Low Level Design (LLD)

Share Bike Demand Prediction

WRITTEN BY
SADASHIV NANDANIKAR

Revision Number: 1.0

Last data of revision: 13/03/202

Contents

1. Introduction

1.1 What is Low-Level design document?

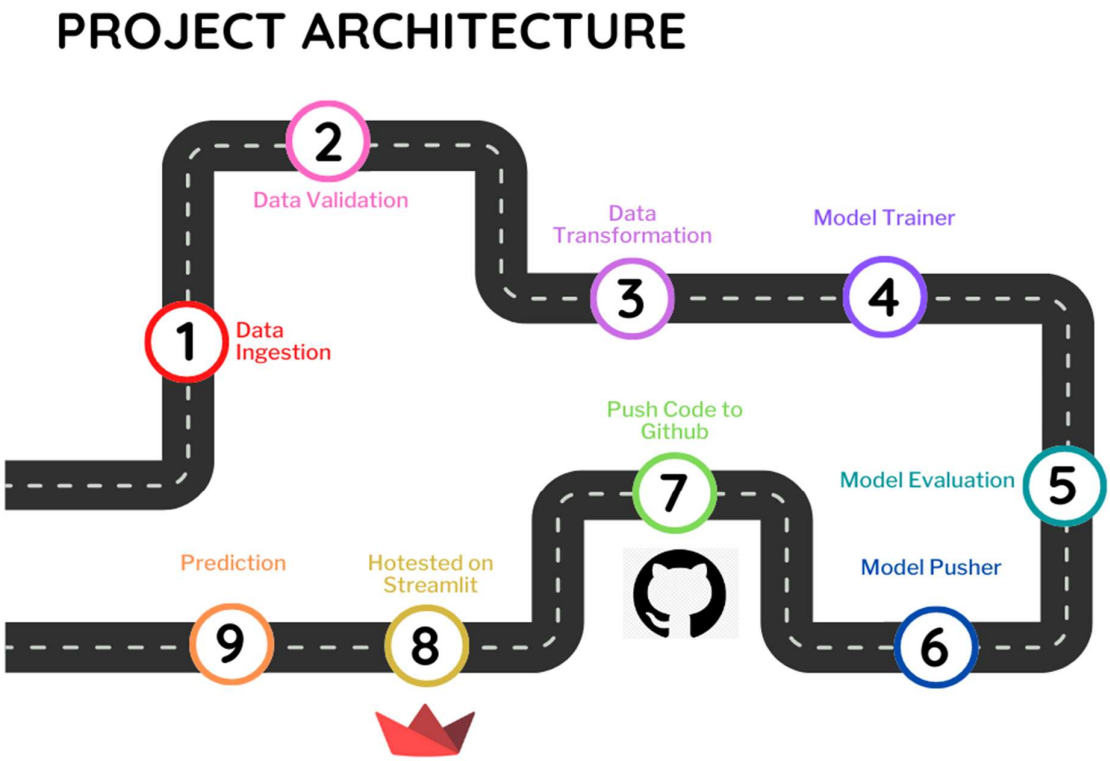
A low-level design document (LLD) is a detailed technical document that describes how a software system or application will be implemented at a low level. The purpose of an LLD is to provide a blueprint for developers and stakeholders to understand the technical details of the system, including the architecture, components, modules, interfaces, algorithms, data structures, and other implementation details.

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

2.1 Project Architecture:



3. Architecture Description

3.1 Data Gathering:

The current project utilizes data sourced from The UCI Machine Learning Repository.

[UCI Machine Learning Repository: Bike Sharing Dataset Data Set]

3.2 Data Description:

Bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather conditions, precipitation, day of week, season, hour of the day, etc. can affect the rental behaviours. The core data set is related to the two-year historical log corresponding to years 2011 and 2012 from Capital Bikeshare system, Washington D.C., USA which is publicly available in http://capitalbikeshare.com/system-data. We aggregated the data on two hourly and daily basis and then extracted and added the corresponding weather and seasonal information. Weather information are extracted from http://www.freemeteo.com.

3.3 Tool Used:

- The environment was created using Python 3.8.10.
- The data is stored in a MongoDB database.
- VS Code is used as IDE.
- GitHub serves as the code repository.
- The Streamlit application is utilized to host the application.

Other libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-Learn were utilized in building the whole model.

3.4 Insert data to MongoDB database:

The data contains a large number of records and has been stored in a MongoDB database to enable efficient storage and retrieval.

3.5 Data Transformation:

- Duplicate records have been eliminated from the data.
- Outliers have been removed from the dataset.
- Categorical variables have been encoded using OneHotEncoder.
- Numerical features have been transformed.

3.6 Model Building:

The pre-processed data has been fed into different machine learning regression algorithms, and their performance has been evaluated using r2-scores. Among the given regression algorithms, the LightGBM regressor outperforms the others.

3.7 Model Evaluation:

The currently trained model is evaluated against any previously saved models, if they exist. If the current model is found to perform better than the saved model, the current model and transformer objects are saved for future use.

3.8 Model Pusher:

If the currently trained model outperforms the previous model, it will be stored in the "saved models" directory.

3.9 Prediction:

To generate predictions, the Streamlit application requires user inputs, which can then be displayed on the screen as output.

3.10 Deployment:

The application has been deployed using the Streamlit framework.