.2.3 Cusines with Higher Ratings

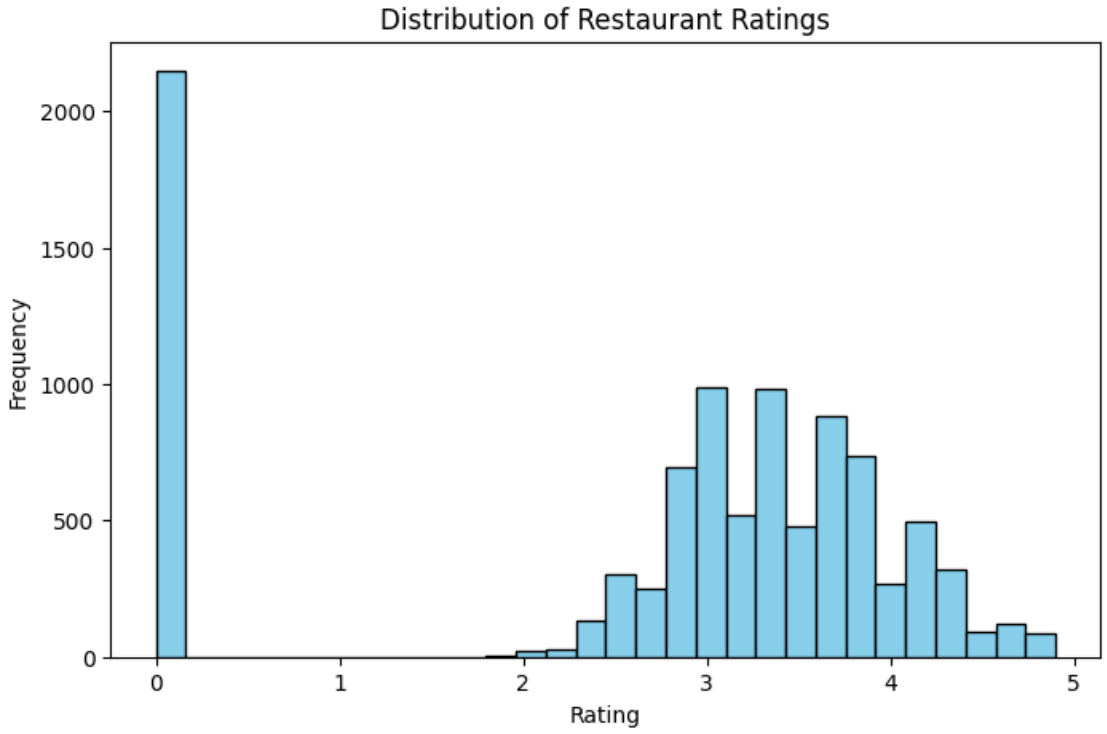
[ ]:
```
# Determine if there are any specific cuisines that tend to receive higher␣
 ↪ratings
# Group the data by cuisine and calculate the mean rating for each cuisine
cuisine_ratings = df.groupby('Cuisines')['Aggregate rating'].mean().
 ↪reset_index()

# Sort cuisines by rating in descending order
cuisine_ratings = cuisine_ratings.sort_values(by='Aggregate rating',␣
 ↪ascending=False)

# Count Plot Visualization Code for the cuisines with the highest ratings
plt.figure(figsize=(10, 5))
plt.barh(cuisine_ratings['Cuisines'][:10], cuisine_ratings['Aggregate rating'][:
 ↪10], color='skyblue')

# Set labels
plt.xlabel('Average Rating')
plt.ylabel('Cuisine Type')
plt.title('Top 10 Cuisines with the Highest Average Ratings')
plt.gca().invert_yaxis()  # To display the highest rating at the top
plt.tight_layout()

# Display Chart
plt.show()
```



0.3 *Data Visualization*

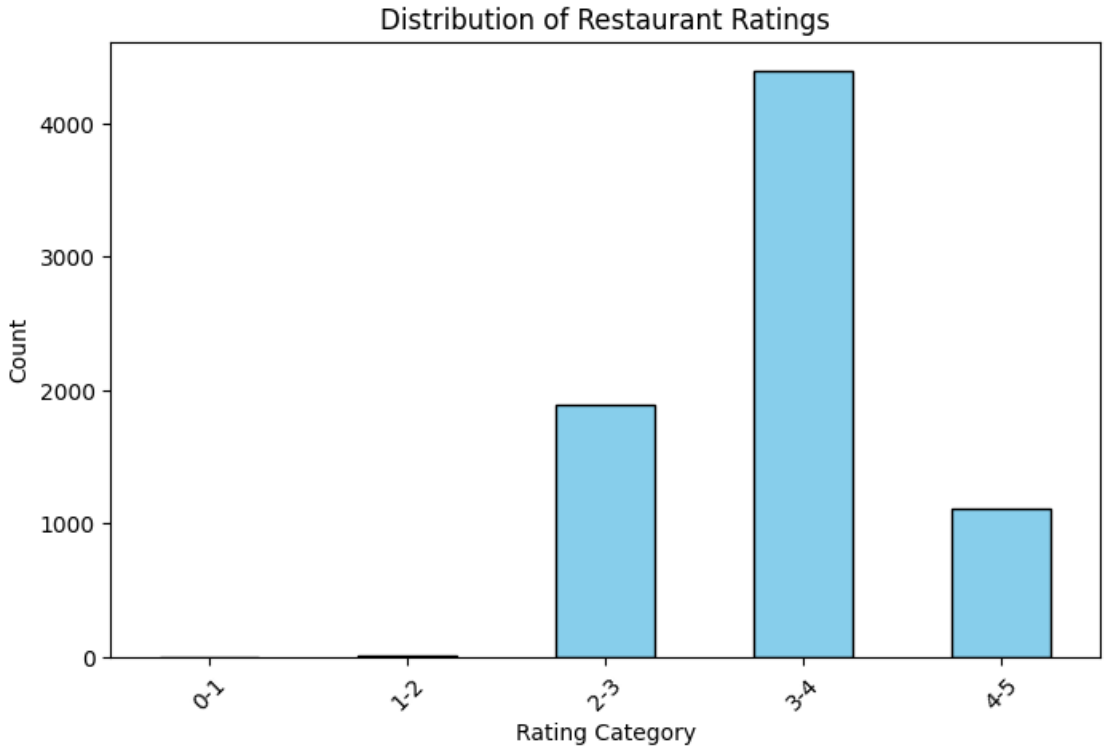
0.3.1 Distribution of Ratings

[ ]:
```
# Distribution of ratings by using a histogram
plt.figure(figsize=(8, 5))
plt.hist(df['Aggregate rating'], bins=30, color='skyblue', edgecolor='black')

# Add labels and title
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('Distribution of Restaurant Ratings')

# Display Chart
plt.show()
```



[ ]:
```
# Distribution of ratings by using a bar plot
# Group ratings into categories (e.g., 0-1, 1-2, 2-3, etc.)
bins = [0, 1, 2, 3, 4, 5]
labels = ['0-1', '1-2', '2-3', '3-4', '4-5']
df['Rating Category'] = pd.cut(df['Aggregate rating'], bins=bins, labels=labels)

# Count the number of restaurants in each rating category
rating_counts = df['Rating Category'].value_counts().sort_index()
```

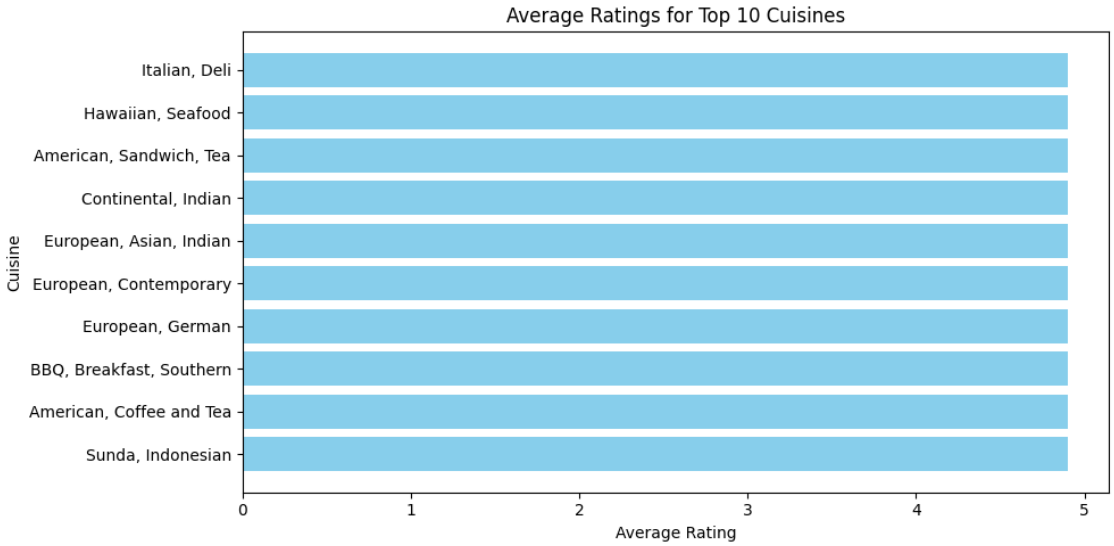
```
# Create a bar plot
plt.figure(figsize=(8, 5))
rating_counts.plot(kind='bar', color='skyblue', edgecolor='black')

# Add labels and title
plt.xlabel('Rating Category')
plt.ylabel('Count')
plt.title('Distribution of Restaurant Ratings')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Display Chart
plt.show()
```



0.3.2 Comparing Average Ratings for Different Cuisines

[ ]:
```
# Comparing average ratings of different cuisines by using a bar plot
# Group the data by cuisine and calculate the mean rating for each cuisine
cuisine_ratings = df.groupby('Cuisines')['Aggregate rating'].mean().
 ↪reset_index()
```

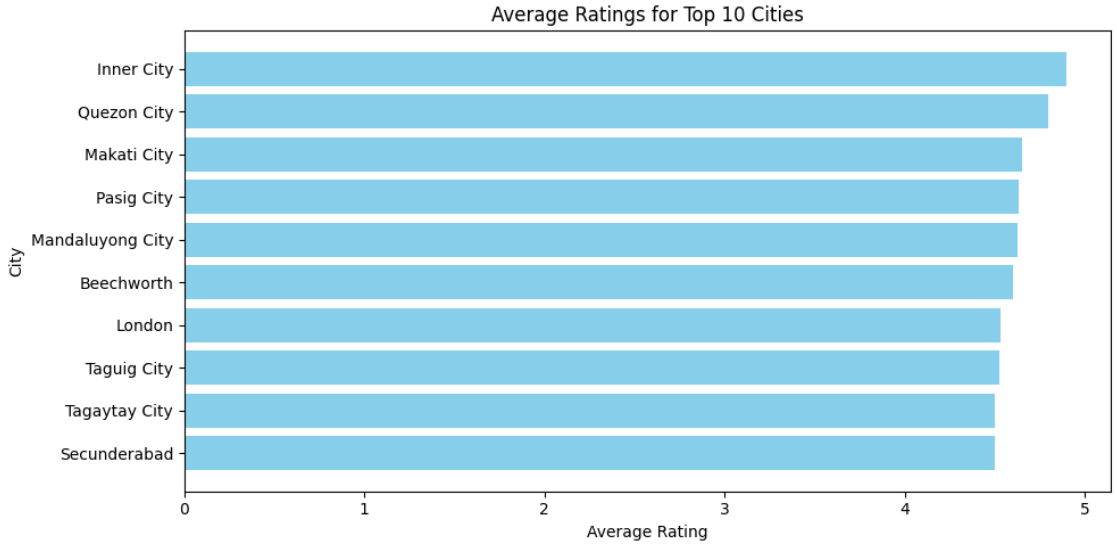
```
# Sort cuisines by rating in descending order
cuisine_ratings = cuisine_ratings.sort_values(by='Aggregate rating',␣
 ↪ascending=False)

# Count Plot Visualization Code for the cuisines with the highest ratings
plt.figure(figsize=(10, 5))
plt.barh(cuisine_ratings['Cuisines'][:10], cuisine_ratings['Aggregate rating'][:
 ↪10], color='skyblue')

# Add labels and title
plt.ylabel('Cuisine')
plt.xlabel('Average Rating')
plt.title('Average Ratings for Top 10 Cuisines')
plt.gca().invert_yaxis()  # To display the highest rating at the top
plt.tight_layout()

# Display Chart
plt.show()
```



0.3.3 Comparing Average Ratings for Different Cities

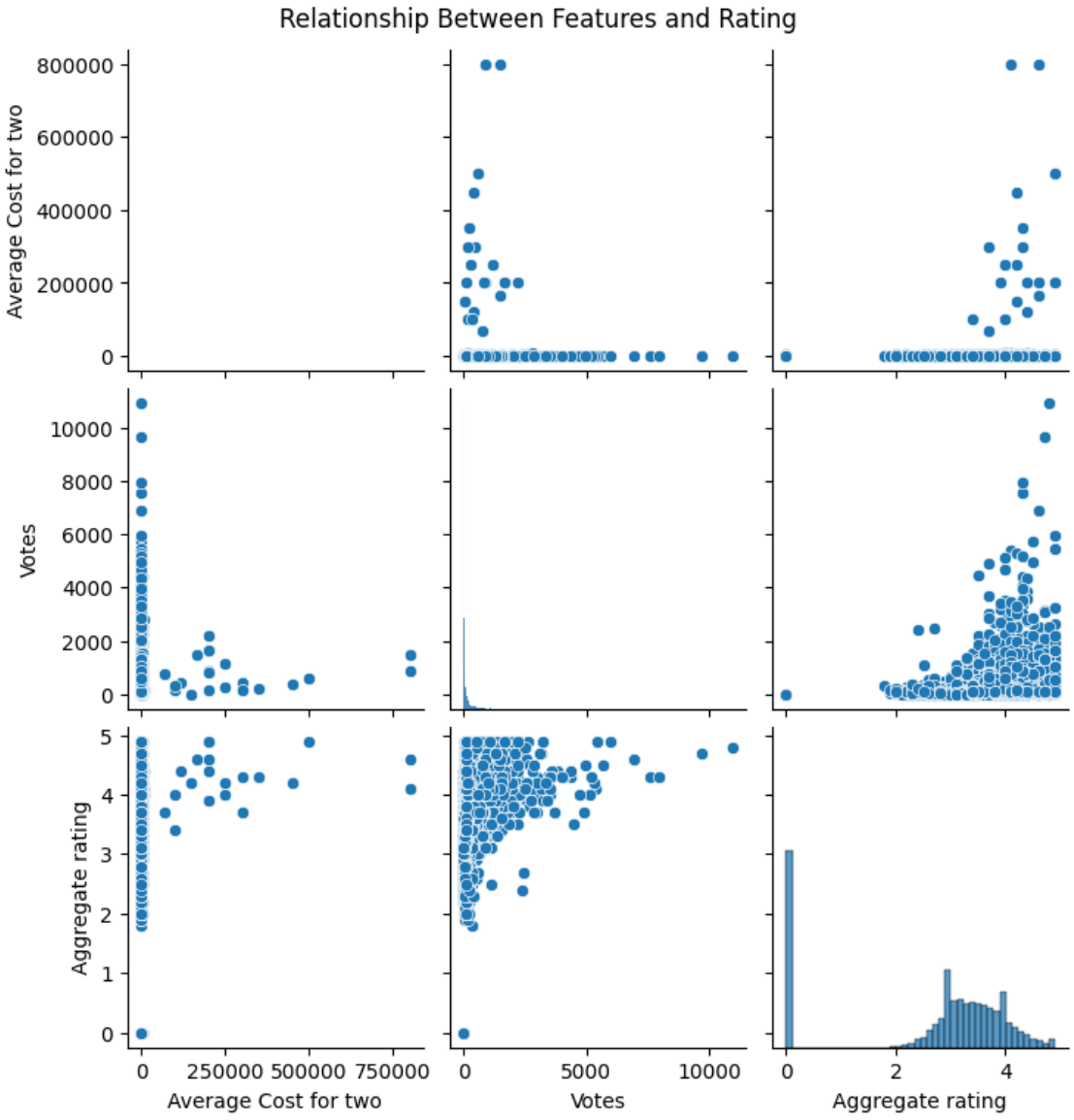
[ ]:
```
# Comparing average ratings of different cities by using a bar plot
# Group the data by city and calculate the average rating for each city
city_ratings = df.groupby('City')['Aggregate rating'].mean().reset_index()

# Sort by average rating in descending order
city_ratings = city_ratings.sort_values(by='Aggregate rating', ascending=False)
```

```
# Count Plot Visualization Code for the cities with the highest ratings
plt.figure(figsize=(10, 5))
plt.barh(city_ratings['City'][:10], city_ratings['Aggregate rating'][:10],␣
 ↪color='skyblue')

# Add labels and title
plt.ylabel('City')
plt.xlabel('Average Rating')
plt.title(f'Average Ratings for Top 10 Cities')
plt.gca().invert_yaxis()  # To display the highest rating at the top
plt.tight_layout()

# Display Chart
plt.show()
```



0.3.4 Relationship Between Various Features and the Target Variable

[ ]:
```
# Visualizing the relationship between various features and the target variable␣
 ↪by using pair plot
sns.pairplot(data=df, vars=['Average Cost for two', 'Votes', 'Aggregate␣
 ↪rating'])
plt.suptitle("Relationship Between Features and Rating", y=1.02)

# Display Chart
plt.show()
```


Relationship Between Features and Rating
Average Cost for two
Votes
Aggregate rating
0
200000
400000
600000
800000
2000
4000
6000
8000
10000
1
2
3
4
5
250000
500000
750000
5000