Metro Interstate Traffic Volume Prediction System

Low-Level Design Documentation

- ***Fazal Rehman***

Contents

Abstract

The Metro Interstate Traffic Volume Prediction System aims to build a machine learning model that can accurately predict traffic volume on interstate highways based on various factors such as weather conditions, time, and holidays. By forecasting traffic volume, city planners and individuals can make informed decisions to reduce congestion and improve commuting experiences. The dataset used for this project has been sourced from the UCI Machine Learning Repository and contains traffic volume recorded hourly, along with corresponding weather and time features.

Introduction

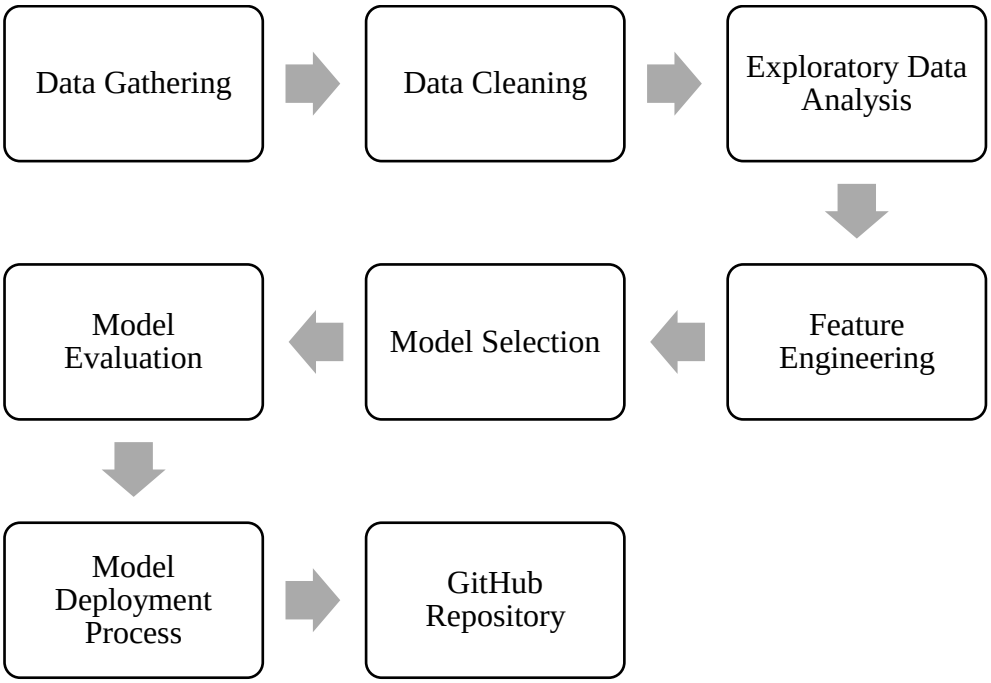
What is Low-Level Design?

The goal of LLD or a low-level design document is to give the internal logical design of the actual program code for the Metro Interstate Traffic Volume Prediction System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so the programmer can directly code the program from the document.

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 Gathering

- Download the dataset from the UCI Repository.

Data Cleaning

- Handle missing values using SimpleImputer.
- Standardize numeric columns.

Exploratory Data Analysis

- Analyse correlations.
- Identify significant features.

Feature Engineering

- Encoding for categorical variables using OrdinalEncoder.
- Scaling for numerical variables using StandardScaler.

Model Selection

- Train multiple models like Random Forest, CatBoost, etc.
- Select the model based on the R2 score.

Model Evaluation

- Evaluate selected model with R2-score.

Model Deployment Process

- Save trained model as a .pkl file.
- Build a Flask server for model interaction.
- Serve prediction results through frontend forms.

GitHub Repository

- Push the entire project, including model, code, and documentation, to GitHub.

Unit Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the application loads completely for the user when the URL is accessed.	The application URL is accessible, and the application is deployed.	The application should load completely for the user when the URL is accessed.
Verify whether the user can see input fields.	The application is accessible.	The user should be able to see input fields.
Verify whether the user can edit all input fields.	The application is accessible.	The user should be able to edit all input fields.
Verify whether the user gets the Submit button to submit the inputs.	The application is accessible.	The user gets a Submit button to submit the inputs.
Verify whether the user is presented with results on clicking submit.	The application is accessible.	The user should be presented with the results on clicking submit.
Verify whether the results are as per the selections the user made.	The application is accessible.	The results should be as per the selections the user made.