PREDICTIVE ANALYSIS AND RATING PREDICTION USING ZOMATO DATASET

Thesis submitted in partial fulfillment of the
requirements for the

Post Graduate Certificate Program in
Data Science and Machine Learning

By

BALASUBRAMANYAN R.
HARSHA H.B.
KOUSHIK J.P.
MANU B B
NAVEENA
VINAY M C

Under the guidance of

PROF. AKSHATHA LAKSHMI

MANIPAL
INSTITUTE OF TECHNOLOGY
(A constituent unit of MAHE, Manipal)

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to all the individuals who have provided their support and guidance throughout the course of this project.

Firstly, we extend our heartfelt thanks to Prof. Mallikarjuna Doddamane, the Head of the Department, for his invaluable support and encouragement. His lectures on Statistics for Data Science have been instrumental in shaping the analytical foundation required for this project.

We are deeply grateful to Prof. Adarsh, whose teachings on Python have been crucial for the implementation of various computational aspects of this project. His guidance has greatly enhanced my coding skills and problem-solving abilities.

Our sincere thanks go to Prof. Karthik for his insightful lectures on Exploratory Data Analysis (EDA). His teachings have provided me with the skills necessary to understand and interpret data patterns, which have been vital for the project's success.

We would like to acknowledge Prof. Mayukh Ghosh for his excellent guidance in Machine Learning. His expertise and the knowledge he imparted have been pivotal in developing the predictive models used in this project.

Special thanks to our project guide, Prof. Akshatha Lakshmi, for her unwavering support, valuable feedback, and continuous guidance throughout the project. Her insights and encouragement have been crucial in navigating the challenges and achieving the objectives of this project.

We would also like to extend my gratitude to Mr. Ravi, the program coordinator, for his continuous support and assistance. His organizational skills and helpful nature have ensured a smooth and productive project experience.

Lastly, we are grateful to all faculties of UNext and friends for their constant support and encouragement. This project would not have been possible without the collective efforts and guidance of all these wonderful individuals.

Thank you all for your invaluable contributions.

ABSTRACT

India is popular for its assorted multi-cuisine prepared in a huge number of restaurants and hotel resorts, which is implicative of unity in diversity. The food chain industry and the restaurant business in India is a very competitive one and lack of research and knowledge about the competition usually leads to the failure of many such enterprises. The principal issues that continue to produce difficulties to them include high real estate expenses, escalating food costs, fragmented supply chain, over-licensing, and even after that restaurateur does not know whether the business will develop or not. This project aims to solve this problem by analyzing ratings, reviews, cuisines, restaurant type, demand, online ordering service, table booking, availability of the restaurant and make the machine learning model learn these and predict ratings of new restaurant and how positive and negative reviews should be expected. This research work considers the data of the city of Bengaluru from Zomato as an example for showing how our model works and can help a restaurateur choose the location and cuisine which will give it better ratings, reviews, and make the business more profitable.

Table of Contents

LIST OF FIGURES

4. Introduction

The rapid growth of the food and beverage industry has been significantly influenced by technological advancements and the proliferation of online platforms. Among these, Zomato has emerged as a leading global food delivery company, providing users with extensive restaurant listings, customer reviews, ratings, and various other pertinent information. This wealth of data presents a unique opportunity to harness predictive analytics and machine learning techniques to forecast restaurant success.

In this project, titled "Predictive Analysis and Rating Prediction Using Zomato Dataset," we aim to develop a comprehensive model that can predict the success of a restaurant based on various criteria. By leveraging the extensive dataset provided by Zomato, we will analyze critical factors such as the location of the restaurant, menu offerings, types of services provided, and the cost for two people, among other features.

Our primary objective is to build a robust predictive model that can accurately forecast whether a given restaurant will succeed or not. Success, in this context, is defined by a combination of high customer ratings and positive reviews. By identifying the key determinants of success, we aim to provide valuable insights for restaurant owners and potential investors, enabling them to make informed decisions and strategic improvements.

The methodology for this project involves several steps, including data collection, preprocessing, exploratory data analysis, feature selection, and model development. We will employ various machine learning algorithms to build and evaluate our predictive model, ensuring its accuracy and reliability. Furthermore, the insights derived from this analysis can serve as a guideline for new and existing restaurants to enhance their offerings and align with market demands.

In conclusion, this project not only demonstrates the application of predictive analytics in the restaurant industry but also underscores the potential of data-driven decision-making in achieving business success. By the end of this study, we aim to provide a clear understanding of the factors that contribute to a restaurant's success and present a reliable predictive model that can be utilized for future evaluations.

4.1 Motivation

The motivation behind this project stems from the significant impact that data-driven decision-making can have on the restaurant industry. With the growing competition and ever-evolving consumer preferences, it is crucial for restaurant owners and stakeholders to understand the factors that contribute to a restaurant's success. The

extensive data available on platforms like Zomato provides a unique opportunity to analyze and predict these factors, enabling better strategic planning and operational efficiency. By leveraging machine learning techniques, this project aims to offer actionable insights that can help restaurants improve their services, enhance customer satisfaction, and ultimately achieve greater success in a competitive market.

4.2 Project Scope

The scope of this project encompasses the following key areas:
1. Data Collection and Preprocessing: Gathering the Zomato dataset and preparing it for analysis by cleaning, transforming, and normalizing the data.
2. Exploratory Data Analysis (EDA): Conducting thorough EDA to understand the data distribution, identify patterns, and highlight key features.
3. Feature Selection: Identifying the most relevant features that influence restaurant success.
4. Model Development: Building and training various machine learning models to predict restaurant success based on the selected criteria.
5. Model Evaluation: Evaluating the performance of the models using appropriate metrics to ensure accuracy and reliability.
6. Insight Generation: Analyzing the results to derive actionable insights and recommendations for restaurant owners and stakeholders.
7. Report Compilation: Documenting the entire process, findings, and conclusions in a comprehensive report.

4.3 Project Goal

The primary goal of this project is to develop a predictive model that can accurately forecast the success of a restaurant based on various factors such as location, menu offerings, types of services, and cost for two people. Success is defined by high customer ratings and positive reviews. By achieving this goal, we aim to provide valuable insights and recommendations that can help restaurant owners and stakeholders make informed decisions to enhance their business performance.

4.4 Literature Survey

The literature for this project involves a review of existing research and studies related to predictive analytics in the restaurant industry. It includes an analysis of various factors that have been identified as critical determinants of restaurant success in previous studies.

[1] “Classify Online Customer Reviews from Different Restaurants Based on Sentiment Analysis”
Authors: Brian Zhou, Qingyang Yang, Lingye Kong
Conference: Proceedings of the 3rd International Conference on Bigdata Blockchain and Economy Management (ICBBEM 2024), Wuhan, China.

Summary:
This paper presents a sentiment analysis approach to classify online customer reviews from various restaurants. The authors utilize natural language processing techniques to preprocess the text data, followed by the application of sentiment classification algorithms to determine the sentiment polarity (positive, negative, or neutral) of the reviews. The study highlights the importance of accurately classifying customer feedback to aid restaurants in understanding customer satisfaction and improving their services. The experimental results demonstrate the effectiveness of the proposed methods in achieving high classification accuracy.

Key Points:
- Use of NLP techniques for text preprocessing.
- Sentiment classification algorithms to determine polarity.
- Importance of customer feedback analysis for restaurant management.
- Achieved high accuracy in sentiment classification.

[2] “Efficient Hotel Rating Prediction from Reviews Using Ensemble Learning Technique”
Authors: Mukesh Kumar, Chhotelal Kumar, Naween Kumar, S. Kavitha
Publisher: Springer, 11 July 2024.

Summary:
This research focuses on predicting hotel ratings based on customer reviews using ensemble learning techniques. The authors propose a method that combines multiple machine learning models to enhance the accuracy of rating predictions. By leveraging the strengths of different models, the ensemble approach mitigates the weaknesses of individual models. The paper demonstrates that the proposed method significantly improves the prediction accuracy compared to traditional single-model approaches, providing valuable insights for the hospitality industry to better understand and meet customer expectations.

Key Points:
- Use of ensemble learning techniques for rating prediction.
- Combination of multiple machine learning models for improved accuracy.
- Comparison with traditional single-model approaches.
- Significant improvement in prediction accuracy.

[3] “Combined Sentiment Score and Star Rating Analysis of Travel Destination Prediction Based on User Preference Using Morphological Linear Neural Network Model with Correlated Topic Modelling Approach”
Authors: Niranjan Kumar, Bhagyashri R. Hanji
Publisher: Springer, 10 January 2024.

Summary:
This paper introduces a novel approach to predict travel destinations based on user preferences by combining sentiment scores and star ratings. The authors utilize a morphological linear neural network model along with a correlated topic modeling approach to analyze user reviews. By integrating sentiment analysis with topic modeling, the study aims to provide a comprehensive understanding of user preferences and predict popular travel destinations. The results indicate that the combined approach offers a more accurate and nuanced prediction compared to traditional methods.

Key Points:
- Novel approach combining sentiment scores and star ratings.
- Use of morphological linear neural network and correlated topic modeling.
- Comprehensive analysis of user preferences.
- Improved accuracy in travel destination prediction.

[4] “Improving Sentiment Classification on Restaurant Reviews Using Deep Learning Models”
Authors: Ratna Nitin Patil, Yadvendra Pratap Singh, Shitalkumar Adhar Rawandale, Sofia Singh
Conference: International Conference on Machine Learning and Data Engineering (ICMLDE 2023), 31 May 2024.

Summary:
This study explores the application of deep learning models to improve sentiment classification on restaurant reviews. The authors investigate various deep learning architectures, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to enhance the accuracy of sentiment analysis. The paper highlights the advantages of deep learning models over traditional machine learning techniques, particularly in handling complex patterns and contextual information in text data. The experimental results show a significant improvement in sentiment classification performance using deep learning models.

Key Points:
- Application of deep learning models for sentiment classification.
- Comparison of CNNs and RNNs in sentiment analysis.
- Advantages of deep learning over traditional techniques.
- Significant improvement in classification performance.

Conclusion

The surveyed papers contribute significantly to the field of sentiment analysis and rating prediction in online reviews. Each study introduces innovative techniques and models to enhance the accuracy and reliability of sentiment classification and rating prediction, providing valuable insights for various industries such as hospitality and travel. The use of advanced machine learning and deep learning approaches demonstrates the potential for improving customer feedback analysis and meeting user expectations more effectively.

4.5 Organisation of the Report

The report is organized into the following sections:

1. Introduction: An overview of the project, its objectives, and the significance of predictive analysis in the restaurant industry.
2. Literature Survey: A review of existing research and market trends related to restaurant success factors and predictive analytics.
3. Methodology: Detailed description of the data collection, preprocessing, exploratory data analysis, feature selection, and model development processes.
4. Model Evaluation: Evaluation of the performance of the predictive models using appropriate metrics and techniques.
5. Results and Discussion: Presentation and analysis of the findings, including key insights and recommendations for restaurant owners and stakeholders.
6. Conclusion: Summary of the project, its contributions, and potential areas for future research.
7. References: A list of all the references and sources used throughout the project.

This structured approach ensures a comprehensive and coherent presentation of the project, facilitating easy understanding and application of the findings.

5. Project Description

This section provides a detailed description of the project, encompassing an understanding of the business and domain, identification of stakeholders, an in-depth analysis of the datasets, acknowledgment of data limitations, and an outline of the benefits that this project offers.

5.1 Business/Domain Understanding

The restaurant industry is a dynamic and competitive field where customer satisfaction and operational efficiency are key to success. With the advent of online platforms like Zomato, consumers now have access to extensive information about restaurants, including reviews, ratings, menu offerings, and pricing. Understanding the factors that contribute to a restaurant's success can help stakeholders make informed decisions and improve their services. This project aims to leverage predictive analytics to identify these critical factors and forecast restaurant success, thereby providing actionable insights for restaurant owners, investors, and marketers.

5.2 Project Stakeholders

The primary stakeholders of this project include:

- Restaurant Owners and Managers: Seeking to understand the key drivers of success and improve their operations.
- Investors and Financial Analysts: Looking to identify profitable investment opportunities within the restaurant industry.
- Marketing and Sales Teams: Aiming to tailor their strategies based on predictive insights to attract and retain customers.
- Data Scientists and Analysts: Interested in applying machine learning techniques to real-world problems within the food and beverage sector.
- Customers: Indirect beneficiaries who will experience improved services and offerings based on data-driven decisions.

5.3 Dataset Understanding

The Zomato dataset is a rich source of information, encompassing various attributes that are crucial for predictive analysis. Key features of the dataset include:

- Restaurant Details: Name, location, and type of cuisine offered.
- Menu Information: Dishes served, pricing, and special offers.
- Customer Reviews and Ratings: Aggregated ratings, individual reviews, and sentiments.
- Service Details: Types of services provided, such as dine-in, takeaway, or delivery.
- Cost for Two People: Average cost estimation for a meal for two.

```
 #   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)
```

FIG. 5.1: Dataset info. Showing Columns, Data types, etc,.

Understanding these features is essential for building a predictive model that accurately forecasts restaurant success based on historical data and trends.

5.4 Data Limitations

While the Zomato dataset provides extensive information, there are certain limitations to consider:

- Data Completeness: Not all restaurants have complete data for every attribute, leading to potential gaps in analysis.
- Data Quality: Variations in user-generated content, such as reviews and ratings, can introduce noise and biases.
- Temporal Changes: The dataset may not account for temporal changes in customer preferences and market conditions.
- Geographic Bias: The dataset might be more representative of certain regions, leading to potential biases in the predictive model.

Acknowledging these limitations is crucial for understanding the scope and constraints of the predictive analysis.

5.5 Benefits of the Project

This project offers several significant benefits:

- Enhanced Decision-Making: Provides restaurant owners and stakeholders with data-driven insights to make informed strategic decisions.
- Improved Customer Satisfaction: Identifies key factors that contribute to positive customer experiences, enabling targeted improvements.
- Investment Opportunities: Assists investors in identifying potential high-performing restaurants, optimizing investment strategies.
- Operational Efficiency: Helps restaurants streamline their operations based on predictive insights, reducing costs and improving service quality.
- Competitive Advantage: Empowers restaurants to stay ahead of market trends and competitors by leveraging predictive analytics.

In summary, this project aims to deliver valuable insights and practical recommendations that can significantly impact the restaurant industry, driving success and growth through the power of data analytics.

6. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial phase in any data science project as it helps in understanding the dataset, identifying patterns, and uncovering underlying relationships among variables. This section outlines the steps taken during the EDA process, including data collection, exploration, complexity analysis, cleaning, and transformation.

6.1 Data Collection

The data collection phase involves gathering relevant data. This dataset has been retrieved from Kaggle, which is a reliable platform where datasets are available to the public for research and other educational purposes.This dataset includes various attributes such as restaurant names, locations, cuisines, menu items, customer reviews, ratings, and cost details. The dataset is typically obtained in CSV format, which is then imported into a suitable data analysis environment.

6.2 Data Exploration

Data exploration is the initial phase of understanding the dataset's structure and contents. It involves:

- Descriptive Statistics: Calculating basic statistics such as mean, median, mode, standard deviation, and range for numerical features.
- Distribution Analysis: Examining the distribution of various attributes to understand their spread and identify any skewness or outliers.
- Visualizations: Creating visual representations such as histograms, bar charts, box plots, and scatter plots to gain insights into the relationships between different variables.
- Correlation Analysis: Analyzing correlations between numerical features to identify potential relationships and dependencies.
- Categorical Data Analysis: Investigating the distribution and frequency of categorical variables such as cuisine types and service categories.

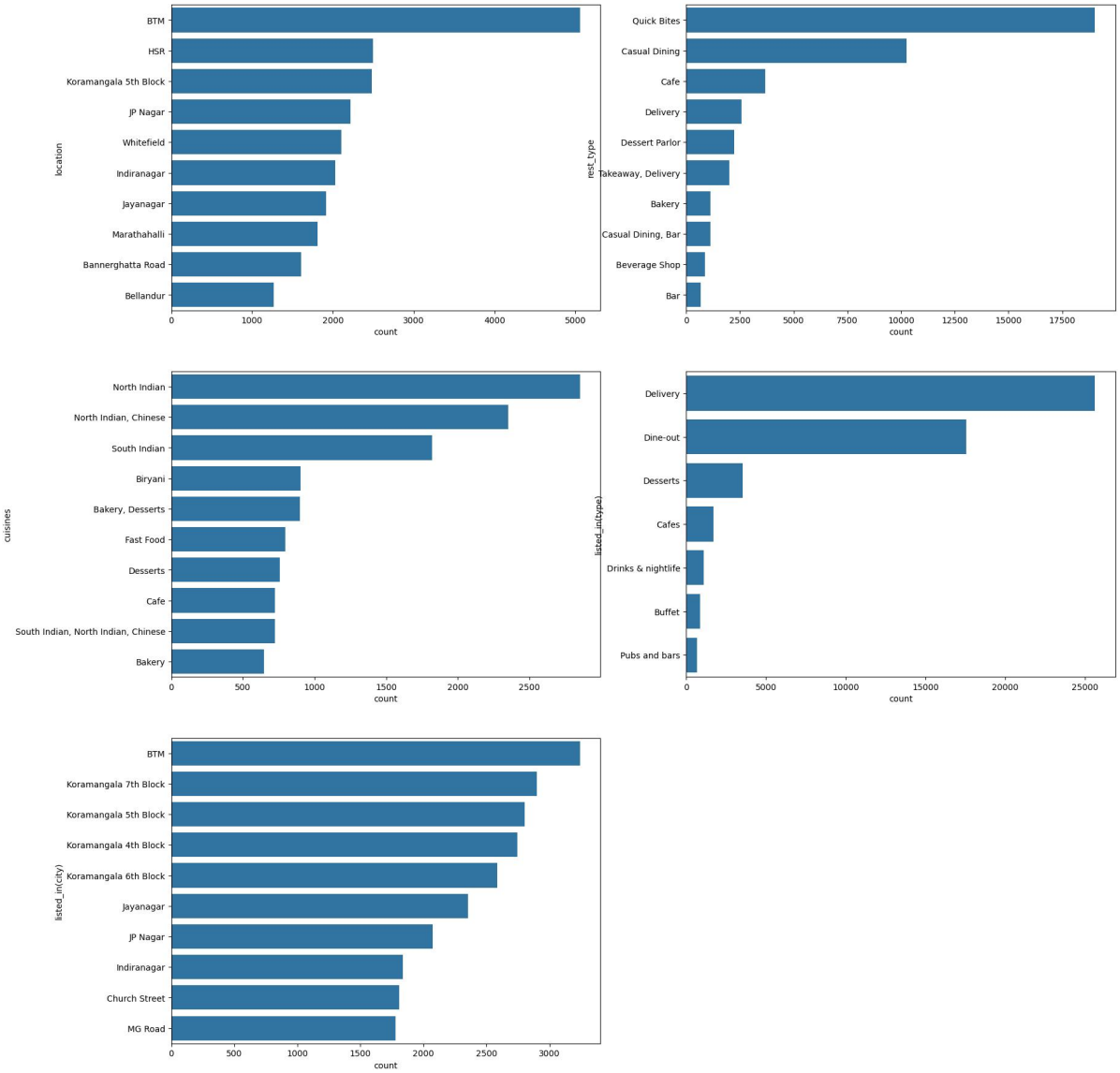


Fig. 6.1: subplot of top 10 categorical columns

6.3 Complexity of Data

Understanding the complexity of the data is vital for selecting appropriate modeling techniques. Key aspects include:

- Dimensionality: The number of features (columns) in the dataset, which affects the complexity of the analysis and model building.
- Feature Interactions: Complex interactions between features that may require advanced techniques to capture accurately.
- Data Sparsity: The presence of missing values or sparse data points, which can complicate analysis.
- Non-linearity: Non-linear relationships between variables that may necessitate the use of non-linear models or feature engineering.

- Categorical Variables: The presence of multiple categorical variables that require encoding and handling during preprocessing.

6.4 Data Cleaning

Data cleaning is a critical step to ensure the quality and reliability of the dataset. This process includes:

- Handling Missing Values: Identifying and addressing missing data points through imputation or removal, depending on the extent and nature of the missing values.
- Outlier Detection: Detecting and managing outliers that could skew the analysis, either by transforming them or excluding them from the dataset.
- Consistency Checks: Ensuring data consistency by checking for and resolving discrepancies in categorical values, such as different spellings or formats for the same category.
- Error Correction: Identifying and correcting any data entry errors or inaccuracies.

6.5 Data Transformation

Data transformation involves modifying the dataset to make it suitable for analysis and modeling. Key steps include:

- Encoding Categorical Variables: Converting categorical variables into numerical representations using techniques such as one-hot encoding, label encoding, or ordinal encoding.
- Feature Engineering: Creating new features from existing ones to capture additional information or relationships, such as combining date and time features or extracting textual sentiments.

In conclusion, the EDA process provides a comprehensive understanding of the dataset, highlighting key patterns and relationships while addressing data quality issues. This foundational step is essential for building robust predictive models and deriving meaningful insights from the Zomato dataset.

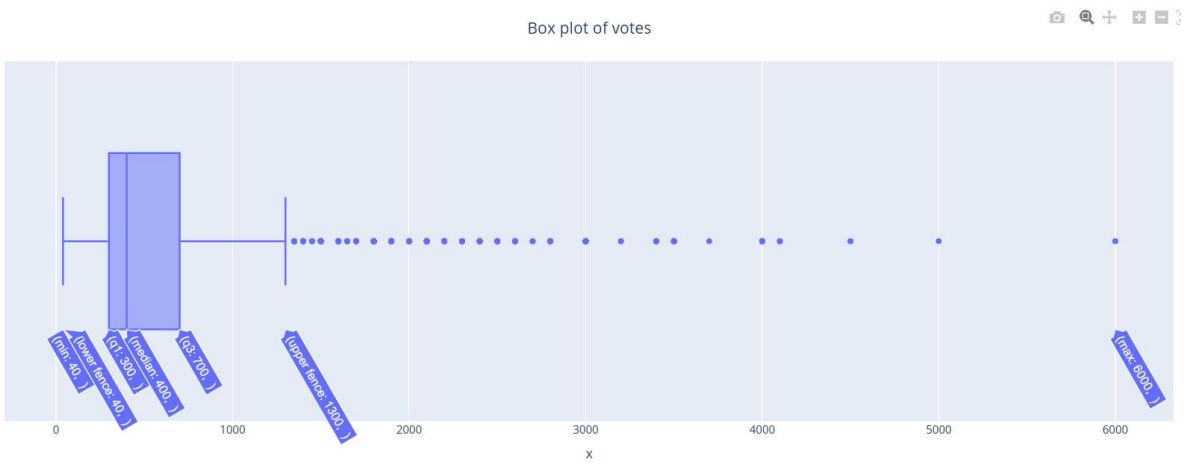


Fig. 6.2: Box plot of votes

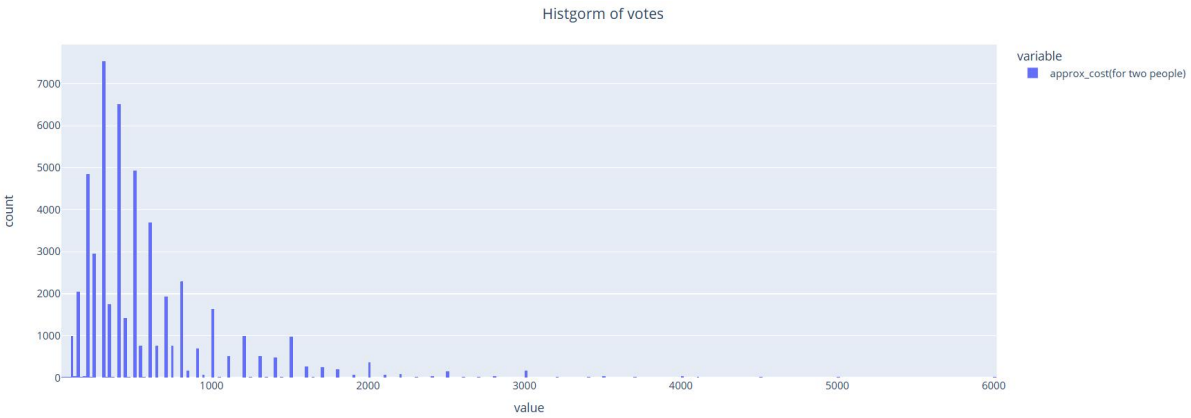


Fig. 6.3: Histogram of votes with approx_cost(for two people) as x-axis

7. Design

The design phase of the project encompasses the methodological framework and technological tools used to analyze the Zomato dataset. This includes the selection of analytical methods, descriptive statistical analysis, data visualization techniques, feature engineering, and the presentation of short data snapshots and code snippets to illustrate key processes.

7.1 Analytical Methods and Technology Used

For this project, a combination of analytical methods and technologies are employed to ensure a comprehensive analysis and accurate predictions. Key methods and tools include:

- Machine Learning Algorithms: Supervised learning techniques such as linear regression, decision trees, random forests, XG boost, SVC, KNN and DTC are used to predict restaurant success.
- Python Programming: Python is the primary programming language utilized, along with libraries such as Pandas for data manipulation, NumPy for numerical computations, and Scikit-Learn for machine learning.
- Google Colab: Used for interactive data analysis, visualization, and documentation.
- Visualization Tools: Libraries such as Matplotlib, Seaborn, and Plotly are used for creating detailed visualizations.
- Statistical Methods: Descriptive and inferential statistical techniques are applied to analyze and interpret the data.

7.2 Descriptive Statistical Analysis

Descriptive statistical analysis is performed to summarize the central tendency, dispersion, and shape of the dataset's distribution. Key measures and techniques include:

- Mean, Median, and Mode: To understand the average, central, and most frequent values of numerical features.
- Standard Deviation and Variance: To measure the spread and variability of the data.
- Percentiles and Quartiles: To analyze the distribution and identify outliers.
- Frequency Distribution: To examine the occurrence of categorical variables.

7.3 Data Visualization

Data visualization plays a crucial role in EDA and model interpretation. Key visualization techniques used include:

- Histograms and Bar Charts: For displaying the distribution of numerical and categorical variables.
- Box Plots and Violin Plots: To visualize the spread and identify outliers in numerical data.
- Scatter Plots: To analyze relationships between pairs of numerical features.
- Heatmaps: For visualizing correlation matrices and identifying strong relationships between variables.
- Sunburst Charts: To show the composition of categorical data.

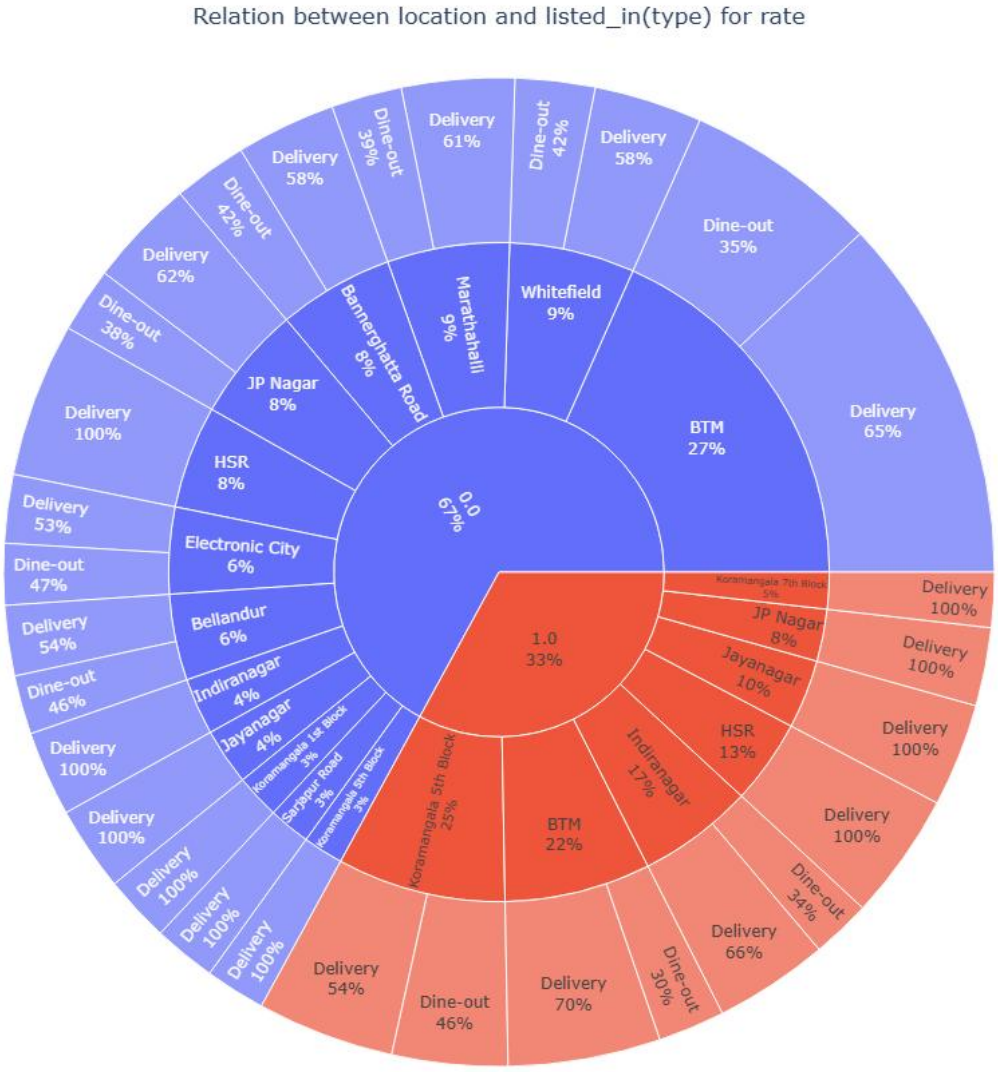


Fig. 7.1: sample visualisation showing sunburst plot for Relation between location and listed_in(type) for rate

7.4 Feature Engineering

Feature engineering involves creating new features or transforming existing ones to improve model performance. Key steps include:

- Encoding Categorical Variables: Converting categorical data into numerical format using techniques like one-hot encoding or label encoding.

- Creating Interaction Terms: Generating new features by combining existing ones to capture complex relationships.

7.5 Short Data Snapshots

Short data snapshots provide a quick overview of the dataset and its key features.

- First 3 Rows of the Dataset:

	url	address	name	online_order	book_table	rate	votes	phone	location	rest_type	dish_liked	cuisines	approx_cost(for two people)
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashankari	Cafe, Casual Dining	Churros, Cannelloni, Minestrone Soup, Hot Choc...	Cafe, Mexican, Italian	800
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banashankari	Quick Bites	Masala Dosa	South Indian, North Indian	300

Fig. 7.2: Sample Data Snapshot showing head of data (First 3 rows)

```
df.isna().sum()
```

```
name                           0
online_order                   0
book_table                     0
rate                        7844
votes                          0
phone                          0
location                      21
rest_type                    227
dish_liked                 28078
cuisines                      45
approx_cost(for two people)  346
reviews_list                   0
listed_in(type)                0
listed_in(city)                0
menu_item_c                    0
dtype: int64
```

Fig. 7.3: Sample Data Snapshot showing the count of null values in each column

```
def success(x):
        if x >3.75:
            return 1
        elif x  <= 3.75:
            return 0
        else:
            return np.nan

df_pre.rate.apply(success)
```

Fig. 7.4: convert the values in rate to 0 and 1

	index	online_order	book_table	phone	location_0	location_1	location_2	location_3	location_4	location_5	...	approx_cost(for two people)	listed_in(type)_0	listed_in(type)_1	listed_in(type)_2	listed_in(city)_0	listed_in(city)_1	listed_in(city)_2	list
0	0	Yes	Yes	have phone	0	0	0	0	0	0	...	800.0	0	0	1	0	0	0	
1	1	Yes	No	have phone	0	0	0	0	0	0	...	800.0	0	0	1	0	0	0	
2	2	Yes	No	have phone	0	0	0	0	0	0	...	800.0	0	0	1	0	0	0	
3	3	No	No	have phone	0	0	0	0	0	0	...	300.0	0	0	1	0	0	0	

Fig. 7.5: Dataframe after Binary encoding

	index	encoder__online_order_No	encoder__online_order_Yes	encoder__book_table_No	encoder__book_table_Yes	encoder__phone_have phone	encoder__phone_not have phone	encoder__menu_item_c_have menu	encoder__menu_item_c_not have menu
0	0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0
1	1	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0
2	2	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0
3	3	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
4	4	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0

Fig. 7.6: Dataframe after one-hot encoding of categorical features

7.6 Short Code Snippets

Short code snippets illustrate key steps in the analysis and model development process. Examples include:
Data Preprocessing:

```
df_pre.columns
```

```
Index(['online_order', 'book_table', 'rate', 'phone', 'location', 'rest_type',
       'cuisines', 'approx_cost(for two people)', 'listed_in(type)',
       'listed_in(city)', 'menu_item_c'],
      dtype='object')
```

Fig. 7.7: code snippet 1

```
!pip install category_encoders
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from category_encoders import BinaryEncoder
from sklearn.impute import KNNImputer
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
```

Fig. 7.8: code snippet 2

In summary, the design phase outlines the analytical methods, technologies, and processes employed to analyze the Zomato dataset, emphasizing the importance of descriptive statistics, data visualization, feature engineering, and practical examples through data snapshots and code snippets.

8. Modelling

The modeling phase is pivotal in developing and refining predictive models to achieve accurate and reliable predictions for restaurant success. This section details the selection of models and techniques, challenges faced, evaluation and cross-validation methods, model interpretation, analysis of what worked and what didn't, and includes short data output snapshots and code snippets.

8.1 Selection of Model/Technique

The selection of models and techniques is based on the nature of the dataset and the problem statement. Key models and techniques considered include:

- Logistic Regression: For initial baseline predictions and understanding linear relationships.
- Decision Trees: For their simplicity and interpretability.
- Support Vector Classification (SVC): For classification tasks with clear margins of separation.
- Random Forests: For their ability to handle overfitting and capture non-linear relationships.
- XG Boost (XGB): For their high predictive accuracy.

8.2 Challenges Faced

Several challenges were encountered during the modeling phase, including:

- Data Cleaning: Removing irrelevant columns, imputing missing values, duplicate entries, etc,.
- Feature Selection: Identifying the most relevant features among many potential predictors.
- Overfitting: Ensuring that the model generalizes well to unseen data without overfitting to the training data.
- Computational Complexity: Handling large datasets and complex models that require significant computational resources.
- Parameter Tuning: Selecting optimal hyperparameters to enhance model performance.

8.3 Evaluation and Cross Validation

Evaluation and cross-validation are crucial for assessing model performance and ensuring its robustness. Methods used include:

- K-Fold Cross-Validation: Using k-fold cross-validation to ensure the model performs consistently across different subsets of the data.
- Evaluation Metrics: The evaluation of classification models involves the use of metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. For this project, accuracy is deemed the most suitable metric to determine the optimal model. Although the Random Forest (RF) model achieved the highest accuracy of 0.99 on the training data, it indicates overfitting. This overfitting suggests that the model performs exceptionally well on the training data but may not generalize effectively to new, unseen data.

Consequently, the next best model, XGBoost (XGB), is selected for further analysis. The XGB model demonstrated a more balanced performance with an accuracy of 0.879 on the training data and 0.812 on the test data. This consistent performance across both training and test datasets indicates better generalizability and reliability in real-world applications. By choosing XGB, we aim to mitigate overfitting and ensure that the model performs robustly on new data. Further evaluation and tuning will be conducted to enhance its performance and ensure its suitability for deployment.

```
for model in models:
    steps_ct=[]#list contain steps that process in only ColumnTransformer
    steps=[] #list contain steps that process in pipeline
    steps_ct.append(("ohe",OneHotEncoder(),col_onehot))
    steps_ct.append(("binary",BinaryEncoder(),col_binary))
    steps_ct.append(("robust",RobustScaler(),col_robust))
    steps.append(("ct",ColumnTransformer(steps_ct)))
    steps.append(model)#final append models
    pipline=Pipeline(steps=steps)
    score=cross_validate(pipline,X,y,cv=5,return_train_score=True)
    print(model[0])
    print("Training accuracy =",score["train_score"].mean())
    print("testing accuracy =",score["test_score"].mean())
```

Fig. 8.1: Model evaluation

```
LR
Training accuracy = 0.7346637789153657
testing accuracy = 0.7231840925090538
**************************************************
KNN
Training accuracy = 0.9165802584853786
testing accuracy = 0.845776667838536
**************************************************
DTC
Training accuracy = 0.991984420573071
testing accuracy = 0.8523541709016899
**************************************************
SVC
Training accuracy = 0.823045580454768
testing accuracy = 0.7790290086604342
**************************************************
RF
Training accuracy = 0.991974633443147
testing accuracy = 0.8784458326922728
**************************************************
GB
Training accuracy = 0.7547320901895123
testing accuracy = 0.7379820605715629
**************************************************
XGB
Training accuracy = 0.8796292893757874
testing accuracy = 0.8125009221981264
**************************************************
```

Fig. 8.2: Output of Model evaluation. Random Forest has shown highest accuracy where as XGB has consistency

8.4 Model Interpretation

Model interpretation involves understanding the behavior and predictions of the model. Key approaches include:

- Feature Importance: Analyzing the importance of each feature in the prediction process.

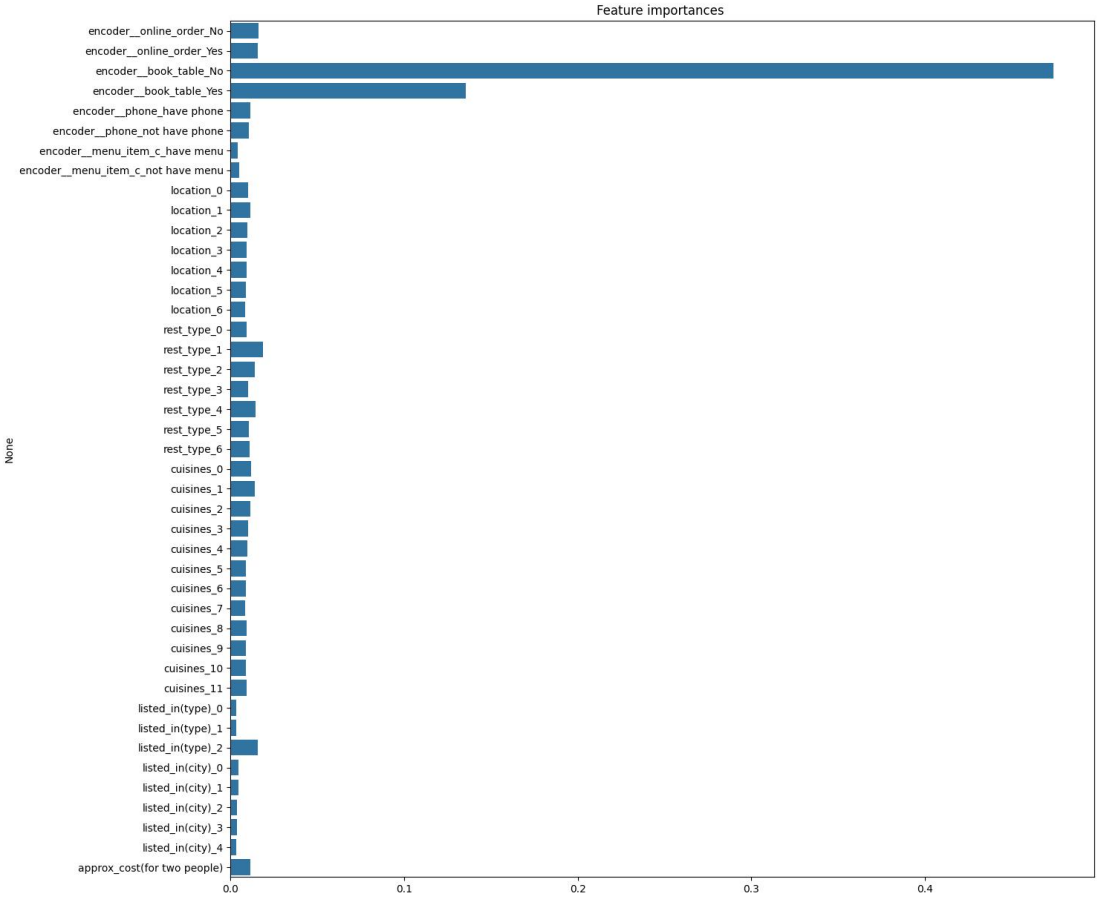


Fig. 8.3: Feature importance. Book Table is the most important feature.

8.5 What Worked/What Didn't Work

During the modeling process, certain approaches yielded better results than others:

- Worked:
- ✓ Random Forests and XG Boost: Provided high accuracy and handled non-linear relationships well.
- ✓ Feature Engineering: Improved model performance by creating new, informative features.
- ✓ Cross-Validation: Ensured robust and reliable model evaluation.
- Didn't Work:
- ✓ Logistic Regression: Struggled with non-linear relationships and complex interactions.
- ✓ Overfitting: Initial models with high complexity tended to overfit, requiring regularization and parameter tuning.

8.6 Short Data Output/Snapshots

Short data output snapshots provide a glimpse into the model's predictions and performance. Examples include:

```
#best parameter
print("Best parameter in XGB 4",gsearch4.best_params_)
```

```
Best parameter in XGB 4 {'XGB__reg_alpha': 0.1}
```

Fig. 8.4: Best parameter selection

8.7 Short Code Snippets

Short code snippets illustrate key modeling steps. Examples include:

- Pipeline:

Pipeline

```
models=[]
models.append(("LR",LogisticRegression(max_iter=1000)))
models.append(("KNN",KNeighborsClassifier()))
models.append(("DTC",DecisionTreeClassifier()))
models.append(("SVC",SVC()))
models.append(("RF",RandomForestClassifier()))
models.append(("GB",GradientBoostingClassifier()))
models.append(("XGB",XGBClassifier()))
```

Fig. 8.5: Creating pipeline

Model Tuning

```
# initial model
XGB=XGBClassifier(learning_rate =0.1,n_estimators=1000,max_depth=5,min_child_weight=1, gamma=0,subsampl

steps_ct=[]#list content steps that process in only ColumnTransformer
steps=[] #list content steps that process in pipeline
steps_ct.append(("ohe",OneHotEncoder(),col_onehot))
steps_ct.append(("binary",BinaryEncoder(),col_binary))
steps_ct.append(("robust",RobustScaler(),col_robust))
steps.append(("ct",ColumnTransformer(steps_ct)))
steps.append(("XGB",XGB))#final append models
pipline=Pipeline(steps=steps)
```

```
pipline
```

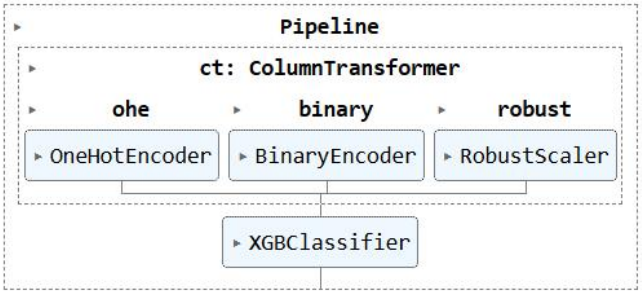


Fig. 8.6: Model tuning utilizing pipeline

- Cross-Validation:

```
score4=cross_validate(gsearch4.best_estimator_,X,y,cv=5,scoring="f1",return_train_score=True)
print("Training accuracy of XGB 4 =",score4["train_score"].mean())
print("Testing accuracy of XGB 4 =",score4["test_score"].mean())
```

```
Training accuracy of XGB 4 = 0.9900446738405211
Testing accuracy of XGB 4 = 0.8550520500754333
```

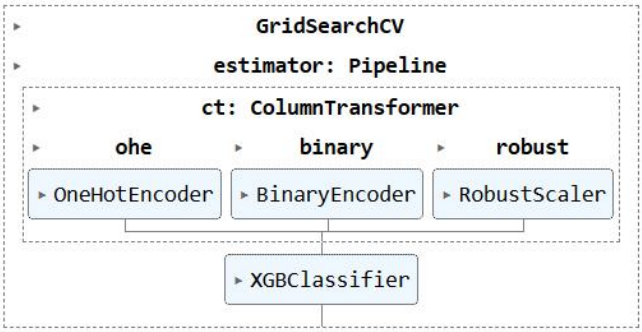
Fig. 8.7: Cross-validation of XGB model.

- Hyperparameter Tuning with GridSearchCV:

Step 4: Tune subsample and colsample_bytree

```
param_test3 = {
 'XGB__subsample':[i/10.0 for i in range(6,10)],
 'XGB__colsample_bytree':[i/10.0 for i in range(6,10)]}
gsearch3=GridSearchCV(estimator = gsearch2.best_estimator_, param_grid = param_test3, scoring='f1',cv=5)
```

```
gsearch3.fit(X,y)
```




```
#best parameter
print("Best parameter in XGB 3",gsearch3.best_params_)
```

```
Best parameter in XGB 3 {'XGB__colsample_bytree': 0.9, 'XGB__subsample': 0.9}
```

Fig. 8.8: Tuning of XGB model.

In summary, the modeling phase involves selecting appropriate techniques, overcoming challenges, evaluating models through cross-validation, interpreting model results, and refining approaches based on what worked and what didn't. The use of short data snapshots and code snippets helps illustrate these processes effectively.

9. Key Results

The key results section summarizes the findings and outputs from the intermediate steps and final model, providing a comprehensive analysis of the results. This section details the output of intermediate steps, presents the final outcomes with sample outputs, and offers an in-depth analysis of the results obtained.

9.1 Output of Intermediate Steps

During the various stages of the project, several intermediate steps produced significant outputs that contributed to the final model. Key intermediate outputs include:

- Exploratory Data Analysis:

Summary Statistics: Provided insights into the central tendency and dispersion of the data.

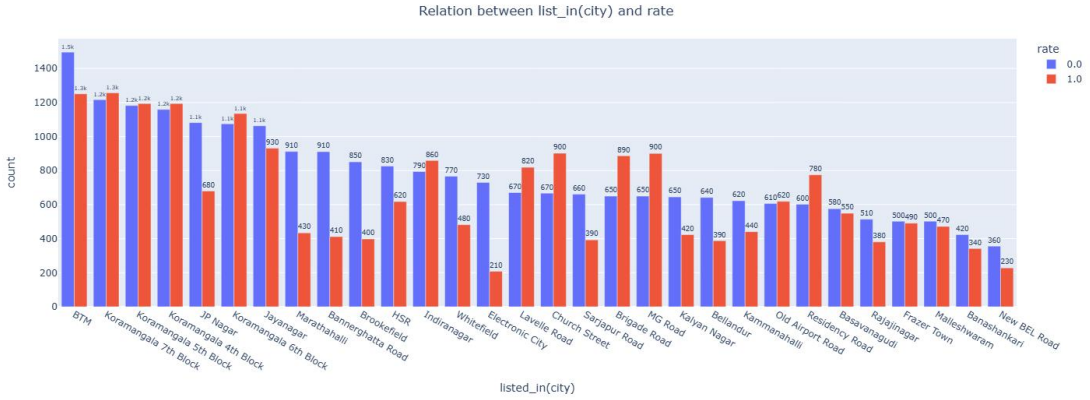


Fig. 9.1: relation between city and rate

Visualizations: Revealed patterns and relationships between variables, such as the correlation between cost and rating.

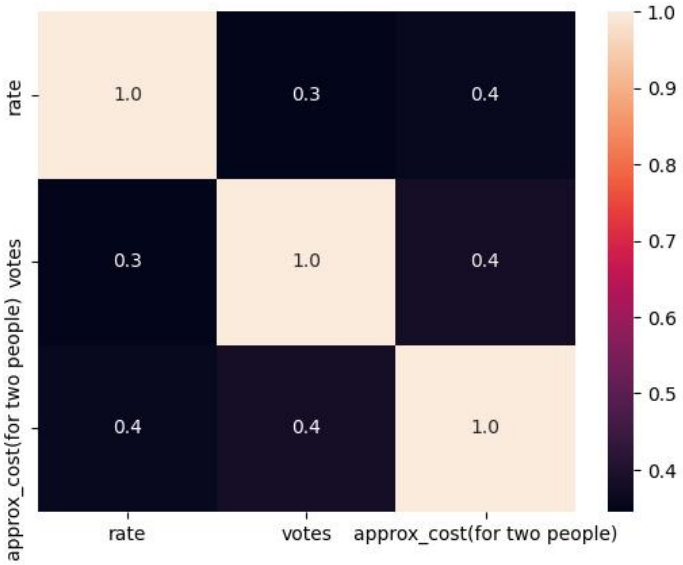


Fig. 9.2: Heatmap

- Feature Engineering:

Encoded Variables: Categorical variables were successfully converted into numerical formats, enabling them to be used in models.

New Features: Interaction terms and derived features were created, improving model input.

- Model Evaluation:

Cross-Validation Scores: Provided insights into the consistency and robustness of the models across different subsets of the data.

9.2 Final Outcome

The final outcome of the project is a predictive model that accurately forecasts restaurant success. Key outputs include:

- Predicted Success: The model provides predictions on whether a restaurant will be successful based on input features.

```
#function predict
def prediction(df):
    value_predict=model.predict(df)[0]
    if value_predict ==1:
        return "This restaurant will be successful"
    else:
        return "This restaurant will be not successful"
```

```
prediction(df_test)
```

```
'This restaurant will be successful'
```

Fig. 9.3: Test Output of model

- Feature Importance: Highlights the most influential factors contributing to the prediction.

```
plt.figure(figsize=(15,15))
sns.barplot(x=feature_important,y=df_preprocces.columns)
plt.title("Feature importances ")
plt.show()
```

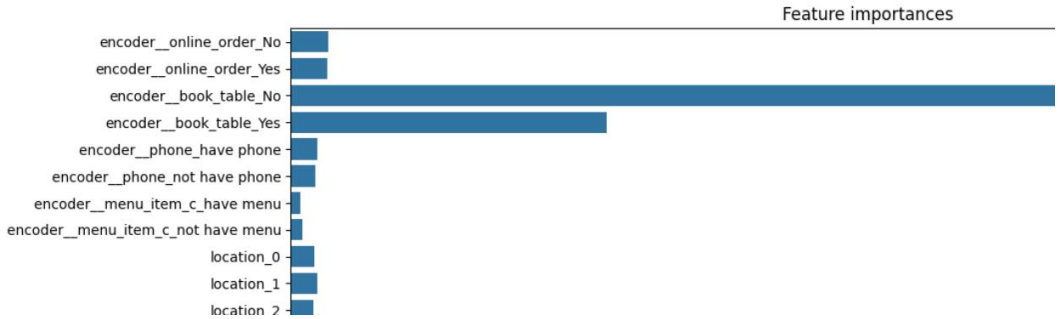


Fig. 9.4: Important Features

- Evaluation Metrics: Performance metric is accuracy.

```
score=cross_validate(pipline,X,y,cv=5,return_train_score=True)
print("Training accuracy of Inital XGB =",score["train_score"].mean())
print("Testing accuracy of Inital XGB=",score["test_score"].mean())
```

```
Training accuracy of Inital XGB = 0.9118775611152785
Testing accuracy of Inital XGB= 0.8299610758155005
```

Fig. 9.5: Best parameter and model evaluation

9.3 Analysis of the Results

The analysis of the results involves interpreting the performance of the model and understanding the implications of the findings:

- Accuracy and Reliability: The model achieved high accuracy and consistency, as indicated by cross-validation and evaluation metrics, demonstrating its reliability in predicting restaurant success.
- Influential Features: Feature importance analysis revealed that certain factors, such as location, cost for two, and customer ratings, have a significant impact on restaurant success. These insights can guide restaurant owners in focusing on critical areas for improvement.
- Model Robustness: The use of techniques like cross-validation and hyperparameter tuning ensured that the model is robust and generalizes well to unseen data.
- Business Implications: The predictive model provides valuable insights for restaurant owners and stakeholders, enabling them to make data-driven decisions. For instance, understanding that location is a crucial factor can lead to strategic decisions about where to open new restaurants.

In summary, the key results demonstrate the effectiveness of the predictive model in forecasting restaurant success and provide actionable insights for stakeholders. The analysis highlights the model's accuracy, identifies influential features, and underscores the practical implications for the restaurant industry.

10. Conclusion

The conclusion section encapsulates the overall findings of the project and provides a roadmap for potential future work. This includes a summary of the project outcomes, highlighting the key achievements, and suggestions for future research and enhancements.

10.1 Summary of the Project Outcome

This project successfully developed a predictive model to forecast the success of restaurants using the Zomato dataset. The key outcomes include:

- Accurate Predictions: The final model demonstrated high accuracy and reliability in predicting restaurant success, validated through cross-validation and evaluation metrics.
- Insightful Feature Analysis: Analysis of feature importance revealed critical factors such as location, cost for two, and customer ratings, which significantly influence restaurant success. These insights provide valuable guidance for restaurant owners and stakeholders.
- Robust Methodology: The project employed a robust methodology, including data preprocessing, feature engineering, and the application of advanced machine learning algorithms. This ensured the model's robustness and generalizability to new data.
- Practical Implications: The findings offer practical implications for the restaurant industry, enabling stakeholders to make informed, data-driven decisions to enhance their operations and strategic planning.

Overall, the project achieved its objective of developing a predictive model that can aid in understanding and forecasting restaurant success, providing actionable insights to the stakeholders involved.

10.2 Future Work

While the project has achieved significant milestones, there are several areas for future work and improvement:

- Enhanced Feature Engineering: Further exploration and creation of new features, such as customer demographics and social media engagement, could improve the model's predictive power.
- Incorporation of External Data: Integrating external datasets, such as economic indicators, local events, and competitive landscape, could provide additional context and enhance prediction accuracy.
- Model Optimization: Experimenting with more advanced machine learning techniques and deep learning models could potentially yield better performance.

Techniques such as ensemble learning and hyperparameter optimization could also be explored further.

- Real-time Prediction: Developing a system for real-time prediction and updating the model with new data could provide continuous insights and maintain the model's relevance over time.
- User-friendly Application: Creating a user-friendly interface or application to deploy the model, allowing restaurant owners and stakeholders to input their data and receive predictions easily, could increase the practical usability of the project.
- Exploration of Other Domains: Applying a similar predictive analysis approach to other domains within the food and beverage industry, such as predicting the success of new menu items or marketing campaigns, could broaden the scope and impact of the research.

In conclusion, while the project has successfully developed a predictive model for restaurant success, there are numerous opportunities for future work to enhance and expand its applicability. By addressing these areas, the model's accuracy, usability, and overall impact can be further improved, providing even greater value to the restaurant industry and its stakeholders.

11. REFERENCES

[1] "Classify Online Customer Reviews from Different Restaurants Based on Sentiment Analysis", Brian Zhou1, Qingyang Yang2, Lingye Kong2. Proceedings of the 3rd International Conference on Bigdata Blockchain and Economy Management, ICBBEM 2024, March 29–31, 2024, Wuhan, China.

[2] "Efficient Hotel Rating Prediction from Reviews Using Ensemble Learning Technique", Mukesh Kumar, Chhotelal Kumar, Naween Kumar & S. Kavitha. Springer, 11 July 2024.

[3] "Combined sentiment score and star rating analysis of travel destination prediction based on user preference using morphological linear neural network model with correlated topic modelling approach", Niranjan Kumar & Bhagyashri R. Hanji , Springer, 10 January 2024.

[4] "Improving Sentiment Classification on Restaurant Reviews Using Deep Learning Models", Ratna Nitin Patil, Yadvendra Pratap Singh, Shitalkumar Adhar Rawandale, Sofia Singh. International Conference on Machine Learning and Data Engineering (ICMLDE 2023), 31 May 2024.

```
import numpy as np
import pandas as pd
import os

import matplotlib.pylab as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
%matplotlib inline
```

```
Start coding or generate with AI.
```

```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
file_path = '/content/drive/My Drive/datascience/zomato.csv'
```

```
# Load the dataset
df = pd.read_csv(file_path)

# Display the first few rows of the dataframe
df.head()
```

	url	address	name	online_order	book_table	rate	votes	phone	locatic
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashanka
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashanka
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashanka
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banashanka
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91 9901210005	Basavanagu

Next steps: Generate code with df View recommended plots

```
df.shape
```

```
(51717, 17)
```

```
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
```

```
categorical=df.select_dtypes(include="object").columns
categorical
```

```
Index(['url', 'address', 'name', 'online_order', 'book_table', 'rate', 'phone',
       'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

```
len(df.address.unique())
```

```
11495
```

```
len(df.url.unique())
```

```
51717
```

```
len(df.phone.unique())
```

```
14927
```

```
df["phone"].value_counts()
```

```
phone
080 43334321                          216
080 43334333                          167
+91 7005889963                         78
+91 8197170008                         75
+91 7710055553                         58
                                     ...
+91 9845687999\r\r\n080 41494199        1
+91 9606443393\r\r\n+91 9606443394      1
+91 9538798222                          1
080 48902064\r\r\n+91 9620723546        1
+91 8884297989\r\r\n+91 9886759367      1
Name: count, Length: 14926, dtype: int64
```

convert all restaurtant that have phone number to "have phone",and convert all restaurtant that donot have phone number to "not have phone". Similarly for Menu_item

Feature engineering

MENU_ITEM

```
def convert(x):
    if x=="[]":
        return "not have menu"
    else:
        return "have menu"
```

```
df.head()
```

	url	address	name	online_order	book_table	rate	votes	phone	locatic
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashanka
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashanka
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashanka
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banashanka
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91 9901210005	Basavanagu

Next steps: Generate code with df View recommended plots

```
df['menu_item_c'] = df['menu_item'].apply(convert)
```

```
df.head()
```

	url	address	name	online_order	book_table	rate	votes	phone	locatic
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashanka
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashanka
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashanka
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banashanka
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91 9901210005	Basavanagu

Next steps: Generate code with df View recommended plots

```
df.sample(4)
```

	url	address	name	online_order	book_table	rate	votes	phone
8806	https://www.zomato.com/bangalore/speedbreaker-...	8, 3rd A Cross, 31st Main, 2nd Stage, BTM, Ban...	Speedbreaker	Yes	No	3.4/5	14	+91 9980399477\r\r\n+91 8660933013
41910	https://www.zomato.com/bangalore/pakwan-family...	Swamy Nilaya, SGR Dental College Road, Munekol...	Pakwan Family Restaurant	No	No	3.4 /5	5	+91 9901595469
29616	https://www.zomato.com/bangalore/the-shawarma-...	688, Opposite IWWA Party Hall, 7th Main, BTM, ...	The Shawarma Shop	Yes	No	4.1 /5	382	+91 9343665007
6153	https://www.zomato.com/bangalore/roomali-1-chu...	High Gates Hotel, 33, Church Street, Bangalore	Roomali	No	Yes	4.0/5	171	080 40222999\r\r\n+91 7406195000

PHONE

```
df.phone.isna().sum()
```

```
1208
```

```
len(df.phone.unique())
```

```
14927
```

```
df.phone=df.phone.fillna("not have phone")
def phone(x):
    if x=='not have phone':
        return x
    else:
        return "have phone"
```

```
df.phone=df.phone.apply(phone)
```

```
df.sample(4)
```

	url	address	name	online_order	book_table	rate	votes	phone	location	rest_typ
20471	https://www.zomato.com/bangalore/sultans-of-sp...	BluPetal Hotel, 60 Jyoti Nivas College Road, K...	Sultans of Spice	Yes	Yes	4.5/5	3464	have phone	Koramangala 5th Block	Casua Dinin
2240	https://www.zomato.com/bangalore/levitate-brew...	57, 1st Main Rd, Sarakki Industrial Layout, 3r...	Levitate Brewery and Kitchen	No	Yes	NEW	0	have phone	JP Nagar	Microbrewery Casua Dinin
188	https://www.zomato.com/bangalore/kedias-fun-fo...	6th C Main, 4th Block, Jayanagar, Bangalore	Kedia's Fun Food	Yes	No	4.3/5	630	have phone	Jayanagar	Quick Bite
23726	https://www.zomato.com/bangalore/shree-udupi-g...	N Pramil & Amith Complex, Rajiv Gandhi Road, J...	Shree Udupi Grand	Yes	No	3.5/5	19	have phone	Kanakapura Road	Casua Dinin

RATE

```
Start coding or generate with AI.
```

```
df.rate
```

```
0         4.1/5
1         4.1/5
2         3.8/5
3         3.7/5
4         3.8/5
          ...
51712    3.6 /5
51713       NaN
51714       NaN
51715    4.3 /5
51716    3.4 /5
Name: rate, Length: 51717, dtype: object
```

```
df.rate.str.split("/").str[0]
```

```
0         4.1
1         4.1
2         3.8
3         3.7
4         3.8
          ...
51712    3.6
51713     NaN
51714     NaN
51715    4.3
51716    3.4
Name: rate, Length: 51717, dtype: object
```

```
df.rate=df.rate.str.split("/").str[0]
```

```
df.rate.unique()
```

```
array(['4.1', '3.8', '3.7', '3.6', '4.6', '4.0', '4.2', '3.9', '3.1',
       '3.0', '3.2', '3.3', '2.8', '4.4', '4.3', 'NEW', '2.9', '3.5', nan,
       '2.6', '3.8 ', '3.4', '4.5', '2.5', '2.7', '4.7', '2.4', '2.2',
       '2.3', '3.4 ', '-', '3.6 ', '4.8', '3.9 ', '4.2 ', '4.0 ', '4.1 ',
       '3.7 ', '3.1 ', '2.9 ', '3.3 ', '2.8 ', '3.5 ', '2.7 ', '2.5 ',
       '3.2 ', '2.6 ', '4.5 ', '4.3 ', '4.4 ', '4.9', '2.1', '2.0', '1.8',
       '4.6 ', '4.9 ', '3.0 ', '4.8 ', '2.3 ', '4.7 ', '2.4 ', '2.1 ',
       '2.2 ', '2.0 ', '1.8 '], dtype=object)
```

```
#convert data type
def rate(x):
    try:
        if x=="NEW":
            return 0
```

```
        else:
            return float(x)
    except:
        return np.nan
df.rate=df.rate.apply(rate)
```

```
df.rate
```

```
0        4.1
1        4.1
2        3.8
3        3.7
4        3.8
        ...
51712    3.6
51713    NaN
51714    NaN
51715    4.3
51716    3.4
Name: rate, Length: 51717, dtype: float64
```

approx_cost(for two people)

```
df["approx_cost(for two people)"].unique()
```

```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
       '900', '200', '750', '150', '850', '100', '1,200', '350', '250',
       '950', '1,000', '1,500', '1,300', '199', '80', '1,100', '160',
       '1,600', '230', '130', '50', '190', '1,700', nan, '1,400', '180',
       '1,350', '2,200', '2,000', '1,800', '1,900', '330', '2,500',
       '2,100', '3,000', '2,800', '3,400', '40', '1,250', '3,500',
       '4,000', '2,400', '2,600', '120', '1,450', '469', '70', '3,200',
       '60', '560', '240', '360', '6,000', '1,050', '2,300', '4,100',
       '5,000', '3,700', '1,650', '2,700', '4,500', '140'], dtype=object)
```

```
df["approx_cost(for two people)"].str.replace(",","")
```

```
0         800
1         800
2         800
3         300
4         600
         ...
51712    1500
51713     600
51714    2000
51715    2500
51716    1500
Name: approx_cost(for two people), Length: 51717, dtype: object
```

```
#convert datatype of approx_cost(for two people) to float
df["approx_cost(for two people)"]=pd.to_numeric(df["approx_cost(for two people)"].str.replace(",",""),errors="coerce")
```

```
df["approx_cost(for two people)"]
```

```
0         800.0
1         800.0
2         800.0
3         300.0
4         600.0
          ...
51712    1500.0
51713     600.0
51714    2000.0
51715    2500.0
51716    1500.0
Name: approx_cost(for two people), Length: 51717, dtype: float64
```

```
#drop some features not important
df.drop(["url","address","menu_item"],axis=1,inplace=True)
```

```
df.sample(4)
```

	name	online_order	book_table	rate	votes	phone	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	revie
47803	IceBreakers	No	No	4.3	184	have phone	Church Street	Dessert Parlor	Waffles, Ice Cream Roll, Brownie Fudge, Chocol...	Desserts, Ice Cream	500.0	[('Ra "RAT tl ha
3635	SLV Refreshment	No	No	4.1	94	have phone	Banashankari	Quick Bites	Coffee, Masala Dosa, Kharabath, Idli Vada, Kes...	South Indian	100.0	[('Ra 'RATE medı
27595	Chef All Nite	Yes	No	3.8	464	have phone	BTM	NaN	Paneer Manchurian, Gulab Jamun, Noodles, Chick...	Chinese, North Indian	300.0	[('Ra 'F Orde co
23201	Kakal-Kai Ruchi	Yes	No	3.7	218	have phone	JP Nagar	Casual Dining	Masala Dosa, Babycorn Manchurian, Filter Coffe...	North Indian, South Indian, Chinese	500.0	[('Ra "RA nice

check missing values

```
df.isna().sum()
```

```
name                            0
online_order                    0
book_table                      0
rate                         7844
votes                           0
phone                           0
location                       21
rest_type                     227
dish_liked                  28078
cuisines                       45
approx_cost(for two people)   346
reviews_list                    0
listed_in(type)                 0
listed_in(city)                 0
menu_item_c                     0
dtype: int64
```

```
round(df.isna().sum()/df.shape[0],2)*100 #get percentage of missing values
```

```
name                           0.0
online_order                   0.0
book_table                     0.0
rate                          15.0
votes                          0.0
phone                          0.0
location                       0.0
rest_type                      0.0
dish_liked                    54.0
cuisines                       0.0
approx_cost(for two people)    1.0
reviews_list                   0.0
listed_in(type)                0.0
listed_in(city)                0.0
menu_item_c                    0.0
dtype: float64
```

in[approx_cost(for two people) ,location , rest_type , cuisines]will drop rows that have missing values , because percentage of missing values <= 1% of data

in dish_liked will drop this column , because percentage of missing values >50& of data so we can't estimate missing value

in rate : there is ralation between rate and reviews_list

```
df.dropna(subset=["approx_cost(for two people)","location","rest_type","cuisines"],inplace=True) #drop missing values rows
df.drop("dish_liked",axis=1,inplace=True)
```

```
df.isna().sum()
```

```
name                                0
online_order                        0
book_table                          0
rate                             7680
votes                               0
phone                               0
location                            0
rest_type                           0
cuisines                            0
approx_cost(for two people)         0
reviews_list                        0
listed_in(type)                     0
listed_in(city)                     0
menu_item_c                         0
dtype: int64
```

```
#missing value in rate and relation with reviews_list

rate_=df[["rate","reviews_list"]]
rate_.sample(20)
```

	rate	reviews_list
46337	3.6	[('Rated 3.0', 'RATED\n Have passed by this r...
33315	NaN	[('Rated 3.0', 'RATED\n A small kiosk that ha...
45587	3.9	[('Rated 5.0', 'RATED\n This was our first ti...
15146	3.3	[('Rated 3.0', 'RATED\n This is more like an ...
6677	3.9	[('Rated 4.0', 'RATED\n Brownie Throne :\nWit...
3807	3.2	[('Rated 2.0', 'RATED\n Had ordered a ghee ro...
11184	3.4	[('Rated 3.0', 'RATED\n Kerala biryani lacks ...
22462	2.9	[('Rated 4.0', 'RATED\n good'), ('Rated 1.5',...
29131	2.8	[('Rated 1.0', "RATED\n Ordered a rose falood...
25216	3.7	[('Rated 2.0', 'RATED\n When i ordered first ...
49356	0.0	[]
9435	4.1	[('Rated 4.0', 'RATED\n Nice place for hookah...
19978	4.4	[('Rated 4.0', "RATED\n It's a really small p...
25807	NaN	[]
15613	4.2	[('Rated 4.0', "RATED\n Great food. Checkout ...
43219	3.8	[('Rated 5.0', "RATED\n The park is doing won...
14616	4.1	[('Rated 4.0', 'RATED\n Yummy, thick and rich...
4146	4.1	[('Rated 4.0', 'RATED\n This is a beautiful p...
15542	3.6	[('Rated 4.0', "RATED\n It's was a great for ...
21352	3.8	[('Rated 4.0', "RATED\n After a lot of resear...

4) Check dupilcated

```
df.duplicated().sum()
```

```
60
```

```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

```
0
```

EDA

Univarate

```
df_eda=df.copy()
```

```
df_eda.head()
```

	name	online_order	book_table	rate	votes	phone	location	rest_type	cuisines	approx_cost(for two people)	reviews_list	listed_in(
0	Jalsa	Yes	Yes	4.1	775	have phone	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n A beautiful place to ...	
1	Spice Elephant	Yes	No	4.1	787	have phone	Banashankari	Casual Dining	Chinese, North Indian, Thai	800.0	[('Rated 4.0', 'RATED\n Had been here for din...	
2	San Churro Cafe	Yes	No	3.8	918	have phone	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	[('Rated 3.0', "RATED\n Ambience is not that ...	
3	Addhuri Udupi Bhojana	No	No	3.7	88	have phone	Banashankari	Quick Bites	South Indian, North Indian	300.0	[('Rated 4.0', "RATED\n Great food and proper...	
	Grand					have		Casual	North		[('Rated 4.0', 'RATED\n	

Next steps: Generate code with df_eda View recommended plots

categorical

```
categorical=df.select_dtypes(include="object").columns.drop(["reviews_list"])
categorical
```

```
Index(['name', 'online_order', 'book_table', 'phone', 'location', 'rest_type',
       'cuisines', 'listed_in(type)', 'listed_in(city)', 'menu_item_c'],
      dtype='object')
```

```
for cat in categorical:
    print(df_eda[cat].value_counts())
    print("*"*50)
```

```
Frazer Town              1179
Malleshwaram             1093
Rajajinagar              1069
Banashankari              859
New BEL Road              735
Name: count, dtype: int64
**************************************************
menu_item_c
not have menu    39048
have menu        12040
Name: count, dtype: int64
**************************************************
```

Note:

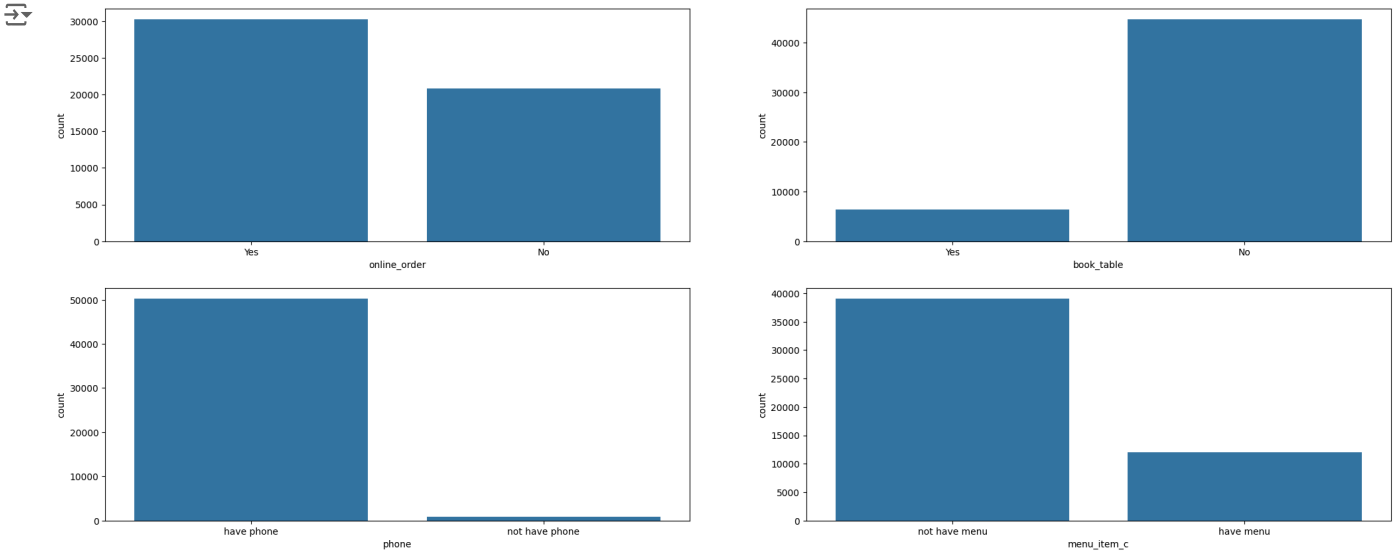
in preprocessing process

- drop name because it dosen't mean any thing in business case
- use one_Hot encoder with (online_order , book_table , phone , menu_item)
- use binary encoder with other

```
categorical
```

```
Index(['name', 'online_order', 'book_table', 'phone', 'location', 'rest_type',
       'cuisines', 'listed_in(type)', 'listed_in(city)', 'menu_item_c'],
      dtype='object')
```

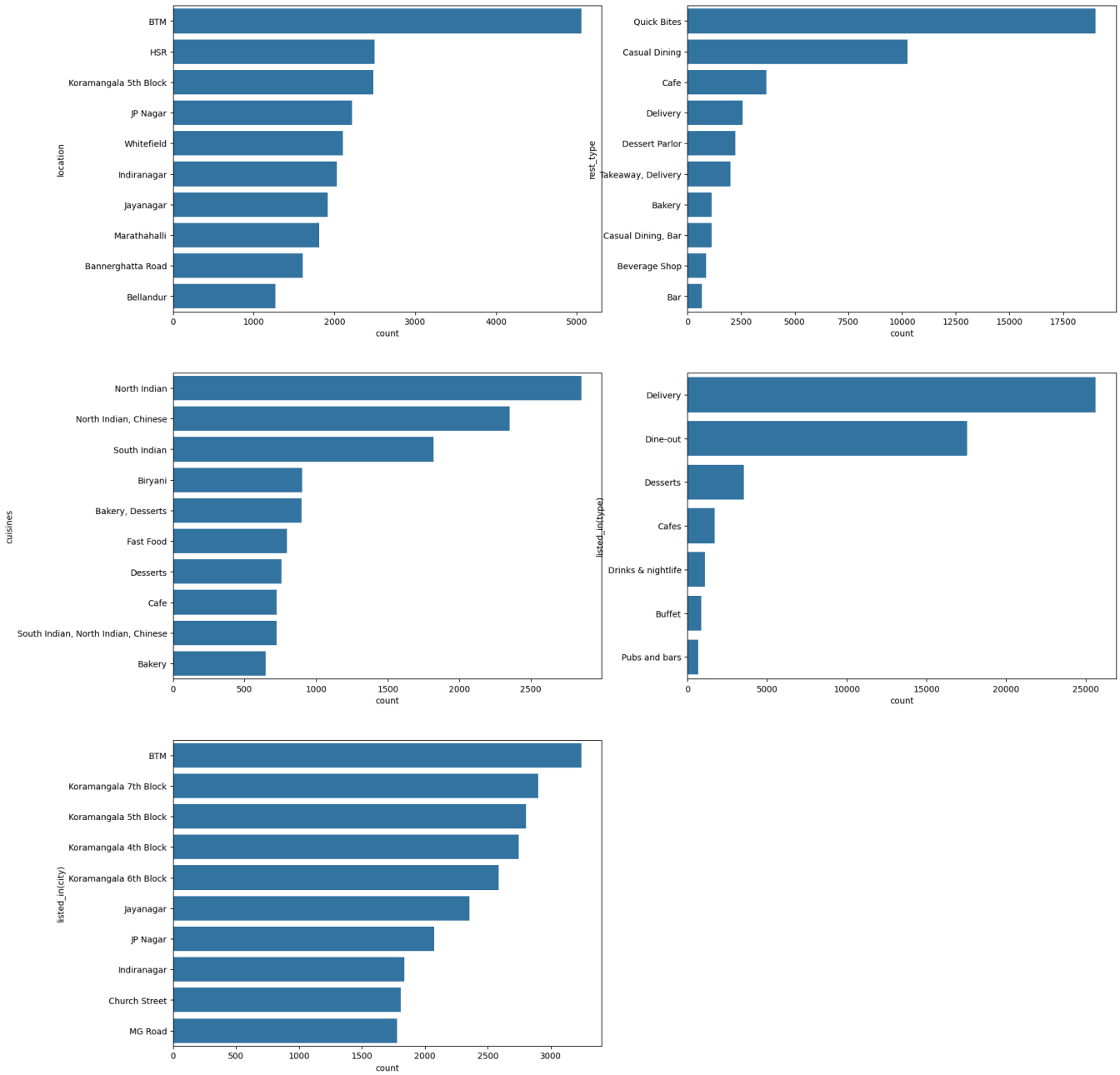
```
#subplot has columns that have yes or no
plt.figure(figsize=(25,10))
for i,cat in enumerate(["online_order","book_table","phone","menu_item_c"],1):
    plt.subplot(2,2,i)
    sns.countplot(x=df_eda[cat])
```




```
df_eda["location"].value_counts().head(10).reset_index()#test
```

	location	count
0	BTM	5056
1	HSR	2495
2	Koramangala 5th Block	2480
3	JP Nagar	2218
4	Whitefield	2107
5	Indiranagar	2033
6	Jayanagar	1916
7	Marathahalli	1808
8	Bannerghatta Road	1609
9	Bellandur	1268

```
#subplot of other  categorical columns, but get top 10 only
plt.figure(figsize=(20,30))
for i,cat in enumerate(categorical.drop(["name","online_order","book_table","phone","menu_item_c"]),1):
    plt.subplot(4,2,i)
    df_=df_eda[cat].value_counts().head(10).reset_index()
    sns.barplot(data=df_,y=cat,x="count")
```




```
numerical=df.select_dtypes(exclude="object").columns
numerical
```

```
Index(['rate', 'votes', 'approx_cost(for two people)'], dtype='object')
```

```
df_eda[numerical].describe().round(2)
```

	rate	votes	approx_cost(for two people)
count	43421.00	51088.00	51088.00
mean	3.51	285.14	556.13
std	0.92	806.90	439.65
min	0.00	0.00	40.00
25%	3.30	7.00	300.00
50%	3.70	41.00	400.00
75%	4.00	200.00	700.00
max	4.90	16832.00	6000.00

Note:

In rate there is no restaurant having rating of 5

somehow there is outliers in votes and approx_cost(for two people)

In this busniess case if restuarant rate > 3.75 will be Successful restaurant (1) , else will be Failed restaurant (0)

Double-click (or enter) to edit

```
df_eda.rate.isna().sum()
```

```
7667
```

```
df_eda[df_eda.rate.isna()]
```

	name	online_order	book_table	rate	votes	phone	location	rest_type	cuisines	approx_cost(for two people)	reviews_lis
84	Chill Out	No	No	NaN	0	have phone	Banashankari	Quick Bites	South Indian, Chinese	100.0	[('Rated 4.('RATED Goc Locatio Small
90	Me And My Cake	No	No	NaN	0	have phone	Banashankari	Delivery	Bakery, Desserts	500.0	[('Rated 1.('RATED\n D not ord anything
91	Sunsadm	No	No	NaN	0	have phone	Banashankari	Takeaway, Delivery	South Indian	400.0	
92	Annapooraneshwari Mess	No	No	NaN	0	have phone	Banashankari	Mess	South Indian	200.0	
107	Coffee Shopee	No	No	NaN	0	have phone	Banashankari	Takeaway, Delivery	Beverages	250.0	
...	...	...	...	...	...	...	...	...	...	...	
51644	Punjabi Thadka	No	No	NaN	0	have phone	Brookefield	Quick Bites	North Indian	400.0	
51675	Topsy Turvey	No	No	NaN	0	have phone	Whitefield	Bar	Finger Food	900.0	[('Rated 4.("RATED\n divine joint f Fl
51710	Topsy Turvey	No	No	NaN	0	have phone	Whitefield	Bar	Finger Food	900.0	[('Rated 4.("RATED\n divine joint f Fl
51713	Vinod Bar And Restaurant	No	No	NaN	0	have phone	Whitefield	Bar	Finger Food	600.0	
51714	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No	NaN	0	not have phone	Whitefield	Bar	Finger Food	2000.0	

7667 rows × 14 columns

```
def success(x):
       if x >3.75:
           return 1
       elif x  <= 3.75:
           return 0
       else:
           return np.nan

df_eda.rate.apply(success)
```

```
0        1.0
1        1.0
2        1.0
3        0.0
4        1.0
        ...
51712    0.0
51713    NaN
51714    NaN
51715    1.0
51716    0.0
Name: rate, Length: 51088, dtype: float64
```

```
df_eda.rate=df_eda.rate.apply(success)
```

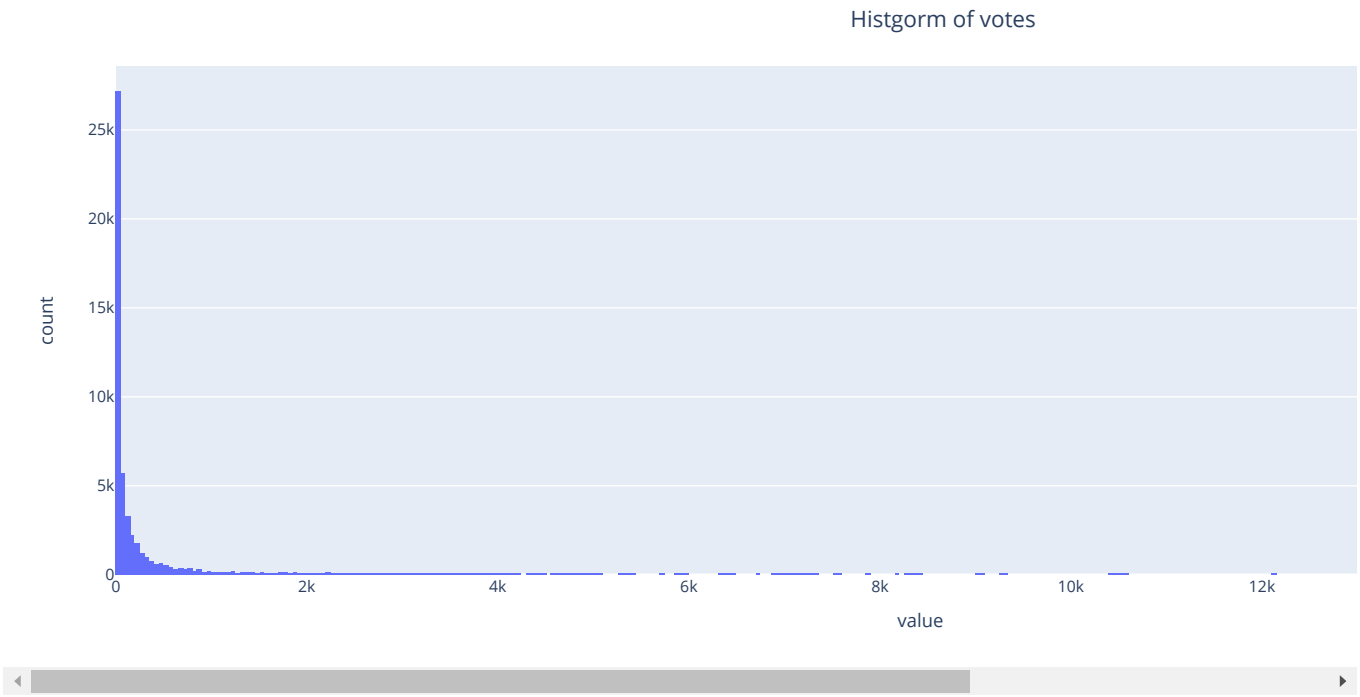
```
df_eda.rate.value_counts()
```

```
rate
0.0    23356
1.0    20065
Name: count, dtype: int64
```

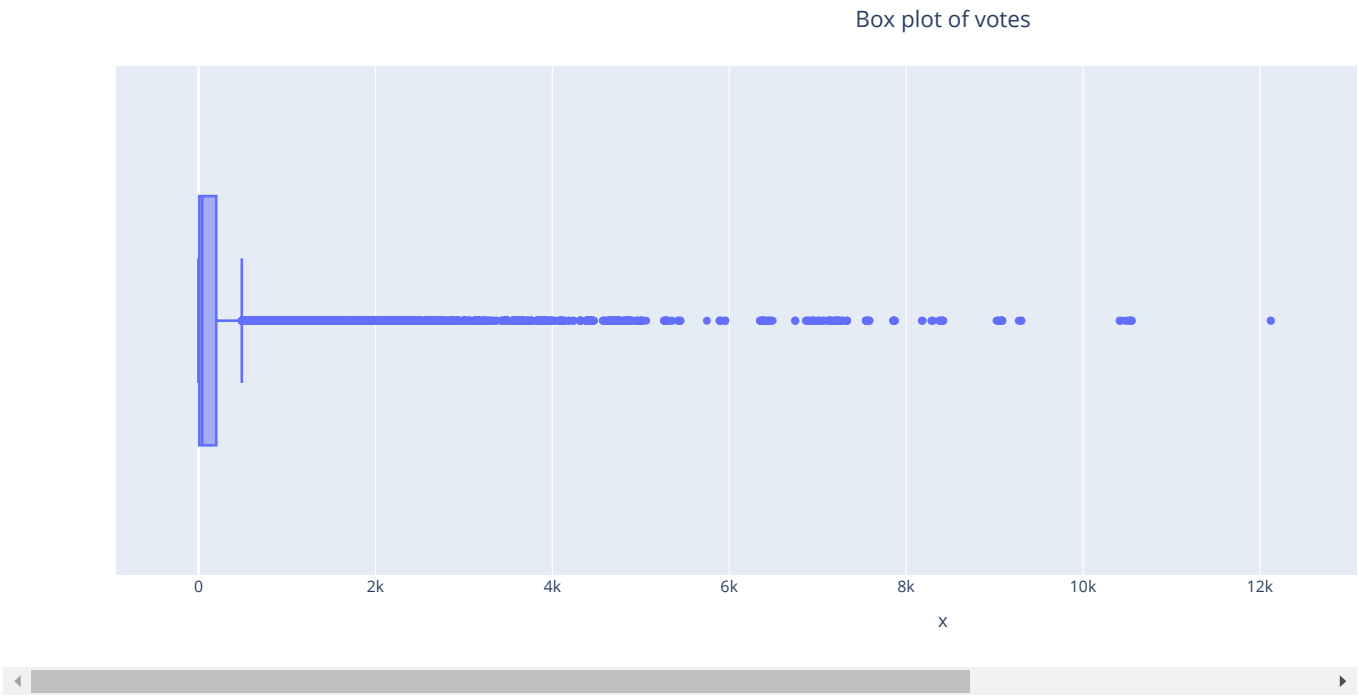
```
# check outilers in votes

df_eda.votes.quantile([0.25,0.50,0.80,0.90,0.95,0.99,0.995,0.999,0.9995,1]).to_frame().T
```

	0.2500	0.5000	0.8000	0.9000	0.9500	0.9900	0.9950	0.9990	0.9995	1.0000
votes	7.0	41.0	283.0	712.0	1382.65	3849.39	4851.78	9080.0	10548.3695	16832.0

```
fig=px.histogram(df_eda.votes)
fig.update_layout(title_text="Histgorm of votes",title_x=0.5)
```




```
fig=px.box(x=df_eda.votes)
fig.update_layout(title_text="Box plot of votes",title_x=0.5)
```




```
df_eda[df_eda.votes>=df_eda.votes.quantile(0.9995)]
```

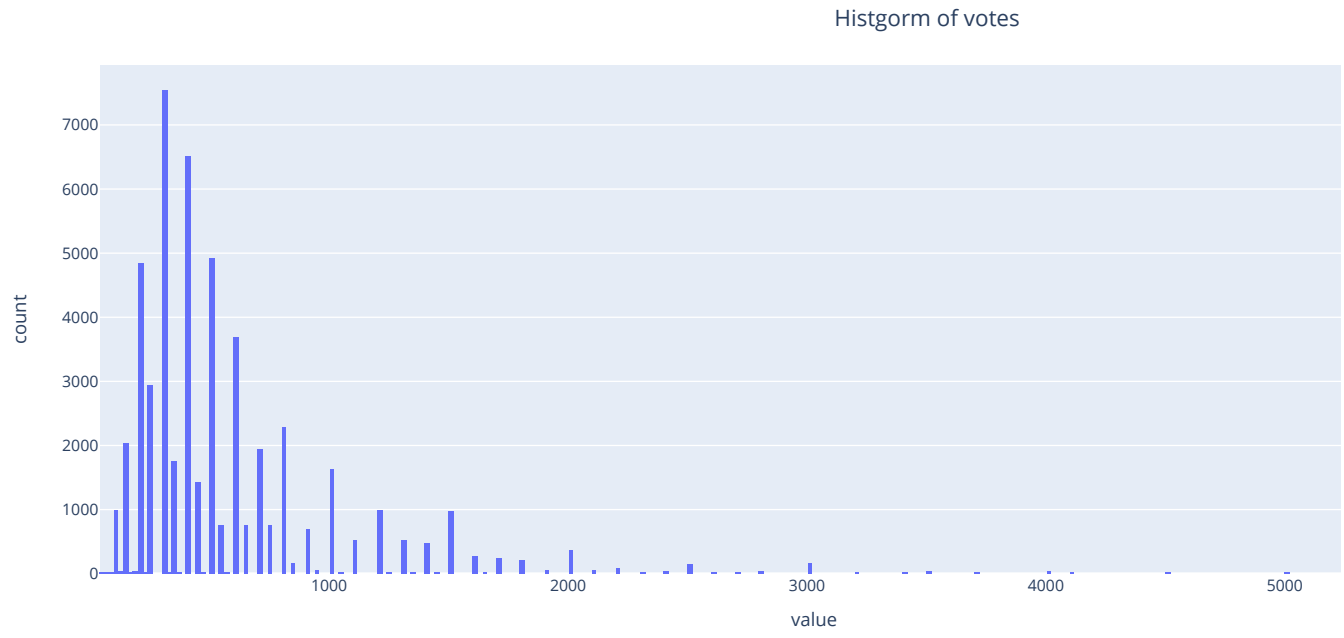
	name	online_order	book_table	rate	votes	phone	location	rest_type	cuisines	approx_cost(for two people)	reviews_li
3921	Byg Brewski Brewing Company	Yes	Yes	1.0	16345	have phone	Sarjapur Road	Microbrewery	Continental, North Indian, Italian, South Indi...	1600.0	[('Rated 5. 'RATED\ have been this p
4801	Byg Brewski Brewing Company	Yes	Yes	1.0	16345	have phone	Sarjapur Road	Microbrewery	Continental, North Indian, Italian, South Indi...	1600.0	[('Rated 5. 'RATED\ have been this p
4944	Byg Brewski Brewing Company	Yes	Yes	1.0	16345	have phone	Sarjapur Road	Microbrewery	Continental, North Indian, Italian, South Indi...	1600.0	[('Rated 5. 'RATED\ have been this p
8330	Truffles	No	No	1.0	14654	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 5. 'RATED Menu extensi and
9807	Truffles	No	No	1.0	14654	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 5. 'RATED Menu extensi and
10860	Truffles	No	No	1.0	14654	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 5. 'RATED Menu extensi and
18643	Toit	No	No	1.0	14956	have phone	Indiranagar	Microbrewery	Italian, American, Pizza	1500.0	[('Rated 4. "RATED\n T crowd kee flow
19268	Toit	No	No	1.0	14956	have phone	Indiranagar	Microbrewery	Italian, American, Pizza	1500.0	[('Rated 4. "RATED\n T crowd kee flow
26549	Truffles	No	No	1.0	14690	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 4. 'RATED Packag 4/5\nFood: 4
27806	Truffles	No	No	1.0	14694	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 4. 'RATED Packag 4/5\nFood: 4
28384	Truffles	No	No	1.0	14704	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 2. "RATED Overhype Deserts a
29303	Truffles	No	No	1.0	14710	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 2. "RATED Overhype Deserts a
29414	Truffles	No	No	1.0	14710	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 2. "RATED Overhype Deserts a
31055	Truffles	No	No	1.0	14710	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 2. "RATED Overhype Deserts a
32152	Truffles	No	No	1.0	14717	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 2. "RATED Overhype Deserts a
33312	Truffles	No	No	1.0	14717	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 2. "RATED Overhype Deserts a
33913	Truffles	No	No	1.0	14723	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 3. "RATED\n N bad but t burge
34779	Truffles	No	No	1.0	14723	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 3. "RATED\n N bad but t

									Burger, Steak		burge
36000	Truffles	No	No	1.0	14723	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 3. "RATED\n N bad but t burge
36668	Truffles	No	No	1.0	14726	have phone	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	900.0	[('Rated 3. "RATED\n N bad but t burge
37606	The Black Pearl	No	Yes	1.0	10550	have phone	Koramangala 5th Block	Casual Dining, Bar	North Indian, European, Mediterranean	1400.0	[('Rated 5. 'RATED Amazi experien ar
40506	AB's - Absolute Barbecues	No	Yes	1.0	12121	have phone	Marathahalli	Casual Dining	European, Mediterranean, North Indian, BBQ	1600.0	[('Rated 4. 'RATED\n V came here a te
41525	AB's - Absolute Barbecues	No	Yes	1.0	12121	have phone	Marathahalli	Casual Dining	European, Mediterranean, North Indian, BBQ	1600.0	[('Rated 4. 'RATED\n V came here a te
49170	Byg Brewski Brewing Company	Yes	Yes	1.0	16832	have phone	Sarjapur Road	Microbrewery	Continental, North Indian, Italian, South Indi...	1600.0	[('Rated 5. "RATED This is absolute g
49627	Byg Brewski Brewing Company	Yes	Yes	1.0	16832	have phone	Sarjapur Road	Microbrewery	Continental, North Indian, Italian, South Indi...	1600.0	[('Rated 4. 'RATED Visiting microbrewe
50059	Byg Brewski Brewing Company	Yes	Yes	1.0	16832	have phone	Sarjapur Road	Microbrewery	Continental, North Indian, Italian, South Indi...	1600.0	[('Rated 4. 'RATED Visiting microbrewe

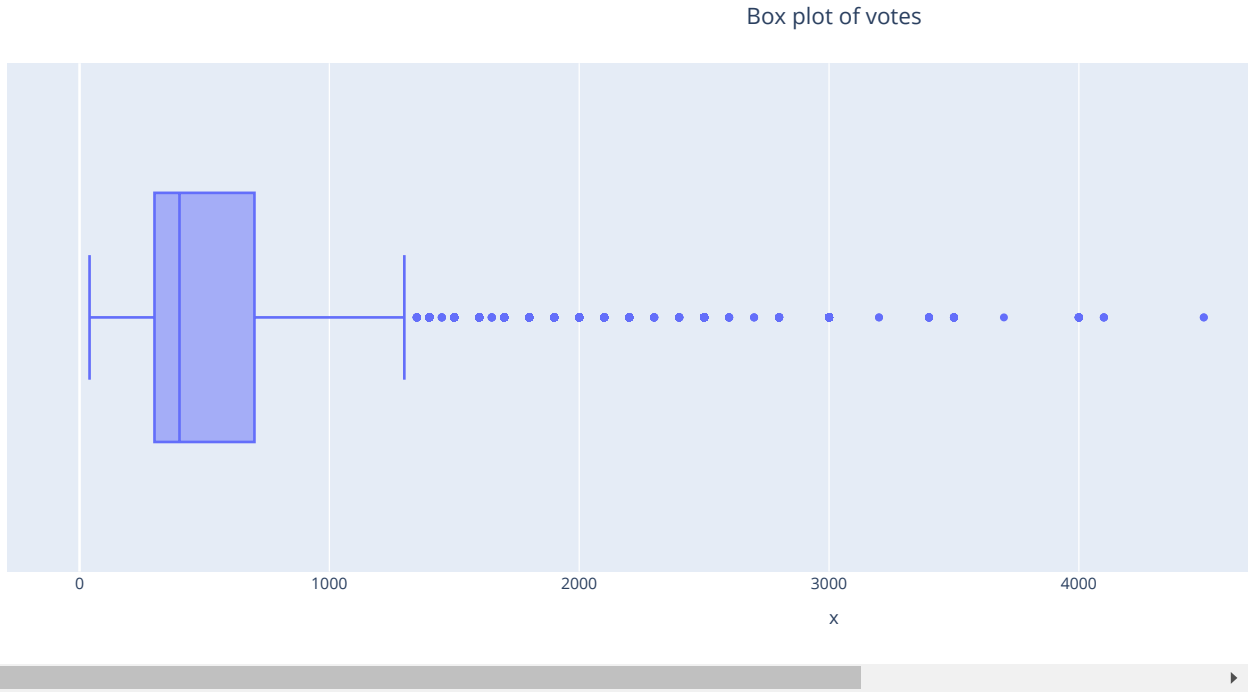
```
# check outilers in votes

df_eda["approx_cost(for two people)"].quantile([0.25,0.50,0.80,0.90,0.95,0.99,0.995,0.999,0.9995,1]).to_frame().T
```

	0.2500	0.5000	0.8000	0.9000	0.9500	0.9900	0.9950	0.9990	0.9995	1.0000
approx_cost(for two people)	300.0	400.0	750.0	1100.0	1500.0	2200.0	2800.0	3500.0	4000.0	6000.0

```
fig=px.histogram(df_eda["approx_cost(for two people)"])
fig.update_layout(title_text="Histgorm of votes",title_x=0.5)
```



```
fig=px.box(x=df_eda["approx_cost(for two people)"])
fig.update_layout(title_text="Box plot of votes",title_x=0.5)
```




```
Start coding or generate with AI.
```

yes. there are outiers , but its normal so dont drop any value but use Robust scale in preprocessing

Conclusion from Unviarate Analysis

in preprocessing :

- drop name because it doesn't matter in business case
- use one_Hot encoder with (online_order , book_table , phone , menu_item)
- use binary encoder with other
- drop all value that have quantitle > 0.9995 in votes
- use Robust scale with (votes , approx_cost(for two people))

Bivarate


```
df_eda.head()
```

	name	online_order	book_table	rate	votes	phone	location	rest_type	cuisines	approx_cost(for two people)	reviews_list	listed_in(
0	Jalsa	Yes	Yes	1.0	775	have phone	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n A beautiful place to ...	
1	Spice Elephant	Yes	No	1.0	787	have phone	Banashankari	Casual Dining	Chinese, North Indian, Thai	800.0	[('Rated 4.0', 'RATED\n Had been here for din...	
2	San Churro Cafe	Yes	No	1.0	918	have phone	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	[('Rated 3.0', "RATED\n Ambience is not that ...	
3	Addhuri Udupi Bhojana	No	No	0.0	88	have phone	Banashankari	Quick Bites	South Indian, North Indian	300.0	[('Rated 4.0', "RATED\n Great food and proper...	
4	Grand					have		Casual	North		[('Rated 4.0', 'RATED\n	

Next steps: Generate code with df_eda View recommended plots

```
#convert datatype of rate to object to make easy in Visualization
df_eda.rate=df_eda.rate.astype("O")
```

```
df_eda.rate.dtype
```

```
dtype('O')
```

```
# Visualization Function
def bar(data_frame,x,y,title_text,color=None):
    fig=px.bar(data_frame=data_frame,x=x,y=y,color=color, barmode='group',text_auto="0.2s")
    fig.update_traces(textfont_size=12,textposition="outside")
    fig.update_layout(title_text=title_text,title_x=0.5)
    return fig
```

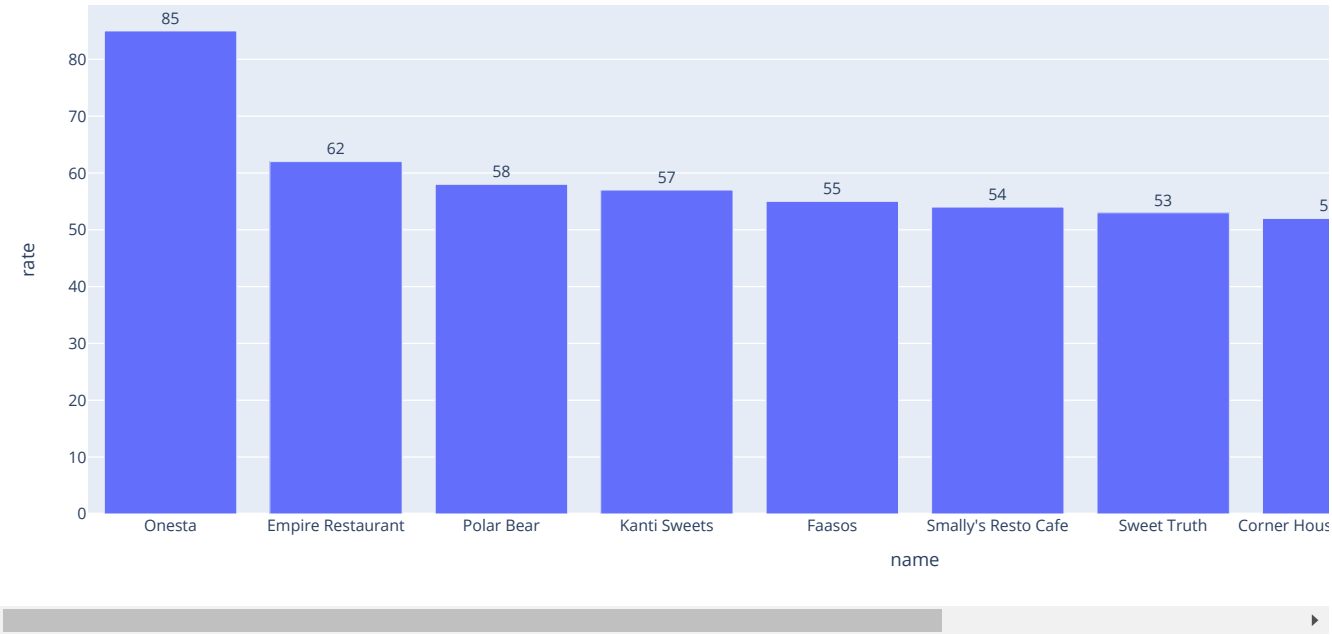
Top 10 restaurant successful (rate=1) (name,rate)

```
#pandas
data=df_eda[df_eda.rate==1].groupby("name")["rate"].count().reset_index().sort_values(by="rate",ascending=False).head(10)
data
```

	name	rate
1712	Onesta	85
787	Empire Restaurant	62
1804	Polar Bear	58
1216	Kanti Sweets	57
805	Faasos	55
2123	Smally's Resto Cafe	54
2261	Sweet Truth	53
615	Corner House Ice Cream	52
377	Burger King	46
2762	eat.fit	46

Next steps: Generate code with data View recommended plots

```
bar(data,"name","rate","Top 10 restaurant successful (rate=1)")
```



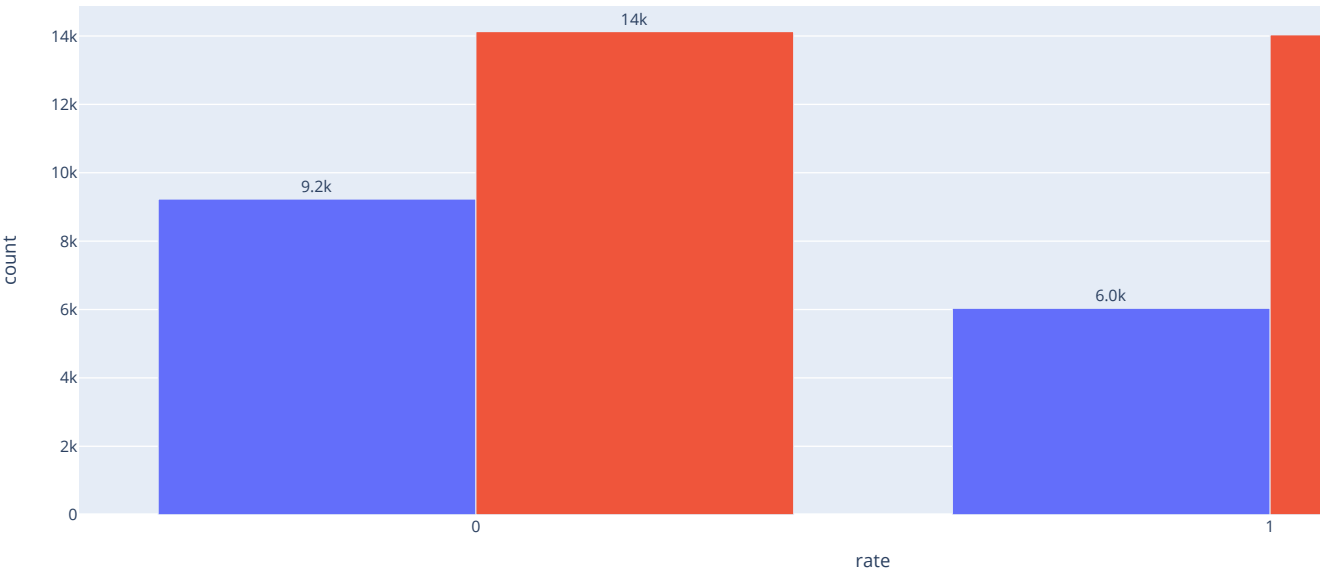
Relation between online_order and rate

```
#pandas
data=df_eda.groupby(["rate","online_order"]).agg({"online_order":"count"}).rename(columns={"online_order":"count"}).reset_index()
data
```

	rate	online_order	count
0	0.0	No	9229
1	0.0	Yes	14127
2	1.0	No	6033
3	1.0	Yes	14032

Next steps: Generate code with data View recommended plots

```
bar(data,"rate","count",color="online_order",title_text="Relation between online_order and rate")
```



- To be sucessful restaurant one should offer online_order service

---

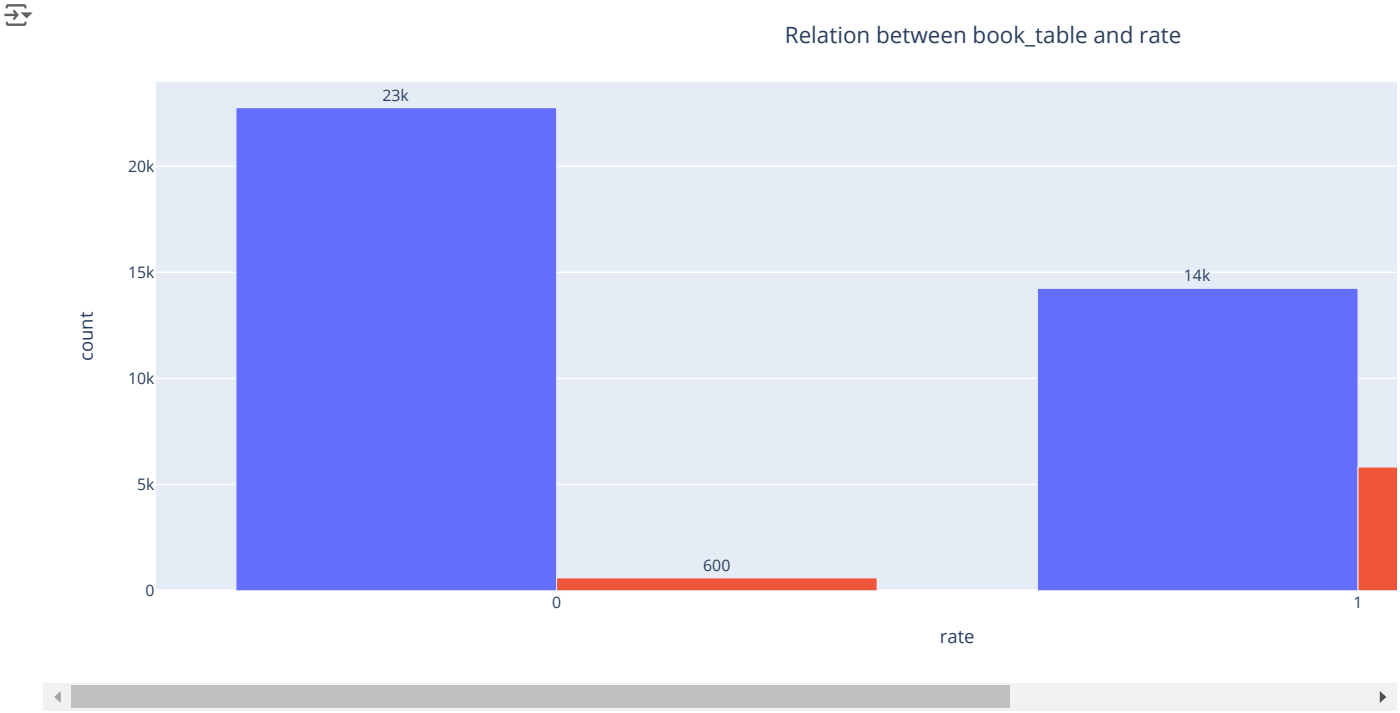
Relation between book_table and rate

```
#pandas
data=df_eda.groupby(["rate","book_table"]).agg({"book_table":"count"}).rename(columns={"book_table":"count"}).reset_index()
data
```

	rate	book_table	count
0	0.0	No	22755
1	0.0	Yes	601
2	1.0	No	14243
3	1.0	Yes	5822

Next steps: Generate code with data View recommended plots

```
bar(data,"rate","count",color="book_table",title_text="Relation between book_table and rate")
```



- There is no relation between book_table and rate , that mean you can be a successful restaurant without book table service is available

---

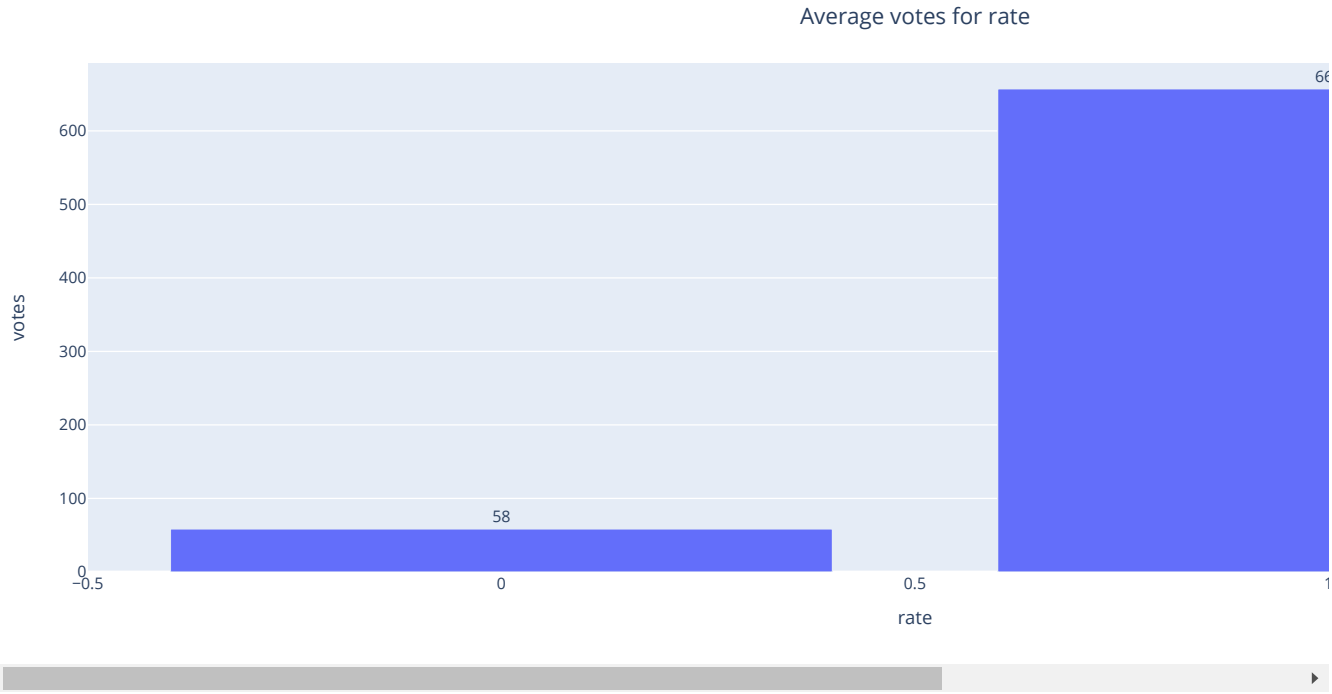
Average votes for rate

```
data=df_eda.groupby("rate")["votes"].mean().reset_index()
data
```

	rate	votes
0	0.0	58.324114
1	1.0	657.110441

Next steps: Generate code with data View recommended plots

```
bar(data,"rate","votes","Average votes for rate")
```



- Most successful restaurant have more votes

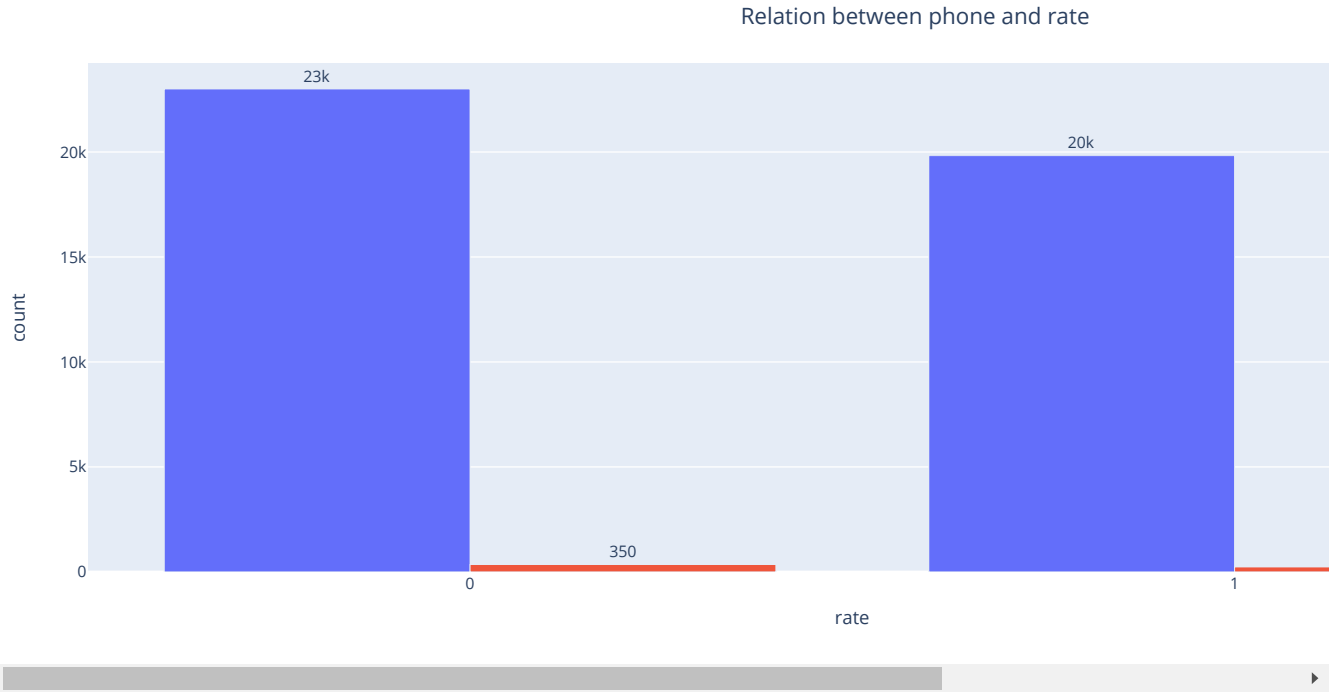
Relation between phone and rate

```
#pandas
data=df_eda.groupby(["rate","phone"]).agg({"phone":"count"}).rename(columns={"phone":"count"}).reset_index()
data
```

	rate	phone	count
0	0.0	have phone	23004
1	0.0	not have phone	352
2	1.0	have phone	19828
3	1.0	not have phone	237

Next steps: Generate code with data | View recommended plots

```
bar(data,"rate","count",color="phone",title_text="Relation between phone and rate")
```



- To open any resturant must have phone number

Top 10 locations that have a successful restaurant and Competition is high((rate=1)

```
#pandas
data=df_eda[df_eda.rate==1].groupby("location")["rate"].count().reset_index().sort_values(by="rate",ascending=False).head(10)
data
```

	location	rate
43	Koramangala 5th Block	1766
0	BTM	1207
26	Indiranagar	1183
21	HSR	975
30	Jayanagar	955
28	JP Nagar	830
45	Koramangala 7th Block	689
82	Whitefield	628
44	Koramangala 6th Block	614
42	Koramangala 4th Block	613

Next steps: Generate code with data View recommended plots

```
bar(data,"location","rate",title_text="Top 10 locations that have a successful restaurant and Competition is high((rate=1)")
```

- Koramangala 5th Block, BTM and Indiranagar have successful restaurant and Competition is high.

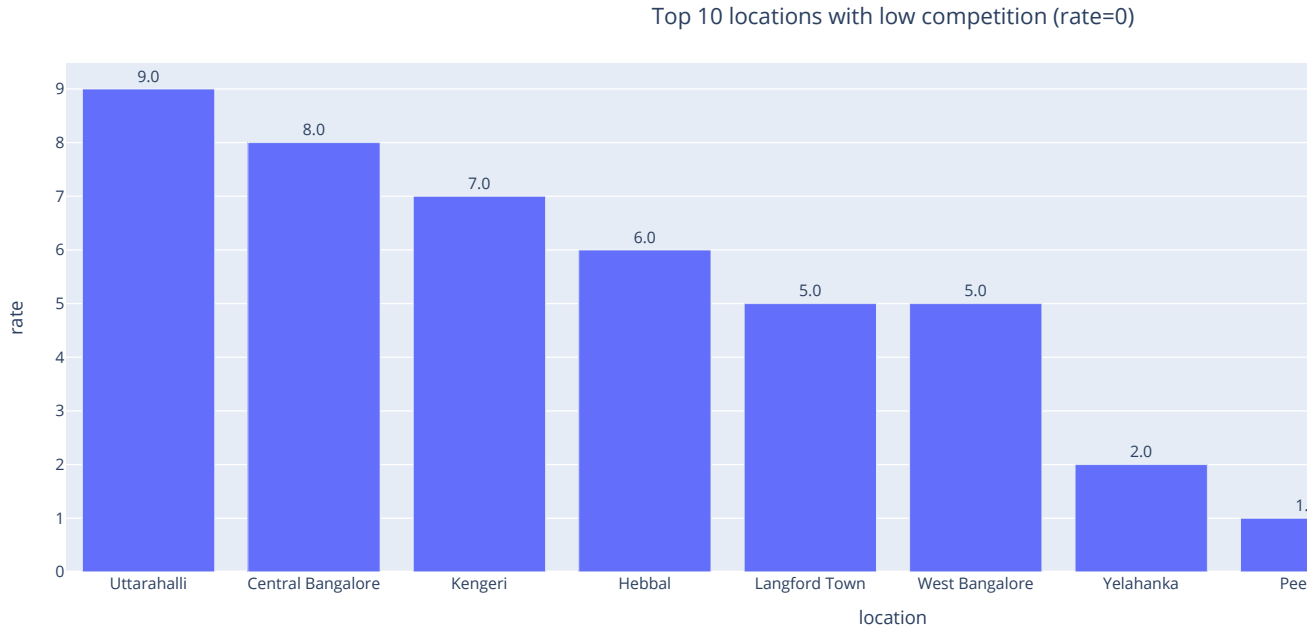
Top 10 locations with low competition (rate=0)

```
#pandas
data=df_eda[df_eda.rate==0].groupby("location")["rate"].count().reset_index().sort_values(by="rate",ascending=False).tail(10)
data
```

	location	rate
83	Uttarahalli	9
11	Central Bangalore	8
38	Kengeri	7
23	Hebbal	6
49	Langford Town	5
87	West Bangalore	5
90	Yelahanka	2
63	Peenya	1
57	Nagarbhavi	1
67	Rajarajeshwari Nagar	1

Next steps: Generate code with data View recommended plots

```
bar(data,"location","rate",title_text="Top 10 locations with low competition (rate=0)")
```



- in Rajarajeshwari Nagar, Peenya and Nagarabhavi competition is low
- **opinion**: you can open resturant in Rajarajeshwari Nagar, Peenya or Nagarabhavi that have lower competition make good reputation after that open in Koramangala 5th Block or BTM that have higher competition.

Relation between list_in(city) and rate

```
#pandas
data=df_eda.groupby(["rate","listed_in(city)"]).agg({"listed_in(city)":"count"}).rename(columns={"listed_in(city)":"count"}).reset_index(
data
```

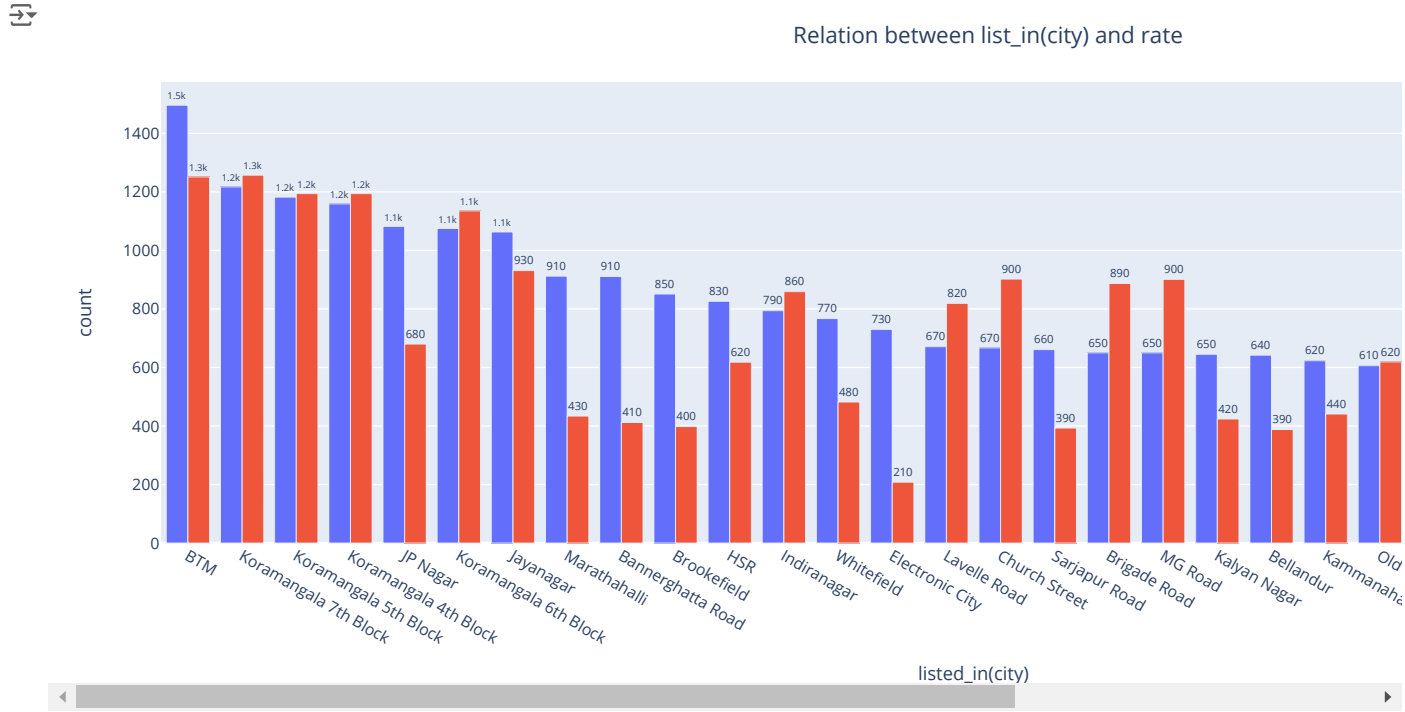
	rate	listed_in(city)	count
0	0.0	BTM	1496
49	1.0	Koramangala 7th Block	1257
30	1.0	BTM	1251
19	0.0	Koramangala 7th Block	1217
47	1.0	Koramangala 5th Block	1194
46	1.0	Koramangala 4th Block	1194
17	0.0	Koramangala 5th Block	1182
16	0.0	Koramangala 4th Block	1159
48	1.0	Koramangala 6th Block	1135
12	0.0	JP Nagar	1082
18	0.0	Koramangala 6th Block	1075
13	0.0	Jayanagar	1063
43	1.0	Jayanagar	931
23	0.0	Marathahalli	912
2	0.0	Bannerghatta Road	911
37	1.0	Church Street	902
51	1.0	MG Road	901
35	1.0	Brigade Road	887
41	1.0	Indiranagar	859
6	0.0	Brookefield	851
10	0.0	HSR	826
50	1.0	Lavelle Road	819
11	0.0	Indiranagar	794
57	1.0	Residency Road	775
29	0.0	Whitefield	767
8	0.0	Electronic City	730
42	1.0	JP Nagar	680
20	0.0	Lavelle Road	671
7	0.0	Church Street	667
28	0.0	Sarjapur Road	661
5	0.0	Brigade Road	650
21	0.0	MG Road	650
14	0.0	Kalyan Nagar	645
4	0.0	Bellandur	642
15	0.0	Kammanahalli	623
55	1.0	Old Airport Road	619
40	1.0	HSR	618
25	0.0	Old Airport Road	606
27	0.0	Residency Road	602
3	0.0	Basavanagudi	576
33	1.0	Basavanagudi	549
26	0.0	Rajajinagar	514
9	0.0	Frazer Town	502
22	0.0	Malleshwaram	502
39	1.0	Frazer Town	491
59	1.0	Whitefield	482
52	1.0	Malleshwaram	472
45	1.0	Kammanahalli	441
53	1.0	Marathahalli	434

44	1.0	Kalyan Nagar	424
1	0.0	Banashankari	424
32	1.0	Bannerghatta Road	412
36	1.0	Brookefield	398
58	1.0	Sarjapur Road	393
34	1.0	Bellandur	388
56	1.0	Rajajinagar	381
24	0.0	New BEL Road	356
31	1.0	Banashankari	342
54	1.0	New BEL Road	228
38	1.0	Electronic City	208

Next steps: Generate code with `data` View recommended plots

```
data.rate=data.rate.astype("O")
```

```
bar(data,"listed_in(city)","count",color="rate",title_text="Relation between list_in(city) and rate")
```



- **Conclusion**:

BTM , Koramangala 7th Block and Koramangala 5th Block, have the highest percentage of restaurant successful or failed , so competition is very high

---

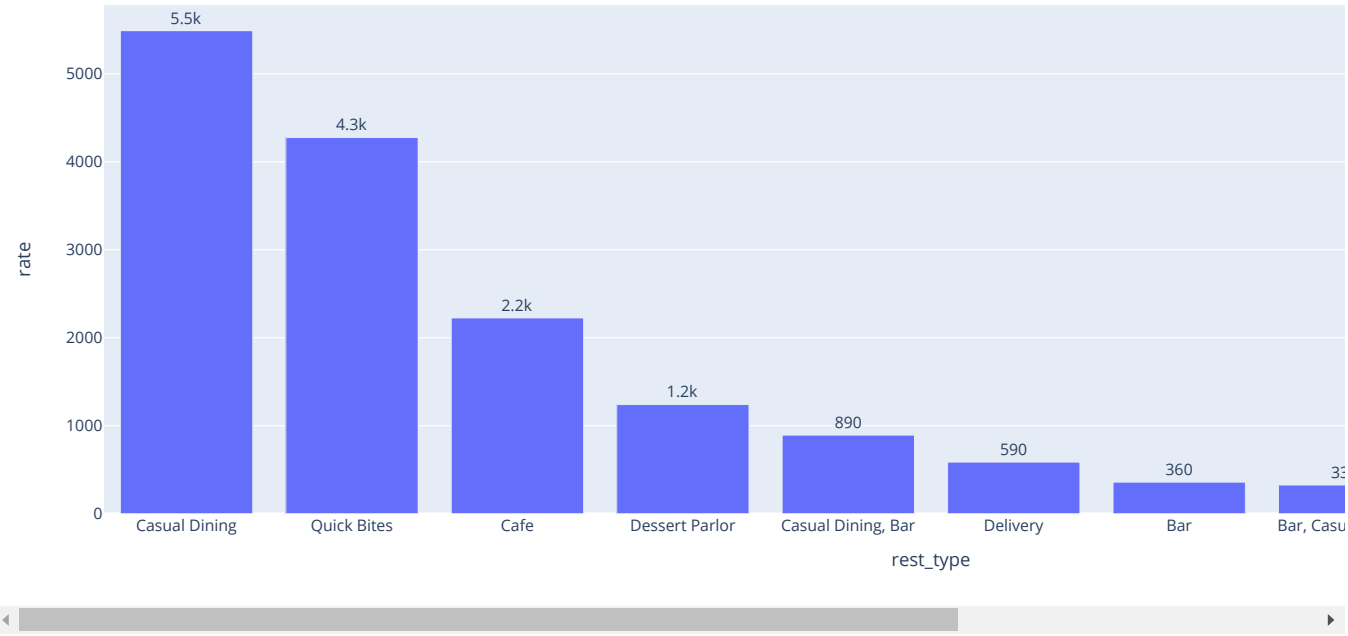
Top 10 rest_type has good reputation (rate=1)

```
#pandas
data=df_eda[df_eda.rate==1].groupby("rest_type")["rate"].count().reset_index().sort_values(by="rate",ascending=False).head(10)
data
```

	rest_type	rate
21	Casual Dining	5490
62	Quick Bites	4275
13	Cafe	2225
32	Dessert Parlor	1240
22	Casual Dining, Bar	893
31	Delivery	586
4	Bar	358
5	Bar, Casual Dining	326
37	Fine Dining	313
71	Takeaway, Delivery	310

Next steps: Generate code with data View recommended plots

```
bar(data,"rest_type","rate",title_text="Top 10 rest_type has good reputation (rate=1)")
```



- To be successful restaurant you have to rest type will be Casual Dining or Quick Bites

---

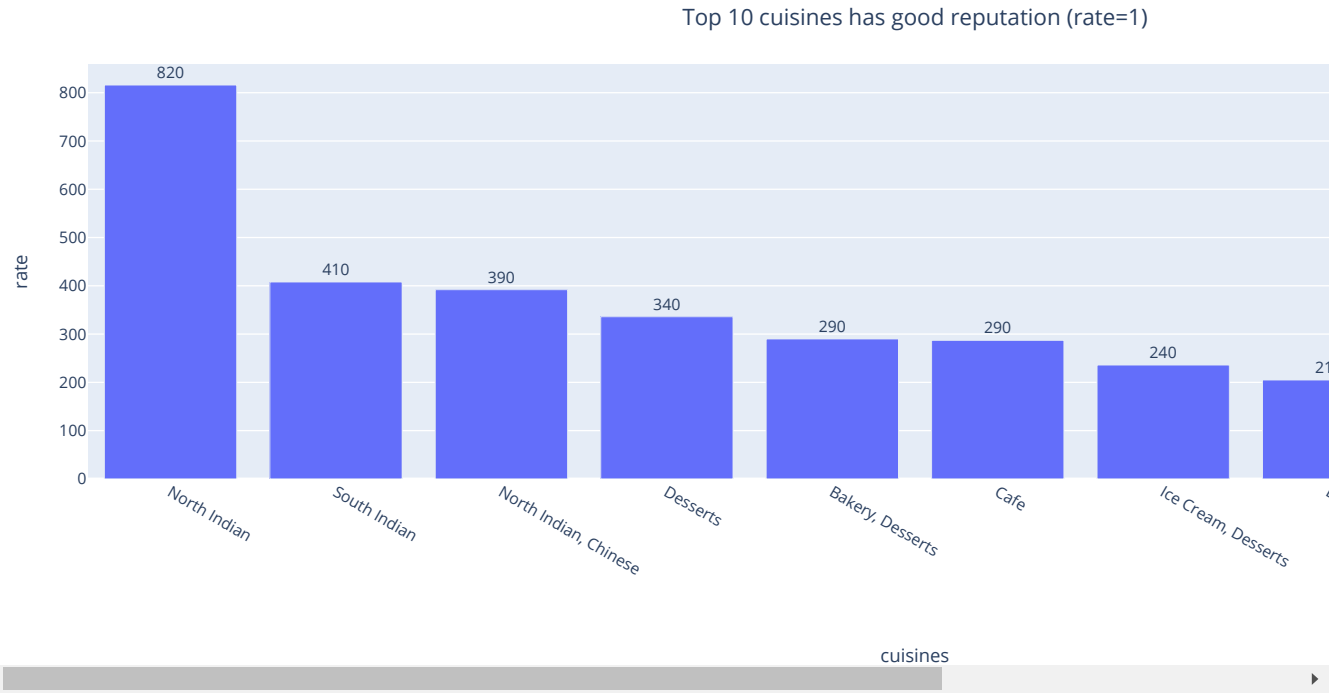
Top 10 cuisines has good reputation (rate=1)

```
#pandas
data=df_eda[df_eda.rate==1].groupby("cuisines")["rate"].count().reset_index().sort_values(by="rate",ascending=False).head(10)
data
```

	cuisines	rate
1047	North Indian	816
1379	South Indian	408
1080	North Indian, Chinese	392
681	Desserts	336
156	Bakery, Desserts	290
285	Cafe	287
859	Ice Cream, Desserts	236
687	Desserts, Beverages	205
1421	South Indian, North Indian, Chinese	190
1017	Mithai, Street Food	168

Next steps: Generate code with data View recommended plots

```
bar(data,"cuisines","rate",title_text="Top 10 cuisines has good reputation (rate=1)")
```



- To be successful restaurant you have to cuisines type will be North Indian or Chinese or South Indian

---

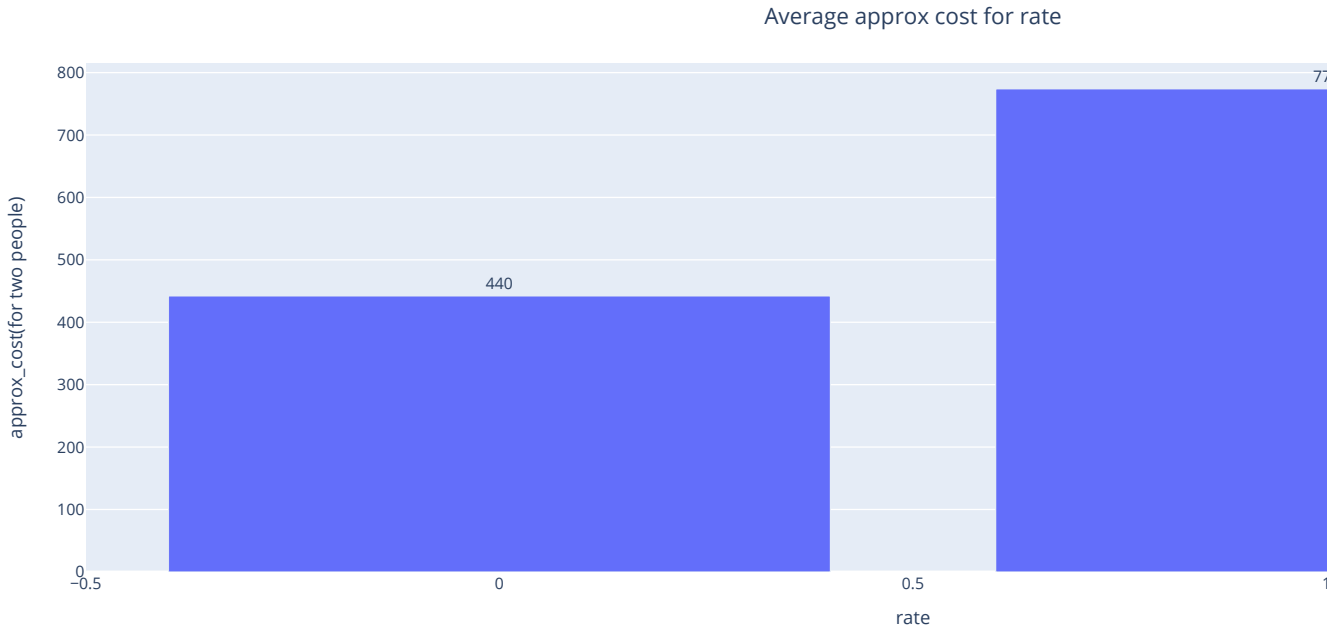
Average approx cost for rate

```
data=df_eda.groupby("rate")["approx_cost(for two people)"].mean().reset_index()
data
```

	rate	approx_cost(for two people)
0	0.0	442.067392
1	1.0	773.866434

Next steps: Generate code with data View recommended plots

```
bar(data,"rate","approx_cost(for two people)","Average approx cost for rate")
```



Relation between listed_in(type) and rate

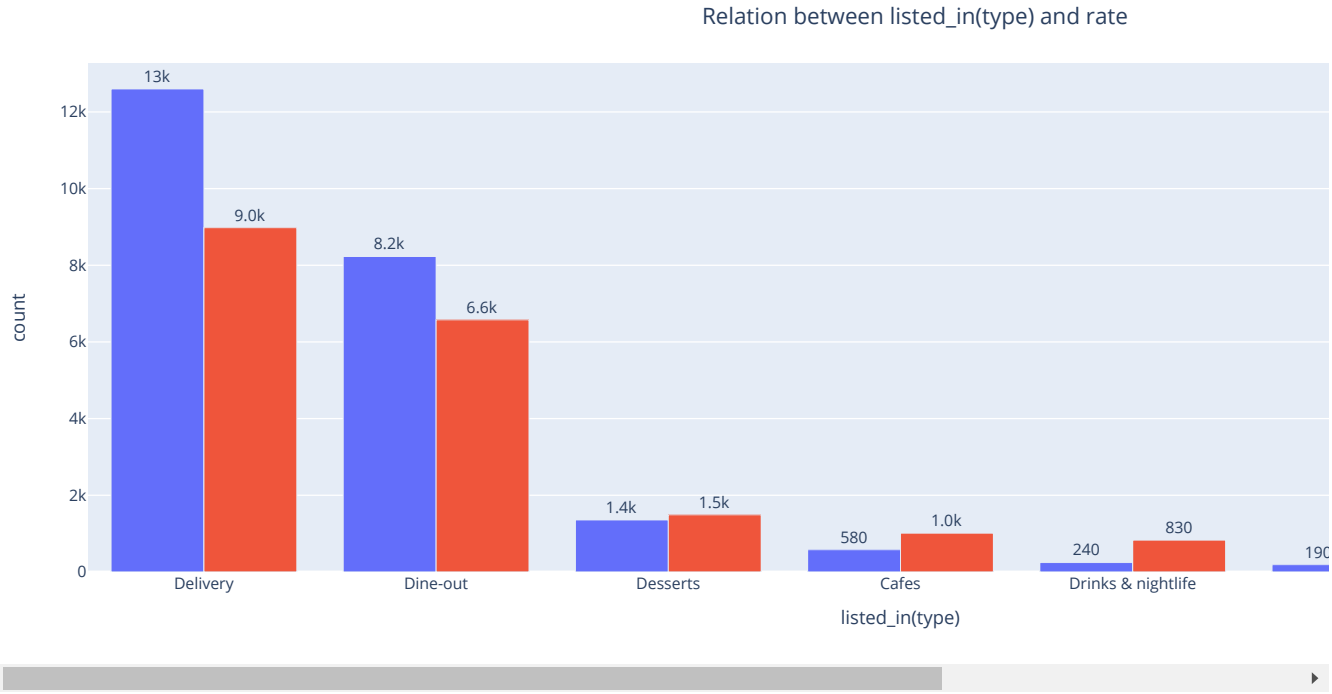
```
#pandas
data=df_eda.groupby(["rate","listed_in(type)"]).agg({"listed_in(type)":"count"}).rename(columns={"listed_in(type)":"count"}).reset_index(

data
```

	rate	listed_in(type)	count
2	0.0	Delivery	12601
9	1.0	Delivery	8981
4	0.0	Dine-out	8229
11	1.0	Dine-out	6577
10	1.0	Desserts	1490
3	0.0	Desserts	1356
8	1.0	Cafes	1010
12	1.0	Drinks & nightlife	829
7	1.0	Buffet	676
1	0.0	Cafes	579
13	1.0	Pubs and bars	502
5	0.0	Drinks & nightlife	244
0	0.0	Buffet	189
6	0.0	Pubs and bars	158

Next steps: Generate code with data View recommended plots

```
data.rate=data.rate.astype("O")
```

```
bar(data,"listed_in(type)","count",color="rate",title_text="Relation between listed_in(type) and rate")
```



The type of service provided by the most resturants either successful or failed is Delivery or Dine-out.

- To be successful restaurant one has to provide service of Delivery or Dine-out

---

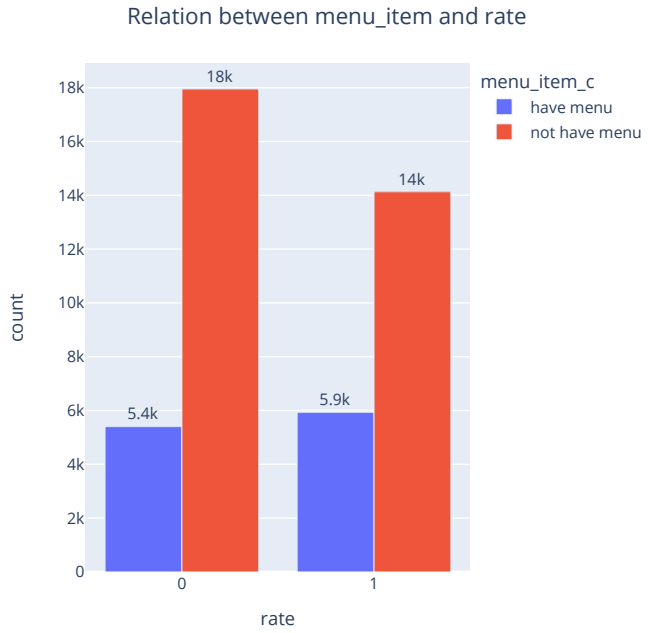
Relation between menu_item and rate

```
#pandas
data=df_eda.groupby(["rate","menu_item_c"]).agg({"menu_item_c":"count"}).rename(columns={"menu_item_c":"count"}).reset_index()
data
```

	rate	menu_item_c	count
0	0.0	have menu	5401
1	0.0	not have menu	17955
2	1.0	have menu	5932
3	1.0	not have menu	14133

Next steps: Generate code with data | View recommended plots

```
bar(data,"rate","count",color="menu_item_c",title_text="Relation between menu_item and rate")
```



- There is no relation between successful or failed restaurant have menu or not , but for restaurant to be successful perfer put menu.

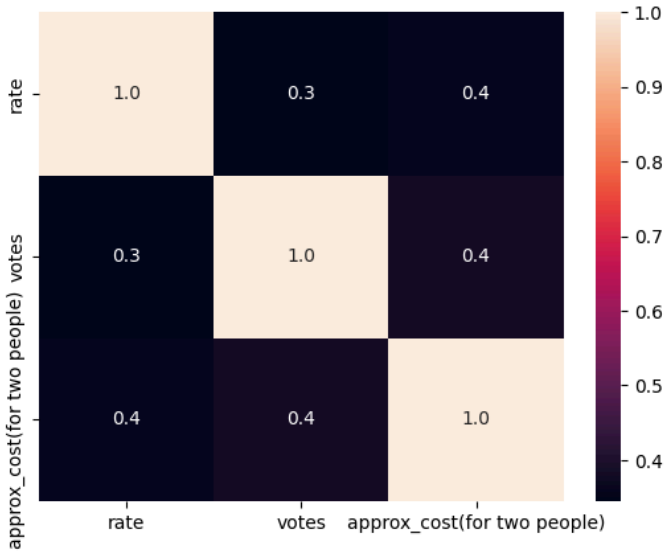
Correlation

```
df_eda[numerical].corr()
```

	rate	votes	approx_cost(for two people)
rate	1.000000	0.344999	0.359934
votes	0.344999	1.000000	0.381649
approx_cost(for two people)	0.359934	0.381649	1.000000

```
sns.heatmap(df_eda[numerical].corr(),annot=True, fmt=".1f")
```

```
<Axes: >
```



- There is positive relation between rate and (votes , approx_cost(for two people))

Conclusion for Bivarate :

Some advice to be a successful restaurant :

1) online_order service is available

2) must have phone number

3) open your resturant in Mysore Road or Hebbal that have lower competition make good reputation after that open in Koramangala 5th Block or BTM that have higher competition.

4) Must have restaurant type will be Casual Dining or Quick Bites

5) Must have cuisines type will be North Indian or Chinese or South Indian

6) prefer put menu in website

7) service provided should be Delivery or Dine-out

Multivarate

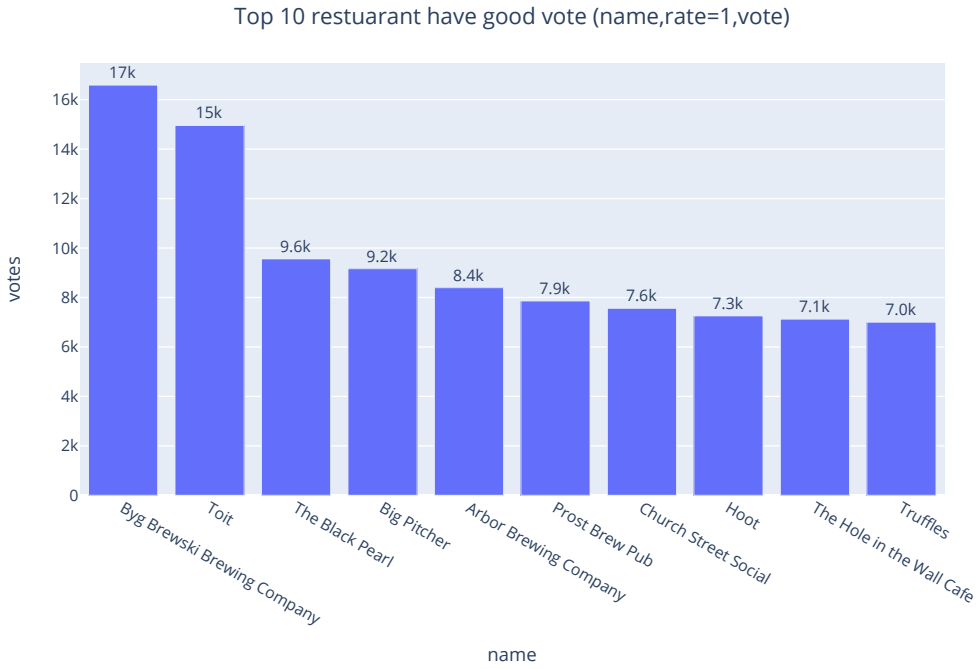
Top 10 restuarant have good vote(name,rate=1,vote)

```
data=df_eda[df_eda.rate==1].groupby("name")["votes"].mean().reset_index().sort_values(by="votes",ascending=False).head(10)
data
```

	name	votes
386	Byg Brewski Brewing Company	16588.500000
2569	Toit	14956.000000
2357	The Black Pearl	9562.333333
283	Big Pitcher	9164.500000
131	Arbor Brewing Company	8396.545455
1825	Prost Brew Pub	7860.900000
576	Church Street Social	7561.727273
1022	Hoot	7257.000000
2419	The Hole in the Wall Cafe	7124.875000
2578	Truffles	7001.372093

Next steps: Generate code with data View recommended plots

```
bar(data,"name",'votes',"Top 10 restuarant have good vote (name,rate=1,vote)")
```



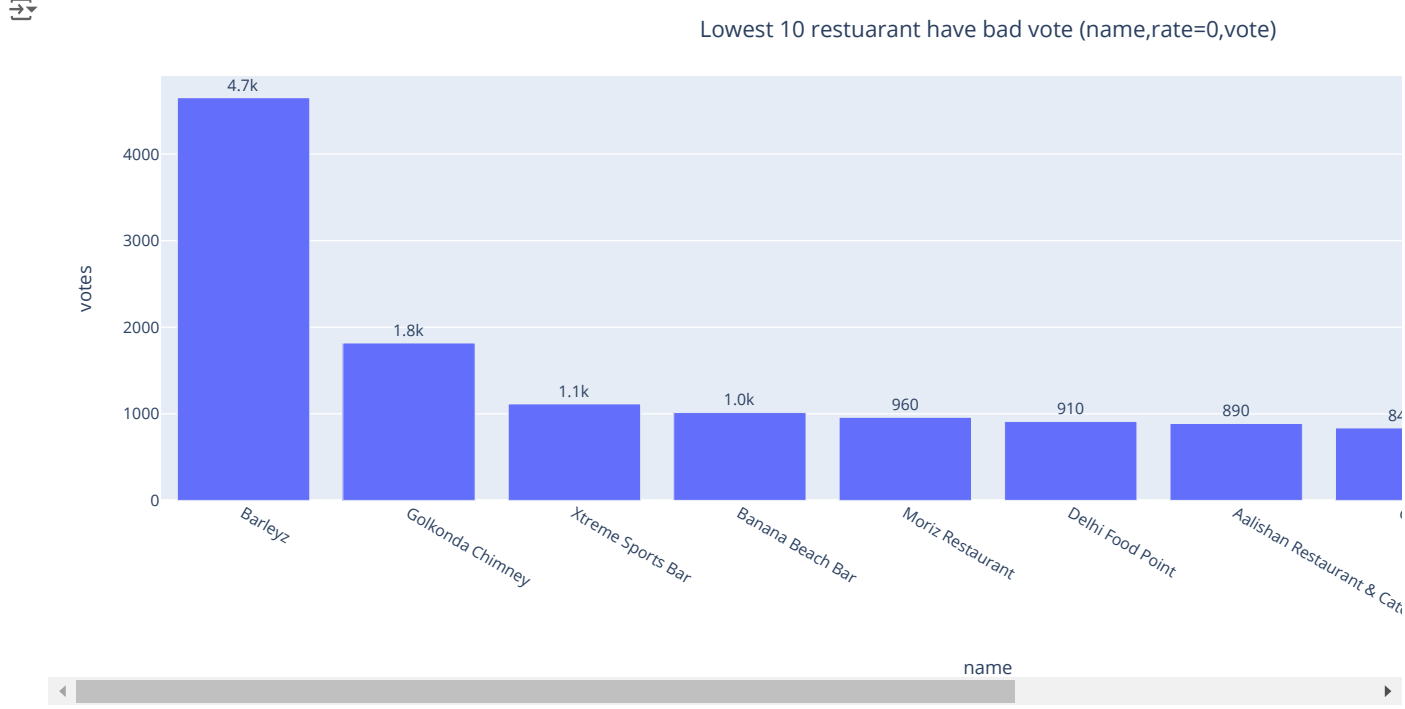
Lowest 10 restuarant have bad vote(name,rate=0,vote)

```
data=df_eda[df_eda.rate==0].groupby("name")["votes"].mean().reset_index().sort_values(by="votes",ascending=False).head(10)
data
```

	name	votes
464	Barleyz	4651.750000
1628	Golkonda Chimney	1818.500000
4803	Xtreme Sports Bar	1114.500000
428	Banana Beach Bar	1014.687500
2757	Moriz Restaurant	959.375000
1201	Delhi Food Point	910.700000
83	Aalishan Restaurant & Caterer	888.454545
3138	Orzuv	837.000000
2527	M.M.A. Kabab's & Biriyani Center	795.500000
2518	Luo Han	777.000000

Next steps: Generate code with `data` View recommended plots

```
bar(data,"name",'votes',"Lowest 10 restuarant have bad vote (name,rate=0,vote)")
```



Relation between location and rest_type for rate

```
df_eda.head(5)
```

	name	online_order	book_table	rate	votes	phone	location	rest_type	cuisines	approx_cost(for two people)	reviews_list	listed_in(
0	Jalsa	Yes	Yes	1.0	775	have phone	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n A beautiful place to ...	
1	Spice Elephant	Yes	No	1.0	787	have phone	Banashankari	Casual Dining	Chinese, North Indian, Thai	800.0	[('Rated 4.0', 'RATED\n Had been here for din...	
2	San Churro Cafe	Yes	No	1.0	918	have phone	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	[('Rated 3.0', "RATED\n Ambience is not that ...	
3	Addhuri Udupi Bhojana	No	No	0.0	88	have phone	Banashankari	Quick Bites	South Indian, North Indian	300.0	[('Rated 4.0', "RATED\n Great food and proper...	
4	Grand					have		Casual	North		[('Rated 4.0', 'RATED\n	

Next steps: Generate code with df_eda View recommended plots

```
#pandas
data=df_eda.groupby(["rate","location","rest_type"]).agg({"rest_type":"count"}).rename(columns={"rest_type":"count"}).reset_index().sort_
data
```

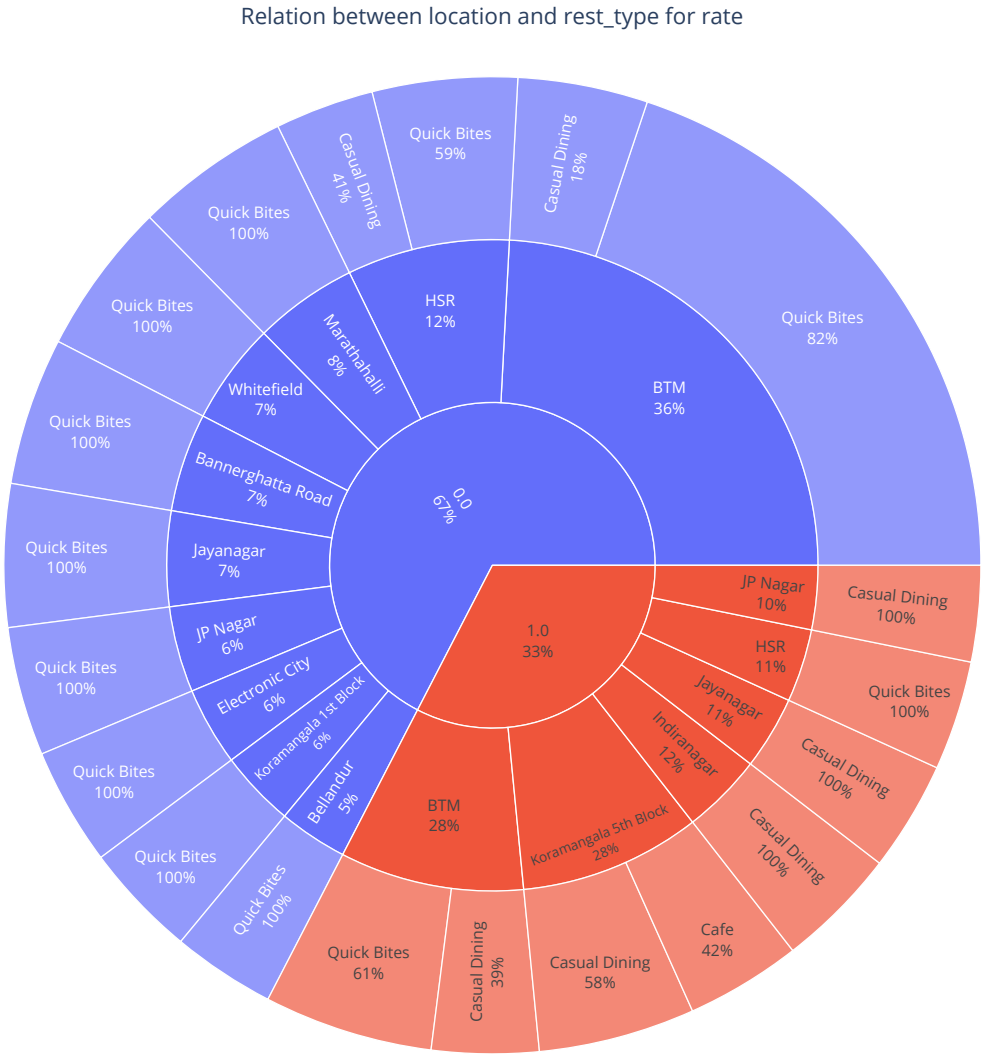
	rate	location	rest_type	count
14	0.0	BTM	Quick Bites	1694
954	1.0	BTM	Quick Bites	477
1421	1.0	Koramangala 5th Block	Casual Dining	442
650	0.0	Marathahalli	Quick Bites	439
910	0.0	Whitefield	Quick Bites	430
75	0.0	Bannerghatta Road	Quick Bites	417
308	0.0	HSR	Quick Bites	409
410	0.0	Jayanagar	Quick Bites	403
7	0.0	BTM	Casual Dining	367
385	0.0	JP Nagar	Quick Bites	364
1249	1.0	Indiranagar	Casual Dining	343
262	0.0	Electronic City	Quick Bites	332
1417	1.0	Koramangala 5th Block	Cafe	326
485	0.0	Koramangala 1st Block	Quick Bites	323
1310	1.0	Jayanagar	Casual Dining	310
1209	1.0	HSR	Quick Bites	309
946	1.0	BTM	Casual Dining	302
134	0.0	Bellandur	Quick Bites	291
298	0.0	HSR	Casual Dining	279
1285	1.0	JP Nagar	Casual Dining	271

Next steps: Generate code with `data` View recommended plots

```
data.rate=data.rate.astype("O")
```

```
def sunburst(data,names,path,values,title_text):
    fig=px.sunburst(data_frame=data,names=names,path=path,values=values,
                width=900,height=900)
    fig.update_traces(textinfo="label+percent parent")
    fig.update_layout(title_text=title_text,title_x=0.5)
    return fig
```

```
sunburst(data,"rate",["rate","location","rest_type"],"count","Relation between location and rest_type for rate")
```



Note:

- Most perferred rest_type for all loctaion is Quick Bites

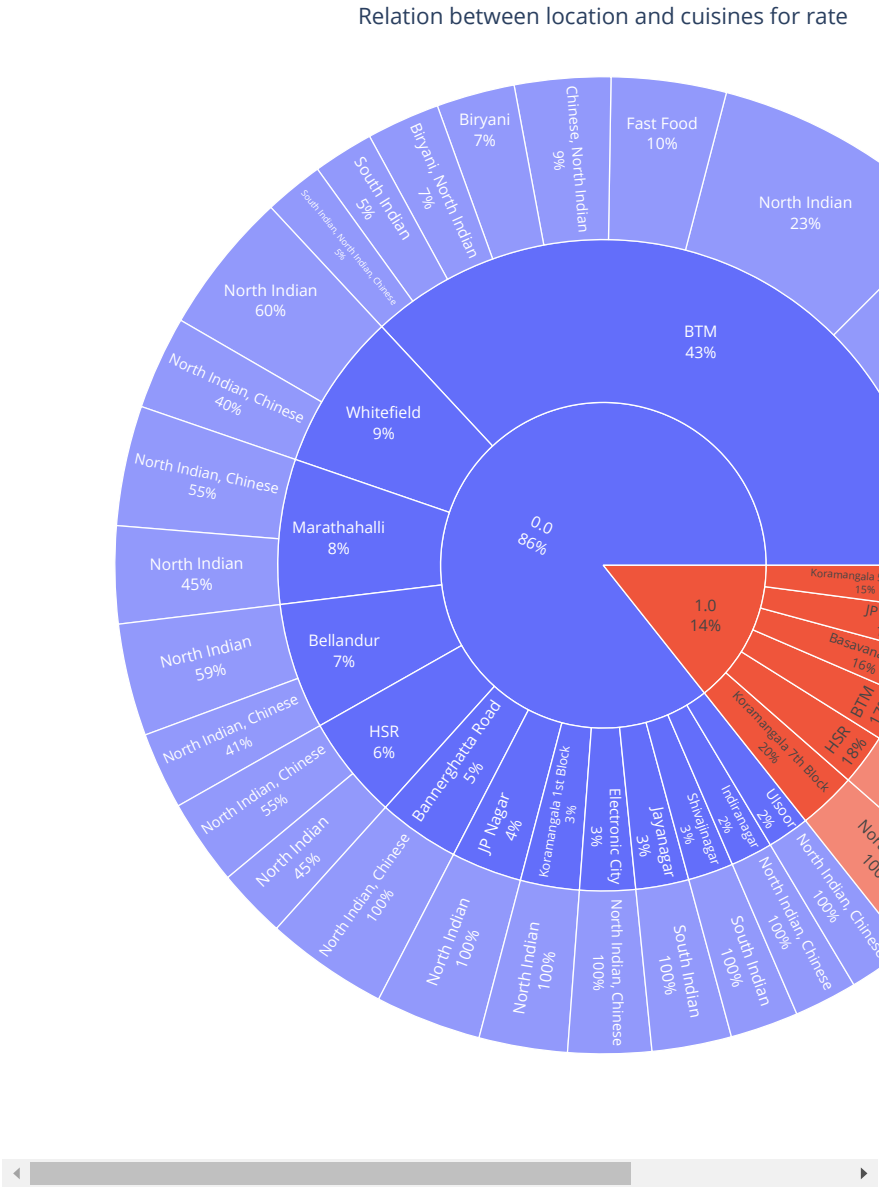
Relation between location and cuisines for rate

```
#pandas
data=df_eda.groupby(["rate","location","cuisines"]).agg({"cuisines":"count"}).rename(columns={"cuisines":"count"}).reset_index().sort_val
data
```

	rate	location	cuisines	count
164	0.0	BTM	North Indian, Chinese	299
153	0.0	BTM	North Indian	202
4155	0.0	Whitefield	North Indian	113
528	0.0	Bannerghatta Road	North Indian, Chinese	97
3109	0.0	Marathahalli	North Indian, Chinese	95
93	0.0	BTM	Fast Food	91
770	0.0	Bellandur	North Indian	89
2036	0.0	JP Nagar	North Indian	84
3101	0.0	Marathahalli	North Indian	77
80	0.0	BTM	Chinese, North Indian	76
4161	0.0	Whitefield	North Indian, Chinese	74
2505	0.0	Koramangala 1st Block	North Indian	70
6449	1.0	Koramangala 7th Block	North Indian	69
1642	0.0	HSR	North Indian, Chinese	68
1345	0.0	Electronic City	North Indian, Chinese	66
2206	0.0	Jayanagar	South Indian	63
40	0.0	BTM	Biryani	62
5272	1.0	HSR	North Indian	61
776	0.0	Bellandur	North Indian, Chinese	61
4394	1.0	BTM	North Indian	58
51	0.0	BTM	Biryani, North Indian	58
1638	0.0	HSR	North Indian	55
4660	1.0	Basavanagudi	South Indian	54
3820	0.0	Shivajinagar	South Indian	54
5682	1.0	JP Nagar	North Indian	51
3923	0.0	Ulsoor	North Indian, Chinese	50
6263	1.0	Koramangala 5th Block	Desserts	50
1884	0.0	Indiranagar	North Indian, Chinese	50
207	0.0	BTM	South Indian	48
220	0.0	BTM	South Indian, North Indian, Chinese	47

Next steps: Generate code with `data` View recommended plots

```
sunburst(data,"rate",["rate","location","cuisines"],"count","Relation between location and cuisines for rate")
```



Note:

- Most perfer cuisines for all loctaion is North Indian or Deserts or South Indian

---

Relation between location and approx_cost for rate

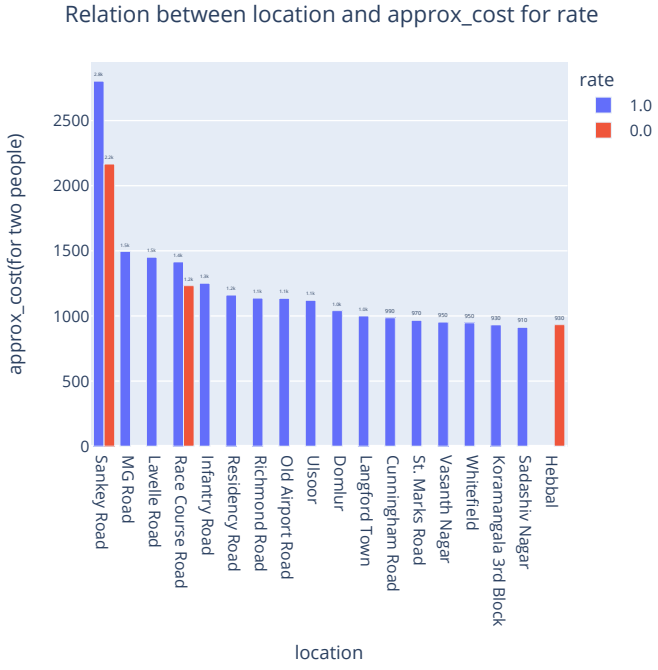
```
data=df_eda.groupby(["rate","location"])["approx_cost(for two people)"].mean().reset_index().sort_values(by="approx_cost(for two people)"
data
```

	rate	location	approx_cost(for two people)
162	1.0	Sankey Road	2802.941176
74	0.0	Sankey Road	2166.666667
142	1.0	MG Road	1495.551601
141	1.0	Lavelle Road	1450.970874
153	1.0	Race Course Road	1415.322581
119	1.0	Infantry Road	1251.111111
65	0.0	Race Course Road	1232.666667
157	1.0	Residency Road	1160.732323
158	1.0	Richmond Road	1137.257618
150	1.0	Old Airport Road	1135.120482
170	1.0	Ulsoor	1119.958848
107	1.0	Domlur	1041.007194
140	1.0	Langford Town	1000.000000
106	1.0	Cunningham Road	986.212121
168	1.0	St. Marks Road	966.356877
172	1.0	Vasanth Nagar	953.271028
174	1.0	Whitefield	947.452229
23	0.0	Hebbal	933.333333
133	1.0	Koramangala 3rd Block	931.962025
159	1.0	Sadashiv Nagar	912.500000

Next steps: Generate code with `data` View recommended plots

```
data.rate=data.rate.astype("O")
```

```
bar(data,"location","approx_cost(for two people)",color="rate",title_text="Relation between location and approx_cost for rate")
```



Note:

- In Sankey Road has higher approx_cost(for two people) may be Sankey Road has expensive resturant

---

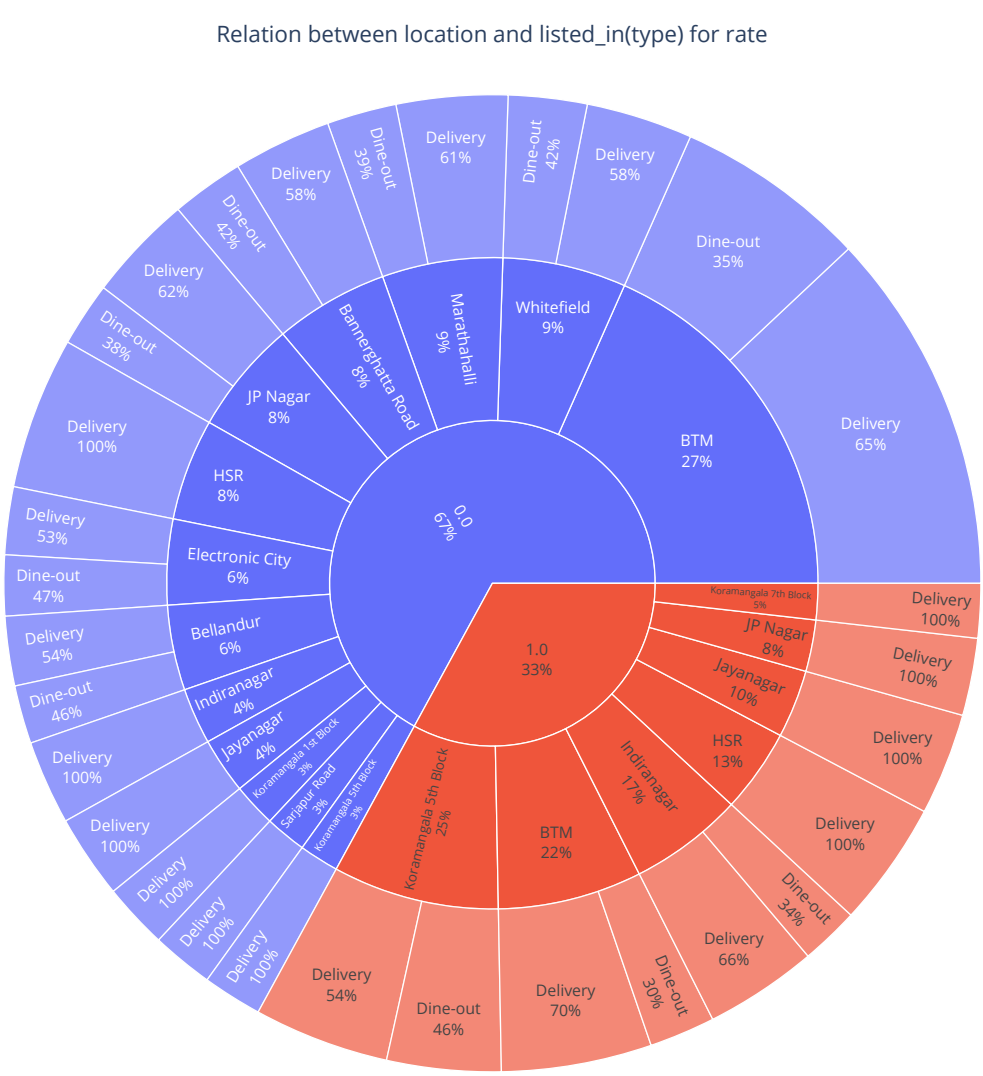
Relation between location and listed_in(type) for rate

```
#pandas
data=df_eda.groupby(["rate","location","listed_in(type)"]).agg({"listed_in(type)":"count"}).rename(columns={"listed_in(type)":"count"}).r
data
```

	rate	location	listed_in(type)	count
2	0.0	BTM	Delivery	1848
4	0.0	BTM	Dine-out	979
115	0.0	HSR	Delivery	777
414	1.0	BTM	Delivery	766
621	1.0	Koramangala 5th Block	Delivery	690
524	1.0	HSR	Delivery	639
623	1.0	Koramangala 5th Block	Dine-out	581
546	1.0	Indiranagar	Delivery	575
263	0.0	Marathahalli	Delivery	567
147	0.0	JP Nagar	Delivery	544
395	0.0	Whitefield	Delivery	543
567	1.0	Jayanagar	Delivery	520
20	0.0	Bannerghatta Road	Delivery	501
135	0.0	Indiranagar	Delivery	429
157	0.0	Jayanagar	Delivery	427
397	0.0	Whitefield	Dine-out	401
559	1.0	JP Nagar	Delivery	394
22	0.0	Bannerghatta Road	Dine-out	370
265	0.0	Marathahalli	Dine-out	355
37	0.0	Bellandur	Delivery	354
99	0.0	Electronic City	Delivery	345
416	1.0	BTM	Dine-out	336
188	0.0	Koramangala 1st Block	Delivery	334
149	0.0	JP Nagar	Dine-out	327
335	0.0	Sarjapur Road	Delivery	315
101	0.0	Electronic City	Dine-out	309
207	0.0	Koramangala 5th Block	Delivery	303
548	1.0	Indiranagar	Dine-out	298
39	0.0	Bellandur	Dine-out	297
635	1.0	Koramangala 7th Block	Delivery	277

Next steps: Generate code with `data` View recommended plots

```
sunburst(data,"rate",["rate","location","listed_in(type)"],"count","Relation between location and listed_in(type) for rate")
```



Note:

- Higher percentage for listed_in(type) to all rate in any location is **Delivery**

Preprocessing

in preprocessing

- drop name , reviews_list and votes because its dont mean any thing in business case
- Impute missing value
- DataFrame after preprocessing
- create Pipeline content all steps of preprocessing

```
df_pre=df_eda.copy()
```

```
df_pre.head()
```

	name	online_order	book_table	rate	votes	phone	location	rest_type	cuisines	approx_cost(for two people)	reviews_list	listed_in(
0	Jalsa	Yes	Yes	1.0	775	have phone	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n A beautiful place to ...	
1	Spice Elephant	Yes	No	1.0	787	have phone	Banashankari	Casual Dining	Chinese, North Indian, Thai	800.0	[('Rated 4.0', 'RATED\n Had been here for din...	
2	San Churro Cafe	Yes	No	1.0	918	have phone	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	[('Rated 3.0', "RATED\n Ambience is not that ...	
3	Addhuri Udupi Bhojana	No	No	0.0	88	have phone	Banashankari	Quick Bites	South Indian, North Indian	300.0	[('Rated 4.0', "RATED\n Great food and proper...	
	Grand					have		Casual	North		[('Rated 4.0', 'RATED\n	

Next steps: Generate code with df_pre View recommended plots

```
#1) drop name because its doesn't mean any thing in business case
df_pre.drop(["name","reviews_list","votes"],axis=1,inplace=True)
```

```
df_pre.head()
```

	online_order	book_table	rate	phone	location	rest_type	cuisines	approx_cost(for two people)	listed_in(type)	listed_in(city)	menu_i
0	Yes	Yes	1.0	have phone	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800.0	Buffet	Banashankari	no
1	Yes	No	1.0	have phone	Banashankari	Casual Dining	Chinese, North Indian, Thai	800.0	Buffet	Banashankari	no
2	Yes	No	1.0	have phone	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	Buffet	Banashankari	no

Next steps: Generate code with df_pre View recommended plots

```
df_pre.columns
```

```
Index(['online_order', 'book_table', 'rate', 'phone', 'location', 'rest_type',
       'cuisines', 'approx_cost(for two people)', 'listed_in(type)',
       'listed_in(city)', 'menu_item_c'],
      dtype='object')
```

```
!pip install category_encoders
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from category_encoders import BinaryEncoder
from sklearn.impute import KNNImputer
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 976.9 kB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.25.2)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.4)
```

```
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.2)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (2.0.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encode
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2023.
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (202
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_e
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encoders
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

```
# !pip install category_encoders
```

```
col_onehot=['online_order', 'book_table', 'phone','menu_item_c'] #columns that apply one hot encoder
col_binary=['location','rest_type', 'cuisines', 'listed_in(type)', 'listed_in(city)']#columns that apply binary encoder
col_robust=['approx_cost(for two people)']#columns that apply Robust scaler
```

```
Start coding or generate with AI.
```

Impute missing value

```
#first create piepline to convert all categorcail to numerical
ct=ColumnTransformer([
    ("oh",OneHotEncoder(),col_onehot),
    ("binary",BinaryEncoder(),col_binary)
])
pipline=Pipeline(steps=[
    ("ct",ct)
])
```

```
encoder=pipline.fit_transform(df_pre)
encoder #array represented to dataframe after enconder and this array not included in col_robust and rate
```

```
array([[0., 1., 0., ..., 0., 0., 1.],
       [0., 1., 1., ..., 0., 0., 1.],
       [0., 1., 1., ..., 0., 0., 1.],
       ...,
       [1., 0., 1., ..., 1., 1., 0.],
       [1., 0., 0., ..., 1., 1., 0.],
       [1., 0., 1., ..., 1., 1., 0.]])
```

```
robust_array=np.array(df_pre[col_robust])
robust_array
```

```
array([[ 800.],
       [ 800.],
       [ 800.],
       ...,
       [2000.],
       [2500.],
       [1500.]])
```

```
robust_array.shape
```

```
(51088, 1)
```

```
rate_array=np.array(df.rate).reshape(-1,1) #i will take rate of df before conert rate to 0,1
rate_array
```

```
array([[4.1],
       [4.1],
       [3.8],
       ...,
       [nan],
       [4.3],
       [3.4]])
```

```
rate_array.shape
```

```
(51088, 1)
```

```
num_array=np.concatenate((robust_array,rate_array),axis=1) #concatenate columns in data (votes','approx_cost(for two people) , rate)
num_array
```

```
array([[ 800. ,    4.1],
       [ 800. ,    4.1],
```

```
            [ 800. ,    3.8],
            ...,
            [2000. ,    nan],
            [2500. ,    4.3],
            [1500. ,    3.4]])
```

```
data_array=np.concatenate((encoder,num_array),axis=1) # array of data after making encoder enable to impute missing value
data_array
```

```
array([[0.0e+00, 1.0e+00, 0.0e+00, ..., 1.0e+00, 8.0e+02, 4.1e+00],
       [0.0e+00, 1.0e+00, 1.0e+00, ..., 1.0e+00, 8.0e+02, 4.1e+00],
       [0.0e+00, 1.0e+00, 1.0e+00, ..., 1.0e+00, 8.0e+02, 3.8e+00],
       ...,
       [1.0e+00, 0.0e+00, 1.0e+00, ..., 0.0e+00, 2.0e+03,     nan],
       [1.0e+00, 0.0e+00, 0.0e+00, ..., 0.0e+00, 2.5e+03, 4.3e+00],
       [1.0e+00, 0.0e+00, 1.0e+00, ..., 0.0e+00, 1.5e+03, 3.4e+00]])
```

```
#impute
imputer = KNNImputer(n_neighbors=5)
imputer_data=imputer.fit_transform(data_array)
```

```
imputer_data # array after impute missing value
```

```
array([[0.00e+00, 1.00e+00, 0.00e+00, ..., 1.00e+00, 8.00e+02, 4.10e+00],
       [0.00e+00, 1.00e+00, 1.00e+00, ..., 1.00e+00, 8.00e+02, 4.10e+00],
       [0.00e+00, 1.00e+00, 1.00e+00, ..., 1.00e+00, 8.00e+02, 3.80e+00],
       ...,
       [1.00e+00, 0.00e+00, 1.00e+00, ..., 0.00e+00, 2.00e+03, 3.82e+00],
       [1.00e+00, 0.00e+00, 0.00e+00, ..., 0.00e+00, 2.50e+03, 4.30e+00],
       [1.00e+00, 0.00e+00, 1.00e+00, ..., 0.00e+00, 1.50e+03, 3.40e+00]])
```

```
imputer_data[:,-1].reshape(-1,1) # column of rate after imputer missing value
```

```
array([[4.1 ],
       [4.1 ],
       [3.8 ],
       ...,
       [3.82],
       [4.3 ],
       [3.4 ]])
```

```
df_pre["rate"]=imputer_data[:,-1].reshape(-1,1) #impute result in our data
```

```
#check missing value
compare=pd.DataFrame({"rate with nan":df_eda["rate"],"rate without nan":df_pre["rate"]})
compare[compare["rate with nan"].isna()]
```

	rate with nan	rate without nan
84	NaN	0.00
90	NaN	2.98
91	NaN	4.00
92	NaN	3.46
107	NaN	3.54
...	...	...
51644	NaN	2.68
51675	NaN	4.16
51710	NaN	4.16
51713	NaN	3.82
51714	NaN	3.82

7667 rows × 2 columns

```
compare[compare["rate with nan"].isna()]["rate without nan"].unique()
```

```
array([0.  , 2.98, 4.  , 3.46, 3.54, 2.9 , 3.06, 2.86, 2.22, 2.96, 3.46,
       3.64, 1.36, 1.48, 2.78, 2.66, 1.52, 3.5 , 2.  , 2.8 , 3.7 , 3.52,
       3.76, 2.62, 3.32, 3.36, 3.8 , 3.62, 3.38, 2.82, 3.86, 2.92, 3.6 ,
       3.92, 3.9 , 3.28, 3.42, 2.7 , 3.4 , 2.72, 1.38, 2.88, 3.66, 1.94,
       2.08, 2.14, 3.62, 3.72, 3.82, 0.76, 3.22, 3.3 , 4.14, 3.02, 2.68,
       2.36, 3.48, 2.76, 1.34, 2.1 , 3.68, 3.44, 3.98, 3.64, 2.02, 3.48,
       2.56, 3.22, 3.56, 3.56, 2.06, 3.24, 3.24, 3.36, 3.58, 1.5 , 2.48,
       2.16, 2.66, 2.04, 2.76, 2.74, 2.54, 3.28, 3.2 , 3.84, 2.68, 2.64,
       3.54, 3.26, 3.06, 3.38, 2.12, 3.94, 3.1 , 3.08, 2.84, 3.34, 2.78,
       1.98, 3.84, 1.94, 4.22, 2.72, 3.74, 1.26, 3.68, 1.32, 2.82, 3.58,
```

```
            2.02, 4.3 , 3.16, 2.92, 3.16, 3.88, 3.96, 3.12, 4.08, 3.78, 3.66,
            3.1 , 3.42, 3.74, 3.34, 2.44, 2.86, 2.98, 2.18, 2.08, 3.18, 2.84,
            1.34, 1.96, 4.04, 2.26, 4.26, 3.02, 1.42, 2.52, 1.56, 3.4 , 3.96,
            4.02, 3.52, 2.64, 3.32, 2.92, 2.7 , 1.4 , 4.06, 2.24, 2.28, 3.88,
            2.94, 3.72, 3.76, 3.18, 3.78, 3.5 , 1.36, 1.3 , 3.82, 1.8 , 2.46,
            1.88, 1.24, 2.44, 2.6 , 2.2 , 2.18, 3.14, 3.04, 2.58, 3.08, 3.14,
            2.56, 1.28, 2.4 , 3.12, 1.18, 2.52, 1.76, 2.26, 3.1 , 3.98, 1.82,
            3.44, 3.92, 1.92, 0.62, 4.1 , 0.88, 3.  , 2.74, 1.26, 4.04, 2.9 ,
            3.2 , 2.36, 2.94, 1.86, 3.  , 1.24, 3.3 , 0.74, 4.32, 1.84, 3.18,
            2.58, 2.9 , 1.9 , 2.88, 0.7 , 2.48, 2.34, 1.92, 2.22, 3.26, 1.98,
            2.28, 2.58, 2.14, 2.42, 1.82, 3.94, 2.62, 1.46, 4.02, 0.6 , 2.82,
            2.32, 2.68, 1.44, 0.66, 4.28, 1.84, 3.94, 2.5 , 2.6 , 2.42, 3.86,
            3.06, 3.8 , 1.68, 3.3 , 1.54, 1.78, 2.34, 4.06, 1.6 , 0.68, 3.84,
            1.46, 2.3 , 4.14, 3.4 , 3.02, 3.16, 4.12, 1.12, 0.9 , 4.4 , 4.2 ,
            4.18, 4.28, 1.58, 1.2 , 4.14, 3.54, 1.22, 4.16, 4.04, 0.52, 2.6 ,
            2.42, 1.64, 4.24, 3.34, 4.18, 4.22, 2.24, 1.62, 0.78, 2.38, 4.3 ,
            3.96, 1.88, 3.08, 1.54, 0.64, 1.38, 0.56, 2.96, 1.78, 4.  , 4.5 ,
            2.48, 1.58, 3.7 , 2.16, 2.32, 3.6 , 1.16, 3.6 , 2.5 , 0.8 , 3.8 ,
            3.7 , 0.82, 0.58, 4.16])
```

```
def success(x):
        if x >3.75:
            return 1
        elif x  <= 3.75:
            return 0
        else:
            return np.nan

df_pre.rate.apply(success)
```

```
0        1
1        1
2        1
3        0
4        1
        ..
51712    0
51713    1
51714    1
51715    1
51716    0
Name: rate, Length: 51088, dtype: int64
```

```
df_pre.rate=df_pre.rate.apply(success)
```

```
df_pre.rate.value_counts()
```

```
rate
0    30289
1    20799
Name: count, dtype: int64
```

```
#final data after impute
df_pre
```

	online_order	book_table	rate	phone	location	rest_type	cuisines	approx_cost(for two people)	listed_in(type)	listed_in(city)	m
0	Yes	Yes	1	have phone	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800.0	Buffet	Banashankari	
1	Yes	No	1	have phone	Banashankari	Casual Dining	Chinese, North Indian, Thai	800.0	Buffet	Banashankari	
2	Yes	No	1	have phone	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	Buffet	Banashankari	
3	No	No	0	have phone	Banashankari	Quick Bites	South Indian, North Indian	300.0	Buffet	Banashankari	
4	No	No	1	have phone	Basavanagudi	Casual Dining	North Indian, Rajasthani	600.0	Buffet	Banashankari	
...	...	...	...	...	...	...	...	...	...	...	...
51712	No	No	0	have phone	Whitefield	Bar	Continental	1500.0	Pubs and bars	Whitefield	
51713	No	No	1	have phone	Whitefield	Bar	Finger Food	600.0	Pubs and bars	Whitefield	
				not			Finger				

Next steps: Generate code with df_pre View recommended plots

```
#check null
df_pre.isna().sum()
```

```
online_order                  0
book_table                    0
rate                          0
phone                         0
location                      0
rest_type                     0
cuisines                      0
approx_cost(for two people)   0
listed_in(type)               0
listed_in(city)               0
menu_item_c                   0
dtype: int64
```

```
#save data
df_pre.to_csv("sample.csv",index=False)
```

```
Start coding or generate with AI.
```

DataFrame after preprocessing

```
X=df_pre.drop("rate",axis=1)
y=df_pre.rate
```

```
encoder=OneHotEncoder()
ct1=ColumnTransformer([("encoder",encoder,col_onehot)])
```

```
X_df=ct1.fit_transform(X)
```

```
X_df
```

```
array([[0., 1., 0., ..., 0., 0., 1.],
       [0., 1., 1., ..., 0., 0., 1.],
       [0., 1., 1., ..., 0., 0., 1.],
       ...,
       [1., 0., 1., ..., 1., 0., 1.],
       [1., 0., 0., ..., 0., 0., 1.],
       [1., 0., 1., ..., 0., 0., 1.]])
```

```
X_df.shape
```

In [142...
```
#save data
df_pre.to_csv("sample.csv",index=False)
```

In [142...
```

```

DataFrame after preprocessing

In [143...
```
X=df_pre.drop("rate",axis=1)
y=df_pre.rate
```

In [144...
```
encoder=OneHotEncoder()
ct1=ColumnTransformer([("encoder",encoder,col_onehot)])
```

In [145...
```
X_df=ct1.fit_transform(X)
```

In [146...
```
X_df
```

Out[146]:
```
array([[0., 1., 0., ..., 0., 0., 1.],
       [0., 1., 1., ..., 0., 0., 1.],
       [0., 1., 1., ..., 0., 0., 1.],
       ...,
       [1., 0., 1., ..., 1., 0., 1.],
       [1., 0., 0., ..., 0., 0., 1.],
       [1., 0., 1., ..., 0., 0., 1.]])
```

In [147...
```
X_df.shape
```

Out[147]:
```
(51088, 8)
```

In [148...
```
df_one=pd.DataFrame(X_df,columns=ct1.get_feature_names_out()).reset_index() #datafr
df_one
```

Out[148]:

	index	encoder__online_order_No	encoder__online_order_Yes	encoder__book_table_No	enco
0	0	0.0	1.0	0.0	
1	1	0.0	1.0	1.0	
2	2	0.0	1.0	1.0	
3	3	1.0	0.0	1.0	
4	4	1.0	0.0	1.0	
...	...	...	...	...	
51083	51083	1.0	0.0	1.0	
51084	51084	1.0	0.0	1.0	
51085	51085	1.0	0.0	1.0	
51086	51086	1.0	0.0	0.0	
51087	51087	1.0	0.0	1.0	

51088 rows × 9 columns

In [149...
```
encoder = BinaryEncoder(cols=col_binary)
df_encoded = encoder.fit_transform(X)
```

```
# Get the column names after binary encoding
encoded_column_names = df_encoded.columns.tolist()
```

In [150…
```
df_encoded=df_encoded.reset_index()
```

In [151…
```
df_encoded # dataframe contains all columns (before applying onehot encoder and aft
```

Out[151]:

	index	online_order	book_table	phone	location_0	location_1	location_2	location_3	loc
0	0	Yes	Yes	have phone	0	0	0	0	
1	1	Yes	No	have phone	0	0	0	0	
2	2	Yes	No	have phone	0	0	0	0	
3	3	No	No	have phone	0	0	0	0	
4	4	No	No	have phone	0	0	0	0	
...	...	...	...	...	...	...	...	...	
51083	51712	No	No	have phone	0	0	1	1	
51084	51713	No	No	have phone	0	0	1	1	
51085	51714	No	No	not have phone	0	0	1	1	
51086	51715	No	Yes	have phone	1	0	0	0	
51087	51716	No	No	have phone	1	0	0	0	

51088 rows × 40 columns

In [152…
```
df_encoded.drop(col_onehot,axis=1,inplace=True)# drop onehot columns
```

In [153…
```
df_encoded
```

Out[153]:

	index	location_0	location_1	location_2	location_3	location_4	location_5	location_6	res
0	0	0	0	0	0	0	0	1	
1	1	0	0	0	0	0	0	1	
2	2	0	0	0	0	0	0	1	
3	3	0	0	0	0	0	0	1	
4	4	0	0	0	0	0	1	0	
...	...	...	...	...	...	...	...	...	
51083	51712	0	0	1	1	1	1	1	
51084	51713	0	0	1	1	1	1	1	
51085	51714	0	0	1	1	1	1	1	
51086	51715	1	0	0	0	0	0	1	
51087	51716	1	0	0	0	0	0	1	

51088 rows × 36 columns

In [154…
```
df_encoded.columns
```

Out[154]:
```
Index(['index', 'location_0', 'location_1', 'location_2', 'location_3',
       'location_4', 'location_5', 'location_6', 'rest_type_0', 'rest_type_1',
       'rest_type_2', 'rest_type_3', 'rest_type_4', 'rest_type_5',
       'rest_type_6', 'cuisines_0', 'cuisines_1', 'cuisines_2', 'cuisines_3',
       'cuisines_4', 'cuisines_5', 'cuisines_6', 'cuisines_7', 'cuisines_8',
       'cuisines_9', 'cuisines_10', 'cuisines_11',
       'approx_cost(for two people)', 'listed_in(type)_0', 'listed_in(type)_1',
       'listed_in(type)_2', 'listed_in(city)_0', 'listed_in(city)_1',
       'listed_in(city)_2', 'listed_in(city)_3', 'listed_in(city)_4'],
      dtype='object')
```

In [155…
```
p=df_encoded['approx_cost(for two people)']
p
```

Out[155]:
```
0         800.0
1         800.0
2         800.0
3         300.0
4         600.0
          ...
51083    1500.0
51084     600.0
51085    2000.0
51086    2500.0
51087    1500.0
Name: approx_cost(for two people), Length: 51088, dtype: float64
```

In [156…
```
df_encoded.drop("approx_cost(for two people)",axis=True,inplace=True)
#add agin in last position
df_encoded['approx_cost(for two people)']=p
```

In [157…
```
df_encoded # dataframe don't have result of one hot encoder so merge one hot encode
```

Out[157]:

	index	location_0	location_1	location_2	location_3	location_4	location_5	location_6	re
0	0	0	0	0	0	0	0	1	
1	1	0	0	0	0	0	0	1	
2	2	0	0	0	0	0	0	1	
3	3	0	0	0	0	0	0	1	
4	4	0	0	0	0	0	1	0	
...	...	...	...	...	...	...	...	...	
51083	51712	0	0	1	1	1	1	1	
51084	51713	0	0	1	1	1	1	1	
51085	51714	0	0	1	1	1	1	1	
51086	51715	1	0	0	0	0	0	1	
51087	51716	1	0	0	0	0	0	1	

51088 rows × 36 columns

In [158…

```
#merge
df_preprocces=df_one.merge(df_encoded,on="index",how="inner").drop("index",axis=1)
df_preprocces #final dataframe after encoder
```

Out[158]:

	encoder__online_order_No	encoder__online_order_Yes	encoder__book_table_No	encoder__bo
0	0.0	1.0	0.0	
1	0.0	1.0	1.0	
2	0.0	1.0	1.0	
3	1.0	0.0	1.0	
4	1.0	0.0	1.0	
...	...	...	...	
50462	1.0	0.0	1.0	
50463	1.0	0.0	1.0	
50464	1.0	0.0	1.0	
50465	1.0	0.0	0.0	
50466	1.0	0.0	1.0	

50467 rows × 43 columns

In [158…

Pipeline

In [159…
```
models=[]
models.append(("LR",LogisticRegression(max_iter=1000)))
models.append(("KNN",KNeighborsClassifier()))
models.append(("DTC",DecisionTreeClassifier()))
models.append(("SVC",SVC()))
models.append(("RF",RandomForestClassifier()))
models.append(("GB",GradientBoostingClassifier()))
models.append(("XGB",XGBClassifier()))
```

In [159…

In [160…
```
col_onehot=['online_order', 'book_table', 'phone','menu_item_c'] #columns that appl
col_binary=['location','rest_type', 'cuisines', 'listed_in(type)', 'listed_in(city)
col_robust=['approx_cost(for two people)']#columns that apply Robust scaler
```

In [161…
```
for model in models:
    steps_ct=[]#list contain steps that process in only ColumnTransformer
    steps=[] #list contain steps that process in pipeline
    steps_ct.append(("ohe",OneHotEncoder(),col_onehot))
    steps_ct.append(("binary",BinaryEncoder(),col_binary))
    steps_ct.append(("robust",RobustScaler(),col_robust))
    steps.append(("ct",ColumnTransformer(steps_ct)))
    steps.append(model)#final append models
    pipline=Pipeline(steps=steps)
    score=cross_validate(pipline,X,y,cv=5,scoring="f1",return_train_score=True)
    print(model[0])
    print("Training accuracy =",score["train_score"].mean())
    print("testing accuracy =",score["test_score"].mean())
    print("*"*50)
```

```
LR
Training accuracy = 0.614711219411042
testing accuracy = 0.5779706874808719
**************************************************
KNN
Training accuracy = 0.8937118254613663
testing accuracy = 0.8020532653550901
**************************************************
DTC
Training accuracy = 0.9900754866669679
testing accuracy = 0.8189402086437865
**************************************************
SVC
Training accuracy = 0.7492184318313995
testing accuracy = 0.6763450183928476
**************************************************
RF
Training accuracy = 0.990125408885014
testing accuracy = 0.8369264616336899
**************************************************
GB
Training accuracy = 0.6488610622580216
testing accuracy = 0.6105385447220639
**************************************************
XGB
Training accuracy = 0.8387613975140626
testing accuracy = 0.7399854962864651
**************************************************
```

best model is Random Forest

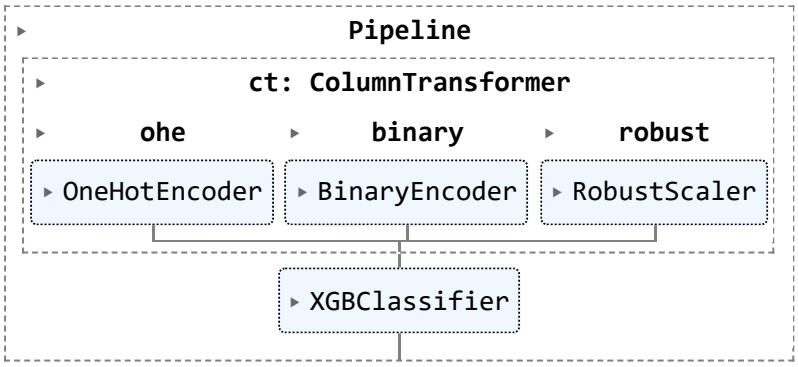
In [161…

Model Tuning

In [162…
```
# initial model
XGB=XGBClassifier(learning_rate =0.1,n_estimators=1000,max_depth=5,min_child_weight

steps_ct=[]#list content steps that process in only ColumnTransformer
steps=[] #list content steps that process in pipeline
steps_ct.append(("ohe",OneHotEncoder(),col_onehot))
steps_ct.append(("binary",BinaryEncoder(),col_binary))
steps_ct.append(("robust",RobustScaler(),col_robust))
steps.append(("ct",ColumnTransformer(steps_ct)))
steps.append(("XGB",XGB))#final append models
pipline=Pipeline(steps=steps)
```

In [163…
```
pipline
```

Out[163]:



In [164…
```
score=cross_validate(pipline,X,y,cv=5,scoring="f1",return_train_score=True)
print("Training accuracy of Inital XGB =",score["train_score"].mean())
print("Testing accuracy of Inital XGB=",score["test_score"].mean())
```

```
Training accuracy of Inital XGB = 0.8849122458358101
Testing accuracy of Inital XGB= 0.7702350583411369
```

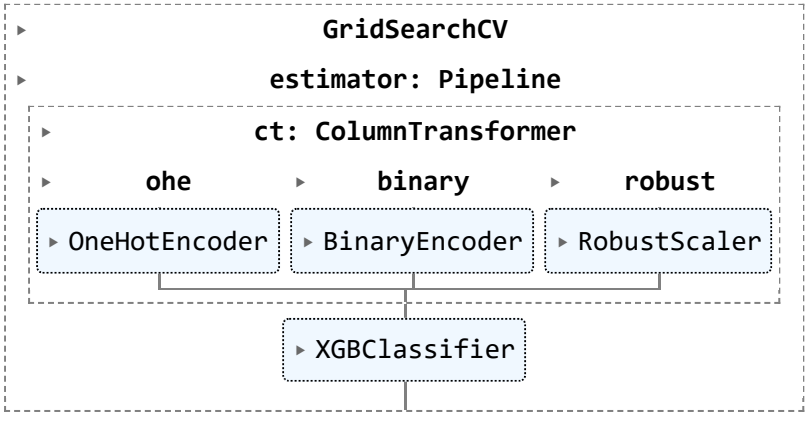
In [164…

Step 2: Tune max_depth and min_child_weight.

In [165…
```
param_test1 = {'XGB__max_depth':range(3,10,2),'XGB__min_child_weight':range(1,6,2)}
gsearch1=GridSearchCV(estimator = pipline, param_grid = param_test1, scoring='f1',c
```

In [166…
```
gsearch1.fit(X,y)
```

Out[166]:



In [167…
```
#best parameter
print("Best parameter in XGB 1",gsearch1.best_params_)
```

```
Best parameter in XGB 1 {'XGB__max_depth': 9, 'XGB__min_child_weight': 1}
```

In [168…
```
score1=cross_validate(gsearch1.best_estimator_,X,y,cv=5,scoring="f1",return_train_s
print("Training accuracy of XGB 1 =",score1["train_score"].mean())
print("Testing accuracy of XGB 1 =",score1["test_score"].mean())
```

```
Training accuracy of XGB 1 = 0.9900157829989146
Testing accuracy of XGB 1 = 0.849148056097867
```

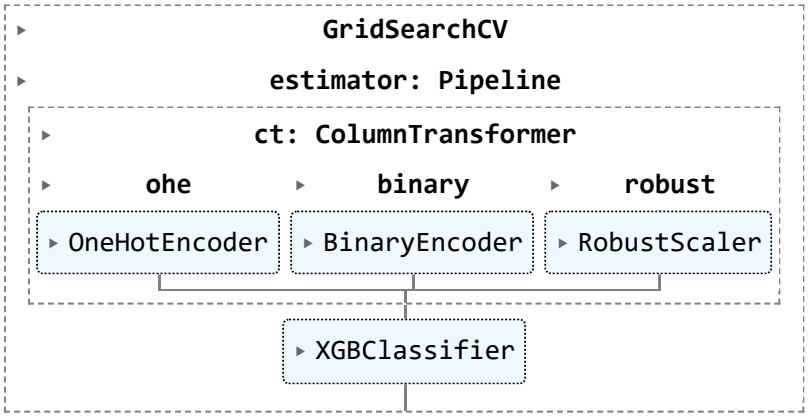
In [168…

Step 3: Tune gamma.

In [169…
```
param_test2 = {'XGB__gamma':[i/10.0 for i in range(0,5)]}
gsearch2=GridSearchCV(estimator = gsearch1.best_estimator_, param_grid = param_test
```

In [170…
```
gsearch2.fit(X,y)
```

Out[170]:



In [171…
```
#best parameter
print("Best parameter in XGB 2",gsearch2.best_params_)
```

```
Best parameter in XGB 2 {'XGB__gamma': 0.0}
```

In [172…
```
score2=cross_validate(gsearch2.best_estimator_,X,y,cv=5,scoring="f1",return_train_s
print("Training accuracy of XGB 2 =",score2["train_score"].mean())
print("Testing accuracy of XGB 2 =",score2["test_score"].mean())
```

```
Training accuracy of XGB 2 = 0.9900157829989146
Testing accuracy of XGB 2 = 0.849148056097867
```

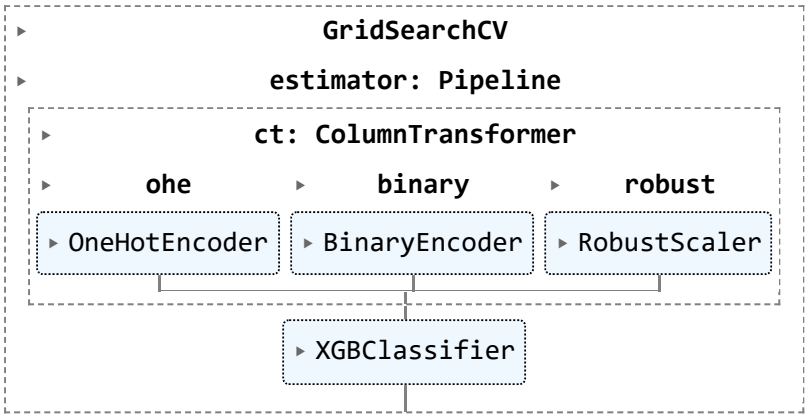
In [172…

Step 4: Tune subsample and colsample_bytree

In [173…
```
param_test3 = {
 'XGB__subsample':[i/10.0 for i in range(6,10)],
 'XGB__colsample_bytree':[i/10.0 for i in range(6,10)]}
gsearch3=GridSearchCV(estimator = gsearch2.best_estimator_, param_grid = param_test
```

In [174…
```
gsearch3.fit(X,y)
```

Out[174]:



In [175…
```
#best parameter
print("Best parameter in XGB 3",gsearch3.best_params_)
```

```
Best parameter in XGB 3 {'XGB__colsample_bytree': 0.9, 'XGB__subsample': 0.9}
```

In [176…
```
score3=cross_validate(gsearch3.best_estimator_,X,y,cv=5,scoring="f1",return_train_s
print("Training accuracy of XGB 3 =",score3["train_score"].mean())
print("Testing accuracy of XGB 3 =",score3["test_score"].mean())
```

```
Training accuracy of XGB 3 = 0.9900523833434185
Testing accuracy of XGB 3 = 0.8536208359965105
```

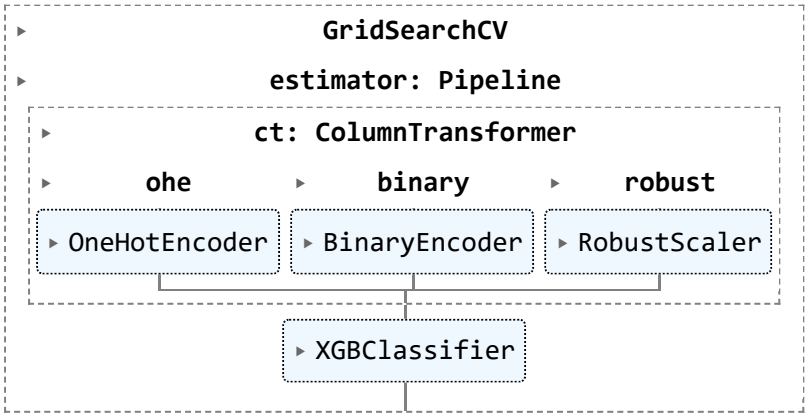
In [176…

Step 5: Tuning regularization parameters

In [177…
```
param_test4 = {'XGB__reg_alpha':[1e-5, 1e-2, 0.1, 1, 100]}

gsearch4=GridSearchCV(estimator = gsearch3.best_estimator_, param_grid = param_test
```

In [178…
```
gsearch4.fit(X,y)
```

Out[178]:



In [179…
```
#best parameter
print("Best parameter in XGB 4",gsearch4.best_params_)
```

```
Best parameter in XGB 4 {'XGB__reg_alpha': 0.1}
```

In [180…
```
score4=cross_validate(gsearch4.best_estimator_,X,y,cv=5,scoring="f1",return_train_s
print("Training accuracy of XGB 4 =",score4["train_score"].mean())
print("Testing accuracy of XGB 4 =",score4["test_score"].mean())
```

```
Training accuracy of XGB 4 = 0.9900446738405211
Testing accuracy of XGB 4 = 0.8550520500754333
```

In [180…

Features importance

In [181…
```
gsearch4.best_estimator_.steps[1]
```

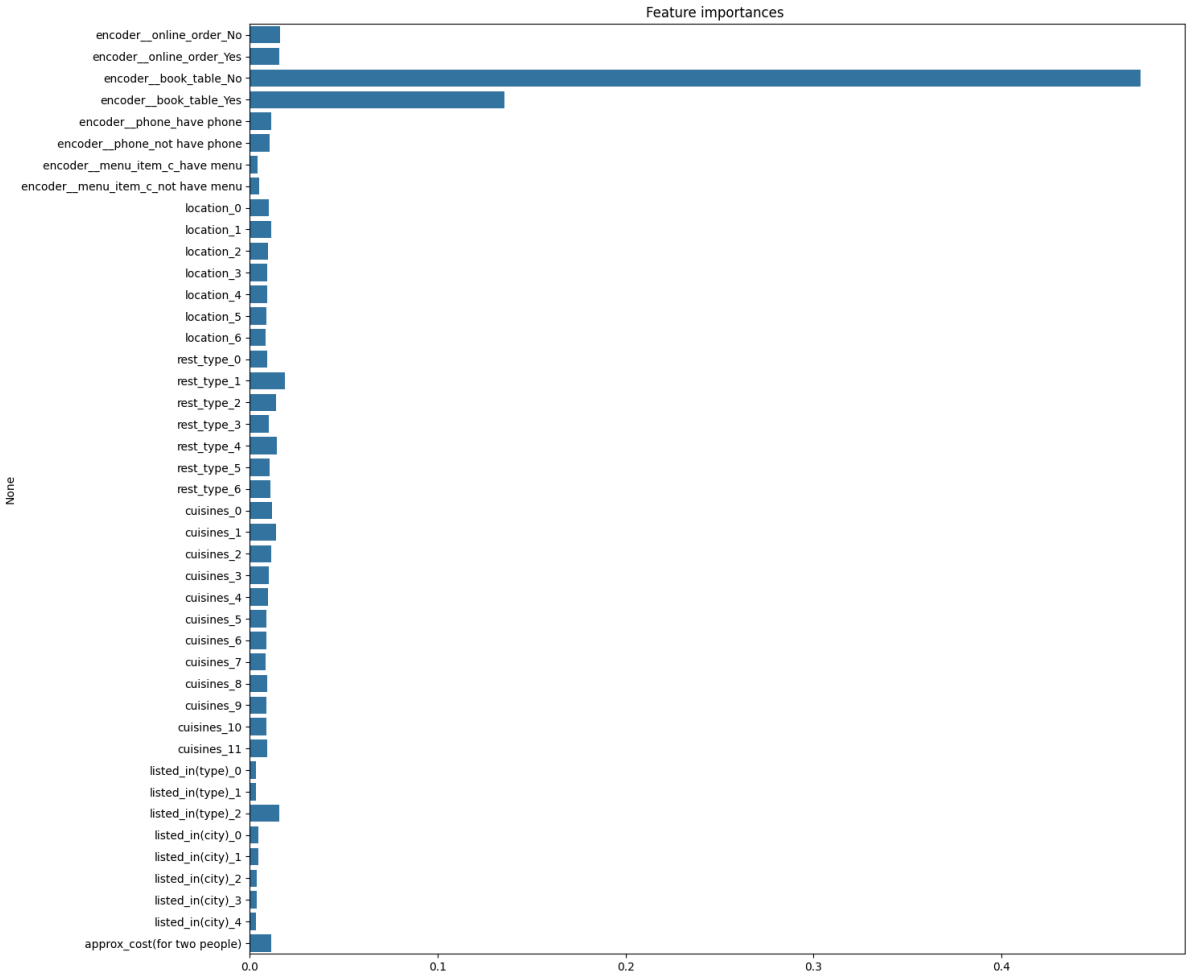
Out[181]:
```
('XGB',
 XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=0.9, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=0.0, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=0.1, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=9, max_leaves=None,
               min_child_weight=1, missing=nan, monotone_constraints=None,
               multi_strategy=None, n_estimators=1000, n_jobs=None, nthread=4,
               num_parallel_tree=None, ...))
```

In [182…
```
gsearch4.best_estimator_.steps[1][1].feature_importances_
```

Out[182]:
```
array([0.01612954, 0.01545956, 0.4738267 , 0.13559932, 0.01120178,
       0.01050962, 0.00414811, 0.00504766, 0.01008303, 0.01126624,
       0.00970723, 0.00942084, 0.0092627 , 0.00893976, 0.00831349,
       0.00944343, 0.01856108, 0.01415569, 0.0102718 , 0.01427508,
       0.01044921, 0.01079125, 0.01199666, 0.01394541, 0.01146486,
       0.01015314, 0.00985266, 0.00867313, 0.0088745 , 0.00847328,
       0.00920875, 0.00894928, 0.00889232, 0.00912378, 0.00324162,
       0.0033684 , 0.01584267, 0.00440645, 0.00451782, 0.00367548,
       0.00371658, 0.00325272, 0.01150736], dtype=float32)
```

In [183…
```
feature_important=gsearch4.best_estimator_.steps[1][1].feature_importances_
```

In [184…
```
plt.figure(figsize=(15,15))
sns.barplot(x=feature_important,y=df_preprocces.columns)
plt.title("Feature importances ")
plt.show()
```



- Note : **Book Table** is the most important feature in data

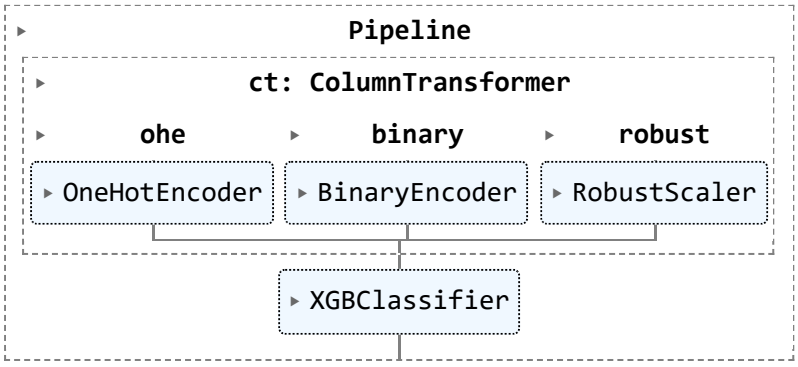
The final accuracy before and after tuning :

In [185…
```
print("Training accuracy of before tuning =",score["train_score"].mean().round(4))
print("Testing accuracy of before tuning =",score["test_score"].mean().round(4))
print("*"*50)
print("Training accuracy of after tuning  =",score3["train_score"].mean().round(4))
print("Testing accuracy of after tuning =",score3["test_score"].mean().round(4))
```

```
Training accuracy of before tuning = 0.8849
Testing accuracy of before tuning = 0.7702
**************************************************
Training accuracy of after tuning  = 0.9901
Testing accuracy of after tuning = 0.8536
```

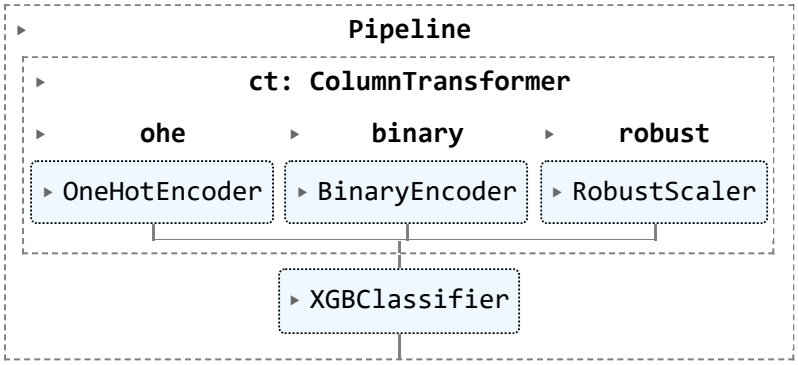
In [186…
```
model=gsearch4.best_estimator_
model
```

Out[186]:



In [187…
```
model.fit(X,y)
```

Out[187]:



In [187…

Save and load Model

In [188…
```
import joblib
joblib.dump(model,"model.pkl")#save model
joblib.dump(X.columns,"input.pkl")#save input columns

model=joblib.load("model.pkl") #Load model
```

In [188…

Testing model

In [189…
```
# in this sample data rate = 1
data={'online_order': 'Yes',
 'book_table': 'No',
 'phone':'have phone',
 'location':'MG Road',
 'rest_type':'Cafe, Bakery',
 'cuisines':'Cafe, Bakery',
 'approx_cost(for two people)':500.0,
 'menu_item_c':'have menu',
 'listed_in(type)':'Desserts',
 'listed_in(city)':'Church Street'}
df_test=pd.DataFrame(data,index=[0])
df_test
```

Out[189]:

	online_order	book_table	phone	location	rest_type	cuisines	approx_cost(for two people)	menu_item_c	l
0	Yes	No	have phone	MG Road	Cafe, Bakery	Cafe, Bakery	500.0	have menu	

In [190…
```
model.predict(df_test)
```

Out[190]:
```
array([1])
```

In [191…
```
#function predict
def prediction(df):
    value_predict=model.predict(df)[0]
    if value_predict ==1:
        return "This restaurant will be successful"
```

```
    else:
        return "This restaurant will be not successful"
```

In [192…
```
prediction(df_test)
```

Out[192]:
```
'This restaurant will be successful'
```

In [194…
```
!pip install nbconvert
```

In [193…
```
!jupyter nbconvert --to html /content/Capstone.ipynb
```

In [194…
```

```