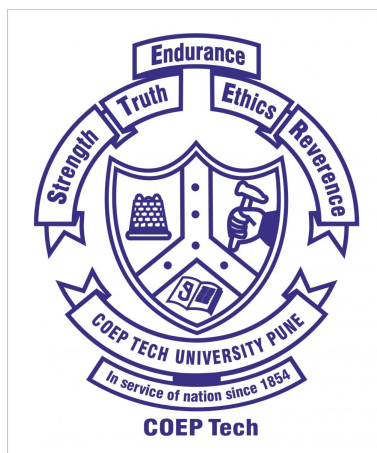


A
Report
on
Disease Prediction using Symptoms
submitted in partial fulfillment of the requirements
for completion of Data Science Project
of
THIRD YEAR
in
Computer Engineering
by
112003004 Advait Karmalkar
112003021 Aditya Bornare
112003032 Aneesh Damle
Under the guidance of
Mrs. Archana Patil
Professor
Department of Computer Engineering



Project Title:

Disease Prediction using Symptoms

Problem Statement:

Detection of Diseases is one of the preliminary steps in the treatment of a disease. Our project aims to detect diseases smartly based on the symptoms entered. We propose to implement a disease prediction system based on symptoms by processing and analysing relevant symptom data.

Issues in existing methods:

1. Many diseases with common symptoms
2. Time consuming tests
3. Human Error

Proposed Approach:

One viable solution to the problems defined above is to make use of highly sophisticated data mining and ML (Machine Learn-

ing) algorithms to classify and detect the diseases. Machine Learning applications in healthcare and biomedical domain has lead to early disease detection and better diagnosis. This has enhanced patient care in recent times.

Currently, there are many disease detection systems available such as heart disease prediction, neurological disorders prediction, and skin disease prediction. But universal prediction system for diseases based on symptoms is rarely in practice. Also, in many cases of frequently occurring diseases like fever, common cold, etc. users do not want to spend money and go through tests.

The proposed approach is to take the symptoms as input from the user in common and natural language and perform query expansion using the synonyms of each symptom and match with the symptoms present in the dataset and prompt the user to select their symptoms. As a real-world doctor would ask the patient more about any other symptoms they are having, by asking the co-occurring symptoms with the ones which were initially stated by patient, this is applied to the proposed system. The user is prompted till they have marked all the symptoms. These symptoms are then given as input to the trained model to make predic-

tions. Top 10 most probable detected diseases are shown to the user with their respective independent probabilities or appropriate score.

Web scraping:

Scraping of dataset consists 2 parts:

1. Diseases — Diseases are scraped from the National Health Portal of India, developed and maintained by Center for Health Informatics (CHI). This is combined with a predefined list of diseases to account more diseases in the final prepared dataset.
2. Symptoms — Symptoms are scraped using a script that uses the Google Search package to perform searching and fetch the disease's Wikipedia page among the various search results obtained. The HTML code of the page is processed to fetch the symptoms of the disease using the 'infobox' available on the Wikipedia page. Figure shows an example of Wikipedia's infobox.

Final count of diseases in the dataset were a total of 261 and 488 symptoms.

```

webscraping > + scraping.py > ...
1  from selenium import webdriver
2  from selenium.webdriver.chrome.options import Options
3  from selenium.webdriver.common.by import By
4  from googlesearch import search
5  from bs4 import BeautifulSoup
6  import warnings
7  import pickle
8  warnings.filterwarnings("ignore")
9  import requests
10 import time
11 import re
12
13 DRIVER_PATH = '/Downloads/chromedriver_linux64/chromedriver'
14 options = Options()
15 options.headless = True
16 options.add_argument("--window-size=1920,1200")
17
18
19
20 small_alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
21 diseases=[]
22 '''f = open('data.csv', 'w')'''
23 for c in small_alpha:
24     URL = 'https://www.nhp.gov.in/disease-a-z/'+c
25     time.sleep(1)
26     driver = webdriver.Chrome(options = options, executable_path=DRIVER_PATH)
27     driver.get(URL)
28     all_diseases = driver.find_element(By.CLASS_NAME, 'all-disease')
29     for element in all_diseases.find_elements(By.TAG_NAME, 'li'):
30         diseases.append(element.text.strip())
31 'f.close()'
32 '''for a in diseases:
33     print(a)'''
34 #print(diseases)
35
36 driver.quit()
37
38 with open('list_diseaseNames.pkl', 'rb') as handle:
39     diseases2 = pickle.load(handle)

```

Figure 1: Web scraping code

Preprocessing of Data set:

The scraped symptoms are preprocessed to remove similar symptoms with different names. To do so, symptoms are expanded by appending synonyms of terms in the symptom string and computing Jaccard Similarity Coefficient for each pair of symptoms. We have used threshold as 0.75. The synonyms are taken from Thesaurus.com and Princeton University's Wordnet. If the score is greater than the threshold, both symptoms are alike and one can be used interchangeably used in place of other.

Preprocessing of User-entered Symptoms:

The system accepts symptom(s) in a single line, separated by comma (,). Subsequently, the following preprocessing steps are involved:

- Split symptoms into a list based on comma
- Convert the symptoms into lowercase
- Removal of stop words
- Tokenization of symptoms to remove any punctuation marks
- Lemmatization of tokens in the symptoms

Symptom matching and selection:

Each user symptom is expanded by appending a list of synonyms of the terms in the synonym string. The expanded symptom query is used to find the related symptoms in the dataset. To find such symptoms, each symptom from the dataset is split into tokens and each token is checked for its presence in the expanded query. Based on this, a similarity score is calculated and if the symptom's score is more than the threshold value, that symptom qualifies for being similar to the user's symptom and is suggested to the user.

The user selects one or more symptoms from the list. Based on the selected symptoms, other symptoms are shown to the user for selection which is among the top co-occurring symptoms with the ones selected by the user initially. The user can select any symptom, skip, or stop the symptom selection process. The final list of symptoms is compiled and shown to the user.

Prediction Using Model:

A binary vector is computed that consists of 1 for the symptoms present in the user's selection list and 0 otherwise. A machine

learning model is trained on the dataset, which is used here for prediction. The model accepts the symptom vector and outputs a list of top K diseases, sorted in the decreasing order of individual probabilities. Multinomial Naïve Bayes, Random Forest, K-Nearest Neighbor, Logistic Regression, Decision Tree were trained and tested with a train-test split of 90:10. Evaluation of the dataset is done by applying various machine learning algorithms and comparing the accuracy obtained from them. The highest accuracy is reported by Decision Tree (91.97%) while the lowest is of Multinomial Naïve Bayes (85.63%).

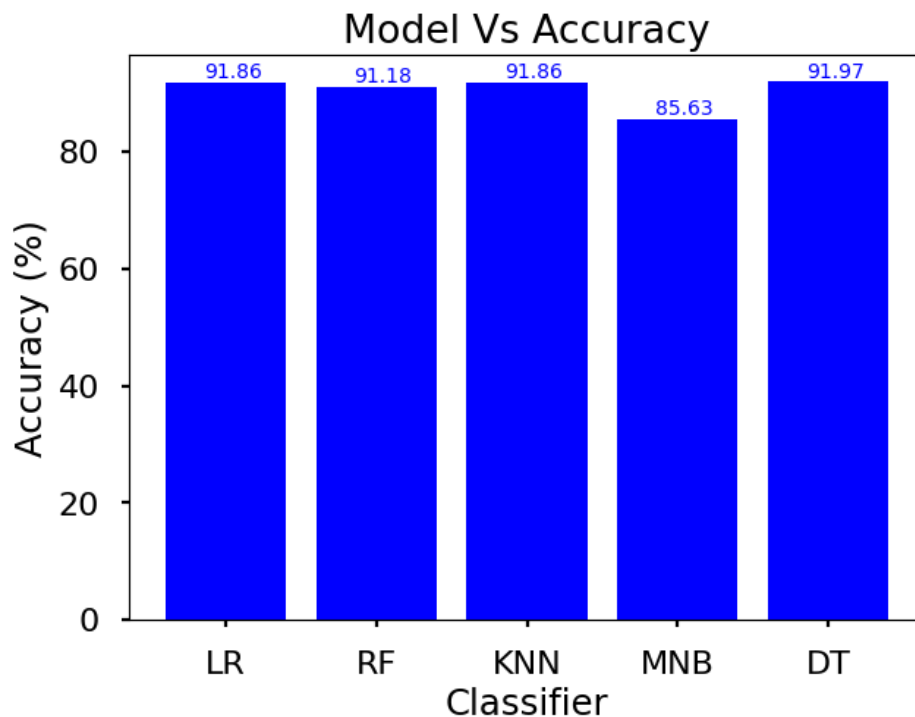
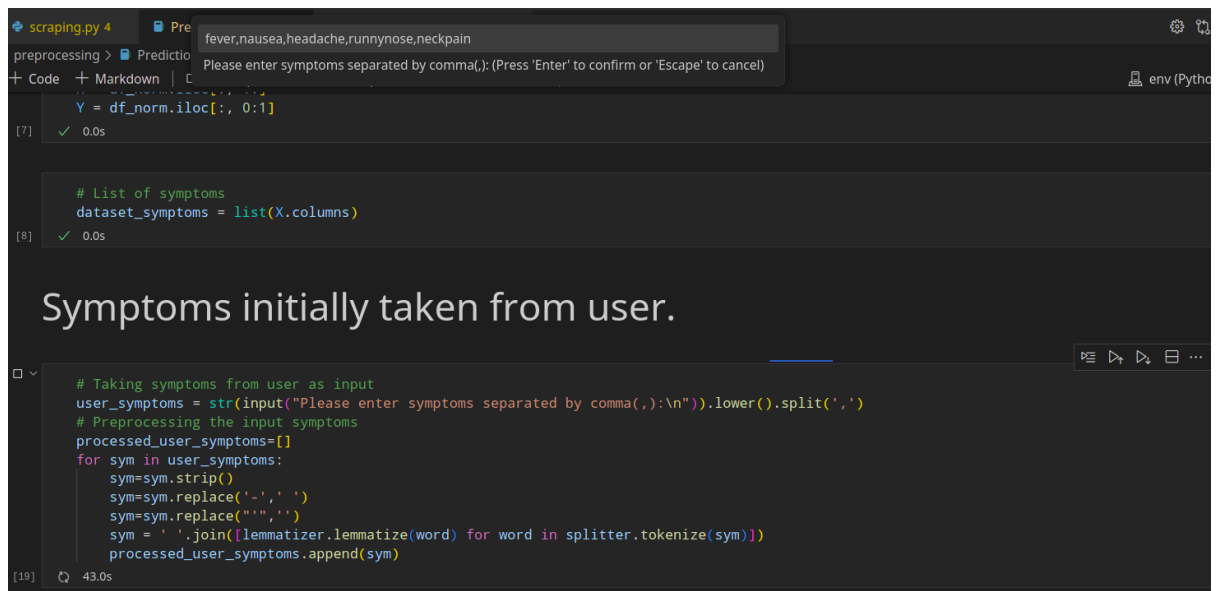


Figure 2: Model Accuracy

Challenges Faced:

- **Web scraping** - diseases were scraped from www.nhp.gov.in, but scraping symptoms was a difficult task as the same site mentioned them in paragraphs.
- **Symptom scraping** - we shifted to Wikipedia; the HTML code of the page is processed to fetch the symptoms of the disease using the `;infobox;` available on the Wikipedia page.
- **User input processing** – User input was a challenging task, as user is not bound to symptom set we generated. So, we had to match synonyms from Thesaurus for each user symptom and check their similarity scores with our symptom dataset, and provide a final appended list of symptoms for better disease prediction.

Output Screenshots:



```
scrapping.py 4 | Pre
preprocessing > Prediction
Please enter symptoms separated by comma(,): (Press 'Enter' to confirm or 'Escape' to cancel)
fever,nausea,headache,runnynose,neckpain

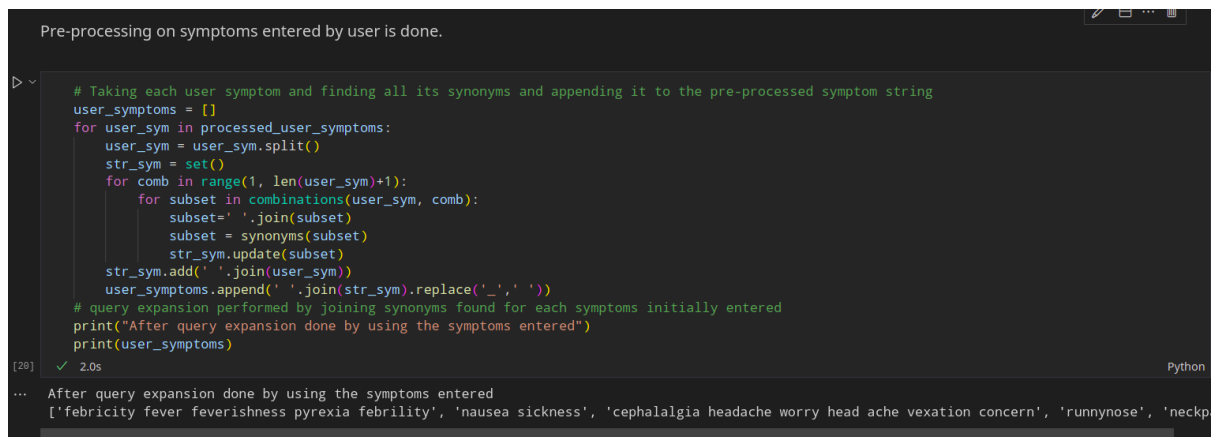
Y = df_norm.iloc[:, 0:1]
[7] ✓ 0.0s

# List of symptoms
dataset_symptoms = list(X.columns)
[8] ✓ 0.0s

Symptoms initially taken from user.

# Taking symptoms from user as input
user_symptoms = str(input("Please enter symptoms separated by comma(,):\n")).lower().split(',')
# Preprocessing the input symptoms
processed_user_symptoms=[]
for sym in user_symptoms:
    sym=sym.strip()
    sym=sym.replace('-', ' ')
    sym=sym.replace("'", '')
    sym = ' '.join([lemmatizer.lemmatize(word) for word in splitter.tokenize(sym)])
    processed_user_symptoms.append(sym)
[19] ✓ 43.0s
```

Figure 3: Input Symptoms



```
Pre-processing on symptoms entered by user is done.

# Taking each user symptom and finding all its synonyms and appending it to the pre-processed symptom string
user_symptoms = []
for user_sym in processed_user_symptoms:
    user_sym = user_sym.split()
    str_sym = set()
    for comb in range(1, len(user_sym)+1):
        for subset in combinations(user_sym, comb):
            subset=' '.join(subset)
            subset = synonyms(subset)
            str_sym.update(subset)
        str_sym.add(' '.join(user_sym))
    user_symptoms.append(' '.join(str_sym).replace('-', ' '))
# query expansion performed by joining synonyms found for each symptoms initially entered
print("After query expansion done by using the symptoms entered")
print(user_symptoms)
[20] ✓ 2.0s Python

... After query expansion done by using the symptoms entered
['febricity fever feverishness pyrexia febrility', 'nausea sickness', 'cephalalgia headache worry head ache vexation concern', 'runnynose', 'neckp
```

Figure 4: Preprocessing

Prompt the user to select the relevant symptoms by entering the corresponding indices.

```
# Print all found symptoms
print("Top matching symptoms from your search!")
for idx, symp in enumerate(found_symptoms):
    print(idx,":",symp)

# Show the related symptoms found in the dataset and ask user to select among them
select_list = input("\nPlease select the relevant symptoms. Enter indices (separated-space):\n").split()

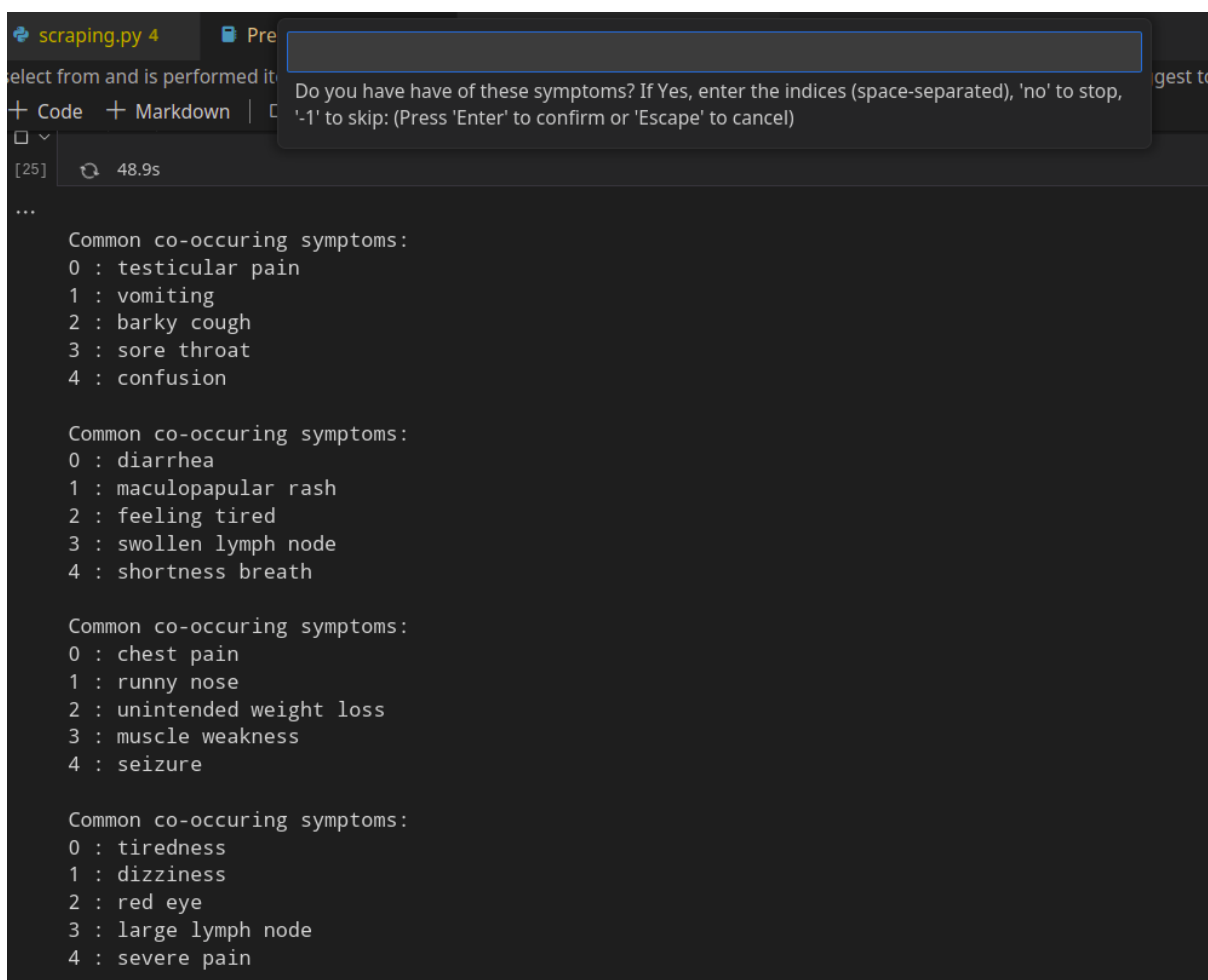
# Find other relevant symptoms from the dataset based on user symptoms based on the highest co-occurrence with the
# ones that is input by the user
dis_list = set()
final_symp = []
counter_list = []
for idx in select_list:
    symp=found_symptoms[int(idx)]
    final_symp.append(symp)
    dis_list.update(set(df_norm[df_norm[symp]==1]['label_dis']))

for dis in dis_list:
    row = df_norm.loc[df_norm['label_dis'] == dis].values.tolist()
    row[0].pop(0)
    for idx,val in enumerate(row[0]):
        if val!=0 and dataset_symptoms[idx] not in final_symp:
            counter_list.append(dataset_symptoms[idx])

[23] ✓ 9.5s Python
```

... Top matching symptoms from your search!
0 : nausea
1 : headache
2 : fever

Figure 5: Matching Symptoms



The screenshot shows a Jupyter Notebook window with a file named 'scraping.py 4'. A modal dialog box is open, asking: 'Do you have have of these symptoms? If Yes, enter the indices (space-separated), 'no' to stop, '-1' to skip: (Press 'Enter' to confirm or 'Escape' to cancel)'. Below the dialog, the notebook output displays four lists of common co-occurring symptoms, each with five items indexed from 0 to 4.

```
...  
Common co-occurring symptoms:  
0 : testicular pain  
1 : vomiting  
2 : barky cough  
3 : sore throat  
4 : confusion  
  
Common co-occurring symptoms:  
0 : diarrhea  
1 : maculopapular rash  
2 : feeling tired  
3 : swollen lymph node  
4 : shortness breath  
  
Common co-occurring symptoms:  
0 : chest pain  
1 : runny nose  
2 : unintended weight loss  
3 : muscle weakness  
4 : seizure  
  
Common co-occurring symptoms:  
0 : tiredness  
1 : dizziness  
2 : red eye  
3 : large lymph node  
4 : severe pain
```

Figure 6: Common co-occurring symptoms

```
Final Symptom list

# Create query vector based on symptoms selected by the user
print("\nFinal list of Symptoms that will be used for prediction:")
sample_x = [0 for x in range(0,len(dataset_symptoms))]
for val in final_symp:
    print(val)
    sample_x[dataset_symptoms.index(val)]=1

[26] ✓ 0.0s

...

Final list of Symptoms that will be used for prediction:
nausea
headache
fever
vomiting
confusion
diarrhea
feeling tired
shortness breath
runny nose
muscle weakness
```

Figure 7: Final symptoms

```
...

Top 10 diseases predicted based on symptoms
0 Disease name: Anthrax Probability: 48.65%
1 Disease name: Iron Deficiency Anemia Probability: 40.54%
2 Disease name: Acute encephalitis syndrome Probability: 40.54%
3 Disease name: Carbon monoxide poisoning Probability: 40.54%
4 Disease name: Japanese Encephalitis Probability: 40.54%
5 Disease name: Influenza Probability: 40.54%
6 Disease name: Crimean Congo haemorrhagic fever (CCHF) Probability: 40.54%
7 Disease name: Hepatitis A Probability: 40.54%
8 Disease name: Anaemia Probability: 32.43%
9 Disease name: Black Death Probability: 32.43%
```

Figure 8: Final predicted diseases

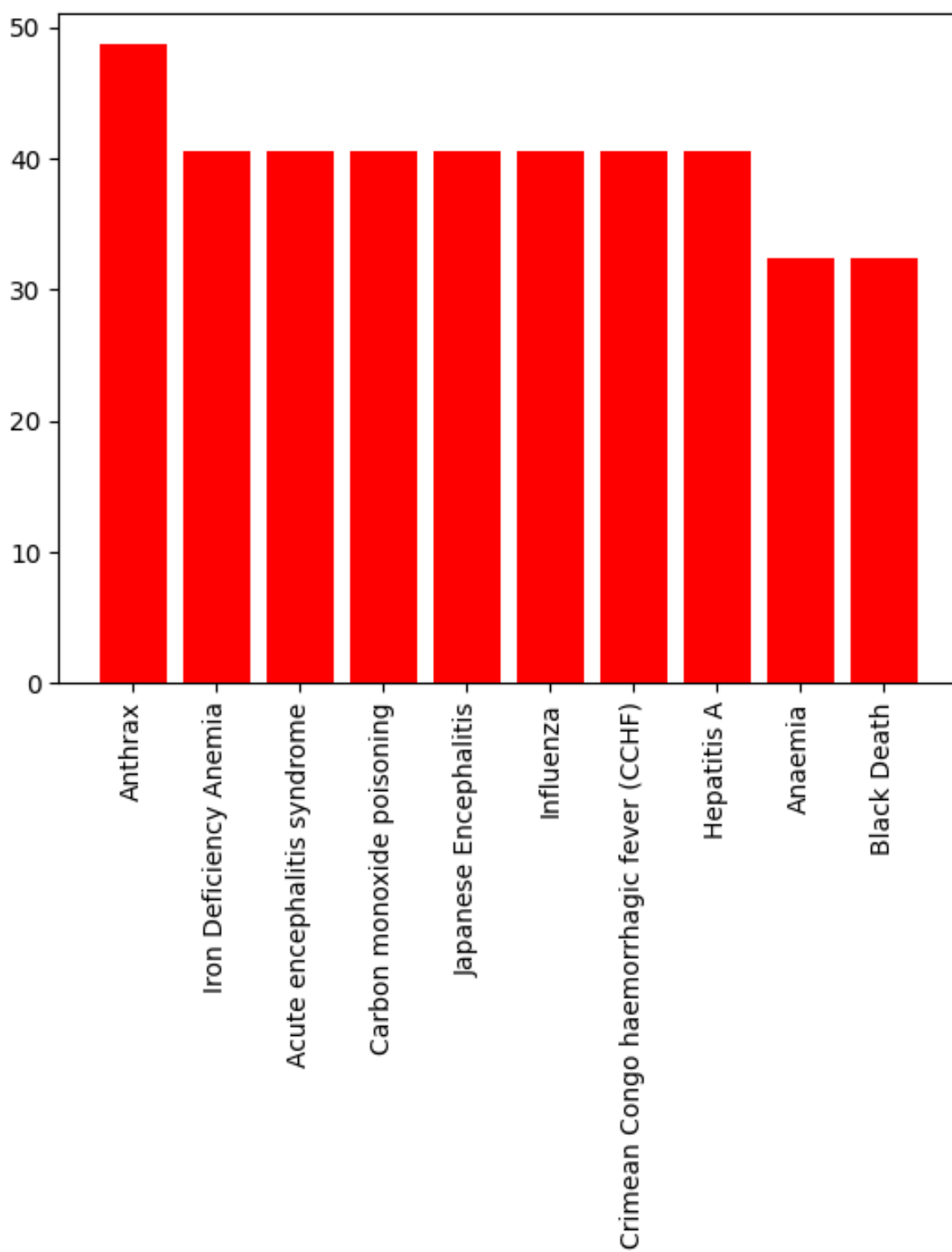


Figure 9: Final predicted diseases visualization