Predicting Restaurant Metrics: A Comprehensive Data Analysis and Machine Learning Approach

1. Problem Definition

Imagine you're planning a dinner date with a friend. You've narrowed down the restaurant options based on cuisine and location, but budgeting is key. Zomato displays details like user ratings and cuisines, but often lacks a clear indication of price range. This project aims to bridge that gap by leveraging the power of machine learning. Here's the two-fold challenge we're tackling:

Predicting Average Cost for Two: Develop a model that accurately predicts the average cost a couple can expect to pay at a particular restaurant based on available Zomato data points. This goes beyond a simple category label (e.g., "Cheap") and provides a more specific estimate.

Classifying Price Range: Categorize restaurants into predefined price ranges (e.g., Budget-friendly, Mid-range, Expensive) based on the same data points. This offers a quick and easy way to understand the general affordability of a restaurant without needing a precise cost estimate.

With these models in place, users can gain a clearer picture of restaurant affordability before making a decision. This can be particularly helpful for those on a tight budget or seeking a specific dining experience (e.g., a luxurious fine-dining experience vs. a casual and affordable meal).

2. Data Analysis

The foundation of this project lies in the Zomato dataset, a treasure trove of information on restaurants across various locations. To enrich this data and gain a broader perspective, we incorporated a complementary "Country Code" dataset. This allows us to explore potential cost variations across geographical regions.

Here's a breakdown of some key data points we'll be working with:

Restaurant Details: Name, cuisine type, location (latitude & longitude), average user rating, and aggregate rating.

Additional Information: Whether the restaurant offers table booking, online delivery, in-house delivery, and access to a digital menu. This can provide insights into the overall service level and potentially influence pricing strategies.

Cost Indicators: Listed price range (e.g., Cheap, Moderate, Posh) and average cost for two people. These existing labels serve as a valuable reference point for model development and evaluation.

By analyzing this data, we aim to identify patterns and relationships that influence restaurant pricing. For instance, we might find a correlation between user ratings, location, and average cost, suggesting that highly-rated restaurants in prime locations tend to be more expensive.

import pandas as pd  
import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier

from sklearn.metrics import mean\_squared\_error, accuracy\_score  
df1=pd.read\_excel('C:\\Users\\ALSHA MOHAMMED\\Desktop\\ZOMATO.xlsx')

df1  
df2=pd.read\_excel('C:\\Users\\ALSHA MOHAMMED\\Desktop\\Country Code.xlsx')

df2  
df3 = pd.merge(df1, df2, on='Country Code')

### df3 df3.isnull().sum() df3.info()

### 3. EDA Concluding Remarks

### Exploratory Data Analysis (EDA) plays a crucial role in understanding the data and guiding the model development process. Here are some key insights gleaned from the EDA phase:

### Correlation Heatmap: Using a correlation heatmap, we can explore the relationships between various features. This helps us identify features that might be highly predictive of cost (e.g., user ratings) and those that might be redundant or have weak correlations (e.g., restaurant name and average cost). By understanding these relationships, we can focus on the most relevant data points when building our models.

### These insights from EDA inform our feature selection process, ensuring we focus on the data points that hold the most significant power in predicting restaurant costs.plt.figure(figsize=(15,10)) sns.heatmap(merged\_data.corr(), annot=True, cmap='coolwarm') plt.show() 4. Pre-processing Pipeline

Data pre-processing is an essential step to prepare the data for machine learning algorithms. Here's an in-depth look at the steps involved:

Handling Missing Values: Some data points like cuisine type or average rating might be missing. We can employ a two-pronged approach: replacing missing cuisine types with a generic category like "Unknown" and filling missing average ratings with the dataset's average value. This approach minimizes the impact of missing data while maintaining the overall integrity of the dataset.

Encoding Categorical Variables: Many features like "Has Table booking" or "Rating color" are categorical. We'll use a technique called Label Encoding to transform these into numerical values that machine learning algorithms can understand. For instance, "Has Table booking" can be encoded as

For instance, "Has Table booking" can be encoded as 1 (Yes) and 0 (No). This allows the model to learn the relationships between these categorical variables and the target variables (average cost and price range).

Numerical Conversion: Features like "Average Cost for two" were originally listed as strings. We'll convert them to numerical format (floats) for accurate calculations in our models. This ensures the model can perform mathematical operations on these values during the training process.

Feature Selection: Through feature selection, we'll focus on a subset of features likely to have the strongest influence on price prediction. This might include aspects like location (latitude and longitude), user engagement (votes), ratings, service availability (table booking, online delivery, etc.), and potentially even cuisine type (if our analysis suggests a correlation between cuisine and cost). By selecting the most relevant features, we can improve the efficiency and accuracy of our models.

By cleaning and transforming the data, we ensure our models are trained on a high-quality foundation. This meticulous pre-processing step helps the models learn from the most valuable information within the data.

df3.fillna({'Cuisines': 'Unknown', 'Aggregate rating': df3['Aggregate rating'].mean()}, inplace=True)  
df3  
df3.isnull().sum()  
  
label\_encoders = {}

for column in ['City', 'Address', 'Locality', 'Locality Verbose', 'Cuisines', 'Currency', 'Has Table booking', 'Has Online delivery', 'Is delivering now', 'Switch to order menu', 'Rating color', 'Rating text', 'Country']:

label\_encoders[column] = LabelEncoder()

df3[column] = label\_encoders[column].fit\_transform(df3[column])

df3

df3.info()

df3.drop(['Switch to order menu'],axis=1,inplace=True)

df3

5. Building Machine Learning Models

With the pre-processed data ready, it's time to build the machine learning models!

Predicting Average Cost:

Model Choice: A Random Forest Regressor is a powerful choice for this task. This model excels at handling complex, non-linear relationships between features and the target variable (average cost). It works by creating an ensemble of decision trees, each making predictions based on a random subset of features. The final prediction is an average of all the individual tree predictions, leading to increased accuracy and robustness. Random Forest Regressors are particularly well-suited for problems where the relationship between features and the target variable might not be easily modeled by a single equation.

Classifying Price Range:

Model Choice: To categorize restaurants into different price ranges, we'll employ a Random Forest Classifier. Similar to the regressor, this model utilizes multiple decision trees to classify a data point into one of the pre-defined price categories (e.g., Budget-friendly) based on the extracted features. Random Forest Classifiers are adept at handling classification tasks where the data can be neatly divided into distinct categories.

Model Training and Tuning:

Both the Random Forest Regressor and Classifier will be trained using a specific portion of the pre-processed data: the training set. This set represents a significant majority of the data (typically 80%). The model learns from patterns within the training set, establishing relationships between features like location and user ratings with the target variables (average cost and price range).

A crucial step involves hyperparameter tuning. Hyperparameters are settings within the model that can significantly influence its performance. In this project, we might adjust the "n\_estimators" hyperparameter, which controls the number of decision trees created in the Random Forest models. By experimenting with different values for n\_estimators, we aim to find a sweet spot that optimizes accuracy without overfitting the model to the training data.

Overfitting and the Importance of the Test Set:

Overfitting occurs when a model becomes too attuned to the training data, losing its ability to generalize effectively to unseen data. To prevent this, we utilize a separate "test set" containing the remaining portion of the data (usually 20%). This test set is kept unseen by the model during training. After training is complete, we evaluate the model's performance on the test set. This provides a more realistic assessment of how well the model will perform on new, real-world data.features = ['Latitude', 'Longitude', 'Votes', 'Aggregate rating', 'Has Table booking', 'Has Online delivery', 'Is delivering now']

X = df3[features]

# Predicting 'Average Cost for two'

y\_cost = df3['Average Cost for two']

# Predicting 'Price range'

y\_range = df3['Price range']

X\_train\_cost, X\_test\_cost, y\_train\_cost, y\_test\_cost = train\_test\_split(X, y\_cost, test\_size=0.2, random\_state=42)

X\_train\_range, X\_test\_range, y\_train\_range, y\_test\_range = train\_test\_split(X, y\_range, test\_size=0.2, random\_state=42)

# Model for 'Average Cost for two'

regressor = RandomForestRegressor(n\_estimators=100, random\_state=42)

regressor.fit(X\_train\_cost, y\_train\_cost)

# Model for 'Price range'

classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

classifier.fit(X\_train\_range, y\_train\_range)

6. Evaluating the Models

Once the models are trained and tuned, it's time to assess their effectiveness. Here's how we'll evaluate each model:

Predicting Average Cost:

Metric: Root Mean Squared Error (RMSE)

Explanation: RMSE measures the difference between the model's predicted average cost for two and the actual cost. Lower RMSE values indicate better performance. Evaluating the model on the test set, we calculate the RMSE to understand how closely the model's predictions align with real-world restaurant costs. For instance, an RMSE of 50 might suggest that the model's predictions typically deviate from the actual cost by around 50 units (which could be currency specific).

Classifying Price Range:

Metric: Accuracy Score

Explanation: Accuracy score measures the proportion of restaurants where the model correctly predicted the price range category (e.g., Budget-friendly). A higher accuracy score signifies better classification performance. We calculate the accuracy score on the test set to assess the model's ability to consistently categorize restaurants into the correct price ranges.

Interpreting the Results:

The achieved evaluation metrics provide a benchmark for the model's performance. Depending on the specific application and desired level of accuracy, the results might require further optimization. Techniques like feature engineering, exploring different machine learning algorithms, or adjusting hyperparameters can potentially enhance the model's performance.

Here are some additional considerations to keep in mind:

Data Quality: The quality and completeness of the data significantly impact model performance. Biases or inconsistencies in the data can lead to inaccurate predictions. It's important to ensure the data is reliable and representative of the target population (restaurants in this case).

Model Complexity: While complex models can capture intricate relationships, they are also more prone to overfitting. Striking a balance between model complexity and generalizability is crucial. A simpler model that performs well on the test set might be preferable to a complex model that performs exceptionally well on the training set but fails to generalize to unseen data.

Real-World Applicability: While the models offer valuable insights, it's important to acknowledge that restaurant pricing can be influenced by various factors beyond the data captured in Zomato. These might include factors like ambience, menu offerings, or special promotions. The models provide a data-driven prediction, but it's always wise to consider other aspects when making final decisions about restaurant choices.

from sklearn.metrics import mean\_squared\_error

y\_pred\_cost = regressor.predict(X\_test\_cost)

rmse\_cost = np.sqrt(mean\_squared\_error(y\_test\_cost, y\_pred\_cost))

print(f'RMSE for Average Cost for two: {rmse\_cost}')

**Classification Model**: The Random Forest Classifier's performance is evaluated using the accuracy score, which indicates the proportion of correct predictions.

from sklearn.metrics import accuracy\_score

y\_pred\_range = classifier.predict(X\_test\_range)

accuracy\_range = accuracy\_score(y\_test\_range, y\_pred\_range)

print(f'Accuracy for Price range: {accuracy\_range}')

This project successfully leveraged Zomato data and machine learning to build models capable of predicting restaurant costs. While further refinements can enhance accuracy, the project demonstrates the potential of data-driven approaches to demystify restaurant pricing for users.

Future Directions:

The world of restaurant price prediction is constantly evolving. Here are some exciting possibilities for future exploration:

Expanding Feature Set: Incorporating additional data points like menu items, restaurant ambience, or user reviews could potentially improve model performance by providing a more comprehensive picture of the dining experience.

Geographical Specificity: Developing separate models for different regions can account for potential variations in pricing structures across locations. Restaurant pricing can be influenced by factors like local ingredient costs and overall cost of living, and these factors might vary significantly across geographical regions.

Recommender Systems: Building recommender systems that leverage both price prediction and user preferences can offer a more personalized dining experience. Imagine a system that recommends restaurants that fit your desired cuisine, location, and budget.

This project not only highlights the potential of machine learning in the restaurant industry but also sets a foundation for future enhancements. Possible next steps include exploring other machine learning algorithms, tuning hyperparameters for better performance, and incorporating additional features such as user reviews and external economic indicators to further refine predictions.

In conclusion, the ability to predict restaurant metrics accurately can significantly impact business strategies, leading to improved customer satisfaction and better resource management. This project showcases the power of data-driven decision-making in achieving these goals.