Payal Sanjay Talmale

Payaltalmale07@gmail.com

Low level design (LLD)

Thyroid Disease Detection

Table of Contents

**1 Introduction2**

1.1 What is lor-level document?2

1.2 Scope2

**2 Architecture2**

**3 Architecture Design3**

3.1 Data Description3

3.2 Data Collection3

3.3 Data Processing3

3.4 Data Transformation3

3.5 Model Building3

3.6 Data from user3

3.7 Model call4

**4 Unit test cases4**

1 Introduction

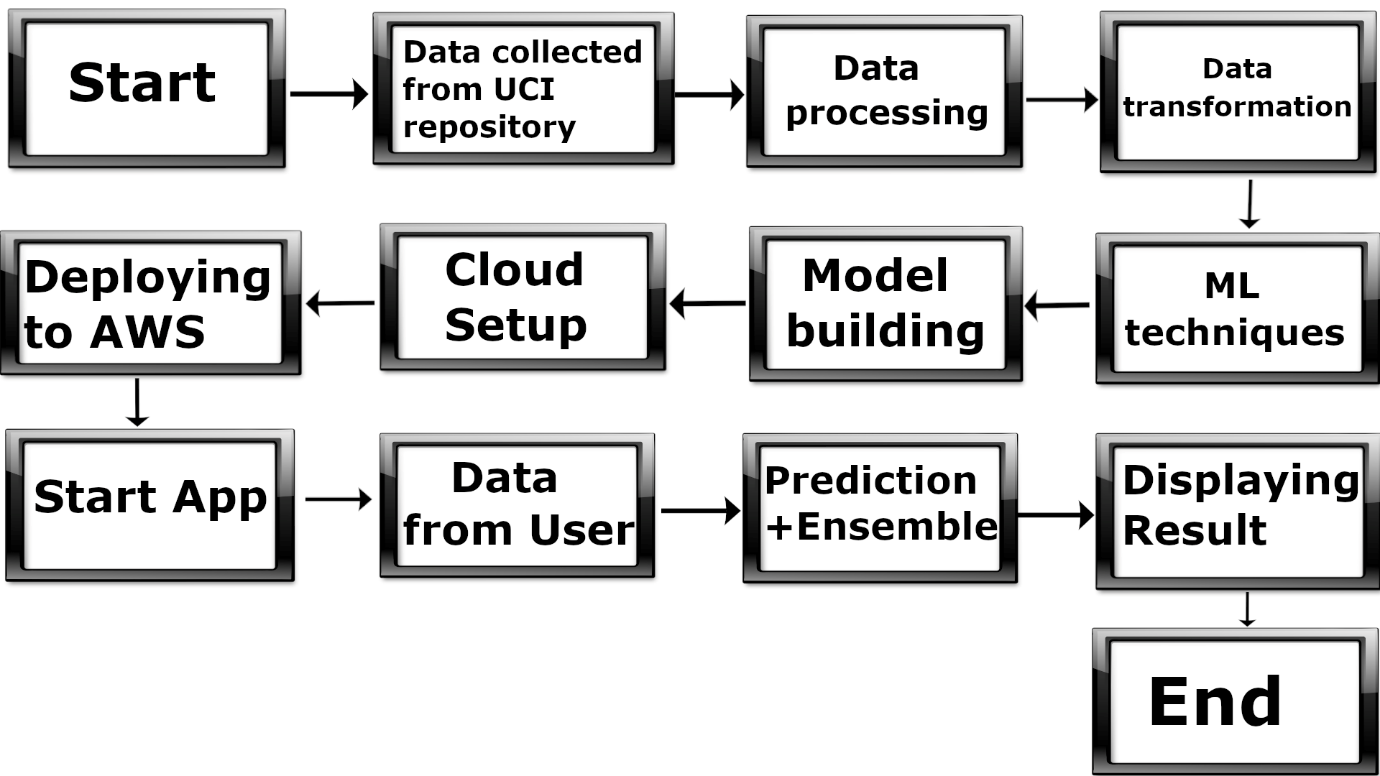
* 1. What is Low-Level design document?

The goal of LLD or a low-level design document is to give the internal logical design of the actual program code for Thyroid Disease Detection. 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. Scope

Low-level design 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



1. Architecture Design
   1. Data Description

This directory contains 6 databases, corresponding test set, and corresponding documentation. They were left at the University of California at Irvine by Ross Quinlan during his visit in 1987 for the 1987 Machine Learning Workshop.

* 1. Data Collection

The dataset read through the link address directly from the UCI repository page.

* 1. Data Processing

In this process, raw data is transformed into valid form for the machine learning models. This includes, imputing missing values, removing outliers, creating new features, etc.

* 1. Data transformation

The data is rescaled into the range of 0 and 1. This is done to make the data simple and put more weightage to outliers.

3.5 Model Building

Multiple models were trained and evaluated based on accuracy score and recall. The best performing models were selected as final models. In our case, they were Random Forest and XGBoost.

* 1. Data from user

Here we will collect patient’s TSH, FTI, TT4 and Pregnant\_t.

3.7 Model call

Once predict button is pressed, both the models will predict and then the result will be ensembled and presented to the user.

1. 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 user is able to see input fields on logging in | 1. Application is  accessible  2. User is signed up 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 | 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 | User should get Submit button to submit the inputs |
| Verify whether user is presented with recommended results on clicking submit | 1. Application is  accessible  2. User is signed up to the application | 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  2. User is signed up to the application | The recommended results should be in accordance to the selections user made |