Low Level Design

Thyroid Disease Detection System

Written By	Manish Kumawat
Document Version	1.0
Last Revised Date	

Document Version Control

Change Record:

Version	Date	Author	Comments
1.0	10-09-2023	Manish Kumawat	Introduction and Architechture Defined

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

Document Version control--2

1 Introduction--4
1.1 What is Low-Level Design Document-------------------------------------4
1.2 Scope---4

2 Architecture--5

2 Architecture Description-- 5
2.1 Data Description --5
2.2 Export Data from DB to CSV for training-----------------------------------5
2.3 Data Preprocessing---5
2.4 Get best model ---5
2.5 Model saving---6
2.6 Cloud setup--6
2.7 Push App to cloud---6
2.8 Data from client side for prediction---6
2.9 Export prediction to CSV---6

1. Introduction

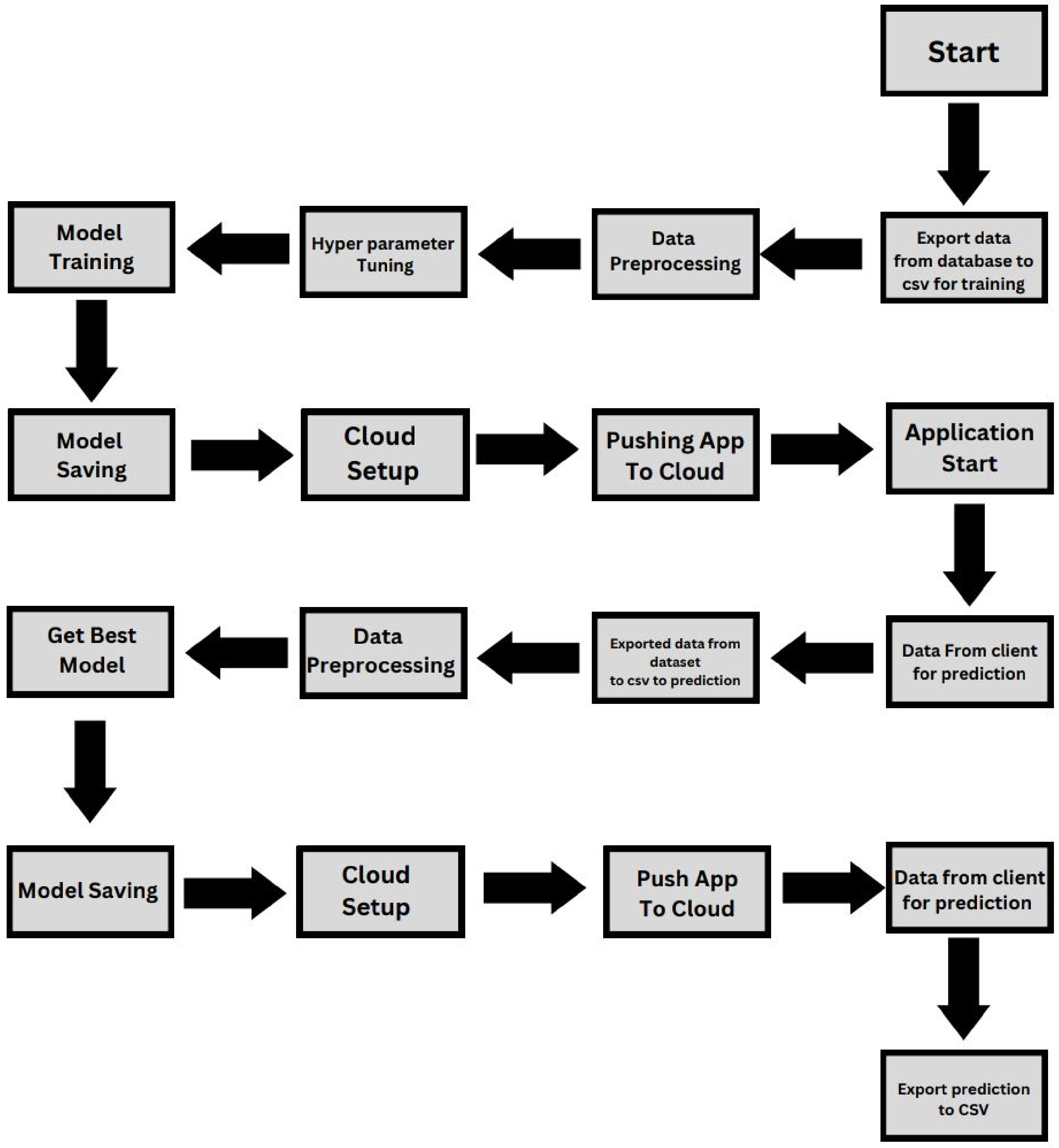
1.1 What is Low-Level design document?

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 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 Get best model

Here we will train various model on train data which we will obtain accuracy scores We will choose the model which gives us high accuracy.

3.5 Model saving

After performing model training, we will save our model so that we can use them for predicti on purpose.

3.6 Cloud setup

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

3.7 Push App to cloud

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

3.8 Data from client side for prediction

Now our application on cloud is ready for doing prediction. The prediction data which we receive from client side as input for prediction on which we 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 pipeline steps and after that we will use our saved model for predicti ons.

3.9 Export prediction to CSV

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