Low Level Design for Restaurant Rating Prediction

Prepared by

Vikram Nayyar

vikramnayyar@live.com

November 30, 2021

Abstract

In the past six years, over 1400 new eateries have opened in Bangalore. Evidently, the population's willingness to visit restaurants; has phenomenally increased. Subsequently, numerous restaurant categories have emerged at several locations. Furthermore, they offer several cuisines, price ranges, services and offers. Therefore, predicting the customer's liking is remarkably challenging. To overcome the limitation, an app is developed that predicts the rating of a restaurant. This is based on six user inputs. The prediction model was trained and validated using various data science models. The best validated model was developed to form a User Application.

1. Introduction

1.1 Purpose

The document provides a technical description of the Restaurant Rating Prediction Application. The application was developed to accurately predict a restaurant's rating.

This report provides essential insights to overcome any difficulties in application development.

1.2 Scope

The document is aimed to be a foundational basis for detailed technical insights of the application. The targeted audience is

- Development Team
- Administration

1.3 Definitions

- **GUI**- Graphical User Interface is the front end of the application. This is an interface between the user and the model.
- **Model**- This is a data science model; created after training on a suitable dataset.
- **Prediction-** The output of the model; that represents the outcome.
- **User-** The customer using the application.

2. General Description

2.1 Proposed Solution

A general product design is represented in figure 1. Based on the particular restaurant, a user enters desired inputs. GUI accepts the inputs; and assigns them values according to the model dictionary. Succeeding inputs are fed to the model.

The model processes the inputs and sends the predictions to the GUI. GUI displays the output to the user.

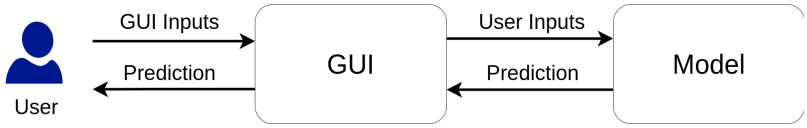

Fig. 1. General design of the data science application.

2.2 Wireframe

Wireframe of the product's UI is depicted in figure 2. In order to predict rating, the application expects 6 user inputs. Each input is properly labeled and is clearly annotated with the user selection. This prevents any confusion in selection. Please note; usual values are pre-selected for each user field. This is for the user's convenience.

The user selections are followed by the predict button. Clicking this button; inputs the user selections to the model, and the predicted rating is displayed below the predict button.

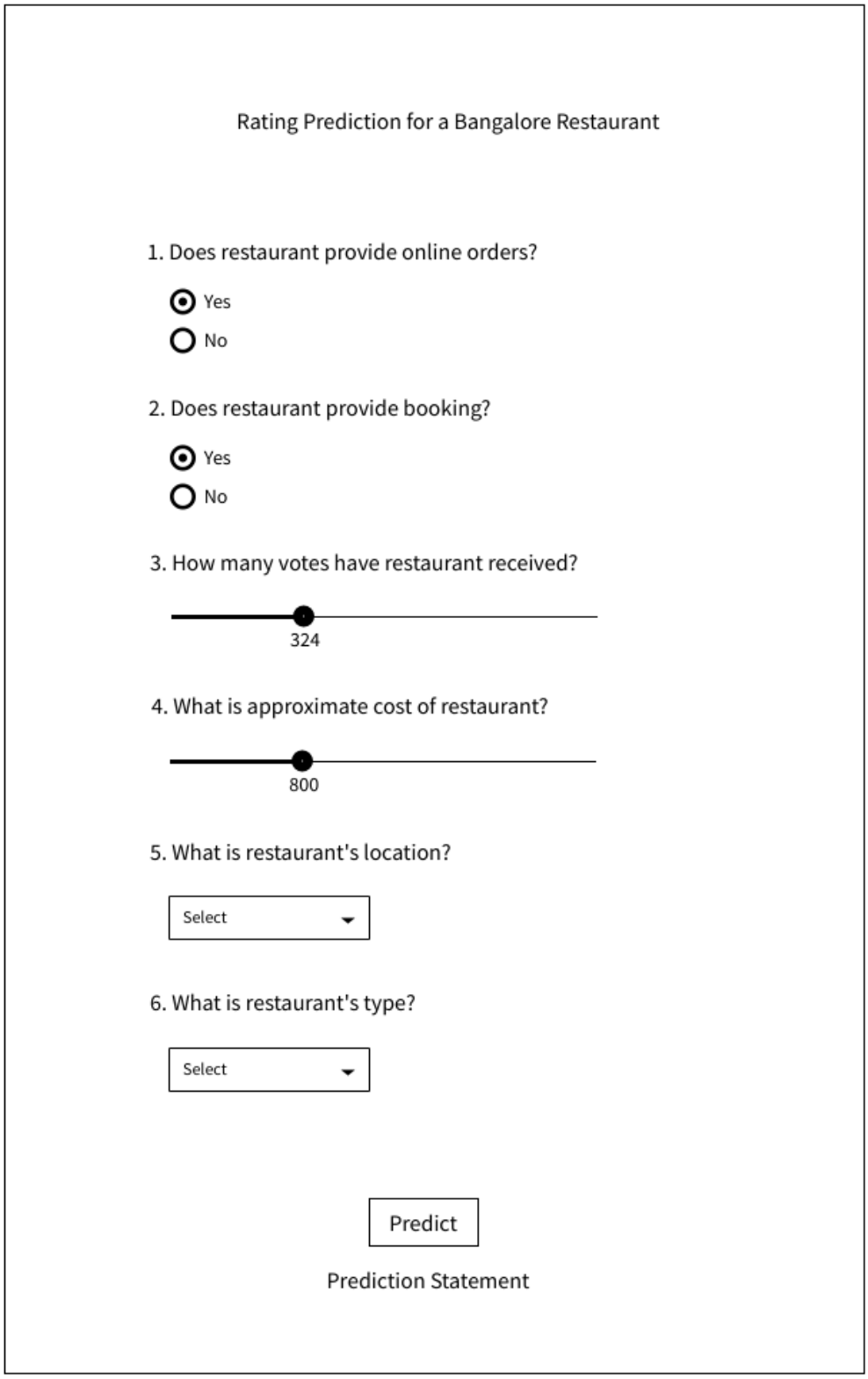

Fig. 2. Wireframe of the data science application.

2.3 Further Improvements

As the product is based on Bangalore restaurants, this can be extended to a number of cities. For deeper insights, the dataset from other platforms; like Swiggy or Dineout can be utilized.

2.4 System Environment

- **Programming Language:** Python 3.8.8 or later
- **Data Analysis:** Pandas
- **Logs:** Logzero
- **Configuration:** PyYAML
- **Directory Management:** Pathlib
- **Numerical Calculations:** Numpy, Scipy
- **Plots:** Matplotlib, Seaborn, Plotly
- **Model Training:** Scikit Learn, XGBoost, LightGBM, CatBoost
- **Application Interface:** Streamlit
- **Deployment:** Heroku

2.5 Constraints

Previously discussed application wireframe; depicts a suitable layout and describes necessary inputs. Therefore, referring to the wireframe will prevent any oversight. Given a number of user inputs, the application should be prefilled with usual values. This will be convenient for the user.

3. Design Details

The main constituents of the project are discussed in the following subsections.

3.1 Data Acquisition

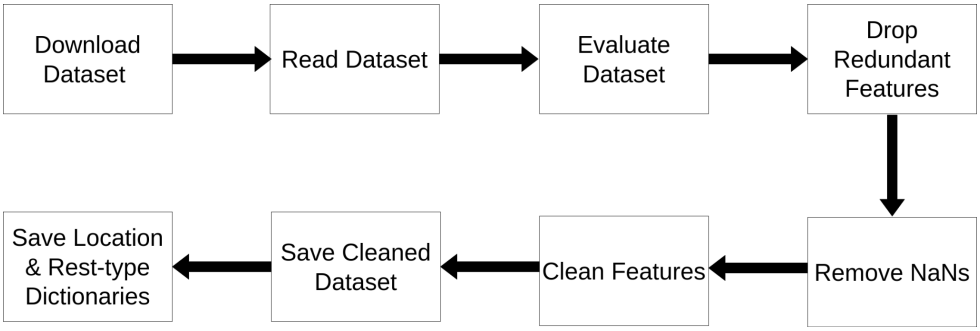

Fig. 3. The scheme of data acquisition.

Data acquisition process is described in figure 3. "**get_data.py"** script acquires the data. The script downloads the dataset using google drive link. The dataset features are evaluated. Redundant features and NaNs are removed. The features requiring cleaning are identified and subsequently cleaned. Besides, locations and restaurant type dictionaries are also saved. These dictionaries are later used by the Streamlit app.

3.2 Data Visualization

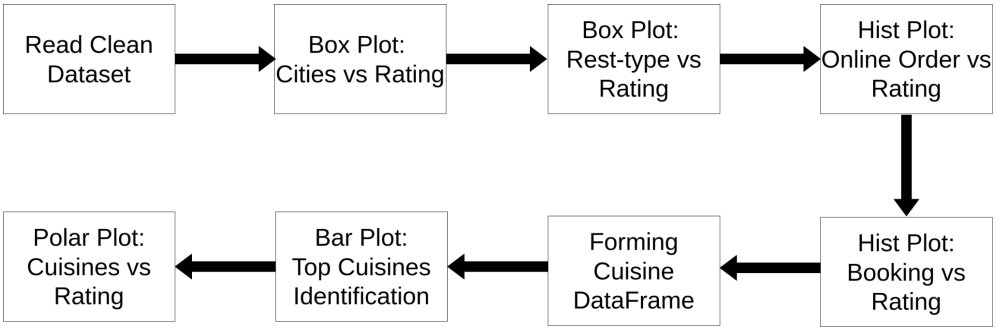

Fig. 4. The scheme of data visualization.

The method of data visualization is described in figure 5. Data visualization is executed using "**data_analysis.py"** script. Functions to form box plots, histograms, bar plots and polar plots are declared in "**data_analysis_util.py"**. Using these functions different

respective visualizations are obtained. Also, these visualizations are saved in **“visualizations”** directory.

3.3 Data Preparation

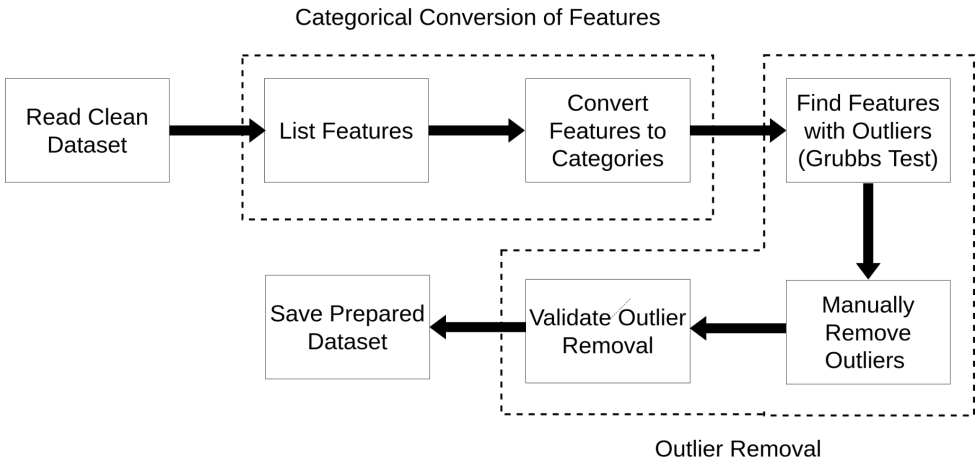

Fig. 5. The scheme of data preparation.

The data preparation process is depicted in figure 5. Data is prepared using the **“prepare_data.py”** script. Features requiring categorical conversion are listed and respectively converted. Subsequent outliers in features are identified using Grubbs Test. Manually, these outliers are removed and validated. Prepared dataset is saved.

3.4 Data Splitting

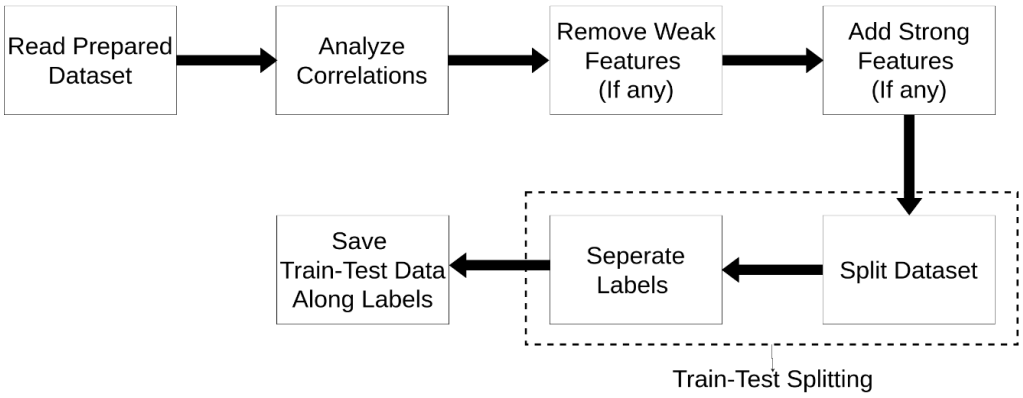

Fig. 6. The scheme of data splitting.

Data splitting is accomplished using the "**split_data.py"** script. Figure 6 depicts the script processing. The script reads the clean dataset. Feature correlations are analyzed to determine weak and strong features. Accordingly, features can be dropped or new features can be added. Further, the dataset is split using stratified sampling. This ensures the fair splitting. Labels are separated from train and test sets. Train-test data along features is saved.

3.5 Modelling

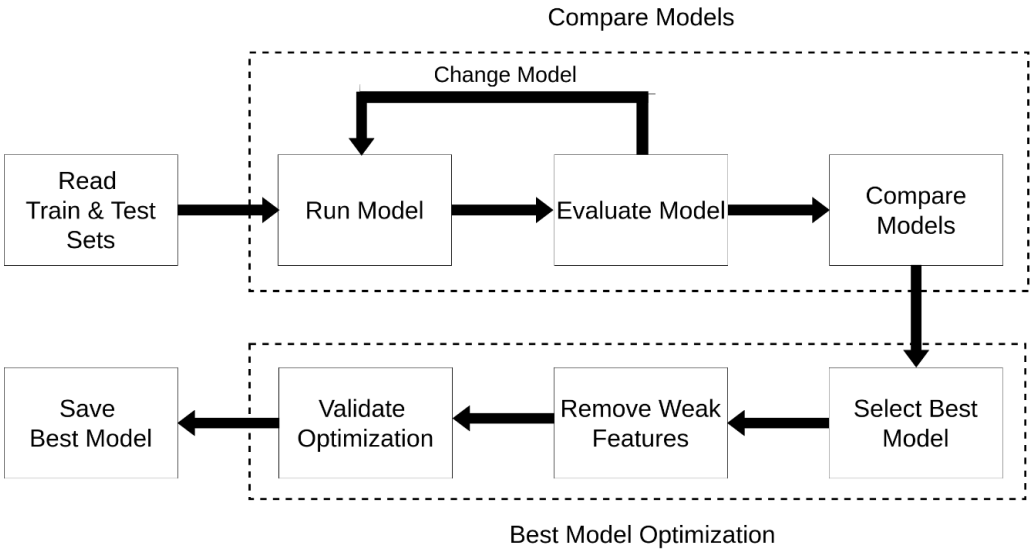

Fig. 7. The process of modelling.

The process of modelling is described in figure 7. The script **"model_data.py"** models the dataframe. This script reads train and test sets and runs a number of models. Each model is evaluated to compare different models. Based on accuracy, the best model is selected. Weak features in the model are identified and eliminated. Improved model performance is validated to finally save the best model.

3.6 Application Development

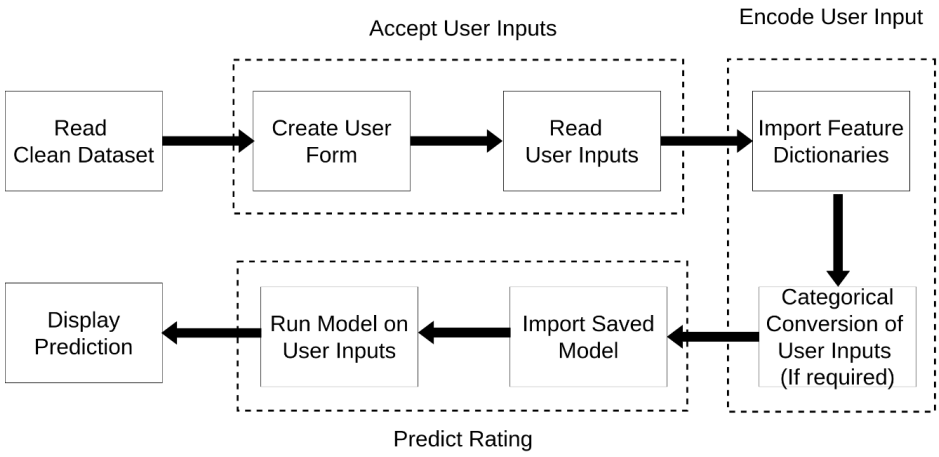

Fig. 8. The process of application development.

To provide user interaction a Streamlit app is developed. This app accepts six user inputs; to predict the restaurant rating. The app is developed using "**app.py"** script. The script reads the clean dataset. To accept the user inputs; a user form is created. The dataset is used to extract user input options like locations. The received user inputs are converted to categorical variables. Features with many categories use dictionaries for categorical conversion. The model obtained from the previous step is imported. User inputs are fed to the model to obtain restaurant rating prediction. This prediction is displayed in the application.

3.7 Utility Scripts

"**get_data_util.py", "data_analysis_util.py", "prepare_data_util.py", "split_data_util.py", "model_data_util.py"** and "**utility.py"** declare vital functions that are required by respective scripts.

Besides, **"run_project.py"** file runs all the project scripts sequentially (including application). Therefore, the entire project is executed with this script. This is shown in figure 9.

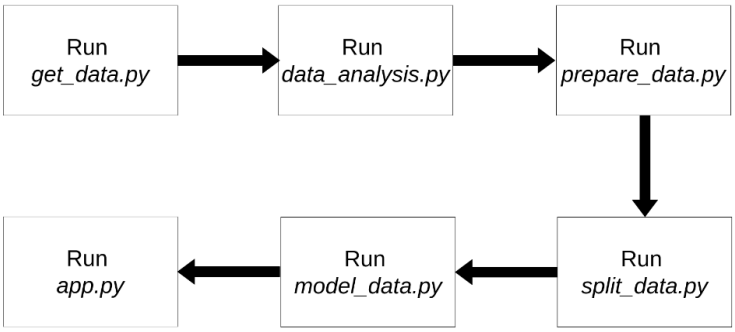

Fig. 9. The operation of "run_project.py".

4. Project Execution

4.1 Install Dependencies

Foremost running the project, installing the dependencies is essential.

- Ensure Python 3.8.8 or later is installed in the system.
- All required libraries are listed in "requirements.txt". These are easily installed; by running the following command in project directory

pip install -r requirements.txt

4.2 Run Project

"src" directory possesses the main scripts. Running the following command in the **"src"** directory executes the entire project

python3 run_project.py

Alternatively, **"src"** and **"app"** directory contain the main project scripts. Any individual project script (say "script.py") can be separately executed using the following command

python3 script.py