BIKE SHARING DEMAND PREDICTION

Low Level Design Document

July 14, 2024

Rishabh Barman

Table of Contents

1. Introduction

The Bike Share Prediction System aims to predict bike availability and demand across different stations in a city using machine learning models. This document provides a detailed description of the system's architecture, modules, data flow, and other critical components.

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.

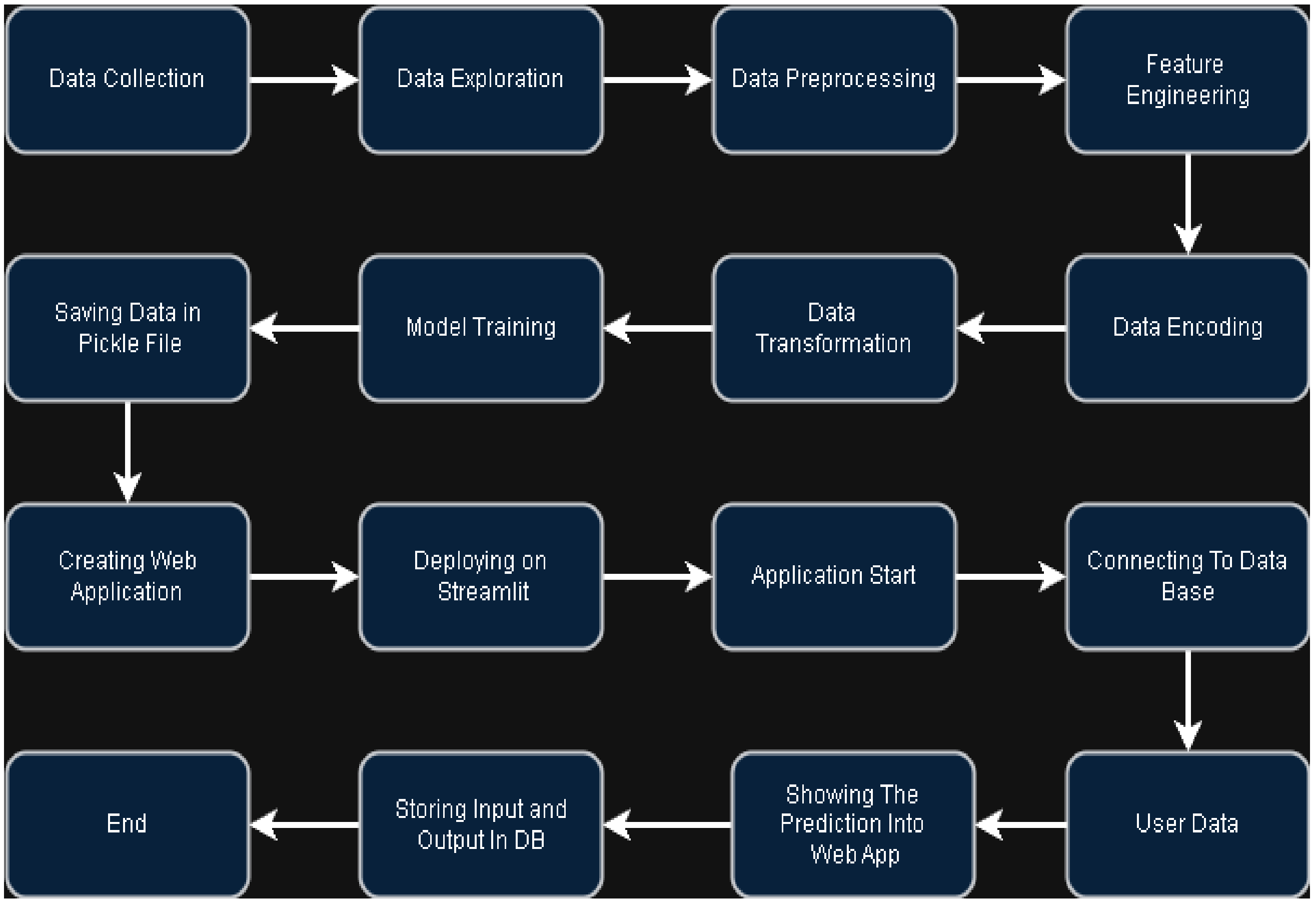
1.2 Scope

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

The diagram outlines the workflow for a bike share prediction system, illustrating each step from data collection to deployment and user interaction. It consists of several key components, including data gathering, data preprocessing, model building, user interaction, and deployment

Figure 1. Architecture Diagram



3. Architecture Description:

3.1 Data Collection:

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:

Table 1. Bike share metadata

Fields	Description
Season	1: winter, 2: spring, 3: summer, 4: fall
year	0: 2011, 1:2012
month	1 to 12
hour	0 to 23
holiday	1 = Holiday, 0 = Not a Holiday
weekday	day of the week
working day	1 = Neither a weekend nor holiday, 0 = Either a weekend or a holiday
weather	1 = Clear, Few clouds, partly cloudy, partly cloudy 2 = Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3 = Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4 = Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp	Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
atemp	Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-16, t_max=+50 (only in hourly scale)
humidity	Normalized humidity. The values are divided to 100 (max)
wind speed	Normalized wind speed. The values are divided to 67 (max)
casual	Number of non-registered user rentals initiated
registered	Number of registered user rentals initiated
count	Number of total rentals (casual + registered)

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 primary programming language used is Python due to its simplicity and extensive range of libraries.
- Libraries such as NumPy and Pandas are used for data manipulation and analysis, Scikit-learn for building and evaluating machine learning models, and Seaborn and Matplotlib for visualizations.

3.4 Data Exploration:

Analyzing the collected data involves understanding its structure, patterns, and anomalies. This process includes using descriptive statistics, visualizations, and correlations to gain insights into the data. Additionally, it helps in identifying trends, detecting outliers, and assessing data quality. This thorough exploration is crucial for informing subsequent steps such as preprocessing, feature engineering, and model selection

3.5 Data Preprocessing:

Cleaning the data involves handling missing values, outliers, and inconsistencies. This includes imputing missing values, removing duplicates, and normalizing data to ensure consistency and reliability. Additionally, data cleaning may involve filtering irrelevant data, correcting errors, and transforming data types. This step is essential to improve the accuracy and performance of the machine learning models.

3.6 Feature Engineering:

Creating new features or modifying existing ones to improve the model's predictive power.

3.7 Data Encoding:

Converting categorical data into numerical format suitable for machine learning algorithms.

3.8 Data Transformation:

Applying transformations to the data to improve model performance.

3.9 Model Training:

Training machine learning models using the processed data. Models used include:

- Linear Regression
- Lasso
- Ridge
- K-Neighbors Regressor
- Decision Tree
- Random Forest Regressor
- AdaBoost Regressor

3.10 Creating Web Application:

Developing a web interface to interact with the prediction system. This setup enables users to input data, receive predictions, and visualize results through a user-friendly interface.

3.11 Deploying on Streamlit:

Deploying the web application using Streamlit for an interactive user interface.

3.12 Application Start:

Initializing the web application involves setting up routes and loading necessary resources. This includes configuring the application to load the trained model and preprocessing steps when the server starts, ensuring that the prediction system is ready for user interactions.

3.13 Connecting to Database:

Setting up a connection to a database to store and retrieve user data and predictions involves connecting to DataStax to manage input and output data efficiently

3.14 User Data:

Handling user inputs required for making predictions involves capturing data such as date, time, weather conditions, etc., from the user via the web app

3.15 Storing Input and Output in DB:

Saving user inputs and prediction results in the database for future reference or analysis involves inserting records into a database table every time a prediction is made

3.16 End:

The final step indicates the completion of the prediction and interaction process. This involves returning control to the user after displaying the prediction and storing the data.

4. Unit Test Cases:

These test cases ensure that the different functionalities of the bike share prediction system are tested thoroughly, covering accessibility, user interactions, prediction accuracy, data storage, performance, and security.

Table 2. Test Cases

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 the User is able to sign up in the application	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 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 prediction 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 prediction results on clicking submit
Verify whether the prediction results are in accordance to the inputs user provided	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The prediction results should be in accordance with the inputs user provided
Verify whether data input and prediction results are stored in the database	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	Data input and prediction results should be stored in the database

5. Additional Test Cases

Table 3. Test Cases

Description	Pre-Requisite	Expected Result
Verify whether the system handles invalid user input	Application is accessible	System should display an appropriate error message and not crash
Verify the application's response time for predictions	1. Application is accessible 2. User is logged in to the application	Prediction response time should be within acceptable limits
Verify the application's behavior under high load	1. Application is accessible 2. Multiple users are logged in to the application	Application should handle concurrent users without significant performance degradation