Zomato Restaurant Analysis And Predict Their Ratings

12.06.2024

Lavish Gangwani

Document Control

Date	Version	Description	Author
12/06/2024	1.0	Abstract, Introduction, General Description	Lavish Gangwani
14/06/2024	1.1	Design Detail, API Deployment	Lavish Gangwani
17/06/2024	1.2	Final Revision	Lavish Gangwani

Abstract

The objective of this project is to perform extensive Exploratory Data Analysis (EDA) on the Zomato dataset and build a predictive Machine Learning model to estimate restaurant ratings based on specific features. This will help Zomato restaurants anticipate their ratings and take strategic actions to enhance customer satisfaction and overall performance. The approach includes data exploration, data cleaning, feature engineering, model building, hyperparameter tuning, and model testing. The Random Forest Regressor achieved the highest predictive accuracy, providing a robust framework for predicting restaurant ratings.

Introduction

1.1 Purpose of High-Level Design Document

The purpose of this High-Level Design (HLD) Document is to add necessary detail to the project description to represent a suitable model for coding. It helps detect contradictions prior to coding

and can be used as a reference manual for module interactions at a high level.

1.2 Scope

The HLD documentation presents the structure of the system, such as database architecture, application architecture (layers), application flow (Navigation), and technology architecture. It uses non-technical to mildly-technical terms understandable to the system administrators. It includes detailed descriptions of user interfaces, hardware and software interfaces, performance requirements, design features, and non-functional attributes like security, reliability, maintainability, portability, application compatibility, and resource utilization.

General Description

2.1 Product Perspective & Problem Statement

The goal of this project is to analyze restaurants in Bangalore and predict their ratings using a dataset containing information on 70-80 thousand restaurants. The problem involves calculating the ratings of individual restaurants based on provided data.

2.2 Tools Used

- Python: Programming language for model building and API development.
- Jupyter Notebooks: For exploratory data analysis and model development.
- Pandas, NumPy: Libraries for data manipulation and numerical operations.
- Scikit-Learn: Machine learning library for model building.
- FastAPI: Framework for building APIs.
- SQLite: Database for storing processed data.
- Google Colab: For collaborative code development and testing.

Design Detail

3.1 Functional Architecture

The functional architecture includes the following components:

- Data Ingestion: Collect data from Zomato's dataset.
- Data Cleaning: Handle missing values, correct inconsistencies, and prepare the dataset for modeling.
- Feature Engineering: Transform raw data into meaningful features.
- Model Building: Train various machine learning models (Random Forest, Linear Regression, etc.).
- Model Evaluation: Evaluate models using appropriate metrics to determine the best fit.
- API Development: Develop an API using FastAPI to serve the model predictions.
- Frontend Interface: Simple HTML form for inputting restaurant data and displaying predictions.

3.2 Optimization

Optimization strategies include:

- Data Strategy: Minimize the number of fields and records, optimize extracts to speed up future queries.
- Filter Optimization: Limit the number of filters and use include filters, continuous date filters, Boolean or numeric filters.
- Calculation Optimization: Perform calculations in the database, reduce nested calculations, use MIN or MAX instead of AVG, make groups with calculations, use Booleans or numeric calculations.

Key Performance Indicators (KPIs)

- Restaurant Rating Summary: Displaying a summary of restaurant ratings and their relationship with different metrics.

- Online Table Bookings: Percentage of people booking tables online or offline.
- Location Metrics: Location and neighborhood of restaurants.
- Online Orders: Whether restaurants accept online orders.
- Popular Dishes: Most liked dishes of the restaurants.
- Cuisines: Types of cuisines offered by the restaurants.

Deployment

FastAPI for API Deployment: Using FastAPI to build and deploy APIs for model predictions. FastAPI offers high performance, rapid development speed, fewer bugs, intuitive design, and ease of use. It reduces code duplication and supports efficient feature development.

Deployment Steps:

1. Setup Environment: Configure the Python environment and install required libraries.
2. Develop API: Implement endpoints for predictions using FastAPI.
3. Testing: Test the API endpoints to ensure correct functionality.
4. Deployment: Deploy the API to a cloud service (e.g., AWS, Google Cloud) for scalability and availability.
5. Monitor and Maintain: Monitor the API for performance and maintain it for any necessary updates or bug fixes.