BIKE SHARING DEMAND PREDICTION

Low Level Design

09/11/2024

Ritik Patel

1 Introduction:

The low-level design document provides a detailed overview of the bike sharing demand prediction project. It outlines the architecture, data flow, algorithms, and implementation specifics of the system. This document serves as a guide for developers and stakeholders involved in the project, ensuring a comprehensive understanding of the design.

1.1 What is Low-Level Design Document?

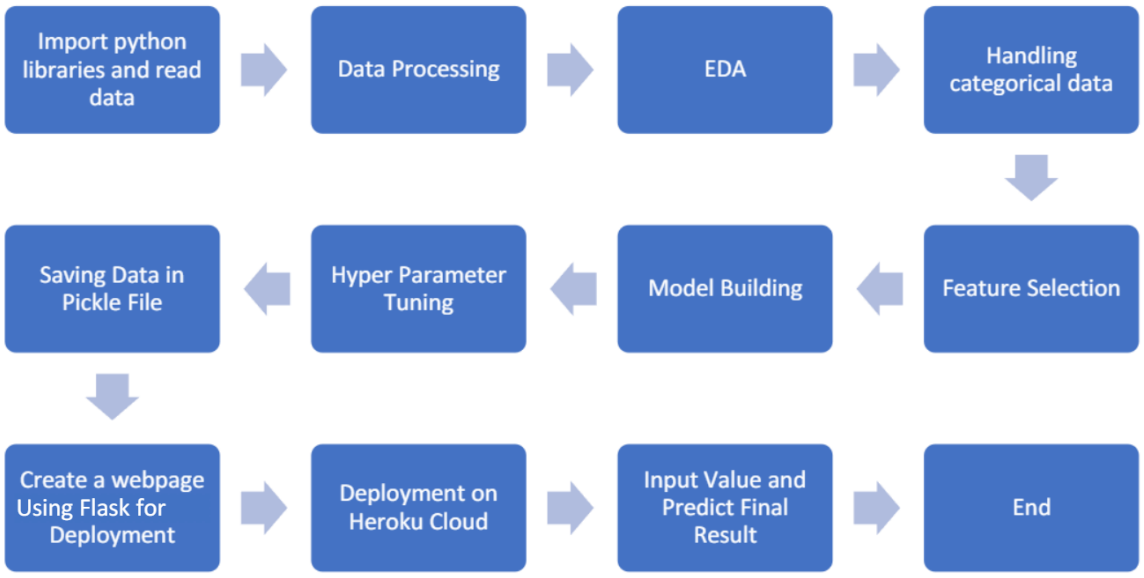
A low-level design document is a comprehensive document that describes the detailed design aspects of a project. It includes information about the system's architecture, components, data flow, algorithms, and specific implementation details. This document acts as a blueprint for developers, guiding them in the development and implementation process.

1.2 Scope:

The scope of the bike sharing demand prediction project is to accurately forecast the number of bikes required at each hour, considering factors such as season, year, month, hour, weather conditions, temperature, and humidity. The project aims to develop a reliable prediction model and provide insights for efficient bike availability and management.

2 Architecture:

The architecture of the bike sharing demand prediction system follows a modular and scalable approach. It consists of several key components, including data gathering, data preprocessing, model building, user interaction, data validation, result rendering, and deployment. These components work together to enable accurate demand prediction and enhance the availability of rental bikes.



3 Architecture Description:

3.1 Data Gathering:

The project utilizes historical bike rental data, weather data, and other relevant data sources for analysis and prediction. Data gathering involves the collection of these datasets, either through API integration or manual data acquisition methods. The collected data serves as the foundation for building the prediction model.

3.2 Data Description:

The dataset used in the project contains attributes such as

- Dteday
- Season
- Yr
- mnth
- hr
- holiday
- weekday
- workingday
- weathersit
- temp
- atemp
- hum
- windspeed
- casual
- registered
- cnt.

These attributes provide crucial information about temporal factors, weather conditions, and bike usage patterns, enabling accurate prediction of bike demand.

3.3 Tool Used:

- The project leverages Python as the primary programming language for its flexibility, extensive libraries, and machine learning frameworks.
- Popular libraries and tools such as scikit-learn,Flask, are employed for data preprocessing, model building, and deployment.

3.4 Data Pre-processing:

Data pre-processing involves various steps, including handling missing values, outlier detection and treatment, feature engineering, and data normalization. These steps ensure that the data is in a suitable format for modelling, minimizing the impact of noise or inconsistencies in the dataset.

3.5 Model Building:

The project employs machine learning algorithms, such as

- Linear Regression
- Random Forest
- Extra Trees Regressor
- LightGBM

to build the prediction model. The models are trained on the pre-processed data, and techniques like cross-validation and hyperparameter optimization are utilized to enhance performance and generalization capabilities.

3.6 Data from User:

The system allows users to provide additional data or specific requirements to tailor the prediction according to their needs. User input, such as special events, holidays, or specific weather conditions, can be incorporated into the prediction model to fine-tune the results.

3.7 Data Validation:

The model's performance and accuracy are evaluated through data validation techniques. Cross-validation, train-test splits, and evaluation metrics like mean squared error or R-squared are employed to assess the model's predictive capabilities and ensure its reliability.

3.8 Rendering Result:

The prediction results are rendered through interactive interfaces or visualizations created using flask. Users can access reports or visual representations that provide insights into the predicted bike demand for different hours, days, or weather conditions. This enables effective decision-making and resource allocation.

3.9 Deployment:

The deployment strategy involves setting up the prediction system on a web server or cloud platform such as Heroku or AWS using Flask. Considerations for scalability, security, and maintenance are taken into account to ensure the system can handle real-time prediction demands and accommodate future growth.

4.Unit Test Cases:

To ensure the quality and reliability of the bike sharing demand prediction project, a comprehensive set of unit test cases were developed. These test cases target individual units or components of the system to validate their functionality and behaviour. Here are some sample unit test cases that were implemented:

Test Case Description	Pre-Requisite	Expected Result
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 sign up in the application	1. Application is accessible	The User should be able to sign up in the application
Verify whether user is able to successfully login to the application	1. Application is accessible 2. User is signed up to the application	User should be able to successfully login to the application
Verify whether user is able to see input fields on logging in	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	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. User is signed up to the application 3. User is logged in to the application	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The recommended results should be in accordance to the selections user made