**Low Level Design**

Bike Share Prediction

|  |  |
| --- | --- |
| Written by | Biswajit Rajguru Mohapatra |
| Document Version | 0.1 |
| Last revision date |  |

**Document Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | **19-Nov-21** | **Biswajit Rajguru Mohapatra** | **Introduction** |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review date** | **Reviewed by** | **Approved by** | **Comments** |
|  |  |  |  |  |

**Contents**

**1. Introduction-----------------------------------------------------------------------------------------------4**

**1.1 What is Low-Level design document? ---------------------------------------------------4**

**1.2 Scope--------------------------------------------------------------------------------------------4**

**2. Architecture-----------------------------------------------------------------------------------------------5**

**3. Architecture Description-------------------------------------------------------------------------------6**

**3.1 Data Description------------------------------------------------------------------------------6**

**3.2 Pushing Data into Database----------------------------------------------------------------6**

**3.3 Extracting data from database------------------------------------------------------------6**

**3.4 Data pre-processing--------------------------------------------------------------------------6**

**3.5 Data clustering--------------------------------------------------------------------------------7**

**3.6 Model Building--------------------------------------------------------------------------------7**

**3.7 Receive data from user/client---------------------------------------------------------------7**

**3.8 Data Validation--------------------------------------------------------------------------------7**

**3.9 Data push to database------------------------------------------------------------------------7**

**3.10 Data Clustering-------------------------------------------------------------------------------7**

**3.11 Predict the output and save to data base------------------------------------------------7**

**3.12 Deployment------------------------------------------------------------------------------------7**

**4. Unit Test Case---------------------------------------------------------------------------------------------8**

**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 programme code for Bike Share Prediction model. LLD describe the class diagrams with the methods and relation between classes and programme specs. It describes the modules so that the programmer can directly code the programme from the document.

**1.2 Scope**

Low-Level design documentation is a component of model design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization maybe defined during requirement analysis and then refined during data design work.

**2 Architecture**

**START**

**DATA COLLECTION**

**DATA VALIDATION**

**PUSHING DATA INTO DATABASE**

**EXTRACTING DATA FROM DATABASE**

**DATA AGGREGATION**

**DATA PREPROCESSING**

**CLUSTERING**

**MULTI-MODEL SYSTEM FOR EACH MODEL**

**TEST MODEL**

**CLOUD SETUP**

**PUSHING APP TO CLOUD**

**START APPLICATION**

**RECIVE DATA FROM USER/CLIENT**

**DATA VALIDATION**

**DATA PUSH TO DATABASE**

**DATA CLUSTERING**

**MODEL CALL FOR SPECIFIC CLUSTER**

**PREDICT THE OUTCOME**

**SAVE THE PRDICTION INTO DATABASE**

**END**

**3 Architecture Description**

**3.1 Data description**

We have two sets of datasets; one is for hourly basis and another is for day basis. Both of the datasets contain the weather conditions such as humidity, temperature, air temperature and windspeed essential factor to determine and conclude the weather outside of the particular place. These variables in the datasets are our prime data for prediction. The datasets also contain some variables columns such as instant, year, dteday, casual and registered which doesn’t contribute to our prediction. The datasets also hold the information about month, season, holiday, weekday, working day and weather outside.

**3.2 Data validation**

Before reading the data, we will check the validation of the data that we receive on the basis of file format, name of the file, number of columns, column names and null value if present. If the data satisfies all the requirements, then we proceed further else we move the data into archive/bad file folder.

* File format - .csv
* File name – hour.csv, day.csv
* Number of columns – 16

**3.3 Pushing data into Database**

1. Database connection and creation – Create a database with name ‘bikeshareprediction’. If the database is already present and is empty, connect to the database.
2. Create table in the database.
3. Insertion of the data into the respective table.

**3.4 Extracting data from database**

Extracting data from database – The data from the respective database is being exported as ‘.csv’ file for data pre-processing and model training.

**3.5 Data pre-processing**

In the data pre-processing we can handle the null-values, drop the columns that doesn’t contribute in the prediction process and converting some of the numerical data such as month, season, weather and weekday to categorical data.

**3.6 Data clustering**

K-mean++ algorithm will be used to create clusters in the pre-processed data. The idea behind clustering is to implement different algorithm to train data in different clusters. The K-mean++ model is trained over pre-processed data and the model is saved for further use in the prediction.

**3.7 Model Building**

After clusters are created, we will find the best model for each cluster. For each cluster, algorithms will be passed with best parameters derived from Grid-Search. We will calculate AUC scores for the models and select the model with the best score. Similarly, the models will be selected for each cluster. All the models for every cluster will be saved for use in Recommendations.

**3.8 Receive data from user/client**

Here, we will be collecting the data about various parameters essential for prediction of the demand in bike sharing form the user.

**3.9 Data Validation**

Validation of the data provided by the user will be done.

**3.10 Data push to database**

Here, we will push the collected data from user/client to the database. The database could be Cassandra database.

**3.11 Data Clustering**

Based on the cluster number, the respective model will be loaded and will be used to predict the data for the cluster.

**3.12 Predict the output and save to data base**

Calling the suitable ML regression model for the cluster to which data belongs, we will predict the demand of the bike share according to the condition provided buy the client. We will the save the prediction to the database.

**3.13 Deployment**

We will be deploying the model to AWS.

4 Unit Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Results** |
| Verify whether the Application URL is accessible to the user | 1. Application URL must 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 sin 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 the 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 |