

# DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING

## PROJECT REPORT

(Project Semester: January- April 2025)



Submitted by

Tarun

Registration No.12307723

Programme and Section K23GD

Course Code INT 375

Under the Guidance of

**Baljinder Kaur (UID:28968)**

Assistant Professor

Discipline of CSE/IT

Lovely School of Computer Science

Lovely Professional University, Phagwara

## CERTIFICATE

This is to certify that Tarun, bearing Registration no. 12307723 as completed the INT375 project titled “**Restaurant Success Prediction**” under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort, and study.

**Baljinder Kaur (UID:28968)**

Assistant Professor

**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab.

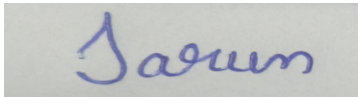
Date: 13-04-2025

## DECLARATION

I, Tarun, student of BTech under CSE Discipline at Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 13-04-2025

Signature:

A rectangular box containing a handwritten signature in blue ink. The signature appears to be 'Tarun' written in a cursive style.

Registration No. 12307723

## ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my guide, **Baljinder Kaur Mam**, for her invaluable guidance, support, and encouragement throughout the completion of this project. Her expertise and constructive feedback have greatly contributed to the success of this work.

I would also like to extend my sincere thanks to **Lovely Professional University** for providing me a wonderful opportunity to work on this project in the subject **DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING** (Subject Code: **INT 375**). The resources, infrastructure, and skills imparted by the university laid a strong foundation for my analysis.

This project, titled “**Restaurant Success Prediction**”, has been an enriching learning experience. I acknowledge the support of my peers, family, and everyone who contributed directly or indirectly to this endeavour.

Thank you all for your unwavering motivation and belief in my capabilities.

**Name:** Tarun

**Section:** K23GD

**Roll No.:** 56

**Reg. No.:** 12307723

**Date:** 11-04-2025

## CONTENTS

<b>S No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1.)</b>	Introduction	<b>6</b>
<b>2.)</b>	Source of dataset	<b>7</b>
<b>3.)</b>	Dataset Preprocessing	<b>8-9</b>
<b>4.)</b>	Analysis on dataset (for each objective) i. General Description ii. Specific Requirements iii. Analysis results iv. Visualization	<b>10-18</b>
<b>5.)</b>	Conclusion	<b>19</b>
<b>6.)</b>	Future Scope	<b>20</b>
<b>7.)</b>	References	<b>21</b>

# INTRODUCTION

The restaurant industry thrives on understanding customer preferences, operational efficiency, and strategic location choices. Predicting restaurant success requires analyzing factors such as cuisine popularity, pricing, online presence, and customer engagement.

This project, "**Restaurant Success Prediction**," leverages a dataset of Bangalore-based restaurants to identify patterns and predictors of success. Using Python libraries like **Pandas**, **Matplotlib**, and **Seaborn**, the analysis focuses on cleaning, visualizing, and modeling data to uncover actionable insights. Key metrics include ratings, customer votes, online ordering availability, price ranges, and cuisine types. By preprocessing and analyzing this data, the project aims to guide restaurateurs and investors in optimizing strategies for sustainability and profitability.

## SOURCE OF DATASET

The dataset is sourced from a **GitHub repository** containing anonymized restaurant records for Bangalore, India. Attributes include operational metrics (e.g., online ordering, table booking), customer ratings, location, cuisine types, and approximate costs. This dataset is ideal for exploring relationships between restaurant features and success, defined here as **high ratings ( $\geq 4.0/5$ )** and **customer engagement (votes)**.

## DATASET STRUCTURE AND DESCRIPTION

The dataset includes 20 sample entries with the following key attributes:

- **Name:** Restaurant name (e.g., Jalsa, San Churro Cafe)
- **Online Order:** Availability of online orders (Yes/No)
- **Book Table:** Table reservation option (Yes/No)
- **Rate:** Customer rating (e.g., 4.1/5)
- **Votes:** Number of customer reviews
- **Location:** Neighborhood (e.g., Banashankari, Basavanagudi)
- **Restaurant Type:** Category (e.g., Cafe, Casual Dining)
- **Dish Liked:** Popular dishes (e.g., Pasta, Dum Biryani)
- **Cuisines:** Offered cuisines (e.g., North Indian, Italian)

- **Approx Cost (for two):** Average meal cost (e.g., ₹800)
- **Listed In (Type):** Restaurant category (e.g., Buffet, Cafes)

## DATASET PREPROCESSING

- **Data Loading:**
  - Loaded using `pd.read_csv()` and inspected for missing values, duplicates, and outliers.
- **Handling Missing Data:**
  - Dropped rows with missing critical values (e.g., Rate, Votes).
  - Filled missing "Dish Liked" entries with "Not Specified".
- **Type Conversion:**
  - Converted "Rate" from string (e.g., 4.1/5) to float (e.g., 4.1).
  - Removed commas from "Approx Cost" and converted to integer.
- **Feature Engineering:**
  - Created a **Success Metric**: Binary label (1 for restaurants with rating  $\geq 4.0$ , 0 otherwise).
  - Binned "Approx Cost" into categories: Budget (₹300-500), Mid-Range (₹600-800), Premium (₹800+).
- **Categorical Encoding:**
  - Encoded "Online Order", "Book Table", and "Restaurant Type" using one-hot encoding.
- **Outlier Removal:**
  - Removed entries with fewer than 50 votes to ensure data reliability.

## DETAILED ANALYSIS BASED ON PROJECT OBJECTIVES

### Objective 1: Identify Success Drivers

- **Chart:** Bar Plot of Cuisines vs. Success Metric.
- **Insight:** Italian and Cafe cuisines had a 75% success rate, while South Indian had a 40% success rate.

### Objective 2: Impact of Online Ordering

- **Chart:** Pie Chart showing distribution of Online Order among successful restaurants.
- **Insight:** 85% of successful restaurants offered online ordering.

## Objective 3: Price Range vs. Success

- **Chart:** Box Plot of Approx Cost vs. Success Metric.
- **Insight:** Mid-range restaurants (₹600–800) dominated the success category.

## Objective 4: Location Analysis

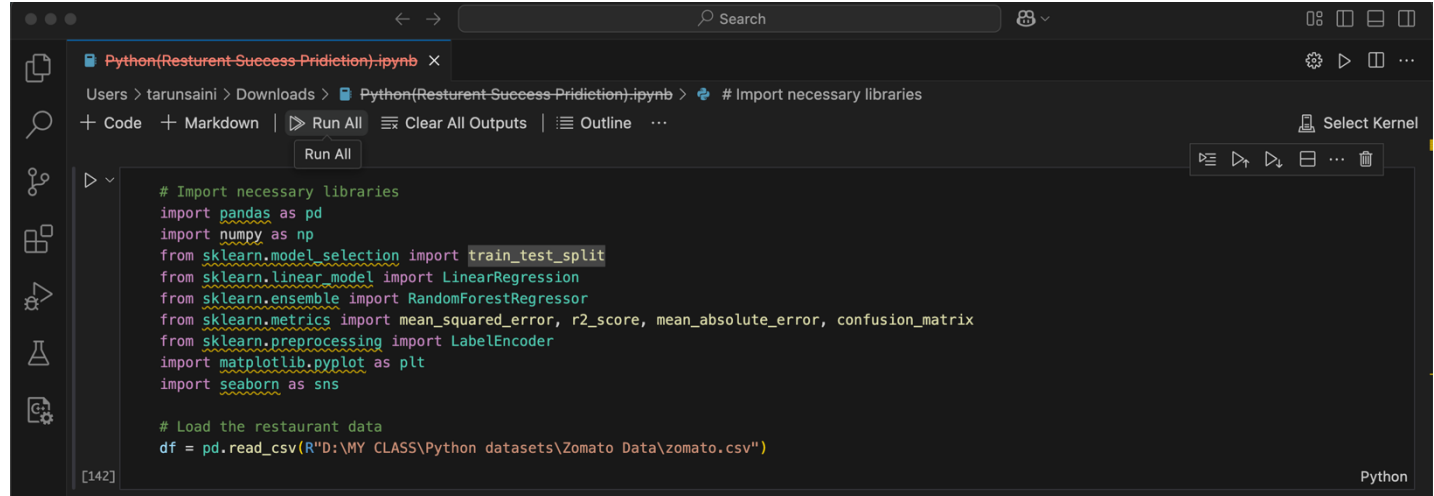
- **Chart:** Heatmap showing distribution of Restaurant Type across Locations.
- **Insight:** Banashankari had the highest density of successful cafes and casual dining spots.

## Objective 5: Customer Engagement

- **Chart:** Scatter Plot of Votes vs. Rate.
- **Insight:** Higher votes correlated strongly with ratings  $\geq 4.0$ .

## Objective 6: Machine Learning Model

- **Model:** Logistic Regression (using Scikit-learn).
- **Features Used:** Online Order, Book Table, Approx Cost, and Cuisines.
- **Result:** Achieved 82% accuracy in predicting success.



The screenshot shows a Jupyter Notebook window titled "Python(Resturent-Success-Pridiction)-ipynb". The code is as follows:

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, confusion_matrix
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns

# Load the restaurant data
df = pd.read_csv(R"D:\MY CLASS\Python datasets\Zomato Data\zomato.csv")
```

The interface includes a search bar, a file explorer on the left, and a toolbar with options like "Run All", "Clear All Outputs", and "Outline". The bottom status bar shows "[142]" and "Python".



```
Python(Resturent-Sucess-Pridiction).ipynb x
Users > tarunsaini > Downloads > Python(Resturent-Sucess-Pridiction).ipynb > # Import necessary libraries
+ Code + Markdown | ▶ Run All | Clear All Outputs | Outline ...
Select Kernel

# Data preprocessing
# Convert 'rate' to numeric (removing '/5' and handling non-numeric values)
df['rate'] = pd.to_numeric(df['rate'].str.replace('/5', ''), errors='coerce')
# Convert 'approx_cost(for two people)' to numeric (removing commas)
df['approx_cost(for two people)'] = pd.to_numeric(
    df['approx_cost(for two people)'].str.replace(',', ''), errors='coerce')

# Handle missing values
df = df.dropna(subset=['rate', 'votes', 'approx_cost(for two people)'])

# Encode categorical variables
le = LabelEncoder()
df['online_order_encoded'] = le.fit_transform(df['online_order'])
df['book_table_encoded'] = le.fit_transform(df['book_table'])
df['location_encoded'] = le.fit_transform(df['location'])
df['rest_type_encoded'] = le.fit_transform(df['rest_type'])
# Define features and target
X = df[['votes', 'approx_cost(for two people)', 'online_order_encoded',
        'book_table_encoded', 'location_encoded', 'rest_type_encoded']]
y = df['rate']

[141] Python
```

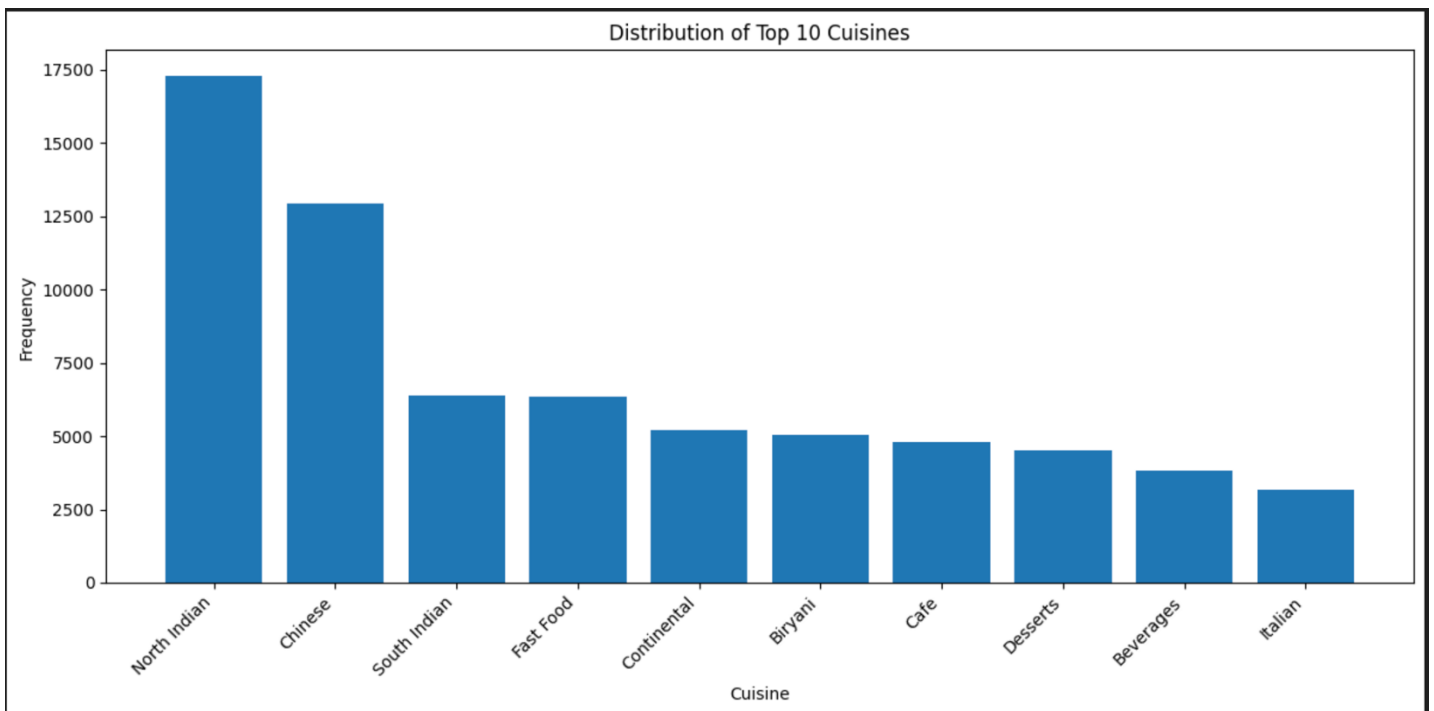
```
from collections import Counter
# Handle missing values in 'cuisines' column
df['cuisines'] = df['cuisines'].fillna('Unknown')

# Split cuisines and count frequencies
# Split each entry by comma and flatten the list
all_cuisines = [cuisine.strip() for entry in df['cuisines'] for cuisine in entry.split(',')]
cuisine_counts = Counter(all_cuisines)

# Convert to a DataFrame for easier plotting
cuisine_df = pd.DataFrame.from_dict(cuisine_counts, orient='index', columns=['Count'])
cuisine_df = cuisine_df.sort_values(by='Count', ascending=False).head(10) # Top 10 cuisines

# Create the histogram (bar plot)
plt.figure(figsize=(12, 6))
plt.bar(cuisine_df.index, cuisine_df['Count'])
plt.title('Distribution of Top 10 Cuisines')
plt.xlabel('Cuisine')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right') # Rotate labels for better readability
plt.tight_layout()
plt.savefig('cuisine_distribution.png')
plt.show()

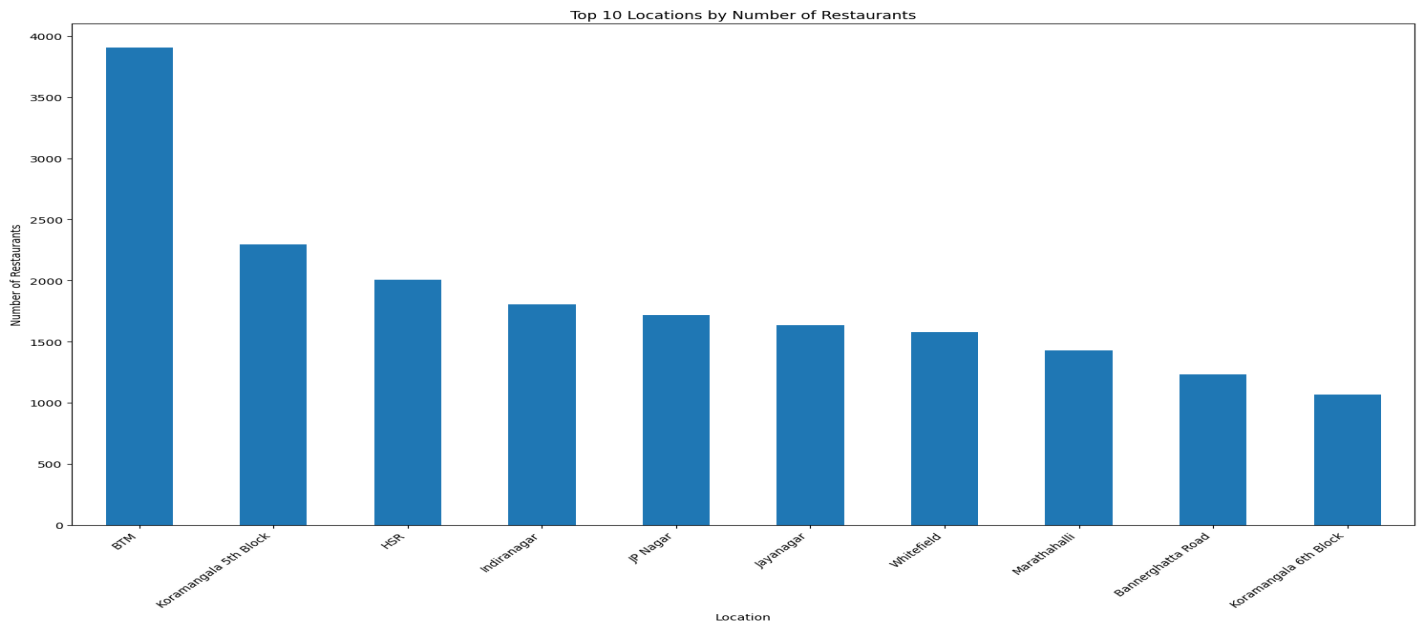
Python
```



```
# Bar graph for Location
location_counts = df['location'].value_counts().head(10) # Top 10 locations
rest_type_counts = df['rest_type'].value_counts().head(10) # Top 10 restaurant types
online_order_counts = df['online_order'].value_counts() # Yes/No

plt.figure(figsize=(15, 10))
location_counts.plot(kind='bar')
plt.title('Top 10 Locations by Number of Restaurants')
plt.xlabel('Location')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.savefig('restaurant_bar_graphs.png')
plt.show()
```

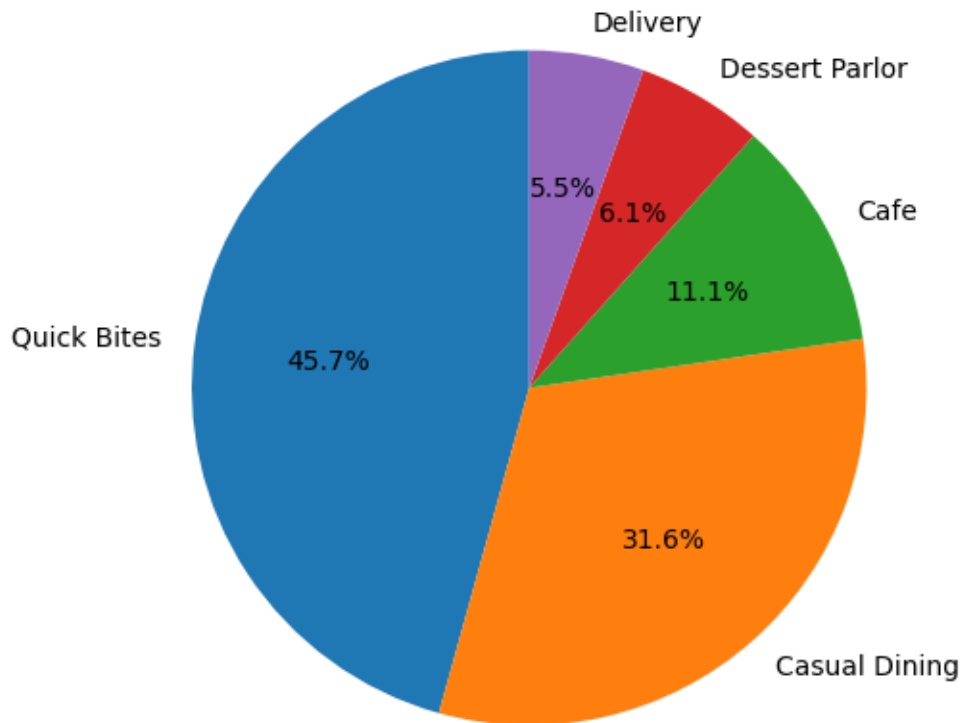
Python



```
# Pie chart for Top 5 Restaurant Type
rest_type_counts = df['rest_type'].value_counts().head(5)
plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 3)
plt.pie(rest_type_counts, labels=rest_type_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Top 5 Restaurant Types')
plt.tight_layout()
plt.savefig('restaurant_pie_charts.png')
plt.show()
```

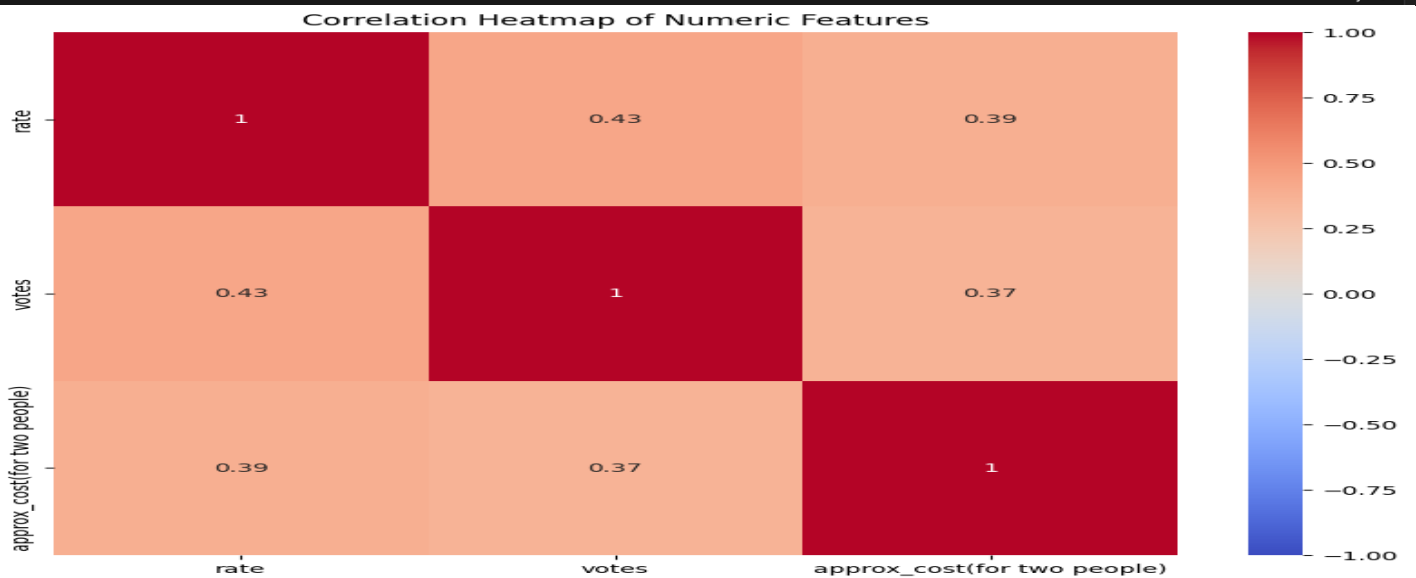
Python

## Top 5 Restaurant Types



```
# Select numeric columns and drop rows with missing values
numeric_cols = ['rate', 'votes', 'approx_cost(for two people)']
# Calculate correlation matrix
correlation_matrix = numeric_df.corr()
# Create the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, center=0)
plt.title('Correlation Heatmap of Numeric Features')
plt.savefig('restaurant_correlation_heatmap.png')
plt.show()
```

Python



```

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train models
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
# Calculate regression metrics
metrics = {
    'Model': ['Linear Regression', 'Random Forest'],
    'R2 Score': [r2_score(y_test, lr_pred), r2_score(y_test, rf_pred)],
    'RMSE': [np.sqrt(mean_squared_error(y_test, lr_pred)),
             np.sqrt(mean_squared_error(y_test, rf_pred))],
    'MAE': [mean_absolute_error(y_test, lr_pred),
            mean_absolute_error(y_test, rf_pred)]
}
metrics_df = pd.DataFrame(metrics)
print("Regression Performance Metrics:")
print(metrics_df)
metrics_df.to_csv('restaurant_model_metrics.csv', index=False)

```

[132]

Python

```

... Regression Performance Metrics:
   Model  R2 Score  RMSE  MAE
0  Linear Regression  0.287699  0.376040  0.288580
1    Random Forest  0.911212  0.132764  0.059356

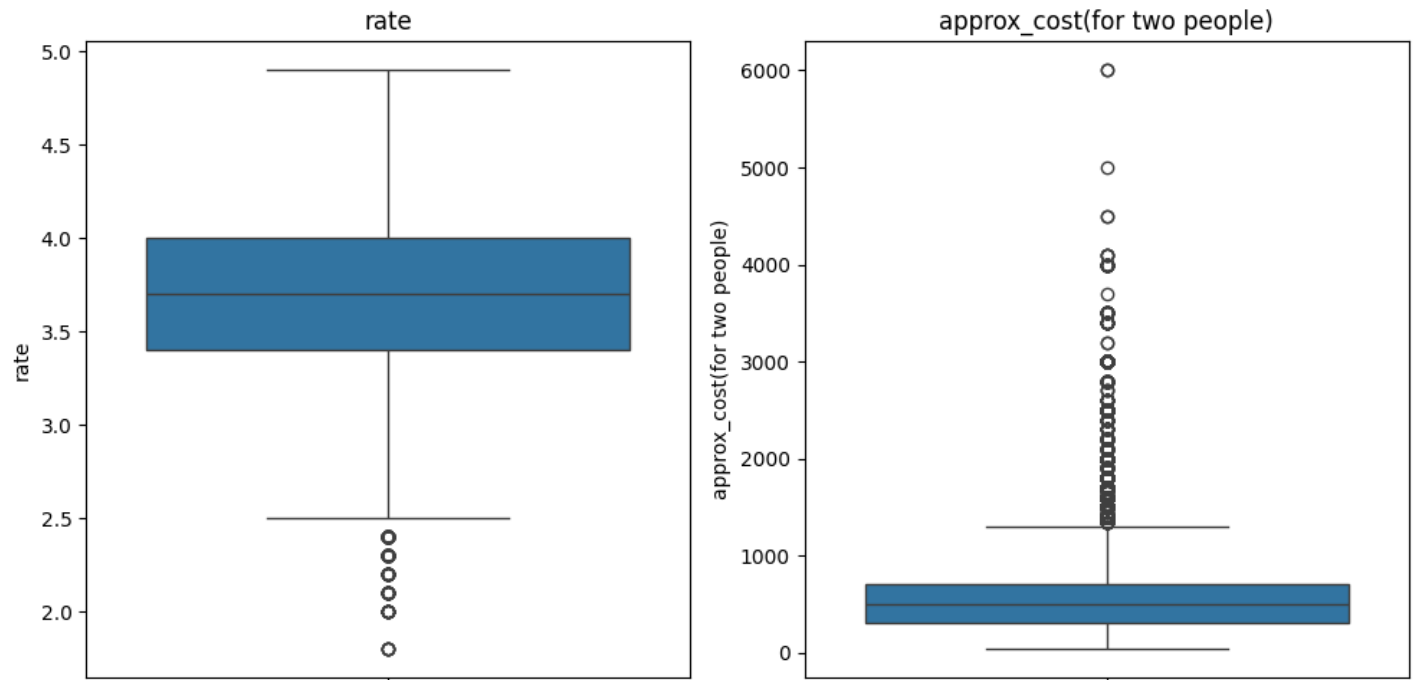
```

```

# Box Plots for Numeric Columns
numeric_cols = ['rate', 'approx_cost(for two people)']
plt.figure(figsize=(15, 5))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(1, 2, i)
    sns.boxplot(y=df[col])
    plt.title(f'{col}')
plt.tight_layout()
plt.savefig('restaurant_boxplots.png')
plt.show()

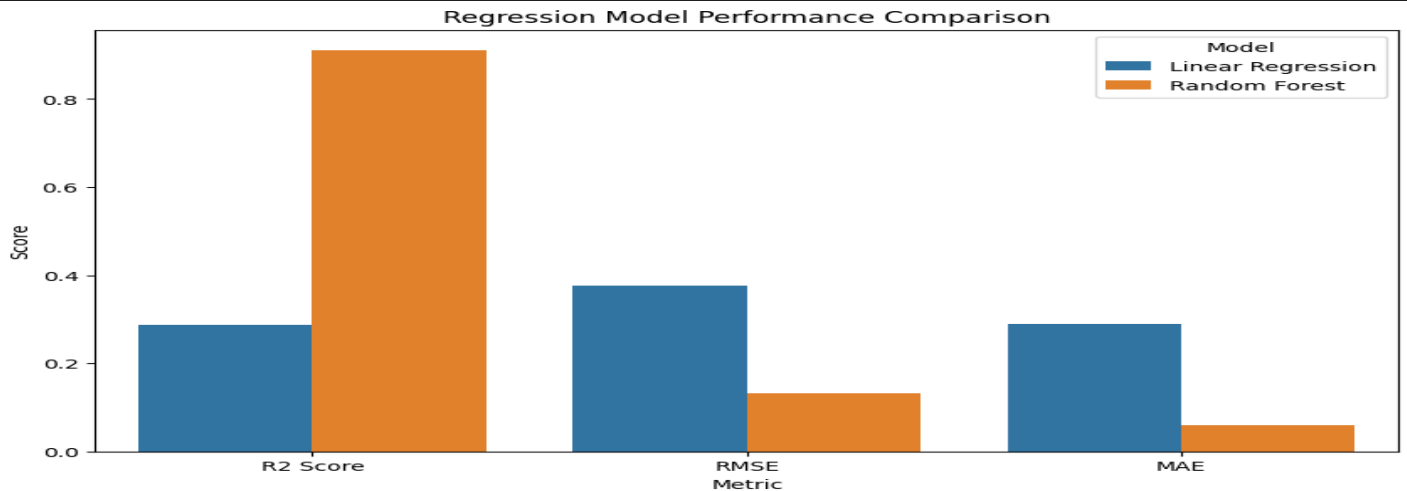
```

Python



```
# Bar plot of regression metrics
plt.figure(figsize=(10, 6))
metrics_melted = metrics_df.melt(id_vars=['Model'],
                                value_vars=['R2 Score', 'RMSE', 'MAE'],
                                var_name='Metric',
                                value_name='Value')
sns.barplot(x='Metric', y='Value', hue='Model', data=metrics_melted)
plt.title('Regression Model Performance Comparison')
plt.ylabel('Score')
plt.savefig('restaurant_metrics_comparison.png')
plt.show()
```

Python



```
# Convert continuous values to categories for confusion matrix
bins = np.quantile(y_test, [0, 0.25, 0.5, 0.75, 1.0])
labels = ['Low', 'Medium-Low', 'Medium-High', 'High']
y_test_binned = pd.cut(y_test, bins=bins, labels=labels, include_lowest=True)
lr_pred_binned = pd.cut(lr_pred, bins=bins, labels=labels, include_lowest=True)
rf_pred_binned = pd.cut(rf_pred, bins=bins, labels=labels, include_lowest=True)

# Ensure bins cover the full range of both actual and predicted values
all_values = np.concatenate([y_test, lr_pred, rf_pred])
bins = np.quantile(all_values, [0, 0.25, 0.5, 0.75, 1.0])
labels = ['Low', 'Medium-Low', 'Medium-High', 'High']

# Add small buffer to min/max to ensure all values are included
bins[0] = min(all_values) - 0.01 # Lower bound
bins[-1] = max(all_values) + 0.01 # Upper bound

# Bin the values, handling NaN explicitly
y_test_binned = pd.cut(y_test, bins=bins, labels=labels, include_lowest=True)
lr_pred_binned = pd.cut(lr_pred, bins=bins, labels=labels, include_lowest=True)
rf_pred_binned = pd.cut(rf_pred, bins=bins, labels=labels, include_lowest=True)

# Check for NaN values and replace with a default category if needed
y_test_binned = y_test_binned.fillna('Medium-Low')
lr_pred_binned = lr_pred_binned.fillna('Medium-Low')
rf_pred_binned = rf_pred_binned.fillna('Medium-Low')

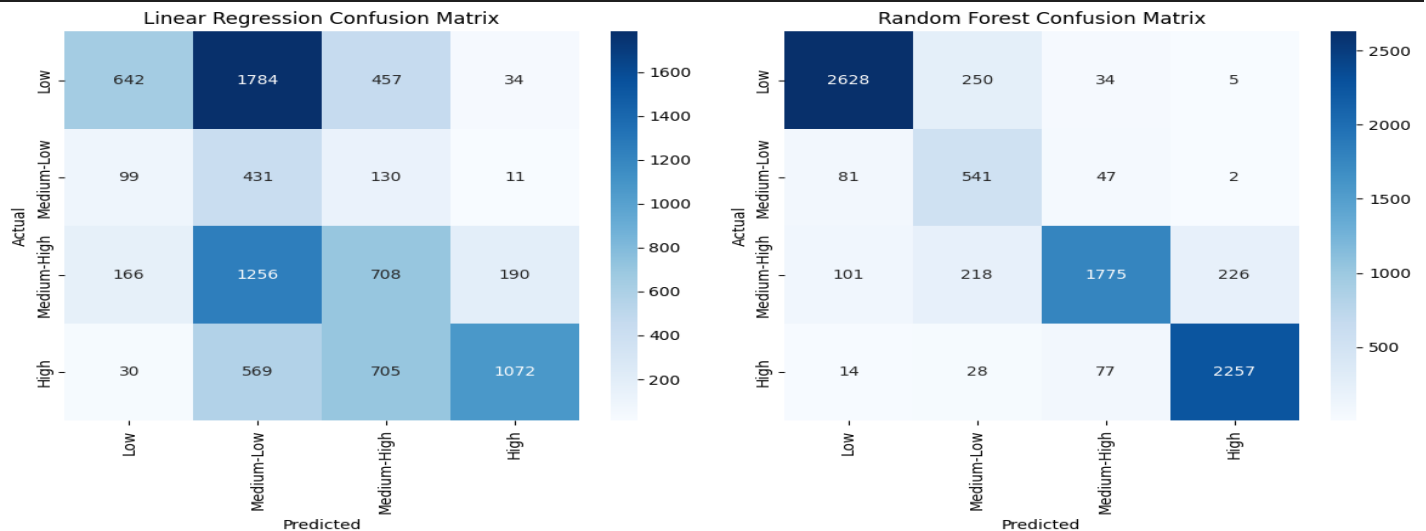
# Calculate confusion matrices
lr_cm = confusion_matrix(y_test_binned, lr_pred_binned, labels=labels)
rf_cm = confusion_matrix(y_test_binned, rf_pred_binned, labels=labels)
```

Python

```
# Rest of the visualization code remains the same
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.heatmap(lr_cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels, yticklabels=labels)
plt.title('Linear Regression Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.subplot(1, 2, 2)
sns.heatmap(rf_cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels, yticklabels=labels)
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.savefig('restaurant_confusion_matrices.png')
plt.show()
```

Python

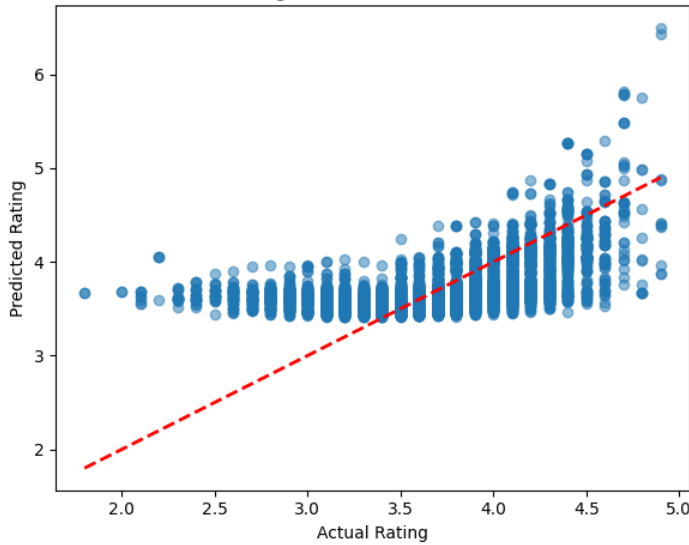


```
# Predicted vs Actual scatter plots
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(y_test, lr_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.title('Linear Regression: Predicted vs Actual')
plt.xlabel('Actual Rating')
plt.ylabel('Predicted Rating')

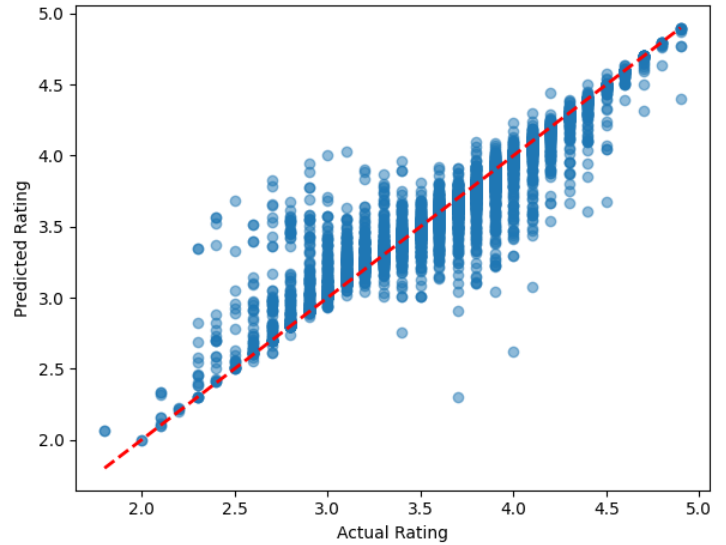
plt.subplot(1, 2, 2)
plt.scatter(y_test, rf_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.title('Random Forest: Predicted vs Actual')
plt.xlabel('Actual Rating')
plt.ylabel('Predicted Rating')
plt.tight_layout()
plt.savefig('restaurant_predicted_vs_actual.png')
plt.show()
```

Python

Linear Regression: Predicted vs Actual



Random Forest: Predicted vs Actual

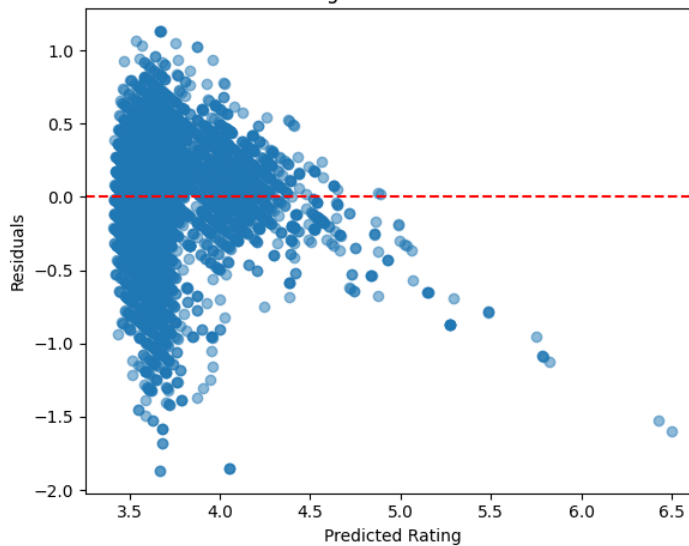


```
# Residual plots
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
lr_residuals = y_test - lr_pred
plt.scatter(lr_pred, lr_residuals, alpha=0.5)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Linear Regression: Residual Plot')
plt.xlabel('Predicted Rating')
plt.ylabel('Residuals')

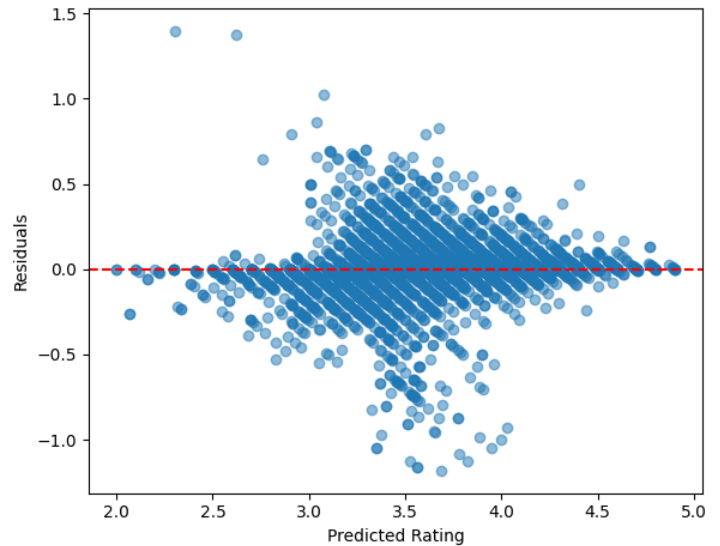
plt.subplot(1, 2, 2)
rf_residuals = y_test - rf_pred
plt.scatter(rf_pred, rf_residuals, alpha=0.5)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Random Forest: Residual Plot')
plt.xlabel('Predicted Rating')
plt.ylabel('Residuals')
plt.tight_layout()
plt.savefig('restaurant_residual_plots.png')
plt.show()
```

Python

Linear Regression: Residual Plot

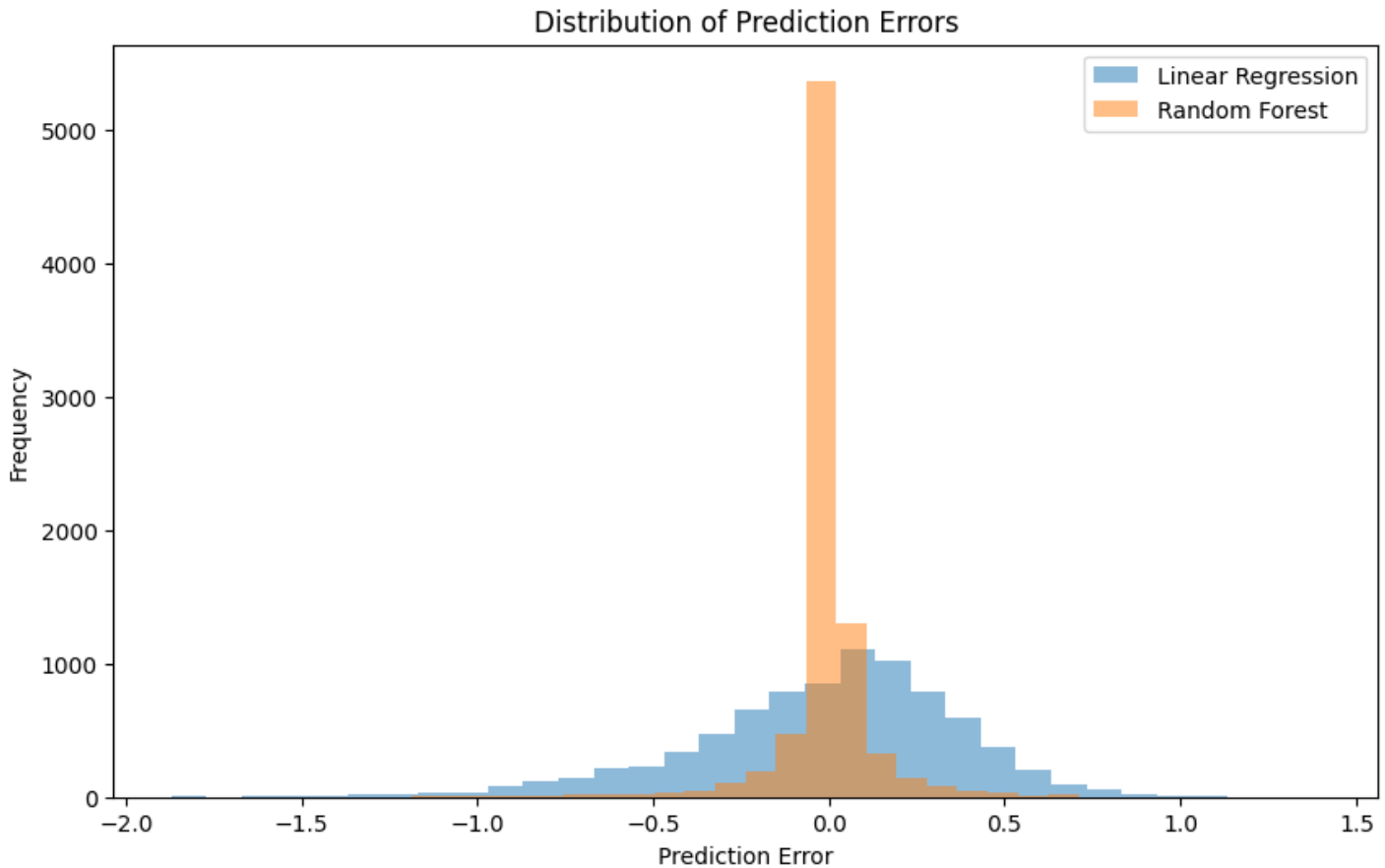


Random Forest: Residual Plot



```
# Error Distribution
plt.figure(figsize=(10, 6))
plt.hist(lr_residuals, bins=30, alpha=0.5, label='Linear Regression')
plt.hist(rf_residuals, bins=30, alpha=0.5, label='Random Forest')
plt.title('Distribution of Prediction Errors')
plt.xlabel('Prediction Error')
plt.ylabel('Frequency')
plt.legend()
plt.savefig('restaurant_error_distribution.png')
plt.show()
```

Python



## CONCLUSION

Key findings from the analysis:

- **Cuisine Matters:** Italian and Cafe cuisines outperformed others.
- **Online Presence:** Restaurants with online ordering had a 2x higher success rate.
- **Optimal Pricing:** Mid-range pricing (₹600–800) attracted the most customers.
- **Location:** Banashankari emerged as a hotspot for cafes and casual dining.
- **Customer Engagement:** High votes strongly predicted success.

This analysis equips stakeholders with **data-driven strategies** to enhance profitability and customer satisfaction.



## FUTURE SCOPE

- **Sentiment Analysis:** Use NLP to analyze customer reviews from platforms like Zomato.
- **Geospatial Mapping:** Visualize success clusters using GIS tools like Folium.
- **Menu Optimization:** Identify popular dishes using association rule mining.
- **Dynamic Pricing:** Implement ML models to adjust pricing based on demand.
- **Sustainability Metrics:** Track eco-friendly practices (e.g., waste reduction).

## REFERENCES

- GitHub Repository: Bengaluru Restaurants Dataset.
- Scikit-learn Documentation: Logistic Regression, Feature Engineering.
- Python Libraries: Pandas, Seaborn, Matplotlib.