# THYROID DISEASE CLASSIFICATION USING MACHINE LEARNING

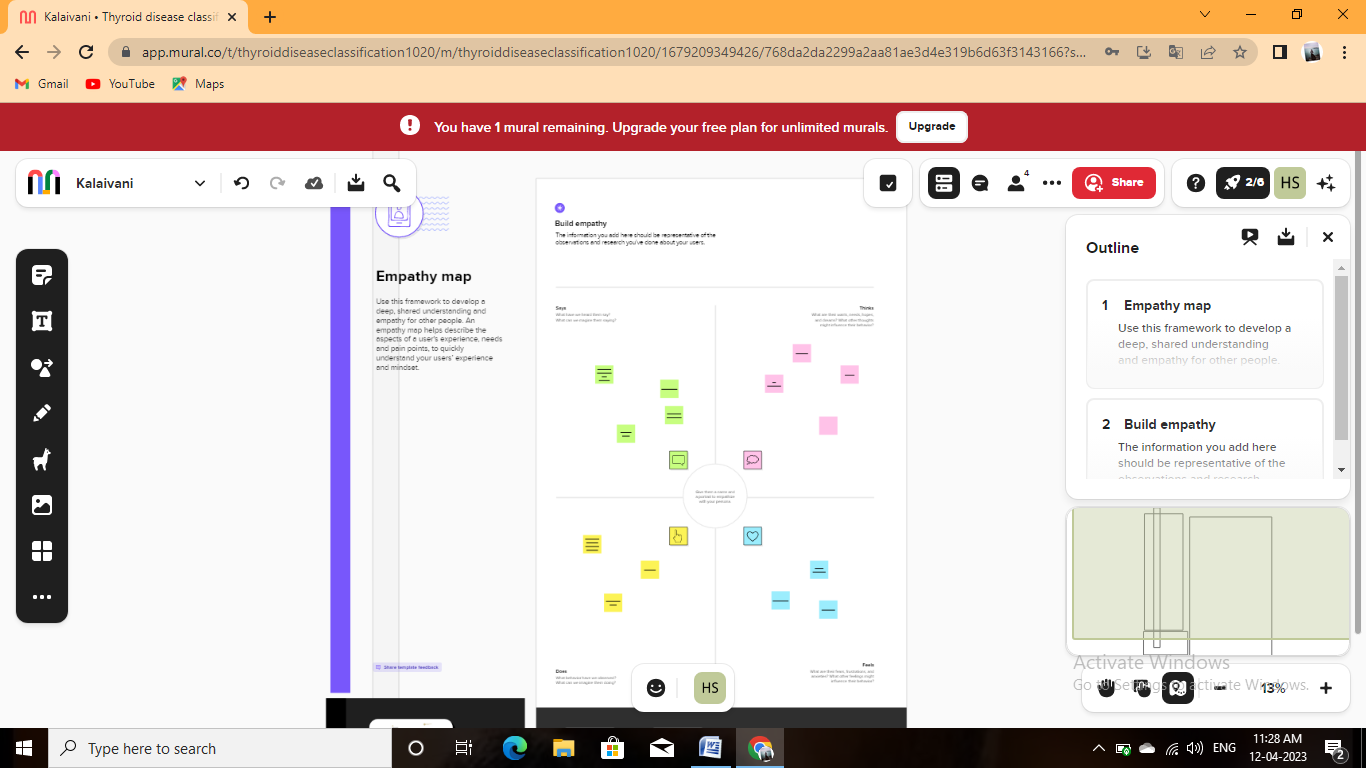
* Introduction
* Problem definition and design thinking
* Result
* Advantages and Disadvantages
* Applications
* Conclusion
* Future scope

**INTRODUCTION:**

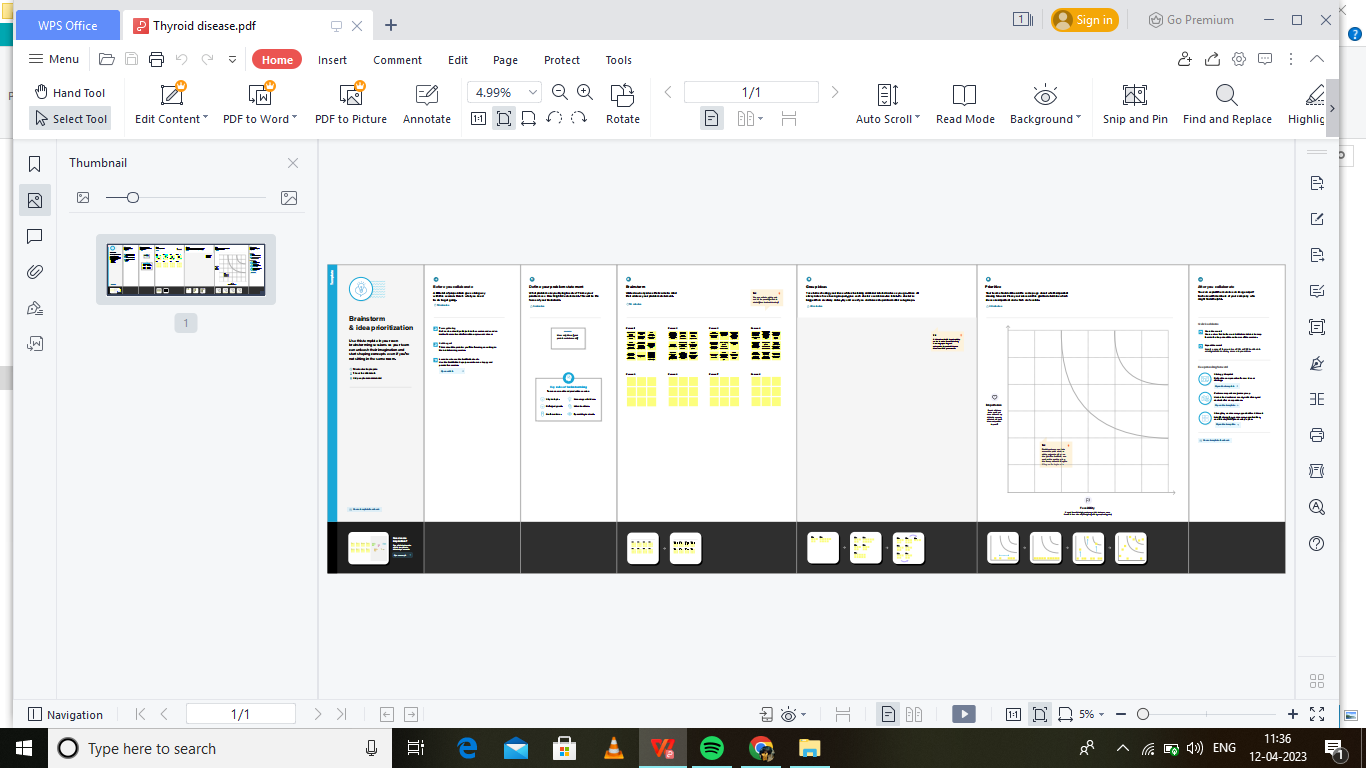
* The Thyroid gland is a vascular gland and one of the most important organs of the human body.
* This gland secretes two hormones which help in controlling the metabolism of the body.
* The two types of thyroid disorders are hyperthyroidism and hypothyroidism.
* When this disorder occurs in the body, they release certain types of hormones into the body which imbalances body’s metabolism.
* A thyroid-related blood test is used to detect this disease but it is often blurred and the analytics to show the risk of patients getting this disease.
* Machine learning plays a very deciding role in disease prediction.

**PROBLEM DEFINITION AND DESIGN THINKING:**

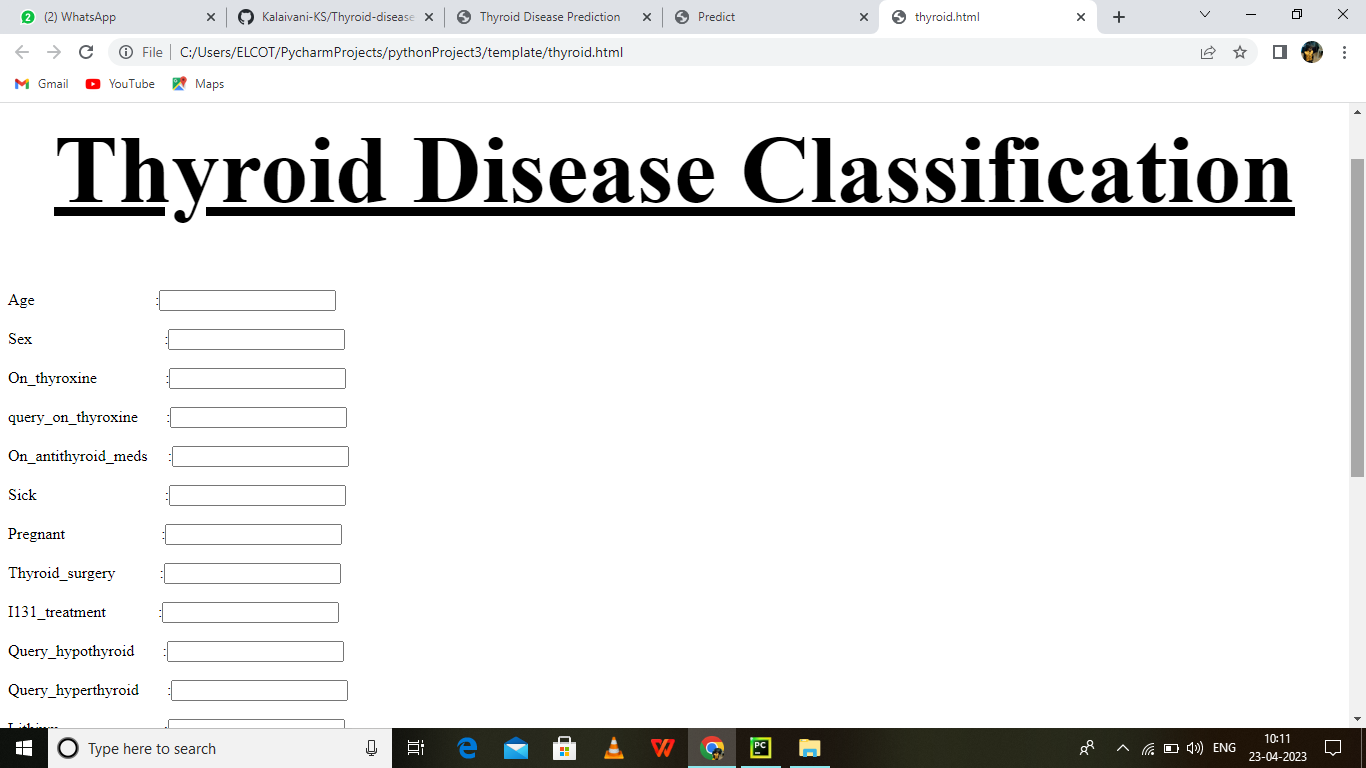
**Empathy map:**

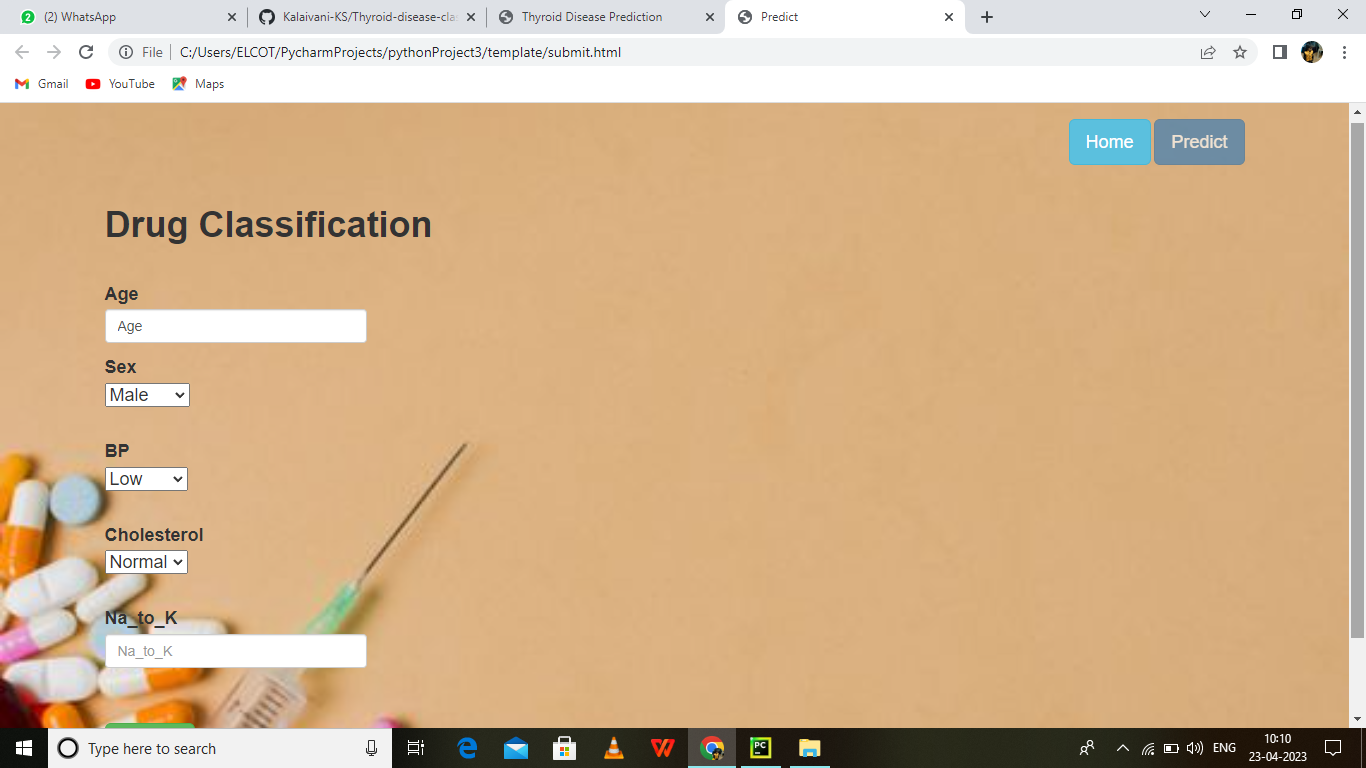
******

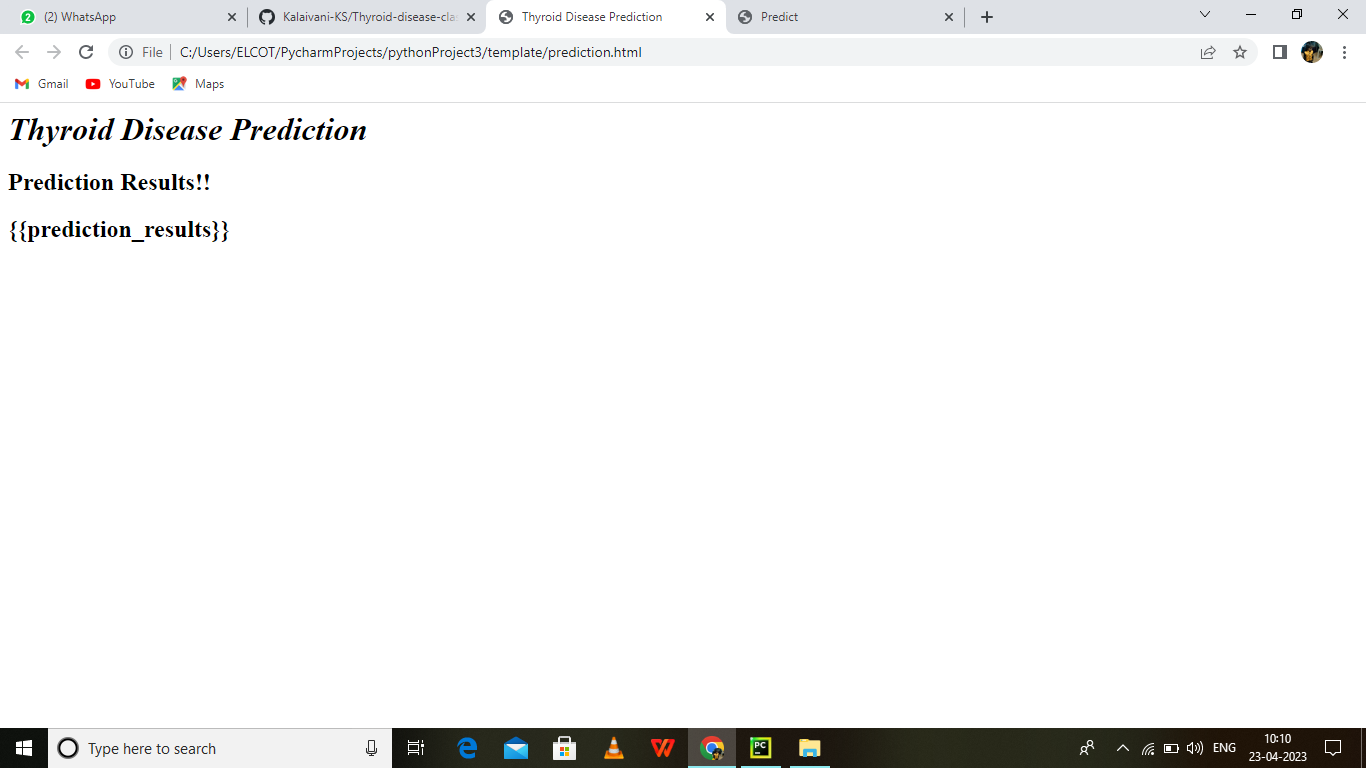
**Brainstorming map:**

****

**RESULT:**

****





ADVANTAGES:

* In this prediction process, machine learning plays a key role, and paper research and the classification of models used in thyroid disease are based on information from UCI machine learning repositories.
* A decent knowledge base that can be centered and used as a hybrid paradigm must be preserved in order to address complex learning issues, such as medical diagnostics and statistical tasks.
* We also proposed different approaches for machine learning and thyroid diagnosis.
* Machine learning algorithms, vector support machine, decision trees, K-NN where used to calculate an estimated probability of a patient having thyroid disease.
* Avoids long-term risks of anti-thyroid and radioactive iodine.
* Provides histology tissue, for childbearing instantly.

DISADVANTAGES:

* Thyroid disorder significant cause of formation medical diagnosis and estimation, which is a challenging axiom of medical science.
* The secretions of thyroid hormones are guilty of metabolism regulation.
* Hyperthyroidism and hyperthyroidism are one of the two prevalent thyroid disorders that release thyroid hormones to control body metabolism.

Energy level, weakness, breathing.

**APPLICATION:**

* It’s a part of your endocrine system and controls many of your body’s important functions by producing and releasing certain hormones.
* Your thyroid’s main job is to control the speed of your metabolism, which is the process of how your body transforms the food you consume into energy.

**CONCLUSION:**

Thyroid Detection using Machine Learning is a project idea that aims a smart and precise way to predict thyroid disease. We have made use of logistic regression algorithm to train our dataset and to predict thyroid disease with more accuracy. Here the machine is trained to detect whether the person normal, hyper-hypothyroidism based on the user’s input. So when user enters data in web app the data will processed in backend and the result will be displayed on the screen. Our objective was to give society an efficient and precise way of machine learning which can be used in application aiming to perform disease detection.Further development can be do by using image processing of ultrasonic scanning of thyroid images to predict thyroid nodules and cancer, which cannot be recognized in blood test report.By combining both the results, thyroid disease prediction can cover all thyroid related diseases.

**FUTURE SCOPE:**

Thyroid is a critical medical condition which can be caused either due to increased levels of TSH (Thyroid Stimulating Organ) or due to some infection in thyroid organs itself. The machine learning algorithms have been employed to model the prediction and diagnosis of thyroid patients. A variety of these algorithms including Decision trees, Random forest, Support vector machine, Artificial Neural Network and Logistic regression have been widely used in development of predictive models of thyroid disease. The paper presents a review of recent ML algorithms applied in the prediction and diagnosis of thyroid detection. The proposed system is used for thyroid disease prediction of patients, based on various symptoms and reports of thyroid. With comparative study, different ML techniques are used by the proposed system to achieve better accuracy in disease prediction. Among these, Decision tree algorithm is found to be better with the accuracy of 99.46%.

APPENDEX:

**Thyriod 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>