Architecture Design

Thyroid Disease Detection

Developed by	Saurabh Gupta
Version	1.0
Date	12-06-2023

Table of Contents

1 Document Change/History Control 3

2 Approval Status: 3

1 Document Change/History Control

Version	Date	Done By	Review By	Remarks
1.0	07-11-2021	Saurabh Gupta		

2 Approval Status:

Version	Review Date	Reviewed By	Approved By	Remarks

3 Abstract

Machine Learning is a category of algorithms that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build models and employ algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. These models can be applied in different areas and trained to match the expectations of management so that accurate steps can be taken to achieve the organization's target. In this paper, Thyroid disease is a common cause of medical diagnosis and prediction, with an onset that is difficult to forecast in medical research. The thyroid gland is one of our body's most vital organs. Thyroid hormone releases are responsible for metabolic regulation. Hyperthyroidism and hypothyroidism are one of the two common diseases of the thyroid that releases thyroid hormones in regulating the rate of body's metabolism. The main goal is to predict the estimated risk on a patient's chance of obtaining thyroid disease or not.

4 Introduction

4.1 What is Architecture Design?

The goal of Architecture Design (AD) or a low-level design document is to give the internal design of the actual program code for the ` Thyroid Disease Prediction System`. AD describes the class diagrams with the methods and relation between classes and program specification. It describes the modules so that the programmer can directly code the program from the document.

4.2 Scope

Architecture Design (AD) 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 and the complete workflow.

4.3 Constraints

We only predict the expected casual prediction by using thyroid hormones and body metabolism of the patient to predict the estimated risk on a patient's chance of obtaining thyroid disease or not.

5 Technical Specification

5.1 Dataset

Dataset Characteristics: Multivariate, Domain-Theory

Subject Area: Life

Associated Tasks: Classification

The dataset looks like as follow:

	count	unique	top	freq
age	3772	94	59	95
sex	3772	3	F	2480
on thyroxine	3772	2	f	3308
query on thyroxine	3772	2	f	3722
on antithyroid medication	3772	2	f	3729
sick	3772	2	f	3625
pregnant	3772	2	f	3719
thyroid surgery	3772	2	f	3719
I131 treatment	3772	2	f	3713
query hypothyroid	3772	2	f	3538
query hyperthyroid	3772	2	f	3535
lithium	3772	2	f	3754
goitre	3772	2	f	3738
tumor	3772	2	f	3676
hypopituitary	3772	2	f	3771
psych	3772	2	f	3588
TSH measured	3772	2	t	3403

TSH	3772	288	?	369
T3 measured	3772	2	t	3003
T3	3772	70	?	769
TT4 measured	3772	2	t	3541
TT4	3772	242	?	231
T4U measured	3772	2	t	3385
T4U	3772	147	?	387
FTI measured	3772	2	t	3387
FTI	3772	235	?	385
TBG measured	3772	1	f	3772
TBG	3772	1	?	3772
referral source	3772	5	other	2201
binaryClass	3772	2	P	3481

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

```
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   age                        3772 non-null   object
 1   sex                        3772 non-null   object
 2   on thyroxine               3772 non-null   object
 3   query on thyroxine         3772 non-null   object
 4   on antithyroid medication  3772 non-null   object
 5   sick                       3772 non-null   object
 6   pregnant                   3772 non-null   object
 7   thyroid surgery            3772 non-null   object
 8   I131 treatment             3772 non-null   object
 9   query hypothyroid          3772 non-null   object
 10  query hyperthyroid         3772 non-null   object
 11  lithium                    3772 non-null   object
 12  goitre                     3772 non-null   object
 13  tumor                      3772 non-null   object
 14  hypopituitary              3772 non-null   object
 15  psych                      3772 non-null   object
 16  TSH measured               3772 non-null   object
 17  TSH                        3772 non-null   object
 18  T3 measured                3772 non-null   object
 19  T3                         3772 non-null   object
 20  TT4 measured               3772 non-null   object
 21  TT4                        3772 non-null   object
 22  T4U measured               3772 non-null   object
 23  T4U                        3772 non-null   object
 24  FTI measured               3772 non-null   object
 25  FTI                        3772 non-null   object
 26  TBG measured               3772 non-null   object
 27  TBG                        3772 non-null   object
 28  referral source            3772 non-null   object
 29  binaryClass                3772 non-null   object
```

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.

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.

Replace “?” with

```
df['TSH']=df['TSH'].replace({"?":0})
df['T3']=df['T3'].replace({"?":0})
df['TT4']=df['TT4'].replace({"?":0})
df['T4U']=df['T4U'].replace({"?":0})
df['FTI']=df['FTI'].replace({"?":0})
```

Change Target class into 0 and 1

```
df["binaryClass"]=df["binaryClass"].map({"P":0,"N":1})
```

5.2 Logging

We should be able to log every activity done by the user

- The system identifies at which step logging require.
- The system should be able to log each and every system flow.
- Developers can choose logging methods. Also can choose database logging.
- The system should be not be hung even after using so much logging. Logging just because we can easily debug issuing so logging is mandatory to do.

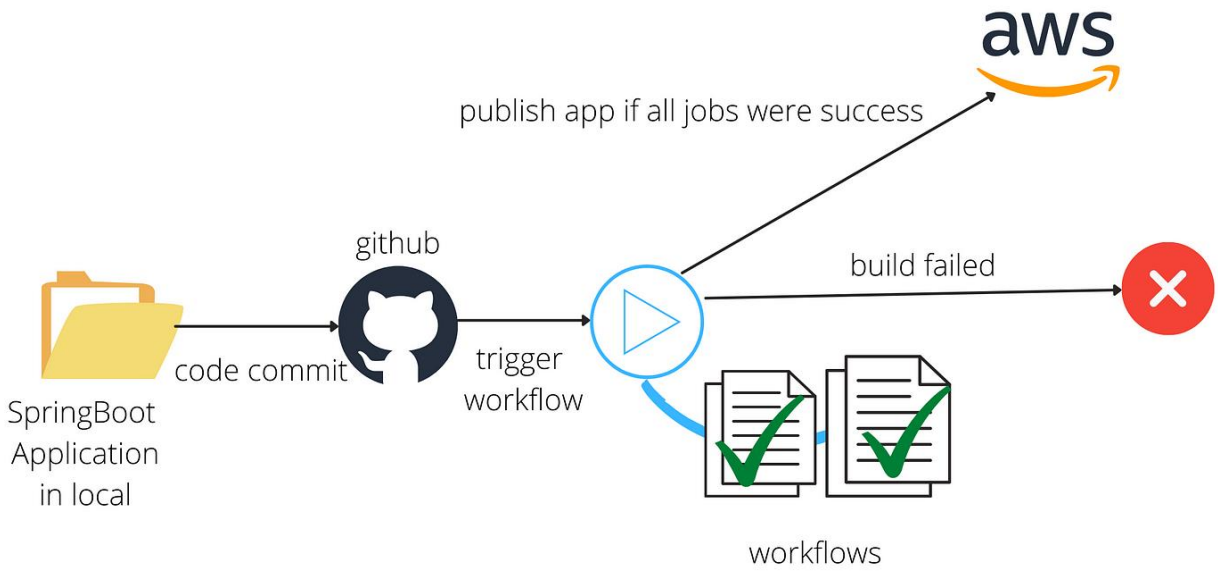
5.3 Database

The system needs to store every request into the database and we need to store it in such a way that it is easy to retain and look into the records.

The system should capture every data that any user gave and the prediction that has been made by that input.

2.4 Deployment

For the hosting of the project, we will use AWS Elastic beanstalk.



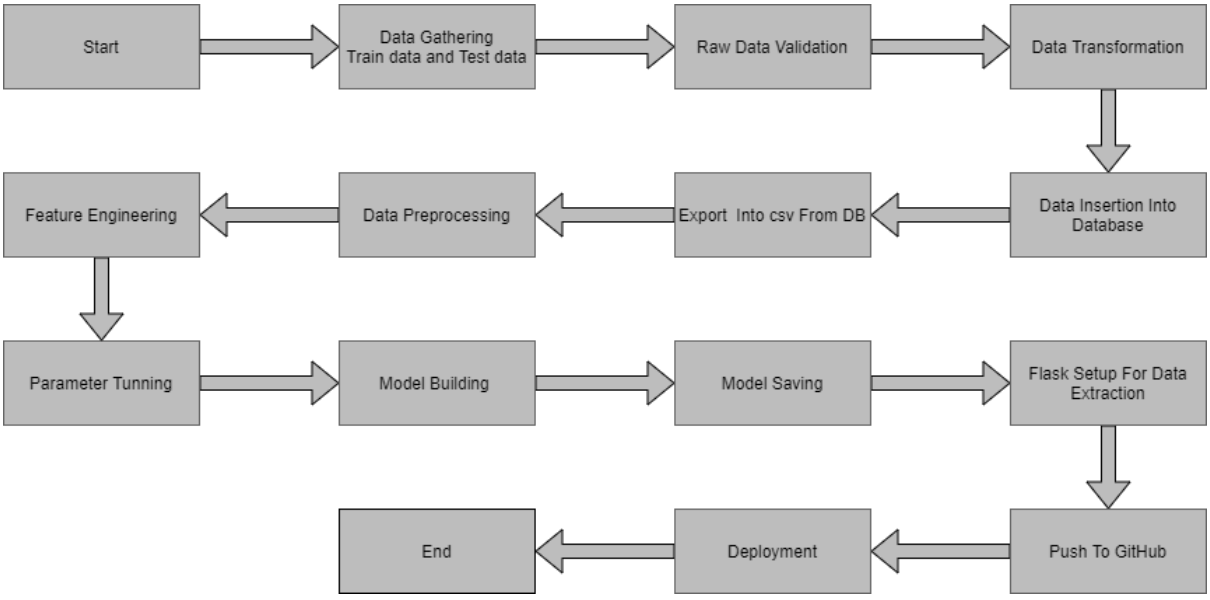
6 Technology Stack

Front End	HTML/JavaScript
Backend	Python
Deployment	AWS, Github

7 Proposed Solution

We will use performed EDA to find the important relation between different attributes and will use a machine-learning algorithm to predict the future sales demand. The client will be filled the required feature as input and will get results through the web application. The system will get features and it will be passed into the backend where the features will be validated and preprocessed and then it will be passed to a hyperparameter tuned machine learning model to predict the final outcome.

8 Architecture detail



8.1 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.

8.2 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.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(x_train)
x_train = sc.transform(x_train)
x_test = sc.transform(x_test)
```

8.3 Data Preprocessing

In data preprocessing all the processes required before sending the data for model building are performed.

Convert data to numeric datatypes

Columns which are converted are:

```
numerical_cols = ['age', 'sex', 'on thyroxine', 'query on thyroxine',
                  'on antithyroid medication', 'sick', 'pregnant', 'thyroid surgery',
                  'I131 treatment', 'query hypothyroid', 'query hyperthyroid', 'lithium',
                  'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH measured', 'TSH',
                  'T3 measured', 'T3', 'TT4 measured', 'TT4', 'T4U measured', 'T4U',
```

'FTI measured', 'FTI']

We implemented data pipeline to convert to fill missing value with median and mean then change to standard scaler.

```
num_pipeline=Pipeline(
    steps=[
    ('imputer',SimpleImputer(strategy='median')),
    ('scaler',StandardScaler())
    ]
)

preprocessor=ColumnTransformer([
('num_pipeline',num_pipeline,numerical_cols)
])
```

8.4 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())
```

8.5 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.

8.6 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%.

8.7 Model Saving

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

5.10 GitHub

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

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

5.11 Deployment

The cloud environment was set up and the project was deployed from GitHub into the AWS.

9 User Input / Output Workflow.