**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP HÀNG TUẦN**

Họ và tên: Nguyễn Hoàng Dũng

Mã sinh viên: B18DCAT034

Điện thoại: 0369105658

# Tuần 1: (22/9 🡪 29/9)

Trong tuần qua em đã tìm hiểu một số khái niệm sau

1. Hiểu ngôn ngữ tự nhiên (NLU)

- Là thành phần quan trọng nhất của chatbot. Chatbot thông minh do phần này quyết định

- Mục tiêu của NLU là trích xuất thông tin từ câu nói hoặc viết của người dùng với 2 bước tuần tự:

* Phân loại ý định (intent classification): Xác định được ý định chẩn đoán bệnh
* Trích chọn thông tin: lấy ra được tên triệu chứng để chẩn đoán bệnh, cách đề phòng bệnh và trả lời cho người dùng

Các bước phân lớp:

* Tiền xử lý dữ liệu
* Trích chọn đặc trưng
* Huấn luyện mô hình
* Phân lớp

2. Xử lý ngôn ngữ tự nhiên (NLP)

- Mục tiêu: giúp chatbot hiểu được câu đầu vào của người dùng, xử lý và trích chọn các từ liên quan đến triệu chứng bệnh để phân loại bệnh tương ứng

- Các bước xử lý câu đầu vào:

Bước 1: Phân đoạn câu (Segmentation): Tách thành các câu riêng biệt

VD: My head hurts. My eyes are red 🡪 tách thành 2 câu:

My head hurts.

My eyes are red.

Bước 2: Mã hóa các từ (word tokenization): Chia câu thành các từ riêng lẻ, gọi là words hoặc tokens

VD: “my head hurts” 🡪 “My”, “head”, “hurts”

Bước 3: Xác định các từ dừng (stop words): Các từ xuất hiện thường xuyên trên câu, gây ra nhiều nhiễu

Bước 4: Stemming và Lemmatization:

Stemming là biến đổi từ về dạng gốc bằng cách loại bỏ một số ký tự hậu tố

VD: “works”, “working” , “worked” 🡪 “work”

Lemmatization là xử lý thông minh hơn bằng bộ từ điển hoặc một bộ ontology. Xử lý được các từ bất quy tắc

VD: “went”, “goes”, “go” 🡪 “go”

Bước 5: Gán nhãn   
3. Mạng nơ-ron

Là mô hình xử lý thông tin mô phỏng dựa trên hệ thần kinh sinh học, gồm số lượng các nơ-ron được gắn kết để xử lý thông tin

Mạng nơ-ron gồm 3 thành phần: Input Layer, Hidden Layer và Output layer

Lớp Hidden Layer nhận dữ liệu từ Input layer và chuyển đổi các input cho các lớp xử lý layer tiếp theo

Một mạng nơ ron có nhiều Hidden layer.

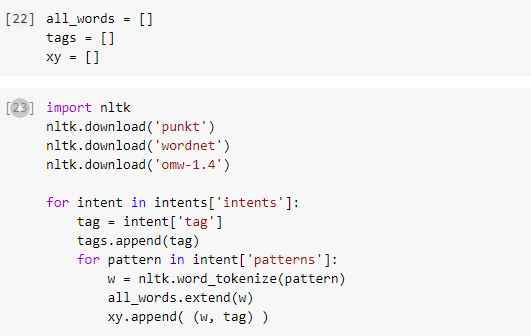
4. Code

- Đầu tiên em tạo ra bộ dữ liệu cho mô hình NLP theo dạng json

* tag là tên của triệu chứng
* patterns là những câu ví dụ nói về triệu chứng đó



- Sau đó em tách tag và pattern từ bộ dữ liệu và gán theo từng cặp tag – tokenized\_words

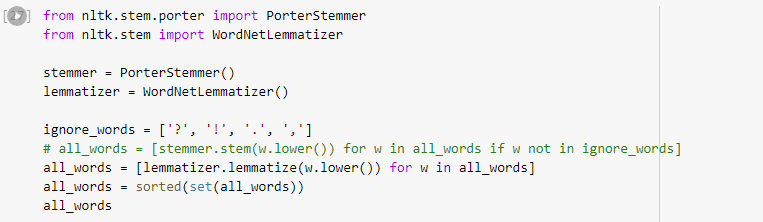


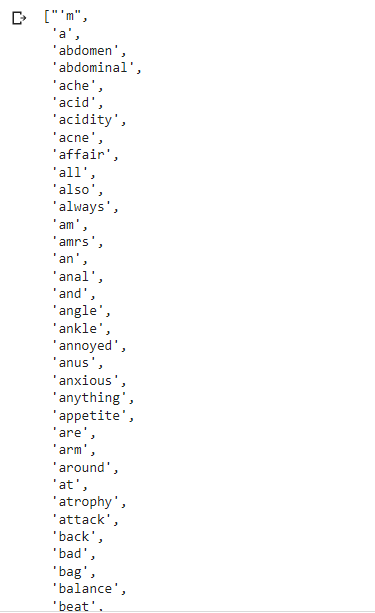


- Tạo ra bộ test cho xy

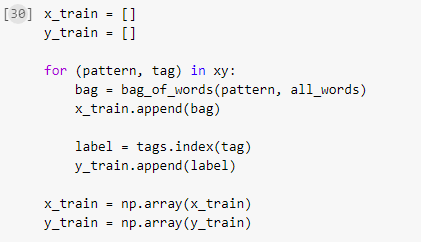


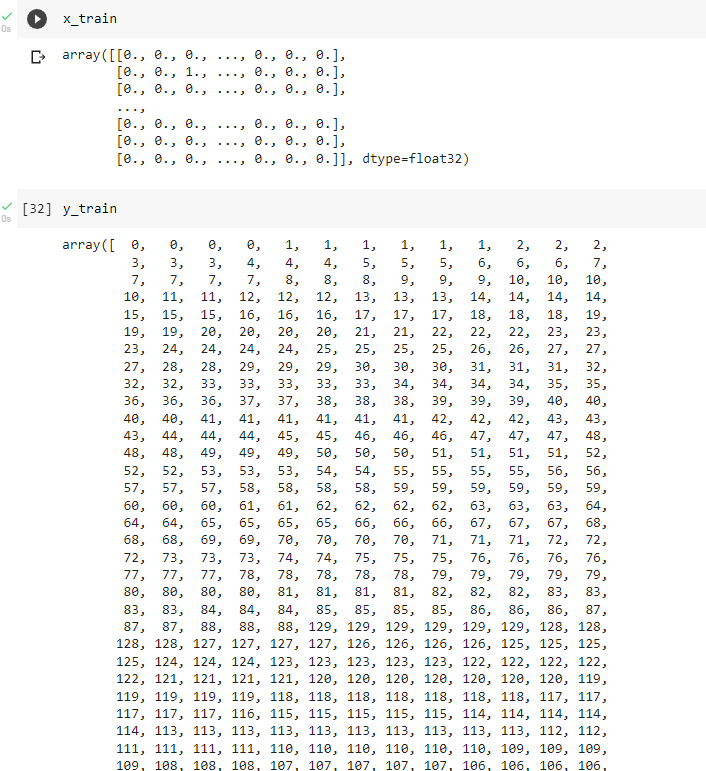
- Sử dụng lemmatize chuyển đổi về từ theo root form



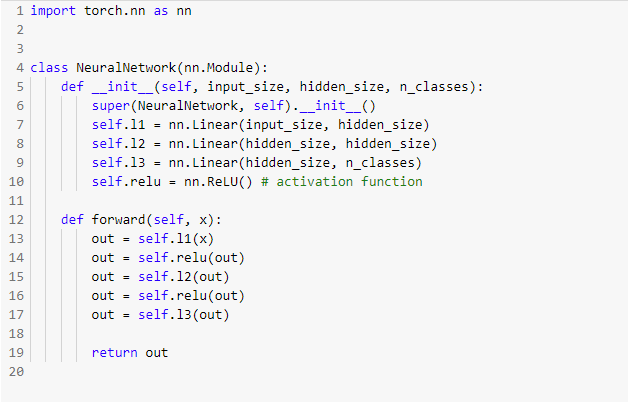


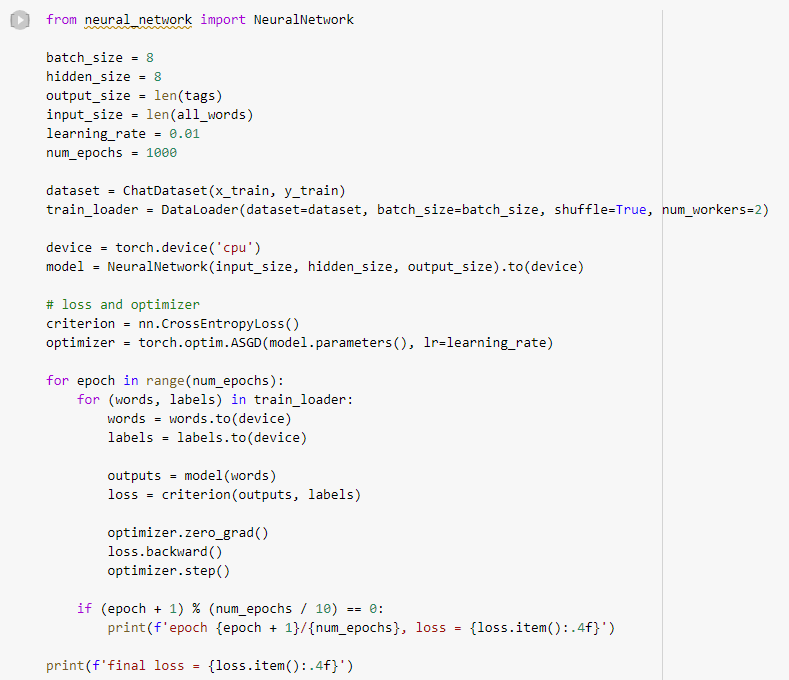
-Sử dụng phương pháp túi từ để tạo bộ dữ liệu huấn luyện

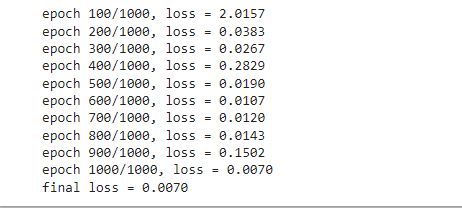




- Sử dụng mạng nơ-ron để train model







- Chạy model đã huấn luyện



Hiện tại model của em độ chính xác chưa được cao nên một vài phân loại còn sai ạ.

# Tuần 2: (30/9 🡪 7/10)

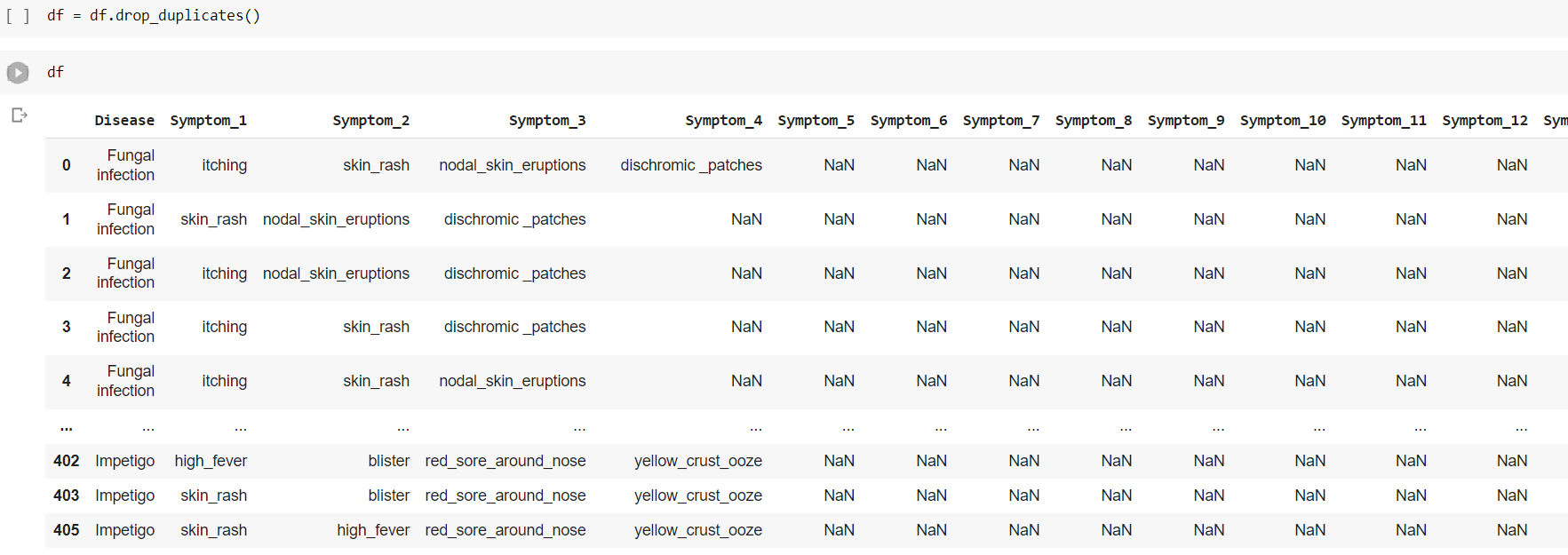
Tuần này em đã đọc tài liệu về các thuật toán áp dụng cho bài toán phân loại (Classification) và chạy thử

I. Chuẩn bị bộ dữ liệu về bệnh

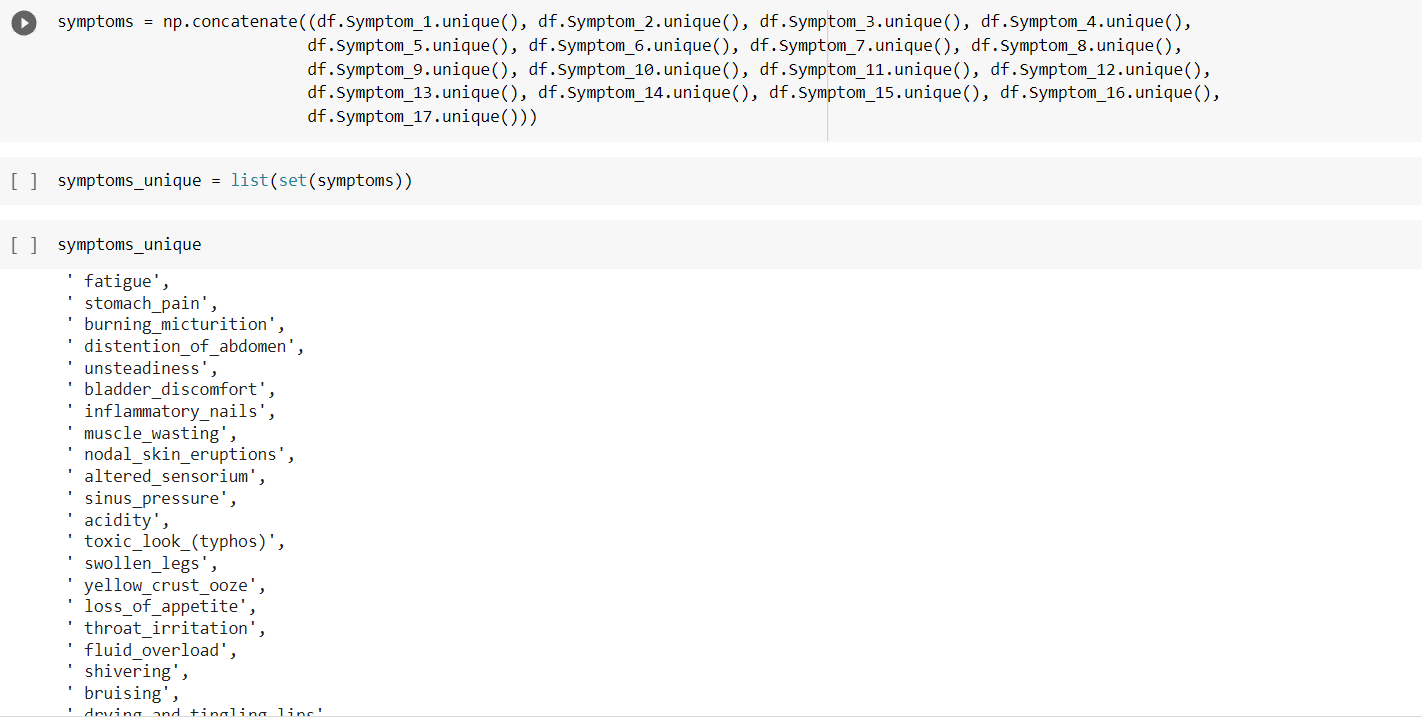
Link: <https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset>

- Đọc dữ liệu và tiền xử lý dữ liệu



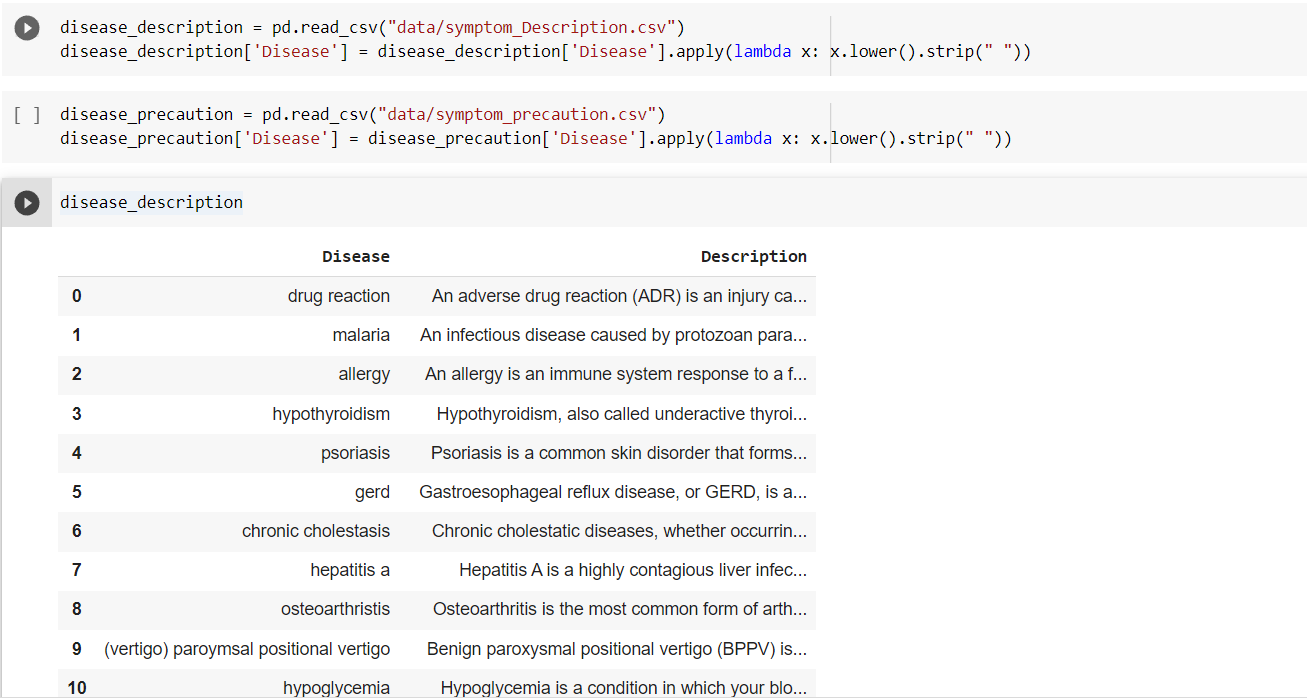


- Lấy danh sách các triệu chứng và gán vào bộ dữ liệu





- Đọc dữ liệu mô tả về bệnh và cách đề phòng



- các bước chuẩn bị cho train model



- Hàm kiểm chứng chéo để ước lượng sai số của mô hình



II. Sử dụng các thuật toán của học máy để phân loại

Sử dụng dữ liệu test ở trên để dự đoán bệnh



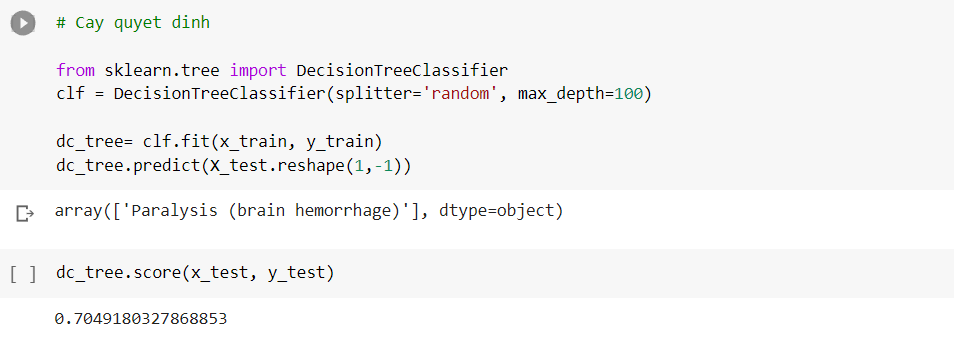
1. K láng giềng gần nhất (K nearest neighbor)



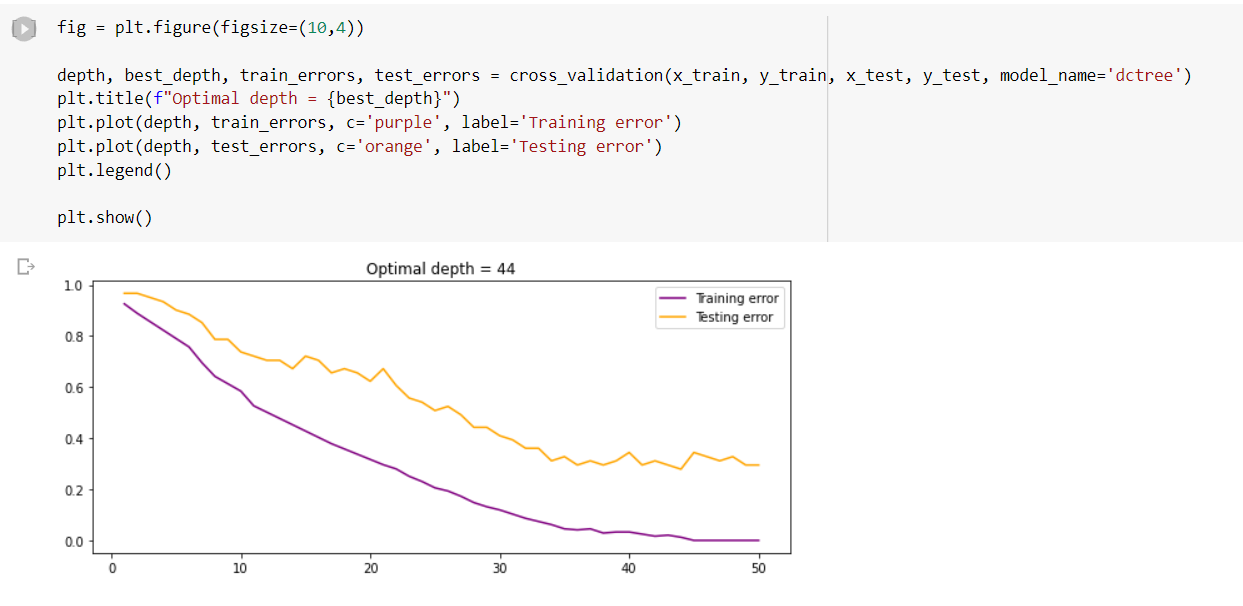
\* Kiểm chứng chéo cho thuật toán (trục hoành: độ phức tạp của mô hình; trục tung: Sai số)



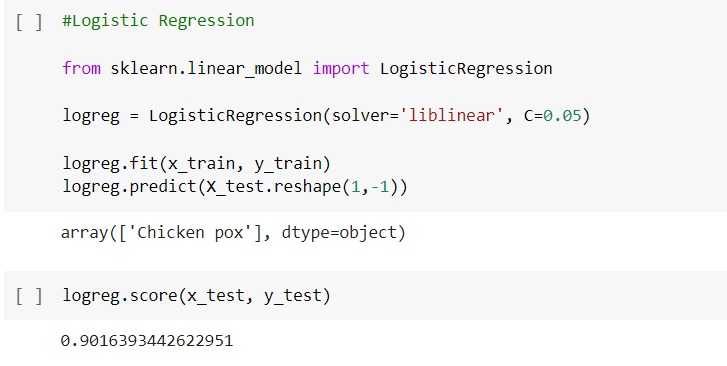
2. Cây quyết định (Decision Tree)



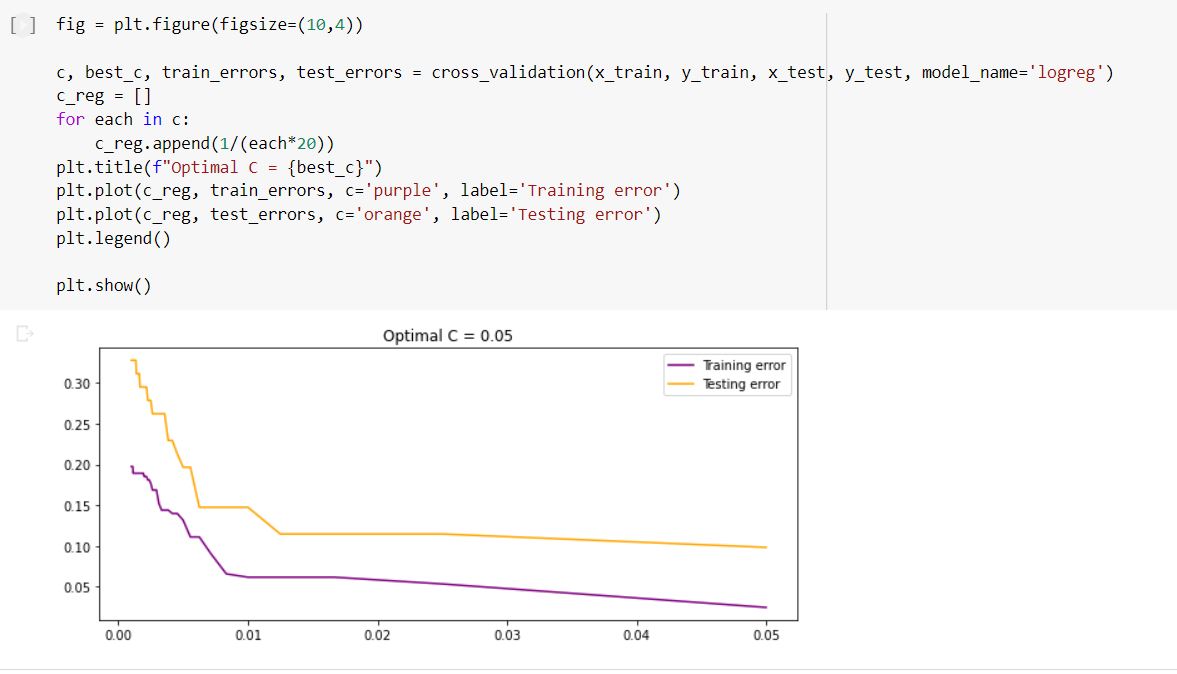
\* Kiểm chứng chéo cho thuật toán



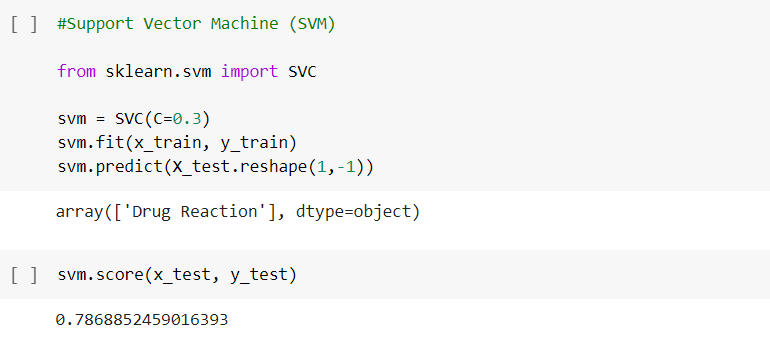
3. Hồi quy Logistic (Logistic Regression)



\*Kiểm chứng chéo cho thuật toán:



4. Support Vector machine (SVM)



\* Kiểm chứng chéo cho thuật toán



**Mục tiêu tuần tới:** Nhanh chóng hoàn thiện giao diện cho chatbot và chạy thử nhận diện bệnh trên giao diện

# Tuần 3 (8/10 🡪 14/10)

Tuần vừa qua em đã xây dựng giao diện cho chatbot trên Flask:

Sử dụng model đã huấn luyện

import json  
import torch  
import nltk  
import pickle  
import random  
from datetime import datetime  
import numpy as np  
import pandas as pd  
  
from models.neural\_network import NeuralNetwork  
from models.nlp\_utils import bag\_of\_words  
from flask import Flask, render\_template, url\_for, request, jsonify  
  
random.seed(datetime.now())  
  
device = torch.device('cpu')  
FILE = "models/data.pth"  
model\_data = torch.load(FILE)  
  
input\_size = model\_data['input\_size']  
hidden\_size = model\_data['hidden\_size']  
output\_size = model\_data['output\_size']  
all\_words = model\_data['all\_words']  
tags = model\_data['tags']  
model\_state = model\_data['model\_state']  
  
nlp\_model = NeuralNetwork(input\_size, hidden\_size, output\_size).to(device)  
nlp\_model.load\_state\_dict(model\_state)  
nlp\_model.eval()  
  
diseases\_description = pd.read\_csv("data/symptom\_Description.csv")  
diseases\_description['Disease'] = diseases\_description['Disease'].apply(lambda x: x.lower().strip(" "))  
  
disease\_precaution = pd.read\_csv("data/symptom\_precaution.csv")  
disease\_precaution['Disease'] = disease\_precaution['Disease'].apply(lambda x: x.lower().strip(" "))  
  
symptom\_severity = pd.read\_csv("data/Symptom-severity.csv")  
symptom\_severity = symptom\_severity.applymap(lambda s: s.lower().strip(" ").replace(" ", "") if type(s) == str else s)  
  
with open('data/list\_of\_symptoms.pickle', 'rb') as data\_file:  
 symptoms\_list = pickle.load(data\_file)  
  
with open('data/fitted\_model.pickle', 'rb') as modelFile:  
 prediction\_model = pickle.load(modelFile)  
  
user\_symptoms = set()

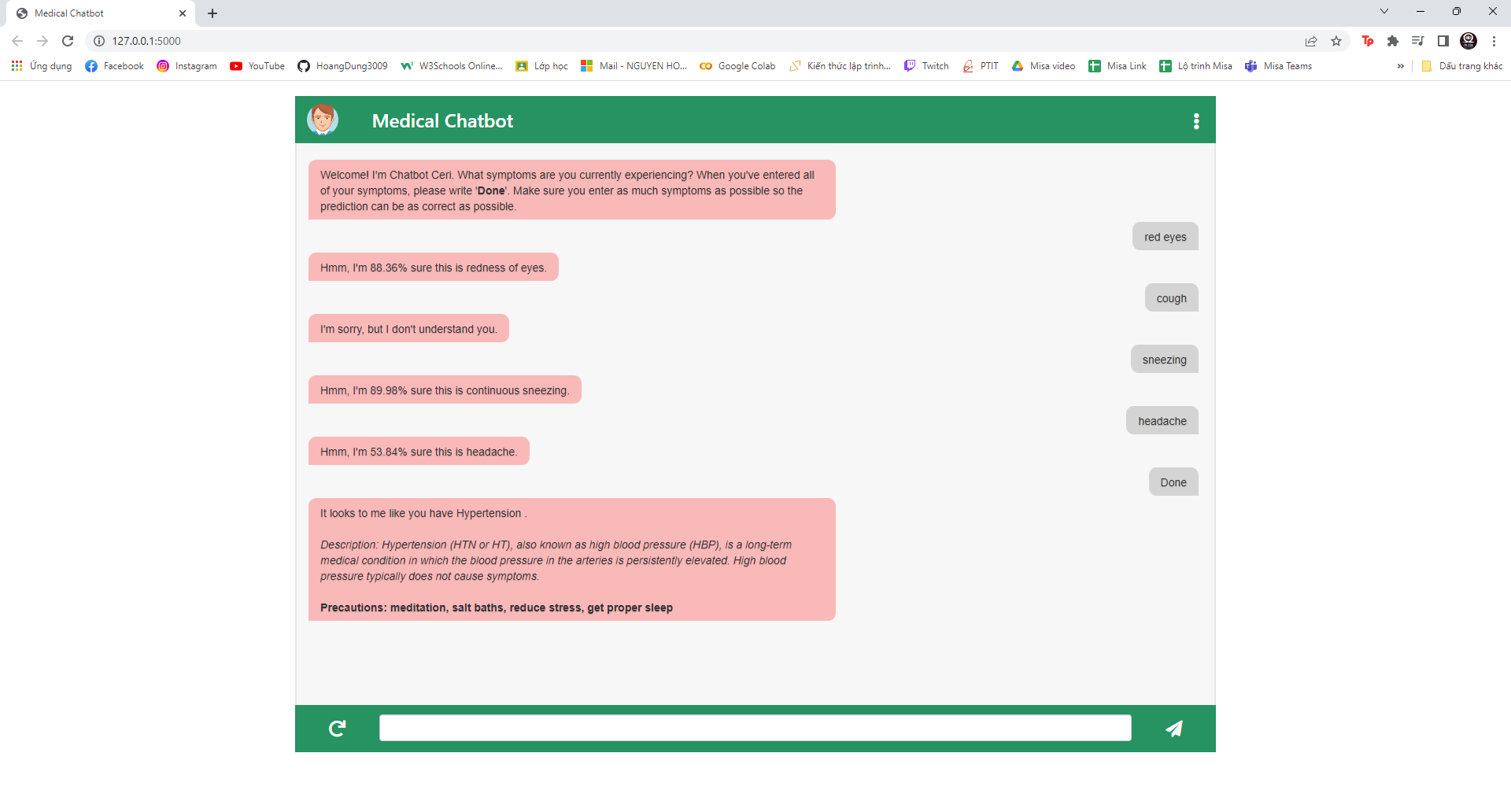
Lấy triệu chứng từ input của người dùng

def get\_symptom(sentence):  
 sentence = nltk.word\_tokenize(sentence)  
 X = bag\_of\_words(sentence, all\_words)  
 X = X.reshape(1, X.shape[0])  
 X = torch.from\_numpy(X)  
  
 output = nlp\_model(X)  
 \_, predicted = torch.max(output, dim=1)  
 tag = tags[predicted.item()]  
  
 probs = torch.softmax(output, dim=1)  
 prob = probs[0][predicted.item()]  
 prob = prob.item()  
  
 return tag, prob

Dự đoán bệnh

@app.route('/symptom', methods=['GET', 'POST'])  
def predict\_symptom():  
 print("Request json:", request.json)  
 sentence = request.json['sentence']  
 if sentence.replace(".", "").replace("!", "").lower().strip() == "done":  
  
 if not user\_symptoms:  
 response\_sentence = random.choice(  
 ["I can't know what disease you may have if you don't enter any symptoms :)",  
 "Ceri can't know the disease if there are no symptoms...",  
 "You first have to enter some symptoms!"])  
 else:  
 x\_test = []  
  
 for each in symptoms\_list:  
 if each in user\_symptoms:  
 x\_test.append(1)  
 else:  
 x\_test.append(0)  
  
 x\_test = np.asarray(x\_test)  
 disease = prediction\_model.predict(x\_test.reshape(1, -1))[0]  
 print(disease)  
  
 description = \  
 diseases\_description.loc[diseases\_description['Disease'] == disease.strip(" ").lower(), 'Description'].iloc[  
 0]  
 precaution = disease\_precaution[disease\_precaution['Disease'] == disease.strip(" ").lower()]  
 precautions = 'Precautions: ' + precaution.Precaution\_1.iloc[0] + ", " + precaution.Precaution\_2.iloc[  
 0] + ", " + precaution.Precaution\_3.iloc[0] + ", " + precaution.Precaution\_4.iloc[0]  
 response\_sentence = "It looks to me like you have " + disease + ". <br><br> <i>Description: " + description + "</i>" + "<br><br><b>" + precautions + "</b>"  
  
 severity = []  
  
 for each in user\_symptoms:  
 severity.append(symptom\_severity.loc[symptom\_severity['Symptom'] == each.lower().strip(" ").replace(" ", ""), 'weight'].iloc[0])  
  
 if np.mean(severity) > 4 or np.max(severity) > 5:  
 response\_sentence = response\_sentence + "<br><br>Considering your symptoms are severe, and Ceri " \  
 "isn't a real doctor, you should consider talking to one. :) "  
 user\_symptoms.clear()  
 severity.clear()  
  
 else:  
 symptom, prob = get\_symptom(sentence)  
 print("Symptom:", symptom, ", prob:", prob)  
 if prob > .5:  
 response\_sentence = f"Hmm, I'm {(prob \* 100):.2f}% sure this is " + symptom + "."  
 user\_symptoms.add(symptom)  
 else:  
 response\_sentence = "I'm sorry, but I don't understand you."  
  
 print("User symptoms:", user\_symptoms)  
  
 return jsonify(response\_sentence.replace("\_", " "))

Kết quả:



Đánh giá:

* Botchat của em mới chỉ hiểu từ đơn, cần huấn luyện cho hiểu được các câu dài
* Thêm kịch bản hội thoại như chào hỏi, chào tạm biệt, cảm ơn…
* Thêm phần nhận diện giọng nói

Rất mong thầy đánh giá cho em để em có thể hoàn thiện hơn ạ.

# Tuần 4 (15/10 🡪 21/10)

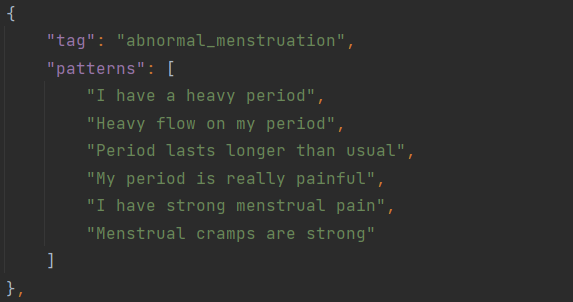
Xây dựng 2 hàm cho voice recognition

import speech\_recognition as sr  
  
  
def Listen():  
 r = sr.Recognizer()  
 with sr.Microphone() as source:  
 print("Listening... ")  
 r.pause\_threshold = 1  
 audio = r.listen(source,0,2)  
  
 try:  
 print("Recognizing...")  
 query = r.recognize\_google(audio, language='en-us')  
 print(f"You: {query}")  
 except:  
 return ""  
  
 query = str(query)  
 return query.lower()

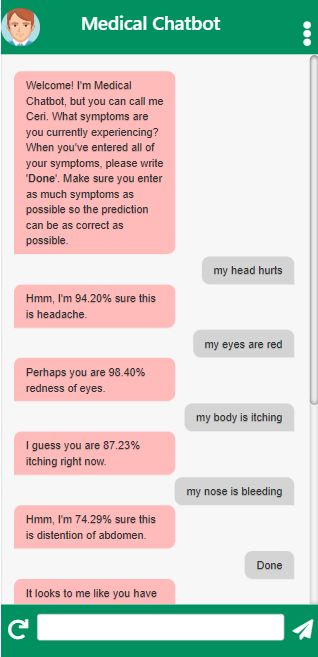
import pyttsx3  
  
  
def Say(Text):  
 engine = pyttsx3.init()  
 voices = engine.getProperty('voices')  
 engine.setProperty('voices', voices[1].id)  
 engine.setProperty('rate', 180)  
 print(f"Ciri: {Text}")  
 engine.say(text=Text)  
 engine.runAndWait()

Thêm các pattern là các câu trả lời để chatbot có thể hiểu rõ về câu của người dùng và huấn luyện lại model

VD: Sửa pattern của 1 triệu chứng bằng cách thêm các câu phản hồi về triệu chứng đó, càng nhiều câu thì bot sẽ hiểu rõ hơn và nhận diện được triệu chứng chuẩn hơn



Thêm giao diện Mobile khi thu nhỏ trang web



Tuần này em bắt đầu viết báo cáo nên làm được hơi ít ạ.

Em đang tìm hiểu xem liệu có hosting cho chạy online ạ.