FB_EDA

May 25, 2019

Social network Graph Link Prediction - Facebook Challenge

0.0.1 Problem statement:

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

0.0.2 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

0.0.3 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

0.0.4 Business objectives and constraints:

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

0.0.5 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 [3]: #reading graph
        if not os.path.isfile('data/after_eda/train_woheader.csv'):
            traincsv = pd.read_csv('data/train.csv')
            print(traincsv[traincsv.isna().any(1)])
            print(traincsv.info())
            print("Number of diplicate entries: ",sum(traincsv.duplicated()))
            traincsv.to_csv('data/after_eda/train_woheader.csv',header=False,index=False)
            print("saved the graph into file")
        else:
            g=nx.read_edgelist('data/after_eda/train_woheader.csv',delimiter=',',create_using=n
            print(nx.info(g))
```

```
Name:
Type: DiGraph
Number of nodes: 1862220
Number of edges: 9437519
Average in degree:   5.0679
Average out degree:   5.0679
```

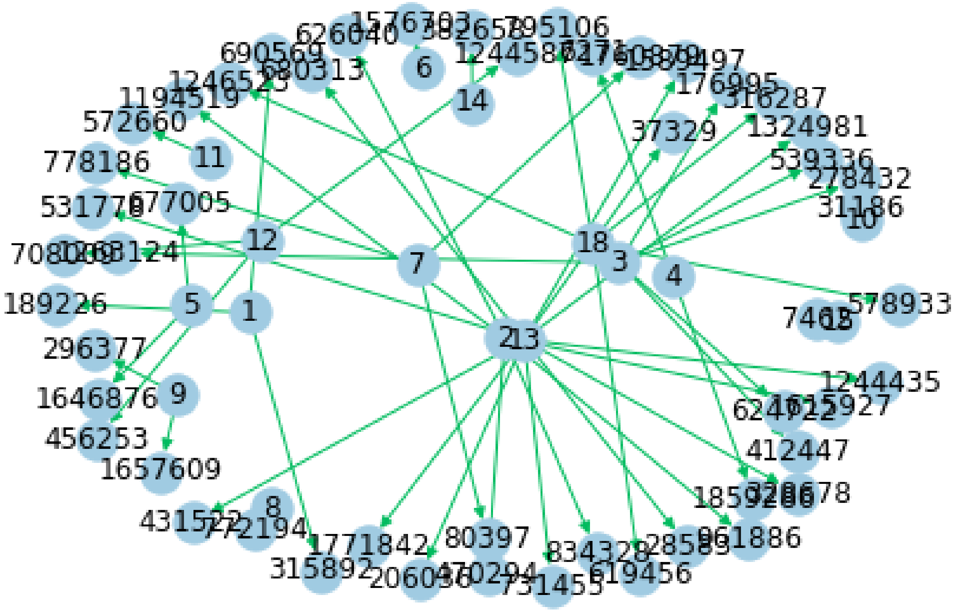
Displaying a sub graph

```
In [6]: if not os.path.isfile('train_woheader_sample.csv'):
            pd.read_csv('data/train.csv', nrows=50).to_csv('train_woheader_sample.csv',header=

        subgraph=nx.read_edgelist('train_woheader_sample.csv',delimiter=',',create_using=nx.Di
        # https://stackoverflow.com/questions/9402255/drawing-a-huge-graph-with-networkx-and-m

        pos=nx.spring_layout(subgraph)
        nx.draw(subgraph,pos,node_color='#A0CBE2',edge_color='#00bb5e',width=1,edge_cmap=plt.c
        plt.savefig("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
```



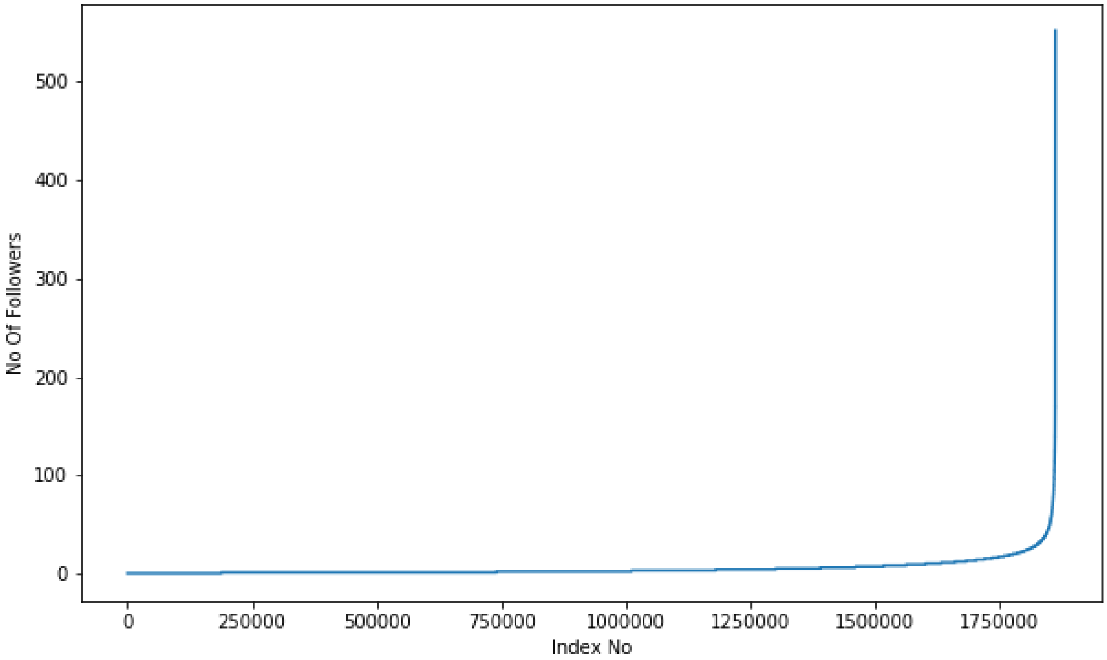
1 1. Exploratory Data Analysis

```
In [7]: # No of Unique persons
        print("The number of unique persons",len(g.nodes()))
```

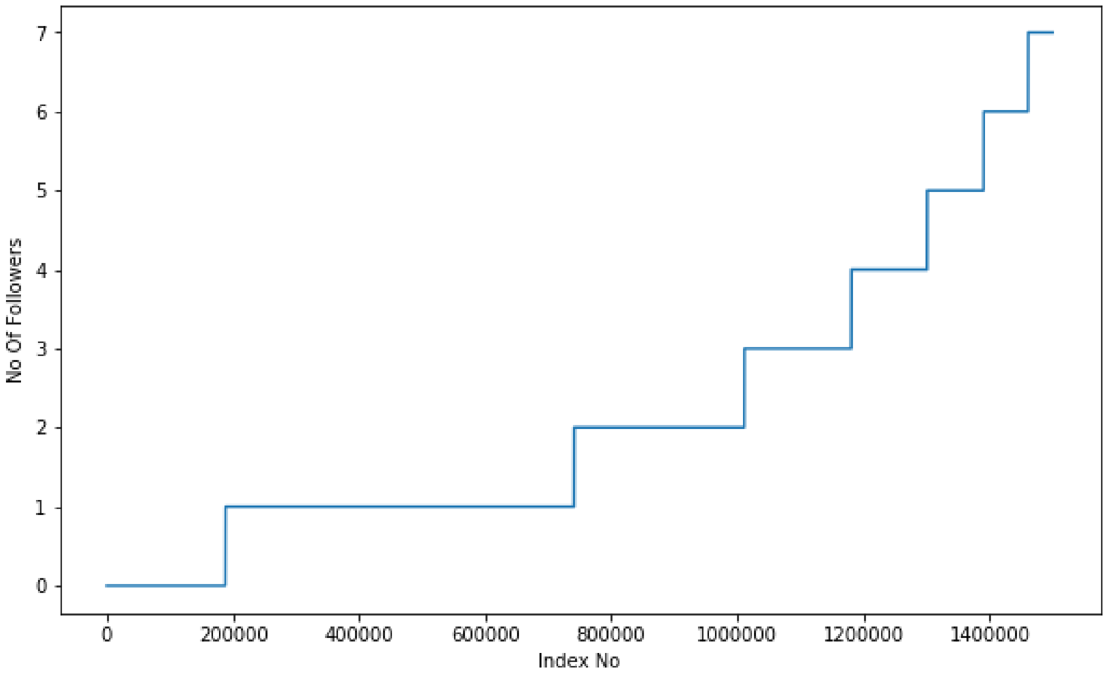
```
The number of unique persons 1862220
```

1.1 1.1 No of followers for each person

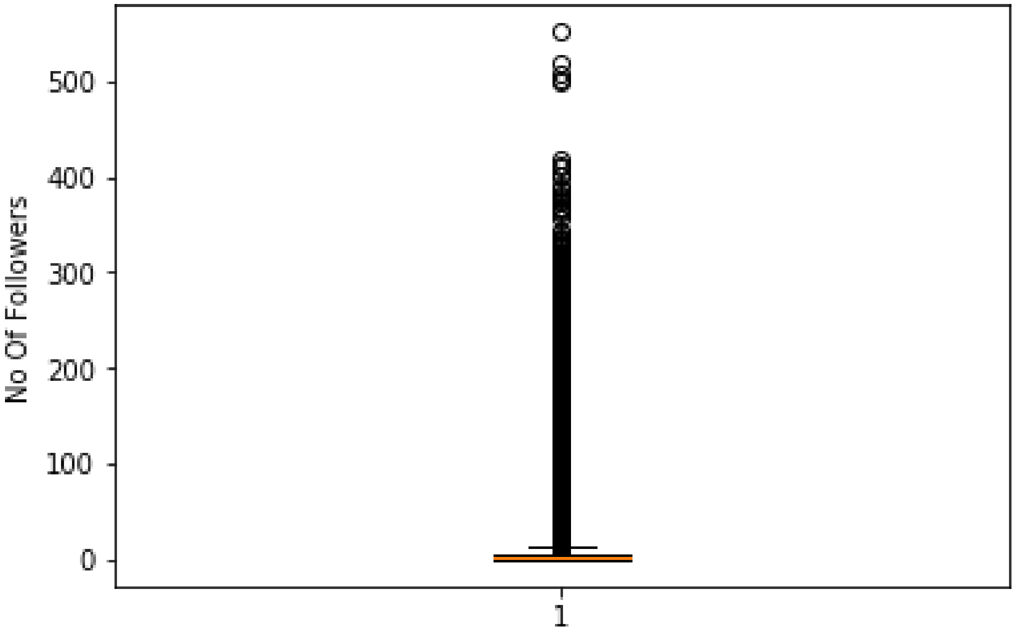
```
In [8]: indegree_dist = list(dict(g.in_degree()).values())
        indegree_dist.sort()
        plt.figure(figsize=(10,6))
        plt.plot(indegree_dist)
        plt.xlabel('Index No')
        plt.ylabel('No Of Followers')
        plt.show()
```



```
In [9]: indegree_dist = list(dict(g.in_degree()).values())
        indegree_dist.sort()
        plt.figure(figsize=(10,6))
        plt.plot(indegree_dist[0:1500000])
        plt.xlabel('Index No')
        plt.ylabel('No Of Followers')
        plt.show()
```



```
In [10]: plt.boxplot(indegree_dist)
         plt.ylabel('No Of Followers')
         plt.show()
```



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

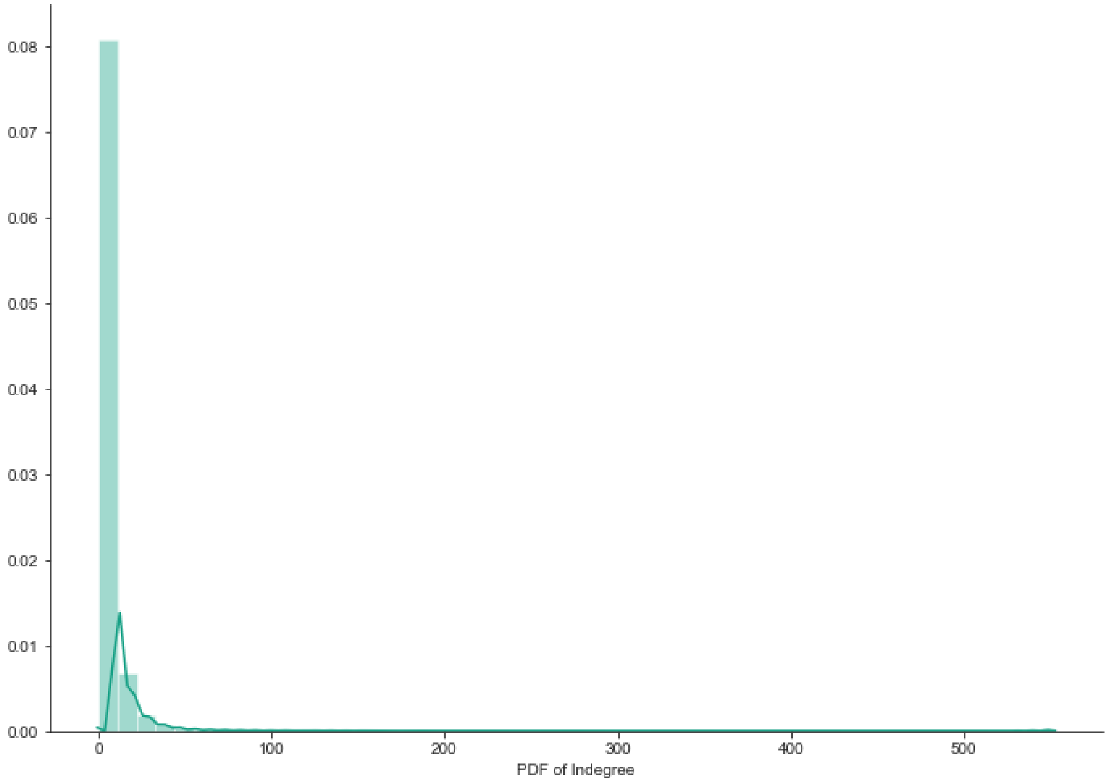
```
90 percentile value is 12.0
91 percentile value is 13.0
92 percentile value is 14.0
93 percentile value is 15.0
94 percentile value is 17.0
95 percentile value is 19.0
96 percentile value is 21.0
97 percentile value is 24.0
98 percentile value is 29.0
99 percentile value is 40.0
100 percentile value is 552.0
```

99% of data having followers of 40 only.

```
In [12]: ### 99-100 percentile
         for i in range(10,110,10):
             print(99+(i/100),'percentile value is',np.percentile(indegree_dist,99+(i/100)))
```

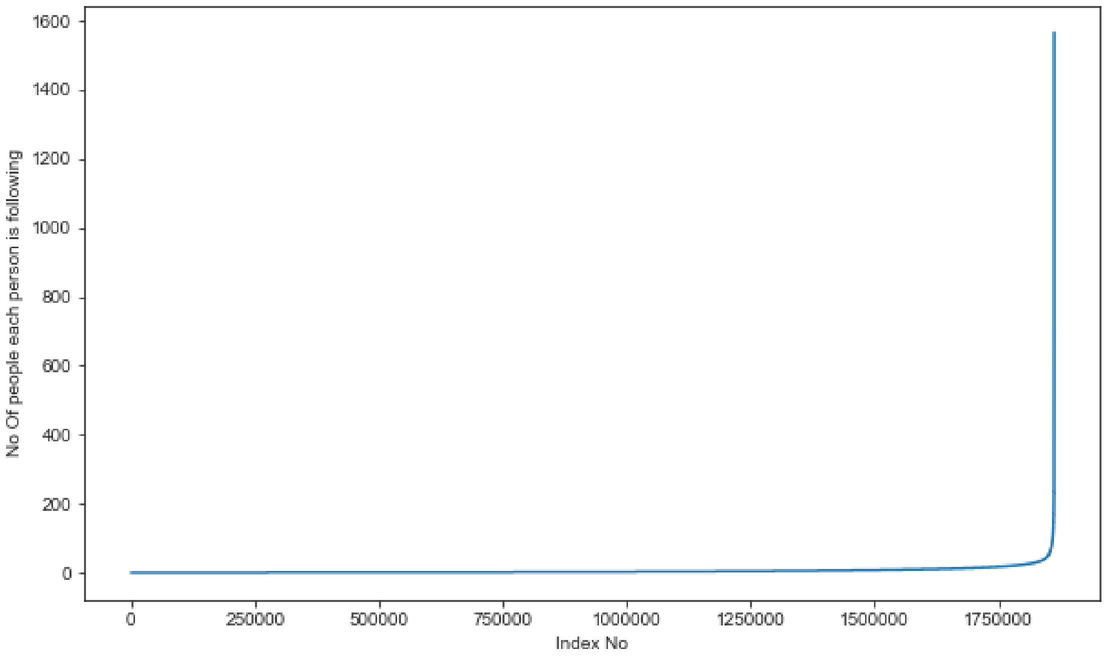
```
99.1 percentile value is 42.0
99.2 percentile value is 44.0
99.3 percentile value is 47.0
99.4 percentile value is 50.0
99.5 percentile value is 55.0
99.6 percentile value is 61.0
99.7 percentile value is 70.0
99.8 percentile value is 84.0
99.9 percentile value is 112.0
100.0 percentile value is 552.0
```

```
In [13]: %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()
```

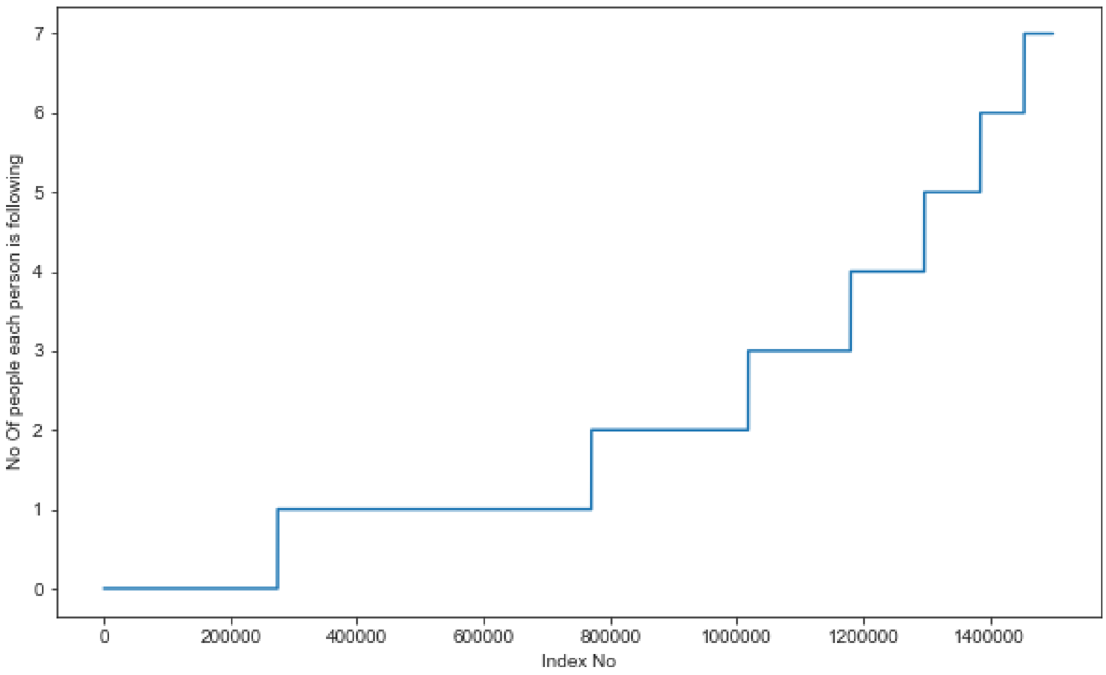


1.2 1.2 No of people each person is following

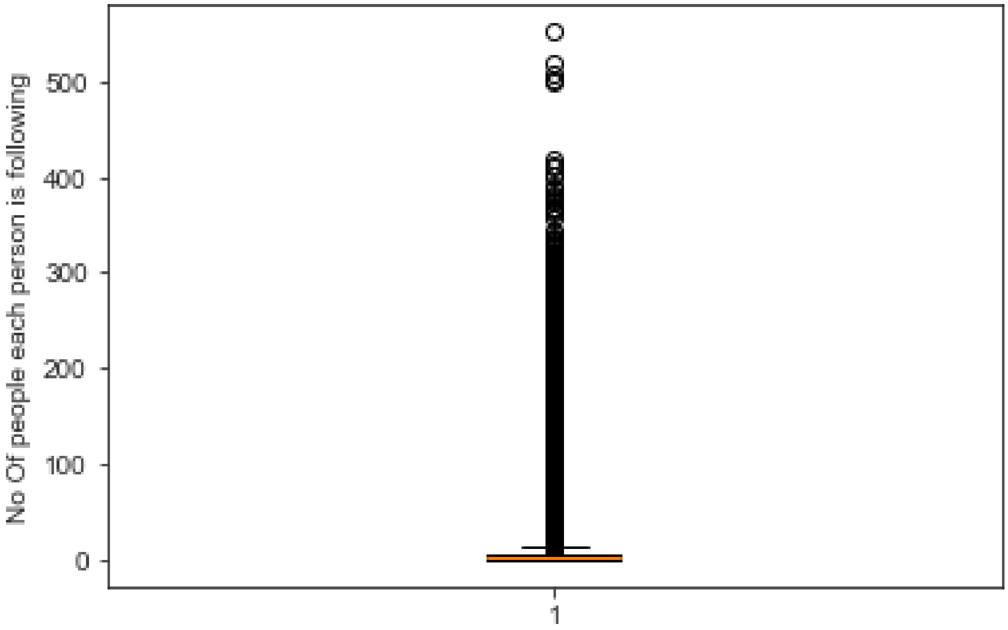
```
In [14]: outdegree_dist = list(dict(g.out_degree()).values())
         outdegree_dist.sort()
         plt.figure(figsize=(10,6))
         plt.plot(outdegree_dist)
         plt.xlabel('Index No')
         plt.ylabel('No Of people each person is following')
         plt.show()
```



```
In [15]: 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 [16]: plt.boxplot(indegree_dist)
         plt.ylabel('No Of people each person is following')
         plt.show()
```



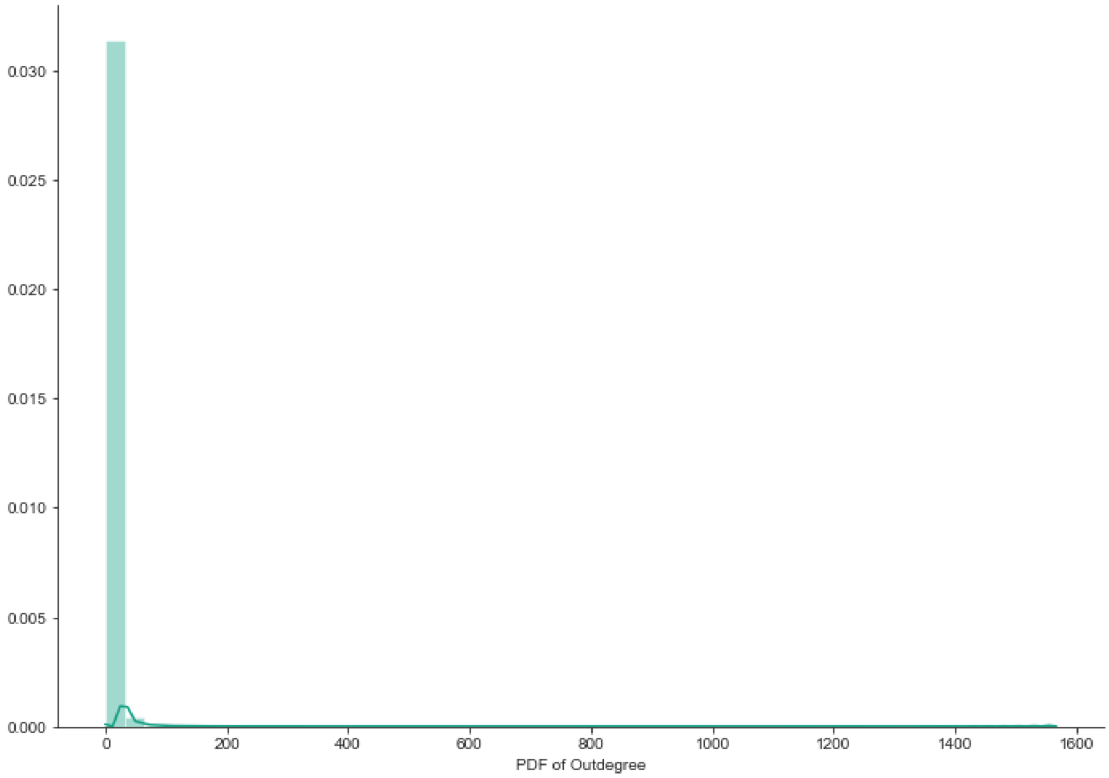
```
In [17]: ### 90-100 percentile
         for i in range(0,11):
             print(90+i,'percentile value is',np.percentile(outdegree_dist,90+i))
```

```
90 percentile value is 12.0
91 percentile value is 13.0
92 percentile value is 14.0
93 percentile value is 15.0
94 percentile value is 17.0
95 percentile value is 19.0
96 percentile value is 21.0
97 percentile value is 24.0
98 percentile value is 29.0
99 percentile value is 40.0
100 percentile value is 1566.0
```

```
In [18]: ### 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 42.0
99.2 percentile value is 45.0
99.3 percentile value is 48.0
99.4 percentile value is 52.0
99.5 percentile value is 56.0
99.6 percentile value is 63.0
99.7 percentile value is 73.0
99.8 percentile value is 90.0
99.9 percentile value is 123.0
100.0 percentile value is 1566.0
```

```
In [19]: 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 [20]: print('No of persons those are not following anyone are' ,sum(np.array(outdegree_dist
                                                    sum(np.array(outdegree_dist)==0)*100/len(outdegree_di

No of persons those are not following anyone are 274512 and % is 14.741115442858524
```

```
In [21]: print('No of persons having zero followers are' ,sum(np.array(indegree_dist)==0),'and
                                                    sum(np.array(indegree_dist)==0)*100/len(indegree_dist

No of persons having zero followers are 188043 and % is 10.097786512871734
```

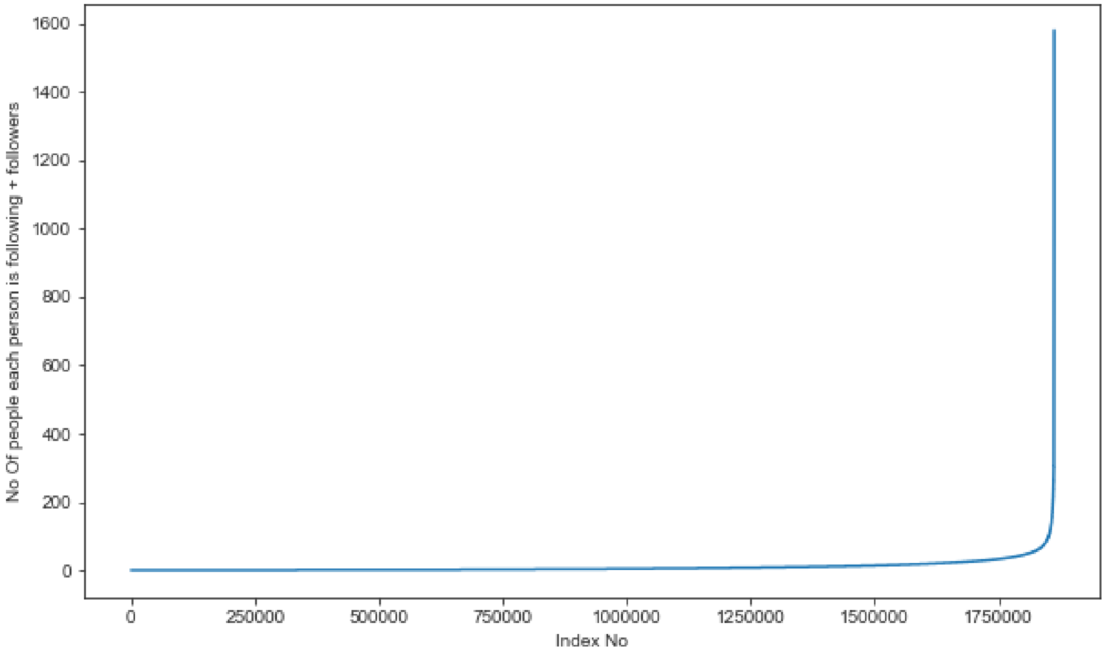
```
In [22]: count=0
         for i in g.nodes():
             if len(list(g.predecessors(i)))==0 :
                 if len(list(g.successors(i)))==0:
                     count+=1
         print('No of persons those are not not following anyone and also not having any follo

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

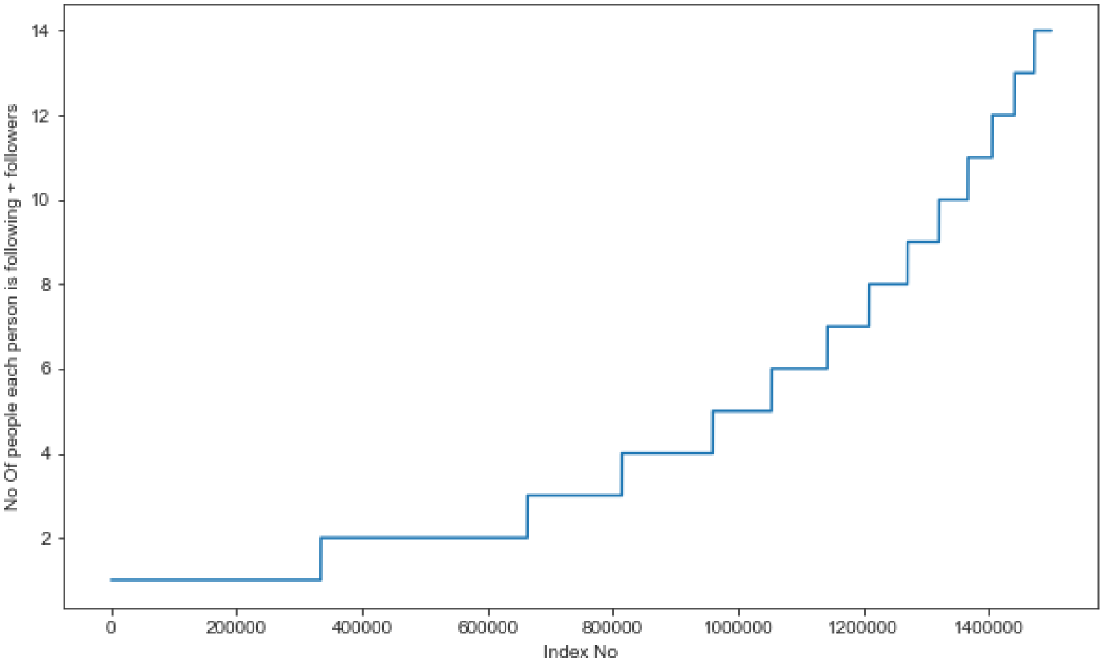
1.3 1.3 both followers + following

```
In [23]: 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 [24]: 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 [25]: 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 [26]: ### 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 24.0
91 percentile value is 26.0
92 percentile value is 28.0
93 percentile value is 31.0
94 percentile value is 33.0
95 percentile value is 37.0
96 percentile value is 41.0
97 percentile value is 48.0
98 percentile value is 58.0
99 percentile value is 79.0
100 percentile value is 1579.0
```

```
In [27]: ### 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 83.0
99.2 percentile value is 87.0
99.3 percentile value is 93.0
99.4 percentile value is 99.0
```

```
99.5 percentile value is 108.0
99.6 percentile value is 120.0
99.7 percentile value is 138.0
99.8 percentile value is 168.0
99.9 percentile value is 221.0
100.0 percentile value is 1579.0
```

```
In [28]: print('Min of no of followers + following is',in_out_degree.min())
         print(np.sum(in_out_degree==in_out_degree.min()),' persons having minimum no of follo
```

```
Min of no of followers + following is 1
334291  persons having minimum no of followers + following
```

```
In [29]: print('Max of no of followers + following is',in_out_degree.max())
         print(np.sum(in_out_degree==in_out_degree.max()),' persons having maximum no of follo
```

```
Max of no of followers + following is 1579
1  persons having maximum no of followers + following
```

```
In [30]: print('No of persons having followers + following less than 10 are',np.sum(in_out_deg
```

```
No of persons having followers + following less than 10 are 1320326
```

```
In [31]: 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 45558
weakly connected components wit 2 nodes 32195
```

2 2. Posing a problem as classification problem

2.1 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 [8]: %%time
        ###generating bad edges from given graph
        import random
        if not os.path.isfile('data/after_eda/missing_edges_final.p'):
            #getting all set of edges
            r = csv.reader(open('data/after_eda/train_woheader.csv','r'))
            edges = dict()
            for edge in r:
                edges[(edge[0], edge[1])] = 1


            missing_edges = set([])
            while (len(missing_edges)<9437519):
                a=random.randint(1, 1862220)
                b=random.randint(1, 1862220)
                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('data/after_eda/missing_edges_final.p','wb'))
        else:
            missing_edges = pickle.load(open('data/after_eda/missing_edges_final.p','rb'))
```

```
Wall time: 6.71 s
```

```
In [47]: len(missing_edges)
```

```
Out[47]: 9437519
```

2.2 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 [9]: from sklearn.model_selection import train_test_split
        if (not os.path.isfile('data/after_eda/train_pos_after_eda.csv')) and (not os.path.isf
            #reading total data df
            df_pos = pd.read_csv('data/train.csv')
            df_neg = pd.DataFrame(list(missing_edges), columns=['source_node', 'destination_nod

            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 positive training dat
            #and for feature generation
            X_train_pos, X_test_pos, y_train_pos, y_test_pos  = train_test_split(df_pos,np.ones
            X_train_neg, X_test_neg, y_train_neg, y_test_neg  = train_test_split(df_neg,np.zer

            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_train_neg.shape[0
            print('='*60)
            print("Number of nodes in the test data graph with edges", X_test_pos.shape[0],"="
            print("Number of nodes in the test data graph without edges", X_test_neg.shape[0],

            #removing header and saving
            X_train_pos.to_csv('data/after_eda/train_pos_after_eda.csv',header=False, index=Fal
            X_test_pos.to_csv('data/after_eda/test_pos_after_eda.csv',header=False, index=False
            X_train_neg.to_csv('data/after_eda/train_neg_after_eda.csv',header=False, index=Fal
            X_test_neg.to_csv('data/after_eda/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 9437519
Number of nodes in the graph without edges 9437519
============================================================
Number of nodes in the train data graph with edges 7550015 = 7550015
Number of nodes in the train data graph without edges 7550015 = 7550015
============================================================
Number of nodes in the test data graph with edges 1887504 = 1887504
Number of nodes in the test data graph without edges 1887504 = 1887504
```

```
In [2]: if (os.path.isfile('data/after_eda/train_pos_after_eda.csv')) and (os.path.isfile('data
            train_graph=nx.read_edgelist('data/after_eda/train_pos_after_eda.csv',delimiter=',
            test_graph=nx.read_edgelist('data/after_eda/test_pos_after_eda.csv',delimiter=',',
            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 -- ',trY_teN)
            print('no of people present in test but not present in train -- ',teY_trN)
            print(' % of people not there in Train but exist in Test in total Test data are {}
```

```
Name:
Type: DiGraph
Number of nodes: 1780722
Number of edges: 7550015
Average in degree:   4.2399
Average out degree:   4.2399
Name:
Type: DiGraph
Number of nodes: 1144623
Number of edges: 1887504
Average in degree:   1.6490
Average out degree:   1.6490
no of people common in train and test --  1063125
no of people present in train but not present in test --  717597
no of people present in test but not present in train --  81498
 % of people not there in Train but exist in Test in total Test data are 7.1200735962845405 %
```

we have a cold start problem here

```
In [10]: #final train and test data sets
         if (not os.path.isfile('data/after_eda/train_after_eda.csv')) and \
         (not os.path.isfile('data/after_eda/test_after_eda.csv')) and \
         (not os.path.isfile('data/train_y.csv')) and \
         (not os.path.isfile('data/test_y.csv')) and \
         (os.path.isfile('data/after_eda/train_pos_after_eda.csv')) and \
         (os.path.isfile('data/after_eda/test_pos_after_eda.csv')) and \
         (os.path.isfile('data/after_eda/train_neg_after_eda.csv')) and \
         (os.path.isfile('data/after_eda/test_neg_after_eda.csv')):

             X_train_pos = pd.read_csv('data/after_eda/train_pos_after_eda.csv', names=['sourc
             X_test_pos = pd.read_csv('data/after_eda/test_pos_after_eda.csv', names=['source_
             X_train_neg = pd.read_csv('data/after_eda/train_neg_after_eda.csv', names=['sourc
             X_test_neg = pd.read_csv('data/after_eda/test_neg_after_eda.csv', names=['source_

             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_train_neg.shape[
             print('='*60)
             print("Number of nodes in the test data graph with edges", X_test_pos.shape[0])
             print("Number of nodes in the test data graph without edges", X_test_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('data/after_eda/train_after_eda.csv',header=False,index=False)
             X_test.to_csv('data/after_eda/test_after_eda.csv',header=False,index=False)
             pd.DataFrame(y_train.astype(int)).to_csv('data/train_y.csv',header=False,index=Fal
             pd.DataFrame(y_test.astype(int)).to_csv('data/test_y.csv',header=False,index=Fals
```

```
============================================================
Number of nodes in the train data graph with edges 7550015
Number of nodes in the train data graph without edges 7550015
============================================================
Number of nodes in the test data graph with edges 1887504
Number of nodes in the test data graph without edges 1887504
```

```
In [11]: 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 (15100030, 2)
Data points in test data (3775008, 2)
Shape of traget variable in train (15100030,)
Shape of traget variable in test (3775008,)
```

```
In [12]: # computed and store the data for featurization
```

```
In [ ]:
```