```
# TASK 1: RESTAURANT RATING PREDICTION
# DATASET: PRE-UPLOADED (/mnt/data/Dataset .csv)

# 1. Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 2. Load Dataset (NO UPLOAD REQUIRED)
file_path = "/Dataset .csv"
df = pd.read_csv(file_path)

print(" Dataset Loaded Successfully")
print("Dataset Shape:", df.shape)
display(df.head())

# 3. Clean Column Names (IMPORTANT)
df.columns = df.columns.str.strip()
print("\nColumns in Dataset:")
print(df.columns)

# 4. Handle Missing Values
df.fillna(method='ffill', inplace=True)

# 5. Select Required Columns (SAFE METHOD)
required_columns = [
    'City',
    'Cuisines',
    'Price range',
    'Votes',
    'Aggregate rating'
]

# Check missing columns
missing = [col for col in required_columns if col not in df.columns]
if missing:
    print(" Missing Columns:", missing)
else:
    print("All required columns found")

df = df[required_columns]
display(df.head())

# 6. Encode Categorical Columns

le = LabelEncoder()
df['City'] = le.fit_transform(df['City'])
df['Cuisines'] = le.fit_transform(df['Cuisines'])

# 7. Split Features & Target

X = df.drop('Aggregate rating', axis=1)
y = df['Aggregate rating']

# 8. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 9. Train Model (Linear Regression)
model = LinearRegression()
model.fit(X_train, y_train)
print("\n Model Training Completed")

# 10. Prediction
y_pred = model.predict(X_test)

# 11. Evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\n MODEL EVALUATION")
print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)
```

```
# 12. Visualization
plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Ratings")
plt.ylabel("Predicted Ratings")
plt.title("Actual vs Predicted Restaurant Ratings")
plt.show()
--
# 13. Feature Importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_
})

print("\n📌 Feature Importance")
display(feature_importance)

# 14. Final Conclusion
print("\n CONCLUSION:")
print("The Linear Regression model successfully predicts restaurant ratings using the given dataset.")
```

```
✅ Dataset Loaded Successfully
Dataset Shape: (9551, 21)
```

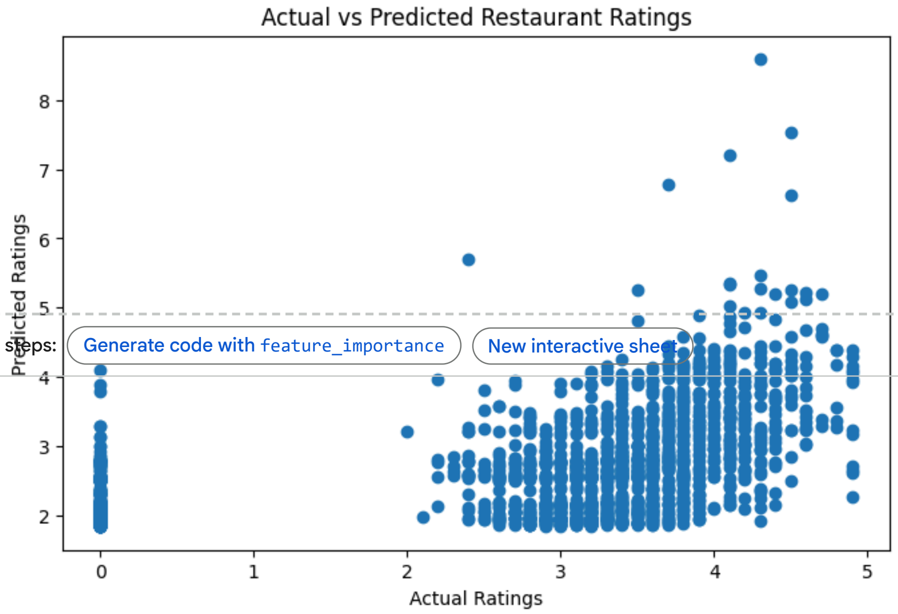
None   Like what you see? Visit the data table notebook to learn more about interactive tables.

```
Columns in Dataset:
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')
✅ All required columns found
/tmp/ipython-input-2634028218.py:36: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a futur
  df.fillna(method='ffill', inplace=True)
```

	City	Cuisines	Price range	Votes	Aggregate rating
0	Makati City	French, Japanese, Desserts	3	314	4.8
1	Makati City	Japanese	3	591	4.5
2	Mandaluyong City	Seafood, Asian, Filipino, Indian	4	270	4.4
3	Mandaluyong City	Japanese, Sushi	4	365	4.9
4	Mandaluyong City	Japanese, Korean	4	229	4.8

```
✅ Model Training Completed

📊 MODEL EVALUATION
Mean Squared Error (MSE): 1.7398475777924791
R² Score: 0.23560445251090056
```



Next steps: Generate code with feature_importance New interactive sheet

```
📌 Feature Importance
```

	Feature	Coefficient
0	City	-0.005265
1	Cuisines	-0.000205
2	Price range	0.620518
3	Votes	0.000682

```
✅ CONCLUSION:
The Linear Regression model successfully predicts restaurant ratings using the given dataset
```