**Main.py**

import os

#-------------------------------------------------model\_code------------------------------------------------------------

import sqlite3

conn = sqlite3.connect('Fetal\_database')

cur = conn.cursor()

try:

cur.execute('''CREATE TABLE user (

name varchar(20) DEFAULT NULL,

email varchar(50) DEFAULT NULL,

password varchar(20) DEFAULT NULL,

gender varchar(10) DEFAULT NULL,

age int(11) DEFAULT NULL

)''')

except:

pass

#!/usr/bin/env python

# coding: utf-8

# importing libraries

import pandas as pd #Easily handles missing data,It provides an efficient way to slice the data,It includes a powerful time series tool to work

import numpy as np

#import matplotlib.pyplot as plt

#import seaborn as sns

#uploading dataset

data = pd.read\_csv('fetal\_health.csv',encoding= 'unicode\_escape')

#independent and dependent variable

X = data.iloc[:,:-7:-1]

y = data.iloc[:,-1]

# spliting up to testing and training dataset

from sklearn.model\_selection import train\_test\_split

X\_train , X\_test , y\_train , y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=100)

# navie bayes

import sklearn

from sklearn.naive\_bayes import BernoulliNB

from sklearn import metrics

from sklearn.metrics import accuracy\_score

BernNB = BernoulliNB(binarize=.1)

BernNB.fit(X\_train,y\_train)

print(BernNB)

y\_expect = y\_test

y\_pred = BernNB.predict(X\_test)

print(accuracy\_score(y\_expect,y\_pred)\*100)

score\_nb = accuracy\_score(y\_expect,y\_pred)\*100

# logistic regression

from sklearn.linear\_model import LogisticRegression

reg = LogisticRegression().fit(X\_train, y\_train)

score\_lr = reg.score(X\_test, y\_test)

score\_lr= score\_lr \* 100

score\_lr

# Random forest

from sklearn.ensemble import RandomForestClassifier

regr = RandomForestClassifier(max\_depth=3, random\_state=0)

regr.fit(X\_train, y\_train)

score\_rf = regr.score(X\_test,y\_test)

score\_rf =score\_rf\*100

score\_rf

# Decision tree

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max\_depth=7)

clf.fit(X\_train, y\_train)

score\_dt = clf.score(X\_test,y\_test)

score\_dt = score\_dt \* 100

score\_dt

scores = [score\_nb,score\_lr,score\_rf,score\_dt]

algorithems = ['navie bayes','logistic regressor','random forest','Decision Tree']

for i in range(len(algorithems)):

print('the accuracy score achieved using' +algorithems[i]+ 'is :'+str(scores[i])+" %")

from flask import Flask,render\_template, url\_for,request, flash, redirect, session

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

app.config['SECRET\_KEY'] = '881e69e15e7a528830975467b9d87a98'

#-------------------------------------home\_page-------------------------------------------------------------------------

@app.route('/')

@app.route('/home')

def home():

if not session.get('logged\_in'):

return render\_template('home.html')

else:

return redirect(url\_for('user\_account'))

#-------------------------------------about\_page-------------------------------------------------------------------------

@app.route("/about")

def about():

return render\_template('about.html')

#-------------------------------------about\_page-------------------------------------------------------------------------

#-------------------------------------user\_login\_page-------------------------------------------------------------------------

@app.route('/user\_login',methods = ['POST', 'GET'])

def user\_login():

conn = sqlite3.connect('Fetal\_database')

cur = conn.cursor()

if request.method == 'POST':

email = request.form['email']

password = request.form['psw']

print('asd')

count = cur.execute('SELECT \* FROM user WHERE email = "%s" AND password = "%s"' % (email, password))

print(count)

#conn.commit()

#cur.close()

l = len(cur.fetchall())

if l > 0:

flash( f'Successfully Logged in' )

return render\_template('user\_account.html')

else:

print('hello')

flash( f'Invalid Email and Password!' )

return render\_template('user\_login.html')

# -------------------------------------user\_login\_page-----------------------------------------------------------------

# -------------------------------------user\_register\_page-------------------------------------------------------------------------

@ app.route('/user\_register', methods=['POST', 'GET'])

def user\_register():

conn = sqlite3.connect('Fetal\_database')

cur = conn.cursor()

if request.method == 'POST':

name = request.form['uname']

email = request.form['email']

password = request.form['psw']

gender = request.form['gender']

age = request.form['age']

cur.execute("insert into user(name,email,password,gender,age) values ('%s','%s','%s','%s','%s')" % (name, email, password, gender, age))

conn.commit()

# cur.close()

print('data inserted')

return redirect(url\_for('user\_login'))

return render\_template('user\_register.html')

# -------------------------------------user\_register\_page-------------------------------------------------------------------------

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

def predict():

baseline\_value = request.form['baseline\_value']

accelerations = request.form['accelerations']

fetal\_movement = request.form['fetal\_movement']

uterine\_contractions = request.form['uterine\_contractions']

light\_decelerations = request.form['light\_decelerations']

severe\_decelerations = request.form['severe\_decelerations']

global clf

if request.method == 'POST':

out = clf.predict([[float(baseline\_value),

float(accelerations),

float(fetal\_movement),

float(uterine\_contractions),

float(light\_decelerations),

float(severe\_decelerations)]])

if out == 1:

print("Abnormal")

flash('Abnormal')

elif out == 2:

print("neutral")

flash('neutral')

else:

print("Normal")

flash('Normal')

return render\_template('user\_account.html')

# ------------------------------------predict\_page-----------------------------------------------------------------

# -------------------------------------user\_logout\_page-------------------------------------------------------------------------

@app.route("/logout")

def logout():

session['logged\_in'] = False

return home()

@app.route("/logoutd",methods = ['POST','GET'])

def logoutd():

return home()

# -------------------------------------user\_logout\_page-------------------------------------------------------------------------

if \_\_name\_\_ == '\_\_main\_\_':

app.secret\_key = os.urandom(12)

app.run(debug=True)