Link Prediction

```
pip install node2vec
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting node2vec
  Downloading node2vec-0.4.3.tar.gz (4.6 kB)
Requirement already satisfied: networkx in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages (from n
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from no
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from nod
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.7/dist-packages (fr
Building wheels for collected packages: node2vec
  Building wheel for node2vec (setup.py) ... done
  Created wheel for node2vec: filename=node2vec-0.4.3-py3-none-any.whl size=5980 sha256
  Stored in directory: /root/.cache/pip/wheels/07/62/78/5202cb8c03cbf1593b48a8a442fca8c
Successfully built node2vec
Installing collected packages: node2vec
Successfully installed node2vec-0.4.3
```

```
pip install scikit-network
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting scikit-network
  Downloading scikit-network-0.27.0.tar.gz (1.8 MB)
     |████████████████████████████████| 1.8 MB 16.1 MB/s
  Downloading scikit_network-0.26.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_
     |████████████████████████████████| 8.1 MB 17.8 MB/s
Requirement already satisfied: numpy>=1.21.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scipy>=1.6.3 in /usr/local/lib/python3.7/dist-packages (
Installing collected packages: scikit-network
Successfully installed scikit-network-0.26.0
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sknetwork
import networkx as nx
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from tqdm import tqdm
import random
from node2vec import Node2Vec
```

```
# Load nodes details
with open('fb-pages-food.nodes') as f:
  fb_nodes = f.read().splitlines()

# Load edges details
with open('fb-pages-food.edges') as f:
  fb_links = f.read().splitlines()


print(len(fb_nodes),len(fb_links))
```

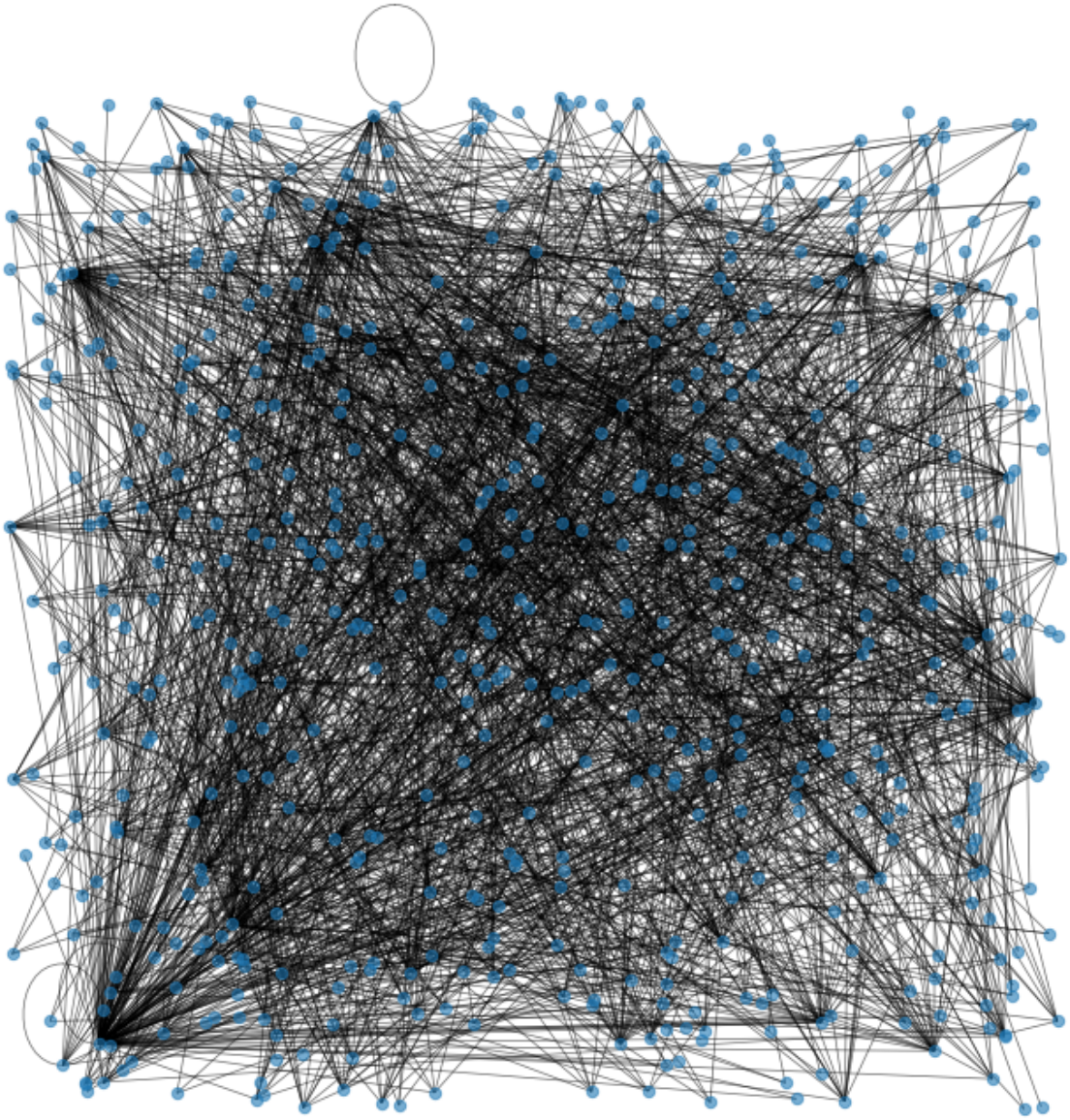
```
621 2102
```

```
# Take nodes in two seperate lists
node_list1 = []
node_list2 = []

for i in tqdm(fb_links):
  node_list1.append(i.split(',')[0])
  node_list2.append(i.split(',')[1])


fb_df = pd.DataFrame({'node_1':node_list1,
                      'node_2':node_list2})
fb_df.head()
```

```
100%|██████████| 2102/2102 [00:00<00:00, 1011522.14it/s]
```

	node_1	node_2
0	0	276
1	0	58
2	0	132
3	0	603
4	0	398

```
# Create Graph
G = nx.from_pandas_edgelist(fb_df,'node_1','node_2',create_using=nx.Graph())

# plot Graph
plt.figure(figsize=(10,10))
pos = nx.random_layout(G)
nx.draw(G,pos,with_labels=False,node_size=40,width=0.7,alpha = 0.6)
plt.show()
```



```
# combine all the nodes in the list
node_list = node_list1 + node_list2

node_list = list(dict.fromkeys(node_list))

# Build Adjacency matrix
adj_G = nx.to_numpy_matrix(G,node_list)
adj_G.shape
```

```
(620, 620)
```

```
# get unconnected node-pairs
```

```
all_unconnected_pairs = []

# traverse adjacency matrix
offset=0
for i in tqdm(range(adj_G.shape[0])):
  for j in range(offset,adj_G.shape[1]):
    if i!=j :
      if nx.shortest_path_length(G,str(i),str(j)) <=2:
        if adj_G[i,j] == 0:
          all_unconnected_pairs.append([node_list[i],node_list[j]])
  offset+=1
len(all_unconnected_pairs)
```

```
100%|██████████| 620/620 [00:09<00:00, 65.88it/s]
19018
```

```
node_1_unlinked = [i[0] for i in all_unconnected_pairs]
node_2_unlinked = [i[1] for i in all_unconnected_pairs]
data = pd.DataFrame({'node_1': node_1_unlinked,
                     'node_2': node_2_unlinked})

# add target variable 'link'
data['link']=0
data
```

	node_1	node_2	link
0	0	22	0
1	0	526	0
2	0	36	0
3	0	54	0
4	0	56	0
...	...	...	...
19013	606	586	0
19014	606	541	0
19015	592	573	0
19016	592	541	0
19017	573	586	0

19018 rows × 3 columns

```
initial_node_count = len(G.nodes)

fb_df_temp = fb_df.copy()
```

```
4
5 # empty list to store removable links
6 omissible_links_index = []
7
8 for i in tqdm(fb_df.index.values):
9
10   # remove a node pair and create a new graph
11   G_temp = nx.from_pandas_edgelist(fb_df_temp.drop(index=i),'node_1','node_2',create_using
12
13   # check there is no splitting of graph and number of nodes is same
14   if(nx.number_connected_components(G_temp)==1) and (len(G_temp.nodes)==initial_node_count
15     omissible_links_index.append(i)
16     fb_df_temp = fb_df_temp.drop(index=i)
17
18 len(omissible_links_index)
```

```
100%|██████████| 2102/2102 [00:08<00:00, 236.94it/s]
1483
```

```
1 # create dataframe of removable edges
2 fb_df_ghost = fb_df.loc[omissible_links_index]
3
4 # Add the target variable link
5 fb_df_ghost['link']=1
6
7 data = data.append(fb_df_ghost[['node_1','node_2','link']],ignore_index=True)
8 data['link'].value_counts()
```

```
0    19018
1     2966
Name: link, dtype: int64
```

```
1 # drop removable edges
2 fb_df_partial = fb_df.drop(index=fb_df_ghost.index.values)
3
4 # build graph
5 G_data = nx.from_pandas_edgelist(fb_df_partial,"node_1","node_2",create_using=nx.Graph())
```

```
1 # Generate Walks
2 node2vec = Node2Vec(G_data,walk_length = 16,num_walks=50,dimensions=100)
3 n2w_model = node2vec.fit(window=7,min_count=1)
```

```
Computing transition probabilities:                    620/620 [00:00<00:00,
100%                                                   9.44it/s]
```

```
1 x = [(n2w_model[str(i)])+(n2w_model[str(j)]) for i,j in zip(data['node_1'],data['node_2'])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Cal
  """Entry point for launching an IPython kernel.
```

```
1 xtrain,xtest,ytrain,ytest = train_test_split(np.array(x),data['link'],test_size=0.2,random
```

```
1 lr = LogisticRegression(class_weight="balanced")
```

```
1 lr.fit(xtrain,ytrain)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Convergen
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression(class_weight='balanced')
```

```
1 predictions = lr.predict_proba(xtest)
```

```
1 roc_auc_score(ytest,predictions[:,1])
```

```
0.8268357861979924
```