THYROID DISEASE CLASSIFICATION USING

MACHINE LEARNING

INTRODUCTION

1. OVERVIEW
2. PURPOSE

* PROJECT FLOW
* PROJECT STRUCTURE

1. PROBLEM DEFINITION & DESIGN THINKING

* EMPATHY MAP
* BRAINSTORMING MAP

4. RESULT

5. ADVANTAGES

6. DISADVANTAGES

7. APPLICATIONS

8. CONCLUSION

9. FUTURE SCOPE

## OVERVIEW

* 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.

## PURPOSE

The Thyroid is defined as the “power house” of our body: if something in this gland fails, the whole body suffers as well. Therefore, the early diagnosis of a possible malfunction plays a fundamental role, as well as the prediction of the course of treatment of a patient with Hypothyroidism.

Which can be of great help for doctors who have patients under treatment. In this study we proposed an approach to predict the thyroid disease treatment .

This approach is intended to be a machine learning –based decision support system for endocrinologists treating patients with thyroid disease. The proposed feature model is evaluated using 10 different machine learning classifiers, belonging to the class of enhancement algorithms.

## PROJECT FLOW

* The user interacts with the UI to enter the input.
* Entered input is analyses by the model which is integrated.
* Once the model analyses the input the prediction is showcased on the UI.

To accomplish this, we have to complete all the activities listed below,

* Define problem/problem understanding
* Specify the business problem
* Business requirements
* Literature Survey
* social or Business Impact
* Data Collection & Preparation
* Collect the dataset
* Data Preparation
* Exploratory Data Analysis
* Descriptive statistical
* Visual Analysis
* Model Building
* Training the model in multiple algorithms
* Testing the model
* Performance Testing & Hyper parameter Tuning
* Testing model with multiple evaluation metrics
* Comparing model accuracy before & after applying hyperparameter tuning
* Model Deployment
* Save the best model
* Integrate with web Framework
* Project Demonstration & Documentation
* Record explanation Video for project end to end solution
* Project Documentation step by step project development procedure

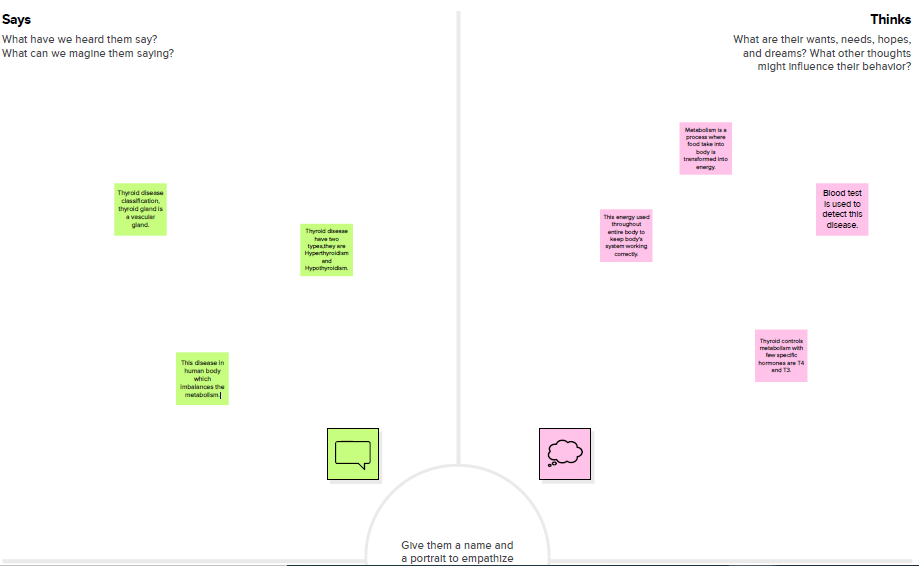
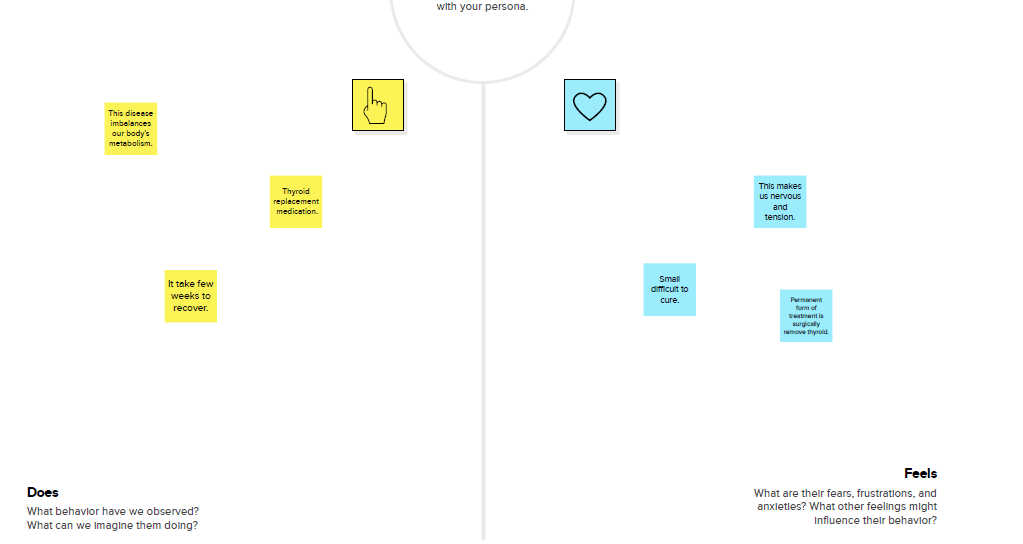
## PROJECT STRUCTURE

Create the project folder which contains files as shown below

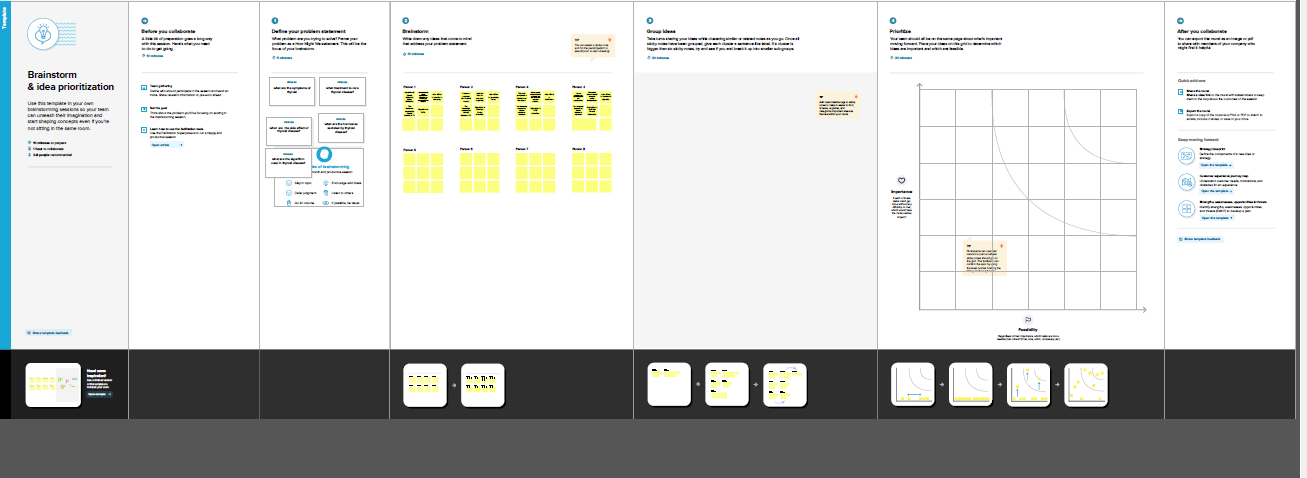
* We are building a flask application which needs HTML pages stored in templates folder and a python script app.py for scripting.
* thyroid\_1\_model.pk1 is our saved model. Further we will use this model for flask integration.
* Training folder contains model training files and the training\_ibm folder contains IBM deployment files

## PROBLEM DEFINITION & DESIGN THINKING

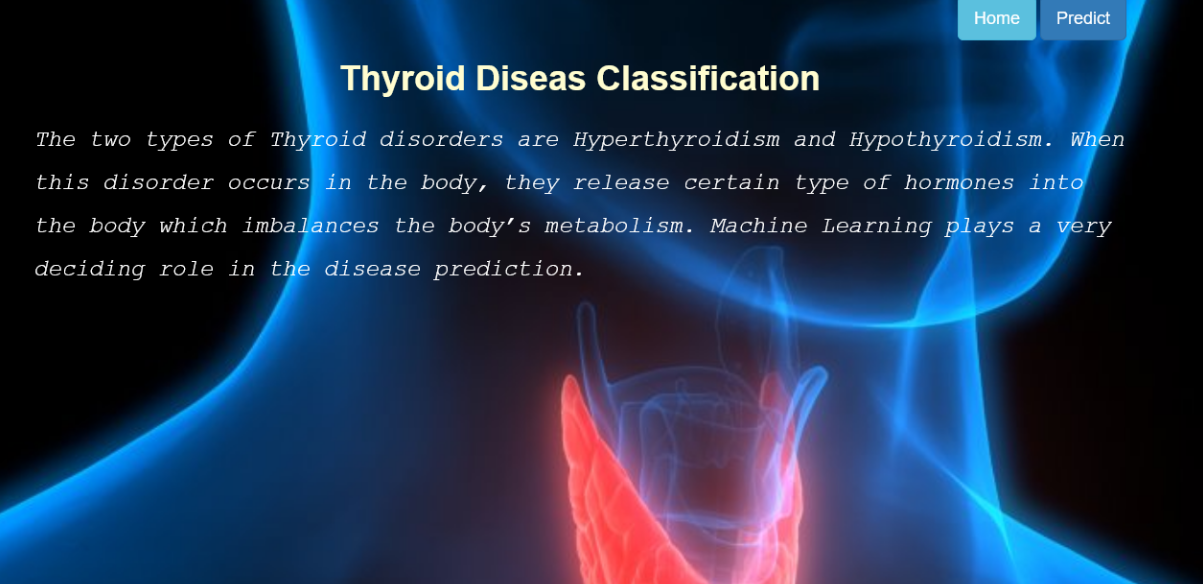
* EMPATHY MAP

* BRAINSTORMING MAP



## RESULT



## 

## ADVANTAGES

1. Machine learning can be very effective in predicting and classifying the thyroid diseases in earlier as well later stages.
2. Machine learning methodology to classify thyroid disease is so simple but very clear to mark the differences in ranges.
3. Thyroid disease classification is done with the machine learning methodology in short span of time as well as less cost of spending.
4. Thyroid disease classification can be done easily with the help of machine learning in all age groups.
5. By using Machine learning accurate classification of thyroid disease can be using emerging and innovative methods

# **DISADVANTAGES**

1. In thyroid disease classification machine learning can be used. but it is a newly growing methodology.
2. There is no empirical evidences in classifying thyroid disease in accurate levels.
3. Error levels cannot be predicted in this machine learning methods.
4. Machine learning is newly developing concept. It is effectiveness will only come through after various research methods.
5. Emerging machine learning concepts needs various skilled and qualified human resourses in thyroid disease classification.

## APPLICATION

* Machine Learning has different algorithm that applies to classify

the thyroid disease for the effective treatment

* Machine Learning application of decision tree method helps to

classify the thyroid disease in wast variety of datas like tree to indentify

and apply treatment acconding to the classification

* Machine Learning Random forest algorithm places data sampling

and

## CONCLUSION

Thyroid disease is one of the disease that afflict the world’s Population, and the number of cases of this disease is increasing .Because of medical reports that show series imbalance in thyroid disease ,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-querry\_thyroxin 2-querry\_hypothyroid 3-querry\_hyperthyroid.Here we have included the increased accuracy of some algorithms , as well as the Retension of the accuracy of others .It was observed that the accuracy of Naïve Bayes algorithms increased the accuracy by 90.67.The highest precision of the MLP algorithm was 96.4 accuracy.

## FUTURE SCOPE

There are many thyroid disease affecting people all over the world. Many disease affect the thyroid gland, like hyperthyroidism, hypothyroidism, and thyroid cancer

.

Thyroid disease classification in the future, to better generalize our finding it is necessary to further expand the set of the data and attributes considered. With more data the training process is likely to produce more effective classifiers also allowing a more reliable estimate of the exhibited performance.

Finally, another aspect that could be investigated concerns the presence of any secondary thyroid disease linked to the patient, to understand if there is a particular additional thyroid disease that can affect hypothyroidism. In fact, it often happens that patients are suffering from more than one thyroid disease at the same time

## APPENDEX

Thyroid disease.ipynb

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from google.colab import drive

drive.mount("/content/drive")

data = pd.read\_csv("/content/thyroidDF.csv")

data.head()

data['target'].unique()

data.shape

data.info()

data.isnull().sum()

data.drop(['TSH\_measured','T3\_measured','TT4\_measured','T4U','FTI\_measured','TBG\_measured','referral\_source','patient\_id','T4U\_measured',],axis=1,inplace=True)

data.head()

data['target']

diagnoses ={'A': 'hyperthyroid conditions',

'B': 'hyperthyroid conditions',

'C': 'hyperthyroid conditions',

'D': 'hyperthyroid conditions',

'E': 'hyperthyroid conditions',

'F': 'hyperthyroid conditions',

'G': 'hyperthyroid conditions',

'H': 'hyperthyroid 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)

data

data.isnull().sum()

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

data['target'].value\_counts()

data['target'].isnull().sum()

data.head()

data.describe()

data[data.age>100]

data['age']=np.where((data.age>100), np.nan, data.age)

data

#splitting the data values as x and y

x=data.iloc[:,0:-1]

y=data.iloc[:,-1]

data.isnull().sum()

x['sex'].unique()

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

x['sex'].value\_counts()

x.isnull().sum()

data.info()

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['FTI']=x['FTI'].astype('float')

x['TBG']=x['TBG'].astype('float')

# converting categorical to numerical values

from sklearn.preprocessing import OrdinalEncoder, LabelEncoder

ordinal\_encoder = OrdinalEncoder(dtype = 'int64')

x.iloc[:, 1:16] = ordinal\_encoder.fit\_transform(x.iloc[:, 1:16])

x.head()

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

x.head()

label\_encoder = LabelEncoder()

y\_dt= label\_encoder.fit\_transform(y)

y=pd.DataFrame(y\_dt, columns=['targat'])

Y

y.value\_counts(normalize=True)

# Checking the Correlation

import seaborn as sns

corrmat = x.corr()

f, ax = plt.subplots(figsize =(9, 8))

sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)

# splitting the train and test split

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)

y\_train.value\_counts()

from imblearn.over\_sampling import SMOTE

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)

x\_bal

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_bal=sc.fit\_transform(x\_bal)

x\_test\_bal= sc.fit\_transform(x\_test\_bal)

x\_bal

x\_test\_bal

y\_bal.value\_counts()

columns=['age','sex','on\_thyroxine','query\_on\_thyroxine','on\_antithyroid\_meds','sick','pregnant','thyroid\_surgery','I131\_treatment','query\_hypothyroid','query\_hyperthyroid','lithium','goitre','tumor','hypopituitary','psych','TSH','T3','TT4','FTI','TBG']

x\_test\_bal= pd.DataFrame(x\_test\_bal,columns=columns)

x\_bal= pd.DataFrame(x\_bal,columns=columns)

x\_bal

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

rfr = RandomForestClassifier().fit(x\_bal,y\_bal)

y\_pred = rfr.predict(x\_test\_bal)

accuracy\_score(y\_test\_bal,y\_pred)

x\_bal.shape,y\_bal.shape,x\_test\_bal.shape,y\_test\_bal.shape

test\_score=accuracy\_score(y\_test\_bal,y\_pred)

test\_score

train\_score = accuracy\_score(y\_bal,rfr.predict(x\_bal))

train\_score

# performing features impotance

#perform feature importance

from sklearn.inspection import permutation\_importance

results = permutation\_importance(rfr,x\_bal,y\_bal, scoring='accuracy')

feature\_importance=['age','sex','on\_thyroxine','query\_on\_thyroxine','on\_antithyroid\_meds','sick','pregnant','throid\_surgery','I131\_treatment','query\_hypothyroid','qurey\_hyperthyroid','lithium','goitre','tumor','hypopituitary','psych','TSH','T3','TT4','FTI','TBG']

importance = results.importances\_mean

importance = np.sort(importance)

for i,v in enumerate(importance):

i=feature\_importance[i]

print('feature: {:<20} score: {}'. format(i,v))

plt.figure(figsize=(10,10))

plt.bar(x=feature\_importance, height= importance)

plt.xticks(rotation=30, ha='right')

plt.show()

x\_bal.drop(['age','sex','on\_thyroxine','query\_on\_thyroxine','on\_antithyroid\_meds','sick','pregnant','thyroid\_surgery','I131\_treatment','query\_hypothyroid','lithium','goitre','tumor','hypopituitary','psych','TSH','T3','TT4','FTI','TBG'],axis=1 , inplace=True)

x\_test\_bal.drop(['age','sex','on\_thyroxine','query\_on\_thyroxine','on\_antithyroid\_meds','sick','pregnant','thyroid\_surgery','I131\_treatment','query\_hypothyroid','query\_hypothyroid','lithium','goitre','tumor','hypopituitary','psych','TSH','T3','TT4','FTI','TBG'],axis=1 , inplace=True)

x\_bal.head()

x\_test\_bal.head()

rfr1 =RandomForestClassifier()

rfr1.fit(x\_bal,y\_bal)

y\_pred=rfr1.predict(x\_test\_bal)

print(classification\_report(y\_test\_bal,y\_pred))

train\_score = accuracy\_score(y\_bal,rfr1.predict(x\_bal))

train\_score

# XGBClassifier Model-2

from xgboost import XGBClassifier

xgb = XGBClassifier()

xgb.fit(x\_bal,y\_bal)

y\_pred=xgb.predict(x\_test\_bal)

print(classification\_report(y\_test\_bal,y\_pred))

train\_score = accuracy\_score(y\_bal, xgb.predict(x\_bal))

train\_score

# SVC Model-3

params = {

'C' : [0.01, 1, 10,100, 1000],

'kernel': ['rbf', 'linear'],

'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

}

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report

sv= SVC()

from sklearn.model\_selection import GridSearchCV

grid\_svc = GridSearchCV(sv,params,scoring='accuracy',cv=5,n\_jobs=-1)

x\_bal

sv.fit(x\_bal,y\_bal)

y\_pred = sv.predict(x\_test\_bal)

print(classification\_report(y\_test\_bal,y\_pred))

train\_score=accuracy\_score(y\_bal,sv.predict(x\_bal))

train\_score

# Grid\_Search for RandomForest

from sklearn.metrics.cluster import entropy

params={

'n\_estimators' :[100, 200, 500],

'criterion': ['gini','entropy'],

'max\_depth': [x for x in range(1,20)]

}

from sklearn.model\_selection import GridSearchCV

grid\_search =GridSearchCV(rfr1, params, scoring='accuracy',cv=5,n\_jobs=-1)

grid\_search.fit(x\_bal,y\_bal)

grid\_search.best\_params\_

xgb1=XGBClassifier(booster='gbtree', gamna= 0, learning\_rate= 0.1, n\_estimators= 500)

xgb1.fit(x\_bal,y\_bal)

y\_pred= xgb1.predict(x\_test\_bal)

print(classification\_report(y\_test\_bal,y\_pred))

train\_score= accuracy\_score(y\_bal,xgb1.predict(x\_bal))

train\_score

# Grid\_Search for SVC

params = {

'C' : [0.01, 1, 10,100, 1000],

'kernel': ['rbf', 'linear'],

'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

}

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report

sv= SVC()

from sklearn.model\_selection import GridSearchCV

grid\_svc = GridSearchCV(sv,params,scoring='accuracy',cv=5,n\_jobs=-1)

x\_bal

grid\_svc.fit(x\_bal,y\_bal)

grid\_svc.best\_params\_

sv1=SVC(C =1000,gamma=1, kernel= 'rbf')

sv1.fit(x\_bal,y\_bal)

y\_pred= sv1.predict(x\_test\_bal)

print(classification\_report(y\_test\_bal,y\_pred))

train\_score= accuracy\_score(y\_bal,sv1.predict(x\_bal))

train\_score

# saving the model

import pickle

pickle.dump(xgb1,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(xgb.predict(features)))

type(features)

pickle.dump(label\_encoder,open('label\_encoder.pkl','wb'))

data['target'].unique()

## app.py

from flask import Flask, render\_template, request

import pickle

import numpy as np

import sklearn

app = Flask(\_\_name\_\_)

model = pickle.load(open('rfd.pkl', 'rb'))

@app.route('/')

def home():

return render\_template('thyroid.html')

@app.route('/getdata', methods=['POST'])

def pred():

age = int(request.form['Age'])

print(age)

sex = request.form['Sex']

print(sex)

thyroxine = request.form['thyroxine']

print(age)

query = request.form['query']

print(age)

antithyro = request.form['antithyroid']

print(age)

sick = request.form['Sick']

print(age)

preg = request.form['Pregnant']

print(age)

thyros = request.form['ThyroSurgery']

print(age)

treat = request.form['I131treat']

print(age)

query1 = request.form['qweryhypo']

print(age)

query2 = request.form['qweryhyper']

print(age)

Li = request.form['Lithium']

print(age)

Go = request.form['Goitre']

print(age)

Tu = request.form['Tumour']

print(age)

hypo = request.form['Hypo']

print(age)

Psycho = request.form['Psycho']

print(age)

tsh = request.form['TSH']

print(age)

t3 = request.form['T3']

print(age)

t4u = request.form['T4U']

print(age)

fti = request.form['FTI']

print(age)

rs = request.form['RS']

print(age)

pid = request.form['PI']

inp\_features = [[age, np.log(float(sex)), int(thyroxine), int(query), int(antithyro), int(sick),

int(preg), int(thyros), int(treat), int(query1), int(query2), int(Li),

int(Go), int(Tu), int(hypo), int(Psycho), np.log(float(tsh)), np.log(float(t3)),

np.log(float(t4u)), np.log(float(fti)), int(rs), int(pid)]]

print(inp\_features)

prediction = model.predict(inp\_features)

print(type(prediction))

t = prediction[0]

print(t)

if t >0.5:

prediction\_text = 'You have thyroid, Thyroid will be predicted'

else:

prediction\_text = 'Thyroid not predicted'

print(prediction\_text)

return render\_template('prediction.html', prediction\_results=prediction\_text)

if \_\_name\_\_ == "\_\_main\_\_":

app.run()

Predict.html:

<html>

<head>

<title>Thyroid Disease Prediction </title>

</head>

<body background="predict.jpg"style="background-repeat:no-repeat; background-size:100% 100%"text='black'>

<h1>

<b>

<i>

Thyroid Disease Prediction

</i>

</b>

</h1>

<h2>Prediction Results!!</h2>

<h2>{{prediction\_results}}</h2>

</body>

</html>

Thyroid.html:

<html>

<head>

<center>

<font size="10">

<h1>

<u>Thyroid Disease Classification</u></h1>

</font>

</center>

</head>

<form action="/getdata". method="post">

Age&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="Age"><br><br>

Sex&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="Sex"><br><br>

On\_thyroxine&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="thyroxine"><br><br>

query\_on\_thyroxine&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="query"><br><br>

On\_antithyroid\_meds&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="antithyroid"><br><br>

Sick&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="Sick"><br><br>

Pregnant&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="Pregnant"><br><br>

Thyroid\_surgery&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="ThyroSurgery"><br><br>

I131\_treatment&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="I131treat"><br><br>

Query\_hypothyroid&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="qweryhypo"><br><br>

Query\_hyperthyroid&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="qweryhyper"><br><br>

Lithium&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type=text name="Lithium"><br><br>

Goitre&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="text"name="Goitre"><br><br>

Tumor&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="Tumour"><br><br>

Hypoituitary&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="Hypo"><br><br>

Psych&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="Psycho"><br><br>

TSH&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="TSH"><br><br>

T3&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="T3"><br><br>

T4U&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="T4U"><br><br>

FTI&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="FTI"><br><br>

Referral source&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="RS"><br><br>

Patient ID&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;:<input type="text"name="PI"><br><br>

<button type="submit"class="btn btn-primary btn-block btn-large"name="btn1">Predict</button>

</font>

</form>

</html>

Submit.html:

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Predict</title>

<link rel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<style>

body

{

background-image: url("https://i.pinimg.com/564x/18/d8/da/18d8da592a999a56398d34c38a1125c3.jpg");

background-size: cover;

}

h3.big

{

line-height: 1.8;

}

</style>

</head>

<body>

<br>

<div class="container">

<div class="row">

<div class="col-md-12 bg-light text-right">

<a href="/home"class="btn btn-info btn-lg">Home</a>

<a href="/predict"class="btn btn-primary disabled btn-lg">Predict</a>

</div>

</div>

<br>

<h1><strong>Drug Classification</strong></h1><br>

<h4>

<form action="/pred", method="post">

<div class="form-group row">

<div class="col-md-3">

<label for="Age">Age</label>

<input type="text"class="form-control"name="Age"id="Age"value="Age"placeholder="Age"required="required"/>

</div>

</div>

<div class="form-group mb-3">

<div class="input-group-prepend">

<label class="input-group-text"for="Sex">Sex</label>

</div>

<select class="custom-select"id="Sex"name="Sex">

<option value="1">Male</option>

<option value="0">Female</option>

</select>

</div><br>

<div class="form-group mb-3">

<div class="input-group-prepend">

<label class="input-group-text"for="BP">BP</label>

</div>

<select class="custom-select"name="BP"id="BP">

<option value="0">Low</option>

<option value="1">Normal</option>

<option value="2">High</option>

</select>

</div><br>

<div class="form-group mb-3">

<div class="input-group-prepend">

<label class="input-group-text"for="Cholesterol">Cholesterol</label>

</div>

<select class="custom-select"name="Cholesterol"id="Cholesterol">

<option value="0">Normal</option>

<option value="1">High</option>

</select>

</div><br>

<div class="form-group row">

<div class="col-md-3">

<label for="Na\_to\_K">Na\_to\_K</label>

<input type="text"name="Na\_to\_K"id="Na\_to\_K"class="form-control"placeholder="Na\_to\_K"required="required"/><br><br>

</div>

</div>

<button type="submit"class="btn btn-success btn-lg">Submit</button>

</form>

<br>

</h4>

</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

</body>

</html>