**Low-Level Design**

**Thyroid Disease Detection**

|  |  |
| --- | --- |
| Developed by | Saurabh Gupta |
| Version | 1.0 |
| Date | 24-06-2023 |

**Table of Contents**

[1 Document Change/History Control 3](#_Toc139448068)

[2 Reviews: 3](#_Toc139448069)

[3 Approval Status: 3](#_Toc139448070)

[4 Introduction 4](#_Toc139448071)

[4.1 What is Low-Level Design Document. 4](#_Toc139448072)

[4.2 Scope 4](#_Toc139448073)

[5 Architecture 4](#_Toc139448074)

[6 Architecture Description 5](#_Toc139448075)

[6.1 Raw Data Validation 6](#_Toc139448076)

[6.2 Data Transformation 7](#_Toc139448077)

[6.3 Data Preprocessing 7](#_Toc139448078)

[6.4 Feature Engineering 7](#_Toc139448079)

[6.5 Balance the data set by using SMOTE 7](#_Toc139448080)

[6.6 Parameter Tuning 8](#_Toc139448081)

[6.7 Model Building 8](#_Toc139448082)

[6.8 Model Saving 8](#_Toc139448083)

[6.9 GitHub 8](#_Toc139448084)

[6.10 Deployment 8](#_Toc139448085)

[7 Unit Test Cases. 9](#_Toc139448086)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Done By** | **Review By** | **Remarks** |
| 1.0 | 24-06-2023 | Saurabh Gupta |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Document Change/History Control

# Reviews:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Remarks** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review Date** | **Reviewed By** | **Approved By** | **Remarks** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Introduction

## What is Low-Level Design Document.

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for **‘Stores Sales Prediction’**. 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.

## Scope

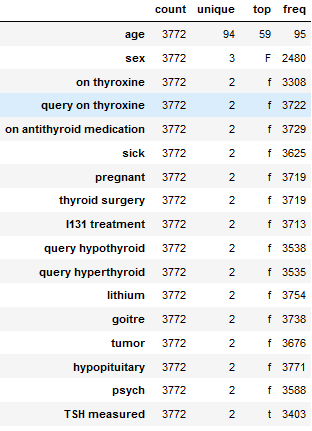
Low-level design (LLD) is a component-level design process that follows a step-by-step [refinement](https://en.wikipedia.org/wiki/Refinement_(computing)) process. This process can be used for designing data structures, required software architecture, source code, and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

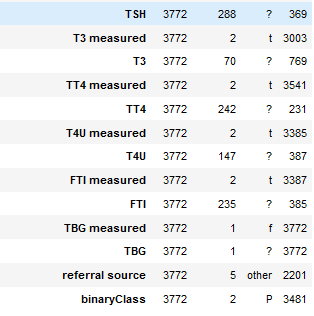
# Architecture



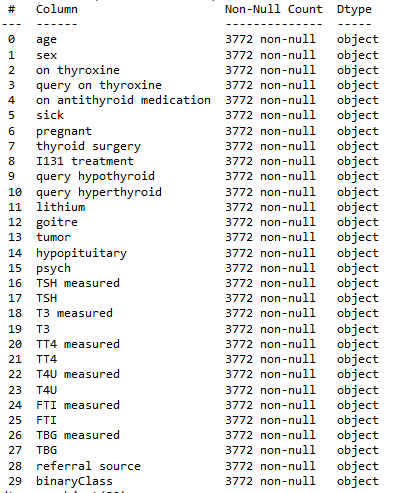
# Architecture Description

The dataset looks like as follow:





The data set consists of object data type shown in Fig.



## 6.1 Raw Data Validation

After data is loaded, various types of validation are required before we proceed further with any operation. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because The attributes which contain these are of no use. It will not play role in contributing to the sales of an item from respective outlets.

Like if any attribute is having zero standard deviation, it means that’s all the values are the same, its mean is zero. This indicates that either the sale is increasing or decrease that attribute will remain the same. Similarly, if any attribute is having full missing values, then there is no use in taking that attribute into an account for operation. It’s unnecessary increasing the chances of dimensionality curse.

## Data Transformation

Before sending the data into the database, data transformation is required so that data are converted into such form with which it can easily insert into the database. In the raw data, there can be various columns of underlying patterns which also gives an in-depth knowledge about the subject of interest and provides insights into the problem. But caution should be observed with respect to data as it may contain null values, or redundant values, or various types of ambiguity, which also demands pre-processing of data.

## Data Preprocessing

Preprocessing of this dataset includes doing analysis on the independent variables like checking for null values in each column and then replacing or filling them with supported appropriate data types so that analysis and model fitting is not hindered from their way to accuracy. Shown above are some of the representations obtained by using Pandas tools which tell about variable count for numerical columns and model values for categorical columns. Maximum and minimum values in numerical columns, along with their percentile values for median, play an important factor in deciding which value to be chosen at priority for further exploration tasks and analysis. Data types of different columns are used further in label processing and a one-hot encoding scheme during the model building.

## Feature Engineering

After preprocessing it was found that some of the attributes are not important to the thyroid detection for the particular outlet. So those attributes are removed. Even one hot encoding is also performed to convert the categorical features into numerical features.

## Balance the data set by using SMOTE

from imblearn.over\_sampling import SMOTE

sm = SMOTE(random\_state = 2)

x\_train\_res, y\_train\_res = sm.fit\_resample(x\_train, y\_train.ravel())

## Parameter Tuning

Parameters are tuned using GridSearchCV. Two algorithms are used in this problem, Logistic Regression, Random Forest Classifier and XGB Classifier. The parameters of these 3 algorithms are tuned and passed into the model.

## Model Building

After doing all kinds of preprocessing operations mention above and performing scaling and hyperparameter tuning, the data set is passed into 3 models, Logistic Regression, Random Forest Classifier and XGB Classifier. It was found that Random Forest Classifier and XGB Classifier performs best with the 99%.

## Model Saving

Model is saved using pickle library in `.sav` format.

## GitHub

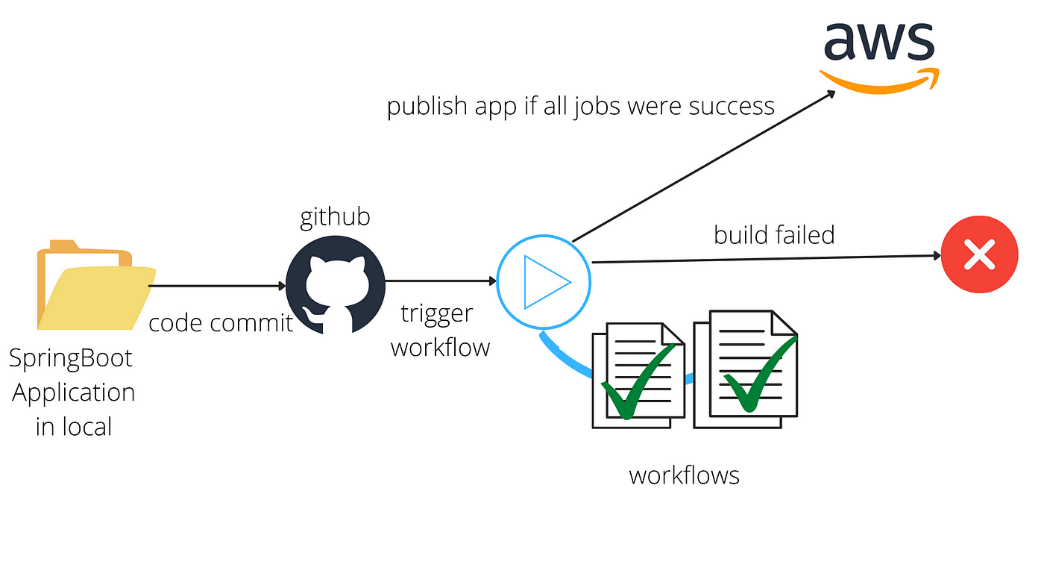
The whole project directory will be pushed into the GitHub repository.

Github: <https://github.com/saurabhg2083/Thyroid_disease_detection>

## Deployment

The cloud environment (AWS Elastic Bean Stack) was set up and the project was deployed from GitHub into the AWS.

App link: <http://thyroid-detection.eba-2nqj3nkm.ap-south-1.elasticbeanstalk.com/>



# Unit Test Cases.

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application URL is accessible to the user | 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 | Application URL is accessible  Application is deployed | The Application should load completely for the user when the URL is accessed |
| Verify whether a user is able to see input fields while opening the application | The user is able to see the input fields | Users should be able to see input fields on logging in |
| Verify whether a user is able to enter the input values. | The user is able to see the input fields | The user should be able to fill the input field |
| Verify whether a user gets predict button to submit the inputs | The user is able to see the input fields | Users should get Submit button to submit the inputs |
| Verify whether a user is presented with recommended results on clicking submit | The user is able to see the input fields.  The user is able to see the submit button | Users should be presented with recommended results on clicking submit |
| Verify whether a result is in accordance with the input that the user has entered | The user is able to see the input fields.  The user is able to see the submit button | The result should be in accordance with the input that the user has entered |