Low-Level Design (LLD)

Restaurant Rating Prediction

Contents

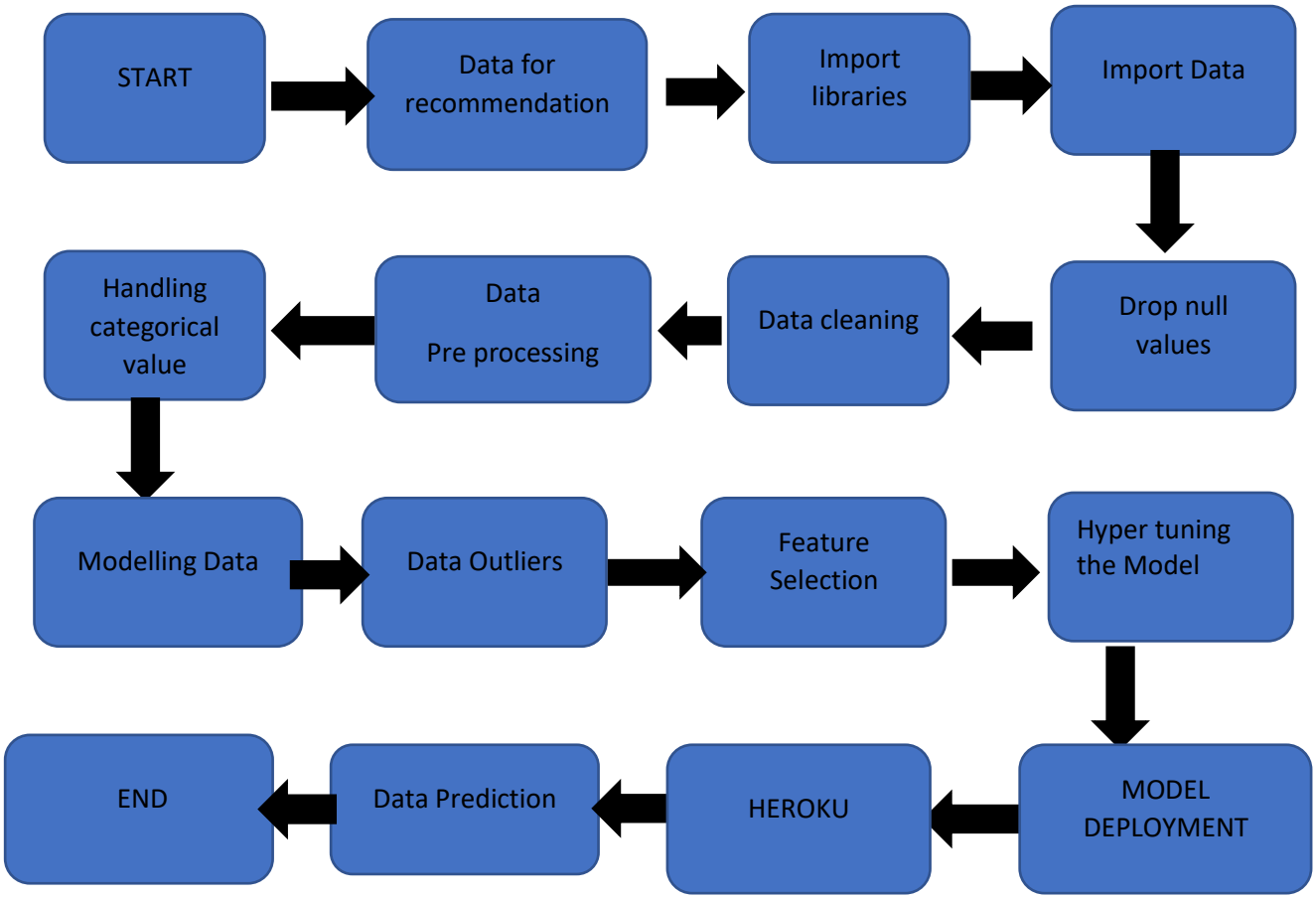
1. Introduction

Restaurants from all over the world can be found here in Bengaluru. From the United States to Japan, Russia to Antarctica, you get all types of cuisines here. Delivery, Dine-out, Pubs, Bars, Drinks, Buffet, Desserts you name it and Bengaluru has it. Bengaluru is the best place for foodies. The number of restaurants is increasing day by day. Currently, it stands at approximately 12,000 restaurants. With such a high number of restaurants. This industry hasn't been saturated yet. And new restaurants are opening every day. However, it has become difficult for them to compete with already established restaurants. The key issues that continue to pose a challenge to them include high real estate costs, rising food costs, shortage of quality manpower, fragmented supply chain and over-licensing. This Zomato data aims at analysing the demography of the location. Most importantly it will help new restaurants in deciding their theme, menus, cuisine, cost etc for a particular location. It also aims at finding similarities between neighbourhoods of Bengaluru on the basis of food.

- Does the demography of the area matter?
- Does the location of a particular type of restaurant depends on people living in that area>
- Does the theme of the restaurant matter?
- Is food chain category restaurant likely to have more customers than its counterpart?
- Are any neighbourhoods similar based on the type of food?
- Is particular neighbours is famous for its own kind of food?
- If two neighbours are similar does that mean these are related or a particular group of people live in a neighbourhood or these are places to eat?
- What kind of food is famous in locality.
- Do entire locality loves veg food, if yes then the locality is populated by a particular set of people e.g. Jain, Gujarati, Marwari who are basically veg.

The dataset also contains reviews for each of the restaurants which will help in finding an overall rating for the place. So, we will try to predict the rating for a particular restaurant. We need to predict rating based on different parameters like Average_cost for two people, Online Order available, foods, menu list, most liked dishes etc features.

2. Architecture



3. Architecture

3.1 Data Description

- The CSV file contains 51717 entries and 17 columns.
- The entries have different parameters like Average_cost for two people, Online Order available, foods, menu list, most liked dishes etc.

3.2 Import libraries

- Libraries are important because they remind us about who we were with archives and reference materials and help us become who we will be.
- Libraries are so much more than a place to read, they are a place to grow and discover.
- We have used visualization, and for importing this excel file we have used pandas.

3.3 Import Data

- Data is all about restaurant prices and some other information.

3.4 Drop null value

- After loading it is important to check null values in a column or a row
- If it is present then the following can be done,
- Filling Nan values with mean, median, mode using fill_na () method
- If fewer missing values, we can drop it as well.

3.5 Data cleaning

- The data types of cost and rate is an object. So, firstly we convert it into date and time for a proper prediction.
- When combining multiple data sources, there are many opportunities for data to be duplicated or mislabelled. If data is

incorrect, outcomes and algorithms are unreliable, even though they may look correct.

3.6 Data Pre-processing

Data preprocessing steps we can use are null value handling, punctuation removal, imbalanced data set handling, handling column with standard deviation zero or below a threshold, etc.

3.7 Handling categorical data

- We are using two main Encoding Techniques to covert Categorical data into some numerical format
- *Nominal data -- Data that are not in any order -->one hot encoding*
- *ordinal data -- Data are in order --> label Encoder*

3.8 Data Modelling

- This is a useful way to model data objects, but the datastore does not require that every entity of a given kind have the same set of properties.
- We have used KN Neighbours Regressor, Linear Regression, Ada Boost Regressor, Bagging Regressor, Extra tree Regressor, Decision Tree Regressor, Gradient Boosting Regressor,
  Random Forest Regressor for predictions.

3.9 Data outlier

- Handling outliers:

  As if there are some outliers in the cost feature, we replace it with a median.
- An outlier is a data point in a data set that is distant from all other observations.

3.10 Hyper tuning the model

- When we build a model for hyper tuning, we also define the hyperparameter search space in addition to the model architecture.
- The model we set up for hyper tuning is called a hyper model
- We have used for increasing our accuracy

3.12 Model Dump:

After comparing all accuracies and checking all r2_score and adjusted r2_score we have chosen hyper parameterized Extra tree Regressor as our best model by their results so we have dumped this model is in a pickle file format with the help of pickle python module.

3.13 User Interface:

In Frontend creation, we have made a user interactive page where user can enter their input values into our application. On this frontend page, we have made a form that has beautiful styling with CSS and Bootstrap. These HTML user input data are transferred in JSON format to the backend. Made these HTML fully in a decoupled format.

3.14. Deployment

We will be deploying the model to Heroku cloud platforms.

3.15 Predictions

We have used many for predicting

- Linear Regression
- KN Neighbours Regressor
- Decision Tree Regressor
- Gradient Boosting Regressor
- Random Forest Regressor
- Ada Boost Regressor
- Extra tree Regressor
- Bagging Regressor
- Hyper tuning for an increase in Accuracy

19. UNIT TEST CASES

Test Case description	Pre- Requisite	Expected Result
Verify whether the Application URL is accessible to user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application load completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify Response time of URL from backend model.	1. Application is accessible	The latency and accessibility of application are very faster we got in the awsec2 service.

Verify whether the user is giving standard input.	1. Handled test cases at backends.	Users should be able to see successfully valid results.
Verify whether the user is able to see input fields on logging in	1. Application is accessible 2. User is logged in to the application	User should be able to see input fields on logging in fields
Verify whether the user is able to edit all input fields	1. Application is accessible 2. User is logged in to the application	User should be able to edit all input
Verify whether the user is presented with recommended results on clicking submit	1. Application is accessible 2. User who is logged in to the application	should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance with the selections user made	1. Application is accessible 2. User is logged in to the application and database	The recommended results should be in accordance to the user of the selection made
Verify whether the user has options to filter the	1. Application is accessible 2. User is logged	User should have options to filter the recommended results as

recommended results as well	in to the application	well
Verify whether the KPIs indicate details of the correct inputs	1. Application is accessible 2. The user is logged in to the application	The KPIs should indicate details of the suggested inputs to users.