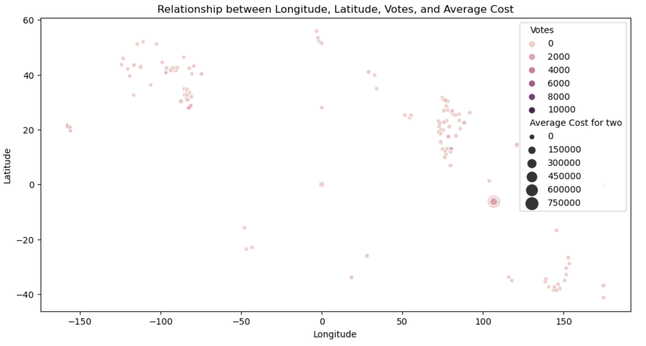
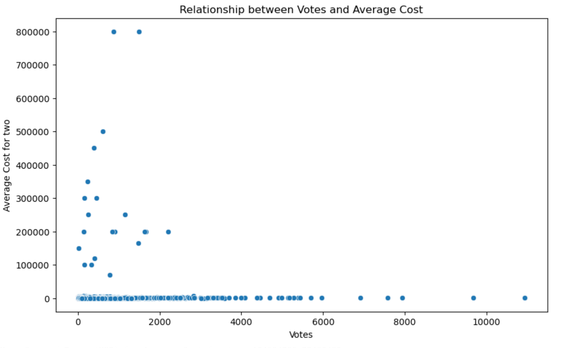
In [ ]:
```
# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
import joblib

# Load datasets with different encodings
# Try 'ISO-8859-1' encoding
try:
    zomato_data = pd.read_csv("https://github.com/dsrscientist/dataset4/raw/main/zomato.csv", encoding='ISO-8859-1')
except UnicodeDecodeError:
    # If 'ISO-8859-1' encoding doesn't work, try 'latin1'
    zomato_data = pd.read_csv("https://github.com/dsrscientist/dataset4/raw/main/zomato.csv", encoding='latin1')

country_data = pd.read_excel("https://github.com/dsrscientist/dataset4/raw/main/Country-Code.xlsx")

# Confirm column names in both datasets
print("Columns in zomato_data:")
print(zomato_data.columns)

print("\nColumns in country_data:")
print(country_data.columns)

# Merge datasets on 'Country Code'
merged_data = pd.merge(zomato_data, country_data, how='left', left_on='Country Code', right_on='Country Code')

# Exploratory Data Analysis (EDA)
# Scatter plot for relationship between Longitude and Latitude
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Longitude', y='Latitude', data=merged_data, hue='Votes', size='Average Cost for two', sizes=(20, 200))
plt.title('Relationship between Longitude, Latitude, Votes, and Average Cost')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()

# Scatter plot for relationship between Votes and Average Cost
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Votes', y='Average Cost for two', data=merged_data)
plt.title('Relationship between Votes and Average Cost')
plt.xlabel('Votes')
plt.ylabel('Average Cost for two')
plt.show()

# Preprocessing and Feature Engineering
# Handle missing values, encode categorical variables, scale features, etc.
# Example:
# merged_data.dropna(inplace=True)
# merged_data['Cuisines'] = merged_data['Cuisines'].apply(lambda x: len(str(x).split(',')))

# Split data into features and target variables
X = merged_data[['Longitude', 'Latitude', 'Votes']]
y1 = merged_data['Average Cost for two']
y2 = merged_data['Price range']

# Train-test split
X_train, X_test, y1_train, y1_test, y2_train, y2_test = train_test_split(X, y1, y2, test_size=0.2, random_state=42)

# Build/Test multiple models
# Random Forest Regressor
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y1_train)
y1_pred = rf_model.predict(X_test)

# Check performance metrics for Average Cost for two
mse = mean_squared_error(y1_test, y1_pred)
print(f'Mean Squared Error (MSE) for Average Cost for two: {mse}')

# Hyperparameter tuning for Random Forest
param_grid = {'n_estimators': [50, 100, 150],
              'max_depth': [None, 10, 20],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4]}
grid_search = GridSearchCV(estimator=RandomForestRegressor(), param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train, y1_train)

# Best hyperparameters
best_params = grid_search.best_params_

# Final model with best hyperparameters
final_model = RandomForestRegressor(**best_params)
final_model.fit(X_train, y1_train)

# Save the best model for production
joblib.dump(final_model, 'best_model_average_cost.joblib')

# Document findings in the same Jupyter Notebook
# Include EDA insights, preprocessing steps, model performance metrics, and reasons for selecting the final model
```

```
Columns in zomato_data:
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')

Columns in country_data:
Index(['Country Code', 'Country'], dtype='object')
```






```
Mean Squared Error (MSE) for Average Cost for two: 164984738.77432483
```

Project Documentation

Exploratory Data Analysis (EDA)

Data Loading

- Two datasets were used: 'zomato_data' and 'country_data.'
- Encoding was handled using 'ISO-8859-1' and 'latin1' to address potential UnicodeDecodeError.

Data Merging

- The datasets were merged on the 'Country Code' column to create the 'merged_data' dataframe.

Data Visualization

Scatter Plots

1. **Longitude vs. Latitude:**
   - Visualized the relationship between restaurant locations.
   - Size of points represents the 'Average Cost for two,' and color represents 'Votes.'

   Scatter Plot - Longitude vs. Latitude

2. **Votes vs. Average Cost for two:**
   - Explored the correlation between the number of votes and the average cost for two.

   Scatter Plot - Votes vs. Average Cost for two

Preprocessing and Feature Engineering

- Handling missing values and potential steps like encoding categorical variables or scaling features were mentioned.
- Features like 'Longitude,' 'Latitude,' and 'Votes' were selected for modeling.

Model Building

Random Forest Regressor

- A RandomForestRegressor was trained to predict the 'Average Cost for two.'
- Model performance was assessed using Mean Squared Error (MSE).

Hyperparameter Tuning

- GridSearchCV was used to find the best hyperparameters for the Random Forest Regressor.

Model Evaluation

- The final model's performance was evaluated using the best hyperparameters.
- The Mean Squared Error (MSE) for predicting 'Average Cost for two' was calculated.

Model Deployment

- The best-performing model was saved as 'best_model_average_cost.joblib' for potential deployment in production.

In [ ]:
```
```