

**NAAN MUDHALVAN PROJECT  
Thyroid Disease Classification Using ML**

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE  
TO THE  
THIRUVALLUVAR UNIVERSITY, SERKKADU, VELLORE-632115  
BY  
MONIKA V ---35620U18022  
SATHISH N ---35620U18040  
RAJESH A ---35620U18031  
NIRMAL E ---35620U18026**



**APRIL-2023  
GOVERNMENT ARTS AND SCIENCE COLLEGE,  
ARAKKONAM-631051  
(AFFILIATED TO THIRUVALLUVAR UNIVERSITY)**

**GUIDED AND HEAD OF DEPARTMENT  
Dr. S. Selvakani**

S.NO	CONTENT
1.	INTRODUCTION
2.	MILESTONE 1: Define Problem /Problem Understanding <ul style="list-style-type: none"> <li>*Specify the business problem</li> <li>*Business requirements</li> <li>*Literature Survey</li> <li>*Social or Business Impact</li> </ul>
3.	MILESTONE 2: Data Collection & Preparation <ul style="list-style-type: none"> <li>* Download the dataset</li> <li>* Read The Dataset</li> <li>* Data Pre-processing</li> <li>* Checking for null values</li> <li>* Splitting the data x and y</li> <li>* Converting the Data Type</li> <li>* Handling Categorical Values</li> <li>* Splitting data into train and test</li> <li>* Handling Imbalanced Data</li> <li>* Applying StandardScaler</li> <li>* Performing Feature Importance</li> <li>* Selecting Output Columns</li> </ul>

4.	<p><b>MILESTONE 3: Exploratory Data Analysis</b></p> <ul style="list-style-type: none"> <li>* Descriptive analysis</li> <li>* Visual analysis</li> <li>* Checking Correlation</li> </ul>
5.	<p><b>MILESTONE 4: Model Building</b></p> <ul style="list-style-type: none"> <li>*Training the model in multiple algorithms</li> <li>*Random Forest Classifier Model</li> <li>*XGBClassifier model</li> <li>*SVC model</li> <li>* ANN Model</li> <li>*Testing the model</li> </ul>
6.	<p><b>MILESTONE 5: Performance Testing &amp; Hyperparameter Tuning</b></p> <ul style="list-style-type: none"> <li>*Testing model with multiple evaluation metrics</li> <li>*Compare the model</li> <li>*Comparing model accuracy before &amp; after applying hyperparameter tuning</li> </ul>
7.	<p><b>MILESTONE 6: Model Deployment</b></p> <ul style="list-style-type: none"> <li>* Save the best model</li> <li>* Integrate with Web Framework</li> <li>* Build Python code:</li> <li>* Run the application</li> </ul>
8.	<p><b>MILESTONE 7: Project Demonstration &amp; Documentation</b></p> <ul style="list-style-type: none"> <li>*Record explanation video for the project end to solution</li> <li>*Project Documentation-Step by step project development</li> </ul>

## INTRODUCTION

Campus recruitment is a strategy for sourcing, engaging and hiring young talent for internship and entry-level positions. College recruiting is typically a tactic for medium-to large-sized companies with high-volume recruiting needs, but can range from small efforts (like working with university career centers to source potential candidates) to large-scale operations (like visiting a wide array of colleges and attending recruiting events throughout the spring and fall semester). Campus recruitment often involves working with university career services centers and attending career fairs to meet in-person with college students and recent graduates. Our solution revolves around the placement season of a Business School in India. Where it has various factors on candidates getting hired such as work experience, exam percentage etc., Finally it contains the status of recruitment and remuneration details. We will be using algorithms such as KNN, SVM and ANN. We will train and test the data with these algorithms. From this the best model is selected and saved in . pkl format. We will be doing flask integration

# **Milestone 1: Define Problem / Problem Understanding**

## **Activity 1: Specify the business problem**

Refer to Project Description

## **Activity 2: Business requirements**

The business requirements for a machine learning model to predict thyroid disease include the ability to accurately predict thyroid disease based on the scan results, Minimise the number of false positives (wrong thyroid disease confirmations) and false negatives (thyroid is there but got as not thyroid disease). Provide an explanation for the model's decision, to comply with regulations and improve transparency.

## **Activity 3: Literature Survey (Student Will Write)**

The thyroid gland is one of the body's most visible endocrine glands. Its size is determined by the individual's age, gender, and physiological states, such as pregnancy or lactation. It is divided into two lobes (right and left) by an isthmus (a band of tissue). It is imperceptible in everyday life yet can be detected when swallowing. The thyroid hormones T4 and T3 are needed for normal thyroid function. These hormones have a direct effect on the body's metabolic rate. It contributes to the stimulation of glucose, fatty acid, and other molecule consumption. Additionally, it enhances oxygen consumption in the majority of the body's cells by assisting in the processing of uncoupling proteins, which contributes to an improvement in the rate of cellular respiration.

Thyroid conditions are difficult to detect in test results, and only trained professionals can do so. However, reading such extensive reports and predicting future results is difficult. Assume a machine learning model can detect the thyroid disease in a patient. The thyroid disease can then be easily identified based on the symptoms in the patient's history. Currently, models are evaluated using accuracy metrics on a validation dataset that is accessible.

## **Activity 4: Social or Business Impact.**

**Social Impact:-** Untreated/undetected thyroid disease is more dangerous at times it can lead to fatal of the person. So, we can detect it at the earliest then people can get treatment and get cured.

**Business Model/Impact:-** We can make this application public, offer services as a subscription based or can collaborate with healthcare centres or specialists.

## **Milestone 2: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

### **Activity 1: Download the dataset**

There are many popular open sources for collecting the data.  
Eg: kaggle.com, UCI repository, etc.

In this project, we have used drug200.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/prathamtripathi/drug-classification>

#### **Activity 1.1: Importing the libraries**

Import the necessary libraries as shown in the image.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Layer, Dense, Dropout
```

#### **Activity 1.2: Read the Dataset**

Our dataset format might be in .csv, excel files, .txt, .json, etc.

We can read the dataset with the help of pandas.

In pandas, we have a function called `read_csv()` to read the dataset. As a parameter, we have to give the directory of the csv file.

The screenshot shows a Jupyter Notebook interface. At the top, there is a code cell containing Python code to read a CSV file and display its head:data = pd.read\_csv("data.csv")
data.head()

```
Below the code cell is the output, which displays the first five rows of the dataset as a Pandas DataFrame. The columns are labeled: age, sex, on_thyroxine, query_on_thyroxine, on_antithyroid_meds, sick, pregnant, thyroid_surgery, I131_treatment, query_hypothyroid, ..., TT4, T4U_measur. The data shows various combinations of these variables for different individuals, with some values being 'f' (false) and others 't' (true). The output ends with the text "5 rows x 31 columns". At the bottom, another code cell shows the shape of the data: data.shape, resulting in (9172, 31).
```

## **Activity 2: Data Pre-processing**

As we have understood how the data is, let's pre-process the collected

data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Descriptive analysis
- Splitting the dataset as x and y
- Handling Categorical Values
- Checking Correlation
- Converting Data Type
- Splitting dataset into training and test set
- Handled Imbalanced Data
- Applying StandardScaler

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

#### Activity 2.1: Checking for null values

- For checking the null values, `data.isnull()` function is used. To sum those null values we use the `.sum()` function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
data.isnull().sum()
```

```
age                      0
sex                     307
on_thyroxine             0
query_on_thyroxine       0
on_antithyroid_meds     0
sick                     0
pregnant                 0
thyroid_surgery           0
I131_treatment            0
query_hypothyroid         0
query_hyperthyroid        0
lithium                   0
goitre                    0
tumor                     0
hypopituitary              0
psych                     0
TSH_measured               0
TSH                      842
T3_measured                0
T3                       2604
TT4_measured                0
TT4                      442
T4U_measured                0
T4U                      809
FTI_measured                0
FTI                      802
TBG_measured                0
TBG                      8823
referral_source              0
target                     0
patient_id                  0
dtype: int64
```

- Removing the Redundant attributes from the dataset.

```
#Removing Redundant attributes from dataset
data.drop(['TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured', 'TBG_measured', 'referral_source', 'patient_id'], axis=1, inplace=True)
```

- Re-mapping the 'target' values to the diagnostic Group

```
#re-mapping target values to diagnostic group
diagnoses = {'A': 'hyperthyroid conditions',
             'B': 'hyperthyroid conditions',
             'C': 'hyperthyroid conditions',
             'D': 'hyperthyroid conditions',
             'E': 'hypothyroid conditions',
             'F': 'hypothyroid conditions',
             'G': 'hypothyroid conditions',
             'H': 'hypothyroid conditions',
             'I': 'binding protein',
             'J': 'binding protein',
             'K': 'general health',
             'L': 'replacement therapy',
             'M': 'replacement therapy',
             'N': 'replacement therapy',
             'O': 'antithyroid treatment',
             'P': 'antithyroid treatment',
             'Q': 'antithyroid treatment',
             'R': 'miscellaneous',
             'S': 'miscellaneous',
             'T': 'miscellaneous'}
data['target'] = data['target'].map(diagnoses) #remapping
```

## Dropping Null Values

```
data.dropna(subset=['target'], inplace=True)
```

```
data['target'].value_counts()
```

```
hypothyroid conditions    593  
general health          436  
binding protein          376  
replacement therapy      336  
miscellaneous            281  
hyperthyroid conditions  182  
antithyroid treatment    33  
Name: target, dtype: int64
```

Checking the 'age' is there any above 100 and we drop the age>100.

```
#Checking whether the age above 100  
data[data.age>100]
```

```
age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  l131_treatment  query_hypothyroid  ...  tumor  hypopituitar  
0 rows × 23 columns
```

### Activity 2.2: Splitting the data x and y

#### Splitting the data x and y

```
#splitting the data values as x and y  
x=data.iloc[:,0:-1]  
y= data.iloc[:, -1]
```

```
x
```

```
age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  l131_treatment  query_hypothyroid  ...  goitre  tumor  
4  32.0  F      f          f      f      f      f      f      f      f      f      f      f  
18 63.0  F      t          f      f      t      f      f      f      f      f      f      f  
32 41.0  M      f          f      f      f      f      f      f      f      f      f      f  
33 71.0  F      t          f      f      f      f      f      f      f      f      f      f  
39 55.0  F      t          f      f      f      f      f      f      f      f      t      f      f  
...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  
9153 64.0  M      f          f      f      f      f      f      f      f      f      f      f  
9157 60.0  M      f          f      t      f      f      f      f      f      f      f      f  
9158 64.0  M      f          f      f      f      f      f      f      f      t      f      f  
9162 36.0  F      f          f      f      f      f      f      f      f      f      f      f  
9169 69.0  M      f          f      f      f      f      f      f      f      f      f      f  
2237 rows × 22 columns
```

- Making 'F' on wherever we have the 'nan' values on data.

```
x['sex'].unique()
```

```
array(['F', 'M', nan], dtype=object)
```

```
x['sex'].replace(np.nan, 'F', inplace=True)
```

```
x['sex'].value_counts()
```

```
F    1781  
M    536  
Name: sex, dtype: int64
```

### Activity 2.3: Converting the Data Type

Converting the data type from object to float. So that we will get output properly. And Checking info about data.

- Here, we have the object values are 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG'and convert them to float values.

```
x['age']=x['age'].astype('float')  
x['TSH']=x['TSH'].astype('float')  
x['T3']=x['T3'].astype('float')  
x['TT4']=x['TT4'].astype('float')  
x['T4U']=x['T4U'].astype('float')  
x['FTI']=x['FTI'].astype('float')  
x['TBG']=x['TBG'].astype('float')
```

- Then we can check the datatype information about the dataset by code `ofx.info()`

#### Activity 2.4: Handling Categorical Values

```
x.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2237 entries, 4 to 9169
Data columns (total 22 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   age         2237 non-null    float64
 1   sex          2237 non-null    int64  
 2   on_thyroxine 2237 non-null    int64  
 3   query_on_thyroxine 2237 non-null    int64  
 4   on_antithyroid_meds 2237 non-null    int64  
 5   sick         2237 non-null    int64  
 6   pregnant     2237 non-null    int64  
 7   thyroid_surgery 2237 non-null    int64  
 8   T131_treatment 2237 non-null    int64  
 9   query_hypothyroid 2237 non-null    int64  
 10  query_hyperthyroid 2237 non-null    int64  
 11  lithium      2237 non-null    int64  
 12  goitre       2237 non-null    int64  
 13  tumor        2237 non-null    int64  
 14  hypopituitary 2237 non-null    int64  
 15  psych         2237 non-null    int64  
 16  TSH          2237 non-null    object 
 17  T3           2237 non-null    object 
 18  TT4          2237 non-null    object 
 19  T4U          2237 non-null    object 
 20  FTI          2237 non-null    object 
 21  TBG          2237 non-null    object 
dtypes: float64(1), int64(15), object(6)
memory usage: 482.0+ KB
```

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using Ordinal Encoding and Label Encoding.

- In our project, categorical features are x and y values.
- Here, applying Ordinal Encoding on x values.

```
#Encoding the categorical data
#Encoding the independent(output) variable
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
#categorical data

ordinal_encoder = OrdinalEncoder(dtype = 'int64')
x.iloc[:, 1:16] = ordinal_encoder.fit_transform(x.iloc[:, 1:16])
#ordinal_encoder.fit_transform(x[['sex']])
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	T131_treatment	query_hypothyroid	...	goitre	tumor
4	32.0	0	0	0	0	0	0	0	0	0	0	...	0
18	63.0	0	1	0	0	0	1	0	0	0	0	...	0
32	41.0	1	0	0	0	0	0	0	0	0	0	...	0
33	71.0	0	1	0	0	0	0	0	0	0	0	...	0
39	55.0	0	1	0	0	0	0	0	0	0	1	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9153	64.0	1	0	0	0	0	0	0	0	0	0	...	0
9157	60.0	1	0	0	1	0	0	0	0	0	0	...	0
9158	64.0	1	0	0	0	0	0	0	0	0	1	...	0
9162	36.0	0	0	0	0	0	0	0	0	0	0	...	0
9169	69.0	1	0	0	0	0	0	0	0	0	0	...	0

- Replacing the nan values with zero (0) values.

```
x.replace(np.nan, '0', inplace=True)
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	T131_treatment	query_hypothyroid	...	goitre	tumor
4	32.0	0	0	0	0	0	0	0	0	0	0	...	0
18	63.0	0	1	0	0	0	1	0	0	0	0	...	0
32	41.0	1	0	0	0	0	0	0	0	0	0	...	0
33	71.0	0	1	0	0	0	0	0	0	0	0	...	0
39	55.0	0	1	0	0	0	0	0	0	0	1	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9153	64.0	1	0	0	0	0	0	0	0	0	0	...	0
9157	60.0	1	0	0	1	0	0	0	0	0	0	...	0
9158	64.0	1	0	0	0	0	0	0	0	0	1	...	0
9162	36.0	0	0	0	0	0	0	0	0	0	0	...	0
9169	69.0	1	0	0	0	0	0	0	0	0	0	...	0

2237 rows × 22 columns

- Now, applying Label Encoding on y(Independent variable) value.

```
label_encoder = LabelEncoder()
y_dt = label_encoder.fit_transform(y)
```

```
y=pd.DataFrame(y_dt, columns=['target'])
```

y

	target
0	5
1	4
2	5
3	1
4	6
...	...
2232	2
2233	2
2234	1
2235	1
2236	1

2237 rows × 1 columns

#### Activity 2.5: Splitting data into train and test

Now let's split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, data is passed with dropping the target variable. And my target variable is passed. For splitting training and testing data we are using the `train_test_split()` function from sklearn. As parameters, we are passing x, y, test\_size, random\_state.

```
#splitting the data values as x and y
x=data.iloc[:,0:-1]
y= data.iloc[:, -1]
```

x

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	goitre	tumor
4	32.0	F	f	f	f	f	f	f	f	f	...	f	f
18	63.0	F	t	f	f	t	f	f	f	f	...	f	f
32	41.0	M	f	f	f	f	f	f	f	f	...	f	f
33	71.0	F	t	f	f	f	f	f	f	f	...	f	f
39	55.0	F	t	f	f	f	f	f	f	t	...	f	f
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9153	64.0	M	f	f	f	f	f	f	f	f	...	f	f
9157	60.0	M	f	f	t	f	f	f	f	f	...	f	f
9158	64.0	M	f	f	f	f	f	f	f	t	...	f	f
9162	36.0	F	f	f	f	f	f	f	f	f	...	f	f
9169	69.0	M	f	f	f	f	f	f	f	f	...	f	f

2237 rows × 22 columns

Activate Windows

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

```
from imblearn.over_sampling import SMOTE
y_train.value_counts()
```

target	471
4	471
2	351
1	382
6	265
5	230
3	144
0	26
	dtype: int64

#### Activity 2.6: Handling Imbalanced Data

```
os = SMOTE(random_state=0,k_neighbors=1)
x_bal,y_bal=os.fit_resample(x_train,y_train)
x_test_bal,y_test_bal=os.fit_resample(x_test,y_test)
```

### Activity 2.7: Applying StandardScaler

- Scaling the features makes the flow of gradient descent smooth and helps algorithms quickly reach the minima of the cost function.
- Without scaling features, the algorithm may be biased toward the feature which has values higher in magnitude. it brings every feature in the same range and the model uses every feature wisely.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_bal = sc.fit_transform(x_bal)
x_test_bal = sc.transform(x_test_bal)

x_bal
array([[-1.62721505, -0.44060477, -0.4238 , ..., -2.50870684,
       -1.40088079,  3.29445097],
       [-0.11561483, -0.44060477,  2.35960359, ..., -0.26259147,
        0.0720981 , -0.19494049],
       [ 1.1874903 ,  2.26960776, -0.4238 , ...,  0.17039463,
        -0.19352184, -0.19494049],
       ...,
       [ 1.395987 , -0.44060477,  2.35960359, ...,  0.43615031,
        0.06181022, -0.19494049],
       [ 0.72802783, -0.44060477,  2.35960359, ...,  0.143333 ,
        0.89086631, -0.19494049],
       [ 1.15628145, -0.44060477,  2.35960359, ...,  0.39723515,
        -0.26588659, -0.19494049]])
```

• Here, we have the data in array format and we are making it data frame(table format).

```
columns=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','...']

x_test_bal= pd.DataFrame(x_test_bal,columns=columns)

x_bal= pd.DataFrame(x_bal,columns=columns)

x_bal
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...
0	-1.627215	-0.440605	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
1	-0.115614	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
2	1.187490	2.269608	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
3	-1.366594	-0.440605	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
4	-0.167738	-0.440605	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
...	...	...	...	...	...	...	...	...	...	...	...
3292	0.546923	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
3293	0.383082	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
3294	1.395987	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
3295	0.728028	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...
3296	1.156281	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	-0.230986	...

3297 rows × 22 columns

### Activity 2.8: Performing Feature Importance

- The idea behind permutation feature importance is simple. The feature importance is calculated by noticing the increase or decrease in error when we permute the values of a feature.
- If permuting the values causes a huge change in the error, it means the feature is important for our model.

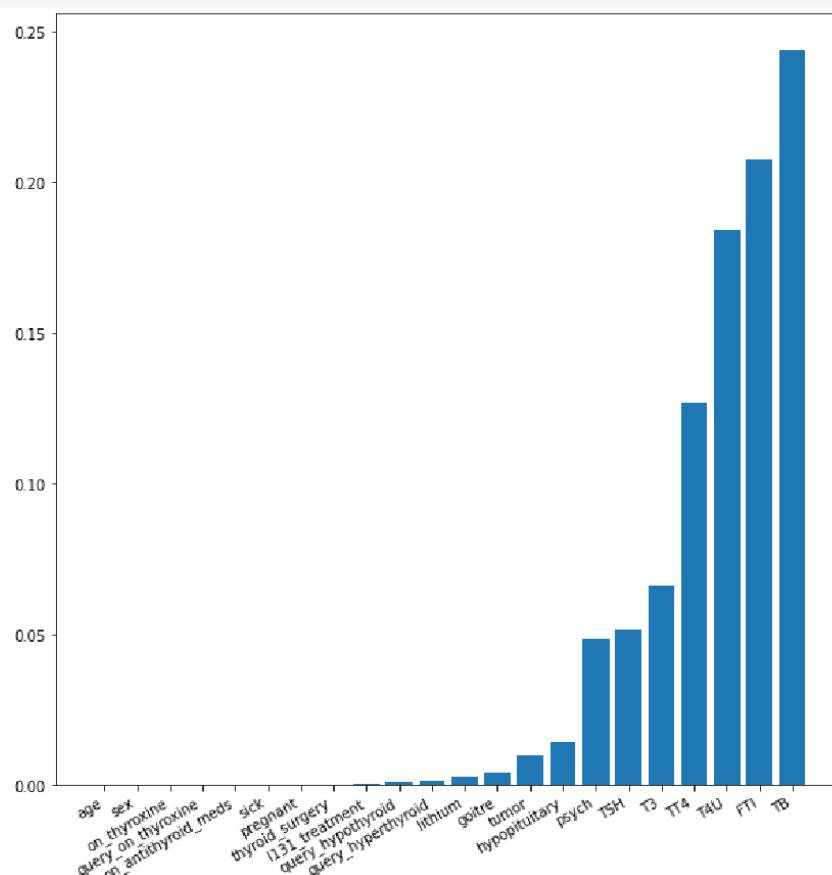
```
#perform feature importance
from sklearn.inspection import permutation_importance
results = permutation_importance(rfr,x_bal,y_bal, scoring='accuracy')
```

```

#gets importance
feature_importance=[‘age’, ‘sex’, ‘on_thyroxine’, ‘query_on_thyroxine’, ‘on_antithyroid_meds’, ‘sick’, ‘pregnant’, ‘thyroid_surgery’, ‘I131_treatment’, ‘query_hypothyroid’, ‘goitre’, ‘tumor’, ‘hypopituitary’, ‘psych’, ‘TSH’, ‘T3’, ‘T4U’, ‘FTI’, ‘TB’]
importance = results.importances_mean
importance = np.sort(importance)
#summerize feature importance
for i,v in enumerate(importance):
    i=feature_importance[i]
    print('feature: {:<20} Score: {}'.format(i,v))
#plot important feature
plt.figure(figsize=(10,10))
plt.bar(x=feature_importance, height = importance)
plt.xticks(rotation=90, ha='right')
plt.show()

```

feature	Score
age	0.0
sex	0.0
on_thyroxine	0.0
query_on_thyroxine	0.0
on_antithyroid_meds	6.066120715801926e-05
sick	0.0002426448286320705
pregnant	0.0003033060357900063
thyroid_surgery	0.0003033060357900063
I131_treatment	0.0006066120715801926
query_hypothyroid	0.0012132241431604518
query_hyperthyroid	0.0015165301789505925
lithium	0.0027904155292689968
goitre	0.00442826812253565
tumor	0.010009899181073733
hypopituitary	0.014376706096451319
psych	0.0488929329693661
TSH	0.05131938125568698
T3	0.06618137700940249
T4U	0.12684258416742494
T4U	0.1841067637245981
FTI	0.20752198968759478
TB	0.24416135881104034



## Activity 2.9: Selecting Output Columns

- Before we have this many columns

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	goitre	tumor	hypopituitary	psych	TSH	T3	T4U	FTI	TB
4	32.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	63.0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
32	41.0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
33	71.0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
39	55.0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	

5 rows × 22 columns

- After Performing Feature Importance by using 'Permutation Importance' we are dropping some columns which are not important for 'target'.

x_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_surgery','I131_treat
x_test_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_surgery','I131_

	goitre	tumor	hypopituitary	psych	TSH	T3	TT4	T4U	FTI	TBG
0	-0.052319	-0.137297	-0.024637	-0.107982	-0.315458	-1.035358	-1.704935	-2.508707	-1.400881	3.294451
1	-0.052319	-0.137297	-0.024637	-0.107982	-0.090056	0.155233	-0.197223	-0.262591	0.072098	-0.194940
2	-0.052319	-0.137297	-0.024637	-0.107982	-0.278907	-0.471394	-0.227079	0.170395	-0.193521	-0.194940
3	-0.052319	7.283487	-0.024637	-0.107982	-0.284999	0.969848	0.041622	0.495134	-0.133153	-0.194940
4	-0.052319	-0.137297	-0.024637	-0.107982	-0.306321	4.541622	1.459767	-0.127283	1.495783	-0.194940

## Milestone 3: Exploratory Data Analysis

### Activity 1: Descriptive analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this described function we can find mean, std, min, max and percentile values of continuous features.

Checking info about data by using data\_info()

```
data.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2237 entries, 4 to 9169
Data columns (total 23 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   age          2237 non-null   int64  
 1   sex          2147 non-null   object  
 2   on_thyroxine 2237 non-null   object  
 3   query_on_thyroxine 2237 non-null   object  
 4   on_antithyroid_meds 2237 non-null   object  
 5   sick         2237 non-null   object  
 6   pregnant     2237 non-null   object  
 7   thyroid_surgery 2237 non-null   object  
 8   I131_treatment 2237 non-null   object  
 9   query_hypothyroid 2237 non-null   object  
 10  query_hyperthyroid 2237 non-null   object  
 11  lithium      2237 non-null   object  
 12  goitre       2237 non-null   object  
 13  tumor        2237 non-null   object  
 14  hypopituitary 2237 non-null   object  
 15  psych        2237 non-null   object  
 16  TSH          2087 non-null   float64 
 17  T3           1643 non-null   float64 
 18  TT4          2148 non-null   float64 
 19  T4U          2059 non-null   float64 
 20  FTI          2060 non-null   float64 
 21  TBG          98 non-null    float64 
 22  target        2237 non-null   object  
dtypes: float64(6), int64(1), object(16)
memory usage: 419.4+ KB
```

### Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

#### Activity 2.1: Checking Correlation.

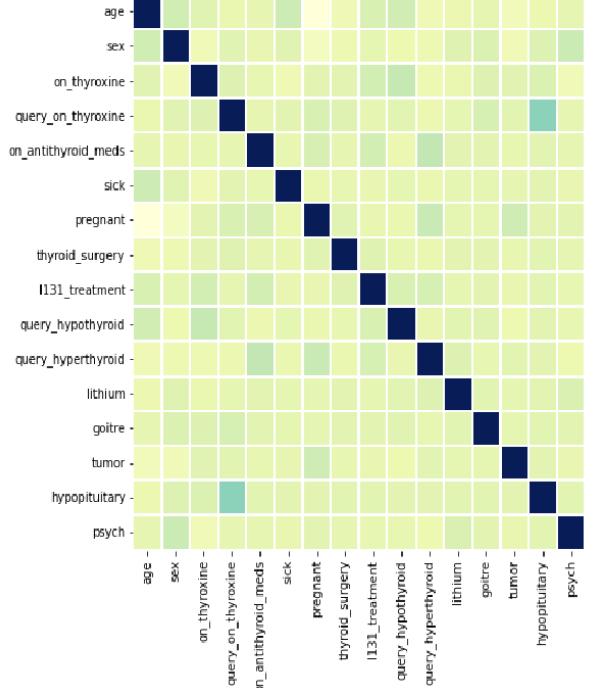
Here, I'm finding the correlation using HeatMap. It visualizes the data in 2-D coloured maps making use of colour variations. It describes the related variables in the form of colours instead of numbers; it will be plotted on both axes.

Here, there is no correlation between columns.

```
#checking correlation using Heatmap
import seaborn as sns
corrmat = x.corr()

f, ax = plt.subplots(figsize =(9, 8))
sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)
```

<AxesSubplot:>



Activate Windows

## Milestone 4: Model Building

### Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

#### Activity 1.1: Random Forest Classifier Model

A function named Random Forest Classifier Model is created and train and test data are passed as the parameters. Inside the function, the Random Forest Classifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, accuracy\_score and classification report is done.

```
from sklearn.ensemble import RandomForestClassifier
rfr1 = RandomForestClassifier().fit(x_os,y_os.values.ravel())
y_pred = rfr1.predict(x_test_os)
```

```
rfr1 = RandomForestClassifier()
```

```
+ RandomForestClassifier
RandomForestClassifier()
```

```
y_pred = rfr1.predict(x_test_os)
```

```
print(classification_report(y_test_os,y_pred))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	122
1	0.76	0.90	0.83	122
2	0.91	0.98	0.94	122
3	0.78	0.83	0.80	122
4	0.46	0.92	0.62	122
5	0.75	0.70	0.73	122
6	0.63	0.48	0.54	122

	precision	recall	f1-score	support
accuracy			0.69	854
macro avg	0.61	0.69	0.64	854
weighted avg	0.61	0.69	0.64	854

	precision	recall	f1-score	support
train_score = accuracy_score(y_os, rfr1.predict(x_os))				
train_score				

#### Activity 1.2: XGBClassifier model

A function named XGBClassifier model is created and train and test data are passed as the parameters. Inside the function, the XGBClassifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, the accuracy score and classification report is done.

```
from xgboost import XGBClassifier
xgb1 = XGBClassifier()
xgb1.fit(x_os,y_os)

XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing='nan', monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', random_state=0, reg_alpha=0, ...)

y_pred = xgb1.predict(x_test_os)

print(classification_report(y_test_os,y_pred))

precision    recall   f1-score   support
          0       0.70      0.13      0.22      122
          1       0.75      0.93      0.84      122
          2       0.95      0.99      0.97      122
          3       0.76      0.77      0.77      122
          4       0.48      0.85      0.61      122
          5       0.79      0.71      0.75      122
          6       0.62      0.52      0.57      122

accuracy                           0.70      854
macro avg       0.72      0.70      0.67      854
weighted avg    0.72      0.70      0.67      854

accuracy_score(y_test_os,y_pred)
0.7014051522248244
```

#### Activity 1.3: SVC model

A function named SVC model is created and train and test data are passed as the parameters. Inside the function, the SVC algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, the accuracy score and classification report is done.

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

sv= SVC()

sv.fit(x_bal,y_bal)

C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\sklearn\utils\validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)

SVC()

y_pred = sv.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

precision    recall   f1-score   support
          0       0.70      0.85      0.77      122
          1       0.76      0.81      0.79      122
          2       0.88      0.93      0.90      122
          3       0.71      0.65      0.68      122
          4       0.71      0.63      0.67      122
          5       0.76      0.54      0.63      122
          6       0.49      0.57      0.52      122

accuracy                           0.71      854
macro avg       0.72      0.71      0.71      854
weighted avg    0.72      0.71      0.71      854
```

```
train_score=accuracy_score(y_bal,sv.predict(x_bal))  
train_score
```

```
0.7154989384288747
```

#### Activity 1.4 ANN Model

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers

```
In [68]: model = Sequential()  
  
In [69]: model.add(Dense(units = 128, activation='relu', input_shape=(10,)))  
  
In [70]: model.add(Dense(units = 128, activation='relu', kernel_initializer='random_uniform'))  
model.add(Dropout(0.2))  
model.add(Dense(units = 256, activation='relu', kernel_initializer='random_uniform'))  
model.add(Dropout(0.2))  
model.add(Dense(units = 128, activation='relu', kernel_initializer='random_uniform'))  
  
In [71]: model.add(Dense(units = 1, activation='sigmoid'))  
  
In [72]: model.summary()  
Model: "sequential"  
  
Layer (type) Output Shape Param #  
===== ====== =====  
dense (Dense) (None, 128) 1408  
dense_1 (Dense) (None, 128) 16512  
dropout (Dropout) (None, 128) 0  
dense_2 (Dense) (None, 256) 33024  
dropout_1 (Dropout) (None, 256) 0  
dense_3 (Dense) (None, 128) 32896  
dense_4 (Dense) (None, 1) 129  
  
=====  
Total params: 83,969  
Trainable params: 83,969  
Non-trainable params: 0
```

```
In [73]: model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])  
  
In [75]: model.fit(x_bal,y_bal, validation_data=[x_test_bal, y_test_bal], epochs=15)  
Epoch 1/15  
104/104 [=====] - 9s 15ms/step - loss: 18416.0605 - accuracy: 0.1429 - val_loss: 142105.5156 - val_accuracy: 0.1429  
Epoch 2/15  
104/104 [=====] - 1s 8ms/step - loss: 2626274.5000 - accuracy: 0.1429 - val_loss: 10219054.0000 - val_accuracy: 0.1429  
Epoch 3/15  
104/104 [=====] - 1s 9ms/step - loss: 42823204.0000 - accuracy: 0.1429 - val_loss: 113329736.0000 - val_accuracy: 0.1429  
Epoch 4/15  
104/104 [=====] - 1s 9ms/step - loss: 277232128.0000 - accuracy: 0.1429 - val_loss: 582218880.0000 - val_accuracy: 0.1429  
Epoch 5/15  
104/104 [=====] - 1s 8ms/step - loss: 1097882752.0000 - accuracy: 0.1429 - val_loss: 1989677696.0000 - val_accuracy: 0.1429  
Epoch 6/15  
104/104 [=====] - 1s 8ms/step - loss: 3208519680.0000 - accuracy: 0.1429 - val_loss: 5285069824.0000 - val_accuracy: 0.1429  
Epoch 7/15
```

#### Activity 2: Testing the model

testing the models

```
In [115]: rfr1.predict([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])  
C:\Users\Mahidhar reddy\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names  
warnings.warn(  
Out[115]: array([4])  
  
In [130]: sv.predict([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])  
C:\Users\Mahidhar reddy\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but SVC was fitted with feature names  
warnings.warn(  
Out[130]: array([1])  
  
In [143]: col = ['goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG']  
da = [[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]]  
da1 = pd.DataFrame(data = da, columns=col)  
xgb1.predict(da1)  
Out[143]: array([4], dtype=int64)  
  
In [140]: model.predict([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])  
1/1 [=====] - 0s 238ms/step  
Out[140]: array([[1.]], dtype=float32)
```

## Milestone 5: Performance Testing & Hyperparameter Tuning

## Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test

set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

### Activity 1.1: Compare the model

For comparing the above four models, the compareModel function is defined.

```
: print(classification_report(y_test_bal,y_pred))

precision    recall   f1-score   support
0           0.87     0.16     0.28      122
1           0.82     0.94     0.87      122
2           0.93     0.98     0.96      122
3           0.77     0.84     0.80      122
4           0.49     0.89     0.63      122
5           0.88     0.68     0.77      122
6           0.59     0.53     0.56      122

accuracy                           0.72      854
macro avg                           0.76      0.70      0.70      854
weighted avg                          0.76      0.72      0.70      854

: train_score = accuracy_score(y_bal,rfr1.predict(x_bal))

: train_score
: 1.0

y_pred=xgb.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

precision    recall   f1-score   support
0           0.80     0.30     0.44      122
1           0.82     0.94     0.88      122
2           0.96     1.00     0.98      122
3           0.77     0.84     0.81      122
4           0.51     0.81     0.62      122
5           0.84     0.70     0.76      122
6           0.59     0.54     0.56      122

accuracy                           0.73      854
macro avg                           0.76      0.73      0.72      854
weighted avg                          0.76      0.73      0.72      854

train_score = accuracy_score(y_bal, xgb.predict(x_bal))
train_score
1.0
y_pred = sv.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

precision    recall   f1-score   support
0           0.70     0.85     0.77      122
1           0.76     0.81     0.79      122
2           0.88     0.93     0.90      122
3           0.71     0.65     0.68      122
4           0.71     0.63     0.67      122
5           0.76     0.54     0.63      122
6           0.49     0.57     0.52      122

accuracy                           0.71      854
macro avg                           0.72     0.71     0.71      854
weighted avg                          0.72     0.71     0.71      854

train_score=accuracy_score(y_bal,sv.predict(x_bal))
train_score
0.7154989384288747
```

```

y_pred = model.predict(x_test_bal)
27/27 [=====] - 0s 3ms/step

print(classification_report(y_test_bal,y_pred))

precision    recall    f1-score   support

          0       0.00      0.00      0.00     122
          1       0.14      1.00      0.25     122
          2       0.00      0.00      0.00     122
          3       0.00      0.00      0.00     122
          4       0.00      0.00      0.00     122
          5       0.00      0.00      0.00     122
          6       0.00      0.00      0.00     122

   accuracy                           0.14      854
  macro avg       0.02      0.14      0.04      854
weighted avg       0.02      0.14      0.04      854

C:\Users\Mahidhar reddy\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Mahidhar reddy\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Mahidhar reddy\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))

accuracy_score(y_test_bal,y_pred)

0.14285714285714285

```

## Activity 2:Comparing model accuracy before & after applying

### hyperparameter tuning

From sklearn, accuracy is used to evaluate the score of the model.

On the parameters, we have given xgb1 (model name), x, y, cv (as 3 folds). Our model is performing well. So, we are saving the model by pickle.dump().

Note: To understand cross validation, refer to this link.

[https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance\\_nce-e51e5430ff85](https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance_nce-e51e5430ff85).

```

params = {

    'C': [0.1, 1, 10, 100, 1000],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf', 'sqrt']

}

random_svc = RandomizedSearchCV(sv,params, scoring='accuracy',cv=5,n_jobs=-1)

random_svc.fit(x_bal,y_bal)

```

```

random_svc.best_params_
{'kernel': 'rbf', 'gamma': 1, 'C': 1}

sv1=SVC(kernel= 'rbf', gamma= 0.1, C= 100)

sv1.fit(x_bal,y_bal)

C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\sklearn\utils\validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)

SVC(C=100, gamma=0.1)

y_pred= sv1.predict(x_test_bal)

```

Activate Windows  
Go to Settings to activate Win

```

print(classification_report(y_test_bal,y_pred))

precision    recall  f1-score   support

          0       0.74      0.75      0.75      122
          1       0.77      0.86      0.81      122
          2       0.95      0.91      0.93      122
          3       0.70      0.66      0.68      122
          4       0.66      0.73      0.70      122
          5       0.72      0.72      0.72      122
          6       0.57      0.48      0.52      122

   accuracy                           0.73      854
  macro avg       0.73      0.73      0.73      854
weighted avg     0.73      0.73      0.73      854

```

```

train_score=accuracy_score(y_bal,sv1.predict(x_bal))
train_score
0.8125568698817106

```

## Saving the model as thyroid1\_model.pkl

```

# saving the model
import pickle
pickle.dump(sv1,open('thyroid_1_model.pkl','wb'))

features = np.array([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])
print(label_encoder.inverse_transform(xgb1.predict(features)))

['hypothyroid conditions']

```

## Here, we are saving label\_encoding also as label\_encoder.pkl

```

pickle.dump(label_encoder,open('label_encoder.pkl','wb'))

data['target'].unique()

array(['miscellaneous', 'hypothyroid conditions', 'binding protein',
       'replacement therapy', 'general health', 'hyperthyroid conditions',
       'antithyroid treatment'], dtype=object)

y['target'].unique()

array([5, 4, 1, 6, 2, 3, 0])

```

## Milestone 6: Model Deployment

### Activity 1:Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```

import pickle
pickle.dump(sv1,open('thyroid_1_model.pkl','wb'))

```

### Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is

integrated to the model we built. A UI is provided for the user where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script

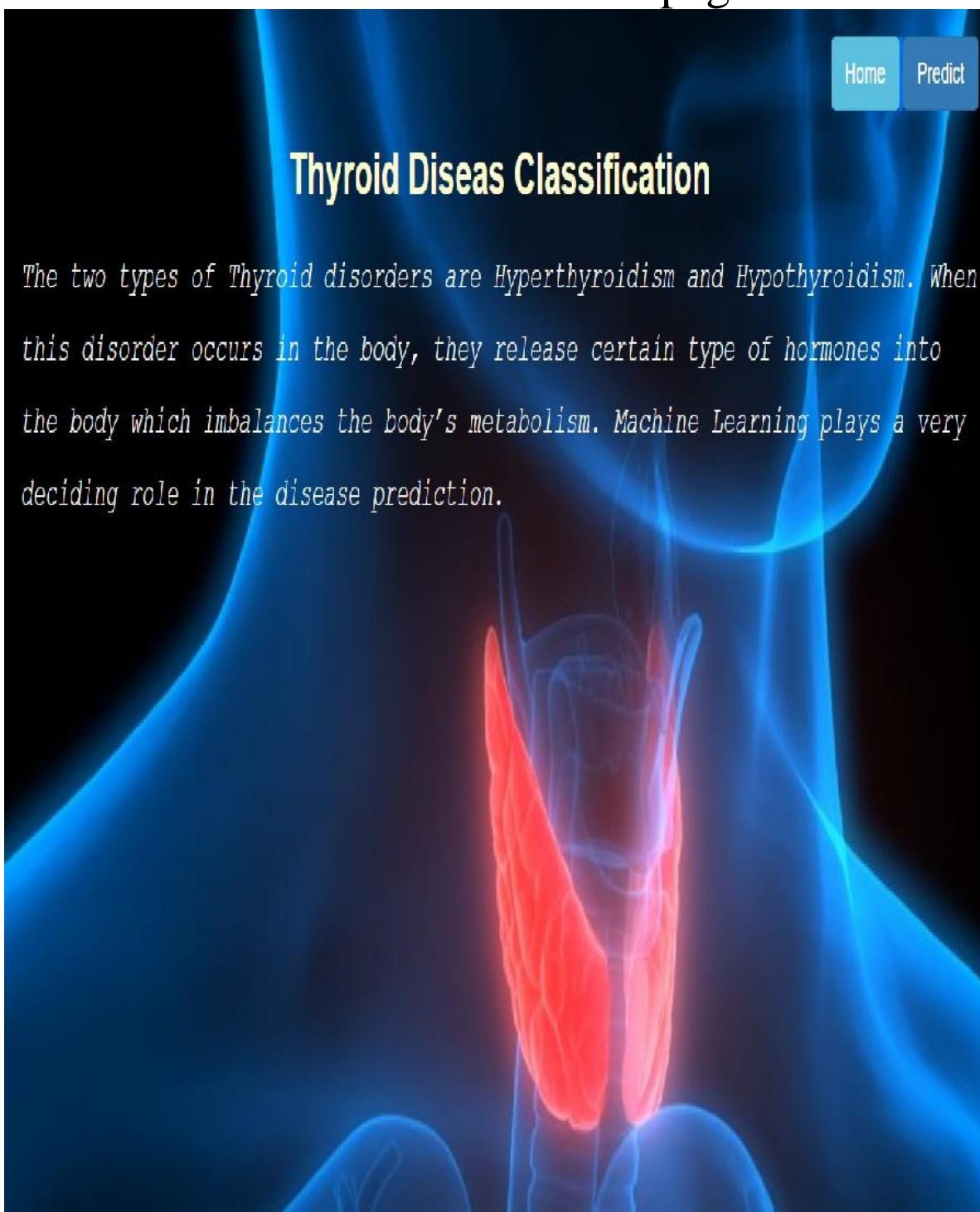
#### Activity 2.1: Building HTML Pages:

For this project create three HTML files namely

- home.html
- predict.html
- submit.html

and save them in the templates folder.

Let's see how our home.html page looks like:



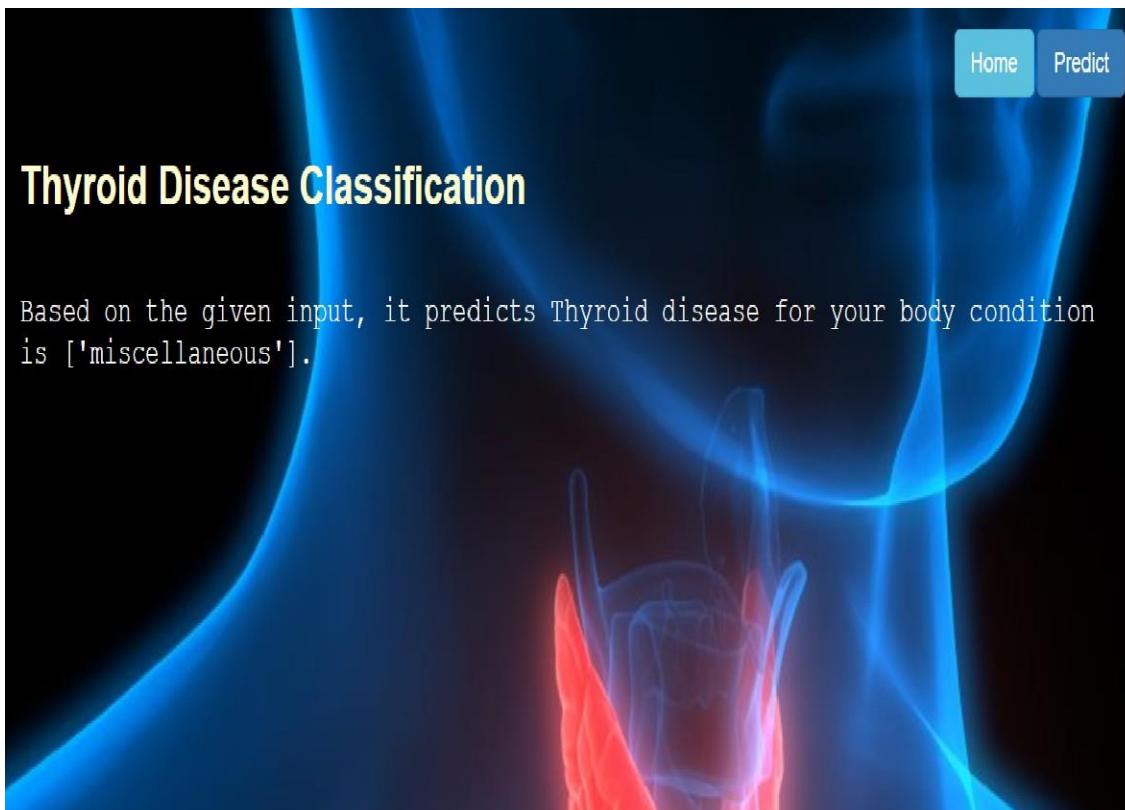
Now when you click on predict button from top right corner you will get redirected to predict.html

Let's look how our predict.html file looks like:

The image shows a web application titled "Thyroid Disease Classification". At the top right are "Home" and "Predict" buttons. On the left, there are dropdown menus for "goitre" (Male), "tumor" (Male), "hypopituitary" (Male), and "psych" (Male). Below these are input fields for "TSH", "T3", "TT4", "T4U", "FTI", and "TBG". A large, semi-transparent illustration of a human neck with a red thyroid gland is centered behind the form. At the bottom left is a green "Submit" button.

Now when you click on submit button from left bottom corner you will get redirected to submit.html

Let's look how our submit.html file looks like: it is ['miscellaneous'].



Activity 2.2: Build Python code:

Import the libraries

```
from flask import Flask, render_template, request
import numpy as np
import pickle
import pandas as pd
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (`_name_`) as argument.

```
model = pickle.load(open(r"C:\Users\SmartBridge-PC\Downloads\Thyroid\thyroid1_model.pkl", 'rb'))
le = pickle.load(open("label_encoder.pkl", 'rb'))

app = Flask(__name__)
```

Render HTML page:

```
@app.route("/")
def about():
    return render_template('home.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the `home.html` function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered.  
Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route("/pred", methods=['POST', 'GET'])
def predict():
    x = [[float(x) for x in request.form.values()]]

    print(x)
    col = ['goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG']
    x = pd.DataFrame(x, columns=col)

    #print(x.shape)

    print(x)
    pred = model.predict(x)
    pred = le.inverse_transform(pred)
    print(pred[0])
    return render_template('submit.html', prediction_text=str(pred))
```

Here we are routing our app to `predict()` function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the

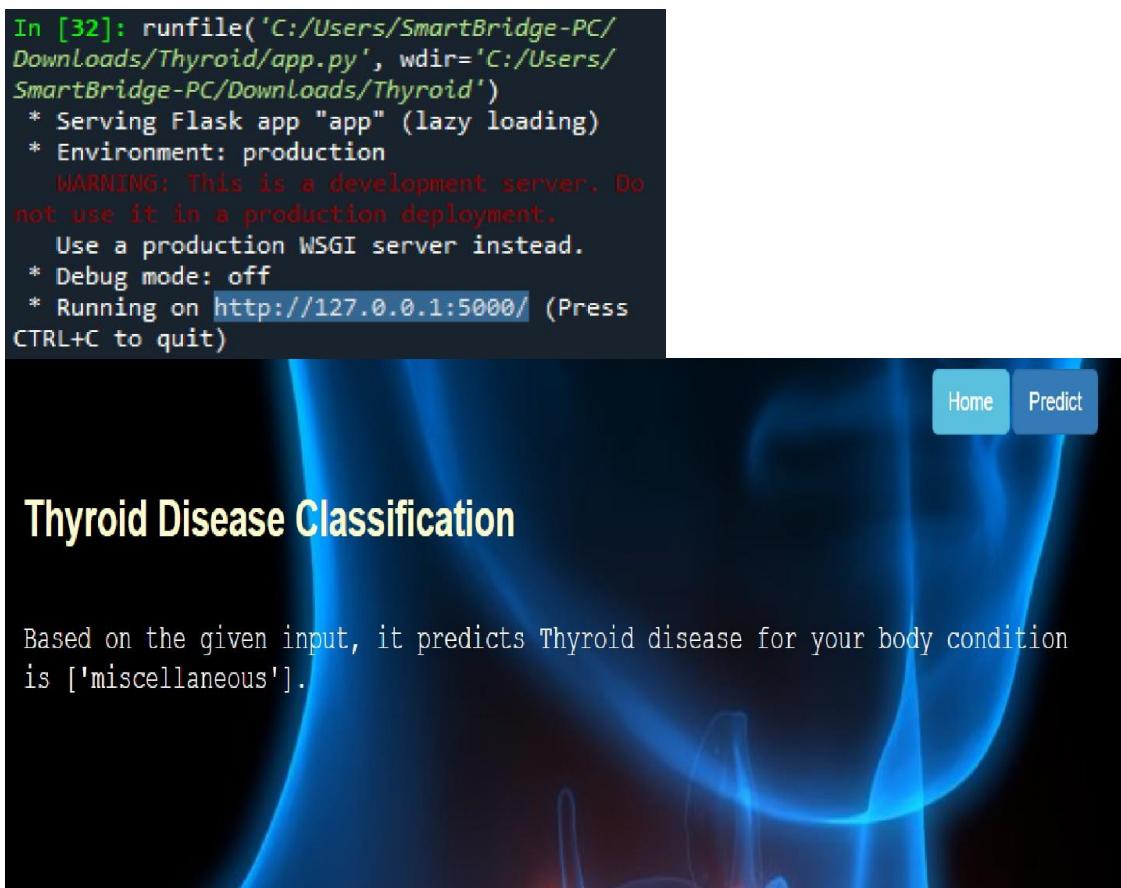
model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

### Main Function:

```
if __name__ == "__main__":
    app.run(debug=False)
```

#### Activity 2.3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



## **Milestone 7: Project Demonstration & Documentation**

Below mentioned deliverables to be submitted along with other deliverables

### **Activity 1:- Record explanation Video for the project end to end solution**

**Activity 2:- Project Documentation-Step by step project development procedure**

Create document as per the template provided