Social network Graph Link Prediction - Facebook Challenge

Problem statement:

Given a directed social graph, have to predict missing links to recommend users (Link Prediction in graph)

Data Overview

Taken data from facebook's recruting challenge on kaggle
https://www.kaggle.com/c/FacebookRecruiting
data contains two columns source and destination eac edge in graph

```
- Data columns (total 2 columns):
- source_node         int64
- destination_node    int64
```

Mapping the problem into supervised learning problem:

- Generated training samples of good and bad links from given directed graph and for each link got some features like no of followers, is he followed back, page rank, katz score, adar index, some svd fetures of adj matrix, some weight features etc. and trained ml model based on these features to predict link.
- Some reference papers and videos :
  - https://www.cs.cornell.edu/home/kleinber/link-pred.pdf
  - https://www3.nd.edu/~dial/publications/lichtenwalter2010new.pdf

- https://kaggle2.blob.core.windows.net/forum-message-attachments/2594/supervised_link_prediction.pdf
- https://www.youtube.com/watch?v=2M77Hgy17cg

Business objectives and constraints:

- No low-latency requirement.
- Probability of prediction is useful to recommend ighest probability links

Performance metric for supervised learning:

- Both precision and recall is important so F1 score is good choice
- Confusion matrix

In [1]:
```
#Importing Libraries
# please do go through this python notebook:
import warnings
warnings.filterwarnings("ignore")

import csv
import pandas as pd#pandas to create small dataframes
import datetime #Convert to unix time
import time #Convert to unix time
# if numpy is not installed already : pip3 install numpy
import numpy as np#Do aritmetic operations on arrays
# matplotlib: used to plot graphs
import matplotlib
import matplotlib.pylab as plt
import seaborn as sns#Plots
from matplotlib import rcParams#Size of plots
from sklearn.cluster import MiniBatchKMeans, KMeans#Clustering
import math
import pickle
import os
```

```
# to install xgboost: pip3 install xgboost
import xgboost as xgb

import warnings
import networkx as nx
import pdb
import pickle
```

In [9]:
```
#reading graph
if not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_woheader.csv'):
    traincsv = pd.read_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train.csv')
    traincsv = traincsv[0:100000]
    print(traincsv[traincsv.isna().any(1)])
    print(traincsv.info())
    print("Number of diplicate entries: ",sum(traincsv.duplicated()))
    traincsv.to_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_woheader.csv',header=False,index=False)
    print("saved the graph into file")
else:
    g=nx.read_edgelist('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_woheader.csv',delimiter=',',create_using=nx.DiGraph(),nodetype=int)
    print(nx.info(g))
```

```
Name:
Type: DiGraph
Number of nodes: 105845
Number of edges: 100000
Average in degree:   0.9448
Average out degree:   0.9448
```

In [10]:
```
abc = pd.read_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecrui
```

```
ting\\train_woheader.csv')
```

In [11]:
```
abc.shape
```

Out[11]:
```
(99999, 2)
```

> Displaying a sub graph

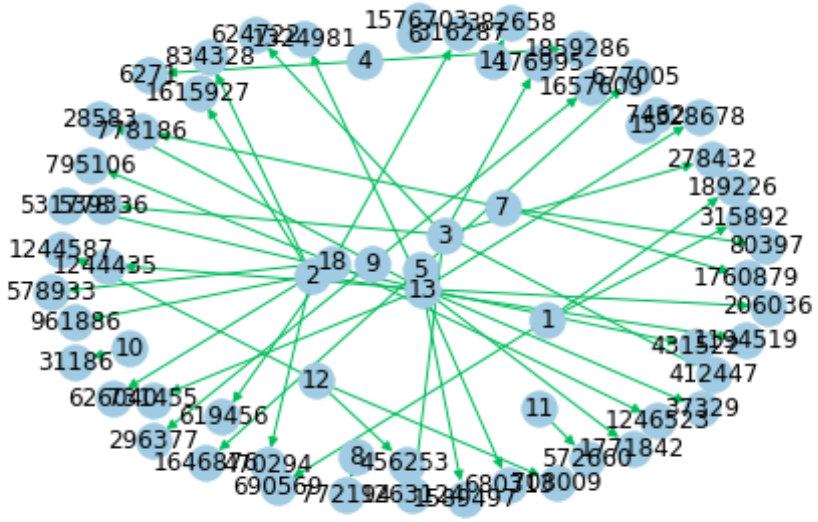
In [12]:
```
if not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_woheader_sample.csv'):
    pd.read_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train.csv', nrows=50).to_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_woheader_sample.csv',header=False,index=False)

subgraph=nx.read_edgelist('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_woheader_sample.csv',delimiter=',',create_using=nx.DiGraph(),nodetype=int)
# https://stackoverflow.com/questions/9402255/drawing-a-huge-graph-with-networkx-and-matplotlib

pos=nx.spring_layout(subgraph)
nx.draw(subgraph,pos,node_color='#A0CBE2',edge_color='#00bb5e',width=1,edge_cmap=plt.cm.Blues,with_labels=True)
plt.savefig("G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\graph_sample.pdf")
print(nx.info(subgraph))
```

```
Name:
Type: DiGraph
Number of nodes: 66
```

```
Number of edges: 50
Average in degree:   0.7576
Average out degree:   0.7576
```



In [13]:
```
xyz = pd.read_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_woheader_sample.csv')
```

In [14]:
```
xyz.shape
```

Out[14]:
```
(49, 2)
```

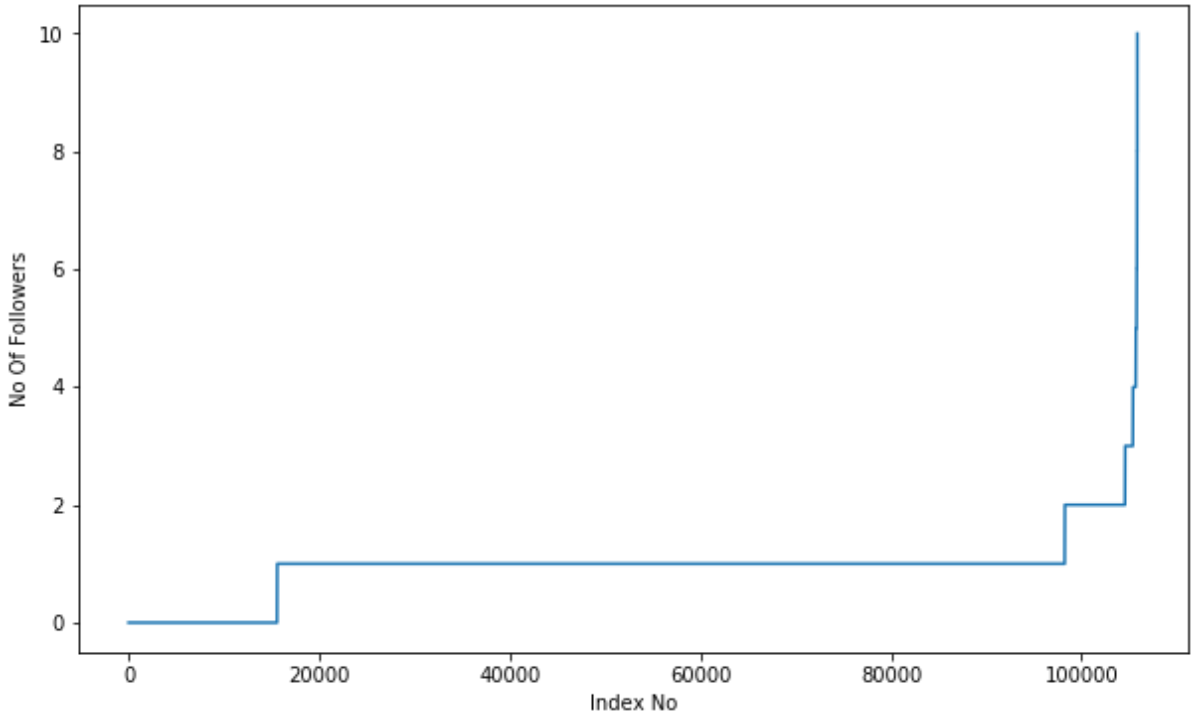
1. Exploratory Data Analysis

In [15]:
```
# No of Unique persons
print("The number of unique persons",len(g.nodes()))
```

```
The number of unique persons 105845
```

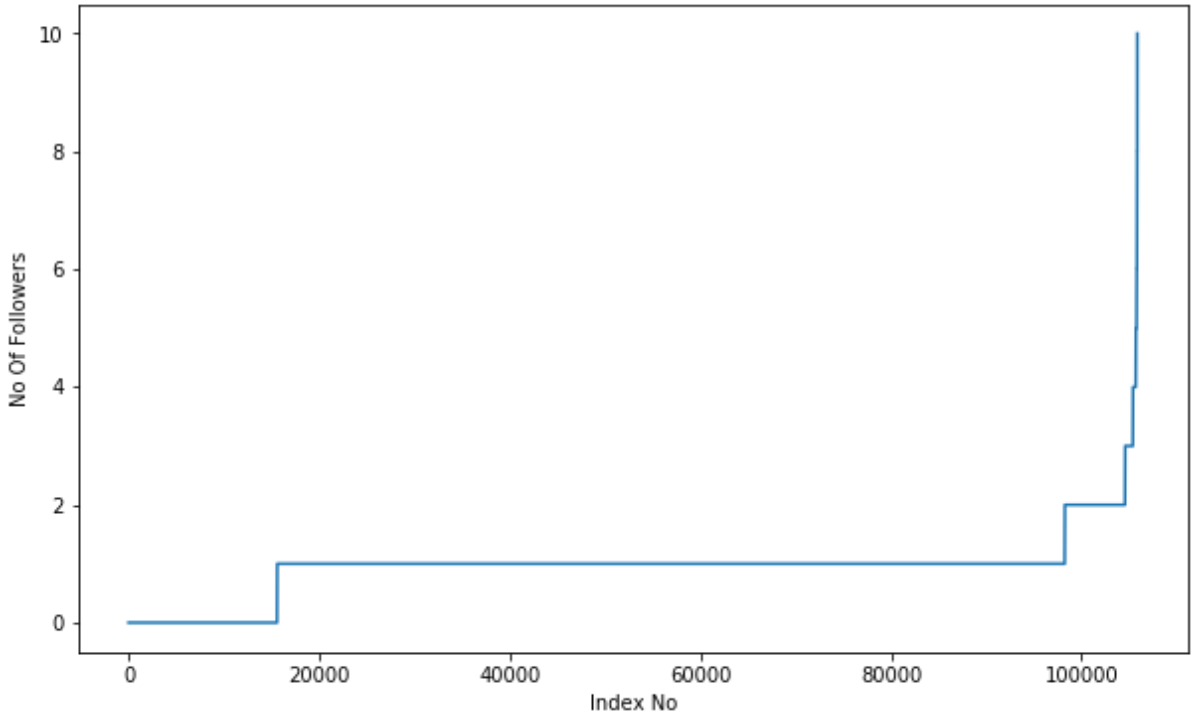
1.1 No of followers for each person

In [16]:
```
indegree_dist = list(dict(g.in_degree()).values())
indegree_dist.sort()
plt.figure(figsize=(10,6))
plt.plot(indegree_dist)
plt.xlabel('Index No')
plt.ylabel('No Of Followers')
plt.show()
```



In [17]:
```
indegree_dist = list(dict(g.in_degree()).values())
indegree_dist.sort()
```

```
plt.figure(figsize=(10,6))
plt.plot(indegree_dist[0:1000000])
plt.xlabel('Index No')
plt.ylabel('No Of Followers')
plt.show()
```



In [18]:
```
plt.boxplot(indegree_dist)
plt.ylabel('No Of Followers')
plt.show()
```

In [19]:
```
### 90-100 percentile
for i in range(0,11):
    print(90+i,'percentile value is',np.percentile(indegree_dist,90+i))
```

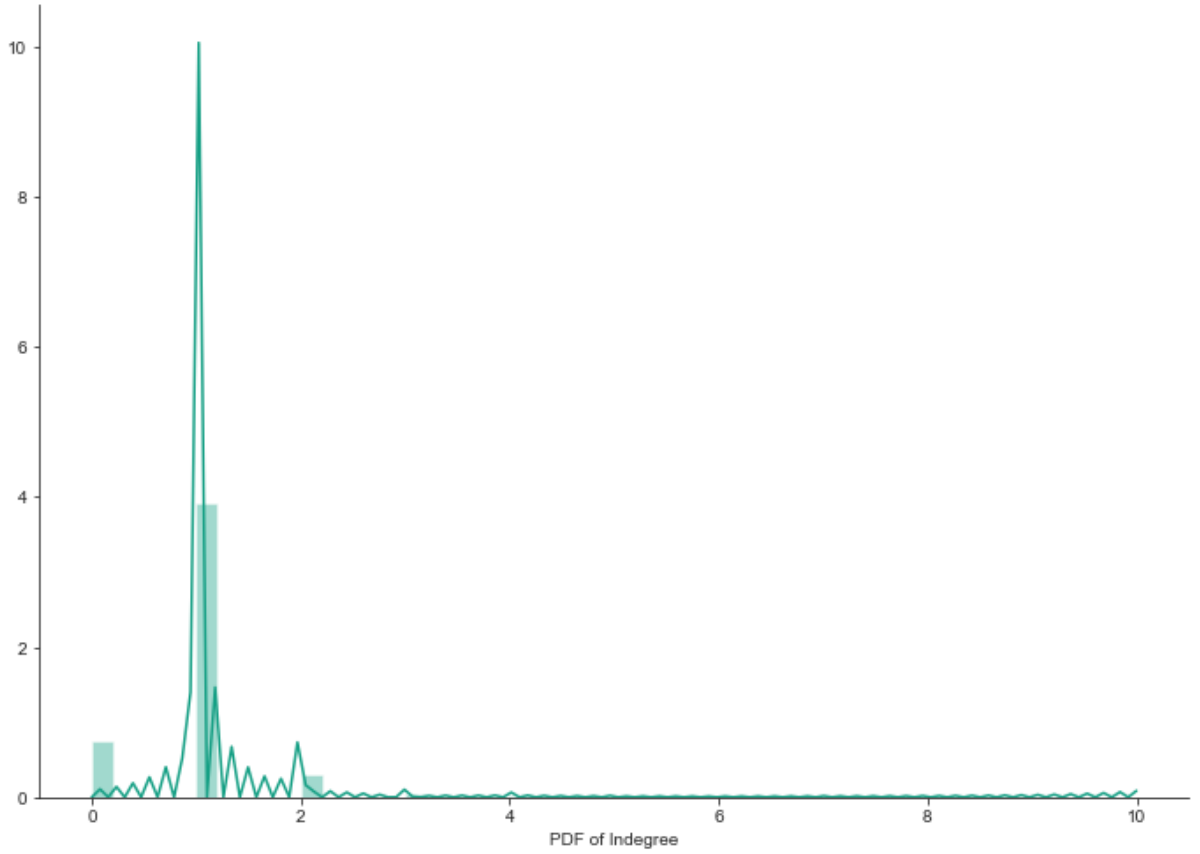
```
90 percentile value is 1.0
91 percentile value is 1.0
92 percentile value is 1.0
93 percentile value is 2.0
94 percentile value is 2.0
95 percentile value is 2.0
96 percentile value is 2.0
97 percentile value is 2.0
98 percentile value is 2.0
99 percentile value is 3.0
100 percentile value is 10.0
```

99% of data having followers of 40 only.

In [20]:
```
### 99-100 percentile
for i in range(10,110,10):
    print(99+(i/100),'percentile value is',np.percentile(indegree_dist,
99+(i/100)))
```

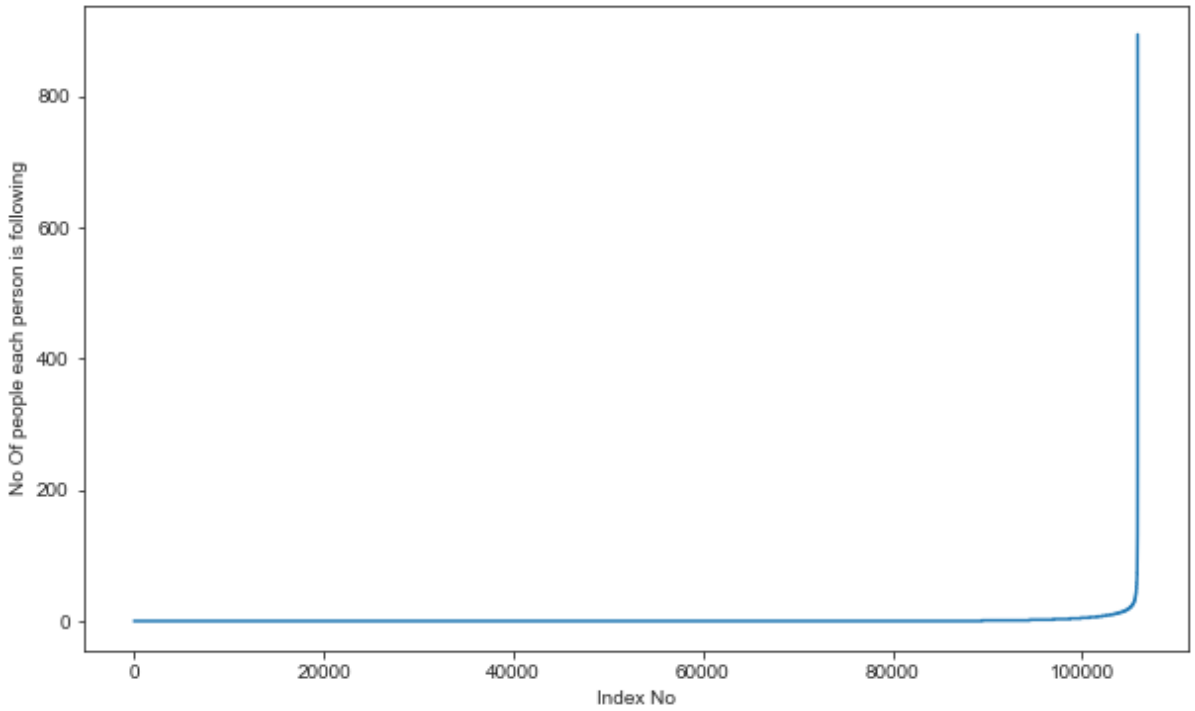
```
99.1 percentile value is 3.0
99.2 percentile value is 3.0
99.3 percentile value is 3.0
99.4 percentile value is 3.0
99.5 percentile value is 3.0
99.6 percentile value is 4.0
99.7 percentile value is 4.0
99.8 percentile value is 4.0
99.9 percentile value is 5.0
100.0 percentile value is 10.0
```

In [21]:
```
%matplotlib inline
sns.set_style('ticks')
fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.distplot(indegree_dist, color='#16A085')
plt.xlabel('PDF of Indegree')
sns.despine()
#plt.show()
```

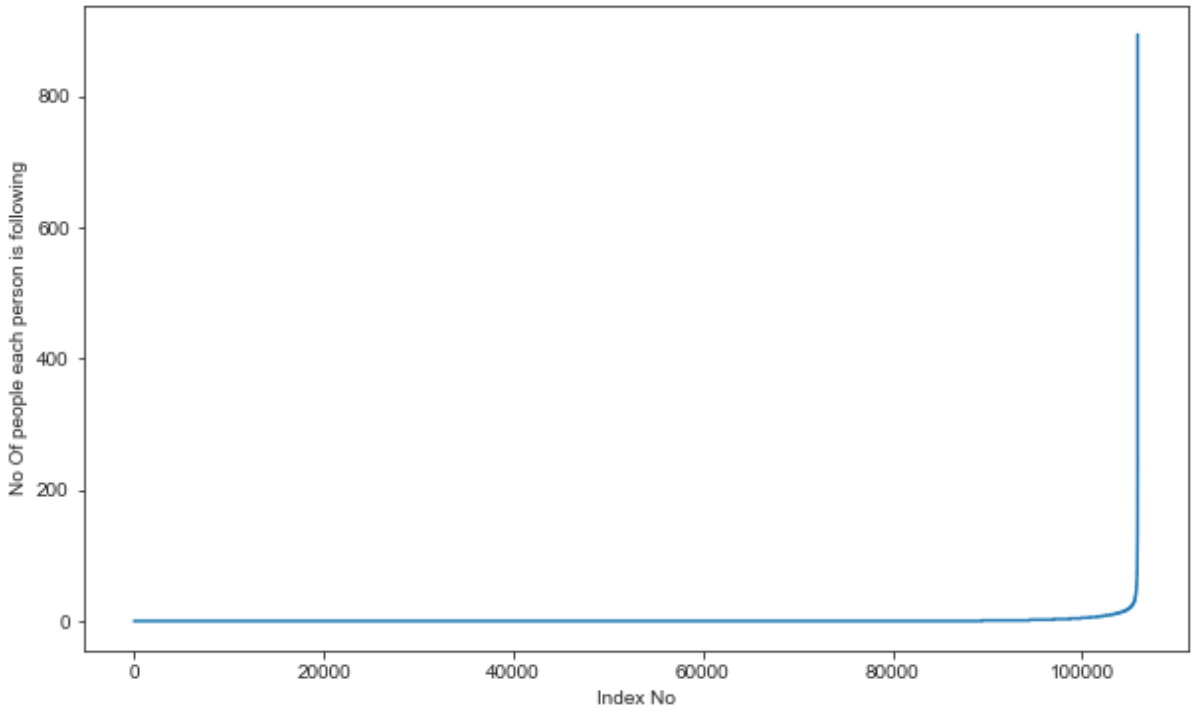


1.2 No of people each person is following

In [22]:
```
outdegree_dist = list(dict(g.out_degree()).values())
outdegree_dist.sort()
plt.figure(figsize=(10,6))
plt.plot(outdegree_dist)
plt.xlabel('Index No')
plt.ylabel('No Of people each person is following')
plt.show()
```



In [23]:
```
indegree_dist = list(dict(g.in_degree()).values())
indegree_dist.sort()
plt.figure(figsize=(10,6))
plt.plot(outdegree_dist[0:1500000])
plt.xlabel('Index No')
plt.ylabel('No Of people each person is following')
plt.show()
```



In [24]:
```
plt.boxplot(indegree_dist)
plt.ylabel('No Of people each person is following')
plt.show()
```

In [25]:
```
### 90-100 percentile
for i in range(0,11):
    print(90+i,'percentile value is',np.percentile(outdegree_dist,90+i
))
```

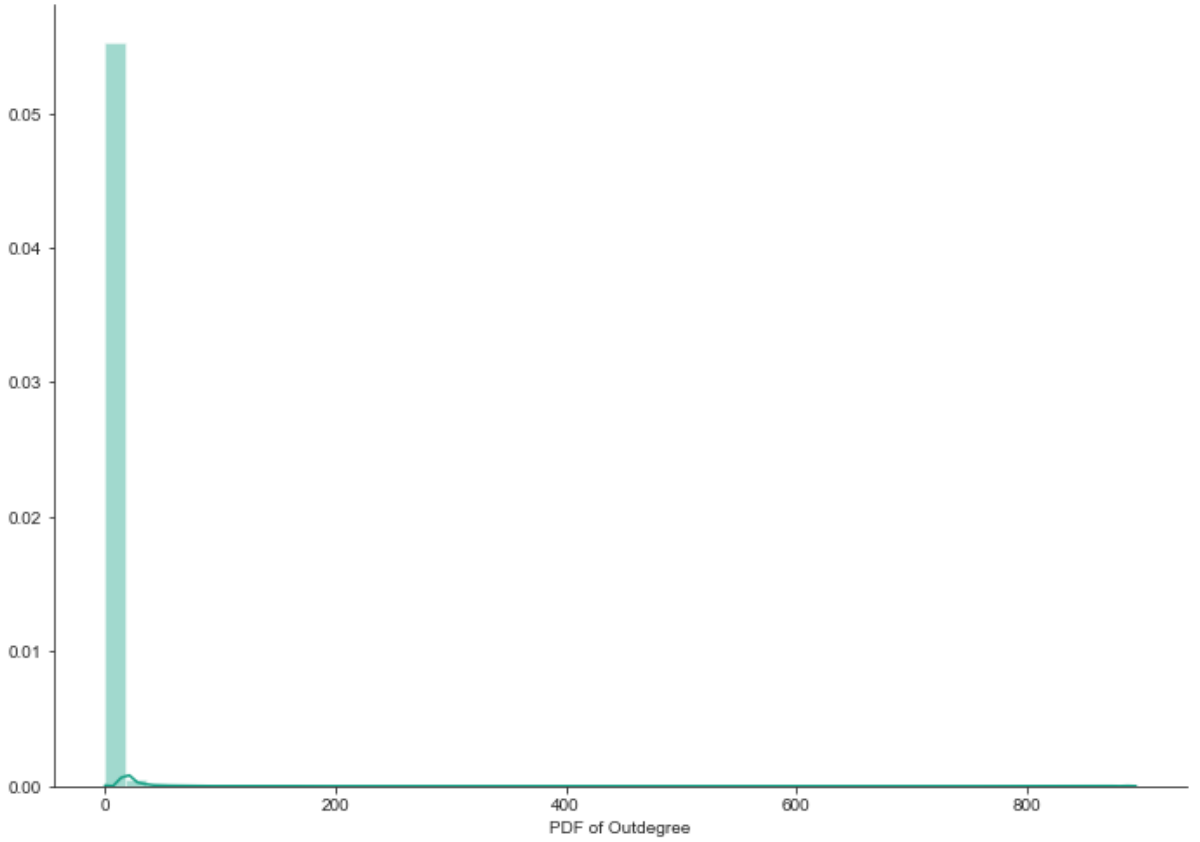
```
90 percentile value is 2.0
91 percentile value is 2.0
92 percentile value is 3.0
93 percentile value is 3.0
94 percentile value is 4.0
95 percentile value is 5.0
96 percentile value is 7.0
97 percentile value is 9.0
98 percentile value is 12.0
99 percentile value is 18.0
100 percentile value is 895.0
```

In [26]:
```
### 99-100 percentile
for i in range(10,110,10):
    print(99+(i/100),'percentile value is',np.percentile(outdegree_dist
,99+(i/100)))
```

```
99.1 percentile value is 19.0
99.2 percentile value is 20.0
```

```
99.3 percentile value is 21.0
99.4 percentile value is 23.0
99.5 percentile value is 25.0
99.6 percentile value is 28.0
99.7 percentile value is 31.46799999999348
99.8 percentile value is 39.0
99.9 percentile value is 56.15600000001723
100.0 percentile value is 895.0
```

In [27]:
```
sns.set_style('ticks')
fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.distplot(outdegree_dist, color='#16A085')
plt.xlabel('PDF of Outdegree')
sns.despine()
```



In [28]:
```
print('No of persons those are not following anyone are' ,sum(np.array(
outdegree_dist)==0),'and % is',
                                    sum(np.array(outdegree_dist)==0)*100/le
n(outdegree_dist) )
```

```
No of persons those are not following anyone are 89403 and % is 84.4659
6438187916
```

In [29]:
```
print('No of persons having zero followers are' ,sum(np.array(indegree_
dist)==0),'and % is',
                                    sum(np.array(indegree_dist)==0)*100/len
(indegree_dist) )
```

```
No of persons having zero followers are 15576 and % is 14.715858094383297
```

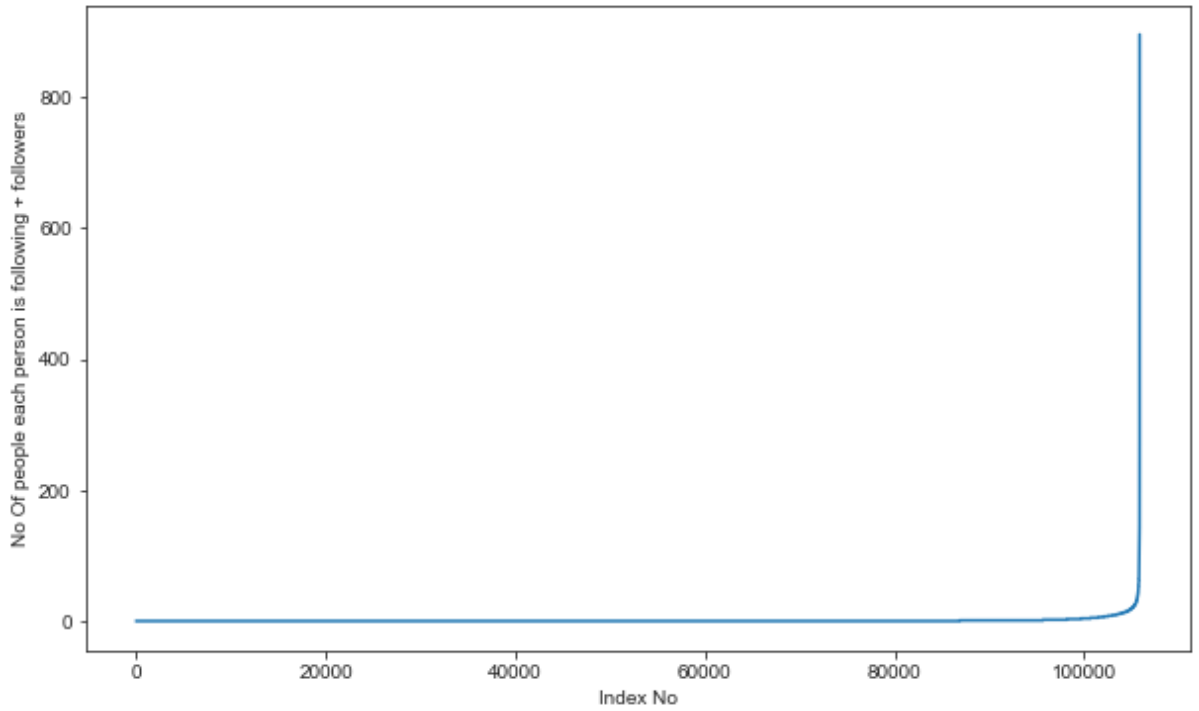
In [30]:
```
count=0
for i in g.nodes():
    if len(list(g.predecessors(i)))==0 :
        if len(list(g.successors(i)))==0:
            count+=1
print('No of persons those are not not following anyone and also not having any followers are',count)
```

```
No of persons those are not not following anyone and also not having any followers are 0
```

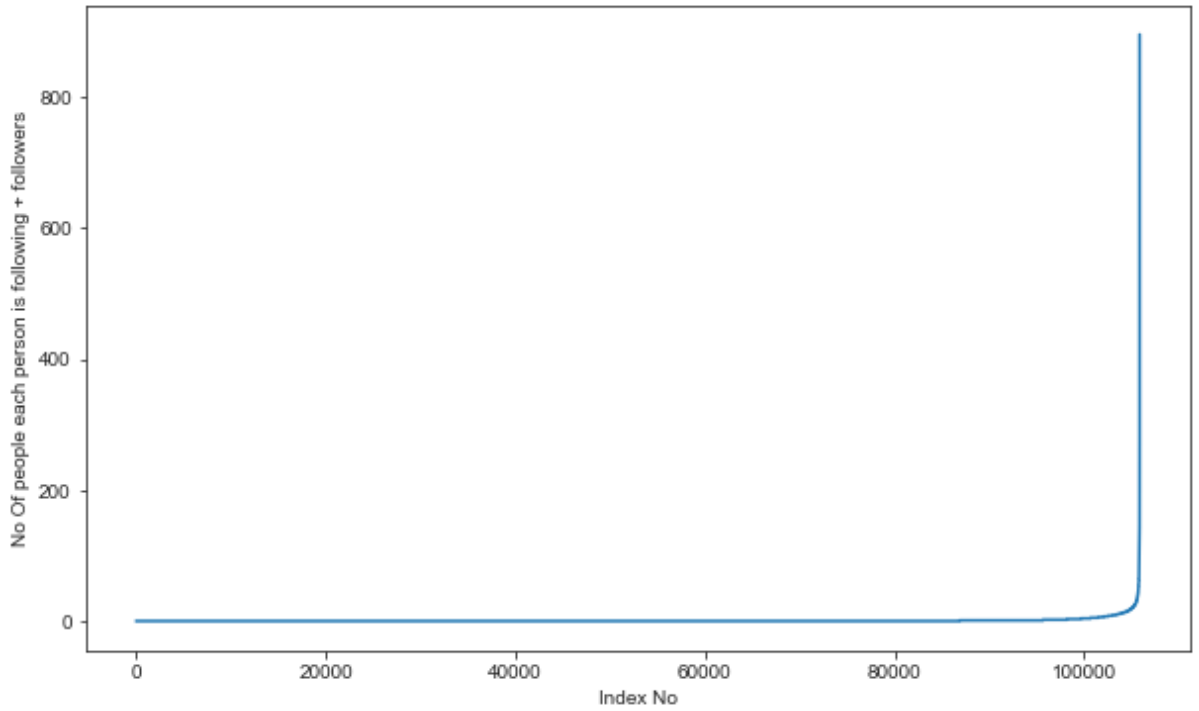
1.3 both followers + following

In [31]:
```
from collections import Counter
dict_in = dict(g.in_degree())
dict_out = dict(g.out_degree())
d = Counter(dict_in) + Counter(dict_out)
in_out_degree = np.array(list(d.values()))
```

In [32]:
```
in_out_degree_sort = sorted(in_out_degree)
plt.figure(figsize=(10,6))
plt.plot(in_out_degree_sort)
plt.xlabel('Index No')
plt.ylabel('No Of people each person is following + followers')
plt.show()
```



In [33]:
```
in_out_degree_sort = sorted(in_out_degree)
plt.figure(figsize=(10,6))
plt.plot(in_out_degree_sort[0:1500000])
plt.xlabel('Index No')
plt.ylabel('No Of people each person is following + followers')
plt.show()
```



In [34]:
```
### 90-100 percentile
for i in range(0,11):
    print(90+i,'percentile value is',np.percentile(in_out_degree_sort,90+i))
```

```
90 percentile value is 2.0
91 percentile value is 3.0
92 percentile value is 3.0
93 percentile value is 4.0
94 percentile value is 4.0
95 percentile value is 5.0
96 percentile value is 7.0
97 percentile value is 9.0
98 percentile value is 12.0
99 percentile value is 18.0
100 percentile value is 895.0
```

In [35]:
```
### 99-100 percentile
for i in range(10,110,10):
    print(99+(i/100),'percentile value is',np.percentile(in_out_degree_
sort,99+(i/100)))
```

```
99.1 percentile value is 19.0
99.2 percentile value is 20.0
99.3 percentile value is 22.0
99.4 percentile value is 23.0
99.5 percentile value is 26.0
99.6 percentile value is 28.0
99.7 percentile value is 32.0
99.8 percentile value is 39.312000000005355
99.9 percentile value is 57.0
100.0 percentile value is 895.0
```

In [36]:
```
print('Min of no of followers + following is',in_out_degree.min())
print(np.sum(in_out_degree==in_out_degree.min()),' persons having minim
um no of followers + following')
```

```
Min of no of followers + following is 1
86847  persons having minimum no of followers + following
```

In [37]:
```
print('Max of no of followers + following is',in_out_degree.max())
print(np.sum(in_out_degree==in_out_degree.max()),' persons having maxim
um no of followers + following')
```

```
Max of no of followers + following is 895
1  persons having maximum no of followers + following
```

In [38]:
```
print('No of persons having followers + following less than 10 are',np.
sum(in_out_degree<10))
```

```
No of persons having followers + following less than 10 are 102945
```

In [39]:
```
print('No of weakly connected components',len(list(nx.weakly_connected_
components(g))))
count=0
```

```
for i in list(nx.weakly_connected_components(g)):
    if len(i)==2:
        count+=1
print('weakly connected components wit 2 nodes',count)
```

```
No of weakly connected components 12030
weakly connected components wit 2 nodes 4526
```

2. Posing a problem as classification problem

2.1 Generating some edges which are not present in graph for supervised learning

Generated Bad links from graph which are not in graph and whose shortest path is greater than 2.

In [40]:
```
%%time
###generating bad edges from given graph
import random
if not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\missing_edges_final.p'):
    #getting all set of edges
    r = csv.reader(open('G:\\machine_learning\\case_study\\Case Study 3 Facebook Friend Recommendation using Graph Mining\\assignment\\Facebook Recruiting\\train_woheader.csv','r'))
    edges = dict()
    for edge in r:
        edges[(edge[0], edge[1])] = 1


    missing_edges = set([])
    while (len(missing_edges)<100000):
        a=random.randint(1, 100000)
```

```
        b=random.randint(1, 100000)
        tmp = edges.get((a,b),-1)
        if tmp == -1 and a!=b:
            try:
                if nx.shortest_path_length(g,source=a,target=b) > 2:

                    missing_edges.add((a,b))
                else:
                    continue
            except:
                    missing_edges.add((a,b))
        else:
            continue
    pickle.dump(missing_edges,open('G:\\machine_learning\\case_study\\C
ase Study 3Facebook Friend Recommendation using Graph Mining\\assignmen
t\\FacebookRecruiting\\missing_edges_final.p','wb'))
else:
    missing_edges = pickle.load(open('G:\\machine_learning\\case_study
\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assign
ment\\FacebookRecruiting\\missing_edges_final.p','rb'))
```

```
Wall time: 1.33 s
```

In [41]:
```
missing_edges = pickle.load(open('G:\\machine_learning\\case_study\\Cas
e Study 3Facebook Friend Recommendation using Graph Mining\\assignment
\\FacebookRecruiting\\missing_edges_final.p','rb'))
len(missing_edges)
```

Out[41]:
```
100000
```

2.2 Training and Test data split:

Removed edges from Graph and used as test data and after removing used that graph for creating features for Train and test data

In [42]:
```
from sklearn.model_selection import train_test_split
if (not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3F
```

```
acebook Friend Recommendation using Graph Mining\\assignment\\FacebookR
ecruiting\\train_pos_after_eda.csv')) and (not os.path.isfile('G:\\mach
ine_learning\\case_study\\Case Study 3Facebook Friend Recommendation us
ing Graph Mining\\assignment\\FacebookRecruiting\\test_pos_after_eda.cs
v')):
    #reading total data df
    df_pos = pd.read_csv('G:\\machine_learning\\case_study\\Case Study
 3Facebook Friend Recommendation using Graph Mining\\assignment\\Facebo
okRecruiting\\train.csv')
    df_pos = df_pos[0:100000]
    df_neg = pd.DataFrame(list(missing_edges), columns=['source_node',
'destination_node'])

    print("Number of nodes in the graph with edges", df_pos.shape[0])
    print("Number of nodes in the graph without edges", df_neg.shape[0
])

    #Trian test split
    #Spiltted data into 80-20
    #positive links and negative links seperatly because we need positi
ve training data only for creating graph
    #and for feature generation
    X_train_pos, X_test_pos, y_train_pos, y_test_pos  = train_test_spli
t(df_pos,np.ones(len(df_pos)),test_size=0.2, random_state=9)
    X_train_neg, X_test_neg, y_train_neg, y_test_neg  = train_test_spli
t(df_neg,np.zeros(len(df_neg)),test_size=0.2, random_state=9)

    print('='*60)
    print("Number of nodes in the train data graph with edges", X_train
_pos.shape[0],"=",y_train_pos.shape[0])
    print("Number of nodes in the train data graph without edges", X_tr
ain_neg.shape[0],"=", y_train_neg.shape[0])
    print('='*60)
    print("Number of nodes in the test data graph with edges", X_test_p
os.shape[0],"=",y_test_pos.shape[0])
    print("Number of nodes in the test data graph without edges", X_tes
t_neg.shape[0],"=",y_test_neg.shape[0])

    #removing header and saving
```

```
    X_train_pos.to_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_pos_after_eda.csv',header=False, index=False)
    X_test_pos.to_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_pos_after_eda.csv',header=False, index=False)
    X_train_neg.to_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_neg_after_eda.csv',header=False, index=False)
    X_test_neg.to_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_neg_after_eda.csv',header=False, index=False)
else:
    #Graph from Traing data only
    del missing_edges
```

```
Number of nodes in the graph with edges 100000
Number of nodes in the graph without edges 100000
=============================================================
Number of nodes in the train data graph with edges 80000 = 80000
Number of nodes in the train data graph without edges 80000 = 80000
=============================================================
Number of nodes in the test data graph with edges 20000 = 20000
Number of nodes in the test data graph without edges 20000 = 20000
```

In [43]:

```
if (os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_pos_after_eda.csv')) and (os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_pos_after_eda.csv')):

    train_graph=nx.read_edgelist('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_pos_after_eda.csv',delimiter=',',create_using=nx.DiGraph(),nodetype=int)
    test_graph=nx.read_edgelist('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_pos_after_eda.csv',delimiter=',',create_using=nx.DiGraph(),nodetype=int)
```

```
    print(nx.info(train_graph))
    print(nx.info(test_graph))

    # finding the unique nodes in the both train and test graphs
    train_nodes_pos = set(train_graph.nodes())
    test_nodes_pos = set(test_graph.nodes())

    trY_teY = len(train_nodes_pos.intersection(test_nodes_pos))
    trY_teN = len(train_nodes_pos - test_nodes_pos)
    teY_trN = len(test_nodes_pos - train_nodes_pos)

    print('no of people common in train and test -- ',trY_teY)
    print('no of people present in train but not present in test -- ',t
rY_teN)

    print('no of people present in test but not present in train -- ',t
eY_trN)
    print(' % of people not there in Train but exist in Test in total T
est data are {} %'.format(teY_trN/len(test_nodes_pos)*100))
```

```
Name:
Type: DiGraph
Number of nodes: 88074
Number of edges: 80000
Average in degree:   0.9083
Average out degree:   0.9083
Name:
Type: DiGraph
Number of nodes: 27902
Number of edges: 20000
Average in degree:   0.7168
Average out degree:   0.7168
no of people common in train and test --  10131
no of people present in train but not present in test --  77943
no of people present in test but not present in train --  17771
 % of people not there in Train but exist in Test in total Test data ar
e 63.69077485484912 %
```

we have a cold start problem here

In [44]:
```
#final train and test data sets
if (not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_after_eda.csv')) and \
(not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_after_eda.csv')) and \
(not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_y.csv')) and \
(not os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_y.csv')) and \
(os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_pos_after_eda.csv')) and \
(os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_pos_after_eda.csv')) and \
(os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_neg_after_eda.csv')) and \
(os.path.isfile('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_neg_after_eda.csv')):

    X_train_pos = pd.read_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\train_pos_after_eda.csv', names=['source_node', 'destination_node'])
    X_test_pos = pd.read_csv('G:\\machine_learning\\case_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\assignment\\FacebookRecruiting\\test_pos_after_eda.csv', names=['source_node', 'destination_node'])
    X_train_neg = pd.read_csv('G:\\machine_learning\\case_study\\Case S
```

```
tudy 3Facebook Friend Recommendation using Graph Mining\\assignment\\Fa
cebookRecruiting\\train_neg_after_eda.csv', names=['source_node', 'dest
ination_node'])
    X_test_neg = pd.read_csv('G:\\machine_learning\\case_study\\Case St
udy 3Facebook Friend Recommendation using Graph Mining\\assignment\\Fac
ebookRecruiting\\test_neg_after_eda.csv', names=['source_node', 'destin
ation_node'])

    print('='*60)
    print("Number of nodes in the train data graph with edges", X_train
_pos.shape[0])
    print("Number of nodes in the train data graph without edges", X_tr
ain_neg.shape[0])
    print('='*60)
    print("Number of nodes in the test data graph with edges", X_test_p
os.shape[0])
    print("Number of nodes in the test data graph without edges", X_tes
t_neg.shape[0])

    X_train = X_train_pos.append(X_train_neg,ignore_index=True)
    y_train = np.concatenate((y_train_pos,y_train_neg))
    X_test = X_test_pos.append(X_test_neg,ignore_index=True)
    y_test = np.concatenate((y_test_pos,y_test_neg))

    X_train.to_csv('G:\\machine_learning\\case_study\\Case Study 3Faceb
ook Friend Recommendation using Graph Mining\\assignment\\FacebookRecru
iting\\train_after_eda.csv',header=False,index=False)
    X_test.to_csv('G:\\machine_learning\\case_study\\Case Study 3Facebo
ok Friend Recommendation using Graph Mining\\assignment\\FacebookRecrui
ting\\test_after_eda.csv',header=False,index=False)
    pd.DataFrame(y_train.astype(int)).to_csv('G:\\machine_learning\\cas
e_study\\Case Study 3Facebook Friend Recommendation using Graph Mining
\\assignment\\FacebookRecruiting\\train_y.csv',header=False,index=False
)
    pd.DataFrame(y_test.astype(int)).to_csv('G:\\machine_learning\\case
_study\\Case Study 3Facebook Friend Recommendation using Graph Mining\\
assignment\\FacebookRecruiting\\test_y.csv',header=False,index=False)
```

```
============================================================
Number of nodes in the train data graph with edges 80000
```

```
Number of nodes in the train data graph without edges 80000
=============================================================
Number of nodes in the test data graph with edges 20000
Number of nodes in the test data graph without edges 20000
```

In [45]:
```
print("Data points in train data",X_train.shape)
print("Data points in test data",X_test.shape)
print("Shape of traget variable in train",y_train.shape)
print("Shape of traget variable in test", y_test.shape)
```

```
Data points in train data (160000, 2)
Data points in test data (40000, 2)
Shape of traget variable in train (160000,)
Shape of traget variable in test (40000,)
```

In [46]:
```
# computed and store the data for featurization
# please check out FB_featurization.ipynb
```