Low Level Design

Thyroid Disease Detection

Written By	Rakesh Uikey
Document Version	1.0
Last Revised Date	

Document Version Control

Change record:

Date Issue	Version	Description	Author
14/12/2022	1	Initial LLD – V 1.0	Rakesh Uikey

Approval Status

Version	Review Date	Review by	Approved by	Comments

Contents

1. Introduction

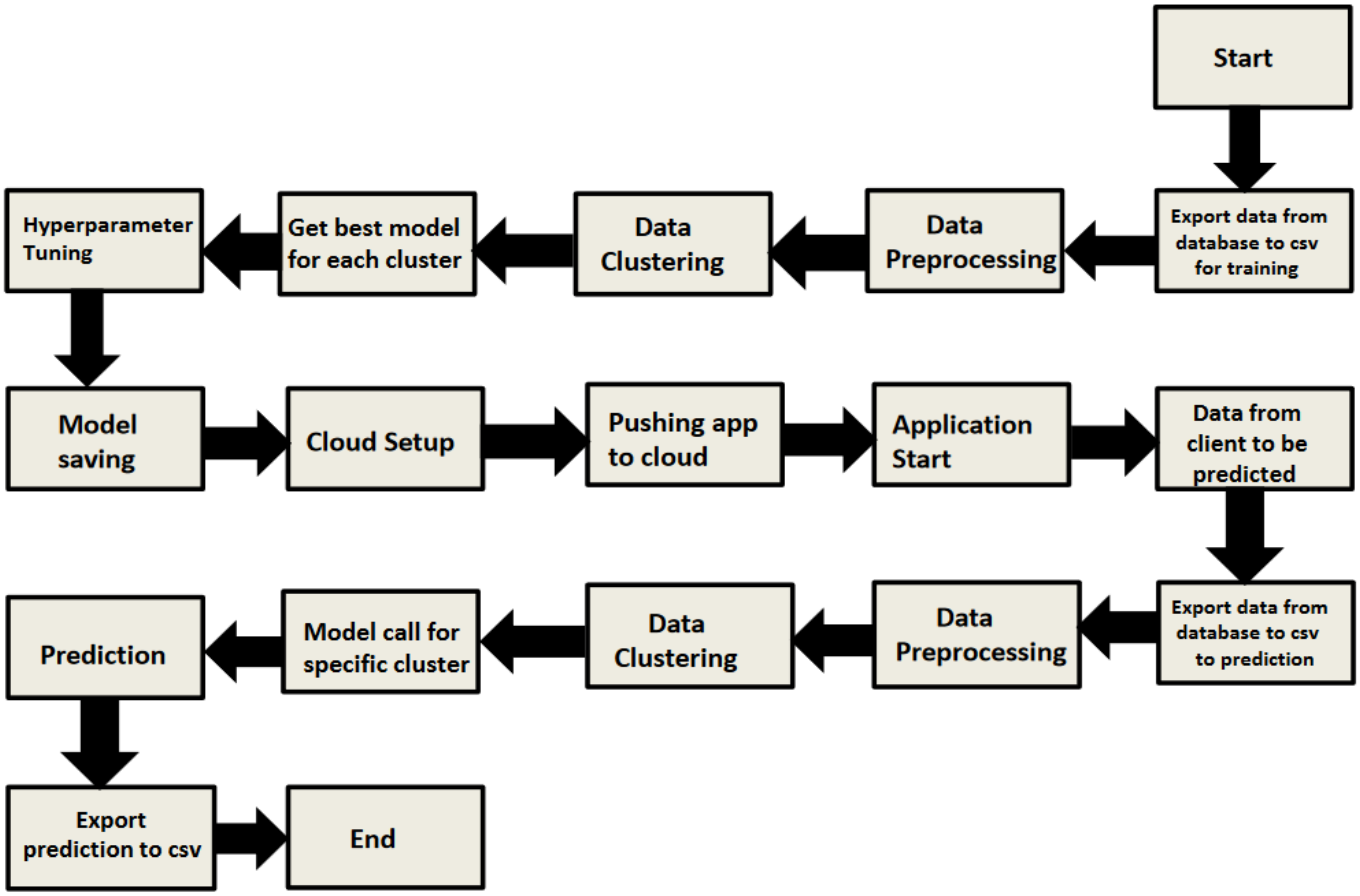
1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program code for Thyroid Disease Detection System. LLD describe the class diagrams with the methods and relations between classes and program specs. It describe 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

We will be using Thyroid Disease Data Set present in UCI Machine Learning Repository. This Data set is satisfying our data requirement. Total 7200 instances present in different batches of data.

3.2 Data Preprocessing

We will be exploring our data set here and do EDA if required and perform data preprocessing depending on the data set. We first explore our data set in Jupyter Notebook and decide what pre-processing and Validation we have to do such as imputation of null values, dropping some column, etc and then we have to write separate modules according to our analysis, so that we can
implement that for training as well as prediction data.

3.3 Data Clustering

K-Means algorithm will be used to create clusters in the pre-processed data. The optimum number of clusters is selected by plotting the elbow plot. The idea behind clustering is to implement different algorithms to train data in different clusters. The K-means model is trained over pre-processed data and the model is saved for further use in prediction.

3.4 Get best model of each cluster

Here we will train various model on each cluster which we will obtain in Data Clustering, and then will try to get best model of each cluster.

3.5 Hyperparameter Tuning

After selecting best model for each cluster, we will do hyperparameter tuning for each selected model, and try to increase performance of the models.

3.6 Model Saving

After performing hyperparameter tuning for models, we will save our models so that we can use them for prediction purpose.

3.7 Cloud Setup

Here We will do cloud setup for model deployment. Here we also create our flask app and user interface and integrate our model with flask app and UI

3.8 Push app to cloud

After doing cloud setup and checking app locally, we will push our app to cloud to start the application.

3.9 Data from client side for prediction purpose

Now our application on cloud is ready for doing prediction. The prediction data which we receive from client side will be exported from DB and further will do same data cleansing process as we have done for training data using modules we will write for training data. Client data will also go along the same process of Exporting data from DB, Data pre-processing, Data clustering and according to each cluster number we will use our saved model for prediction on that cluster.

3.10 Export Prediction to CSV

Finally when we get all the prediction for client data, then our final task is to export prediction to csv file and hand over it to client.

4. Unit Test Cases

Test Case Description	Pre - requisite	Expected result
Verify whether application URL is accessible to the user	1.Application URL should be defined	Application URL should be accessible to the users
Verify whether the application loads successfully when the URL is hit	1. Application URL is accessible 2. Application is deployed	The application loads successfully when the URL is hit
Verify whether user is able to see input fields	1. Application is accessible	User should be able to see input fields
Verify whether user is able to edit all input fields	1. Application is accessible	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. 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	1. Application is accessible	The recommended results should be in accordance to the selections user made