LOW LEVEL DESIGN

Thyroid Disease Detection System

Made by - Arkoprovo Ghosh

Document Version Control

Change Record:

Version	Date	Author	Comments
1.0	28-07-2023	Arkoprovo Ghosh	Architecture

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

1.What is Low-Level design document?

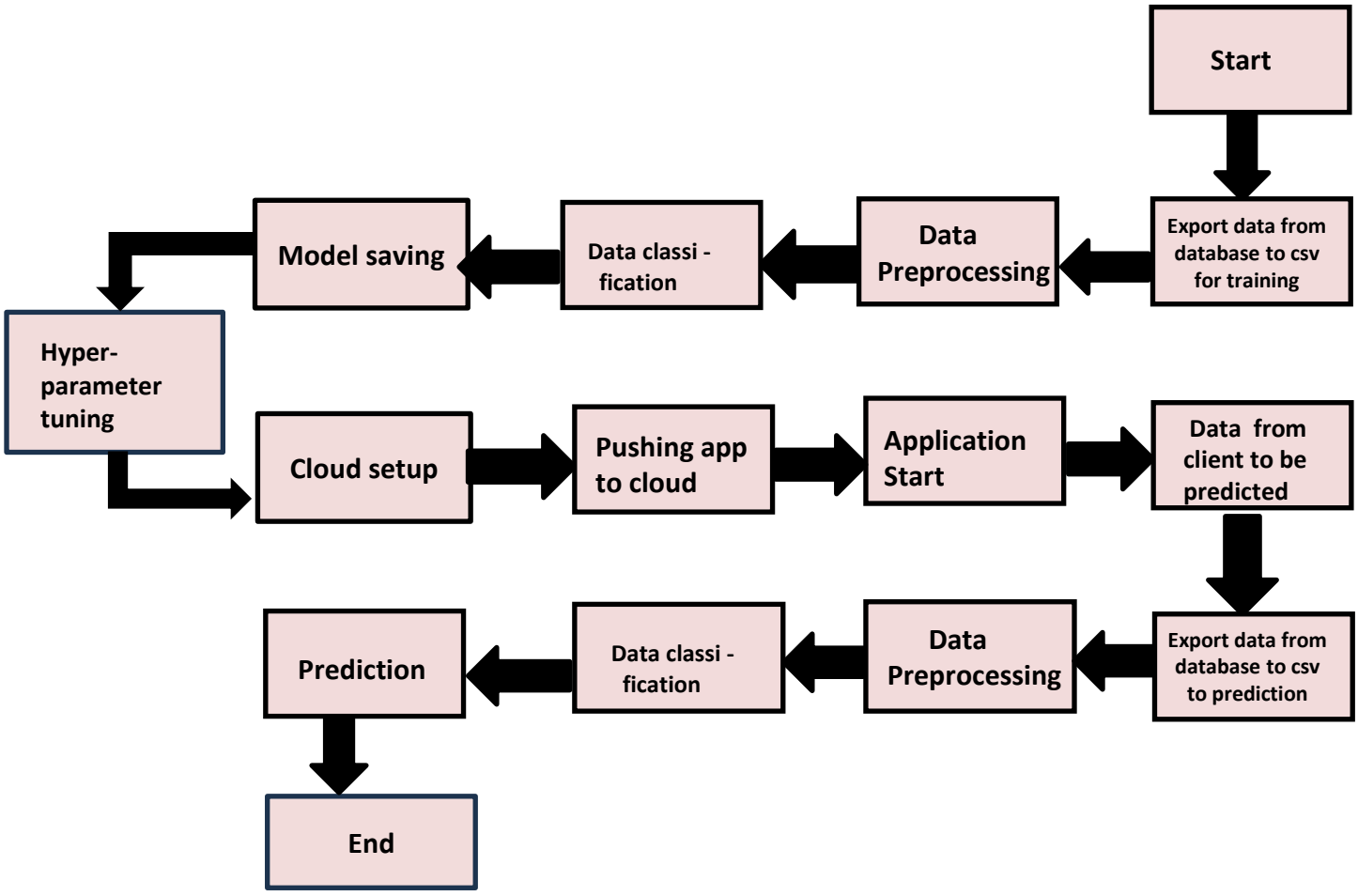
A Low-Level Design Document (LLD) is a detailed technical document that outlines the internal logical design of a software program or system. In the context of the Thyroid Disease Detection System, the LLD would specifically focus on providing a comprehensive design for the actual program code of the system.

The primary goal of the LLD is to offer a blueprint for the software's implementation, describing the various components, classes, methods, and their relationships with each other. It also includes specific program specifications, ensuring that programmers can directly translate the design into executable code using the document as a reference.

1.2 Scope

The scope of Low-Level Design (LLD) is a component-level design process that involves a step-by-step refinement approach. During this process, designers focus on designing data structures, defining the required software architecture, writing the source code, and optimizing performance algorithms. The data organization may have been initially outlined during the requirement analysis phase, and LLD further refines and solidifies those designs during the data design work. Ultimately, LLD plays a crucial role in guiding the development team and ensuring a systematic and efficient coding process for the Thyroid Disease Detection System.

2. Architecture



Architecture Description: Thyroid Disease Prediction

3.1 Data Description:

We will use the Thyroid Disease Data Set from the UCI Machine Learning Repository, which contains 7200 instances in different batches of data.

3.2 Export Data from Database to CSV for Training:

We will export all batches of data from the database into a single CSV file to use it for model training.

3.3 Data Preprocessing:

In this step, we will explore the dataset and perform exploratory data analysis (EDA) if necessary. Based on our analysis, we will conduct data preprocessing tasks like handling null values, dropping unnecessary columns, etc. We will create separate modules for these preprocessing steps to be used during both training and prediction.

3.4 Data Classification:

We will do data classification using the Random Forest technique involves employing an ensemble of decision trees to categorize data into predefined classes or groups. The algorithm builds multiple decision trees on different subsets of the data and combines their predictions to make the final classification. This approach enhances accuracy, reduces overfitting, and handles complex datasets effectively .The trained random forest model will be saved for future predictions.

3.5 Hyperparameter Tuning:

After classification ,we will do hyperparameter tuning to get best results for prediction.

3.5 Model Saving:

After classification, we will save all the trained models so that they can be used for making predictions.

3.6 Cloud Setup:

We will set up the cloud environment for model deployment. This involves creating a Streamlit app and user interface to integrate our trained models. The streamlit app will serve as the backend, while the user interface will interact with the models.

3.7 Push App to Cloud:

Once the cloud setup is complete and the app has been tested locally, we will push the entire application to the cloud to make it publicly accessible.

3.8 Data from Client Side for Prediction Purpose:

With the application on the cloud, we can start receiving prediction data from clients. The data received from the client will go through the same data cleansing process as the training data, utilizing the modules developed earlier. The data will undergo data preprocessing, classification, and will be fed into the appropriate saved models for predictions.

3.9 Prediction displayed:

After undergoing all the above steps, finally the result i.e., for thyroid prediction, is displayed on the application.

1. User Interface

1. For prediction, we will make a separate UI which will take all inputs from a single user and give back the prediction there only.

4 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 is able to sign up in the application	1. Application is Accessible	The User should be able to sign 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 application 3.User is logged in to the application	User should be able to see input fields on 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 get prediction/output back after submitting 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 their output after submitting the inputs.
Verify whether the output which user get is accordance to inputs user made.	1. Application is accessible 2. User is signed up to the application 3.User is logged in to the application	The output should be in accordance with the inputs user made.
Verify whether user have option to download their result or not.	1. Application is accessible 2. User is signed up to the application 3.User is logged in to the application	User should have option to download their output result.