**PSG COLLEGE OF TECHNOLOGY, COIMBATORE 641004**

**Department of Computer Science and Engineering**

# 19ZO02 Social And Economic Network Analysis

## Project Work

**By**

**Gunaal R - 19Z215**

**Jaswanth Krishna V - 19Z220**

**Balasubramanian S - 19Z237**

**Sri Raja Vignesh S - 19Z246**

**Vikneshwar A - 19Z259**

## Problem Statement:

To comprehend the relationship between various symptoms and diseases. Use the network created to anticipate the occurrence of diseases and establish the relationship between various symptom sequences.

## Dataset description:

For our analysis, we took the disease Symptom Prediction dataset from the kaggle website.This dataset consists of symptoms experienced by the body.

| Dataset Statistics | |
|---|---|
| Nodes | 183 |
| Edges | 339 |
| Types of Nodes | 2 - Diseases, Symptoms |
| Graph Type | Bipartite Graph |

## TOOLS USED:

In this section, we briefly discuss about the various tools and packages that we have used in order to accomplish our project

1. **NETWORKX**
    a. NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
    b. With NetworkX you can load and store networks in standard and nonstandard data formats, generate many types of random and classic networks, analyze network structure, build network models, design new network algorithms, draw networks, and much more.
    c. It provides tools for the study of the structure and dynamics of social, biological, and infrastructure networks.

2. **MATPLOTLIB.PYPLOT**
    a. Matplotlib.pyplot is a state-based interface to matplotlib. It provides an implicit, MATLAB-like, way of plotting. It also opens figures on your screen, and acts as the figure GUI manager.
    b. pyplot is mainly intended for interactive plots and simple cases of programmatic plot generation:
    c. The explicit (object-oriented) API is recommended for complex plots, though pyplot is still usually used to create the figure and often the axes in the figure.

3. **PANDAS**
    a. This python package is used for processing csv (Comma Separated Values) files. The dataset is mostly available as .csv files
    b. Pandas is a Python library used for working with data sets.It has functions for analyzing, cleaning, exploring, and manipulating data.
    c. Pandas allows us to analyze big data and make conclusions based on statistical theories.Pandas can clean messy data sets, and make them readable and relevant.

## Challenges Faced:

1. It was challenging to locate the appropriate dataset for this disease's symptom prediction.
2. Finding the common neighbors for a particular disease
3. Identifying the symptoms and predicting the disease

## Contribution of Team Members:

| **Roll No** | **Name** | **Contribution** |
|---|---|---|
| 19Z215 | Gunaal R | Report Documentation, Dataset Cleaning |
| 19Z220 | Jaswanth Krishna V | Dataset Collection, Cleaning Process, Representing as Bipartite Graph |

| | | |
|---|---|---|
| 19Z237 | Balasubramanian S | Analysis in the Disease-Symptoms graph produced |
| 19Z246 | Sri Raja Vignesh S | Dataset Collection, Cleaning Process, Representing as Bipartite Graph |
| 19Z259 | Vikneshwar A | Analysis in the Disease-Symptoms graph produced |

## ANNEXURE 1: CODE

```
import csv

import networkx as nx

import matplotlib.pyplot as pl

from networkx.algorithms import bipartite


disease = set()

symptoms = set()

with open('dataset.csv') as file:

   data = csv.reader(file)

   for line in data:

       if line[0] != '':

           disease.add(line[0])

       for i in line[1:]:

           if(i != ''):

               symptoms.add(i)


print(disease)


G = nx.path_graph(4)
```

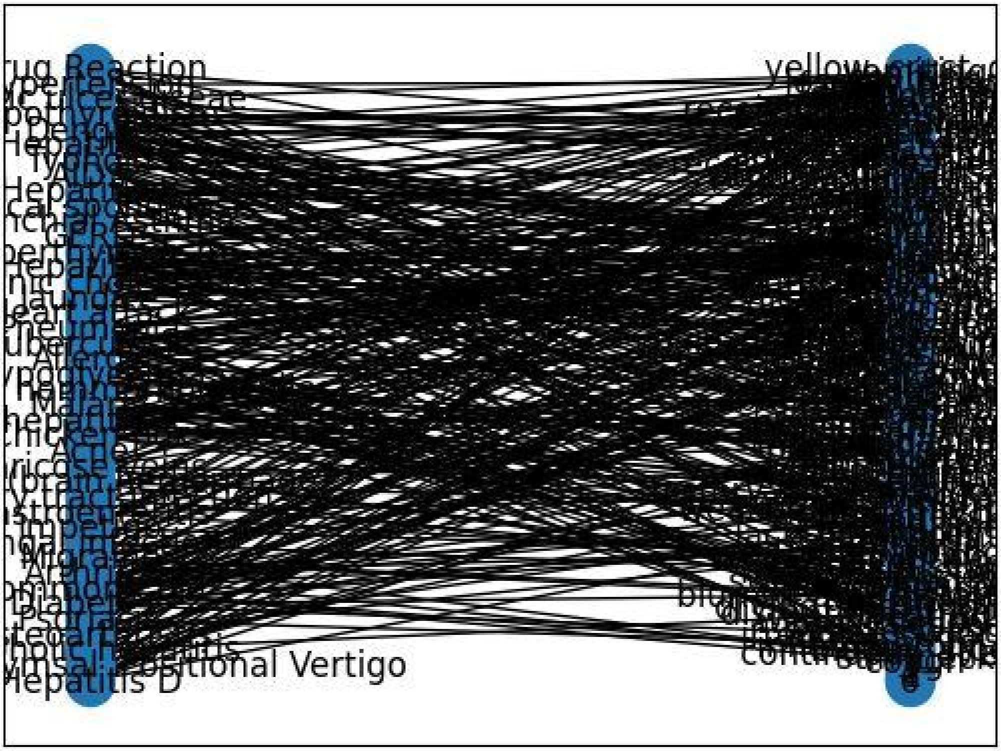
```
with open('dataset.csv') as file:
   data = csv.reader(file)
   for line in data:
       disease = line[0]
       for i in line[1:]:
           if i!= '':
               G.add_edge(disease,i)

nx.draw_networkx(G,pos = nx.drawing.bipartite_layout(G,disease),width =-1 )

disease_degree = {}
for i in G.nodes():
   if(type(i) == str):
       if(i in disease):
           disease_degree[i] = G.degree[i]



sorted_disease_degree = sorted(disease_degree.items(), key=lambda x:x[1])
converted_dict = dict(sorted_disease_degree)

print(converted_dict)



symp_degree = {}
for i in G.nodes():
   if(type(i) == str):
       if(i in symptoms):
           symp_degree[i] = G.degree[i]
```

```
sorted_symp_degree = sorted(symp_degree.items(), key=lambda x:x[1])
converted_dict = dict(sorted_symp_degree)

print(converted_dict)


# MOST COMMON SYMPTOMS
symptoms_degree = {}
for i in G.nodes():
   if(type(i) == str):
       if(i in symptoms):
           if((G.degree[i])>5):
               symptoms_degree[i] = G.degree[i]


sorted_symptoms_degree = sorted(symptoms_degree.items(), key=lambda
x:x[1],reverse=True)
converted_dict = dict(sorted_symptoms_degree)

common_symp = list(converted_dict.keys())
print(common_symp)


# MOST RARE SYMPTOMS
symptoms_degree = {}
for i in G.nodes():
   if(type(i) == str):
       if(i in symptoms):
           if((G.degree[i])<2):
               symptoms_degree[i] = G.degree[i]
```

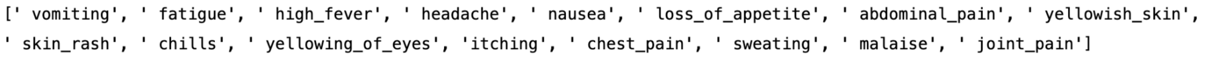
```
sorted_symptoms_degree = sorted(symptoms_degree.items(), key=lambda
x:x[1],reverse=True)
converted_dict = dict(sorted_symptoms_degree)

rare_symp = list(converted_dict.keys())
print(rare_symp)



# NEXT POSSIBLE DISEASE

from mimetypes import init

print("Find the next possible disease for: ")
text_inp = input()
projected_graph = bipartite.weighted_projected_graph(G, [text_inp])

next_possible_dict = {}
for i in projected_graph.edges(data=True):
   next_possible_dict[i[1]] = i[2]['weight']

next_possible_dict = sorted(
   next_possible_dict.items(), key=lambda x: x[1], reverse=True)
sorted_next_possible_dict = dict(next_possible_dict)

next_possible = {}
is_first = True
for s in sorted_next_possible_dict:
   if(is_first == True):
       init_disease = s
       init_disease_severity = sorted_next_possible_dict[s]
       is_first = False
       next_possible[init_disease] = init_disease_severity
   else:
       if sorted_next_possible_dict[s] > init_disease_severity/2:
           next_possible[s] = sorted_next_possible_dict[s]


print(next_possible)



# PREDICTION OF DISEASES GIVEN THE SYMPTOMS
print("No of Symptoms: ")
# no_of_symp = int(input())
find_symp = [' vomiting', ' chest_pain', ' breathlessness', ' sweating']
# [' high_fever',
#               ' blister',
#               ' skin_rash']

# for i in range(0, no_of_symp):
#     inpp = input()
#     find_symp.append(inpp)
temp_dis_list = []
for symp in find_symp:
   temp_dis_list += fetch_connected_nodes_for_symptoms(G, symp)
# print(temp_dis_list)
dis_value_dict = {x: temp_dis_list.count(x) for x in temp_dis_list}
# print(dis_value_dict)
dis_value_dict = sorted(dis_value_dict.items(),
                       key=lambda x: x[1], reverse=True)
sorted_dis_value_dict = dict(dis_value_dict)

predicted_disease = {}
is_first = True
for s in sorted_dis_value_dict:
   if(is_first == True):
       init_disease = s
       init_disease_possibility = sorted_dis_value_dict[s]
       is_first = False
       predicted_disease[init_disease] = init_disease_possibility
   else:
       if sorted_dis_value_dict[s] > init_disease_possibility/2:
           predicted_disease[s] = sorted_dis_value_dict[s]

print(predicted_disease)
```

# ANNEXURE 2: SNAPSHOTS

## 1. The Bipartite graph built

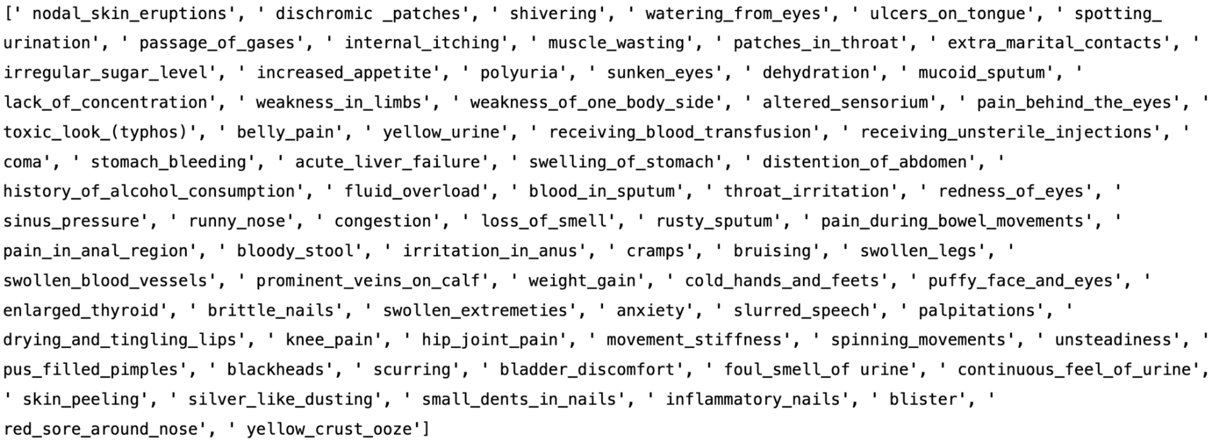


## 2. Most Common Symptoms



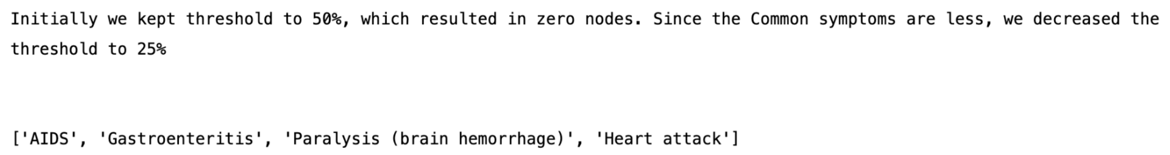
```
[' vomiting', ' fatigue', ' high_fever', ' headache', ' nausea', ' loss_of_appetite', ' abdominal_pain', ' yellowish_skin',
' skin_rash', ' chills', ' yellowing_of_eyes', 'itching', ' chest_pain', ' sweating', ' malaise', ' joint_pain']
```

## 3. Most Rare Symptoms



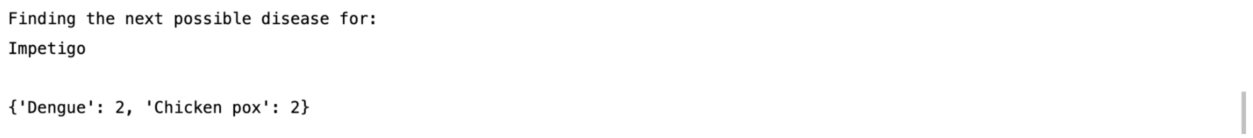
```
[' nodal_skin_eruptions', ' dischromic _patches', ' shivering', ' watering_from_eyes', ' ulcers_on_tongue', ' spotting_
urination', ' passage_of_gases', ' internal_itching', ' muscle_wasting', ' patches_in_throat', ' extra_marital_contacts', '
irregular_sugar_level', ' increased_appetite', ' polyuria', ' sunken_eyes', ' dehydration', ' mucoid_sputum', '
lack_of_concentration', ' weakness_in_limbs', ' weakness_of_one_body_side', ' altered_sensorium', ' pain_behind_the_eyes', '
toxic_look_(typhos)', ' belly_pain', ' yellow_urine', ' receiving_blood_transfusion', ' receiving_unsterile_injections', '
coma', ' stomach_bleeding', ' acute_liver_failure', ' swelling_of_stomach', ' distention_of_abdomen', '
history_of_alcohol_consumption', ' fluid_overload', ' blood_in_sputum', ' throat_irritation', ' redness_of_eyes', '
sinus_pressure', ' runny_nose', ' congestion', ' loss_of_smell', ' rusty_sputum', ' pain_during_bowel_movements', '
pain_in_anal_region', ' bloody_stool', ' irritation_in_anus', ' cramps', ' bruising', ' swollen_legs', '
swollen_blood_vessels', ' prominent_veins_on_calf', ' weight_gain', ' cold_hands_and_feets', ' puffy_face_and_eyes', '
enlarged_thyroid', ' brittle_nails', ' swollen_extremeties', ' anxiety', ' slurred_speech', ' palpitations', '
drying_and_tingling_lips', ' knee_pain', ' hip_joint_pain', ' movement_stiffness', ' spinning_movements', ' unsteadiness', '
pus_filled_pimples', ' blackheads', ' scurring', ' bladder_discomfort', ' foul_smell_of urine', ' continuous_feel_of_urine',
' skin_peeling', ' silver_like_dusting', ' small_dents_in_nails', ' inflammatory_nails', ' blister', '
red_sore_around_nose', ' yellow_crust_ooze']
```

## 4. Most Common Diseases

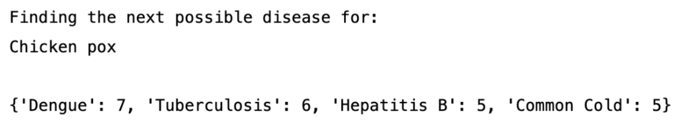


```
Initially we kept threshold to 50%, which resulted in zero nodes. Since the Common symptoms are less, we decreased the
threshold to 25%


['AIDS', 'Gastroenteritis', 'Paralysis (brain hemorrhage)', 'Heart attack']
```

## 5. Finding Next Possible Disease

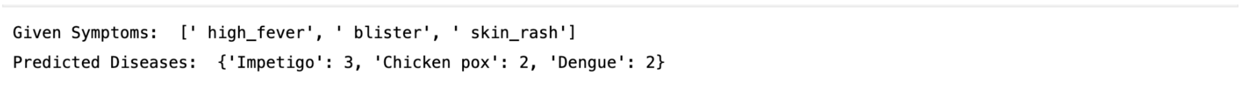


```
Finding the next possible disease for:
Impetigo

{'Dengue': 2, 'Chicken pox': 2}
```

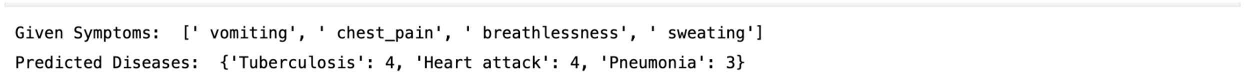


```
Finding the next possible disease for:
Chicken pox

{'Dengue': 7, 'Tuberculosis': 6, 'Hepatitis B': 5, 'Common Cold': 5}
```

## 6. Predicting the Diseases given the list of symptoms



```
Given Symptoms:  [' high_fever', ' blister', ' skin_rash']
Predicted Diseases:  {'Impetigo': 3, 'Chicken pox': 2, 'Dengue': 2}
```



```
Given Symptoms:  [' vomiting', ' chest_pain', ' breathlessness', ' sweating']
Predicted Diseases:  {'Tuberculosis': 4, 'Heart attack': 4, 'Pneumonia': 3}
```

## REFERENCES:

1. https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset?resource=download
2. https://www.kaggle.com/code/kunal2350/disease-prediction-with-gui
3. https://www.kaggle.com/datasets/rabisingh/symptom-checker?select=Training.csv
4. https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html
5. https://www.w3schools.com/python/pandas/pandas_intro.asp
6. https://www.geeksforgeeks.org/introduction-to-social-networks-using-networkx-in-python/
7. https://networkx.org/documentation/stable/tutorial.html
8. https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/
9. https://www.qualtrics.com/experience-management/research/cluster-analysis/
10. https://github.com/itachi9604/healthcare-chatbot