

Praveen Kumar

+91 7569993454

<mailto:codethebug3@gmail.com>

**Restaurant Rating Prediction**

Low Level Design

Domain: Machine Learning

Creator: Praveen Kumar

Date: 17.07.2023

|  |  |  |  |
| --- | --- | --- | --- |
| Date issued | Version | Description | Author |
| July 17, 2023 | 1.1 | First Draft | Praveen Kumar |
| July 20, 2023 | 1.2 | Added unit test cases | Praveen Kumar |

**Document Version Control**

Contents

[Introduction 3](#_Toc110433758)

[What is Low-Level Design Document? 3](#_Toc110433759)

[Scope 3](#_Toc110433760)

[Architecture 4](#_Toc110433761)

[Architecture Description 5](#_Toc110433762)

[Data Preparation 5](#_Toc110433763)

[Data Description 5](#_Toc110433764)

[Data Preprocessing 5](#_Toc110433765)

[Exploratory Data Analysis 5](#_Toc110433766)

[Feature Engineering 5](#_Toc110433767)

[Model Development 6](#_Toc110433768)

[Model implementation 6](#_Toc110433769)

[Hyper-parameter Tuning 6](#_Toc110433770)

[Model Evaluation 6](#_Toc110433771)

[Deployment 6](#_Toc110433772)

[Designing UI with Anvil 6](#_Toc110433773)

[Designing a server 6](#_Toc110433774)

[Code deployment on cloud 7](#_Toc110433775)

[Deployment Process 7](#_Toc110433776)

[Unit cases 8](#_Toc110433777)

# Introduction

## What is Low-Level Design Document?

The goal of LLD or a low-level design document is to give the internal logical of the actual program code for Metro Interstate Traffic Volume Prediction. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

The main objective of the project is to predict if traffic volume is in high or low on particular date. Weather circumstance, special days like holidays, daytime (morning, afternoon, night and etc.), a temperature, a weekday, a numeric percentage of cloud cover are vital attributes for predicting traffic volume.

## Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

# Architecture

**Data Preparation**

**Model**

**Development**

**Deployment**

**Deployment**

# Architecture Description

## Data Preparation

### Data Description

### I was always fascinated by the food culture of Bengaluru. Restaurants from all over the world can be found here in Bengaluru. From United States to Japan, Russia to Antarctica, you get all type of cuisines here. Delivery, Dine-out, Pubs, Bars, Drinks, Buffet, Desserts, you name it and Bengaluru has it. Bengaluru is best place for foodies. The number of restaurants are increasing day by day. Currently which 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.

### Exploratory Data Analysis

This step includes bivariate and univariate analysis of features. Checking outliers using boxplots, and outlier treatment is carried out as well. Distribution of numerical values is plotted to see to what extent our data is skewed.

### Feature Engineering

In this part, compute aggregate statistics from existing features to provide a summary view of the data. For instance, calculate features like average rating per cuisine, average price range per location, or the number of restaurants in a specific category within a certain radius. If available, include features that capture user interactions with the Zomato platform. This might include features such as the number of reviews a user has submitted, average rating given by the user, or the total number of votes received for their reviews.

## Model Development

### Model implementation

After train and test splitting, pipeline containing Standard Scaler and Ordinal Encoder was fitted to several models such as Linear Regression, AdaBoost Regressor, Gradient Boosting Regressor, Random Forest Regressor. Their R2 score were obtained. and it was determined that Random Forest Regressor and Extra Tree Regressor performs better than other models.

### Hyper-parameter Tuning

The best model is chosen, and Grid Search with Cross Validation is applied on that model to get the best parameters. Those parameters are then used on the model to get better result.

### Model Evaluation

Test dataset is used to evaluate the model. 20% of dataset was separated for testing. Predicted results of the model are compared with the actual data to check the amount of error. As there was no considerable change after hyperparameter tuning, it helped us to overcome overfitting and perform better on new data.

## Deployment

### Designing UI with Anvil

For this project, a user interface is built on Anvil. It is a web application that helps us to create applications for projects. It is a free Python-based drag-and-drop web app builder.

### Designing a server

A server should be created to run the UI application continuously. Flask server is built, and it is linked with Anvil uplink that connects Anvil UI with our server.

### Code deployment on cloud

The codes for this machine learning model should be deployed to the cloud, so that when data is entered into the application, our code runs, and a user gets the result online.

## Deployment Process

In this stage, we establish a server using Flask that runs the uplink code (server code) in parallel before developing the UI using Anvil and connecting with our code, where our model is executing, via an uplink. We will post the hole after execution or asynchronous execution. Git and GitHub are used to code in the Heroku cloud. Then, we'll configure a cron job to maintain the server and server code in operation indefinitely.

# Unit cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application URL is  accessible to the user | Application URL should be defined | Application URL should be accessible to the user |
| Verify whether the Application loads 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 whether user is able to see input fields. | Application is accessible | User should be able to see input fields |
| Verify whether user is able to edit all input fields | Application is accessible | User should be able to edit all input fields |
| Verify whether user gets Submit button to submit the inputs | Application is accessible | User should get Submit button to submit the inputs |
| Verify whether user is presented with results on clicking submit | Application is accessible | User should be presented with results on clicking submit |
| Verify whether the results are in accordance to the selections user made | Application is accessible | The results should be in accordance to the selections user made |