225229147

Lab 7. Link Prediction of future connections in Facebook

Step-1:

In [1]:

```
import pandas as pd
import numpy as np
import random
import networkx as nx
from tqdm import tqdm
import re
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

Step-2:

In [2]:

```
# load nodes details
with open("fb-pages-food.nodes", "r+", encoding="utf-8") as f:
    fb_nodes = f.read().splitlines()

# load edges (or links)
with open("fb-pages-food.edges", "r", encoding="utf-8") as f:
    fb_links = f.read().splitlines()

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

Out[2]:

```
(621, 2102)
```

In [3]:

```
# captture nodes in 2 separate lists
node_list_1 = []
node_list_2 = []

for i in tqdm(fb_links):
    node_list_1.append(i.split(',')[0])
    node_list_2.append(i.split(',')[1])

fb_df = pd.DataFrame({'node_1': node_list_1, 'node_2': node_list_2})
```

```
100%|██████████████████████████████████████████████████████████████████████
██| 2102/2102 [00:00<00:00, 262245.37it/s]
```

In [4]:

```
fb_df.head()
```

Out[4]:

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

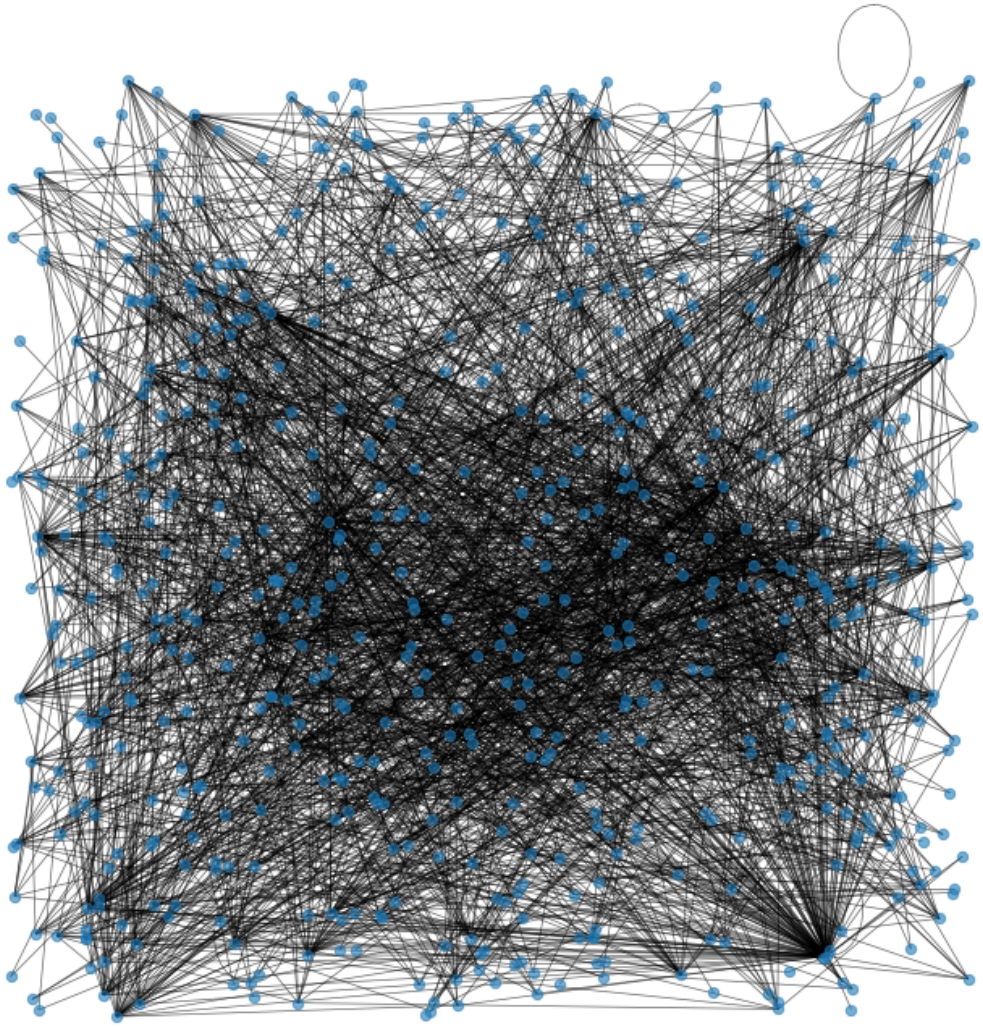
In [5]:

```
# 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, seed=23)
nx.draw(G, with_labels=False,  pos = pos, node_size = 40, alpha = 0.6, width = 0.7)

plt.show()
```



Step-3:

In [6]:

```
# combine all nodes in a list
node_list = node_list_1 + node_list_2

# remove duplicate items from the list
node_list = list(dict.fromkeys(node_list))

# build adjacency matrix
adj_G = nx.to_numpy_matrix(G, nodelist = node_list)
```

In [7]:

```
adj_G.shape
```

Out[7]:

```
(620, 620)
```

In [8]:

```
# 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 = offset + 1
```

```
100%|██████████████████████████████████████████████████████████████████████████
████████| 620/620 [00:32<00:00, 18.96it/s]
```

In [9]:

```
len(all_unconnected_pairs)
```

Out[9]:

```
19018
```

In [10]:

```
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
```

In [11]:

```
initial_node_count = len(G.nodes)

fb_df_temp = fb_df.copy()

# empty list to store removable links
omissible_links_index = []

for i in tqdm(fb_df.index.values):

  # remove a node pair and build a new graph
    G_temp = nx.from_pandas_edgelist(fb_df_temp.drop(index = i), "node_1", "node_2", create

  # check there is no spliting of graph and number of nodes is same
    if (nx.number_connected_components(G_temp) == 1) and (len(G_temp.nodes) == initial_node
        omissible_links_index.append(i)
        fb_df_temp = fb_df_temp.drop(index = i)
```

```
100%|██████████████████████████████████████████████████████████████████████
██████| 2102/2102 [00:26<00:00, 79.05it/s]
```

In [12]:

```
len(omissible_links_index)
```

Out[12]:

```
1483
```

In [13]:

```
# create dataframe of removable edges
fb_df_ghost = fb_df.loc[omissible_links_index]

# add the target variable 'link'
fb_df_ghost['link'] = 1

data = data.append(fb_df_ghost[['node_1', 'node_2', 'link']], ignore_index=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_8344\3242015871.py:7: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a fu
ture version. Use pandas.concat instead.
  data = data.append(fb_df_ghost[['node_1', 'node_2', 'link']], ignore_index=
True)
```

In [14]:

```
data['link'].value_counts()
```

Out[14]:

```
0    19018
1     1483
Name: link, dtype: int64
```

Step-4:

In [15]:

```
# drop removable edges
fb_df_partial = fb_df.drop(index=fb_df_ghost.index.values)

# build graph
G_data = nx.from_pandas_edgelist(fb_df_partial, "node_1", "node_2", create_using=nx.Graph()
```

Step-5:

In [16]:

```
from node2vec import Node2Vec

# Generate walks
node2vec = Node2Vec(G_data, dimensions=100, walk_length=16, num_walks=50)

# train node2vec model
n2w_model = node2vec.fit(window=7, min_count=1)
```

Computing transition probabilities: 100% 620/620 [00:00<00:00, 2056.15it/s]

Generating walks (CPU: 1): 100%|██████████| 50/50 [00:06<00:00, 7.27it/s]

In [17]:

```
x = [(n2w_model.wv[str(i)]+n2w_model.wv[str(j)]) for i,j in zip(data['node_1'], data['node_
```

In [18]:

```
xtrain, xtest, ytrain, ytest = train_test_split(np.array(x), data['link'],
                                                test_size = 0.3,
                                                random_state = 35)
```

In [19]:

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

lr.fit(xtrain, ytrain)
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:81
4: ConvergenceWarning: lbfgs failed to converge (status=1):
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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[19]:

```
LogisticRegression(class_weight='balanced')
```

In [20]:

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

In [21]:

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

Out[21]:

```
0.793512910754333
```

In [22]:

```
!pip install lightgbm==3.2.1 --user
```

```
Requirement already satisfied: lightgbm==3.2.1 in c:\users\hp\anaconda3\lib\s
ite-packages (3.2.1)
Requirement already satisfied: wheel in c:\users\hp\anaconda3\lib\site-packag
es (from lightgbm==3.2.1) (0.37.1)
Requirement already satisfied: numpy in c:\users\hp\appdata\roaming\python\py
thon39\site-packages (from lightgbm==3.2.1) (1.24.4)
Requirement already satisfied: scipy in c:\users\hp\appdata\roaming\python\py
thon39\site-packages (from lightgbm==3.2.1) (1.11.1)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\users\hp\anaconda3
\lib\site-packages (from lightgbm==3.2.1) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hp\anaconda3
\lib\site-packages (from scikit-learn!=0.22.0->lightgbm==3.2.1) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\hp\anaconda3\lib\site
-packages (from scikit-learn!=0.22.0->lightgbm==3.2.1) (1.1.0)
```

In [23]:

```
import lightgbm as lgbm

train_data = lgbm.Dataset(xtrain, ytrain)
test_data = lgbm.Dataset(xtest, ytest)

# define parameters
parameters = {
    'objective': 'binary',
    'metric': 'auc',
    'is_unbalance': True,
    'boosting_type': 'gbdt',
    'feature_fraction': 0.5,
    'bagging_fraction': 0.5,
    'bagging_freq': 20,
    'num_threads' : 2,
    'seed' : 76
}

# train lightGBM model
model = lgbm.train(parameters,
                   train_data,
                   valid_sets=test_data,
                   num_boost_round=1000,
                   early_stopping_rounds=20)
```

```
[6]     valid_0's auc: 0.824651
[7]     valid_0's auc: 0.837053
[8]     valid_0's auc: 0.840197
[9]     valid_0's auc: 0.843371
[10]    valid_0's auc: 0.847606
[11]    valid_0's auc: 0.850713
[12]    valid_0's auc: 0.854906
[13]    valid_0's auc: 0.857276
[14]    valid_0's auc: 0.862201
[15]    valid_0's auc: 0.865094
[16]    valid_0's auc: 0.867512
[17]    valid_0's auc: 0.870469
[18]    valid_0's auc: 0.872359
[19]    valid_0's auc: 0.875084
[20]    valid_0's auc: 0.878144
[21]    valid_0's auc: 0.880514
[22]    valid_0's auc: 0.882833
[23]    valid_0's auc: 0.885202
[24]    valid_0's auc: 0.887437
[25]    valid_0's auc: 0.889702
```