HARDIKKUMAR SUTARIYA

Low Level Design (LLD)

Metro Interstate Traffic Prediction

Contents

Document Version Control 2

1. Introduction 4

    1.1 Why this High-Level Design Document? 4

    1.2 Scope… 4

2. Architecture 5

3. Architecture Description 6

    2.1 Data Description 6

    2.2 Source Data 6

    2.3 Data Insertion into database 6

    2.4 Load data from database 6

    2.5 Data Validation 6

    2.6 Data Transformation 7

    2.7 Splitting data into training and testing data 7

    2.8 Model Training 7

    2.9 Model Evaluation 7

    2.10 Hyperparameter Tuning 7

    2.11 DVC 7

    2.12 MLFlow 7

    2.13 Deployment 8

4. Unit Test Cases 9

1 Introduction

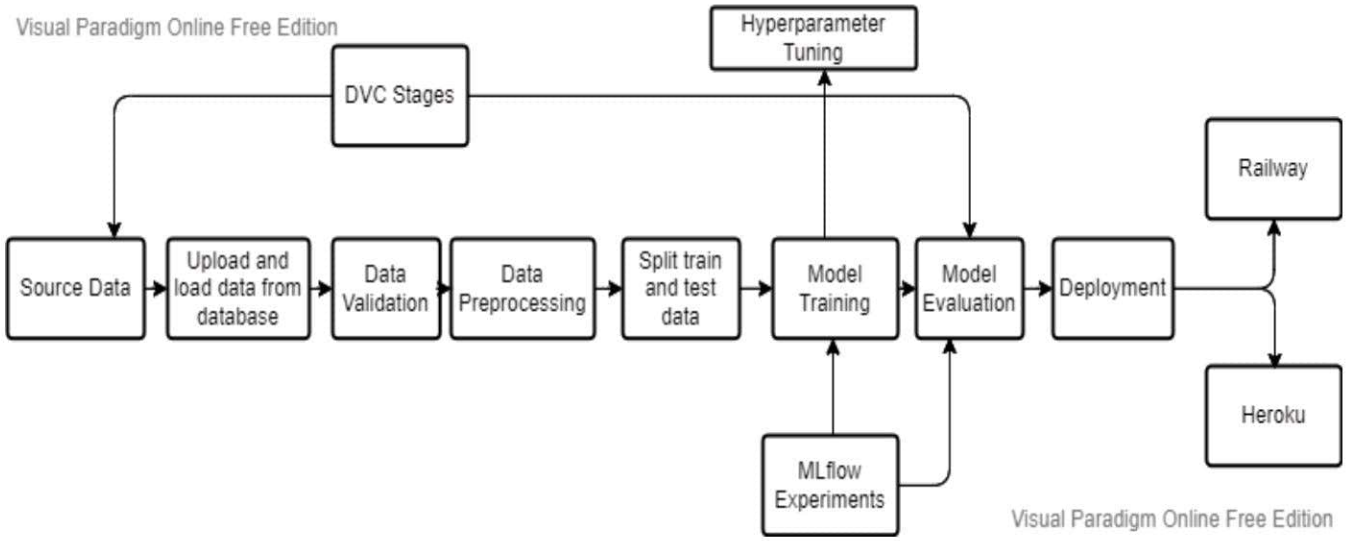
1.1 Why this Low-Level Design Document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2 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.

2 Architecture



3 Architecture Description

3.1 Data Description

The dataset used in this project is a UCI machine learning dataset. The dataset contains hourly interstate 94 westbound traffic volume for MN DoT ATR station 301. The region of data lies between regions of Minneapolis. It includes features such as holiday, time, weather, etc. which impacts the traffic volume traffic volume directly.

Information of attributes of dataset as follows:
holiday: Indicates if the date is a holiday and if it specifies the holiday, if not None.
temp: Indicates the temperature in Kelvin.
rain_1h: Amount in mm of rain that occurred in the hour.
snow_1h: Amount in mm of snow that occurred in the hour.
clouds_all: Percentage of cloud cover.
weather_main: Short textual description of the current weather.
weather_description: Longer textual description of the current weather.
date_time: Hour of the data collected in local CST time.
traffic_volume: Hourly I-94 ATR 301 reported westbound traffic volume.

3.2 Source Data

The source data used for this project is collected through UCI Machine Learning Repository - https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume

3.3 Data Insertion into database

MongoDB database is used for this project as the primary source of storing and saving the data. get_data_from_database.py file is responsible for uploading and loading the data.

3.4 Loading data from database

After the source data is uploaded into database, data is loaded from the database and is stored in the raw data path.

3.5 Data Validation

Validating the data is an important part of the machine learning pipeline.
Data is validated with respect to no. of columns, no. of rows, column names and its data types. If validation is successful, the data is further processed for transformation, otherwise, the stage is failed.

3.6 Data Transformation

Once data is validated, the data is passed for preprocessing. Operations such as handing null values, outliers removal, feature importance, encoding, etc. and other structuring and cleaning process is performed here. The model ready data is then transferred into processed data path for model training.

3.7 Splitting data into training and testing data

Before model training, the data is separated into training and testing data. For our model, the data split is 75% Training data and 25% Testing data.

3.8 Model Training

As we are solving a regression problem, after testing all the regression models we found that XGBoost Regressor was the best algorithm for our project. Through hyperparameter tuning, we tried to achieve best parameters and contribute to model retraining with continuously improved parameters. MLFlow experiment tracking is carried out at the same time for model experimentation.

3.9 Model Evaluation

Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R2 score are the metric systems used for model evaluation and experiment. Detailed report for each experiment metrics can be experimented using MLFlow.

3.10 Hyperparameter Tuning

To achieve the most accurate model, it is necessary to tune the model parameters continuously for improvised results. RandomSearchCV, GridSearchCV or any other type of cross validation can be performed with estimated model.

3.11 DVC

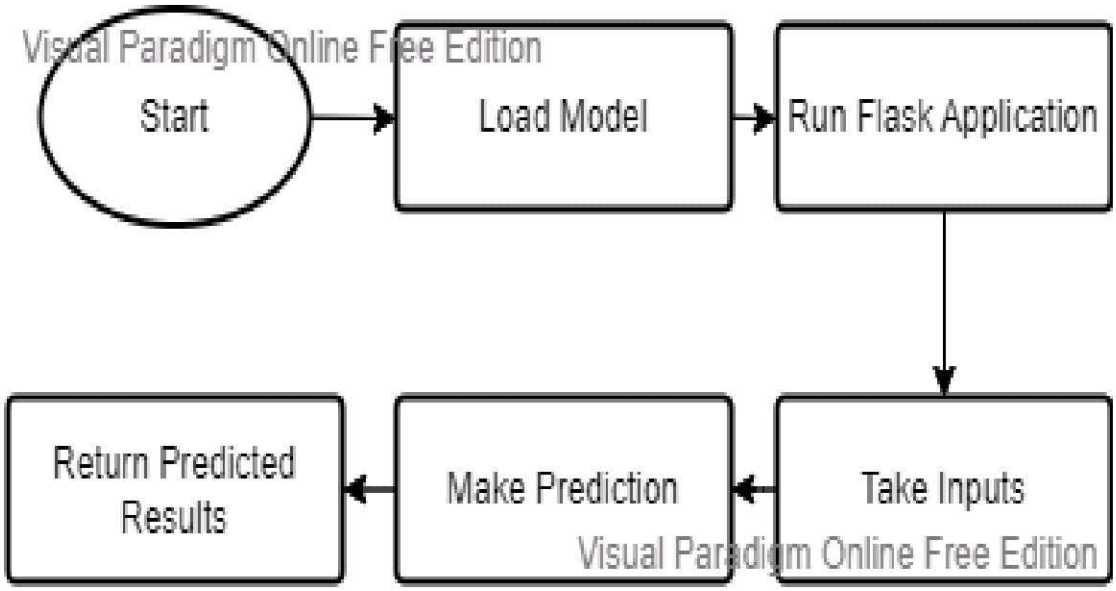
DVC, which goes by Data Version Control, is essentially an experiment management tool for ML projects. DVC represents a complete machine learning pipeline with the help of different DVC stages. Stages, dependencies, parameters, metrics and outputs are defined in the dvc.yaml file to execute the DVC.

3.12 MLFlow

MLflow is a platform to streamline machine learning development, including tracking experiments, packaging code into reproducible runs, and sharing and deploying models. Parameters and metrics to log are defined while model training stage. Experimentating, model comparision, model version stages, etc. are some of the operation we can perform using MLFlow. Model retraining becomes possible using MLFlow.

3.13 Deployment

This project is deployed on cloud platforms such as Railway, Heroku and Render. Below is the deployment flow chart:



4 Unit Test Cases

Test Case Description	Pre-Requisite	Expected Results
Verify whether the Application URL is accessible to the user	1. 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 the user is able to see input fields	1. Application is accessible 2. Application is deployed	User should be able to see input fields once application is loaded
Verify whether the user is able to edit all input fields	1. Application is accessible 2. Application is deployed	User should be able to edit all input fields
Verify all the fields are in domain feature range.	1. Application is accessible 2. Application is Deployed 3. All fields in the form are filled	If values not in range, error page loads. Else, result is printed.
Verify whether user gets Submit button to submit the inputs	1. Application is accessible 2. Application is Deployed 3. All fields in the form are filled	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible 2. Application is Deployed 3. All fields in the form are filled	User should be presented with recommended results on clicking submit
Verify weather api response is working or not.	1. Application is accessible 2. Application is Deployed 3. Data is provided to the api request in json format.	User should get the result of the api response prediction as a dictionary.