BIKE SHARING DEMAND PREDICTION

Architecture Design Document

July 14, 2024

Rishabh Barman

Table of Contents

1. Introduction

1.1 Why this Architecture Design Document?

An Architecture Design Document (ADD) is a comprehensive guide that outlines the structure and components of a software system. It provides detailed descriptions of the system's architecture, including how various components interact, the technologies used, and considerations for performance, security, and scalability. An ADD is typically used to ensure that all stakeholders have a clear understanding of the system's design and to provide a reference for development and maintenance.

1.2 Purpose

The purpose of this document is to provide a detailed architecture design for the bike share prediction system. This document serves as a guide for developers, testers, and other stakeholders involved in the project.

1.3 Scope

This document covers the architecture design of the entire system, including data collection, preprocessing, model building, user interaction, and deployment.

1.4 Definitions, Acronyms, and Abbreviations

- API: Application Programming Interface
- ML: Machine Learning
- ADD: Architecture Design Document

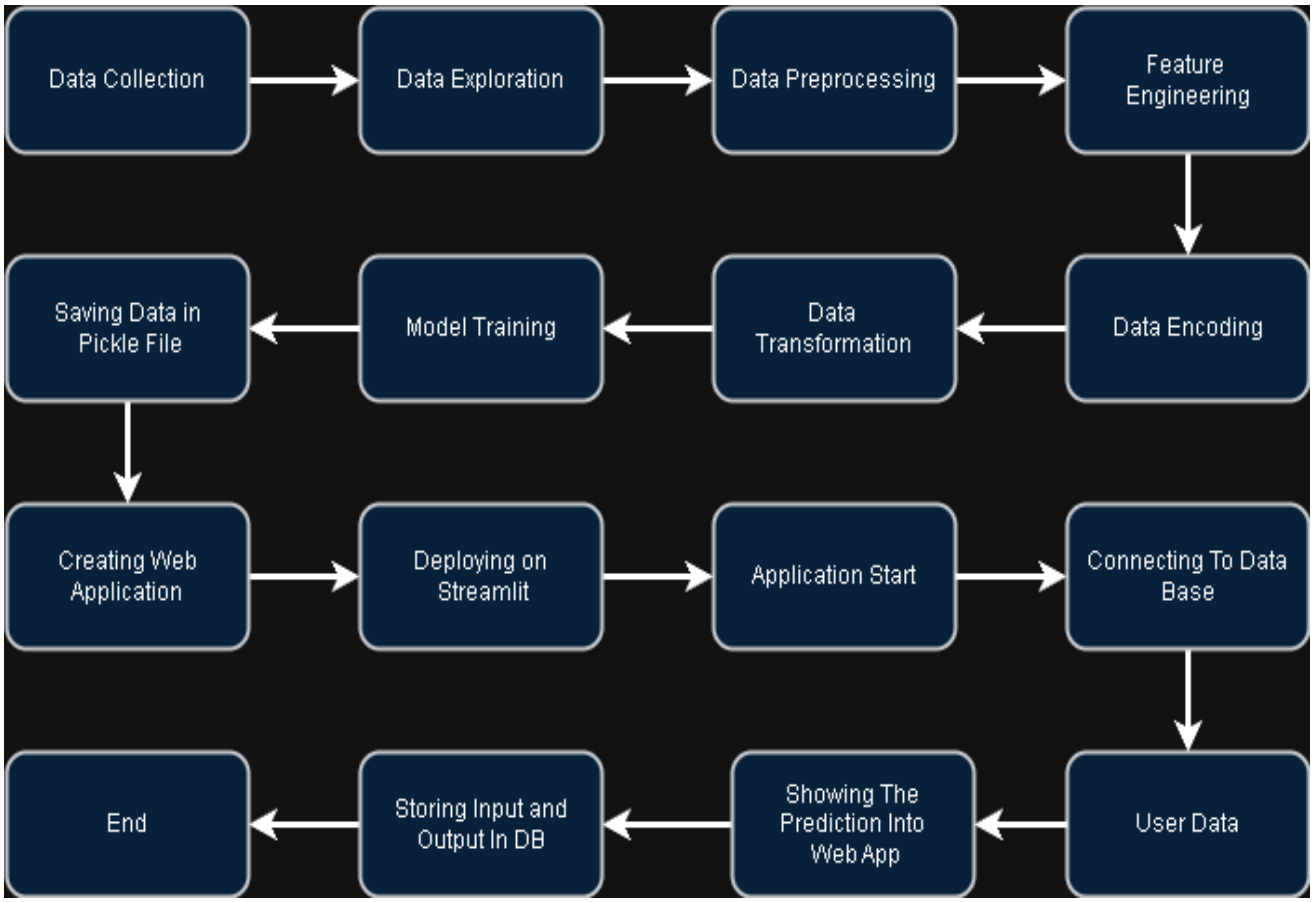
2. System Overview

The bike share prediction system aims to predict bike availability at various stations, helping users plan their rides efficiently. The system preprocesses this data to handle missing values, remove duplicates, and normalize the data. It then builds predictive models using various machine learning algorithms. The models are evaluated to select the best-performing one, which is deployed via a web interface. Users can input data such as date, time, and weather conditions to receive predictions on bike availability. The system also stores user inputs and predictions in a database for future reference and analysis.

3. Architecture Diagram

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



4. Detailed Design

4.1 Data Collection

Data gathering involves the collection of these datasets. The data is collected from the UCI Machine Learning Repository Bike Sharing Dataset and other relevant sources. The collected data serves as the foundation for building the prediction model

4.2 Data Requirements

The data in this study are shared bike riding records and bike station status, as well as local weather information for that period. The data on the operation of bikes contained information on the time of rental, return time, and place of rental. For the purpose of analyzing daily usage, the data were classified on a daily basis and the daily rental volume was set as a class.

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)

4.3 Tool Used

Here are the technical requirements for the bike share prediction model, structured in a detailed and organized manner:

Programming Language and Frameworks

- **Python:** Used as the primary programming language due to its simplicity and the extensive range of libraries available.
- **Libraries**:
  - **NumPy**: For numerical computations.
  - **Pandas**: For data manipulation and analysis.
  - **Scikit-learn**: For building and evaluating machine learning models.
  - **Alternative Modules**: Other relevant modules for specific tasks as needed.

Integrated Development Environment (IDE)

- **Visual Studio Code (VSCode)**: Used as the primary IDE for its versatility and support for Python development.

Data Visualization

- **Seaborn**: For creating attractive and informative statistical graphics.
- **Matplotlib**: Components of Matplotlib are used for creating static, animated, and interactive visualizations in Python.

Front-End Development

- **Flask**: For creating the web application that will serve the prediction model.
- **Stream lit**: For building and deploying the interactive web application.

Version Control and Management

- **GitHub**: Employed for version management and control to track changes, collaborate with team members, and maintain the project's history.

Deployment

- **Stream lit**: Used for the deployment of the web application, allowing for easy sharing and interaction with the model.

4.4 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

4.5 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.

4.6 Feature Engineering:

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

4.7 Data Transformation:

Applying transformations to the data to improve model performance.

4.8 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

4.9 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.

4.10 Deploying on Streamlit:

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

4.11 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.

4.12 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

4.13 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

4.14 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

4.15 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.