Thyroid Disease Classification Using Machine learning

ABSTRACT

With the vast amount of data and information difficult to deal with, especially in the health system, machine learning algorithms and data mining techniques have an important role in dealing with data. In our study, we used machine learning algorithms with thyroid disease. The goal of this study is to categorize thyroid disease into three categories: hyperthyroidism, hypothyroidism, and normal, so we worked on this study using data from Iraqi people, some of whom have an overactive thyroid gland and others who have hypothyroidism, so we used all of the algorithms. Support vector machines, random forest, decision tree, naive bayes, logistic regression, k-nearest neighbors, multilayer
perceptron (MLP), linear discriminant analysis. To classification of thyroid disease.

1. INTRODUCTION

Thyroid disease is a subset of endocrinology which is one of the most misunderstood and undiagnosed diseases [1] [2]. Thyroid gland diseases are among the most prevalent endocrine disorders in the world, second only to diabetes, according to the World Health Organization. Hyper function hyperthyroidism and hypothyroidism affect about 2% and 1% of individuals, respectively. Men have about a tenth of the prevalence of women. Hyper-and hypothyroidism may be caused by thyroid gland dysfunction, secondary to pituitary gland failure, or tertiary to hypothalamic malfunction. Due to dietary iodine deficiency, goiter or active thyroid nodules may become prevalent in some regions, with a prevalence of up to 15%. The thyroid gland can also be the location of different kinds of tumors and can be a dangerous place where endogenous antibodies wreak havoc (autoantibodies) [3]. Early disease detection, diagnosis, and care, according to doctors, are vital in preventing disease progression and even death. For several different forms of anomalies, early identification and differential diagnosis raises the odds of good treatment. Despite multiple trials, clinical diagnosis is often thought to be a difficult task [4]. The thyroid gland is a butterfly-shaped gland situated at the base of the throat. It comprises two active thyroid hormones, levothyroxine (T4) and triiodothyronine (T3), which are involved in brain functions such as body temperature control, blood pressure management, and heart rate regulation

Define Problem / Problem Understanding

In this milestone, we will go through the problem understanding.

Specify The Business Problem

The Thyroid gland is a vascular gland and one of the most important organs of the human body. This gland secretes two hormones which help in controlling the metabolism of the body.
The two types of Thyroid disorders are Hyperthyroidism and Hypothyroidism. When this disorder occurs in the body, they release certain types of hormones into the body which imbalances the body's metabolism. A thyroid-related Blood test is used to detect this disease but it is often blurred and noise will be present. Data cleansing methods were used to make the data primitive enough for the analytics to show the risk of patients getting this disease. Machine Learning plays a very deciding role in disease prediction. Machine Learning algorithms, SVM - support vector machine, Random Forest Classifier, XGB Classifier and ANN - Artificial Neural Networks are used to predict the patient's risk of getting thyroid disease. The web app is created to get data from users to predict the type of disease

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.

Literature Survey

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.
.

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.

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.

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.

Exploratory Data Analysis

In this milestone, we will see the exploratory data analysis.

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                  2140 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
```

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.

Checking Correlation

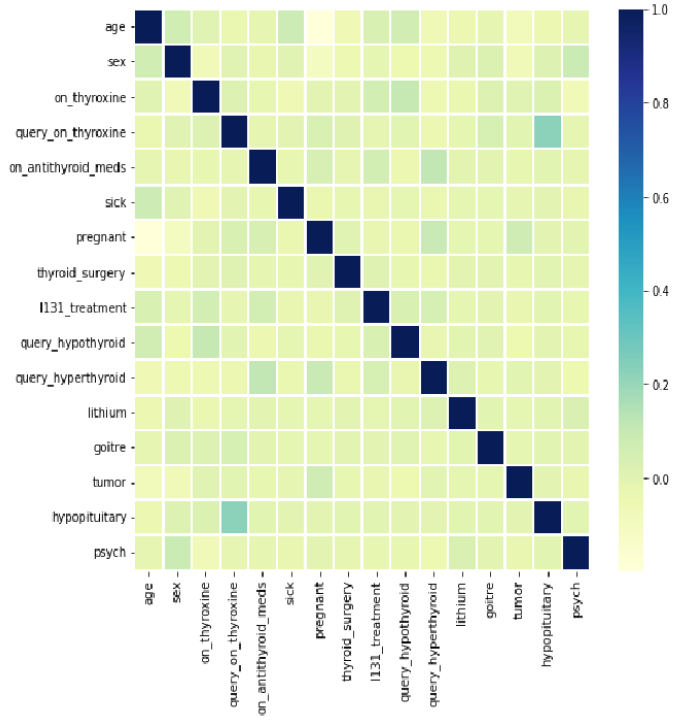
Here, I'm finding the correlation using HeatMap. It visualizes the data in 2-D coloured maps making use of colour variations. It describes the related variables in the form of colours instead of numbers; it will be plotted on both axes.

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:>
```



Model Building

In this milestone, we will see the concepts of model building.

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.

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()
```

```
rfr1.fit(x_os, y_os.values.ravel())
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

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

```
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

    accuracy                           0.69       854
   macro avg       0.61      0.69      0.64       854
weighted avg       0.61      0.69      0.64       854
```

```
train_score = accuracy_score(y_os, rfr1.predict(x_os))
train_score
```

```
1.0
```

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
```

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

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
```

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: -1989677696.00
00 - val_accuracy: 0.1429
Epoch 6/15
104/104 [==============================] - 1s 8ms/step - loss: -3208519680.0000 - accuracy: 0.1429 - val_loss: -5285069824.00
00 - val_accuracy: 0.1429
Epoch 7/15
```

Testing The Model

testing the models

```
In [115]: rfr1.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

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([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

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 = [[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]]
da1 = pd.DataFrame(data = da, columns=col)
xgb1.predict(da1)

Out[143]: array([4], dtype=int64)

In [140]: model.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

1/1 [==============================] - 0s 238ms/step

Out[140]: array([[1.]], dtype=float32)
```

Performance Testing & Hyperparameter Tuning

In this milestone, we will go through performance testing and hyperparameter tuning.

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.

Compare The Model

For comparing the above four models, the compare Model 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      0.72      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
```

```python
y_pred = sv.predict(x_test_bal)
```

```python
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
```

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

```
0.7154989384288747
```

```python
y_pred = model.predict(x_test_bal)
```

```
27/27 [==============================] - 0s 3ms/step
```

```python
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))
```

```python
accuracy_score(y_test_bal,y_pred)
```

```
0.14285714285714285
```

Comparing Model Accuracy Before & After Applying Hyperparameter Tuning

From sk learn, 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-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)
```

```
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([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])
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])
```

Model Deployment

In this milestone, we will see the model deployement.

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'))
```

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 uses where he has to enter the values for predictions. The enter 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

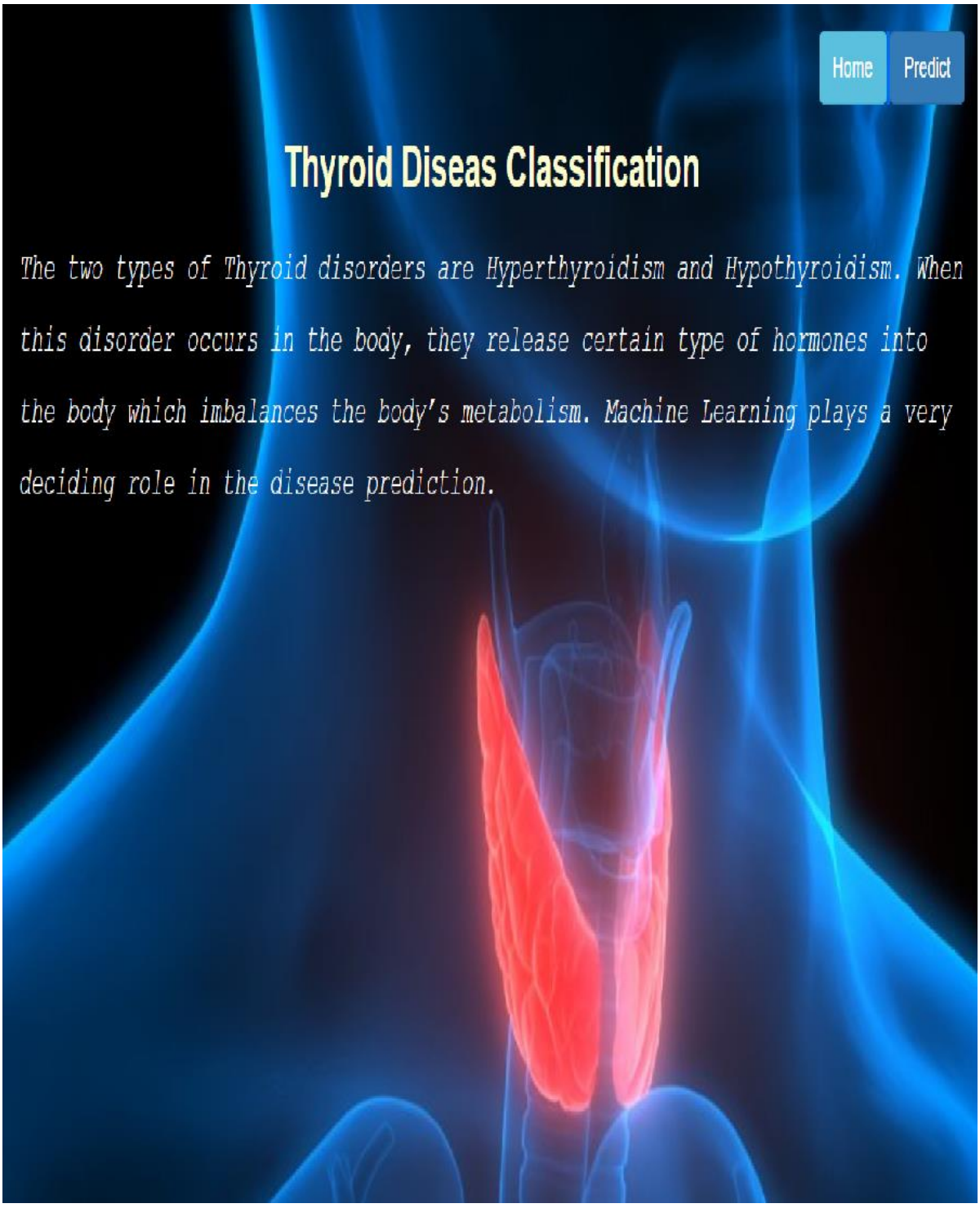
Building Html Pages

For this project create three HTML files namely

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

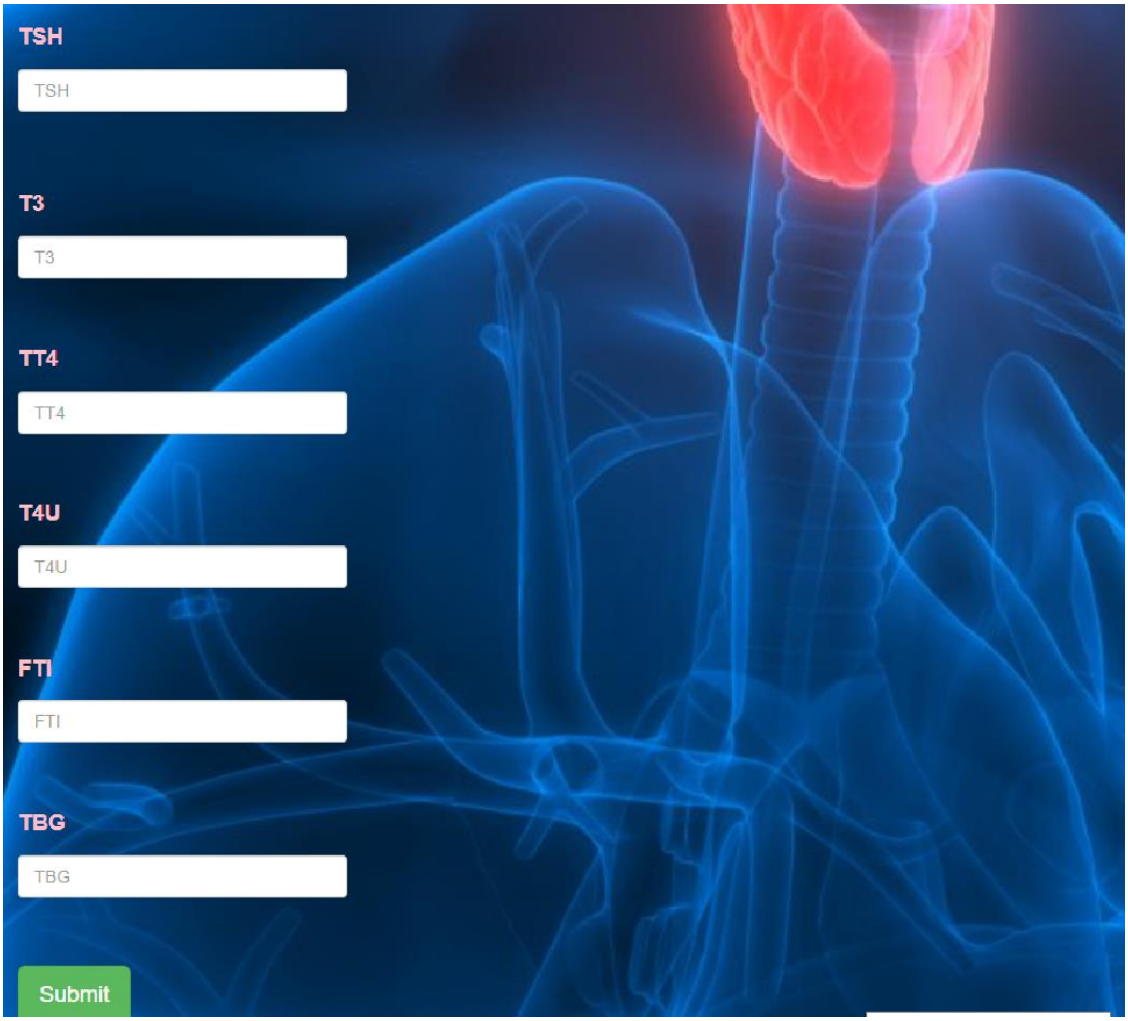
and save them in the templates folder.

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

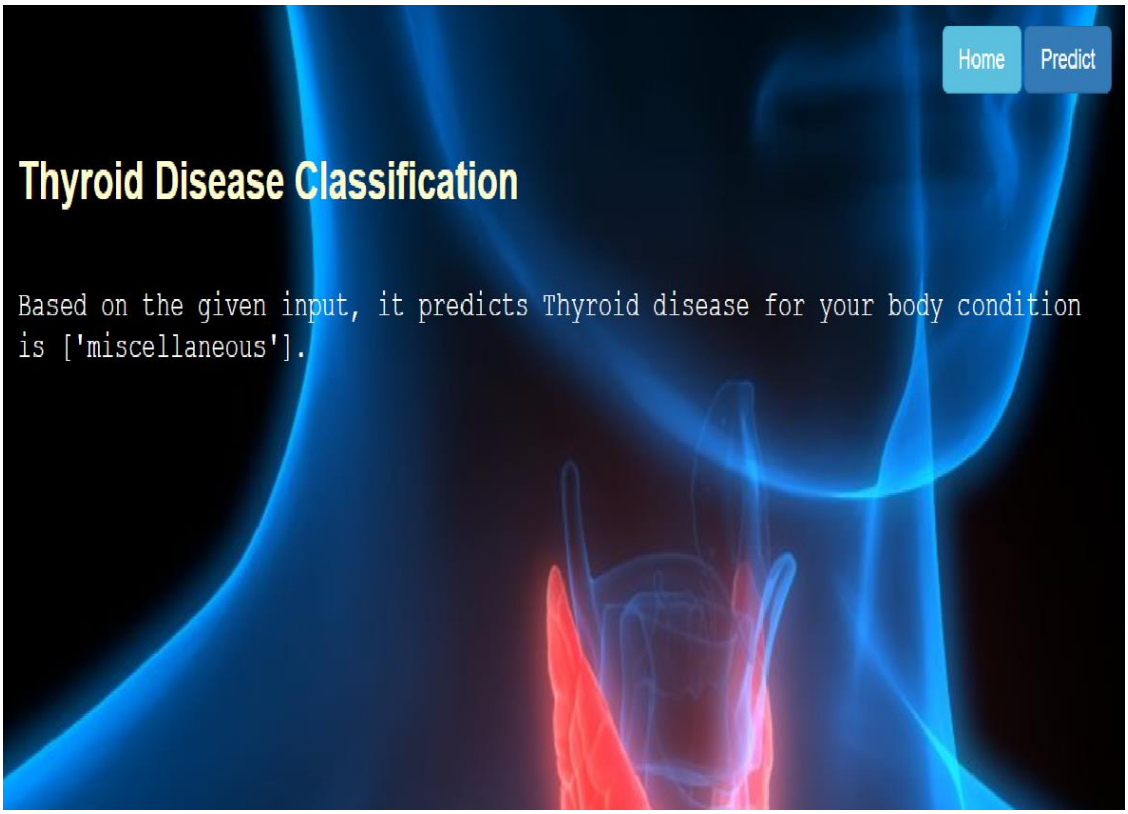


Now when you click on predict button from top right corner you will get redirected to predict.html
Let's look how our predict.html file looks like:


Home
Predict
Thyroid Disease Classification
goitre
Male
tumor
Male
hypopituitary
Male
psych
Male
TSH
TSH


TSH
TSH
T3
T3
TT4
TT4
T4U
T4U
FTI
FTI
TBG
TBG
Submit

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'].



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))
```

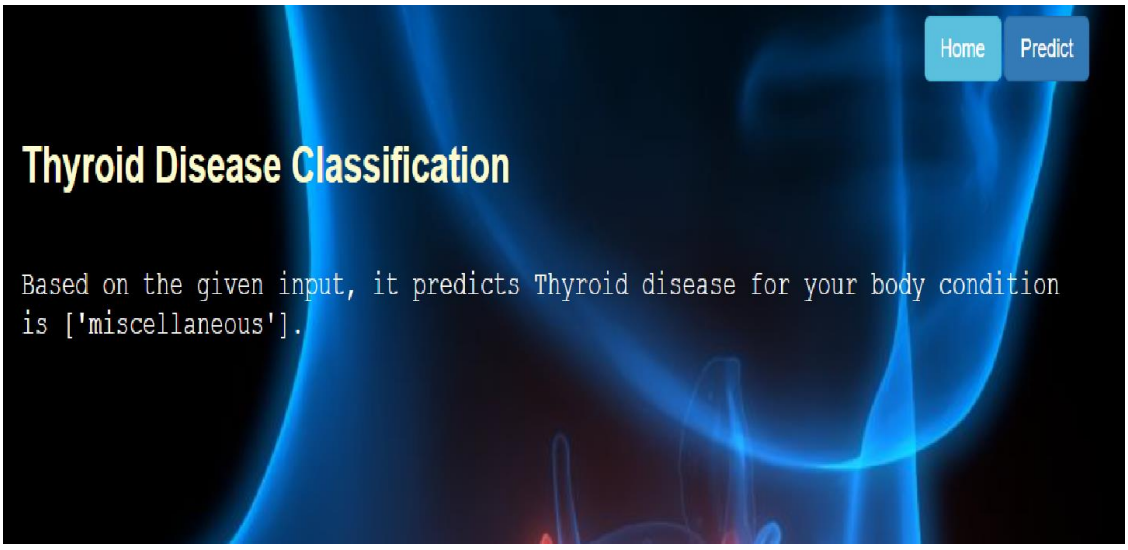
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

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

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

```
In [32]: runfile('C:/Users/SmartBridge-PC/
Downloads/Thyroid/app.py', wdir='C:/Users/
SmartBridge-PC/Downloads/Thyroid')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do
not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press
CTRL+C to quit)
```



Project Demonstration & Documentation

Project deliverables to be submitted along with other deliverables

Record Explanation Video For The Project End To End Solution

Record explanation Video for the project end to end solution

Project Documentation-Step By Step Project Development Procedure

Create document as per the template provided

Conclusion

Thyroid disease is one of the diseases that afflict the world's population, and the number of cases of this disease is increasing. Because of medical reports that show serious imbalances in thyroid diseases, our study deals with the classification of thyroid disease between hyperthyroidism and hypothyroidism. This disease was classified using algorithms. Machine learning showed us good results using several algorithms and was built in the form of two models. In the first model, all the characteristics consisting of 16 inputs and one output were taken, and the result of the accuracy of the random forest algorithm was 98.93, which is the highest accuracy among the other algorithms. In the second embodiment, the following characteristics were omitted based on a previous study. The removed attributes were 1-query_thyroxine 2-query_hypothyorid 3-query_hyperthyroid. Here we have included the increased accuracy of some algorithms, as well as the retention of the accuracy of others. It was observed that the accuracy of Naïve Bayes algorithm increased the accuracy by 90.67. The highest precision of the MLP algorithm was 96.4
accuracy.