Low Level Design (LLD)

Energy Efficiency

Written By	**MANJUNATH BHASHYAM**
Document Version	**0.1**
Last Revised Date	**03 – Aug -2022**

Document Control

Change Record:

Version	Date	Author	Comments

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

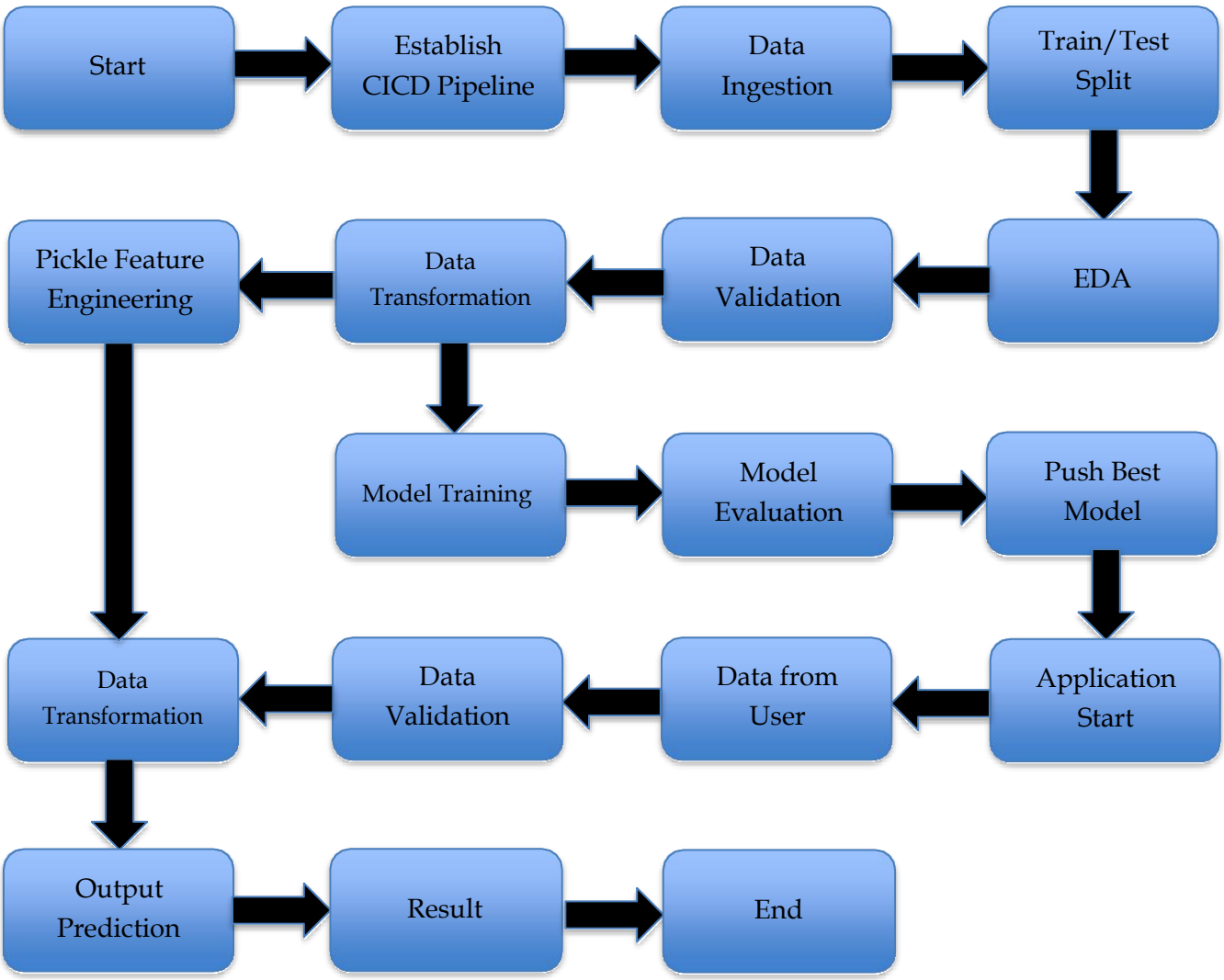
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Energy Efficiency. 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 eight attributes (or features, denoted by X1, X2, X3, X4, X5, X6, X7, X8) respectively denoting relative compactness, surface area, wall area, roof area, overall height, orientation, glazing area, glazing area distribution - the effect of these 8 features on the 2 responses (or outcomes, denoted by y1 and y2) respectively for heating load (HL) and cooling load (CL) is to be investigated. The dataset contains 768 instances and 8 attributes.

3.2. Establish CICD Pipeline

Initially CICD Pipeline is established using Docker, GitHub Actions with the end application deployed to Heroku. The above is implemented in order that every time a new commit is made the same is reflected causing Continuous Integration and Continuous Deployment.

3.3. Data Ingestion

The complete dataset is been ingested from the Data folder from UCI repository which is in Excel format.

3.4. Train/Test Split

The ingested data is split into Train data and Test data using Stratified Shuffle split so that both the train and test data have the same kind of Distribution.

3.5. EDA

Exploratory Data Analysis performed on the train dataset to gather all insights and usage statistics.

3.6. Data Validation

The Data Validation is been done in order to check the presence of any Data Drift, Null Valves, Outliers etc.

3.7. Data Transformation

In the Data Transformation Process, we will transform our train and test data using Standard Scaler and we will convert the train/test data into NumPy arrays.

3.8. Pickle Feature Engineering

All the feature engineering performed is saved as a pickle file to apply the same to the test data set and to any new input dataset.

3.9. Model Training

The transformed data set is used to train the Machine Learning model where algorithms are passed with the best parameters derived from Grid-Search.

3.10. Model Evaluation

All the algorithms are been evaluated with respect to the set base accuracy to be achieved, least difference between train and test accuracy to overcome overfitting and under fitting conditions.

3.11. Push Best Model

The best model selected from the Model Evaluation is saved as pickle file and is used for Prediction.

3.12. Data from User

Here we will collect data directly provided from user such as relative compactness, surface area, wall area, roof area, overall height, orientation, glazing area and glazing area distribution.

3.13. Data Validation & Data Transformation

Here Data Validation is performed for the input data and the data is transformed using the feature engineering pickle file as the same was done during Model Training.

3.14. Output Prediction & Result

Here Heating load and Cooling load the two outcomes are predicted based on the transformed input data from user which is sent to the Best Model selected and the respective results are obtained, displayed.

4. Unit Test Cases

Test Case 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 user is able to edit all input fields	Application is accessible	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	Application is accessible	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	Application is accessible	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	Application is accessible	The recommended results should be in accordance to the selections user made