THYROID DISEASE DETECTION

Low-Level Design (LLD)

Revision Number: 1.0

Last date of revision:03-Aug-2023

Document Version Control

Date Issued	Version	Description	Author
03/08/2023	1.0	Complete LLD	Srikant Jena

Content

1.Introduction

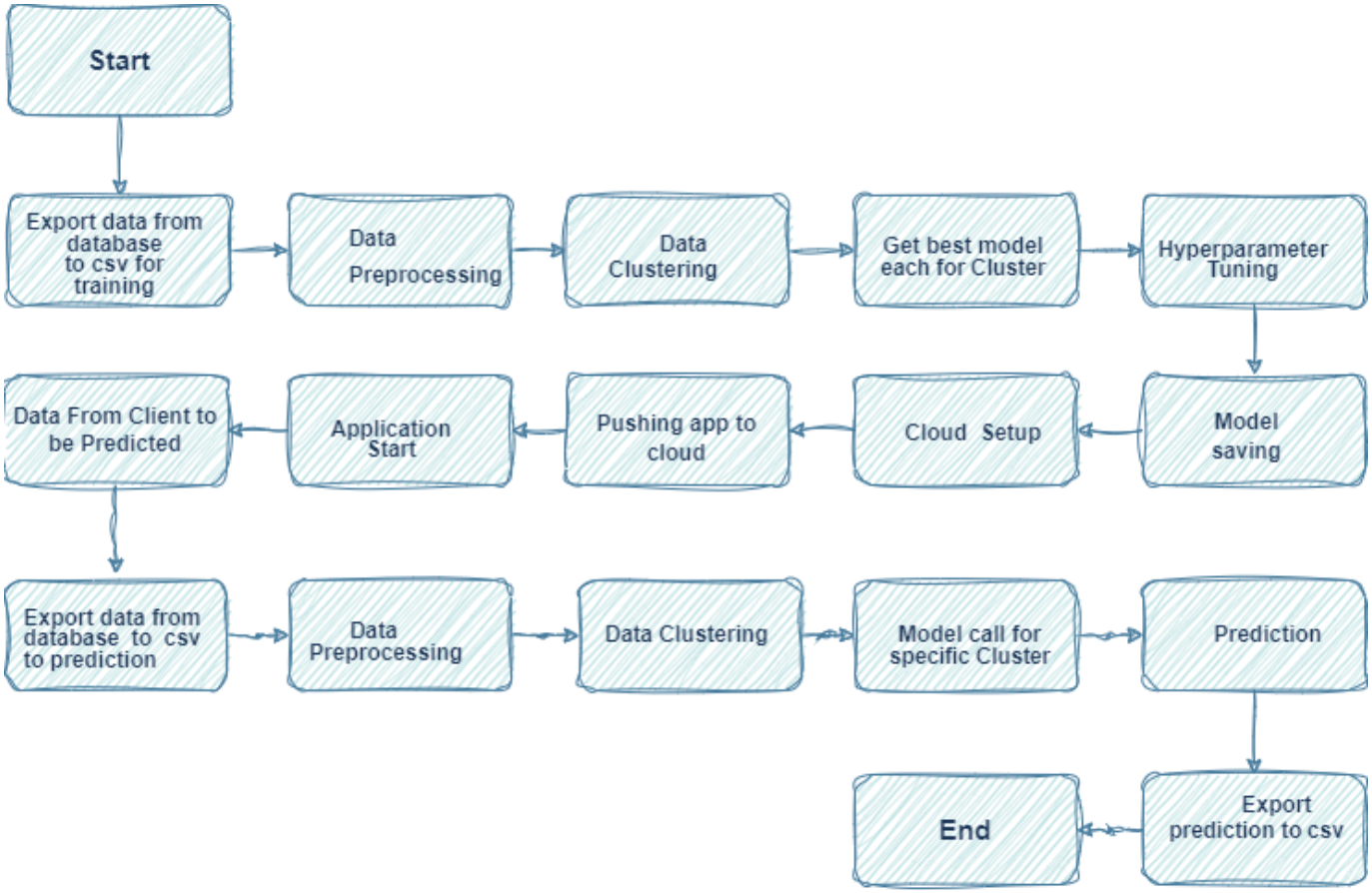
1.1. What is Low-Level design document?

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 classesand 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.

1. 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 Export Data from database to CSV for Training

Here we will be exporting all batches of data from database into one csv file for training.

3.3 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.

3.4 Data Clustering

The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm.

Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models.

3.5 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.

3.6 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.

3.7 Model Saving

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

3.8 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.

3.9 Push app to cloud

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

3.10 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.

3.11 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

3.12 User Interface

We will make two types of user interfaces.

1. For single user input prediction, we will make a separate UI which willtake all inputs from a single user and give back the prediction there only. So that any single user can also use this system for single prediction.

2. For batch or bulk prediction, we will make a separate UI which will takea CSV file from client and make bulk prediction at a single time and save those prediction in result.csv file and UI will give option to download that file.

4.0 Unit Test Cases

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 the User can sign up in the application	1. Application is accessible	The User should be able to sign up in the application
Verify whether user can 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 can 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 logging in
Verify whether user can 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