In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression, LogisticRegressi
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score, mean_squared_error, cla

import streamlit as st
```

In [2]:
```
df = pd.read_csv("../apple/Music/zomato.csv")
```

In [3]:
```
df
```

Out[3]:

	url	address	name
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village

...	...	...	...
51712	https://www.zomato.com/bangalore/best-brews-fo...	Four Points by Sheraton Bengaluru, 43/3, White...	Best Brews - Fou Points by Sherato Bengaluru...
51713	https://www.zomato.com/bangalore/vinod-bar-and...	Number 10, Garudachar Palya, Mahadevapura, Whi...	Vinod Ba An Restauran
51714	https://www.zomato.com/bangalore/plunge-sherat...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Plunge - Sherato Gran Bengalur Whitefiel H...
51715	https://www.zomato.com/bangalore/chime-sherato...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Chime - Sherato Gran Bengalur Whitefiel Ho...
51716	https://www.zomato.com/bangalore/the-nest-the-...	ITPL Main Road, KIADB Export Promotion Industr...	The Nest - The Der Bengalur

51717 rows × 17 columns

```
In [4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   url                          51717 non-null  object
 1   address                      51717 non-null  object
 2   name                         51717 non-null  object
 3   online_order                 51717 non-null  object
 4   book_table                   51717 non-null  object
 5   rate                         43942 non-null  object
 6   votes                        51717 non-null  int64 
 7   phone                        50509 non-null  object
 8   location                     51696 non-null  object
 9   rest_type                    51490 non-null  object
 10  dish_liked                   23639 non-null  object
 11  cuisines                     51672 non-null  object
 12  approx_cost(for two people)  51371 non-null  object
 13  reviews_list                 51717 non-null  object
 14  menu_item                    51717 non-null  object
 15  listed_in(type)              51717 non-null  object
 16  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
url                                0
address                            0
name                               0
online_order                       0
book_table                         0
rate                            7775
votes                              0
phone                           1208
location                          21
rest_type                        227
dish_liked                     28078
cuisines                          45
approx_cost(for two people)      346
reviews_list                       0
menu_item                          0
listed_in(type)                    0
listed_in(city)                    0
dtype: int64
```

In [6]:
```
df.drop(columns=['url', 'phone', 'menu_item', 'reviews_list'], inpl

df.head()
```

Out[6]:

	address	name	online_order	book_table	rate	votes	locat
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	Banashan
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	Banashan
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	Banashan
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashan
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	Basavanag

In [7]:
```
df['rate'] = df['rate'].astype(str).apply(lambda x: x.split('/')[0]
df['rate'] = pd.to_numeric(df['rate'], errors='coerce')  # Convert

df.fillna({'rate': df['rate'].median()}, inplace=True)

df['rate'].isnull().sum()
```

Out[7]:
```
0
```

In [8]:
```
df.fillna({'cuisines': "Unknown", 'rest_type': "Unknown"}, inplace=
```

In [9]:
```
df['approx_cost(for two people)'] = (
    df['approx_cost(for two people)']
    .astype(str)
    .str.replace(',', '', regex=True)  # Ensure regex=True to avoid
    .astype(float)  # Convert to numeric
)

df.fillna({'approx_cost(for two people)': df['approx_cost(for two p
```

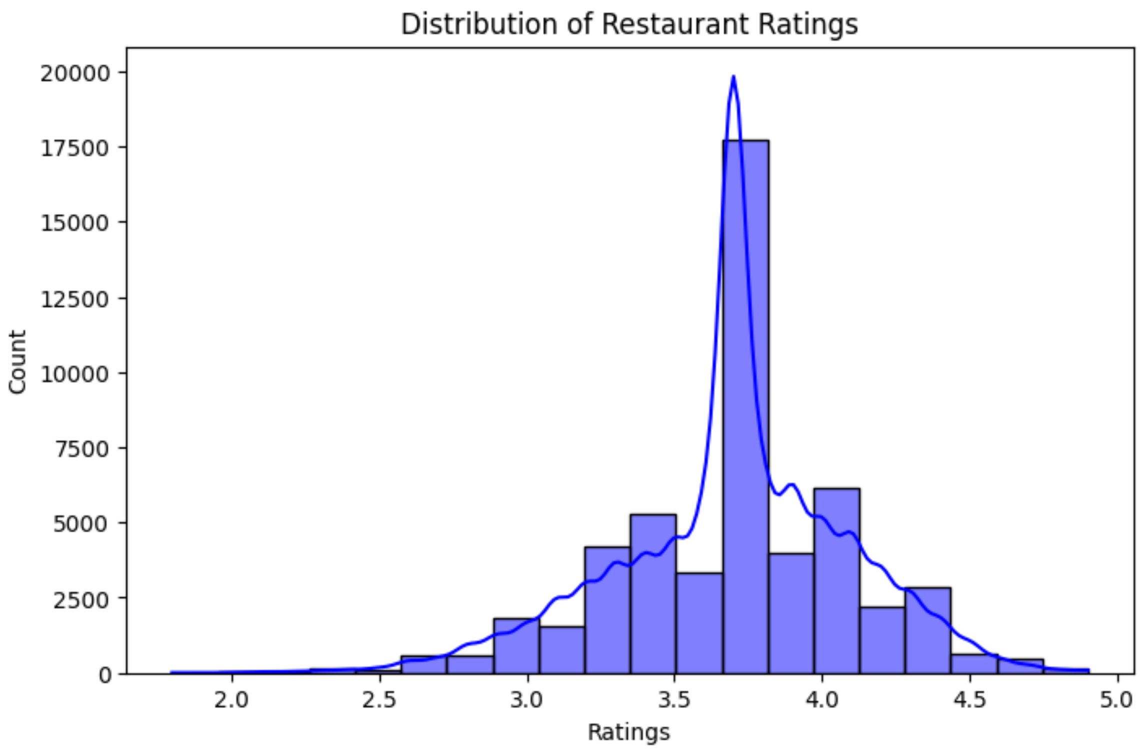
In [10]:
```
df.drop(columns=['dish_liked'], inplace=True)
```

In [11]:
```
df.dropna(inplace=True)
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
address                        0
name                           0
online_order                   0
book_table                     0
rate                           0
votes                          0
location                       0
rest_type                      0
cuisines                       0
approx_cost(for two people)    0
listed_in(type)                0
listed_in(city)                0
dtype: int64
```

In [13]:
```
plt.figure(figsize=(8,5))
sns.histplot(df['rate'], bins=20, kde=True, color="blue")
plt.title("Distribution of Restaurant Ratings")
plt.xlabel("Ratings")
plt.ylabel("Count")
plt.show()
```

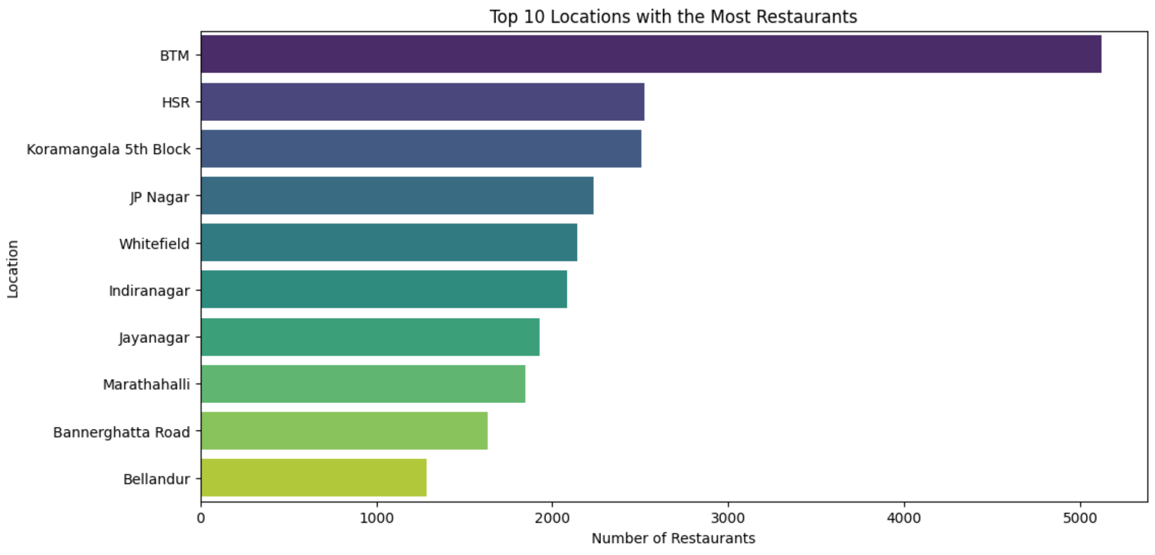


In [14]:
```
plt.figure(figsize=(12,6))
top_locations = df['location'].value_counts().head(10)
sns.barplot(x=top_locations.values, y=top_locations.index, palette=
plt.title("Top 10 Locations with the Most Restaurants")
plt.xlabel("Number of Restaurants")
plt.ylabel("Location")
plt.show()
```

```
/var/folders/zl/crx8hhfs3w31djl7_k5pp6s00000gn/T/ipykernel_4234/3411
144343.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=top_locations.values, y=top_locations.index, palette
="viridis")
```

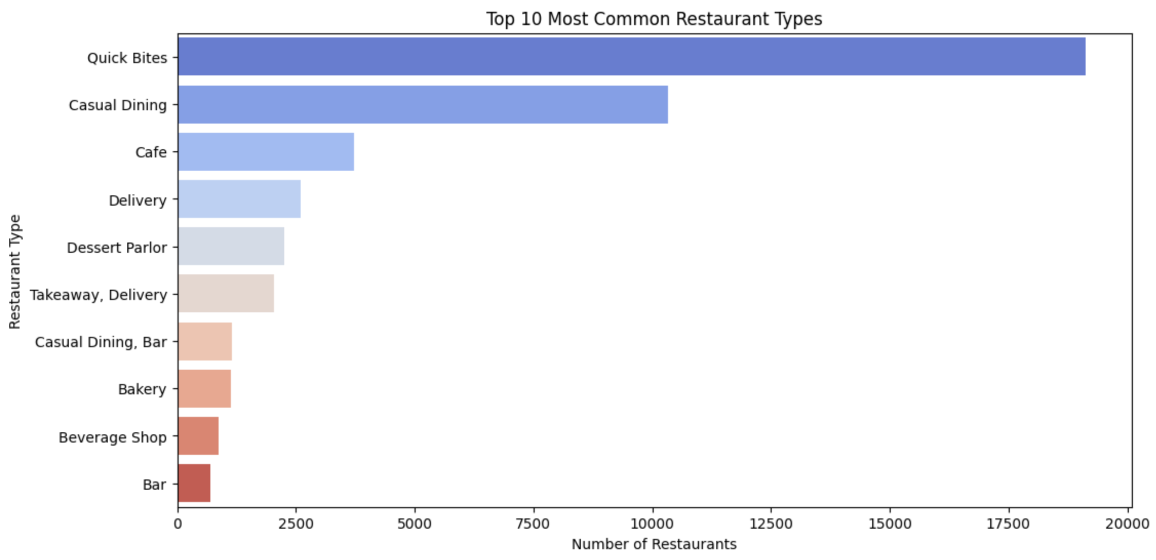


```
In [15]: plt.figure(figsize=(12,6))
         top_rest_types = df['rest_type'].value_counts().head(10)
         sns.barplot(x=top_rest_types.values, y=top_rest_types.index, palett
         plt.title("Top 10 Most Common Restaurant Types")
         plt.xlabel("Number of Restaurants")
         plt.ylabel("Restaurant Type")
         plt.show()
```

```
/var/folders/zl/crx8hhfs3w31djl7_k5pp6s00000gn/T/ipykernel_4234/3686
979636.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=top_rest_types.values, y=top_rest_types.index, palet
te="coolwarm")
```

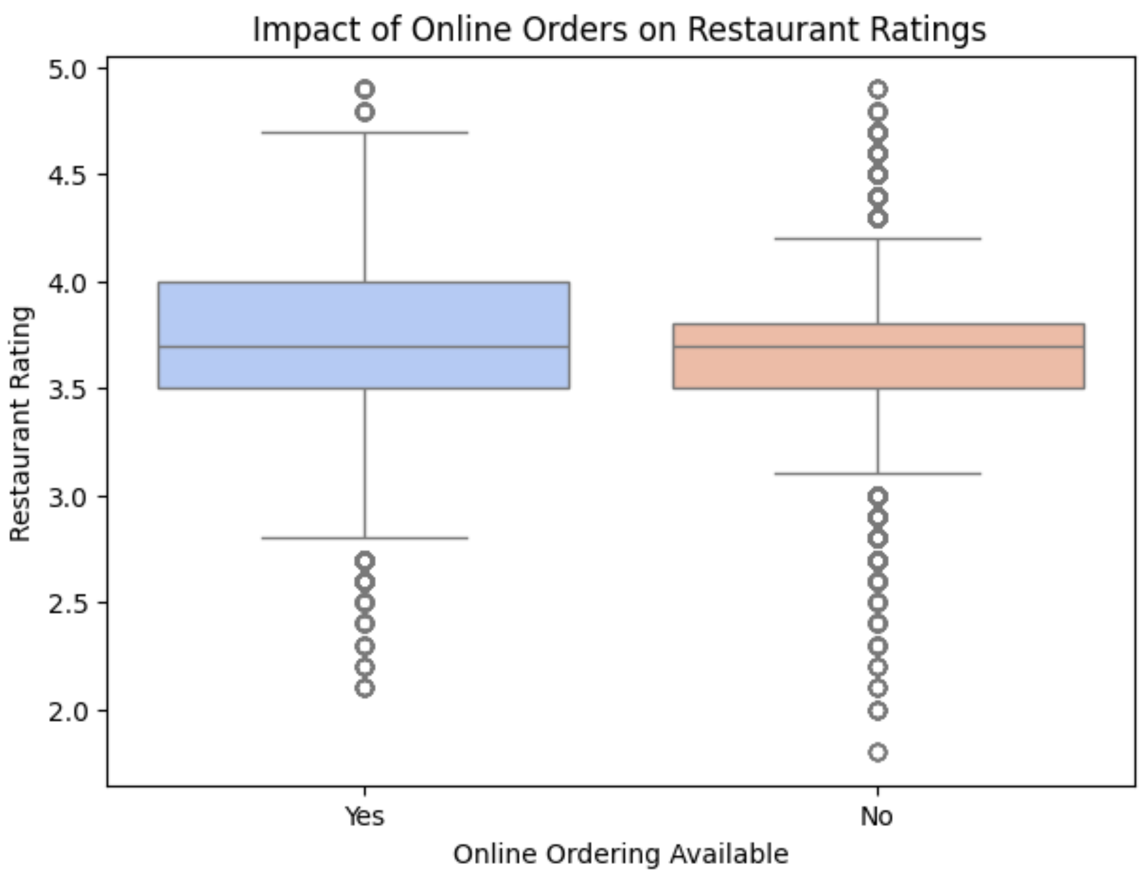


In [16]:
```
plt.figure(figsize=(7,5))
sns.boxplot(x=df['online_order'], y=df['rate'], palette="coolwarm")
plt.title("Impact of Online Orders on Restaurant Ratings")
plt.xlabel("Online Ordering Available")
plt.ylabel("Restaurant Rating")
plt.show()
```

```
/var/folders/zl/crx8hhfs3w31djl7_k5pp6s00000gn/T/ipykernel_4234/3064
332391.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.boxplot(x=df['online_order'], y=df['rate'], palette="coolwar
m")
```

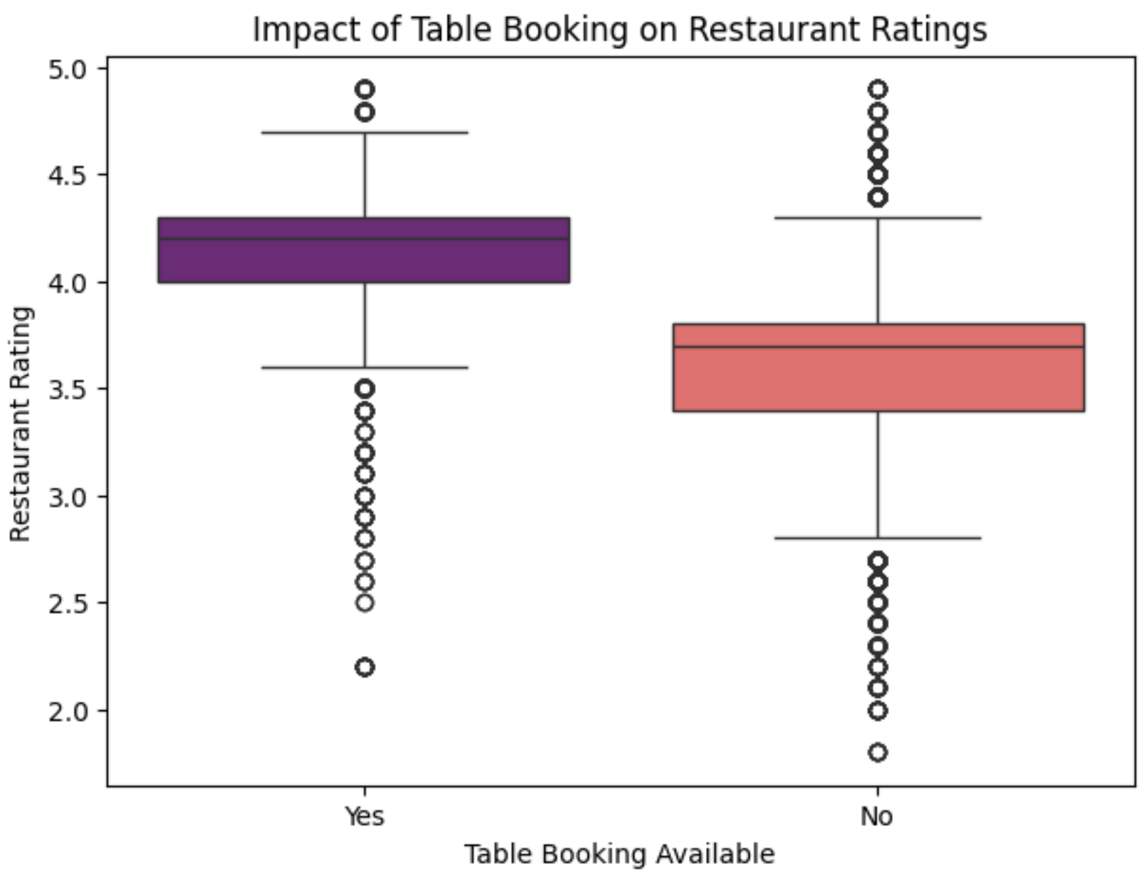


In [17]:
```
plt.figure(figsize=(7,5))
sns.boxplot(x=df['book_table'], y=df['rate'], palette="magma")
plt.title("Impact of Table Booking on Restaurant Ratings")
plt.xlabel("Table Booking Available")
plt.ylabel("Restaurant Rating")
plt.show()
```

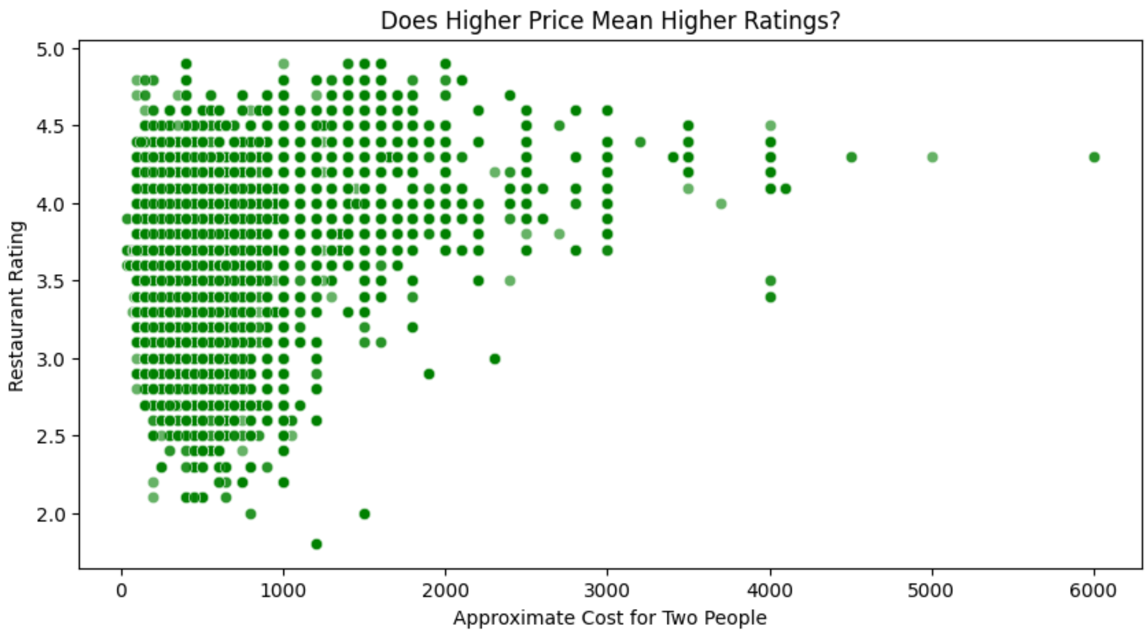
```
/var/folders/zl/crx8hhfs3w31djl7_k5pp6s00000gn/T/ipykernel_4234/1262
283792.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.boxplot(x=df['book_table'], y=df['rate'], palette="magma")
```



In [18]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(x=df['approx_cost(for two people)'], y=df['rate'],
plt.title("Does Higher Price Mean Higher Ratings?")
plt.xlabel("Approximate Cost for Two People")
plt.ylabel("Restaurant Rating")
plt.show()
```



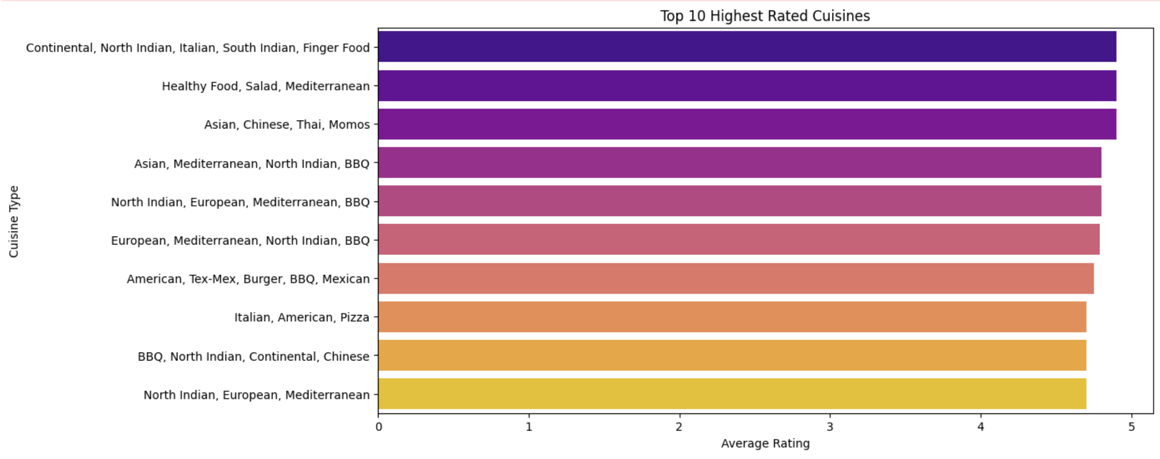
In [19]:
```
cuisine_ratings = df.groupby("cuisines")["rate"].mean().sort_values

plt.figure(figsize=(12,6))
sns.barplot(x=cuisine_ratings.values, y=cuisine_ratings.index, pale
plt.title("Top 10 Highest Rated Cuisines")
plt.xlabel("Average Rating")
plt.ylabel("Cuisine Type")
plt.show()
```

```
/var/folders/zl/crx8hhfs3w31djl7_k5pp6s00000gn/T/ipykernel_4234/1634
943807.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=cuisine_ratings.values, y=cuisine_ratings.index, pal
ette="plasma")
```



In [20]:
```
df
```

Out[20]:

	address	name	online_order	book_table	rate	votes	
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	775	Ba
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1	787	Ba
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8	918	Ba
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7	88	Ba
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8	166	Ba
...	...	...	...	...	...	...	...

51712	Four Points by Sheraton Bengaluru, 43/3, White...	Best Brews - Four Points by Sheraton Bengaluru...	No	No	3.6	27
51713	Number 10, Garudachar Palya, Mahadevapura, Whi...	Vinod Bar And Restaurant	No	No	3.7	0
51714	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No	3.7	0
51715	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes	4.3	236
51716	ITPL Main Road, KIADB Export Promotion Industr...	The Nest - The Den Bengaluru	No	No	3.4	13

51696 rows × 12 columns

In [21]:
```
df['online_order'] = df['online_order'].map({'Yes': 1, 'No': 0})
df['book_table'] = df['book_table'].map({'Yes': 1, 'No': 0})
```

In [22]:
```
df['success'] = df['rate'].apply(lambda x: 1 if x > 3.5 else 0)
```

In [23]:
```
df = df.drop(columns=['rate','name','address'])
```

In [24]:
```
df.head()
```

Out[24]:

	online_order	book_table	votes	location	rest_type	cuisines	app
0	1	1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	
1	1	0	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	
2	1	0	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	
3	0	0	88	Banashankari	Quick Bites	South Indian, North Indian	
4	0	0	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	

In [25]:
```
X = df.drop(columns=['approx_cost(for two people)', 'success'])
Y1 = df['success']
Y2 = df['approx_cost(for two people)']
```

In [26]:
```
X.shape,Y1.shape,Y2.shape
```

Out[26]:
```
((51696, 8), (51696,), (51696,))
```

In [27]:
```
X
```

Out[27]:

	online_order	book_table	votes	location	rest_type	cuisines
0	1	1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese
1	1	0	787	Banashankari	Casual Dining	Chinese, North Indian, Thai
2	1	0	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian
3	0	0	88	Banashankari	Quick Bites	South Indian, North Indian
4	0	0	166	Basavanagudi	Casual Dining	North Indian, Rajasthani
...	...	...	...	...	...	...
51712	0	0	27	Whitefield	Bar	Continental
51713	0	0	0	Whitefield	Bar	Finger Food
51714	0	0	0	Whitefield	Bar	Finger Food
51715	0	1	236	ITPL Main Road, Whitefield	Bar	Finger Food
51716	0	0	13	ITPL Main Road, Whitefield	Bar, Casual Dining	Finger Food, North Indian, Continental

51696 rows × 8 columns

In [28]:
```
categorical_columns = ['location', 'rest_type', 'cuisines', 'listed

le = LabelEncoder()
for col in categorical_columns:
    X[col] = le.fit_transform(X[col])
```

In [29]:
```
X
```

Out[29]:

	online_order	book_table	votes	location	rest_type	cuisines	listed
0	1	1	775	1	27	2159	
1	1	0	787	1	27	952	
2	1	0	918	1	22	766	
3	0	0	88	1	78	2555	
4	0	0	166	4	27	2188	
...	...	...	...	...	...	...	
51712	0	0	27	89	8	1009	
51713	0	0	0	89	8	1391	
51714	0	0	0	89	8	1391	
51715	0	1	236	26	8	1391	
51716	0	0	13	26	10	1418	

51696 rows × 8 columns

In [30]:
```
X_train_cls, X_test_cls, Y_train_cls, Y_test_cls = train_test_split
X_train_reg, X_test_reg, Y_train_reg, Y_test_reg = train_test_split
```

In [31]:
```
print(f"Training Set Shape: {X_train_cls.shape}, Testing Set Shape:
```

```
Training Set Shape: (41356, 8), Testing Set Shape: (10340, 8)
```

In [32]:
```
clf_logistic = LogisticRegression(max_iter=500)
clf_logistic.fit(X_train_cls, Y_train_cls)

Y_pred_cls = clf_logistic.predict(X_test_cls)

from sklearn.metrics import accuracy_score, classification_report

accuracy = accuracy_score(Y_test_cls, Y_pred_cls)
print(f"Logistic Regression Accuracy: {accuracy:.4f}")
print("\nClassification Report:\n", classification_report(Y_test_cl
```

```
Logistic Regression Accuracy: 0.7312

Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00      2779
           1       0.73      1.00      0.84      7561

    accuracy                           0.73     10340
   macro avg       0.37      0.50      0.42     10340
weighted avg       0.53      0.73      0.62     10340
```

```
/Users/apple/ml_env/lib/python3.10/site-packages/sklearn/metrics/_cl
assification.py:1565: UndefinedMetricWarning: Precision is ill-defin
ed and being set to 0.0 in labels with no predicted samples. Use `ze
ro_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(resu
lt))
/Users/apple/ml_env/lib/python3.10/site-packages/sklearn/metrics/_cl
assification.py:1565: UndefinedMetricWarning: Precision is ill-defin
ed and being set to 0.0 in labels with no predicted samples. Use `ze
ro_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(resu
lt))
/Users/apple/ml_env/lib/python3.10/site-packages/sklearn/metrics/_cl
assification.py:1565: UndefinedMetricWarning: Precision is ill-defin
ed and being set to 0.0 in labels with no predicted samples. Use `ze
ro_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(resu
lt))
```

In [33]:

```
clf_rf = RandomForestClassifier(n_estimators=100, random_state=42)
clf_rf.fit(X_train_cls, Y_train_cls)

Y_pred_rf = clf_rf.predict(X_test_cls)

accuracy_rf = accuracy_score(Y_test_cls, Y_pred_rf)
print(f"Random Forest Accuracy: {accuracy_rf:.4f}")
print("\nClassification Report:\n", classification_report(Y_test_cl
```

```
Random Forest Accuracy: 0.9409

Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.86      0.89      2779
           1       0.95      0.97      0.96      7561

    accuracy                           0.94     10340
   macro avg       0.93      0.92      0.92     10340
weighted avg       0.94      0.94      0.94     10340
```

In [34]:

```
clf_knn = KNeighborsClassifier(n_neighbors=5)
clf_knn.fit(X_train_cls, Y_train_cls)

Y_pred_knn = clf_knn.predict(X_test_cls)

accuracy_knn = accuracy_score(Y_test_cls, Y_pred_knn)
print(f"KNN Accuracy: {accuracy_knn:.4f}")
print("\nClassification Report:\n", classification_report(Y_test_cl
KNN Accuracy: 0.8923

Classification Report:
               precision    recall  f1-score   support

           0       0.81      0.79      0.80      2779
           1       0.92      0.93      0.93      7561

    accuracy                           0.89     10340
   macro avg       0.86      0.86      0.86     10340
weighted avg       0.89      0.89      0.89     10340
```

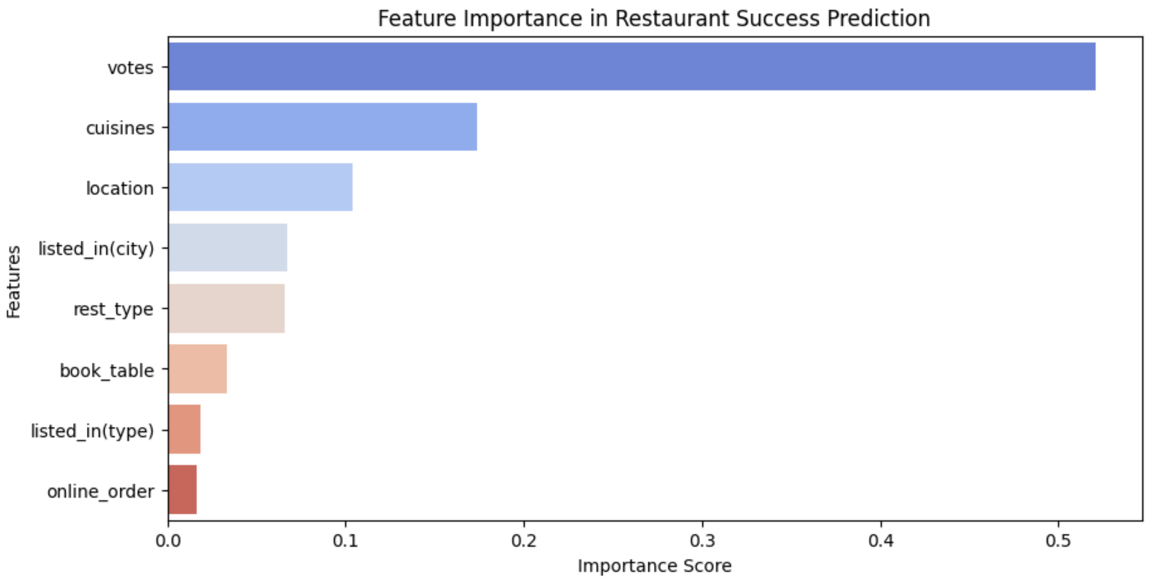
In [35]:
```
importances = pd.Series(clf_rf.feature_importances_, index=X.column

plt.figure(figsize=(10,5))
sns.barplot(x=importances.values, y=importances.index, palette="coo
plt.title("Feature Importance in Restaurant Success Prediction")
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.show()
```

```
/var/folders/zl/crx8hhfs3w31djl7_k5pp6s00000gn/T/ipykernel_4234/1825
49687.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=importances.values, y=importances.index, palette="co
olwarm")
```



In [36]:
```
selected_features = ["location", "rest_type", "cuisines", "book_tab
X_train_cls = X_train_cls[selected_features]
X_test_cls = X_test_cls[selected_features]
```

In [37]:
```
clf_rf = RandomForestClassifier(n_estimators=100, random_state=42)
clf_rf.fit(X_train_cls, Y_train_cls)
```

Out[37]:

```
▾      RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [38]:

```python
Y_pred_cls_rf = clf_rf.predict(X_test_cls)
accuracy_rf = accuracy_score(Y_test_cls, Y_pred_cls_rf)
print(f"Updated Random Forest Accuracy: {accuracy_rf:.4f}")
print("\nUpdated Classification Report:\n", classification_report(Y
```

```
Updated Random Forest Accuracy: 0.9776

Updated Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.96      0.96      2779
           1       0.98      0.98      0.98      7561

    accuracy                           0.98     10340
   macro avg       0.97      0.97      0.97     10340
weighted avg       0.98      0.98      0.98     10340
```

In [39]:

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score


X_train_reg = X_train_reg[selected_features]
X_test_reg = X_test_reg[selected_features]

rf_regressor = RandomForestRegressor(n_estimators=100, random_state
rf_regressor.fit(X_train_reg, Y_train_reg)

Y_pred_rf = rf_regressor.predict(X_test_reg)
mse_rf = mean_squared_error(Y_test_reg, Y_pred_rf)
r2_rf = r2_score(Y_test_reg, Y_pred_rf)
print(f"Random Forest Regressor MSE: {mse_rf:.4f}")
print(f"Random Forest R-Squared Score (R²): {r2_rf:.4f}")
```

```
Random Forest Regressor MSE: 4622.7259
Random Forest R-Squared Score (R²): 0.9764
```

In [40]:

```python
import joblib

joblib.dump(clf_rf, "rf_classifier.pkl")

joblib.dump(rf_regressor, "rf_regressor.pkl")
```

Out[40]:

```
['rf_regressor.pkl']
```

In [41]:

```python
df
```

Out[41]:

	online_order	book_table	votes	location	rest_type	cuisines
0	1	1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese
1	1	0	787	Banashankari	Casual Dining	Chinese, North Indian, Thai
2	1	0	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian
3	0	0	88	Banashankari	Quick Bites	South Indian, North Indian
4	0	0	166	Basavanagudi	Casual Dining	North Indian, Rajasthani
...	...	...	...	...	...	...
51712	0	0	27	Whitefield	Bar	Continental
51713	0	0	0	Whitefield	Bar	Finger Food
51714	0	0	0	Whitefield	Bar	Finger Food
51715	0	1	236	ITPL Main Road, Whitefield	Bar	Finger Food
51716	0	0	13	ITPL Main Road, Whitefield	Bar, Casual Dining	Finger Food, North Indian, Continental

51696 rows × 10 columns

In [42]:
```
df[["location", "rest_type", "cuisines", "book_table", "votes"]]
```

Out[42]:

	location	rest_type	cuisines	book_table	votes
0	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	1	775
1	Banashankari	Casual Dining	Chinese, North Indian, Thai	0	787
2	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	0	918
3	Banashankari	Quick Bites	South Indian, North Indian	0	88
4	Basavanagudi	Casual Dining	North Indian, Rajasthani	0	166
...	...	...	...	...	...
51712	Whitefield	Bar	Continental	0	27
51713	Whitefield	Bar	Finger Food	0	0
51714	Whitefield	Bar	Finger Food	0	0
51715	ITPL Main Road, Whitefield	Bar	Finger Food	1	236
51716	ITPL Main Road, Whitefield	Bar, Casual Dining	Finger Food, North Indian, Continental	0	13

51696 rows × 5 columns

In [43]:

```
df["location"].unique()
```

Out[43]:
```
array(['Banashankari', 'Basavanagudi', 'Mysore Road', 'Jayanagar',
       'Kumaraswamy Layout', 'Rajarajeshwari Nagar', 'Vijay Naga
r',
       'Uttarahalli', 'JP Nagar', 'South Bangalore', 'City Marke
t',
       'Nagarbhavi', 'Bannerghatta Road', 'BTM', 'Kanakapura Roa
d',
       'Bommanahalli', 'CV Raman Nagar', 'Electronic City', 'HSR',
       'Marathahalli', 'Sarjapur Road', 'Wilson Garden', 'Shanti N
agar',
       'Koramangala 5th Block', 'Koramangala 8th Block', 'Richmond
Road',
       'Koramangala 7th Block', 'Jalahalli', 'Koramangala 4th Bloc
k',
       'Bellandur', 'Whitefield', 'East Bangalore', 'Old Airport R
oad',
       'Indiranagar', 'Koramangala 1st Block', 'Frazer Town', 'RT
Nagar',
       'MG Road', 'Brigade Road', 'Lavelle Road', 'Church Street',
       'Ulsoor', 'Residency Road', 'Shivajinagar', 'Infantry Roa
d',
       'St. Marks Road', 'Cunningham Road', 'Race Course Road',
       'Commercial Street', 'Vasanth Nagar', 'HBR Layout', 'Domlu
r',
       'Ejipura', 'Jeevan Bhima Nagar', 'Old Madras Road', 'Malles
hwaram',
       'Seshadripuram', 'Kammanahalli', 'Koramangala 6th Block',
       'Majestic', 'Langford Town', 'Central Bangalore', 'Sanjay N
agar',
       'Brookefield', 'ITPL Main Road, Whitefield',
       'Varthur Main Road, Whitefield', 'KR Puram',
       'Koramangala 2nd Block', 'Koramangala 3rd Block', 'Koramang
ala',
       'Hosur Road', 'Rajajinagar', 'Banaswadi', 'North Bangalor
e',
       'Nagawara', 'Hennur', 'Kalyan Nagar', 'New BEL Road', 'Jakk
ur',
       'Rammurthy Nagar', 'Thippasandra', 'Kaggadasapura', 'Hebba
l',
       'Kengeri', 'Sankey Road', 'Sadashiv Nagar', 'Basaveshwara N
agar',
       'Yeshwantpur', 'West Bangalore', 'Magadi Road', 'Yelahank
a',
       'Sahakara Nagar', 'Peenya'], dtype=object)
```

In [44]:
```
df["rest_type"].unique()
```

Out[44]:
```
array(['Casual Dining', 'Cafe, Casual Dining', 'Quick Bites',
       'Casual Dining, Cafe', 'Cafe', 'Quick Bites, Cafe',
       'Cafe, Quick Bites', 'Delivery', 'Mess', 'Dessert Parlor',
       'Bakery, Dessert Parlor', 'Pub', 'Bakery', 'Takeaway, Deliv
ery',
       'Fine Dining', 'Beverage Shop', 'Sweet Shop', 'Bar',
       'Beverage Shop, Quick Bites', 'Confectionery',
       'Quick Bites, Beverage Shop', 'Dessert Parlor, Sweet Shop',
       'Bakery, Quick Bites', 'Sweet Shop, Quick Bites', 'Kiosk',
       'Food Truck', 'Quick Bites, Dessert Parlor',
       'Beverage Shop, Dessert Parlor', 'Takeaway', 'Pub, Casual D
ining',
       'Casual Dining, Bar', 'Dessert Parlor, Beverage Shop',
       'Quick Bites, Bakery', 'Dessert Parlor, Quick Bites',
       'Microbrewery, Casual Dining', 'Lounge', 'Bar, Casual Dinin
g',
       'Food Court', 'Cafe, Bakery', 'Unknown', 'Dhaba',
       'Quick Bites, Sweet Shop', 'Microbrewery',
       'Food Court, Quick Bites', 'Pub, Bar', 'Casual Dining, Pu
b',
       'Lounge, Bar', 'Food Court, Dessert Parlor',
       'Casual Dining, Sweet Shop', 'Food Court, Casual Dining',
       'Casual Dining, Microbrewery', 'Sweet Shop, Dessert Parlo
r',
       'Bakery, Beverage Shop', 'Lounge, Casual Dining',
       'Cafe, Food Court', 'Beverage Shop, Cafe', 'Cafe, Dessert P
arlor',
       'Dessert Parlor, Cafe', 'Dessert Parlor, Bakery',
       'Microbrewery, Pub', 'Bakery, Food Court', 'Club',
       'Quick Bites, Food Court', 'Bakery, Cafe', 'Bar, Cafe',
       'Pub, Cafe', 'Casual Dining, Irani Cafee', 'Fine Dining, Lo
unge',
       'Bar, Quick Bites', 'Bakery, Kiosk', 'Pub, Microbrewery',
       'Microbrewery, Lounge', 'Fine Dining, Microbrewery',
       'Fine Dining, Bar', 'Mess, Quick Bites', 'Dessert Parlor, K
iosk',
       'Bhojanalya', 'Casual Dining, Quick Bites', 'Pop Up', 'Caf
e, Bar',
       'Casual Dining, Lounge', 'Bakery, Sweet Shop', 'Microbrewer
y, Bar',
       'Cafe, Lounge', 'Bar, Pub', 'Lounge, Cafe', 'Club, Casual D
ining',
       'Quick Bites, Mess', 'Quick Bites, Meat Shop',
       'Quick Bites, Kiosk', 'Lounge, Microbrewery',
       'Food Court, Beverage Shop', 'Dessert Parlor, Food Court',
       'Bar, Lounge'], dtype=object)
```

In [45]:
```
df["cuisines"].unique()
```

Out[45]:
```
array(['North Indian, Mughlai, Chinese', 'Chinese, North Indian, T
hai',
       'Cafe, Mexican, Italian', ...,
       'North Indian, Street Food, Biryani', 'Chinese, Mughlai',
       'North Indian, Chinese, Arabian, Momos'], dtype=object)
```

In [46]:
```
%%writefile app.py
```

```
import streamlit as st
import numpy as np
import pandas as pd
import joblib
import plotly.express as px
import altair as alt
from streamlit_extras.add_vertical_space import add_vertical_space

rf_classifier = joblib.load("rf_classifier.pkl")
rf_regressor = joblib.load("rf_regressor.pkl")

st.set_page_config(page_title="AI-Powered Restaurant Success Predic

st.markdown("""
    <style>
    body { background-color: #0e1117; color: white; }
    .stApp { background-color: #0e1117; }
    .title { font-size: 50px; font-weight: bold; color: #FF4B4B; te
    .subheader { font-size: 30px; font-weight: bold; color: #E1E1E1
    .section-header { font-size: 30px; font-weight: bold; color: #F
    .glass-card {
        background: rgba(255, 255, 255, 0.1);
        border-radius: 15px;
        padding: 20px;
        backdrop-filter: blur(10px);
        box-shadow: 0px 4px 12px rgba(255, 255, 255, 0.1);
    }
    </style>
""", unsafe_allow_html=True)

st.markdown("<h1 class='title'>🍽️ AI-Powered Restaurant Success Pre
st.markdown("<h2 class='subheader'>📊 Predict Success & Estimate Pr
add_vertical_space(2)

locations = ['Banashankari', 'Basavanagudi', 'Mysore Road', 'Jayana
             'Rajarajeshwari Nagar', 'Vijay Nagar', 'Uttarahalli',
             'City Market', 'Nagarbhavi', 'Bannerghatta Road', 'BTM
             'Bommanahalli', 'CV Raman Nagar', 'Electronic City', '
             'Sarjapur Road', 'Wilson Garden', 'Shanti Nagar', 'Kor
             'Koramangala 8th Block', 'Richmond Road', 'Koramangala
             'Koramangala 4th Block', 'Bellandur', 'Whitefield', 'E
             'Old Airport Road', 'Indiranagar', 'Koramangala 1st Bl
             'RT Nagar', 'MG Road', 'Brigade Road', 'Lavelle Road',
             'Ulsoor', 'Residency Road', 'Shivajinagar', 'Infantry
             'Cunningham Road', 'Race Course Road', 'Commercial Str
             'HBR Layout', 'Domlur', 'Ejipura', 'Jeevan Bhima Nagar
             'Malleshwaram', 'Seshadripuram', 'Kammanahalli', 'Kora
             'Majestic', 'Langford Town', 'Central Bangalore', 'San
             'ITPL Main Road, Whitefield', 'Varthur Main Road, Whit
             'Koramangala 2nd Block', 'Koramangala 3rd Block', 'Kor
             'Rajajinagar', 'Banaswadi', 'North Bangalore', 'Nagawa
             'New BEL Road', 'Jakkur', 'Rammurthy Nagar', 'Thippasa
             'Hebbal', 'Kengeri', 'Sankey Road', 'Sadashiv Nagar',
             'Yeshwantpur', 'West Bangalore', 'Magadi Road', 'Yelah
```

```
rest_types = ['Casual Dining', 'Cafe, Casual Dining', 'Quick Bites'
              'Quick Bites, Cafe', 'Cafe, Quick Bites', 'Delivery',
              'Bakery, Dessert Parlor', 'Pub', 'Bakery', 'Takeaway,
              'Beverage Shop', 'Sweet Shop', 'Bar', 'Beverage Shop,

cuisine_options = ['North Indian, Mughlai, Chinese', 'Chinese, Nort
                   'North Indian, Street Food, Biryani', 'Chinese,

st.sidebar.title("🔍 Enter Restaurant Details")

location = st.sidebar.selectbox("📍 Select Location", sorted(locati
rest_type = st.sidebar.multiselect("🍽️ Select Restaurant Type(s)",
cuisines = st.sidebar.multiselect("🍜 Select Cuisine(s)", sorted(cu
book_table = st.sidebar.radio("📅 Table Booking Available?", ["Yes"
votes = st.sidebar.slider("👍 Number of Votes", 0, 5000, 1000)

def hash_encode(val):
    return hash(val) % 1000

location_encoded = hash_encode(location)
rest_type_encoded = hash_encode(",".join(sorted(rest_type)))
cuisines_encoded = hash_encode(",".join(sorted(cuisines)))
book_table_encoded = 1 if book_table == "Yes" else 0

input_data = np.array([
    location_encoded,
    rest_type_encoded,
    cuisines_encoded,
    book_table_encoded,
    votes
]).reshape(1, -1)

col1, col2 = st.columns(2)

with col1:
    st.markdown("<div class='glass-card'>", unsafe_allow_html=True)
    st.markdown("<p class='section-header'>🎯 Will Your Restaurant

    if st.button("🔮 Predict Success", use_container_width=True):
        success_proba = rf_classifier.predict_proba(input_data)[0]
        success_percentage = success_proba[1] * 100
        failure_percentage = success_proba[0] * 100

        st.subheader(f"📊 Success Probability: {success_percentage:

        fig = px.pie(
            values=[success_percentage, failure_percentage],
            names=["Success", "Failure"],
            color=["Success", "Failure"],
            color_discrete_map={"Success": "green", "Failure": "red
        )
        st.plotly_chart(fig)
    st.markdown("</div>", unsafe_allow_html=True)

with col2:
    st.markdown("<div class='glass-card'>", unsafe_allow_html=True)
    st.markdown("<p class='section-header'>💵 Estimate Restaurant P

    if st.button("💲 Predict Approximate Cost", use_container_width
        price_prediction = rf_regressor.predict(input_data)
        st.subheader(f"Estimated Cost for Two: ₹{round(price_predic

        price_range = [price_prediction[0] - 200, price_prediction[
        price_labels = ["Lower Estimate", "Predicted Price", "Highe
        price_fig = px.bar(x=price_labels, y=price_range, title="📊
        st.plotly_chart(price_fig)
    st.markdown("</div>", unsafe_allow_html=True)

st.markdown("<h2 class='section-header'>📈 Success Trends</h2>", un
trend_data = pd.DataFrame({
    'Location': ["BTM", "Indiranagar", "Koramangala", "Whitefield"]
    'Success Rate (%)': [90, 80, 85, 70],
    'Avg Cost (₹)': [1200, 1500, 1300, 1100]
})

trend_chart = alt.Chart(trend_data).mark_bar().encode(
    x="Location",
    y="Success Rate (%)",
    color="Location"
).properties(title="📊 Success Rate Across Locations")

st.altair_chart(trend_chart, use_container_width=True)
```

```
Overwriting app.py
```

In [ ]: