**LOW LEVEL DOCUMENT (LLD)**

Insurance Premium Prediction

**by Himanshu balodi**



**Document Version Control**

Date 04 /04/2023

Version 1.0

Description Initial

Author Himanshu balodi

**Contents**

**1 Introduction**

1.1 What is Low-Level Design Document?

1.2 Scope

**2. Architecture**

**3. Architecture Design**

3.1 Data Requirement.

3.2 Data Collection.

3.3 About the dataset.

3.4 Data Description

3.5 Tools / Software Used.

3.6 Importing data into databases

3.7 Exporting data from the databases

3.8 Data pre-processing

3.9 Modeling

3.10 UI integration

3.11 Data from user

3.12 Data validation

3.13 Rendering the results

**4. Deployment**

**1 Introduction**

**1.1 Why High-Level Design Document?**

The goal of LLD or a low-level design document (LL DD) is to give the internal logical design of the actual program code for Thyroid disease detection System. 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.

**1.2 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 ARCHITECTURE DESIGN**

This project is completely based on the life cycle of machine learning, where we will be predicting the thyroid disease classification. The tools used in this project are Python, Pandas, NumPy, Matplotlib, Seaborn, Scikit learn,

For the Version Control system Git was used and for deployment AWS (amazon web service) was used.

**3.2 Data Requirement.**

Whenever we are working on any project the data is completely dependent on the requirement of the problem statement. For this project the problem statement was to create a Xgboost classification machine learning model which can predict the thyroid disease predictions

**3.3 Data Collection.**

The data which is used in this project was taken from Kaggle

Dataset link: https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction

**3.4 About the dataset.**

File Contain.

- Readme.txt

**.**Thyroid dataset total columns 30 and total raw are 3772.

**3.5 Data Description.**

Below are the various attributes for the Thyroid Dataset. Please go through the Insuranc e\_dataset for better clarity of the attributes when reading this document.

• **Medical Charges:** Total medical expense charged to the plan for the calendar year

• **Age:** Insurance contractor’s age, ranging from 18 to 64

• **Sex**: Insurance contractor’s gender, Male or Female

**• BMI (Body Mass Index)** = Body Mass Index, Weight(Kg)/( Height(m)² )

**• Children:** Number of children covered by the plan/ Number of dependants

**• Region:** The beneficiary’s residential area in the US. Northeast, Southeast, Northwest, Southwest

**• Smoker:** Whether the insurance contractor is a smoker or not, Yes or No

As mentioned, medical charges will be our dependent variable and the rest will be our independent variables.

**3.6 Tools / Software Used:-**

* Python version used for this project 3.6.1 or higher ( This may get updated and some features might not be available in new version. )
* Python libraries such as NumPy, pandas, Matplotlib, seaborn and scikit-learn ( Used for implementation of machine learning algorithms. )
* Jupyter Visual studio code is used as an IDE for writing the code.
* GitHub is used as the version control system.
* AWS is used for deployment.

**3.7 Importing data into the databases.**

MongoDB was used for loading the dataset using Pandas Library was used for training and making the machine-learning model.

**3.8 Exporting data to the database**

The data has been dumped to the MongoDB database..

**3.9 Data Preprocessing**

Have taken the dataset.csv file as my dataset.

* All the necessary libraries were imported first such as Numpy, Pandas, Matplotlib, and Seaborn.
* Checking the basic profile of the dataset. To get a better understanding of the dataset.
  + Using Info method
  + Using Describe method
  + Checking for unique values of each column.
  + Checking duplicated values
* Checking for null values, There are no null values present in our dataset.
* The categorical variable has been encoded with the help of a label encoder.
* After performing all the above steps, the dataset is ready and can be processed into the stage of modeling.

**3.10 Modeling**

* After this the data was split into 2 sets X and y. X contains all the columns except the target column in our case ( expenses), and y contains only the Target column.
* Using train test split we first split the dataset into X\_train, X\_test, y\_train, and y\_test.
* Standard scaling has been used to bring the data on the same scale
* The following libraries were imported to create Classification models.
  + We used different – different machine learning algorithm like Random forest, Decision tree, logistic Regression and K-NeighborsClassifier, XGBClassifier, AdaBoost Classifier
  + After applying above all the algorithm XGBClassifier has been chosen as a final algorithm to create the model

**3.11 UI Integration**

Apache Airflow can be used to monitor the model and predict the new batch dataset. A Flask webapp has been created to get the Insurance predicition based on certain inputs like age, sex, etc

WebApp link: <https://brown-analyst-flfci.ineuron.app:8080/>

**3.12 Data from the user**

User can give the required input and get the thyroid disease as a result in Flask webapp. Data from the user is retrieved using the batch file and using Apache Airflow our machine learning model to give the predicted result.

**3.13 Data Validation**

The data which is entered by the user is validated by the data\_validation.py file which is built using inside the components folder under thyroid and then this data is transformed using data\_transformation.py under the same path and finally transferred to our model.

**3.14 Rendering the result**

The result for the predictions can be obtained in the and also result for our model can be seen in the prediction file generated after running the batch prediction in Apache Airflow.

**3.15 Batch Prediction**

we will have s3 bucket . In s3 bucket just consider it we upload the file , so we create a folder input files . batch\_prediction throw air flow we will first download this file & than run our batch prediction job and than we will generate prediction file.



**4.Deployment**

This model is deployed on AWS Ec2 instances. The following are the steps to deploy the model on the AWS platform:

* Create an ECR
* Create S3 bucket
* Create an AWS account
* Create an EC2 instance
* Edit security group
* Connect to an EC2 instance
* Install Docker
* Add the runner in the GitHub
* Add all the secret keys in the GitHub
* In the GitHub actions, run the continuous delivery and deployment workflow once after starting the runner in the EC2 instance
* A web app has been created and deployed using Streamlit

