LOW LEVEL DESIGN (LLD)

ENERGY EFFICIENCY

Revision Number : 2.0

Last Date of Revision : 02/09/2021

Document Control

Version	Date	Author	Comments
1.0	21/08/2021	Mayur	Document Created
2.0	02/09/2021	Mayur	Update more information It should be in detail.

CONTENT

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

2.1 Model Flowchart

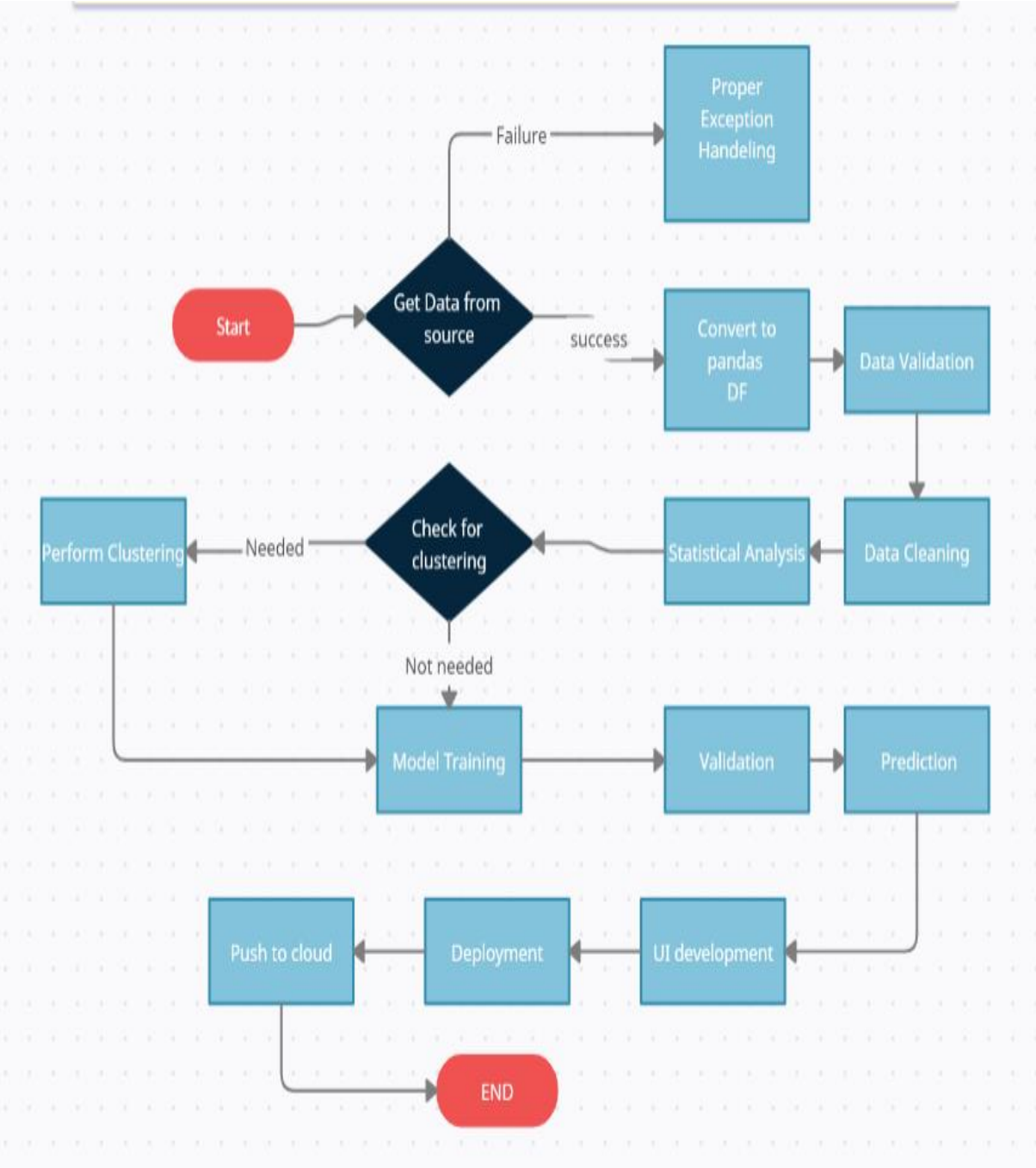

Figure No.1 The Flow Chart For The Model

2.2 Model Flow

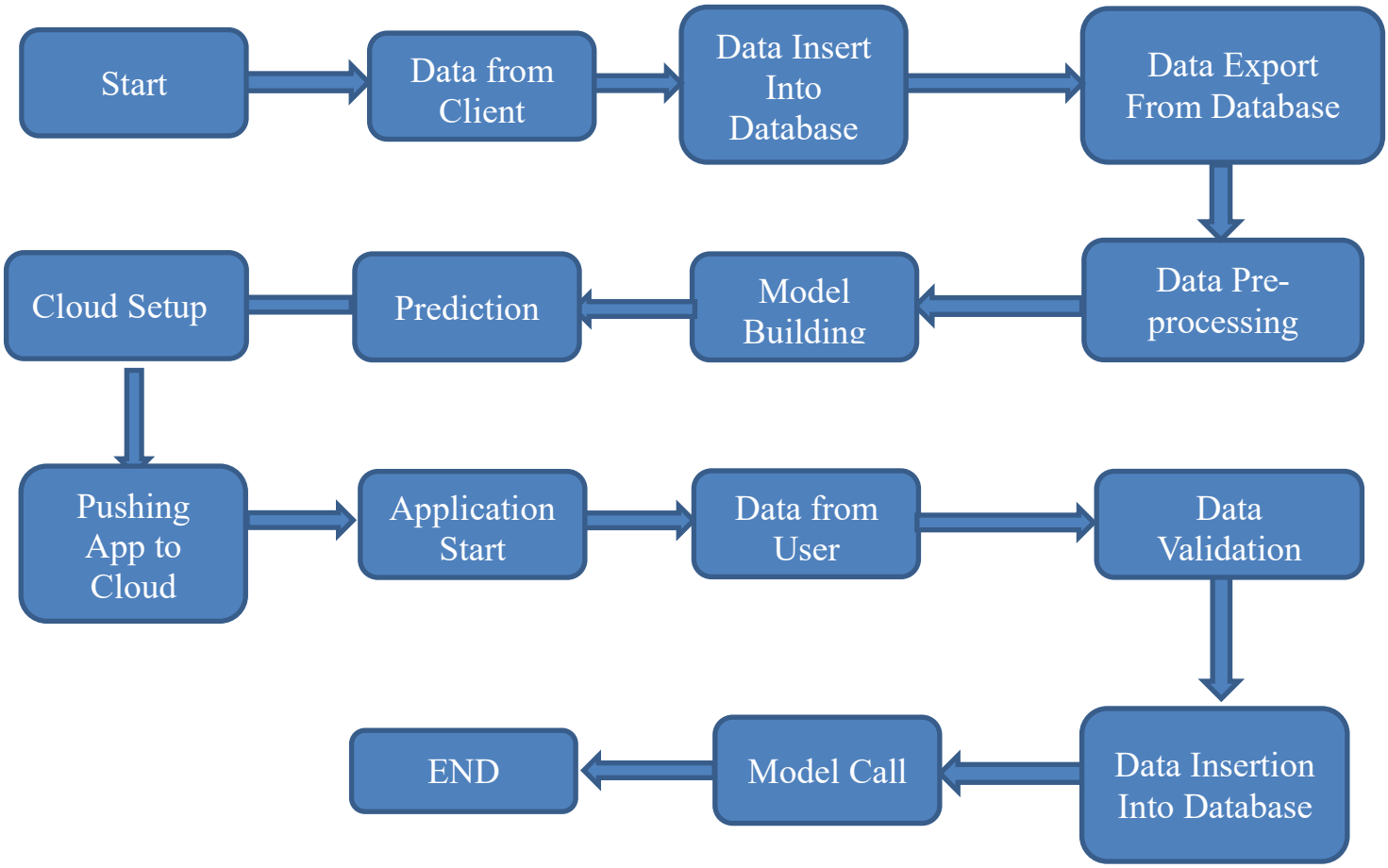

Figure No. 2 The Box Chart of The Model

3. Architecture Description

3.1 Data Description

The data can collect form the client in any file format. They perform energy analysis using 12 different building shapes simulated in Ecotect. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. We simulate various settings as functions of the afore-mentioned characteristics to obtain 768 building shapes. The dataset comprises 768 samples and 8 features, aiming to predict two real valued responses. It can also be used as a multi-class classification problem if the response is rounded to the nearest integer.

The dataset contains eight attributes (or features, denoted by X1...X8) and two responses (or outcomes, denoted by y1 and y2). The aim is to use the eight features to predict each of the two responses

Indication	Names
X1	Relative Compactness
X2	Surface Area
X3	Wall Area
X4	Roof Area
X5	Overall Height
X6	Orientation
X7	Glazing Area
X8	Glazing Area Distribution
Y1	Heating Load
Y2	Cooling Load

3.2 Data Insertion into Database

- Database Creation & Connection - Create a database and create connection.
- Table Present or Not – Check if the table is present, if not create it.
- Insert File – Insert the file in the table that which you are creates.

3.3 Export Data from Database

Data Export from Database - The data in a stored database is exported as a CSV file to be used for Data Pre-processing and Model Training.

3.4 Data Pre-Processing

Data Pre-processing steps we could use are Null value handling, stop words removal, punctuation removal, Tokenization, Lemmatization, TFIDF, Imbalanced data set handling, Handling columns with standard deviation zero or below a threshold, etc.

3.5 Model Building

After clusters are created, we will find the best model for each cluster. For each cluster, algorithms will be passed with the best parameters derived from Grid-Search. We will calculate the AUC scores for 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 Recommendation.

3.6 Hyper Parameter Tuning:

In hyper parameter tuning we have implemented various ensemble techniques like random forest regressor we also done randomized search cv or grid search cv and from that we Also implemented cross validation techniques for that. From that we have chosen best parameters according to hyper parameter tuning and best score from their accuracies so we got 85% accuracy in our random forest regression after hyper parameter tuning.

3.7 Model Dump

After comparing all accuracies and checked all ROC & AUC curve we have chosen hyper parameterized random forest regression as our best model by their results so we have dumped these model in a pickle File format with the help of pickle python module.

3.8 Cloud Setup

After model building we want to deploy the model to server. In deployment we can use different services such as Amazon Web Service (AWS), Azure Service, and Google Cloud Service (GCP). The model will be deployed in AWS, for that 1st we have to create the environment and then after we have to create the code pipeline. Then after we have to connect our pipeline with the environment.

3.9 Data from User

Here we can collect the data from the user. In which we can collect the different type of data such as Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area, and Glazing Area Distribution.

3.10 Data Validation

Here Data Validation will be done, given by the user

3.11 Data Insert into Database

Collecting the data from the user and storing it into the database. The database can be either Cassandra.

3.12 Data Clustering

The model created during training will be loaded, and clusters for the user data will be predicted.

3.13 Model Call for Specific Cluster

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

3.14 Deployment

We will be deploying the model to AWS. This is a workflow diagram for the Recipe recommendation.

4. Technology Stack

Front End	HTML/CSS
Back End	Python/Flask
Database	Cassandra
Deployment	AWS
Model	1. Linear Regression 2. Random Forest
IDE	PyCharm

5. User I/O Workflow

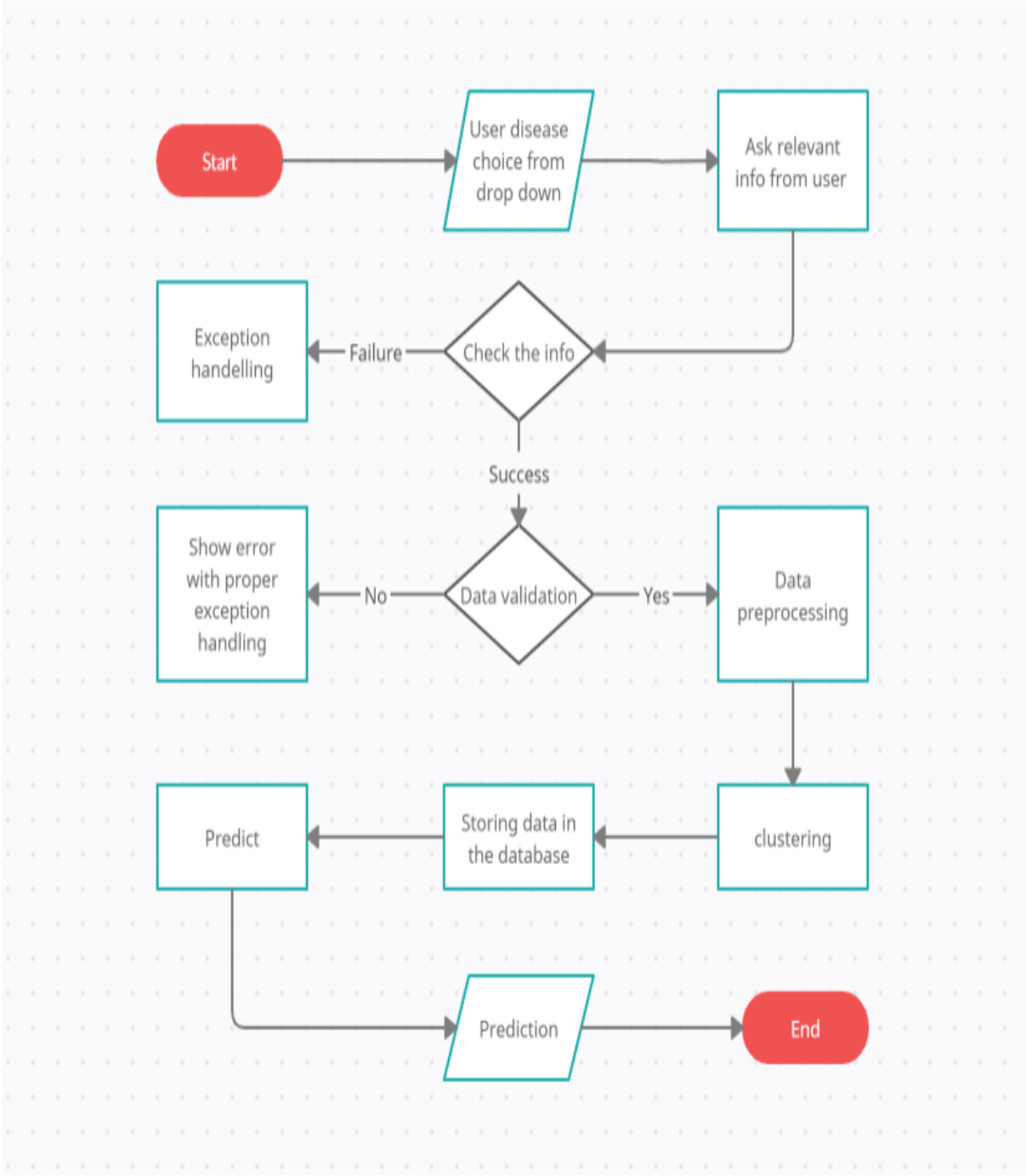

Figure No.3 The User Input Output Flow

6. Unit Test Case

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 user is able to see input fields	1. Application is accessible 2. User is logged in to the application	User should be able to see input fields
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