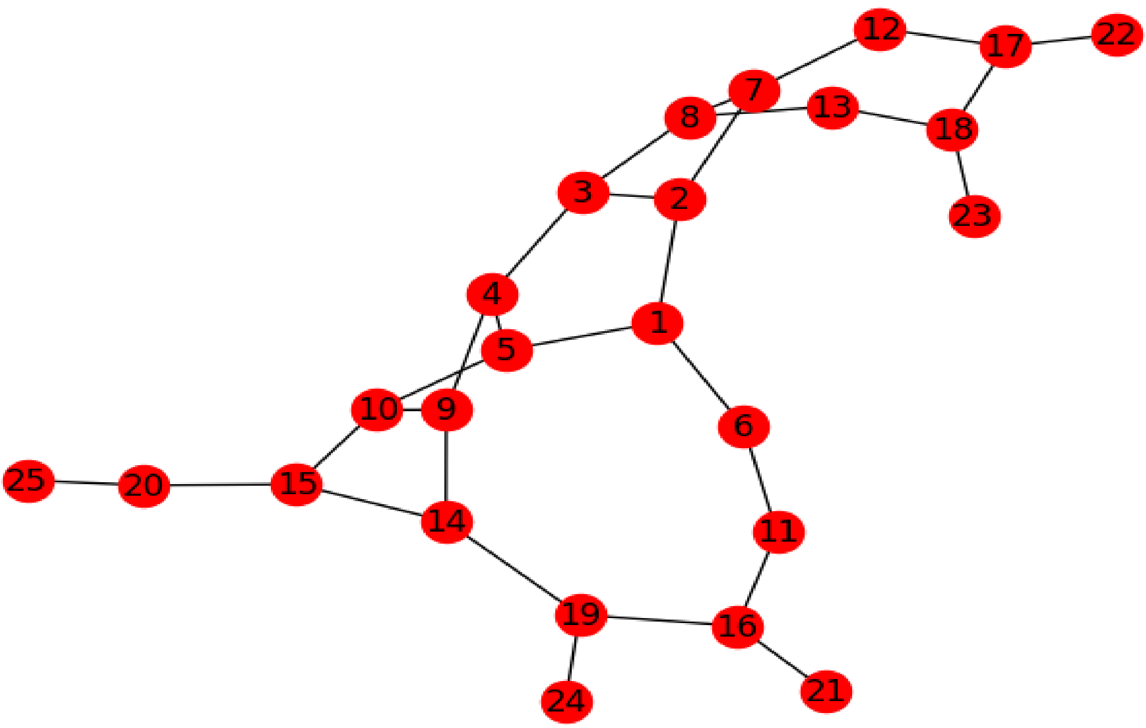
Original graph (without graph reduction)

I have drawn simple graph with 25 nodes and perform node classification without graph reduction. It perfectly classifies using random forest classifier.

Label values:

```
In [5]: node_labels = {
            1: 'Spam',
            2: 'Non-spam',
            3: 'Spam',
            4: 'Non-spam',
            5: 'Spam',
            6: 'Non-spam',
            7: 'Spam',
            8: 'Non-spam',
            9: 'Non-spam',
           10: 'Non-spam',
           11: 'Spam',
           12: 'Non-spam',
           13: 'Spam',
           14: 'Spam',
           15: 'Spam',
           16: 'Non-spam',
           17: 'Spam',
           18: 'Spam',
           19: 'Spam',
           20: 'Non-spam',
           21: 'Non-spam',
           22: 'Spam',
           23: 'Spam',
           24: 'Spam',
           25: 'Spam',
        }
```

Original graph



After train test split

Training nodes: [10, 17, 2, 23, 6, 3, 18, 15, 4, 5, 21, 19, 22, 13, 25, 8, 11, 14, 20, 7]
Testing nodes: [9, 16, 1, 24, 12]

In [20]:
```
ds = pd.DataFrame(data)
ds.head()
```

Out[20]:

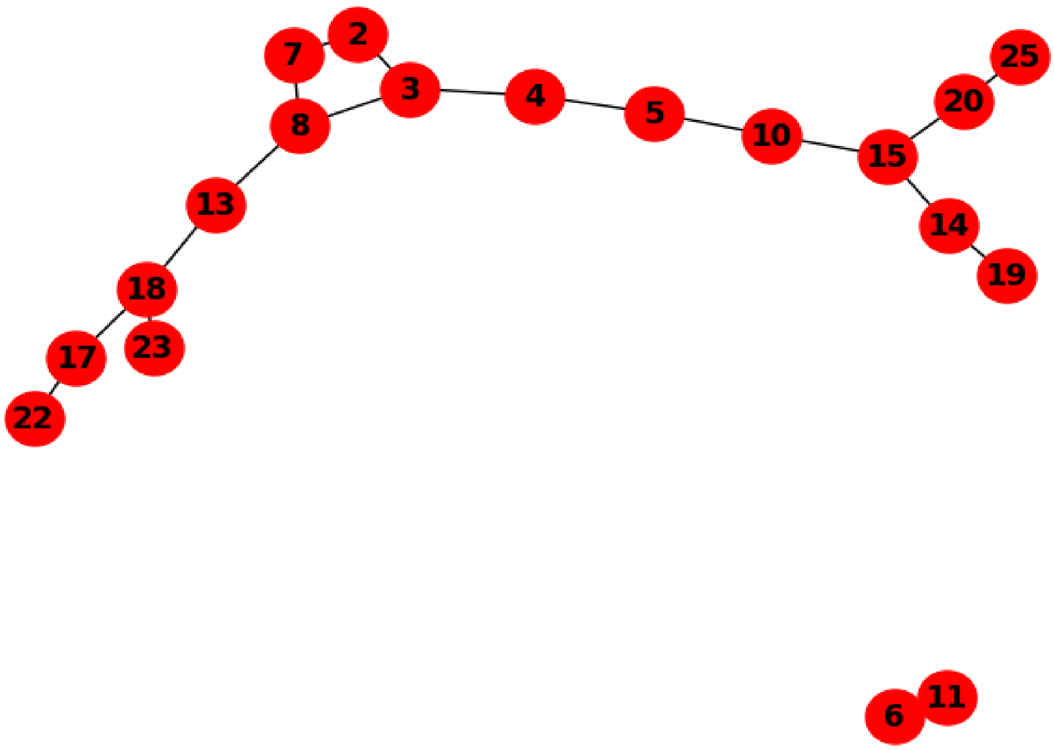
	Test Node	True Labels	Predicted Labels
0	9	Non-spam	Non-spam
1	16	Non-spam	Non-spam
2	1	Spam	Non-spam
3	24	Spam	Spam
4	12	Non-spam	Non-spam

Observations: One data is partially correctly classified. The accuracy is 80%.

Building new graph based on training nodes:

Training nodes : [10, 17, 2, 23, 6, 3, 18, 15, 4, 5, 21, 19, 22, 13, 25, 8, 11, 14, 20, 7]
Testing nodes: [9, 16, 1, 24, 12]

Using training data, the trained data graph is :

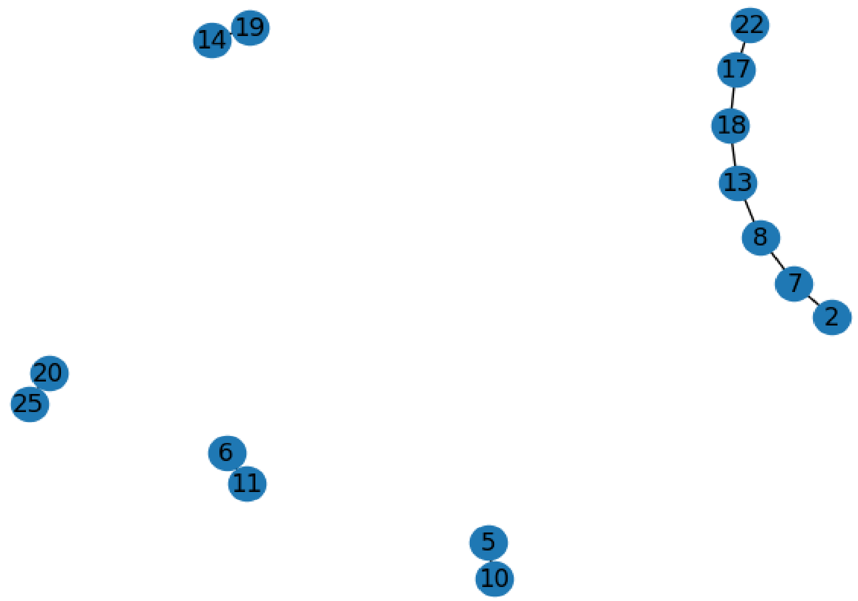


Applying Graph reduction techniques

1. Node sampling: (randomly remove nodes)

# Set the number of nodes to sample
num_nodes_to_sample = 15

Subgraph:




```
Training nodes = [2, 7, 8, 5, 10, 6, 11, 13, 18, 17, 22, 14, 19, 20, 25]

eliminated nodes = [3, 4, 15, 23]
```

Output predictions:

```
In [40]: ds1 = pd.DataFrame(data)
         ds1.head()
Out[40]:
```

	Test Node	True Labels	Predicted Labels
0	9	Non-spam	Spam
1	16	Non-spam	Non-spam
2	1	Spam	Non-spam
3	24	Spam	Spam
4	12	Non-spam	Non-spam

Observations: The accuracy is 60%

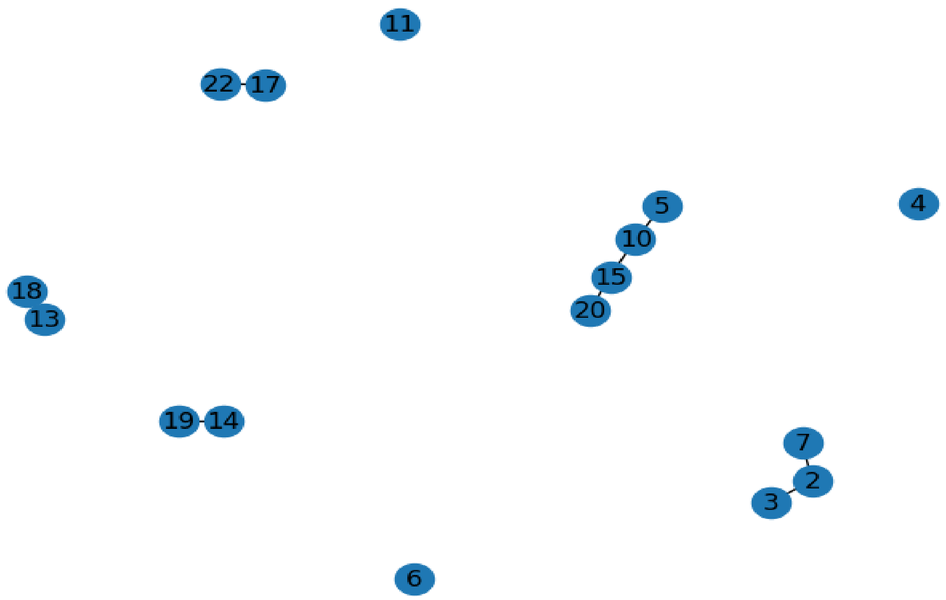
2. Edge sampling: (randomly remove edges)

# Set the number of nodes to sample
num_nodes_to_sample = 15

```
Training nodes = [2, 3, 7, 4, 5, 10, 6, 11, 13, 15, 18, 17, 22, 14, 19
, 20]

eliminated nodes = [8, 23, 25]
```

Subgraph:



Output predictions:

```
In [40]: ds1 = pd.DataFrame(data)
         ds1.head()
Out[40]:
```

	Test Node	True Labels	Predicted Labels
0	9	Non-spam	Spam
1	16	Non-spam	Non-spam
2	1	Spam	Non-spam
3	24	Spam	Spam
4	12	Non-spam	Non-spam

Observations: The accuracy is 60%.

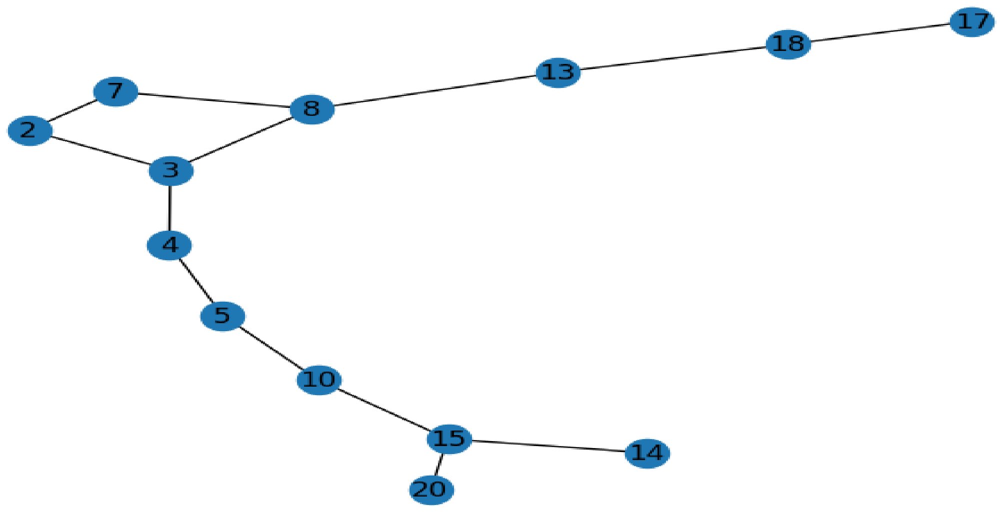
3. Graph Pruning: (remove less degree nodes or less connection nodes)

Set the minimum degree to retain a node : min_degree = 2
Remove nodes with degree less than min_degree

```
Training nodes = [2, 3, 7, 4, 8, 5, 10, 13, 15, 18, 17, 14, 20]

eliminated nodes = [6, 11, 23, 22, 19, 25]
```

Pruned Graph:



Output predictions:

```
In [42]: ds1 = pd.DataFrame(data)
         ds1.head()
```

Out[42]:

	Test Node	True Labels	Predicted Labels
0	9	Non-spam	Non-spam
1	16	Non-spam	Non-spam
2	1	Spam	Non-spam
3	24	Spam	Spam
4	12	Non-spam	Non-spam

Observations:

It is partially correctly classified based on less degree nodes. The accuracy is 80%

4. Graph Partitioning: (Use community detection method)

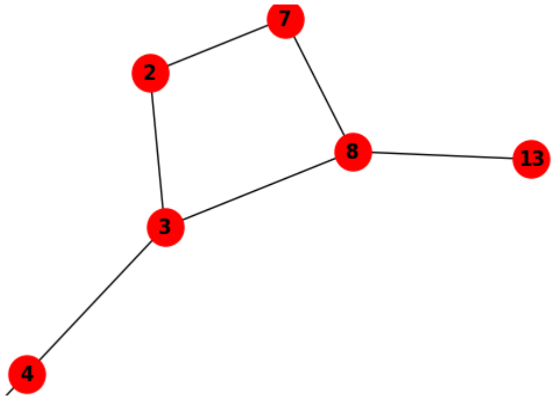
Basically, it removes high distance value. Only consider shortest distance and forms community or groups.

The cluster groups are :

```
[6, 11]
[18, 17, 23, 22]
[2, 3, 7, 4, 8, 5, 13]
[10, 15, 14, 19, 20, 25]
```

Partition graphs

Example of community detection for first iteration:



It goes through each and every graph cluster.

Output predictions:

```
In [34]: ds1 = pd.DataFrame(data)
         ds1.head()
```

Out[34]:

	Test Node	True Labels	Predicted Labels
0	9	Non-spam	Spam
1	16	Non-spam	Non-spam
2	1	Spam	Spam
3	24	Spam	Spam
4	12	Non-spam	Non-spam

Observations:

It is partially correctly classified based on cluster (community detection). The accuracy is 80%

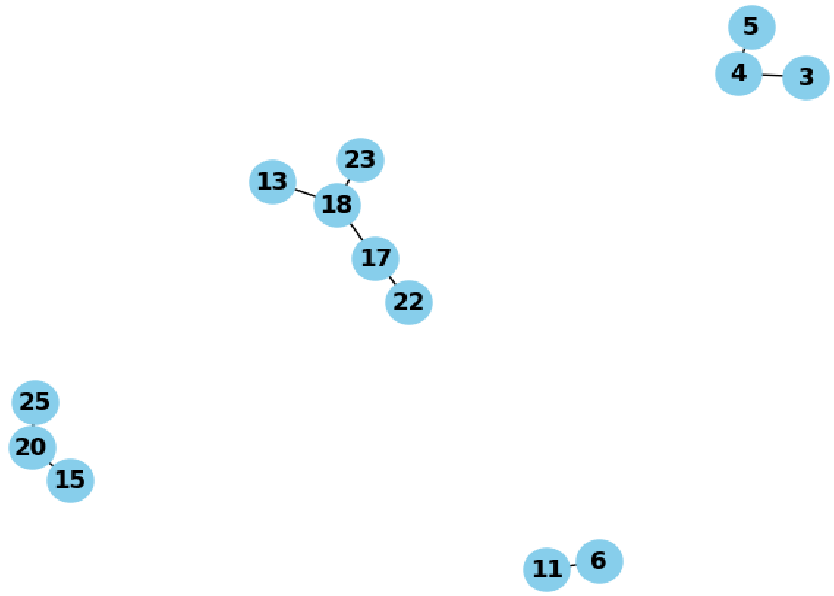
5. Neighbourhood Sampling: (selecting a random subset of nodes and their corresponding edges from a graph, focusing on a specific neighborhood)

# Set the number of nodes to sample
num_nodes_to_sample = 6

```
Training nodes = [3, 4, 5, 6, 11, 13, 15, 17, 18, 20, 22, 23, 25]

eliminated nodes = [2, 7, 8, 10, 14, 19]
```

Subgraph:



Output predictions:

```
In [43]: ds1 = pd.DataFrame(data)
         ds1.head()
```

Out[43]:

	Test Node	True Labels	Predicted Labels
0	9	Non-spam	Spam
1	16	Non-spam	Spam
2	1	Spam	Non-spam
3	24	Spam	Spam
4	12	Non-spam	Spam

Observations:

It is incorrectly classified based on neighbourhood connection from subgraph nodes. The accuracy is 20%.