import re

import pandas as pd

from sklearn import preprocessing

import pyttsx3

from sklearn.tree import DecisionTreeClassifier,\_tree

import numpy as np

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

from sklearn.model\_selection import cross\_val\_score

from sklearn.svm import SVC

import csv

import warnings

warnings.filterwarnings("ignore", category=DeprecationWarning)

training = pd.read\_csv('Data/Training.csv')

testing= pd.read\_csv('Data/Testing.csv')

cols= training.columns

cols= cols[:-1]

x = training[cols]

y = training['prognosis']

y1= y

reduced\_data = training.groupby(training['prognosis']).max()

#mapping strings to numbers

le = preprocessing.LabelEncoder()

le.fit(y)

y = le.transform(y)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.33, random\_state=42)

testx = testing[cols]

testy = testing['prognosis']

testy = le.transform(testy)

clf1 = DecisionTreeClassifier()

clf = clf1.fit(x\_train,y\_train)

# print(clf.score(x\_train,y\_train))

# print ("cross result========")

scores = cross\_val\_score(clf, x\_test, y\_test, cv=3)

# print (scores)

print (scores.mean())

model=SVC()

model.fit(x\_train,y\_train)

print("for svm: ")

print(model.score(x\_test,y\_test))

importances = clf.feature\_importances\_

indices = np.argsort(importances)[::-1]

features = cols

def readn(nstr):

engine = pyttsx3.init()

engine.setProperty('voice', "english+f5")

engine.setProperty('rate', 130)

engine.say(nstr)

engine.runAndWait()

engine.stop()

severityDictionary=dict()

description\_list = dict()

precautionDictionary=dict()

symptoms\_dict = {}

for index, symptom in enumerate(x):

symptoms\_dict[symptom] = index

def calc\_condition(exp,days):

sum=0

for item in exp:

sum=sum+severityDictionary[item]

if((sum\*days)/(len(exp)+1)>13):

print("You should take the consultation from doctor. ")

else:

print("It might not be that bad but you should take precautions.")

def getDescription():

global description\_list

with open('MasterData/symptom\_Description.csv') as csv\_file:

csv\_reader = csv.reader(csv\_file, delimiter=',')

line\_count = 0

for row in csv\_reader:

\_description={row[0]:row[1]}

description\_list.update(\_description)

def getSeverityDict():

global severityDictionary

with open('MasterData/symptom\_severity.csv') as csv\_file:

csv\_reader = csv.reader(csv\_file, delimiter=',')

line\_count = 0

try:

for row in csv\_reader:

\_diction={row[0]:int(row[1])}

severityDictionary.update(\_diction)

except:

pass

def getprecautionDict():

global precautionDictionary

with open('MasterData/symptom\_precaution.csv') as csv\_file:

csv\_reader = csv.reader(csv\_file, delimiter=',')

line\_count = 0

for row in csv\_reader:

\_prec={row[0]:[row[1],row[2],row[3],row[4]]}

precautionDictionary.update(\_prec)

def getInfo():

print("-----------------------------------HealthCare ChatBot-----------------------------------")

print("\nYour Name? \t\t\t\t",end="->")

name=input("")

print("Hello, ",name)

def check\_pattern(dis\_list,inp):

pred\_list=[]

inp=inp.replace(' ','\_')

patt = f"{inp}"

regexp = re.compile(patt)

pred\_list=[item for item in dis\_list if regexp.search(item)]

if(len(pred\_list)>0):

return 1,pred\_list

else:

return 0,[]

def sec\_predict(symptoms\_exp):

df = pd.read\_csv('Data/Training.csv')

X = df.iloc[:, :-1]

y = df['prognosis']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=20)

rf\_clf = DecisionTreeClassifier()

rf\_clf.fit(X\_train, y\_train)

symptoms\_dict = {symptom: index for index, symptom in enumerate(X)}

input\_vector = np.zeros(len(symptoms\_dict))

for item in symptoms\_exp:

input\_vector[[symptoms\_dict[item]]] = 1

return rf\_clf.predict([input\_vector])

def print\_disease(node):

node = node[0]

val = node.nonzero()

disease = le.inverse\_transform(val[0])

return list(map(lambda x:x.strip(),list(disease)))

def tree\_to\_code(tree, feature\_names):

tree\_ = tree.tree\_

feature\_name = [

feature\_names[i] if i != \_tree.TREE\_UNDEFINED else "undefined!"

for i in tree\_.feature

]

chk\_dis=",".join(feature\_names).split(",")

symptoms\_present = []

while True:

print("\nEnter the symptom you are experiencing \t\t",end="->")

disease\_input = input("")

conf,cnf\_dis=check\_pattern(chk\_dis,disease\_input)

if conf==1:

print("searches related to input: ")

for num,it in enumerate(cnf\_dis):

print(num,")",it)

if num!=0:

print(f"Select the one you meant (0 - {num}): ", end="")

conf\_inp = int(input(""))

else:

conf\_inp=0

disease\_input=cnf\_dis[conf\_inp]

break

# print("Did you mean: ",cnf\_dis,"?(yes/no) :",end="")

# conf\_inp = input("")

# if(conf\_inp=="yes"):

# break

else:

print("Enter valid symptom.")

while True:

try:

num\_days=int(input("Okay. From how many days ? : "))

break

except:

print("Enter valid input.")

def recurse(node, depth):

indent = " " \* depth

if tree\_.feature[node] != \_tree.TREE\_UNDEFINED:

name = feature\_name[node]

threshold = tree\_.threshold[node]

if name == disease\_input:

val = 1

else:

val = 0

if val <= threshold:

recurse(tree\_.children\_left[node], depth + 1)

else:

symptoms\_present.append(name)

recurse(tree\_.children\_right[node], depth + 1)

else:

present\_disease = print\_disease(tree\_.value[node])

# print( "You may have " + present\_disease )

red\_cols = reduced\_data.columns

symptoms\_given = red\_cols[reduced\_data.loc[present\_disease].values[0].nonzero()]

# dis\_list=list(symptoms\_present)

# if len(dis\_list)!=0:

# print("symptoms present " + str(list(symptoms\_present)))

# print("symptoms given " + str(list(symptoms\_given)) )

print("Are you experiencing any ")

symptoms\_exp=[]

for syms in list(symptoms\_given):

inp=""

print(syms,"? : ",end='')

while True:

inp=input("")

if(inp=="yes" or inp=="no"):

break

else:

print("provide proper answers i.e. (yes/no) : ",end="")

if(inp=="yes"):

symptoms\_exp.append(syms)

second\_prediction=sec\_predict(symptoms\_exp)

# print(second\_prediction)

calc\_condition(symptoms\_exp,num\_days)

if(present\_disease[0]==second\_prediction[0]):

print("You may have ", present\_disease[0])

print(description\_list[present\_disease[0]])

# readn(f"You may have {present\_disease[0]}")

# readn(f"{description\_list[present\_disease[0]]}")

else:

print("You may have ", present\_disease[0], "or ", second\_prediction[0])

print(description\_list[present\_disease[0]])

print(description\_list[second\_prediction[0]])

# print(description\_list[present\_disease[0]])

precution\_list=precautionDictionary[present\_disease[0]]

print("Take following measures : ")

for i,j in enumerate(precution\_list):

print(i+1,")",j)

# confidence\_level = (1.0\*len(symptoms\_present))/len(symptoms\_given)

# print("confidence level is " + str(confidence\_level))

recurse(0, 1)

getSeverityDict()

getDescription()

getprecautionDict()

getInfo()

tree\_to\_code(clf,cols)

print("----------------------------------------------------------------------------------------")