Low Level Design

Bike share prediction

Written By	Prince , Vaibhav
Document Version	0.3
Last Revised Date	19-July-2023

Contents

Document Control

Change Record:

Version	Date	Author	Comments
0.1	18 – July - 2023	Vaibhav	Introduction & Architecture defined
0.2	18 –July - 2023	Prince	Architecture & Architecture Description appended and updated
0.3	19 – July - 2023	Prince , Vaibhav	Unit Test Cases defined and appended

Reviews:

Version	Date	Reviewer	Comments
0.2	19 – July -2023	Prince	Document Content , Version Control and Unit Test Cases to be added

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

1. Introduction

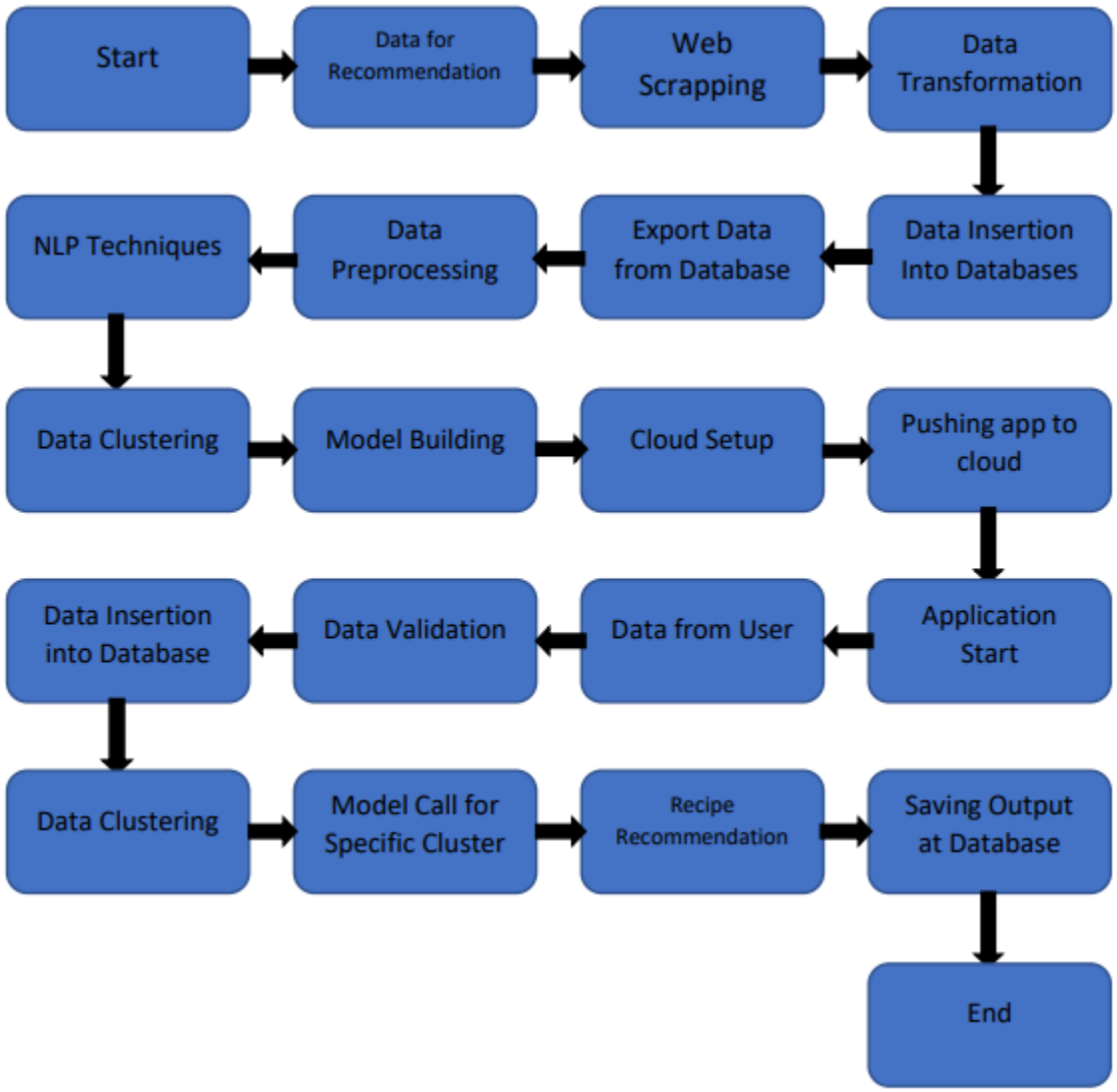
1.1. What is Low-Level design document?

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 contains information about a bike-sharing system. Below is the data description for each attribute:

1. **`instant`**: A unique record index or identifier.
2. **`dteday`**: The date of the record.
3. **`season`**: The season of the year.
   1: Spring
   2: Summer
   3: Fall
   4: Winter
4. **`yr`**: The year of the record.
   0: 2011
   1: 2012
5. **`mnth`**: The month of the record (1 to 12).
6. **`hr`**: The hour of the day (0 to 23).
7. **`holiday`**: Indicates if the day is a holiday or not.
   0: Not a holiday
   1: Holiday
8. **`weekday`**: The day of the week.
   0: Sunday
   1: Monday
   2: Tuesday
   3: Wednesday
   4: Thursday
   5: Friday
   6: Saturday
9. **`workingday`**: Indicates if the day is a working day (neither a weekend nor a holiday).
   0: Not a working day
   1: Working day
10. **`weathersit`**: Weather situation category.
    1: Clear, Few clouds, Partly cloudy
    2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds
    3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
11. **`temp`**: Normalized temperature in Celsius, scaled to a range of 0 to 1 (divided by 41, the maximum temperature).
12. **`atemp`**: Normalized feeling temperature in Celsius, scaled to a range of 0 to 1 (divided by 50, the maximum feeling temperature).
13. **`hum`**: Normalized humidity, scaled to a range of 0 to 1 (divided by 100, the maximum humidity).
14. **`windspeed`**: Normalized wind speed, scaled to a range of 0 to 1 (divided by 67, the maximum wind speed).

15. `casual`: Count of casual users (non-registered users) renting bikes.
16. `registered`: Count of registered users renting bikes.
17. `cnt`: Count of total rental bikes, including both casual and registered users.

3. 2 Data Pre-processing

1. Remove irrelevant columns (e.g., `instant`, `dteday`, `casual`, `registered`) that do not contribute to the prediction task.
2. Encode categorical variables (e.g., `season`, `yr`, `mnth`, `hr`, `holiday`, `weekday`, `workingday`, `weathersit`) using one-hot encoding or label encoding.
3. Split the dataset into input features (X) and the target variable (y) and further divide it into training and testing sets for model training and evaluation.
4. Optionally, handle any missing values or outliers through imputation or data cleansing techniques to ensure the data is suitable for model building.

3.3 Model Building

- Choose an appropriate regression algorithm, such as Linear Regression, Decision Tree Regression, Random Forest Regression, or Gradient Boosting Regression, based on the dataset size and complexity.
- Train the selected regression model on the preprocessed training data to learn the underlying patterns and relationships between input features and the target variable (bike rental count).
- Perform hyperparameter tuning using techniques like Grid Search or Randomized Search with cross-validation to optimize the model's performance.
- Evaluate the trained model on the testing data using regression evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared to assess its ability to predict bike rental counts accurately.
- If the model's performance is satisfactory, deploy it in the desired deployment option, enabling real-time or periodic bike rental predictions for users. Continuously monitor the model's performance and retrain it with fresh data periodically to maintain accuracy and relevance.

3.4 Data from User

The Bike Share Prediction project aims to build an end-to-end regression system using user-provided data, such as weather conditions, date, and time of bike rentals. By leveraging this data, the system will train a machine learning model to predict bike rental counts, providing personalized recommendations and optimizing bike sharing experiences for users worldwide.

3.5 Data Validation

Data validation is a crucial step in the data preprocessing process for the Bike Share Prediction project.

During data validation, we ensure the integrity and quality of the dataset by performing various checks and verifications. This includes checking for missing values, identifying and handling outliers, verifying data types, and ensuring that categorical variables have appropriate labels. Additionally, we validate the data against defined constraints to detect any inconsistencies or errors that may affect the accuracy and reliability of the machine learning model. By conducting thorough data validation, we can ensure that the dataset is robust and reliable for building an accurate prediction system.

3.6 Different model for Day and Hour dataset

- For the Bike Share Prediction project, we can explore and compare different regression models to predict bike rental counts based on the "days" and "hours" datasets. As the datasets have different levels of granularity, we may consider different models for each to account for their unique characteristics.
- For the "days" dataset, which is aggregated on a daily basis, we can experiment with models like Linear Regression, Decision Tree Regression, Random Forest Regression, and Gradient Boosting Regression. These models can capture the daily patterns and trends in bike rentals, considering factors like weather conditions, holidays, and seasonal variations.
- Additionally, we can try ensemble techniques like Stacking, which combines multiple models to leverage their strengths and improve predictive performance. The choice of models will be based on their ability to handle the temporal aspect of the "hours" dataset and the day-to-day variations in the "days" dataset. Through model comparison and evaluation, we can identify the best-performing models for each dataset, enabling accurate bike rental predictions for different time granularities.

4. Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Data Preprocessing - Irrelevant Column Removal	Raw dataset with irrelevant columns (e.g., 'instant', 'dteday', 'casual', 'registered').	After data preprocessing, the dataset should have the specified irrelevant columns removed, and the remaining relevant columns should be retained for further processing.
Categorical Variable Encoding	Raw dataset with categorical variables (e.g., 'season', 'holiday', 'weekday') needing encoding.	Categorical variables should be properly encoded using one-hot encoding or label encoding, resulting in numerical representations that can be used for model training.
Model Training - Training Data Availability	Preprocessed dataset with input features (X) and target variable (y) split into training and testing sets	The selected regression model should be able to successfully train on the training data without encountering any errors.
Hyperparameter Tuning - Best Hyperparameter Selection	Preprocessed training data and hyperparameter tuning algorithm (e.g., Grid Search) implemented.	The hyperparameter tuning process should identify and select the best combination of hyperparameters for the model, optimizing its performance.
Model Evaluation - Evaluation Metrics Calculation	Preprocessed testing data and a trained regression model.	The model evaluation metrics (e.g., RMSE, R-squared) should be accurately calculated based on the model's performance on the testing data.