DATA SCIENCE PROJECT Disease Prediction Using Symptoms

MIS :

- 112003004 : Advait Karmalkar
- 112003021 : Aditya Bornare
- 112003032 : Aneesh Damle

PROBLEM STATEMENT

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

Web Scraping

Tools used: Beautiful Soup, Chrome Driver

- Scraping consists of 2 parts:

1. **Diseases**

- Disease data is scraped from https://www.nhp.gov.in/disease-a-z/
- All the diseases [A-Z] act as rows in the dataset generated

2. **Symptoms**

- Symptom data is generated by parsing the Wikipedia column for each disease fetched above
- The HTML code of the page is processed to fetch the symptoms of the disease using the <infobox> available on the Wikipedia page
- Symptoms act as columns in the generated dataset

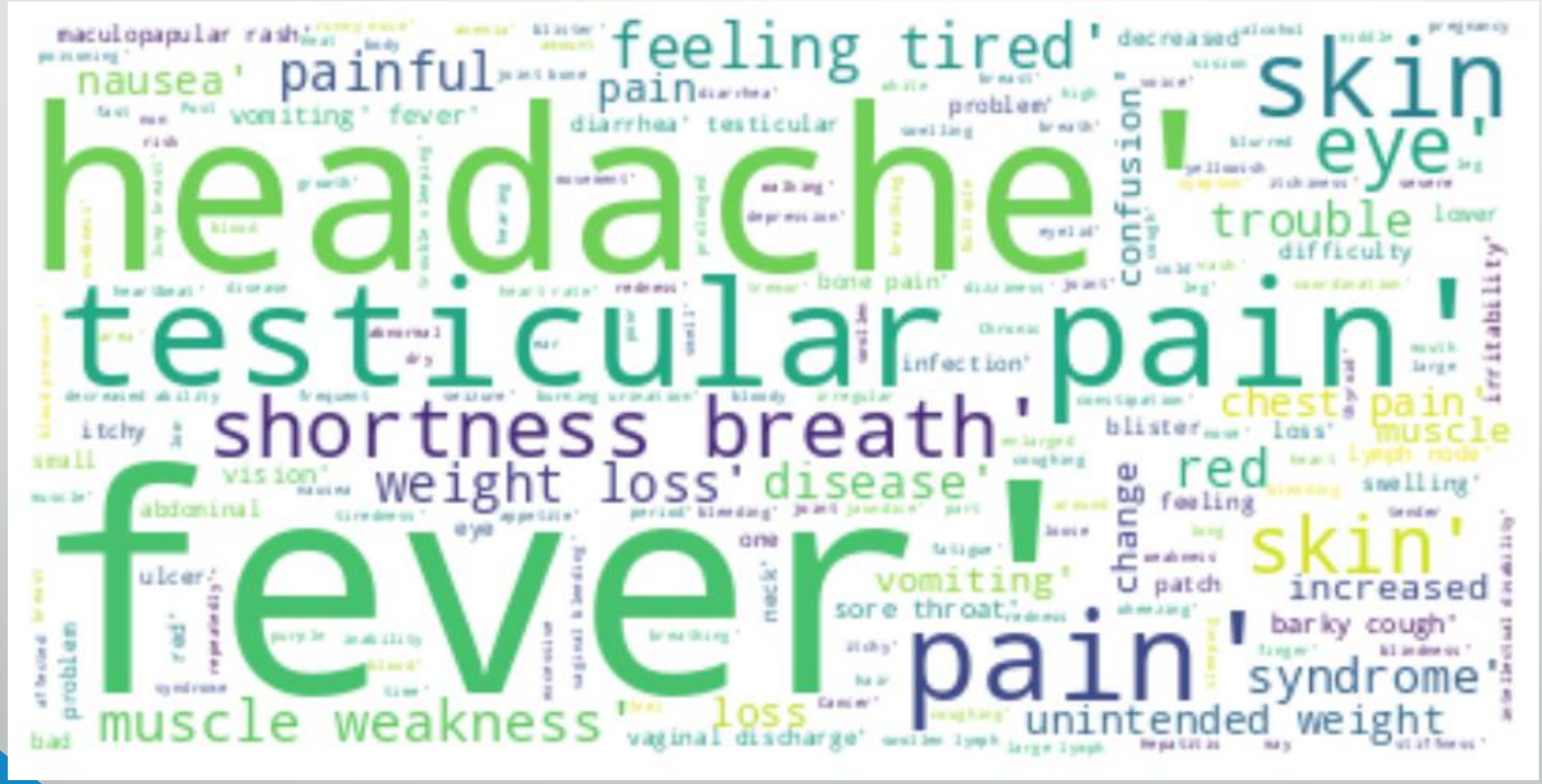
```
df = pd.read_csv("dis_sym_dataset_norm.csv")
display(df)
```

✓ 0.1s

	label_dis	abdominal cramp	abdominal distention	abnormal behavior	abnormal bleeding	abnormal sensation	abnormally frequent	abscess	aching	acne	...
0	Abscess	0	0	0	0	0	0	0	0	0	...
1	Acquired Capillary Haemangioma of Eyelid	0	0	0	0	0	0	0	0	0	...
2	Acquired Immuno Deficiency Syndrome	0	0	0	0	0	0	0	0	0	...
3	Acute encephalitis syndrome	0	0	0	0	0	0	0	0	0	...
4	Adult Inclusion Conjunctivitis	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
256	Yaws	0	0	0	0	0	0	0	0	0	...
257	Yellow Fever	0	0	0	0	0	0	0	0	0	...
258	Zika virus disease	0	0	0	0	0	0	0	0	0	...
259	lactose intolerance	0	0	0	0	0	0	0	0	0	...
260	papilloedema	0	0	0	0	0	0	0	0	0	...

261 rows × 490 columns

Visualization of Symptom Dataset



PREPROCESSING

DATA SET PREPROCESSING

1. The scraped symptoms are preprocessed to remove similar symptoms with different names.
2. 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.
3. We have used threshold as 0.75. The synonyms are taken from Thesaurus.com and Princeton University's Wordnet.
4. If the score is greater than the threshold, both symptoms are alike and one can be used interchangeably used in place of other.

USER SYMPTOM PREPROCESSING

1. When user enters symptoms, following preprocessing steps are involved:

a. Split symptoms into a list based on comma

b. Convert the symptoms into lowercase

c. Removal of stop words

d. Tokenization of symptoms to remove any punctuation marks

e. Lemmatization of tokens in the symptoms

f. Symptom synonyms are considered from Thesaurus and NLTK Wordnet

2. Similarity score is matched with existing symptoms

scraping.py 4 Pre

preprocessing > Predictio

+ Code + Markdown

fever,nausea,headache,runnynose,neckpain

Please enter symptoms separated by comma(,): (Press 'Enter' to confirm or 'Escape' to cancel)

```
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

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

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

Symptom Matching and Selection

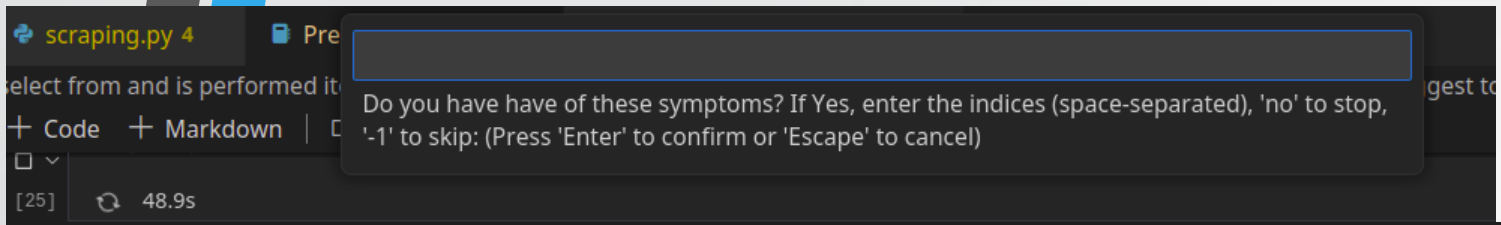
- Each user symptom is expanded by appending a list of synonyms of the terms in the synonym string.
- To find related symptoms, each symptom from the dataset is split into tokens and each token is checked for its presence in the expanded query. Based on this, a similarity score is calculated and if the 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.

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

```python
# 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-occurance 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
```



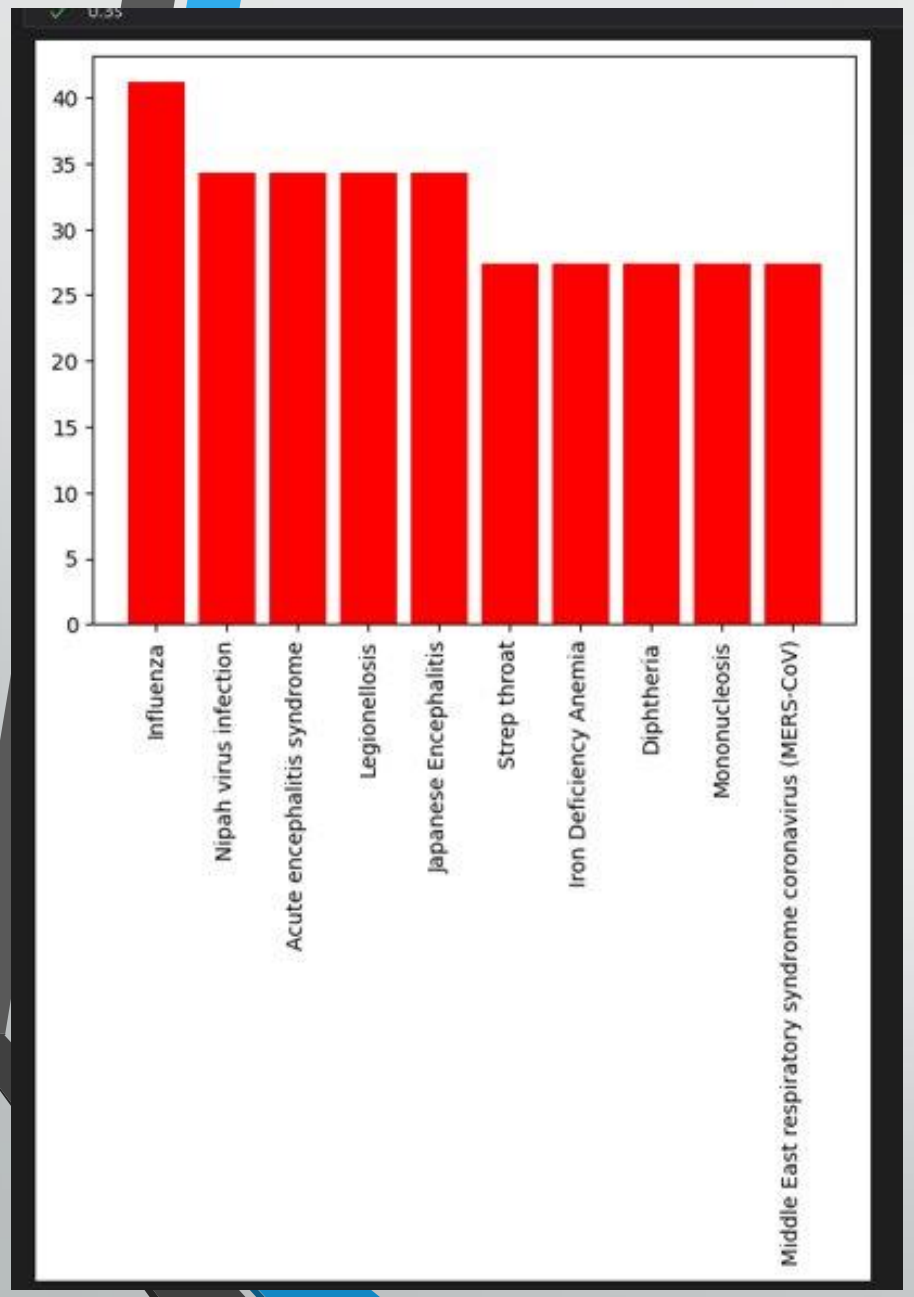

```
...
    Common co-occuring symptoms:
    0 : testicular pain
    1 : vomiting
    2 : barky cough
    3 : sore throat
    4 : confusion

    Common co-occuring symptoms:
    0 : diarrhea
    1 : maculopapular rash
    2 : feeling tired
    3 : swollen lymph node
    4 : shortness breath

    Common co-occuring symptoms:
    0 : chest pain
    1 : runny nose
    2 : unintended weight loss
    3 : muscle weakness
    4 : seizure

    Common co-occuring symptoms:
    0 : tiredness
    1 : dizziness
    2 : red eye
    3 : large lymph node
    4 : severe pain
```

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
```




```
✓ 0.0s

Top 10 diseases predicted based on symptoms
0 Disease name: Influenza       Probability: 41.16%
1 Disease name: Nipah virus infection   Probability: 34.3%
2 Disease name: Acute encephalitis syndrome     Probability: 34.3%
3 Disease name: Legionellosis   Probability: 34.3%
4 Disease name: Japanese Encephalitis   Probability: 34.3%
5 Disease name: Strep throat    Probability: 27.44%
6 Disease name: Iron Deficiency Anemia  Probability: 27.44%
7 Disease name: Diphtheria      Probability: 27.44%
8 Disease name: Mononucleosis   Probability: 27.44%
9 Disease name: Middle East respiratory syndrome coronavirus (MERS-CoV)         Probability: 27.44%
```

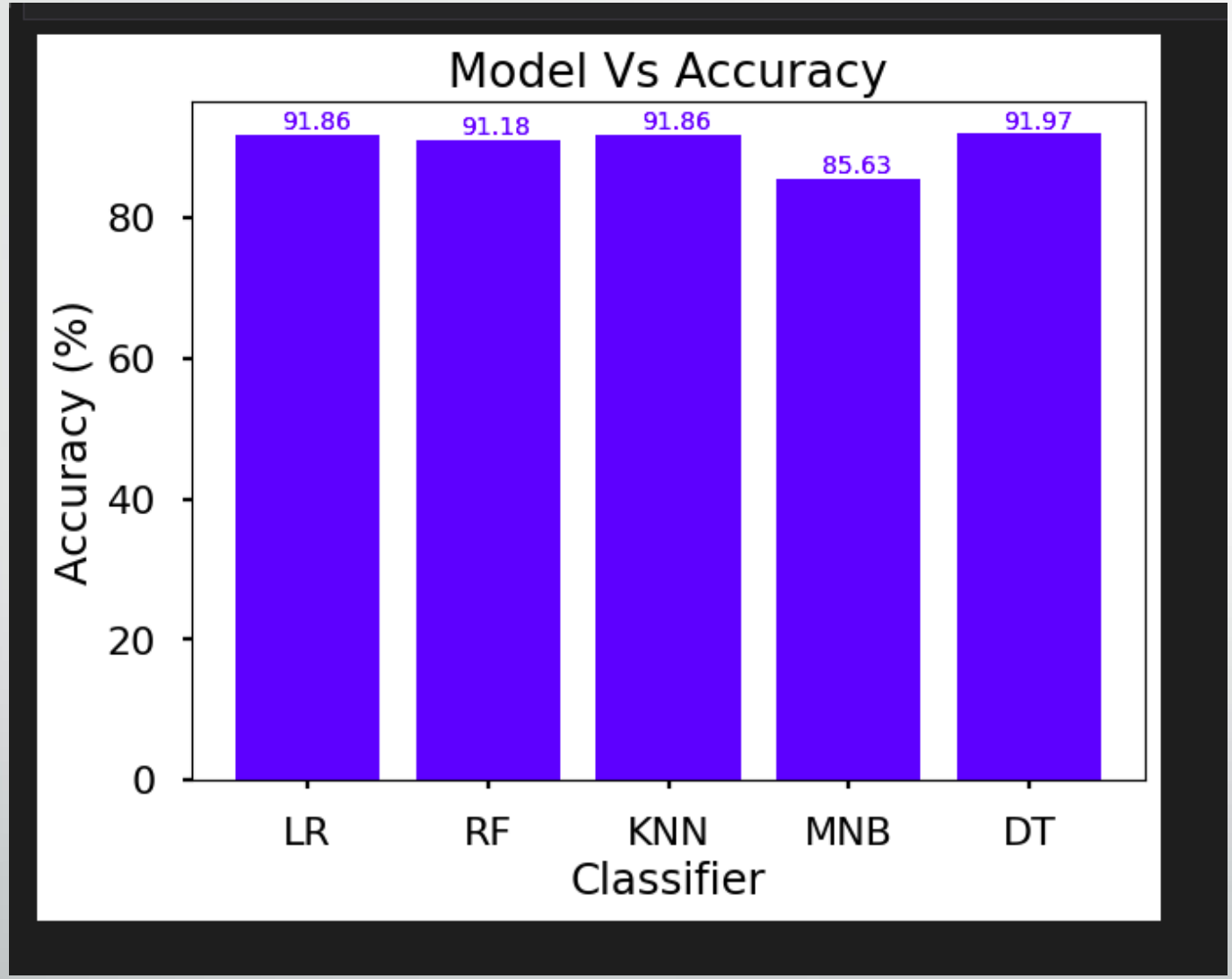
PREDICTION MODELS

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 10 diseases, sorted in the decreasing order of individual probabilities

Models experimented with till now:


Model Vs Accuracy
91.86
91.18
91.86
85.63
91.97
80
60
40
20
0
Accuracy (%)
LR
RF
KNN
MNB
DT
Classifier

Challenges Faced

- 1) **Web scraping** - diseases were scraped from www.nhp.gov.in/disease-a-z/, but scraping symptoms was a difficult task as the same site mentioned them in paragraphs.
- 2) **Symptom scraping** - Hence 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.
- 3) **User input processing and co-occurrence** – 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. Also, we had to mimic what doctors do by asking common co-occurring symptoms.

Future Scope

- Dataset could be increased to incur more diseases and symptoms, and also updated periodically.
- Treatment and precautions against the predicted diseases could be suggested to the user.
- If the user enters a symptom not present in dataset, the symptom could be included in dataset by fetching it in real-time from the internet.
- Deployment of the system on a public domain could help us to expand the scope of the project, with experts suggesting some inputs.
- Regional classification of diseases using real-time updating of dataset, and giving more weightage to certain regional diseases