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 [0]:
```
#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 [0]:
```python
#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,in
dex=False)
    print("saved the graph into file")
else:
    g=nx.read_edgelist('data/after_eda/train_woheader.csv',delimiter=
',',create_using=nx.DiGraph(),nodetype=int)
    print(nx.info(g))
```

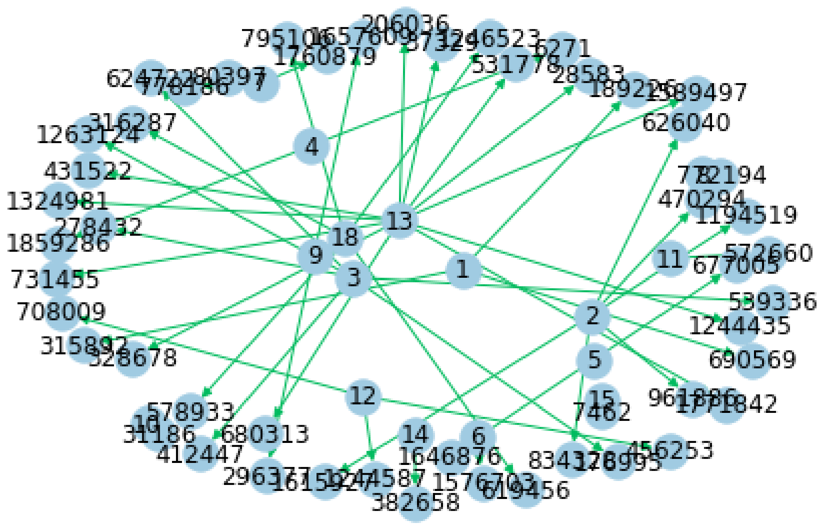
```
Name:
Type: DiGraph
Number of nodes: 1862220
Number of edges: 9437519
Average in degree:   5.0679
Average out degree:   5.0679
```

> Displaying a sub graph

In [0]:
```python
if not os.path.isfile('train_woheader_sample.csv'):
    pd.read_csv('data/train.csv', nrows=50).to_csv('train_woheader_samp
le.csv',header=False,index=False)

subgraph=nx.read_edgelist('train_woheader_sample.csv',delimiter=',',cre
ate_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)
```

```python
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
```



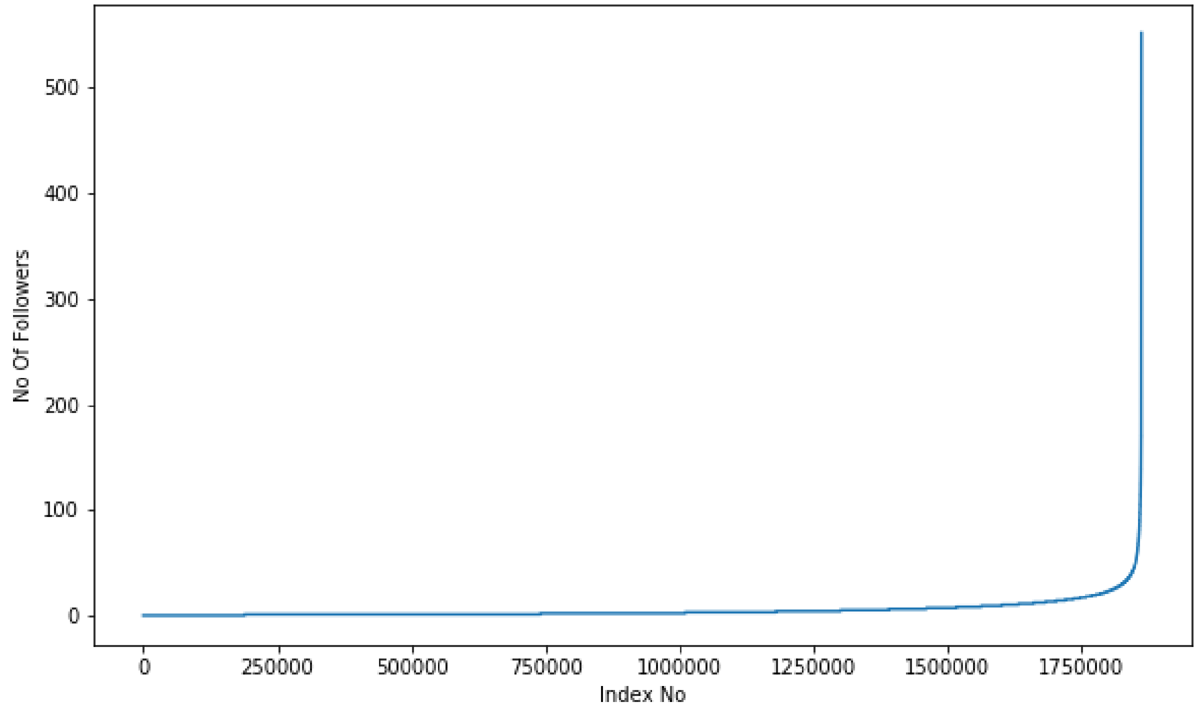
1. Exploratory Data Analysis

In [0]:
```python
# No of Unique persons
print("The number of unique persons",len(g.nodes()))
```

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

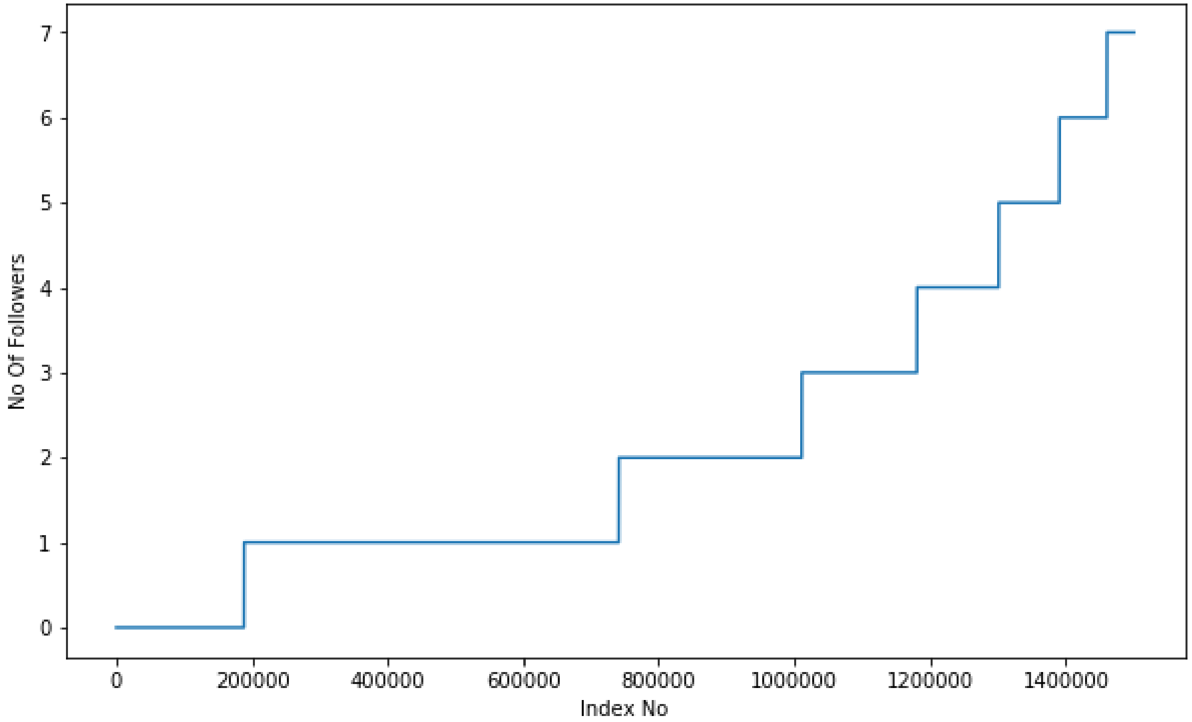
1.1 No of followers for each person

In [0]:
```
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()
```



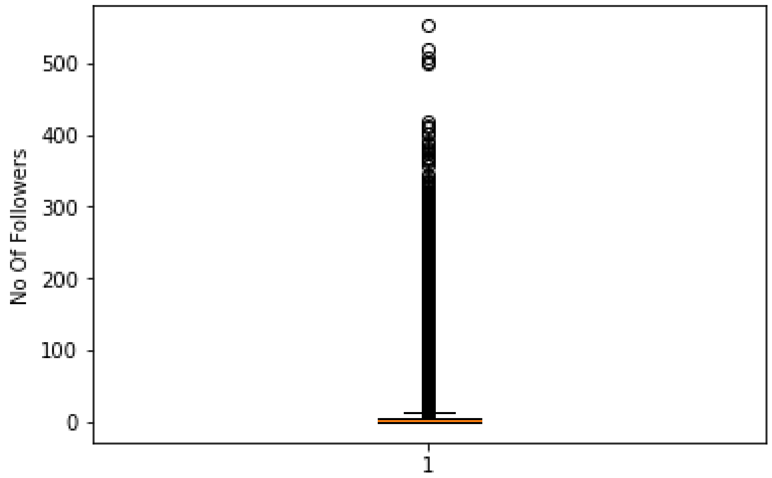
In [0]:
```
indegree_dist = list(dict(g.in_degree()).values())
indegree_dist.sort()
plt.figure(figsize=(10,6))
plt.plot(indegree_dist[0:1500000])
plt.xlabel('Index No')
```

```
plt.ylabel('No Of Followers')
plt.show()
```



In [0]:

```
plt.boxplot(indegree_dist)
plt.ylabel('No Of Followers')
plt.show()
```



In [0]:
```
### 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.

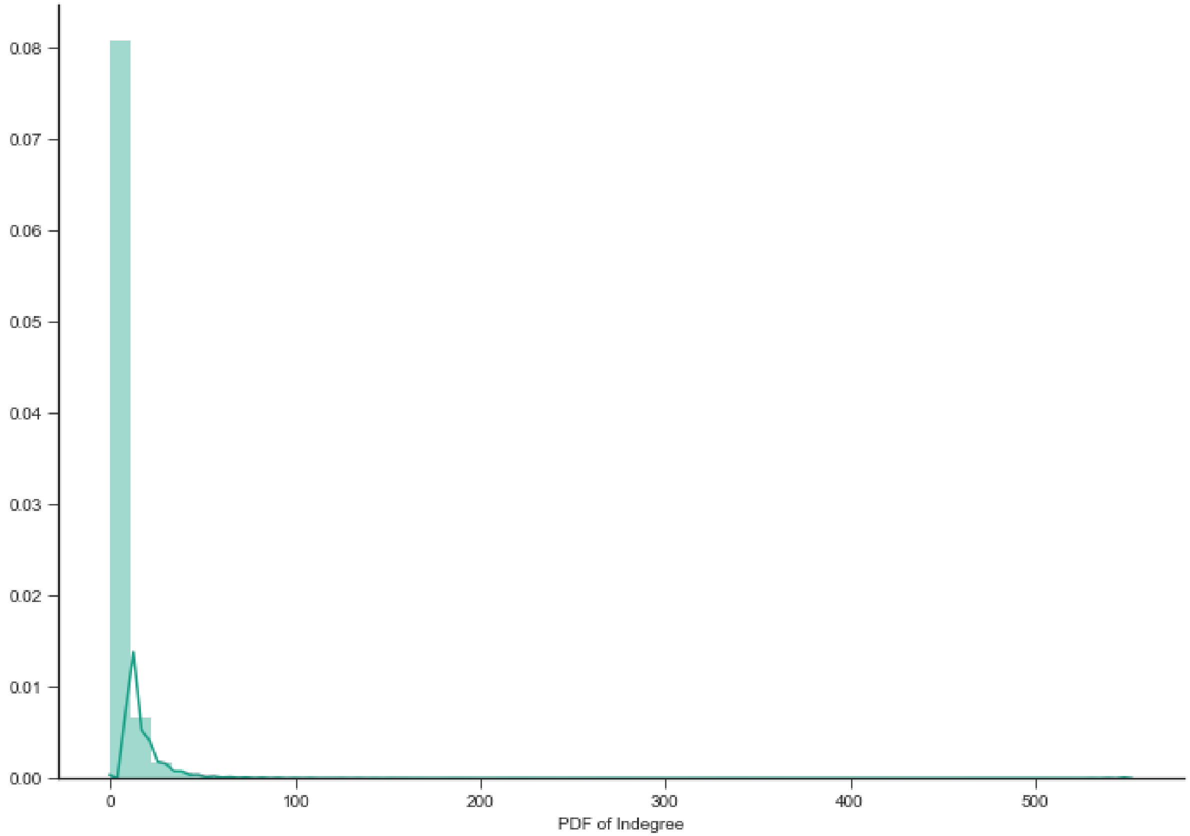
In [0]:
```
### 99-100 percentile
for i in range(10,110,10):
    print(99+(i/100),'percentile value is',np.percentile(indegree_dist,
99+(i/100)))
```

```
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 [0]:

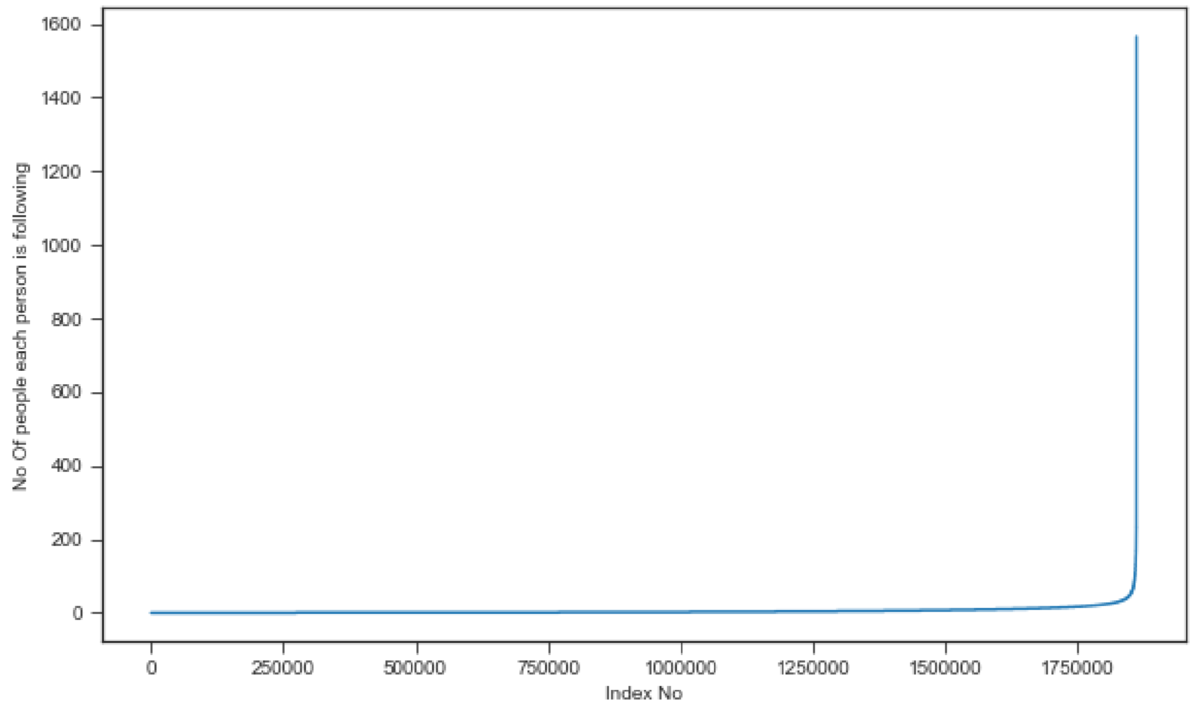
```
%matplotlib inline
sns.set_style('ticks')
fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.distplot(indegree_dist, color='#16A085')
plt.xlabel('PDF of Indegree')
sns.despine()
#plt.show()
```

```
D:\installed\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6571:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by
the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



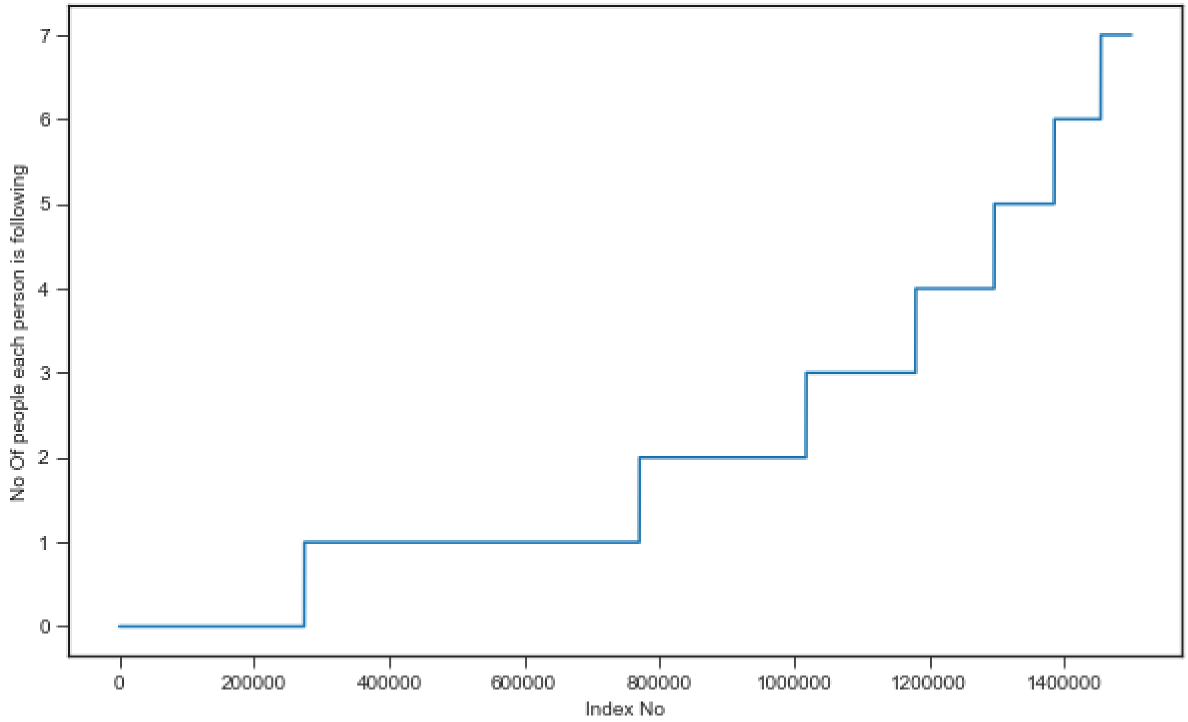
1.2 No of people each person is following

In [0]:
```
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()
```

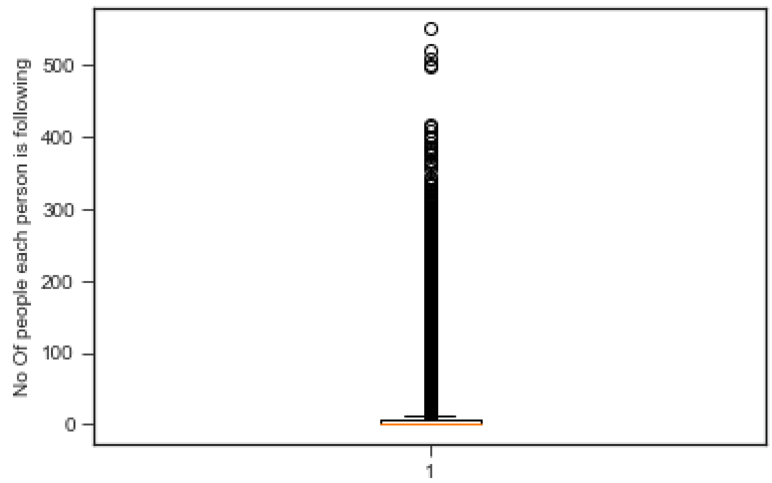


In [0]:

```
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 [0]:
```
plt.boxplot(indegree_dist)
plt.ylabel('No Of people each person is following')
plt.show()
```



In [0]:
```
### 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 [0]:
```
### 99-100 percentile
for i in range(10,110,10):
    print(99+(i/100),'percentile value is',np.percentile(outdegree_dist
,99+(i/100)))
```

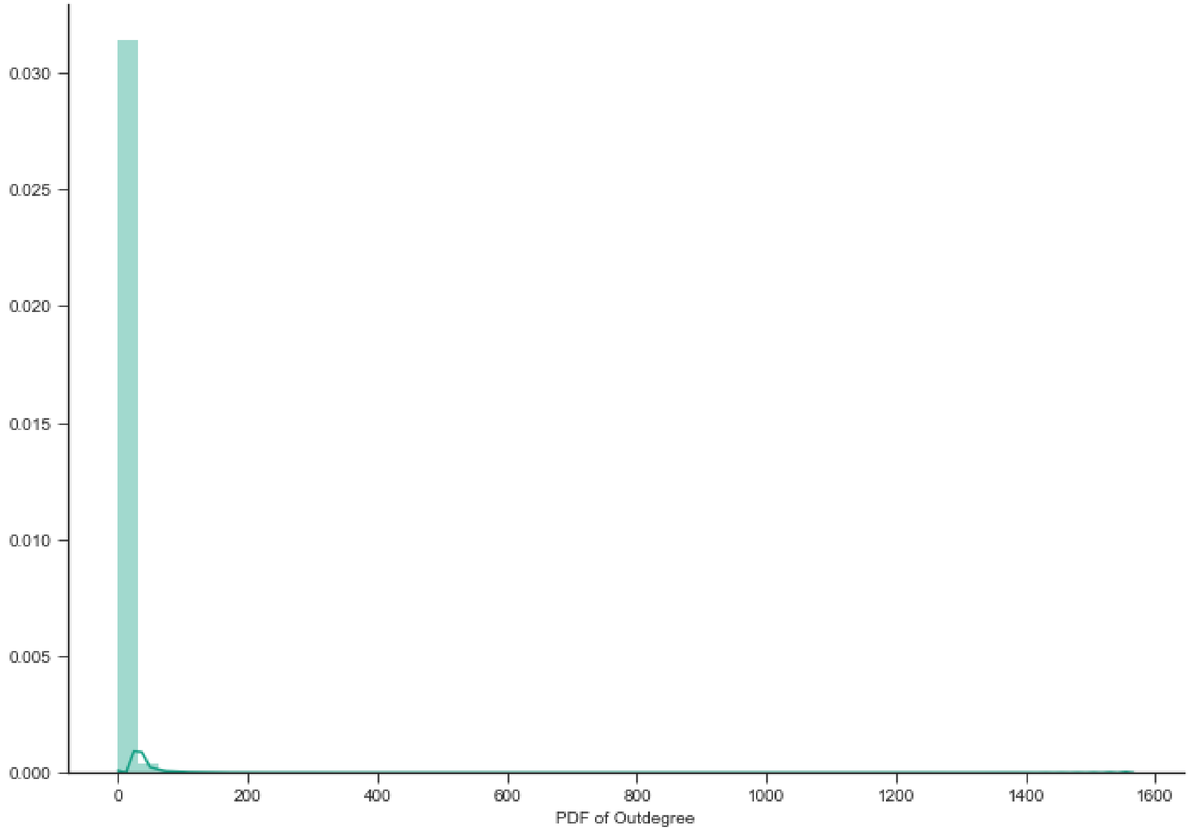
```
99.1 percentile value is 42.0
```

```
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 [0]:

```
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()
```

```
D:\installed\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6571:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by
the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [0]:
```
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 274512 and % is 14.741
115442858524

In [0]:
```
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 188043 and % is 10.097786512871
734
```

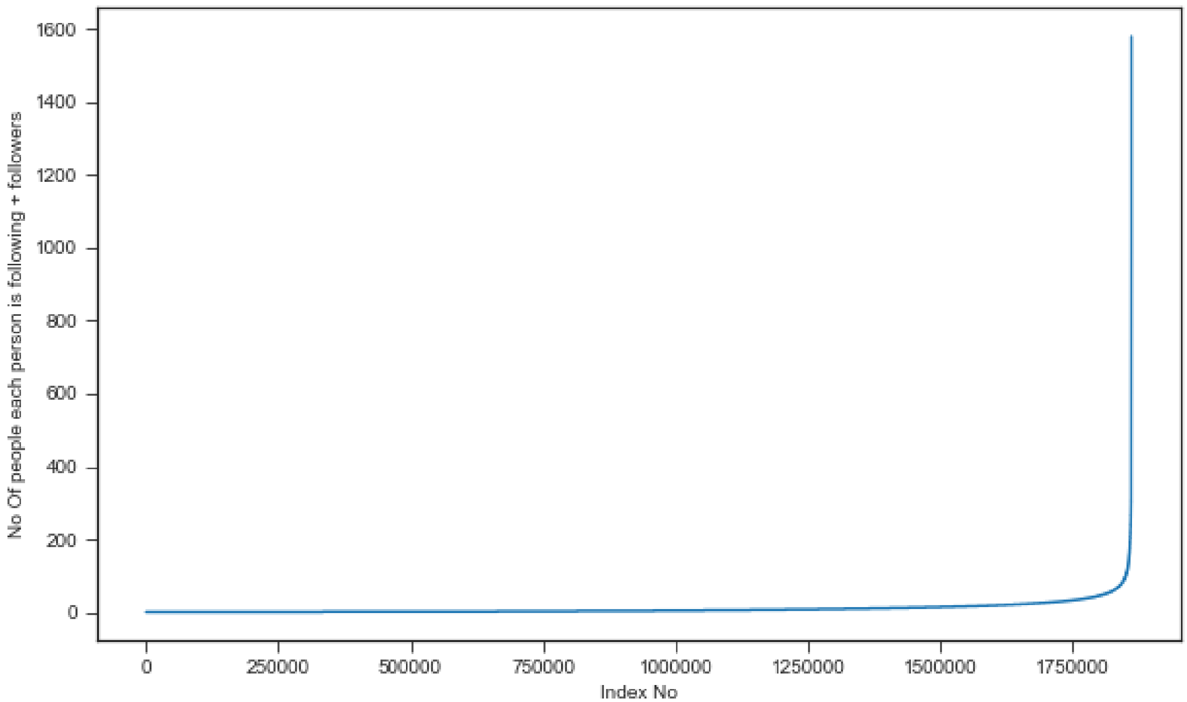
In [0]:
```
count=0
for i in g.nodes():
    if len(list(g.predecessors(i)))==0 :
        if len(list(g.successors(i)))==0:
            count+=1
print('No of persons those are not not following anyone and also not ha
ving any followers are',count)
```

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

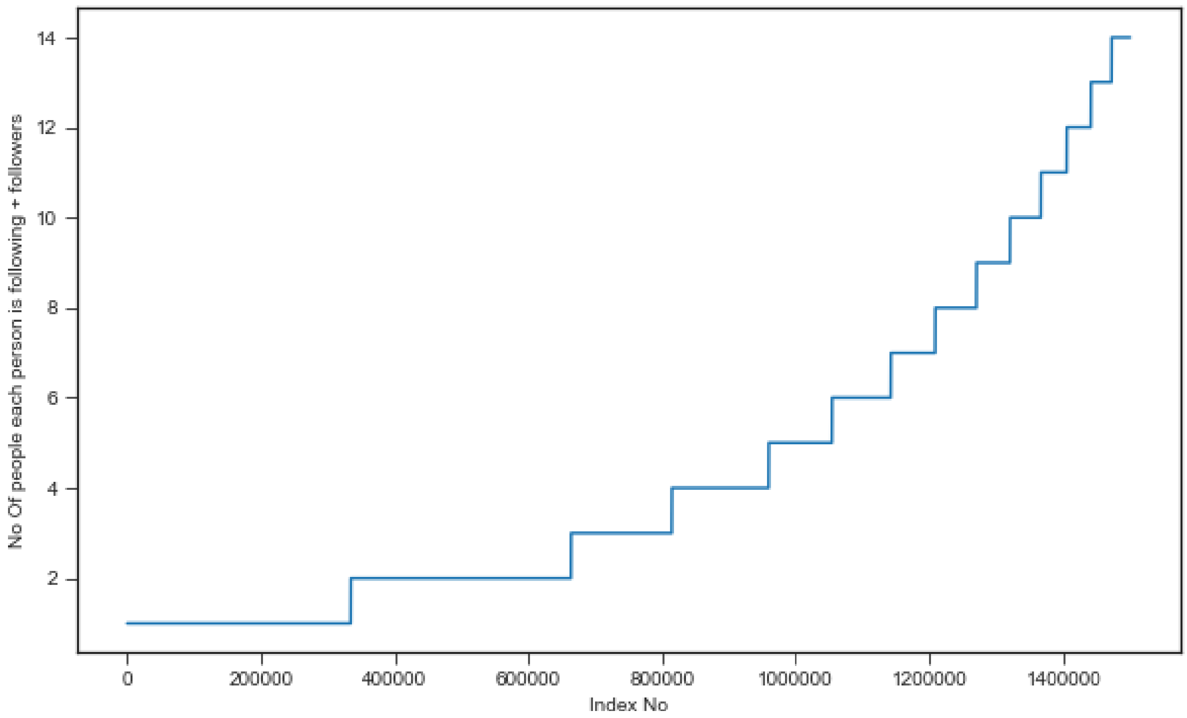
1.3 both followers + following

In [0]:
```
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 [0]:
```
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 [0]:
```
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 [0]:
```
### 90-100 percentile
for i in range(0,11):
    print(90+i,'percentile value is',np.percentile(in_out_degree_sort,9
0+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 [0]:
```
### 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 [0]:
```
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
334291  persons having minimum no of followers + following
```

In [0]:
```
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 1579
1  persons having maximum no of followers + following
```

In [0]:
```
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 1320326
```

In [0]:
```
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. 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 [0]:

```
%%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_fina
l.p','rb'))
```

```
Wall time: 5.08 s
```

In [0]:
```
len(missing_edges)
```

Out[0]:
```
9437519
```

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 [0]:
```
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.isfile('data/after_eda/test_pos_after_eda.csv')):
    #reading total data df
    df_pos = pd.read_csv('data/train.csv')
    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('data/after_eda/train_pos_after_eda.csv',header=
False, index=False)
    X_test_pos.to_csv('data/after_eda/test_pos_after_eda.csv',header=Fa
lse, index=False)
    X_train_neg.to_csv('data/after_eda/train_neg_after_eda.csv',header=
False, index=False)
    X_test_neg.to_csv('data/after_eda/test_neg_after_eda.csv',header=Fa
lse, 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 [0]:

```
if (os.path.isfile('data/after_eda/train_pos_after_eda.csv')) and (os.path.isfile('data/after_eda/test_pos_after_eda.csv')):
    train_graph=nx.read_edgelist('data/after_eda/train_pos_after_eda.csv',delimiter=',',create_using=nx.DiGraph(),nodetype=int)
    test_graph=nx.read_edgelist('data/after_eda/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 -- ',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 {} %'.format(teY_trN/len(test_nodes_pos)*100))
```

```
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 ar
e 7.1200735962845405 %
```

> we have a cold start problem here

In [0]:
```
#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=['source_node', 'destination_node'])
    X_test_pos = pd.read_csv('data/after_eda/test_pos_after_eda.csv', n
ames=['source_node', 'destination_node'])
    X_train_neg = pd.read_csv('data/after_eda/train_neg_after_eda.csv',
 names=['source_node', 'destination_node'])
    X_test_neg = pd.read_csv('data/after_eda/test_neg_after_eda.csv', n
ames=['source_node', 'destination_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('data/after_eda/train_after_eda.csv',header=False,in
dex=False)
    X_test.to_csv('data/after_eda/test_after_eda.csv',header=False,inde
x=False)
    pd.DataFrame(y_train.astype(int)).to_csv('data/train_y.csv',header=
False,index=False)
    pd.DataFrame(y_test.astype(int)).to_csv('data/test_y.csv',header=Fa
lse,index=False)
```

```
============================================================
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 [0]:

```
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 [0]:

```
# computed and store the data for featurization
# please check out FB_featurization.ipynb
```

Social network Graph Link Prediction - Facebook Challenge

1. Reading Data

In [22]:
```
# Install the PyDrive wrapper & import libraries.
# This only needs to be done once per notebook.
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
# This only needs to be done once per notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# Download a file based on its file ID.
#
# A file ID looks like: laggVyWshwcyP6kEI-y_W3P8D26sz
listed = drive.ListFile().GetList()
for file in listed:
    print('title {}, id {}'.format(file['title'], file['id']))
```

```
title Copy of FB_featurization.ipynb, id 14vRP7TT_WR7HuIivgmKXCQqf7pmmI
tq7
title Copy_of_FB_featurization.ipynb, id 1fzYTkpDffxXQdyjL9APV3-c3gqAXb
IFw
title Copy_of_FB_featurization.html.pdf, id 1ZhETwsR2eBrwjP6QS5bYC5EVY0
sxx_ix
title storage_sample_stage5.h5, id 18GnmIGjvK-nZJthQ7wHzh4gnrEsBte4t
title Copy of FB_featurization.ipynb, id 1PVQfwzOsdfZM4B98HF9pry2Lhl_Fr
EXs
title test_pos_after_eda.csv, id 1RvBNCGWUoN8l4hVDBfL6PPWR8q93mAmC
```

```
title test.csv, id 1C91MGecHnhCVeanoNX0GGyjhsyfam8h_
title train.csv, id 1CyMrF4MphxWnKx4zZ2j2BERMCiekIsP1
title Untitled6.ipynb, id 1HXlGe-wXK_HusVquQ5r0QXwIIaK1X7hD
title NEW_NYC_Final (1).slides.html.pdf, id 16RJKSXz-tnhBZRhSe0PnkOdwVs
8n2WxX
title NEW_NYC Final.ipynb, id 1yYiNr-TfJXmeCCzPjaVDVVHZUZV3FIRz
title NYC Final.ipynb, id 1izDLcAE8FwgbQc7oy1xn_3_4hxFDleJf
title Copy_of_NYC_Final.slides.html.pdf, id 1LZv1D2Hnhd8qZW6yBm9AHmzesY
i-DpGs
title Copy of NYC Final.ipynb, id 1HUc0t022Ke5VEA6fuGfIV3gPNyUFsk-n
title Untitled5.ipynb, id 1EZ31A_LcqQgqRwQWpLunRMHSdPDR2ysa
title recent.ipynb, id 1UQrtl431lJwhSP8Xm4xk0AaoM7iWSjM-
title 01-Numbers.ipynb, id 0BzRnZ30e4bHAUU5MSXhyRjFHaDk3TEFjbU9pMDg3TGx
kb2sw
title 10-Objects and Data Structures Assessment Test-Solution.ipynb, id
0BzRnZ30e4bHAbmlfVzIzRk9kZWdkcWNqZDAtYVhfUmFPQkVZ
title 09-Objects and Data Structures Assessment Test.ipynb, id 0BzRnZ30
e4bHAQ3k1UndiUW1zQXRYSm5VSUFTMGlLcVFtNzVz
title 07-Sets and Booleans.ipynb, id 0BzRnZ30e4bHANGNHWXlKUnJMOVhlTVUyR
FIwRWFLSi0zT3dr
title 08-Files.ipynb, id 0BzRnZ30e4bHAY1B5SnNrbThiZ3lVSU13SEViX3l2TXE0L
WJN
title 05-Dictionaries.ipynb, id 0BzRnZ30e4bHAWG42UVc3OXlzYlB4WHFBazF1Z2
1CRFpFZXZR
title test.txt, id 0BzRnZ30e4bHASlRKVURBOTZjZFFqS09sbUQwUTlHcEUyX0Nv
title 02-Strings.ipynb, id 0BzRnZ30e4bHAYTFFcnRfNmk4TmRlYURUM0xKbGswa2h
VcUlv
title 03-Print Formatting with Strings.ipynb, id 0BzRnZ30e4bHAcXpQS3Q3Z
W1hTk5PRXBMMkZCZzN5aVN0Tjhn
title 01-Variable Assignment.ipynb, id 0BzRnZ30e4bHAak8tUkFCYzA5TmZqV1d
aTmVvSVRVcUlFVUhZ
title Courier Service Project.zip, id 1DrPCSZUFsok_pWiSewcgpxpkWMqLY6v-
title 09 Amazon Fine Food Reviews Analysis_RF.ipynb, id 1xDCafQwplfxU3P
GaX7LWpT-qF8qTgE4C
title Live session documents, id 12CfMovckYuSF1Jh25AfSBOmQcebo7f_Z
title 03 Amazon Fine Food Reviews Analysis_KNN.ipynb, id 1x8_Si_bLwpmgx
MQf-Am3PAYcUV4auLvb
title Scholarship Test Topper List, id 13-eP0dFDUMzOa-Hme0PnC9TQ80GkSpT
6cIh1RgzNbP4
```

```
title Group_2__quiz-5cc76b4e8f8c7-1316619-1.pdf, id 1xKSmWvPzkCOkGLtDSp
4O300CqQhyTQVn
title Copy of STACKOVERFLOW.ipynb, id 1ks766ySDgMhAYPCq0wgcHMB88pKZzOI4
title Classroom, id 0BzRnZ30e4bHAflV3TjEyZGx0RkhyblBxcnRUUnplaG8wMXV3cU
xyRUdwQ2lWNkpQS0dfY1E
title AppliedAICourse: Pluto Applied AI Course, id 0BzRnZ30e4bHAfjg3eUd
2d1NBbngzd2hSTVQ3blhOR0hWWl8taVhXRm5acjZJTFFtVXpTM3M
title StackOverFlow.ipynb, id 1qRF8ohBBMnMbONY914ZqaVQRR-UXUy5j
title StackOverFlow.final.slides.html.pdf, id 1DRVPJpCsN_fUZyJVGMXgEBUK
1JsSlba-
title StackOverFlow.ipynb, id 1z79FYl-rdlkf93Chz4N0iTTf1jgp-XeG
title Copy of Copy of STACKOVERFLOW.ipynb, id 1gIg3qj_A403MKUnLS9bHIXGX
a4CyeMmu
title STACKOVERFLOW.ipynb, id 1LZoBtxuYbHL4QUMjRgsx2AQTOP5NkIXO
title SampleSubmission.csv, id 1p7LZdkkrvTl-wjEXh7JpXOU-qTQeHAbl
title Untitled4.ipynb, id 1v0d9xvXc6GSkZqpXEvKq5LvuByD3Xphp
title Copy of STACKOVERFLOW.ipynb, id 1mP6Qt195jT4fgoEvGWOPHd_MMGkTjmnz
title Resume Format_smritika(1).pdf, id 1l23z4BA1Pp8QJKdwEgLTPGb_p2BC95
Qo
title BetterPlace Calling pitch - BP Data, id 1NqKpm5bUT-uPF-2ZY0AWGDqU
10WuTBwLM4GlU9hwkPY
title Untitled3.ipynb, id 1p-5ufBaIat-OvNi3_EjyqmyytiLSukJR
title Assignment_SAMPLE_SOLUTION.ipynb, id 1j0xJr80XlDZkOKNf14nvPZxArc9
bo7zm
title LSTM_IMDB.ipynb, id 1iWpQBiZO95pfOWLdaG6qKwA27d_Q-EDg
title 3.Q_Mean_W2V.ipynb, id 1UIKqDRvSXUiu5lk6AyNbdEtFclvb7um2
title FB_IMG_1507474616935.jpg, id 1_DXBcMkOXTDBC357SMOAMW0_ebGnysPW
title Resume Format_smritika(1).pdf, id 1KCvMA5P0ltdNVWMMOiK40j7rV1HX8M
QQ
title Anubhav Jain Notes, id 0Bx7x8HmM7p_zM3VDMURfWWdvY28
title Keras_Mnist.ipynb, id 1PMyn9bgjDzhaBm2NO3YQ1f0m-Y2SmZe4
title Untitled1.ipynb, id 1szc-lUM6WUS_BxRKIy9tJFUKb5dwzSXt
title Untitled2.ipynb, id 1AP4zTJQzRYOmHJHlLWARgjQu3dzHqAhp
title Ozone_Layer_21_22.ipynb, id 1Lb549aT8wWRcbp1ncK_seCs5Nwcu09aJ
title FB_featurization.ipynb, id 1rQgKqdUvwGvHpWUYCs1b9sIRGYgsQJT4
title ca_tech.csv, id 119YqHBF237R9RXGDWyGND8Z5ASaWsYnS
title File_IO.ipynb, id 0BwNkduBnePt2T2tNaU9nQlJhNkk
title 4_recursive_functions.ipynb, id 0BwNkduBnePt2dVFXaE9sVUJqdnM
title Untitled0.ipynb, id 1UG_gg13mhSEEh94-wSKz55eophzzg__0
```

```
title IMG_20190321_225308.jpg, id 1OEa8wrEdLDxVPlvLYQppwCQU_p04kktB
title PersonalizedCancerDiagnosis1.pdf, id 1B4IuEgVNia3o8-NUE5XiV86TpD2
Y93oO
title PersonalizedCancerDiagnosis1.ipynb, id 11Xs18vQPDlZC4yhC98DcZ_2i1
uNCPIoQ
title Copy of help_PersonalizedCancerDiagnosis.ipynb, id 1fHM5939Wyj0bM
Zzbs98nZoYcfd40vUtd
title Copy of final_PersonalizedCancerDiagnosis.ipynb, id 1eEOXDTlciDhC
db_t_R3_4cYhOR_wnuvw
title 11 Amazon Fine Food Reviews Analysis_Truncated SVD.ipynb, id 1js8
rSiM3f5QVsYaEhZqXlqSoGhheBQkA
title Copy of CV.ipynb, id 1emUvhc6sbBaGPlbgVN0ygWrlNy7OTwOY
title Copy of PersonalizedCancerDiagnosis.ipynb, id 1g-JH1-5WX6FsuKmv4a
TJHVO9bmoOF9oy
title kfold.ipynb, id 1dxzNFcqBLq-RpFa2vLB8NBBRIvTnMwWN
title Copy_of_LSTM_AMAZON_FINAL.html.pdf, id 12kg-Xli2IL5DdrU3jYlngFUuE
1gzNjOO
title Copy_of_LSTM_AMAZON_FINAL.ipynb, id 166NxYaJMzdmkojt9hygcvDTOCPdd
ZuZj
title Copy of LSTM_AMAZON_FINAL.ipynb, id 1WZhmVO_ZsZVzvrukZVofGJnzc4dL
YxL6
title Copy of 3.Q_Mean_W2V.ipynb, id 1I0cQvgG6uJMTv8-DSRQfVZgMcCB6bE5Z
title NYC Final.ipynb, id 1TexiXFhROP0mjbAyk8Rhj-IByqKp7tyv
title STACKOVERFLOW.ipynb, id 1K3R4bGBu90J_Hd2Lo3cXnFQ72FEXmaDM
title STACKOVERFLOW.pdf, id 1RhJ50Vk7j6QIXlfE7QMSrIm4jEvmKelZ
title NEW_preprocessed_summary.pkl, id 1BezaMKIVoBpz40X9UHpq2ohH7uj1_5E
2
title NEW_final['Score'].pkl, id 1n1D2IeNZS7BQoMYNpjtjp_HsUMuhgOI4
title Data Preprocessing.ipynb, id 1f7da-AONEOx3LW31fZvsxTle3iYRQNly
title Data Preprocessing.html.pdf, id 17G3PHiNtF-7l_AITfUNrzMSM0eXD1LZn
title CNN_DEEP.ipynb, id 1eYke87xu0KjePqilAOQyC28Ffuz-nUxl
title Copy of LSTM_IMDB.ipynb, id 1VVaVZMSCbxJzsqPQUdi3GSS9a6gAcHdM
title Copy of LSTM_IMDB.ipynb, id 18Y4EFnwwZy8tMElbSlyb0bGBt8XwyMSg
title Untitled1.ipynb, id 13-MQNl0eiofE5ihV5-QM3Ac-xJZKIAYu
title CNN_DEEP_1.html.pdf, id 1Y50U1Rjh3FxPiXJuu1HSpd5R6_cmIjAi
title CNN_DEEP_1.ipynb, id 16_FqhC0CHOHr0Q3c3-lbIHuGO1crs9r9
title Colab Notebooks, id 12zoKZxGfCC9qhcr7kOrbqIOvRc3oP8J-
title Nptel reg 3e, id 1yAqOBgH_-XNRz-cFUJUQ0KuXa3NSHnCmoej2DRraZy8
title Keras_Mnist.ipynb, id 1PCMSUcrqWRrpg4RwUIJ8bvYR56_L_LY8
```

```
title Keras_Mnist (1).html.pdf, id 1anUb2bbRwzl3SmqZ5TLXIhtOpbp5t0NB
title Tensorflow_Softmax_Classifier.ipynb, id 1cYvTo2mcBpBU2XIihDhbDUGr
frkzwgCp
title pandas_workshop.ipynb, id 1jfB5pzoBltXFURnnCRyT7Umm7eiNGD_X
title pyplot.ipynb, id 1bIFGnkq0r7aMVn3wqx7RBVpkmnv-_qCz
title 11 Amazon Fine Food Reviews Analysis_Truncated SVD.pdf, id 1XNIa2
y1aUZrpQpoqWyO74jxjPuEK8vdF
title 11 Amazon Fine Food Reviews Analysis_Truncated SVD.ipynb, id 1Pa3
4FMXMdRxv4XxQaP35t6-nsT3tJcQd
title 10  Apply K-means, Agglomerative, DBSCAN clustering algorithms on
Amazon reviews data set .ipynb, id 17DLeSuLLmT1RMoToVUiGjefbltxBEIz3
title 10  Apply K-means, Agglomerative, DBSCAN clustering algorith (1).
pdf, id 1sL2os7EJ4fa7ccCsG5Wc1CpO8AdcqqwI
title 09 Amazon Fine Food Reviews Analysis_RF_1-Copy1.ipynb, id 1_zlypa
7ijjOF3K_0LvLsHdnKmnP1dscI
title 09 Amazon Fine Food Reviews Analysis_RF_1-Copy1.pdf, id 1CL8on8Cf
kU9jgoJi958QRO21oUP2oZjH
title TFIDF-XGBOOST__3DSCATTER.svg, id 188wakGOgCxyUfWWZbm24tnhMWmLLOhj
e
title TFIDF-W2V-XGBOOST__3DSCATTER.svg, id 1dPXfxcCRFiTDvWDpzMcib0CNXcz
q-OEE
title TFIDF-W2V-RF__3DSCATTER.svg, id 1Q95fpe2O2_D01bvwX0vo2fTrPEEoHsil
title TFIDF-RF__3DSCATTER.svg, id 18qMF7RisWcbf4iigWFEVTMrMtEfo46oc
title BOW_RF_3DSCATTER.svg, id 1wnNNGAWeVr_EtfDzDOuBiNEHRgyjth80
title AVG-W2V-RF__3DSCATTER.svg, id 1m4Jn1WGSDLqF2Qebe2gwnQkrJtq-co1r
title AVG-W2V-XGBOOST__3DSCATTER.svg, id 1vMk-qpUgT5JW9JJB8HneF5d3bw1nN
XWs
title BOW-XGBOOST__3DSCATTER.svg, id 12mg-lhzFnKjlWDSPbwwm0iEaoZ5TJ66M
title Quora, id 1OWZoiQDvAvgOa-IUnEQ-6QKSEp_pw1XO
title 08 Amazon Fine Food Reviews Analysis_Decision Trees_1 (2).ht.pdf,
id 1IxYFf44padAlV3IMRS7dT7WmuayWAple
title 08 Amazon Fine Food Reviews Analysis_Decision Trees_1.ipynb, id 1
mu8moIzTy1H3neBZFLrCf7poqRN2mwPJ
title 07 Amazon Fine Food Reviews Analysis_Support Vector Machines (1).
ipynb, id 1OwPBYIc03ywnLCnOZZLR3_rxZYJoqPem
title 07 Amazon Fine Food Reviews Analysis_Support Vector Machines (1).
pdf, id 1BNYyd0x7jDBflQV_IVLuDDqUU3TYRuQD
title AAIC-Win-V1.1.zip, id 10KRP2CkGRSYNyqpevjOM0sOD16f9NPGi
title 06 Implement SGD (1).ipynb, id 1X9KZ92401h6Ul4QlE3NOMZuxx5OTrMQ-
```

```
title 06 Implement SGD (1).pdf, id 121oUP-ih_MDwNTkHn3r_2TrGT79bqyZ_
title 05 Amazon Fine Food Reviews Analysis_Logistic Regression.pdf, id
1o9mQOtPFNMZrXC-uq1PCPSk97y_ApeGr
title 05 Amazon Fine Food Reviews Analysis_Logistic Regression.ipynb, i
d 1aTHg9JLRYb1IDqrMHqxkVnoSIvhBATEP
title FB_Models.ipynb, id 1HI7l4Bxi-Mi5TzeQmiShPDhs34F2LDFN
title data, id 1pEBmIl1tbuwYrsfeUv2KDz8a2gR0Fjxe
title key_operations_on_dataframes.ipynb, id 1whrFZLHtk5UzA7UqF_ri4aq_q
QTeA_HG
title 05 Amazon Fine Food Reviews Analysis_Logistic Regression.ipynb, i
d 1z97wn38jg3gBsMPq_iDs-hEkzGv-vGI4
title 04 Amazon Fine Food Reviews Analysis_NaiveBayes.ipynb, id 1Qf0oOL
PYZAI7L7D2o_DmWt-dPiPkln6X
title 04 AMAZON_food_analysis_NaiveBayes-final-checkpoint.ipynb, id 1hb
yX138M03S6TihKauHM9EX9Q7SzXwKo
title 04 AMAZON_food_analysis_NaiveBayes-final-checkpoint.pdf, id 1o0SZ
ldEZjT7UBkU3BxFXOgLvxAP2JXgD
title debugging.ipynb, id 0BwNkduBnePt2Z2taeGk2RVdhb2s
title 03 Amazon Fine Food Reviews Analysis__final_KNN.pdf, id 1XZJL8cQ7
PXERsxfBPV_2DCHQZ32irOVw
title 03 Amazon Fine Food Reviews Analysis__final_KNN.ipynb, id 13NP4JX
MnoAhmve_27Gt2tCIBOau8EWE1
title numerical_operations_on_array.ipynb, id 19Q1gumQGT0Jjf-y0NrND83SM
Gg0cGoLt
title pandas_basics_practice.ipynb, id 1udp0rlzbne9LDKtkPnHJcqUqPUK-G5M
N
title 08 Amazon Fine Food Reviews Analysis_Decision Trees.ipynb, id 15c
P3oZd2OPATj804ZHULfx_2NBy9WIZh
title 1_introduction.ipynb, id 1m1F5IpLeYDhTAnLg9uhJiU12TVv7TR7E
title 06 Implement SGD.ipynb, id 1753az2dtiVl3iiOJrLEiPBjre7TM_51D
title Assignments_AFR_2018, id 1WdBrvwMfr73lOhTIfppBLbTPuC2ou6HA
title 10 Amazon Fine Food Reviews Analysis_Clustering.ipynb, id 1SHme85
9DWtyIXp5iTz5JV1-aZaakhQIR
title summary.JPG, id 1engLQJlLgjCg0bkb2wP4ydwsnUMV1sIz
title 3d_plot.JPG, id 1NVCtTskYQryEmcImGHv6cn3T8M8cDRSZ
title train_cv_auc.JPG, id 1rDjCOgxjFAEmUM_YesVBs-SnLvRLSUOp
title train_test_auc.JPG, id 10Dvk8lyD-os6zuCJMBhVJMF-ep8whhXq
title 3d_scatter_plot.ipynb, id 1ySr629keT65HuMZTsIaDTSMyvSzpQnAP
title heat_map.JPG, id 1rkIye_gU5b1XZ-Z8LQmTdF_tQMKCpt4M
```

```
title 07 Amazon Fine Food Reviews Analysis_Support Vector Machines.ipyn
b, id 1VeDynyDnpAGcF9QmGK3GS0Up-eAvCZrw
title confusion_matrix.png, id 12SUGWOUtMrWPKWAUQ0SNjXyswl-ktGKM
title function_types.ipynb, id 0BwNkduBnePt2MkpUTDZ5RGlqRTg
title for_loop.ipynb, id 0BwNkduBnePt2LWs0QVpkOXFJdEk
title 2.6.ipynb, id 1YIYM42TTeHvmTlAZuHFHX7VA_VSVqoJ7
title break_continue.ipynb, id 0BwNkduBnePt2VDZBT2FicjJBdDA
title 6th Semester, id 1PW6B4s4h0Us3KHNDiVX5ys7unREa_Sbu
title Functions_Introduction.ipynb, id 0BwNkduBnePt2RTR5VTRkYTBaemM
title AMAZON FOOD REVIEW.pdf, id 1TEOYdfDadz2dTKs2qR0aUDEJPfKnUcWy
title AMAZON FOOD REVIEW.ipynb, id 1iYUw6sen-TYJ4Ec4Fl7Kl1Sk3eSnJbNj
title 2.1.ipynb, id 1mLKBK3MqjGzW--IRjW9t2Bbxb7y_jFXw
title 3rd Year Notes, id 0Bx7x8HmM7p_zVUp5bjBwTC04cTA
title AmazonFineFoodReviews, id 1hjjSgfbbP1Qqjw6Ie2qkVaAfZEO8oOF1
title data, id 1c50Q5RcmdpMYj1jCPc3ShOE2y4G8G2ez
title FB_EDA.ipynb, id 1pexG21CM7isPdu82KtoZdv78GNu_m2ge
title EDA.FINAL.ASSIGNMENT.ipynb, id 1agH6v4TDmFgCLBLY2UVjs8Q_gbOrE5Ww
title CNN_MNIST.ipynb, id 1I5kcAaQKEx0IwUNQvZdYkctwCcWFf81N
title EDA.FINAL.ASSIGNMENT.pdf, id 1998p6O_cPUecH2gO1nJXAkWEXN98dzFq
title Exploratory Data Analysis..ipynb, id 0BwNkduBnePt2eU5JdEN0VXR4VTQ
title 13.10#14.9#14.10#15.7.ipynb, id 1JLsIJKC4k79aMO_o7BPx_4jsYLkbSdf2
title pandas_basics_practice (2).pdf, id 1RiD91v8Q3PXqpJuymwR1EuBPjpvHJ
f1s
title pandas_basics_practice (2).ipynb, id 1NqhCaWCSU3tHOwlPTb5EUHesycg
9nQUN
title 1_python_practice.final (1).pdf, id 1uLhFU-rfpUmgNULbbVYIiaAsd5GY
OHkl
title 1_python_practice.final.ipynb, id 1ErFkNwbki5Ikwlw5PojsjyLNPui83f
Ko
title ca_tech.csv, id 1kMlQ5AOSz7Ld-ar08H9oy3xPEMKo1I96
title train_pos_after_eda.csv, id 1XLHsIRXKLx9TA9nuC1SS7JDkLyRVmo69
title train_after_eda.csv, id 1lcxzVZ0-MkPmoH3lS35Q8rRfrecKSXb1
title test_after_eda.csv, id 1_KN7S8zfHdrkRjRYOEtBxBVq8JrGxPXD
title after_eda, id 1ONG0P5YAMlkzyKB7VugPKtLs9fXcdmRx
title storage_sample_stage4.h5, id 1fDJptlCFEWNV5UNGPc4geTykgFI3PDCV
title storage_sample_stage2.h5, id 10qJ04GRcaDxc16gmJXb8rpGPmlyys7E2
title storage_sample_stage1.h5, id 1pIO_nOg9XU0WUD10brRvrgyUXbY5gqMs
title page_rank.p, id 1hp-5BFw9xK1WmovBW17T5POa1BbGSk9N
title hits.p, id 1YVVHZvqfopWwLeAdIu5-oHGocS7CmCZR
```

```
title fea_sample, id 1qYtDPghLMT6rv3xd7NmQUSUKWwCS5375
title test_y.csv, id 1H6qybuXr8i_USWu3k3ulXEOurc-SElUh
title 4.ML_models.ipynb, id 1YQoysuPgCDJ_HyOuclAYMVd04JNAjzP6
title 2.Quora_Preprocessing.ipynb, id 1U-U8T6qgWCetk6vwmtKeRmoKsFdzP_L_
title 1.Quora.ipynb, id 1Kwyq2dYfA_E-3b0S_8Ek17vRQJFkRv6U
title 3RD YEAR STUDENT LIST FOR BANK BATCH.xlsx, id 1awDUnHz1fzCkE8Dasb
IdjIpdPJEeV2NDEDU5wwb75Hw
title PersonalizedCancerDiagnosis.ipynb, id 15U53IwVyymiCpvNRCHDD1Z73lc
g8vyDL
title Functions.pdf, id 1hQCZEVpM2-U9hPBCIDDikLHD3MKYKC6U
title installtion.txt, id 1W2Llff_rJowzWxqDRZP4U7S2pfDrx8Ta
title AAIC_notes_ashu.rar, id 18xBSvh0TB2d1DvYThyp0bMem9mkkx0gC
title 12.1#12.3.ipynb, id 1Q0_LoDMhaJQ-grEJgIKYiLYFMCp3YtXv
title AAIC Syllabus .pdf, id 169GwUXBxjnim7lKL_y8f_2CYUQrzBgch
title Data_Notebooks, id 1okdoeVcz6peAgJRrBC1Hnx1o7zD3OhYE
title OS_PostMidSem.pdf, id 1Y7ZfqPj3yi81tXUqiYQUZaQbt-uk1K0H
title data, id 149ZLCNqL4WBdp-7Y3eHxcMHHhVvIWRSW
title Distance-0.1.3.tar.gz, id 1aBIjtzXnSuYK_9nOfb715IhkxoLDTvmD
title nlp_features_train.csv, id 1JncN1Fyt-ND_yZXOzqEfcRsYMTKqtu7Q
title train_p.txt, id 19KsWPoIMX3O_FrT4We0I1M7reKvFYKRr
title train.db, id 1A5dzEvSpxXYQ5LKgtklor4jWdDV8y_4i
title train_n.txt, id 1peorE_ke1MoaZCkrNQpap1ozJ15fuk2s
title df_fe_without_preprocessing_train.csv, id 1gTfCTD3fz-3NJnfYLm59nZ
FN3WC3fzfD
title final_features.csv, id 1c7Ffo1Go1dtUpKcSWxdbdVyW4dfhEoUp
title lr_with_equal_weight.pkl, id 1bA0Rz69N4wTyxMjjcT9Da2hxkpuwCYup
title Titlemoreweight.db, id 1S_P2E4DxDAwd15YFMgwlvT3NVlfeYqXS
title Quiz5_solutions.pdf, id 1W2dn6wopDSqw6GMoH_5Wtfbr9cOzVz31
title OpenedFilesView.exe, id 1QE1DhnAvR6QxWtjoVf_PBLEIFV0HXk_1
title lr_with_more_title_weight.pkl, id 1BJlNz9HI2OE9w2DjRhVsE8SWuqnbIr
_5
title tokenaizer.JPG, id 1R-DSx3evjyP0FSLdQJenLuBDzjVKUYev
title tag_counts_dict_dtm.csv, id 11N9uTIotaNDpmJ4KmjCHN7YSD3VNMvWV
title train_no_dup.db, id 18tA34r3269sybix_jsrDPnWchaaHXpF7
title Processed.db, id 1MUAVbg0jinwAGi9zwLJDo1K1wdRXKCFB
title CS30002 - OPERATING SYSTEMS, id 1gFCSaiajLFi8S5fPGyrN2TJY8vNn-cdQ
title Exploratory Data Analysis..ipynb, id 0Bxil-CLuWQJBb3BGSDJlSUxBZ00
title C2 OFFICIALS REGISTRATION, id 1uz7_W_wAZocaV1jRleYLoQgORZtRZyseFM
cKIWt1vJY
```

```
title yellow_tripdata_2016-03.csv, id 12hFPRHhGAFZk8eF-WssicyX6OPUriSYR
title yellow_tripdata_2016-02.csv, id 1bWdNt9F3ZakZ1-ZPzGUA7QCGzBS49yBL
title yellow_tripdata_2016-01.csv, id 1zfDwQmNyZUzkVhRys5j09uVk9Fwyv3if
title yellow_tripdata_2015-01.csv, id 1kcIZlf-LQiQhqfSCZb719Nh6Rqkp2zKK
title LinearRegression.ipynb, id 16yIAGm-A61vEgcnmmNY5rWCWFCDQ6Y-k
title 1.6.ipynb, id 1likLzJXlwPNUxPE85b5oooJMWfcU2iHC
title 1.3.ipynb, id 1T2dftEJbsB_HSsnhkvXaUozLb8bPdsu3
title kfold-checkpoint.ipynb, id 13eAxP2tvcVrFBpfCO1LRhs-_DrSzKW4O
title knn-checkpoint.ipynb, id 1RtFnEHmFPWa8CzlxRWF5We5REkbqiCE1
title meshgrid_image.png, id 1P-C4SIUcr_3_D4hzfYGVGXjk4K6b4Jkd
title 8.twospirals.csv, id 1ewnifP-kp2JzooMxI5gZ0zj_oHsDpDcc
title 6.overlap.csv, id 1J680snH4wMxkNcBzl4MYAujJK3PVz3TP
title 7.xor.csv, id 14D8l-5lAsaGzwms5cTa3h8JWIKPsSdL_
title 9.random.csv, id 1qPXhymerAJJVbQPI-836q8KdTSwsgMzM
title 2.concerticcir1.csv, id 1Hw_v6QMHTn3Yt36W9CMEZgfmdLSQr72R
title 4.linearsep.csv, id 18I8-z9tZ1AgFudKwGnChR0AYHmH5zpcb
title 5.outlier.csv, id 1cQhGaqNfvMDV2qto0AjNCo7_JS46SxGP
title 3.concertriccir2.csv, id 1B_vPYznJm2c25CI9z0pnMTrndXFL9WJv
title 1.ushape.csv, id 1Jkm5tuACkUCGvrFJkCmfL_3DxAdPqsdL
title demo_data, id 1fNa1_YC1KyNs1o3pGXvhSExzRY9kZGCj
title .ipynb_checkpoints, id 1frJhKK103UuK9ThBM-HZ9BDVYjOyjL7P
title knn, id 1tMYRWzbrSMxQ7aQ5mc8Qf4190gPSt5fy
title 1.2.ipynb, id 1fHpH3ExDp2CxF5xUVSCr1S9iiO2dDYKT
title pandas_intro.py.ipynb, id 1GGNDSe0HeBpI4P92rKm9VWAnpj1-pXYE
title nyc_weather.csv, id 1KxwFsL6IF7OD_XN28kjxl0-amnELIhZ8
title weather_data_cities.csv, id 1mgPjvAdzUZ8aSnClcMRplPLZqDDPpW98
title new_noIndex.csv, id 1dAU8ydmikaY0zGXyoIsonW_bsspA92ik
title new.xlsx, id 1GVUFZOm7KSi8oE5S9YuqMFnaC2uT5mKy
title weather_data.xlsx, id 1KyYZ1QdQhYKIiwKpiXXHnD8TJkqPfVpu
title new.csv, id 1qfsqE2zOPTZCAG4ZygNblGr_xilAbTg2
title weather_data.csv, id 13vsT6FkqHDk11l5OionNeI5N-R1bQp09
title weather_data.csv, id 1B3C9UtZxx4Ogm3a-0QNA-5fJ7K67Pav_
title 1_getting_started, id 1fmbS5hnkDGPIMbqZs2KJDGf1DMp7slkI
title 2_dataframe_basics, id 1k9lKOUrtLR9kuNp_lpxJm9npCKgy4ynb
title 3_Key operations on Dataframes, id 1WagO0hXnJfS6B0jXTW6xA-wPi38ls
TTf
title 2.5.ipynb, id 10umA6Gosybexu3LDUGP2zYYWz9JiXB8y
title 2.2#2.3.ipynb, id 1WSE_arH4KULmAZextxnRXUkTY6zcZkEu
title 2.4.ipynb, id 1d1UtrcV2N48h4JKteWWtsiwiP9XD1t3B
```

```
title Code, id 0BwNkduBnePt2R3poZHB0U0lRYVk
title Debugging Python, id 12ySNcd7CkmWhlnGmKGFUtvU2cZw9qmUz
title File Operation, id 0BwNkduBnePt2Z2xpYkx0UF9FcHc
title 1.4.ipynb, id 1RU5hnOMKjZ6h8wyC6or_C_bX2P24Y_he
title 1.5.ipynb, id 1jfpLNJeKnsuPj-hc7OdUKfeAXb0msclD
title 6_modules.ipynb, id 0BwNkduBnePt2MXR6Szg4YzJuQWM
title 5_lambda_functions.ipynb, id 0BwNkduBnePt2V2xmLTNPb2hGZjA
title 3_function_arguments.ipynb, id 0BwNkduBnePt2ZGZpY2dPdDZMWGc
title ExceptionHandling, id 0BwNkduBnePt2WmhnQi1ob2hObDA
title Tutorial 1.2, id 1cHwIv_Ul8xQytcOQ7MWSwbn5V3QzLZmWOoRFTpyPAVQ
title Tutorial 1.1, id 1luBvvZR8R4BYgQ4c_dCno9vLTS5buMylTu_qSd9slMU
title packages.ipynb, id 0BwNkduBnePt2LUxPZzdEN2ozbDg
title While-loop.ipynb, id 0BwNkduBnePt2Y0l4dmFBaTBYVE0
title 2_while-loop, id 0BwNkduBnePt2cGdHQUhoUTBoU28
title 4_break-and-continue, id 0BwNkduBnePt2bmttUWhHYzBHWnc
title 3_for-loop, id 0BwNkduBnePt2UGRsazE1MmRxWGs
title 1_if-else, id 0BwNkduBnePt2NXVSdGlvME1hSEk
title Game.Of.Thrones.S01E01.mp4, id 0B1g2CuAPh9GWM2pCZjduRTY4Y1E
title Jio_Service_Upgrade.apk, id 0B8CSShK1LLMMZ1gtdTJxb3d4UEU
title Dawn 17 AW zr cp.pdf, id 0Bzl4dh3CMTE2bTE0MXROSGQ5T00
title week1.pdf, id 0BwYT18N5TMCiUG01blFmRlZWZms
title Student Management System.rar, id 0B7wAukYfGKBkRnI1X2V0enZUZmFIWj
BhLUhPY1JwZw
```

In [0]:

```
download = drive.CreateFile({'id': '1fDJptlCFEWNV5UNGPc4geTykgFI3PDCV'
})
download.GetContentFile('storage_sample_stage4.h5')
#title storage_sample_stage5.h5, id 18GnmIGjvK-nZJthQ7wHzh4gnrEsBte4t
#title storage_sample_stage4.h5, id 1fDJptlCFEWNV5UNGPc4geTykgFI3PDCV
#title train_pos_after_eda.csv, id 1XLHsIRXKLx9TA9nuC1SS7JDkLyRVmo69
```

In [0]:

```
download = drive.CreateFile({'id': '1XLHsIRXKLx9TA9nuC1SS7JDkLyRVmo69'
})
download.GetContentFile('train_pos_after_eda.csv')
```

In [24]:

```
if os.path.isfile('train_pos_after_eda.csv'):
    train_graph=nx.read_edgelist('train_pos_after_eda.csv',delimiter=
',',create_using=nx.DiGraph(),nodetype=int)
```

```
    print(nx.info(train_graph))
else:
    print("please run the FB_EDA.ipynb or download the files from driv
e")
```

```
Name:
Type: DiGraph
Number of nodes: 1780722
Number of edges: 7550015
Average in degree:   4.2399
Average out degree:   4.2399
```

2. Similarity measures

2.1 Jaccard Distance:

http://www.statisticshowto.com/jaccard-index/

$$j = \frac{|X \cap Y|}{|X \cup Y|}$$

In [0]:

```
#for followees
def jaccard_for_followees(a,b):
    try:
        if len(set(train_graph.successors(a))) == 0  | len(set(train_gr
aph.successors(b))) == 0:
            return 0
        sim = (len(set(train_graph.successors(a)).intersection(set(trai
n_graph.successors(b)))))/\
                                    (len(set(train_graph.successors(a))
.union(set(train_graph.successors(b)))))
    except:
        return 0
    return sim
```

In [7]:
```
#one test case
print(jaccard_for_followees(273084,1505602))
```

0

In [8]:
```
#node 1635354 not in graph
print(jaccard_for_followees(273084,1505602))
```

0

In [0]:
```
#for followers
def jaccard_for_followers(a,b):
    try:
        if len(set(train_graph.predecessors(a))) == 0  | len(set(g.pred
ecessors(b))) == 0:
            return 0
        sim = (len(set(train_graph.predecessors(a)).intersection(set(tr
ain_graph.predecessors(b)))))/\
                                 (len(set(train_graph.predecessors(a)).
union(set(train_graph.predecessors(b)))))
        return sim
    except:
        return 0
```

In [0]:
```
print(jaccard_for_followers(273084,470294))
```

0

In [0]:
```
#node 1635354 not in graph
print(jaccard_for_followees(669354,1635354))
```

0

In [0]:
```
Preferential_followee
SVD_followee
```

Preferential Attachment

In [0]:
```
#for followees
def Preferential_followee(a,b):
    try:
        if len(set(train_graph.successors(a))) == 0  | len(set(train_graph.successors(b))) == 0:
            return 0
        sim =  (len(set(train_graph.successors(a))) * len((set(train_graph.successors(b)))))
        return sim
    except:
        return 0
```

In [56]:
```
print(Preferential_followee(273084,1505602))
```

```
120
```

In [0]:
```
def Preferential_followers(a,b):
    try:

        if len(set(train_graph.predecessors(a))) == 0  | len(set(train_graph.predecessors(b))) == 0:
            return 0
        sim =                        (len(set(train_graph.predecessors(a))))*(len(set(train_graph.predecessors(b))))
        return sim
    except:
        return 0
```

SVD features

In [0]:
```
#for followees
def SVD_followee(a,b):
    try:
        if len(set(train_graph.successors(a))) == 0  | len(set(train_gr
```

```
aph.successors(b))) == 0:
                return 0
            sim =  np.dot    (len(set(train_graph.successors(a))) , (len(se
t(train_graph.successors(b) ))))
            return sim
    except:
            return 0
```

In [0]:
```
def SVD_followers(a,b):
    try:

        if len(set(train_graph.predecessors(a))) == 0  | len(set(train_
graph.predecessors(b))) == 0:
            return 0
        sim =  np.dot    (len(set(train_graph.predecessors(a))) , (len(
set(train_graph.predecessors(b) ))))
        return sim
    except:
        return 0
```

2.2 Cosine distance

$$CosineDistance = \frac{|X \cap Y|}{|X| \cdot |Y|}$$

In [0]:
```
#for followees
def cosine_for_followees(a,b):
    try:
        if len(set(train_graph.successors(a))) == 0  | len(set(train_gr
aph.successors(b))) == 0:
            return 0
        sim = (len(set(train_graph.successors(a)).intersection(set(trai
n_graph.successors(b)))))/\
                                    (math.sqrt(len(set(train_graph.succ
essors(a)))*len((set(train_graph.successors(b))))))
        return sim
```

```
    except:
        return 0
```

In [0]:
```
print(cosine_for_followees(273084,1505602))
```

```
0
```

In [0]:
```
print(cosine_for_followees(273084,1635354))
```

```
0
```

In [0]:
```
def cosine_for_followers(a,b):
    try:

        if len(set(train_graph.predecessors(a))) == 0  | len(set(train_
graph.predecessors(b))) == 0:
            return 0
        sim = (len(set(train_graph.predecessors(a)).intersection(set(tr
ain_graph.predecessors(b)))))/\
                                     (math.sqrt(len(set(train_graph.pre
decessors(a))))*(len(set(train_graph.predecessors(b)))))
        return sim
    except:
        return 0
```

In [0]:
```
print(cosine_for_followers(2,470294))
```

```
0.02886751345948129
```

In [0]:
```
print(cosine_for_followers(669354,1635354))
```

```
0
```

3. Ranking Measures

https://networkx.github.io/documentation/networkx-

1.10/reference/generated/networkx.algorithms.link_analysis.pagerank_alg.pagerank.html

PageRank computes a ranking of the nodes in the graph G based on the structure of the incoming links.

Mathematical PageRanks for a simple network, expressed as percentages. (Google uses a logarithmic scale.) Page C has a higher PageRank than Page E, even though there are fewer links to C; the one link to C comes from an important page and hence is of high value. If web surfers who start on a random page have an 85% likelihood of choosing a random link from the page they are currently visiting, and a 15% likelihood of jumping to a page chosen at random from the entire web, they will reach Page E 8.1% of the time. **(The 15% likelihood of jumping to an arbitrary page corresponds to a damping factor of 85%.) Without damping, all web surfers would eventually end up on Pages A, B, or C, and all other pages would have PageRank zero. In the presence of damping, Page A effectively links to all pages in the web, even though it has no outgoing links of its own.**

3.1 Page Ranking

https://en.wikipedia.org/wiki/PageRank

In [0]:
```
if not os.path.isfile('data/fea_sample/page_rank.p'):
    pr = nx.pagerank(train_graph, alpha=0.85)
    pickle.dump(pr,open('data/fea_sample/page_rank.p','wb'))
else:
    pr = pickle.load(open('data/fea_sample/page_rank.p','rb'))
```

In [0]:
```
print('min',pr[min(pr, key=pr.get)])
print('max',pr[max(pr, key=pr.get)])
print('mean',float(sum(pr.values())) / len(pr))
```

```
min 1.6556497245737814e-07
max 2.7098251341935827e-05
mean 5.615699699389075e-07
```

In [0]:
```
#for imputing to nodes which are not there in Train data
```

```
mean_pr = float(sum(pr.values())) / len(pr)
print(mean_pr)
```

```
5.615699699389075e-07
```

4. Other Graph Features

4.1 Shortest path:

Getting Shortest path between twoo nodes, if nodes have direct path i.e directly connected then we are removing that edge and calculating path.

In [0]:
```
#if has direct edge then deleting that edge and calculating shortest pa
th
def compute_shortest_path_length(a,b):
    p=-1
    try:
        if train_graph.has_edge(a,b):
            train_graph.remove_edge(a,b)
            p= nx.shortest_path_length(train_graph,source=a,target=b)
            train_graph.add_edge(a,b)
        else:
            p= nx.shortest_path_length(train_graph,source=a,target=b)
        return p
    except:
        return -1
```

In [0]:
```
#testing
compute_shortest_path_length(77697, 826021)
```

Out[0]:
```
10
```

In [0]:
```
#testing
compute_shortest_path_length(669354,1635354)
```

Out[0]: -1

4.2 Checking for same community

In [0]:
```
#getting weekly connected edges from graph
wcc=list(nx.weakly_connected_components(train_graph))
def belongs_to_same_wcc(a,b):
    index = []
    if train_graph.has_edge(b,a):
        return 1
    if train_graph.has_edge(a,b):
            for i in wcc:
                if a in i:
                    index= i
                    break
            if (b in index):
                train_graph.remove_edge(a,b)
                if compute_shortest_path_length(a,b)==-1:
                    train_graph.add_edge(a,b)
                    return 0
                else:
                    train_graph.add_edge(a,b)
                    return 1
            else:
                return 0
    else:
            for i in wcc:
                if a in i:
                    index= i
                    break
            if(b in index):
                return 1
            else:
                return 0
```

In [0]:
```
belongs_to_same_wcc(861, 1659750)
```

Out[0]: 0

In [0]:
```
belongs_to_same_wcc(669354,1635354)
```

Out[0]: 0

4.3 Adamic/Adar Index:

Adamic/Adar measures is defined as inverted sum of degrees of common neighbours for given two vertices.

$$A(x,y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{log(|N(u)|)}$$

In [0]:
```
#adar index
def calc_adar_in(a,b):
    sum=0
    try:
        n=list(set(train_graph.successors(a)).intersection(set(train_graph.successors(b))))
        if len(n)!=0:
            for i in n:
                sum=sum+(1/np.log10(len(list(train_graph.predecessors(i)))))
            return sum
        else:
            return 0
    except:
        return 0
```

In [0]:
```
calc_adar_in(1,189226)
```

Out[0]: 0

In [0]:
```
calc_adar_in(669354,1635354)
```

Out[0]: 0

4.4 Is persion was following back:

In [0]:
```
def follows_back(a,b):
    if train_graph.has_edge(b,a):
        return 1
    else:
        return 0
```

In [0]:
```
follows_back(1,189226)
```

Out[0]: 1

In [0]:
```
follows_back(669354,1635354)
```

Out[0]: 0

4.5 Katz Centrality:

https://en.wikipedia.org/wiki/Katz_centrality

https://www.geeksforgeeks.org/katz-centrality-centrality-measure/ Katz centrality computes the centrality for a node based on the centrality of its neighbors. It is a generalization of the eigenvector centrality. The Katz centrality for node `i` is

$$x_i = \alpha \sum_j A_{ij} x_j + \beta,$$

where A is the adjacency matrix of the graph G with eigenvalues

$$\lambda$$

.

The parameter

$$\beta$$

controls the initial centrality and

$$\alpha < \frac{1}{\lambda_{max}}.$$

In [0]:
```
if not os.path.isfile('data/fea_sample/katz.p'):
    katz = nx.katz.katz_centrality(train_graph,alpha=0.005,beta=1)
    pickle.dump(katz,open('data/fea_sample/katz.p','wb'))
else:
    katz = pickle.load(open('data/fea_sample/katz.p','rb'))
```

In [0]:
```
print('min',katz[min(katz, key=katz.get)])
print('max',katz[max(katz, key=katz.get)])
print('mean',float(sum(katz.values())) / len(katz))
```

```
min 0.0007313532484065916
max 0.003394554981699122
mean 0.0007483800935562018
```

In [0]:
```
mean_katz = float(sum(katz.values())) / len(katz)
print(mean_katz)
```

```
0.0007483800935562018
```

4.6 Hits Score

The HITS algorithm computes two numbers for a node. Authorities estimates the node value based on the incoming links. Hubs estimates the node value based on outgoing links.

https://en.wikipedia.org/wiki/HITS_algorithm

In [0]:
```
if not os.path.isfile('data/fea_sample/hits.p'):
    hits = nx.hits(train_graph, max_iter=100, tol=1e-08, nstart=None, n
ormalized=True)
    pickle.dump(hits,open('data/fea_sample/hits.p','wb'))
else:
    hits = pickle.load(open('data/fea_sample/hits.p','rb'))
```

In [0]:
```python
print('min',hits[0][min(hits[0], key=hits[0].get)])
print('max',hits[0][max(hits[0], key=hits[0].get)])
print('mean',float(sum(hits[0].values())) / len(hits[0]))
```

```
min 0.0
max 0.004868653378780953
mean 5.615699699344123e-07
```

5. Featurization

5. 1 Reading a sample of Data from both train and test

In [0]:
```python
import random
if os.path.isfile('data/after_eda/train_after_eda.csv'):
    filename = "data/after_eda/train_after_eda.csv"
    # you uncomment this line, if you dont know the lentgh of the file name
    # here we have hardcoded the number of lines as 15100030
    # n_train = sum(1 for line in open(filename)) #number of records in file (excludes header)
    n_train =  15100028
    s = 100000 #desired sample size
    skip_train = sorted(random.sample(range(1,n_train+1),n_train-s))
    #https://stackoverflow.com/a/22259008/4084039
```

In [0]:
```python
if os.path.isfile('data/after_eda/train_after_eda.csv'):
    filename = "data/after_eda/test_after_eda.csv"
    # you uncomment this line, if you dont know the lentgh of the file name
    # here we have hardcoded the number of lines as 3775008
    # n_test = sum(1 for line in open(filename)) #number of records in file (excludes header)
    n_test = 3775006
    s = 50000 #desired sample size
```

```
    skip_test = sorted(random.sample(range(1,n_test+1),n_test-s))
    #https://stackoverflow.com/a/22259008/4084039
```

In [0]:
```
print("Number of rows in the train data file:", n_train)
print("Number of rows we are going to elimiate in train data are",len(s
kip_train))
print("Number of rows in the test data file:", n_test)
print("Number of rows we are going to elimiate in test data are",len(sk
ip_test))
```

```
Number of rows in the train data file: 15100028
Number of rows we are going to elimiate in train data are 15000028
Number of rows in the test data file: 3775006
Number of rows we are going to elimiate in test data are 3725006
```

In [0]:
```
df_final_train = pd.read_csv('data/after_eda/train_after_eda.csv', skip
rows=skip_train, names=['source_node', 'destination_node'])
df_final_train['indicator_link'] = pd.read_csv('data/train_y.csv', skip
rows=skip_train, names=['indicator_link'])
print("Our train matrix size ",df_final_train.shape)
df_final_train.head(2)
```

```
Our train matrix size  (100002, 3)
```

Out[0]:

	source_node	destination_node	indicator_link
0	273084	1505602	1
1	832016	1543415	1

In [0]:
```
df_final_test = pd.read_csv('data/after_eda/test_after_eda.csv', skipro
ws=skip_test, names=['source_node', 'destination_node'])
df_final_test['indicator_link'] = pd.read_csv('data/test_y.csv', skipro
ws=skip_test, names=['indicator_link'])
print("Our test matrix size ",df_final_test.shape)
df_final_test.head(2)
```

```
Our test matrix size  (50002, 3)
```

Out[0]:

	source_node	destination_node	indicator_link
0	848424	784690	1
1	483294	1255532	1

5.2 Adding a set of features

we will create these each of these features for both train and test data points

1. jaccard_followers
2. jaccard_followees
3. cosine_followers
4. cosine_followees
5. num_followers_s
6. num_followees_s
7. num_followers_d
8. num_followees_d
9. inter_followers
10. inter_followees

In [0]:

```
if not os.path.isfile('data/fea_sample/storage_sample_stage1.h5'):
    #mapping jaccrd followers to train and test data
    df_final_train['jaccard_followers'] = df_final_train.apply(lambda r
ow:
                                            jaccard_for_followers(row[
'source_node'],row['destination_node']),axis=1)
    df_final_test['jaccard_followers'] = df_final_test.apply(lambda row
:
                                            jaccard_for_followers(row[
'source_node'],row['destination_node']),axis=1)

    #mapping jaccrd followees to train and test data
    df_final_train['jaccard_followees'] = df_final_train.apply(lambda r
ow:
                                            jaccard_for_followees(row[
```

```
'source_node'],row['destination_node']),axis=1)
    df_final_test['jaccard_followees'] = df_final_test.apply(lambda row
:
                                            jaccard_for_followees(row[
'source_node'],row['destination_node']),axis=1)


        #mapping jaccrd followers to train and test data
    df_final_train['cosine_followers'] = df_final_train.apply(lambda ro
w:
                                            cosine_for_followers(row['s
ource_node'],row['destination_node']),axis=1)
    df_final_test['cosine_followers'] = df_final_test.apply(lambda row:
                                            cosine_for_followers(row['s
ource_node'],row['destination_node']),axis=1)

    #mapping jaccrd followees to train and test data
    df_final_train['cosine_followees'] = df_final_train.apply(lambda ro
w:
                                            cosine_for_followees(row['s
ource_node'],row['destination_node']),axis=1)
    df_final_test['cosine_followees'] = df_final_test.apply(lambda row:
                                            cosine_for_followees(row['s
ource_node'],row['destination_node']),axis=1)
```

In [0]:
```
def compute_features_stage1(df_final):
    #calculating no of followers followees for source and destination
    #calculating intersection of followers and followees for source and
 destination
    num_followers_s=[]
    num_followees_s=[]
    num_followers_d=[]
    num_followees_d=[]
    inter_followers=[]
    inter_followees=[]
    for i,row in df_final.iterrows():
        try:
            s1=set(train_graph.predecessors(row['source_node']))
            s2=set(train_graph.successors(row['source_node']))
```

```
        except:
            s1 = set()
            s2 = set()
        try:
            d1=set(train_graph.predecessors(row['destination_node']))
            d2=set(train_graph.successors(row['destination_node']))
        except:
            d1 = set()
            d2 = set()
        num_followers_s.append(len(s1))
        num_followees_s.append(len(s2))

        num_followers_d.append(len(d1))
        num_followees_d.append(len(d2))

        inter_followers.append(len(s1.intersection(d1)))
        inter_followees.append(len(s2.intersection(d2)))

    return num_followers_s, num_followers_d, num_followees_s, num_follo
wees_d, inter_followers, inter_followees
```

In [0]:

```
if not os.path.isfile('data/fea_sample/storage_sample_stage1.h5'):
    df_final_train['num_followers_s'], df_final_train['num_followers_d'
], \
    df_final_train['num_followees_s'], df_final_train['num_followees_d'
], \
    df_final_train['inter_followers'], df_final_train['inter_followees'
]= compute_features_stage1(df_final_train)

    df_final_test['num_followers_s'], df_final_test['num_followers_d'],
 \
    df_final_test['num_followees_s'], df_final_test['num_followees_d'],
 \
    df_final_test['inter_followers'], df_final_test['inter_followees']=
 compute_features_stage1(df_final_test)

    hdf = HDFStore('data/fea_sample/storage_sample_stage1.h5')
    hdf.put('train_df',df_final_train, format='table', data_columns=Tru
e)
```

```
    hdf.put('test_df',df_final_test, format='table', data_columns=True)
    hdf.close()
else:
    df_final_train = read_hdf('data/fea_sample/storage_sample_stage1.h
5', 'train_df',mode='r')
    df_final_test = read_hdf('data/fea_sample/storage_sample_stage1.h5'
, 'test_df',mode='r')
```

5.3 Adding new set of features

we will create these each of these features for both train and test data points

1. adar index
2. is following back
3. belongs to same weakly connect components
4. shortest path between source and destination

In [0]:

```
if not os.path.isfile('data/fea_sample/storage_sample_stage2.h5'):
    #mapping adar index on train
    df_final_train['adar_index'] = df_final_train.apply(lambda row: cal
c_adar_in(row['source_node'],row['destination_node']),axis=1)
    #mapping adar index on test
    df_final_test['adar_index'] = df_final_test.apply(lambda row: calc_
adar_in(row['source_node'],row['destination_node']),axis=1)

    #-----------------------------------------------------------------
---------------------------------------
    #mapping followback or not on train
    df_final_train['follows_back'] = df_final_train.apply(lambda row: f
ollows_back(row['source_node'],row['destination_node']),axis=1)

    #mapping followback or not on test
    df_final_test['follows_back'] = df_final_test.apply(lambda row: fol
lows_back(row['source_node'],row['destination_node']),axis=1)

    #-----------------------------------------------------------------
---------------------------------------
```

```
    #mapping same component of wcc or not on train
    df_final_train['same_comp'] = df_final_train.apply(lambda row: belo
ngs_to_same_wcc(row['source_node'],row['destination_node']),axis=1)

    ##mapping same component of wcc or not on train
    df_final_test['same_comp'] = df_final_test.apply(lambda row: belong
s_to_same_wcc(row['source_node'],row['destination_node']),axis=1)

    #-------------------------------------------------------------------
--------------------------------------
    #mapping shortest path on train
    df_final_train['shortest_path'] = df_final_train.apply(lambda row:
compute_shortest_path_length(row['source_node'],row['destination_node'
]),axis=1)
    #mapping shortest path on test
    df_final_test['shortest_path'] = df_final_test.apply(lambda row: co
mpute_shortest_path_length(row['source_node'],row['destination_node']),
axis=1)

    hdf = HDFStore('data/fea_sample/storage_sample_stage2.h5')
    hdf.put('train_df',df_final_train, format='table', data_columns=Tru
e)
    hdf.put('test_df',df_final_test, format='table', data_columns=True)
    hdf.close()
else:
    df_final_train = read_hdf('data/fea_sample/storage_sample_stage2.h
5', 'train_df',mode='r')
    df_final_test = read_hdf('data/fea_sample/storage_sample_stage2.h5'
, 'test_df',mode='r')
```

5.4 Adding new set of features

we will create these each of these features for both train and test data points

1. Weight Features
    - weight of incoming edges
    - weight of outgoing edges

- weight of incoming edges + weight of outgoing edges
- weight of incoming edges * weight of outgoing edges
- 2*weight of incoming edges + weight of outgoing edges
- weight of incoming edges + 2*weight of outgoing edges

2. Page Ranking of source
3. Page Ranking of dest
4. katz of source
5. katz of dest
6. hubs of source
7. hubs of dest
8. authorities_s of source
9. authorities_s of dest

Weight Features

In order to determine the similarity of nodes, an edge weight value was calculated between nodes. Edge weight decreases as the neighbor count goes up. Intuitively, consider one million people following a celebrity on a social network then chances are most of them never met each other or the celebrity. On the other hand, if a user has 30 contacts in his/her social network, the chances are higher that many of them know each other. `credit` - Graph-based Features for Supervised Link Prediction William Cukierski, Benjamin Hamner, Bo Yang

$$W = \frac{1}{\sqrt{1 + |X|}}$$

it is directed graph so calculated Weighted in and Weighted out differently

In [0]:
```
#weight for source and destination of each link
Weight_in = {}
Weight_out = {}
for i in  tqdm(train_graph.nodes()):
    s1=set(train_graph.predecessors(i))
```

```
    w_in = 1.0/(np.sqrt(1+len(s1)))
    Weight_in[i]=w_in

    s2=set(train_graph.successors(i))
    w_out = 1.0/(np.sqrt(1+len(s2)))
    Weight_out[i]=w_out

#for imputing with mean
mean_weight_in = np.mean(list(Weight_in.values()))
mean_weight_out = np.mean(list(Weight_out.values()))
```

```
100%|██████████████████████████████████████████████████████████████████
██| 1780722/1780722 [00:11<00:00, 152682.24it/s]
```

In [0]:
```
if not os.path.isfile('data/fea_sample/storage_sample_stage3.h5'):
    #mapping to pandas train
    df_final_train['weight_in'] = df_final_train.destination_node.apply
(lambda x: Weight_in.get(x,mean_weight_in))
    df_final_train['weight_out'] = df_final_train.source_node.apply(lam
bda x: Weight_out.get(x,mean_weight_out))

    #mapping to pandas test
    df_final_test['weight_in'] = df_final_test.destination_node.apply(l
ambda x: Weight_in.get(x,mean_weight_in))
    df_final_test['weight_out'] = df_final_test.source_node.apply(lambd
a x: Weight_out.get(x,mean_weight_out))


    #some features engineerings on the in and out weights
    df_final_train['weight_f1'] = df_final_train.weight_in + df_final_t
rain.weight_out
    df_final_train['weight_f2'] = df_final_train.weight_in * df_final_t
rain.weight_out
    df_final_train['weight_f3'] = (2*df_final_train.weight_in + 1*df_fi
nal_train.weight_out)
    df_final_train['weight_f4'] = (1*df_final_train.weight_in + 2*df_fi
nal_train.weight_out)

    #some features engineerings on the in and out weights
```

```
    df_final_test['weight_f1'] = df_final_test.weight_in + df_final_tes
t.weight_out
    df_final_test['weight_f2'] = df_final_test.weight_in * df_final_tes
t.weight_out
    df_final_test['weight_f3'] = (2*df_final_test.weight_in + 1*df_fina
l_test.weight_out)
    df_final_test['weight_f4'] = (1*df_final_test.weight_in + 2*df_fina
l_test.weight_out)
```

In [0]:

```
if not os.path.isfile('data/fea_sample/storage_sample_stage3.h5'):

    #page rank for source and destination in Train and Test
    #if anything not there in train graph then adding mean page rank
    df_final_train['page_rank_s'] = df_final_train.source_node.apply(la
mbda x:pr.get(x,mean_pr))
    df_final_train['page_rank_d'] = df_final_train.destination_node.app
ly(lambda x:pr.get(x,mean_pr))

    df_final_test['page_rank_s'] = df_final_test.source_node.apply(lamb
da x:pr.get(x,mean_pr))
    df_final_test['page_rank_d'] = df_final_test.destination_node.apply
(lambda x:pr.get(x,mean_pr))
    #=================================================================
==============

    #Katz centrality score for source and destination in Train and test
    #if anything not there in train graph then adding mean katz score
    df_final_train['katz_s'] = df_final_train.source_node.apply(lambda
x: katz.get(x,mean_katz))
    df_final_train['katz_d'] = df_final_train.destination_node.apply(la
mbda x: katz.get(x,mean_katz))

    df_final_test['katz_s'] = df_final_test.source_node.apply(lambda x:
 katz.get(x,mean_katz))
    df_final_test['katz_d'] = df_final_test.destination_node.apply(lamb
da x: katz.get(x,mean_katz))
    #=================================================================
==============
```

```
    #Hits algorithm score for source and destination in Train and test
    #if anything not there in train graph then adding 0
    df_final_train['hubs_s'] = df_final_train.source_node.apply(lambda
x: hits[0].get(x,0))
    df_final_train['hubs_d'] = df_final_train.destination_node.apply(la
mbda x: hits[0].get(x,0))

    df_final_test['hubs_s'] = df_final_test.source_node.apply(lambda x:
 hits[0].get(x,0))
    df_final_test['hubs_d'] = df_final_test.destination_node.apply(lamb
da x: hits[0].get(x,0))
    #==================================================================
==============

    #Hits algorithm score for source and destination in Train and Test
    #if anything not there in train graph then adding 0
    df_final_train['authorities_s'] = df_final_train.source_node.apply(
lambda x: hits[1].get(x,0))
    df_final_train['authorities_d'] = df_final_train.destination_node.a
pply(lambda x: hits[1].get(x,0))

    df_final_test['authorities_s'] = df_final_test.source_node.apply(la
mbda x: hits[1].get(x,0))
    df_final_test['authorities_d'] = df_final_test.destination_node.app
ly(lambda x: hits[1].get(x,0))
    #==================================================================
==============

    hdf = HDFStore('data/fea_sample/storage_sample_stage3.h5')
    hdf.put('train_df',df_final_train, format='table', data_columns=Tru
e)
    hdf.put('test_df',df_final_test, format='table', data_columns=True)
    hdf.close()
else:
    df_final_train = read_hdf('data/fea_sample/storage_sample_stage3.h
5', 'train_df',mode='r')
    df_final_test = read_hdf('data/fea_sample/storage_sample_stage3.h5'
, 'test_df',mode='r')
```

5.5 Adding new set of features

we will create these each of these features for both train and test data points

1. SVD features for both source and destination

In [0]:
```
def svd(x, S):
    try:
        z = sadj_dict[x]
        return S[z]
    except:
        return [0,0,0,0,0,0]
```

In [0]:
```
#for svd features to get feature vector creating a dict node val and in
edx in svd vector
sadj_col = sorted(train_graph.nodes())
sadj_dict = { val:idx for idx,val in enumerate(sadj_col)}
```

In [0]:
```
Adj = nx.adjacency_matrix(train_graph,nodelist=sorted(train_graph.nodes
())).asfptype()
```

In [33]:
```
from scipy.sparse.linalg import svds
U, s, V = svds(Adj, k = 6)
print('Adjacency matrix Shape',Adj.shape)
print('U Shape',U.shape)
print('V Shape',V.shape)
print('s Shape',s.shape)
```

```
Adjacency matrix Shape (1780722, 1780722)
U Shape (1780722, 6)
V Shape (6, 1780722)
s Shape (6,)
```

In [0]:
```
if not os.path.isfile('data/fea_sample/storage_sample_stage4.h5'):
    #=================================================================
=================================
```

```
    df_final_train[['svd_u_s_1', 'svd_u_s_2','svd_u_s_3', 'svd_u_s_4',
'svd_u_s_5', 'svd_u_s_6']] = \
    df_final_train.source_node.apply(lambda x: svd(x, U)).apply(pd.Seri
es)

    df_final_train[['svd_u_d_1', 'svd_u_d_2', 'svd_u_d_3', 'svd_u_d_4',
 'svd_u_d_5','svd_u_d_6']] = \
    df_final_train.destination_node.apply(lambda x: svd(x, U)).apply(pd
.Series)
    #===================================================================
==================================

    df_final_train[['svd_v_s_1','svd_v_s_2', 'svd_v_s_3', 'svd_v_s_4',
'svd_v_s_5', 'svd_v_s_6',]] = \
    df_final_train.source_node.apply(lambda x: svd(x, V.T)).apply(pd.Se
ries)

    df_final_train[['svd_v_d_1', 'svd_v_d_2', 'svd_v_d_3', 'svd_v_d_4',
 'svd_v_d_5','svd_v_d_6']] = \
    df_final_train.destination_node.apply(lambda x: svd(x, V.T)).apply(
pd.Series)
    #===================================================================
==================================

    df_final_test[['svd_u_s_1', 'svd_u_s_2','svd_u_s_3', 'svd_u_s_4',
'svd_u_s_5', 'svd_u_s_6']] = \
    df_final_test.source_node.apply(lambda x: svd(x, U)).apply(pd.Serie
s)

    df_final_test[['svd_u_d_1', 'svd_u_d_2', 'svd_u_d_3', 'svd_u_d_4',
'svd_u_d_5','svd_u_d_6']] = \
    df_final_test.destination_node.apply(lambda x: svd(x, U)).apply(pd.
Series)

    #===================================================================
==================================

    df_final_test[['svd_v_s_1','svd_v_s_2', 'svd_v_s_3', 'svd_v_s_4',
'svd_v_s_5', 'svd_v_s_6',]] = \
```

```
    df_final_test.source_node.apply(lambda x: svd(x, V.T)).apply(pd.Ser
ies)

    df_final_test[['svd_v_d_1', 'svd_v_d_2', 'svd_v_d_3', 'svd_v_d_4',
'svd_v_d_5','svd_v_d_6']] = \
    df_final_test.destination_node.apply(lambda x: svd(x, V.T)).apply(p
d.Series)
    #=====================================================================
=================================

    hdf = HDFStore('data/fea_sample/storage_sample_stage4.h5')
    hdf.put('train_df',df_final_train, format='table', data_columns=Tru
e)
    hdf.put('test_df',df_final_test, format='table', data_columns=True)
    hdf.close()
```

In [0]:

```

```

In [0]:

```
if not os.path.isfile('data/fea_sample/storage_sample_stage5.h5'):
    #=====================================================================
=================================

    #mapping jaccrd followers to train and test data
    df_final_train['Preferential_followee'] = df_final_train.apply(lamb
da row:
                                            Preferential_followee(row[
'source_node'],row['destination_node']),axis=1)
    df_final_test['Preferential_followee'] = df_final_test.apply(lambda
 row:
                                            Preferential_followee(row[
'source_node'],row['destination_node']),axis=1)

    #mapping jaccrd followees to train and test data
    df_final_train['SVD_followee'] = df_final_train.apply(lambda row:
                                            SVD_followee(row['source_no
de'],row['destination_node']),axis=1)
    df_final_test['SVD_followee'] = df_final_test.apply(lambda row:
                                            SVD_followee(row['source_no
```

```
de'],row['destination_node']),axis=1)

    #mapping jaccrd followers to train and test data
    df_final_train['Preferential_followers'] = df_final_train.apply(lam
bda row:
                                            Preferential_followers(row[
'source_node'],row['destination_node']),axis=1)
    df_final_test['Preferential_followers'] = df_final_test.apply(lambd
a row:
                                            Preferential_followers(row[
'source_node'],row['destination_node']),axis=1)

    #mapping jaccrd followees to train and test data
    df_final_train['SVD_followers'] = df_final_train.apply(lambda row:
                                            SVD_followers(row['source_n
ode'],row['destination_node']),axis=1)
    df_final_test['SVD_followers'] = df_final_test.apply(lambda row:
                                            SVD_followers(row['source_n
ode'],row['destination_node']),axis=1)


    hdf =pd.HDFStore('storage_sample_stage5.h5')
    hdf.put('train_df',df_final_train, format='table', data_columns=Tru
e)
    hdf.put('test_df',df_final_test, format='table', data_columns=True)
    hdf.close()
    #Preferential_followee
     #SVD_followee
```

In [0]:

```
    #mapping jaccrd followers to train and test data
    df_final_train['cosine_followers'] = df_final_train.apply(lambda ro
w:
                                            cosine_for_followers(row['s
ource_node'],row['destination_node']),axis=1)
    df_final_test['cosine_followers'] = df_final_test.apply(lambda row:
                                            cosine_for_followers(row['s
ource_node'],row['destination_node']),axis=1)

    #mapping jaccrd followees to train and test data
```

```
    df_final_train['cosine_followees'] = df_final_train.apply(lambda ro
w:
                                            cosine_for_followees(row['s
ource_node'],row['destination_node']),axis=1)
    df_final_test['cosine_followees'] = df_final_test.apply(lambda row:
                                            cosine_for_followees(row['s
ource_node'],row['destination_node']),axis=1)
```

In [0]:
```
from pandas import read_hdf
df_final_train = read_hdf('storage_sample_stage4.h5', 'train_df',mode=
'r')
df_final_test = read_hdf('storage_sample_stage4.h5', 'test_df',mode='r'
)
```

SVD_DOT

In [0]:
```
    TRAIN_SVD_U_SOURCE =  df_final_train.source_node.apply(lambda x: sv
d(x, U)).apply(pd.Series)
    TEST_SVD_U_SOURCE =    df_final_test.source_node.apply(lambda x: sv
d(x, U)).apply(pd.Series)


    TRAIN_SVD_U_DEST =   df_final_train.destination_node.apply(lambda x
: svd(x, U)).apply(pd.Series)
    TEST_SVD_U_DEST =    df_final_test.destination_node.apply(lambda x:
 svd(x, U)).apply(pd.Series)
    TRAIN_SVD_V_DEST =   df_final_train.destination_node.apply(lambda x
: svd(x, V.T)).apply(pd.Series)
    TEST_SVD_V_DEST =    df_final_test.destination_node.apply(lambda x:
 svd(x, V.T)).apply(pd.Series)


    TRAIN_SVD_V_SOURCE =   df_final_train.source_node.apply(lambda x: s
vd(x, V.T)).apply(pd.Series)
    TEST_SVD_V_SOURCE =    df_final_test.source_node.apply(lambda x: sv
d(x, V.T)).apply(pd.Series)
```

In [0]:
```
SVD_U_TRAIN=[]
SVD_U_TEST=[]
SVD_V_TRAIN=[]
SVD_V_TEST=[]


for index,series in TRAIN_SVD_U_DEST.iterrows():
  a=np.dot(TRAIN_SVD_U_SOURCE.iloc[index,:],TRAIN_SVD_U_DEST.iloc[index
,:])
  SVD_U_TRAIN.append(a)

for index,series in TRAIN_SVD_V_DEST.iterrows():
  b=np.dot(TRAIN_SVD_V_SOURCE.iloc[index,:],TRAIN_SVD_V_DEST.iloc[index
,:])
  SVD_V_TRAIN.append(b)

for index,series in TEST_SVD_U_DEST.iterrows():
  c=np.dot(TEST_SVD_U_SOURCE.iloc[index,:],TEST_SVD_U_DEST.iloc[index
,:])
  SVD_U_TEST.append(c)

for index,series in TEST_SVD_V_DEST.iterrows():
  d=np.dot(TEST_SVD_V_SOURCE.iloc[index,:],TEST_SVD_V_DEST.iloc[index
,:])
  SVD_V_TEST.append(d)
```

In [0]:
```
df_final_train['svd_dot_u']=SVD_U_TRAIN
df_final_train['svd_dot_v']=SVD_V_TRAIN

df_final_test['svd_dot_u']=SVD_U_TEST
df_final_test['svd_dot_v']=SVD_V_TEST
```

In [52]:
```
df_final_train.head()
```

Out[52]:

	source_node	destination_node	indicator_link	jaccard_followers	jaccard_followees	c
0	273084	1505602	1	0	0.000000	0
1	832016	1543415	1	0	0.187135	0
2	1325247	760242	1	0	0.369565	0
3	1368400	1006992	1	0	0.000000	0
4	140165	1708748	1	0	0.000000	0

In [63]:
```
df_final_train.columns
```

Out[63]:
```
Index(['source_node', 'destination_node', 'indicator_link',
       'jaccard_followers', 'jaccard_followees', 'cosine_followers',
       'cosine_followees', 'num_followers_s', 'num_followees_s',
       'num_followees_d', 'inter_followers', 'inter_followees', 'adar_i
ndex',
       'follows_back', 'same_comp', 'shortest_path', 'weight_in', 'weig
ht_out',
       'weight_f1', 'weight_f2', 'weight_f3', 'weight_f4', 'page_rank_
s',
       'page_rank_d', 'katz_s', 'katz_d', 'hubs_s', 'hubs_d', 'authorit
ies_s',
       'authorities_d', 'svd_u_s_1', 'svd_u_s_2', 'svd_u_s_3', 'svd_u_s
_4',
       'svd_u_s_5', 'svd_u_s_6', 'svd_u_d_1', 'svd_u_d_2', 'svd_u_d_3',
       'svd_u_d_4', 'svd_u_d_5', 'svd_u_d_6', 'svd_v_s_1', 'svd_v_s_2',
       'svd_v_s_3', 'svd_v_s_4', 'svd_v_s_5', 'svd_v_s_6', 'svd_v_d_1',
       'svd_v_d_2', 'svd_v_d_3', 'svd_v_d_4', 'svd_v_d_5', 'svd_v_d_6',
       'svd_dot_u', 'svd_dot_v', 'Preferential_followee', 'SVD_followe
e',
```

```
       'Preferential_followers', 'SVD_followers'],
      dtype='object')
```

In [0]:
```
y_train = df_final_train.indicator_link
y_test = df_final_test.indicator_link
```

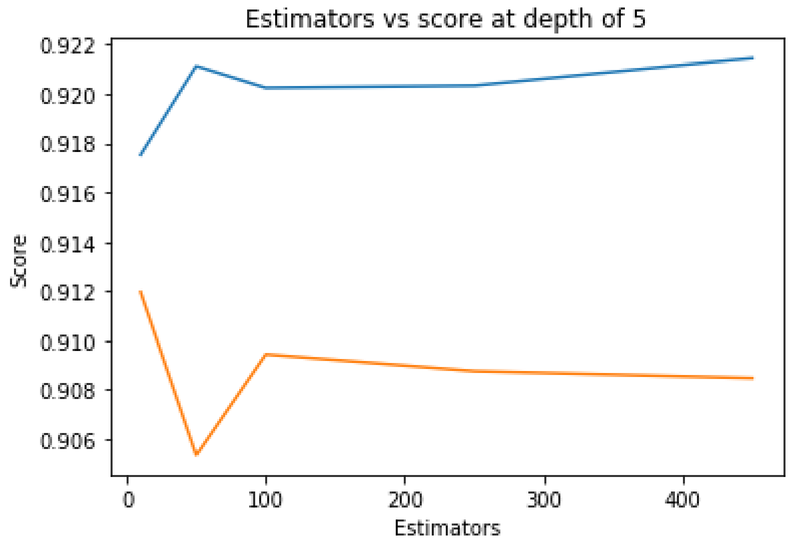
In [0]:
```
df_final_train.drop(['source_node', 'destination_node','indicator_link'
],axis=1,inplace=True)
df_final_test.drop(['source_node', 'destination_node','indicator_link'
],axis=1,inplace=True)
```

In [66]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
estimators = [10,50,100,250,450]
train_scores = []
test_scores = []
for i in estimators:
    clf = RandomForestClassifier(bootstrap=True, class_weight=None, cri
terion='gini',
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=52, min_samples_split=120,
            min_weight_fraction_leaf=0.0, n_estimators=i, n_jobs=-1,ran
dom_state=25,verbose=0,warm_start=False)
    clf.fit(df_final_train,y_train)
    train_sc = f1_score(y_train,clf.predict(df_final_train))
    test_sc = f1_score(y_test,clf.predict(df_final_test))
    test_scores.append(test_sc)
    train_scores.append(train_sc)
    print('Estimators = ',i,'Train Score',train_sc,'test Score',test_sc
)
plt.plot(estimators,train_scores,label='Train Score')
plt.plot(estimators,test_scores,label='Test Score')
plt.xlabel('Estimators')
plt.ylabel('Score')
plt.title('Estimators vs score at depth of 5')
```

```
Estimators =  10 Train Score 0.9175265291585044 test Score 0.9119539166
```

```
631628
Estimators =  50 Train Score 0.9211218778715229 test Score 0.9053501430
236255
Estimators =  100 Train Score 0.9202464237235042 test Score 0.909431894
8078218
Estimators =  250 Train Score 0.9203384518959574 test Score 0.908753192
4773623
Estimators =  450 Train Score 0.9214435594158115 test Score 0.908468993
4756446
```

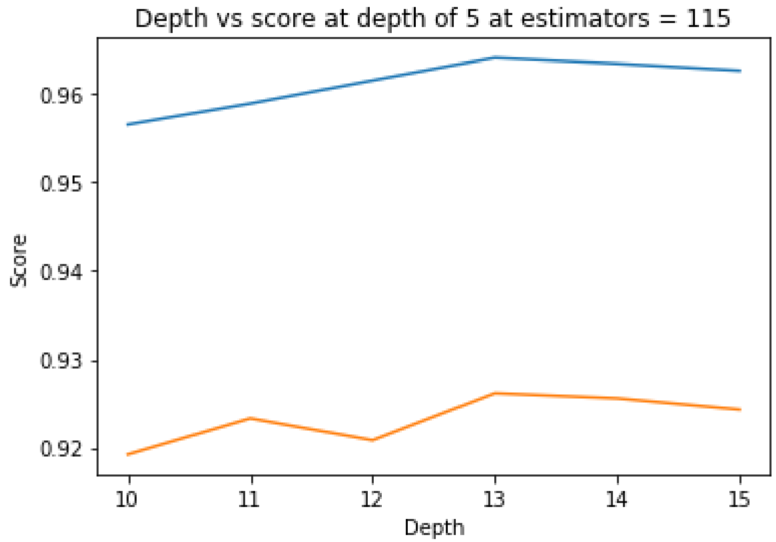
Out[66]: Text(0.5, 1.0, 'Estimators vs score at depth of 5')



In [76]:
```python
depths = [10,11,12,13,14,15]
train_scores = []
test_scores = []
for i in depths:
    clf = RandomForestClassifier(bootstrap=True, class_weight=None, cri
terion='gini',
            max_depth=i, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=52, min_samples_split=120,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,ra
```

```
ndom_state=25,verbose=0,warm_start=False)
    clf.fit(df_final_train,y_train)
    train_sc = f1_score(y_train,clf.predict(df_final_train))
    test_sc = f1_score(y_test,clf.predict(df_final_test))
    test_scores.append(test_sc)
    train_scores.append(train_sc)
    print('depth = ',i,'Train Score',train_sc,'test Score',test_sc)
plt.plot(depths,train_scores,label='Train Score')
plt.plot(depths,test_scores,label='Test Score')
plt.xlabel('Depth')
plt.ylabel('Score')
plt.title('Depth vs score at depth of 5 at estimators = 115')
plt.show()
```

```
depth =  10 Train Score 0.9565707262112056 test Score 0.919235917812294
2
depth =  11 Train Score 0.9589374450420237 test Score 0.923314767360596
4
depth =  12 Train Score 0.961547458023574 test Score 0.9208432102946157
depth =  13 Train Score 0.9641153920462057 test Score 0.926160337552742
5
depth =  14 Train Score 0.9634257848217006 test Score 0.925581885202738
3
depth =  15 Train Score 0.9626563630988184 test Score 0.924324209902415
3
```



In [0]:
```
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
from scipy.stats import uniform

param_dist = {"n_estimators":sp_randint(105,125),
              "max_depth": sp_randint(10,15),
              "min_samples_split": sp_randint(110,190),
              "min_samples_leaf": sp_randint(25,65)}

clf = RandomForestClassifier(random_state=25,n_jobs=-1)

rf_random = RandomizedSearchCV(clf, param_distributions=param_dist,
                                   n_iter=5,cv=10,scoring='f1',random_s
tate=25)

rf_random.fit(df_final_train,y_train)
# print('mean test scores',rf_random.cv_results_['mean_test_score'])
# print('mean train scores',rf_random.cv_results_['mean_train_score'])
```

In [0]:
```
print(rf_random.best_estimator_)
```

In [0]:
```
clf = RandomForestClassifier(bootstrap=True, class_weight=None, criteri
on='gini',
            max_depth=13, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=28, min_samples_split=111,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
            oob_score=False, random_state=25, verbose=0, warm_start=Fal
se)
```

In [0]:
```
clf.fit(df_final_train,y_train)
y_train_pred = clf.predict(df_final_train)
y_test_pred = clf.predict(df_final_test)
```

In [82]:
```
from sklearn.metrics import f1_score
print('Train f1 score',f1_score(y_train,y_train_pred))
print('Test f1 score',f1_score(y_test,y_test_pred))
```

```
Train f1 score 0.9629449444579582
Test f1 score 0.9229829108456077
```

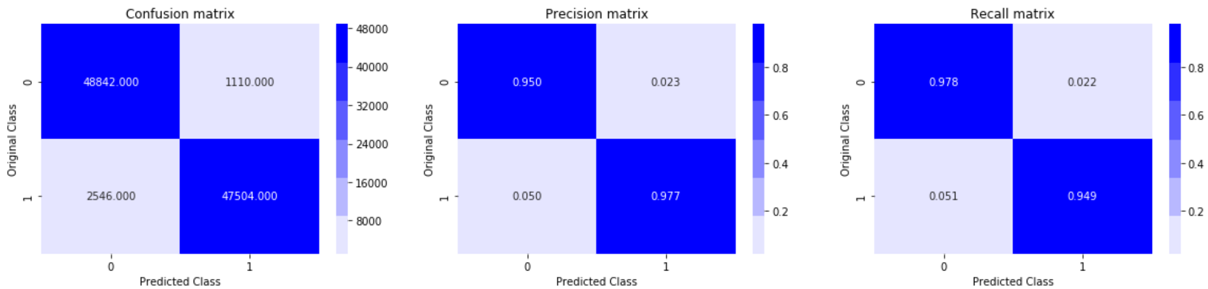
In [0]:
```
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)

    A =(((C.T)/(C.sum(axis=1))).T)

    B =(C/C.sum(axis=0))
    plt.figure(figsize=(20,4))

    labels = [0,1]
    # representing A in heatmap format
    cmap=sns.light_palette("blue")
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels
, yticklabels=labels)
```

```
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels
, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels
, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")

    plt.show()
```

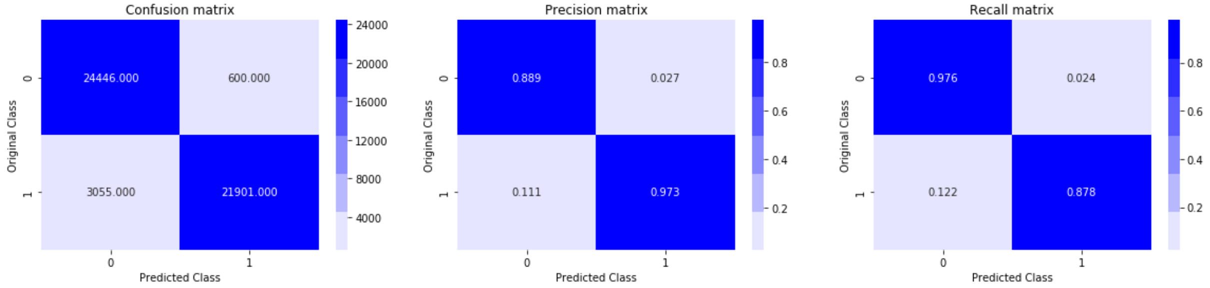
In [84]:

```
print('Train confusion_matrix')
plot_confusion_matrix(y_train,y_train_pred)
print('Test confusion_matrix')
plot_confusion_matrix(y_test,y_test_pred)
```

```
Train confusion_matrix
```

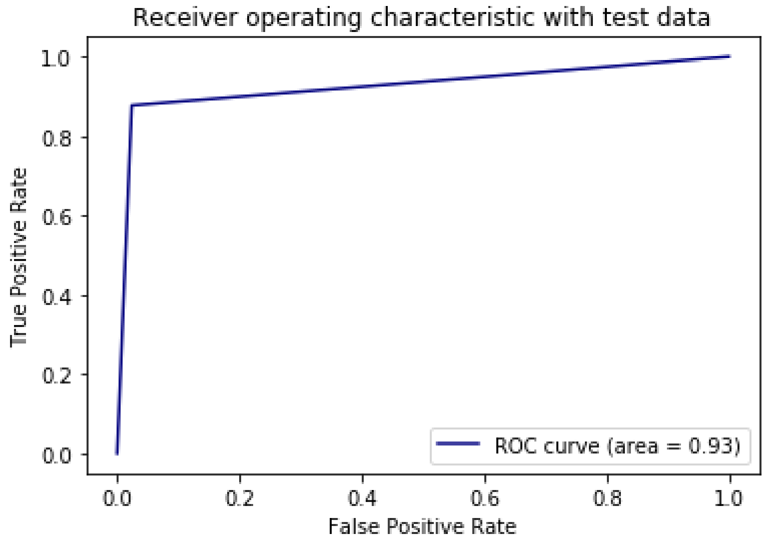



```
Test confusion_matrix
```

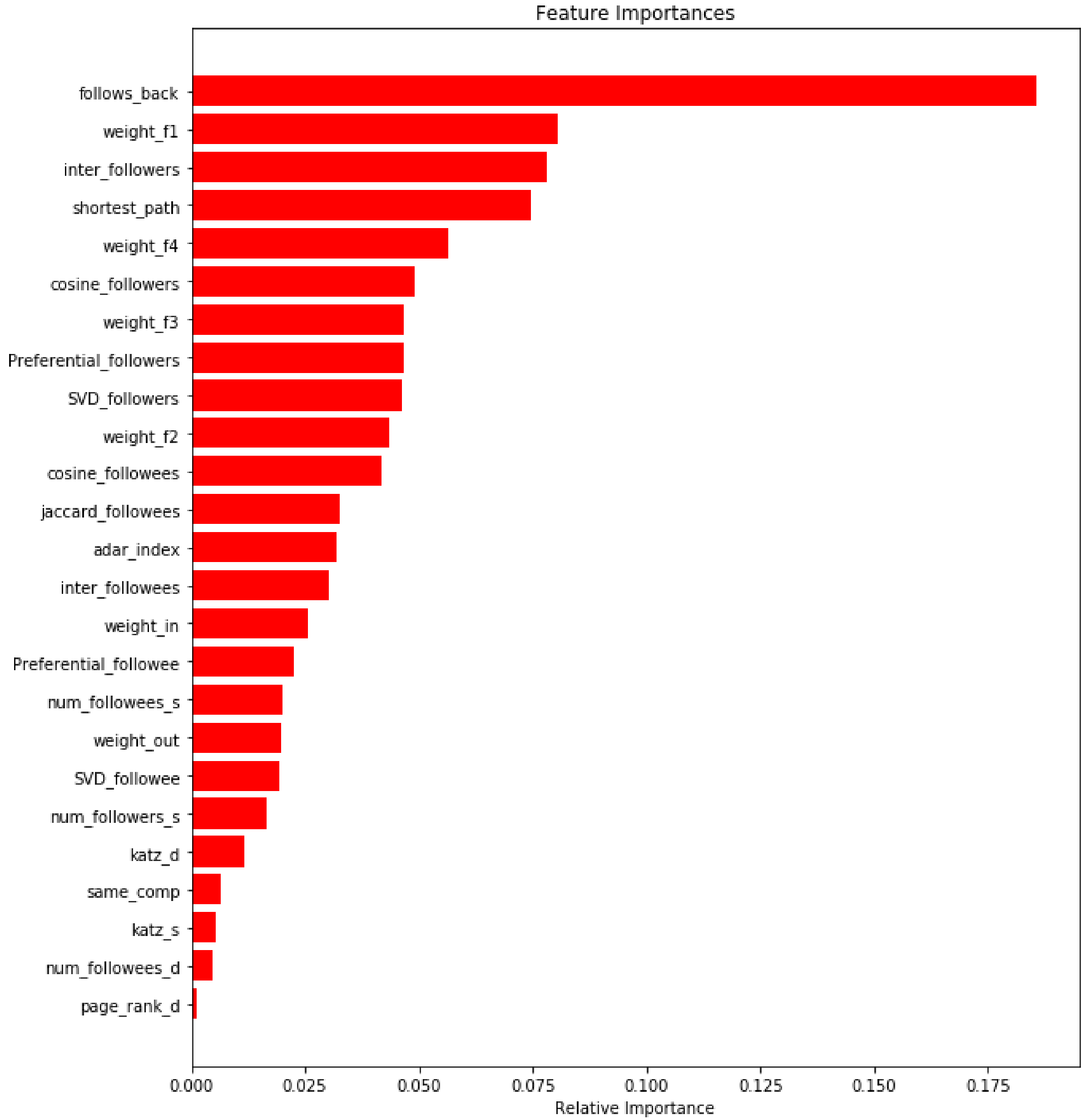


In [0]:
```
#standardizing the data
from sklearn.model_selection import GridSearchCV
# from sklearn.preprocessing import StandardScaler
# train_std = StandardScaler().fit_transform(df_train)
# test_std = StandardScaler().fit_transform(df_test)
```

In [86]:
```
from sklearn.metrics import roc_curve, auc
fpr,tpr,ths = roc_curve(y_test,y_test_pred)
auc_sc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='navy',label='ROC curve (area = %0.2f)' % auc_
sc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic with test data')
plt.legend()
plt.show()
```



In [87]:
```
features = df_final_train.columns
importances = clf.feature_importances_
indices = (np.argsort(importances))[-25:]
plt.figure(figsize=(10,12))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='r', align='c
enter')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In [90]:
```
#hyper-parameter tuning
```

```
hyper_parameter = {"max_depth":[  4,5], "n_estimators":[40,60,80]}
clf = xgb.XGBClassifier()
best_parameter = GridSearchCV(clf, hyper_parameter, cv = 3,scoring='f1'
)
best_parameter.fit(df_final_train,y_train)

estimators = best_parameter.best_params_["n_estimators"]
depth = best_parameter.best_params_["max_depth"]

clf = xgb.XGBClassifier(
 learning_rate =0.1,
 n_estimators=estimators,
 max_depth=depth,
 min_child_weight=3,
 gamma=0,
 subsample=0.8,
 reg_alpha=200, reg_lambda=200,
 colsample_bytree=0.8,nthread=4)
clf.fit(df_final_train,y_train)
```

Out[90]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=3, missing=None, n_estimators=80, n_jobs
=1,
              nthread=4, objective='binary:logistic', random_state=0,
              reg_alpha=200, reg_lambda=200, scale_pos_weight=1, seed=N
one,
              silent=None, subsample=0.8, verbosity=1)
```

In [0]:

```
clf.fit(df_final_train,y_train)
y_train_pred = clf.predict(df_final_train)
y_test_pred = clf.predict(df_final_test)
```

In [92]:

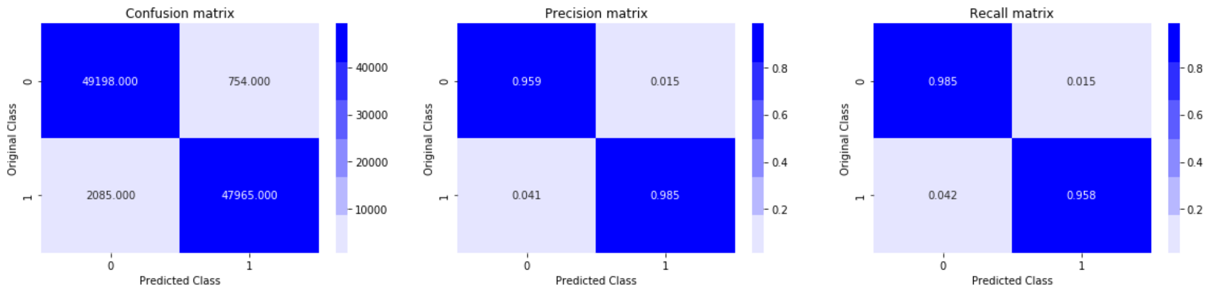
```
from sklearn.metrics import f1_score
print('Train f1 score',f1_score(y_train,y_train_pred))
print('Test f1 score',f1_score(y_test,y_test_pred))
```

```
Train f1 score 0.9712561633710982
```

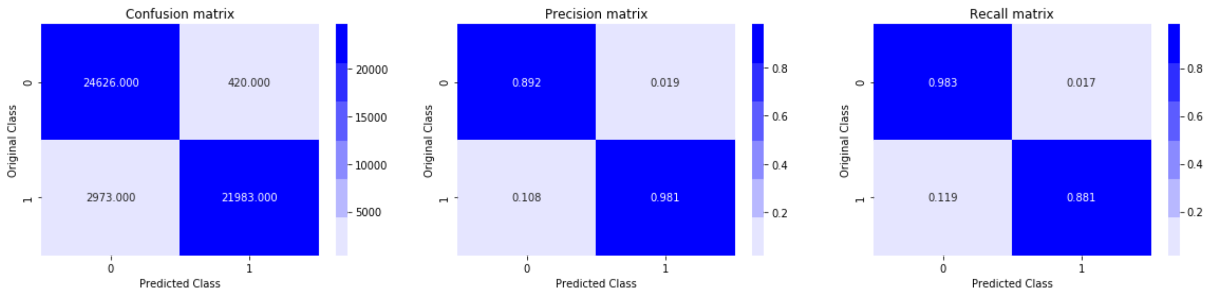
```
Test f1 score 0.9283557507548724
```

In [93]:
```python
print('Train confusion_matrix')
plot_confusion_matrix(y_train,y_train_pred)
print('Test confusion_matrix')
plot_confusion_matrix(y_test,y_test_pred)
```

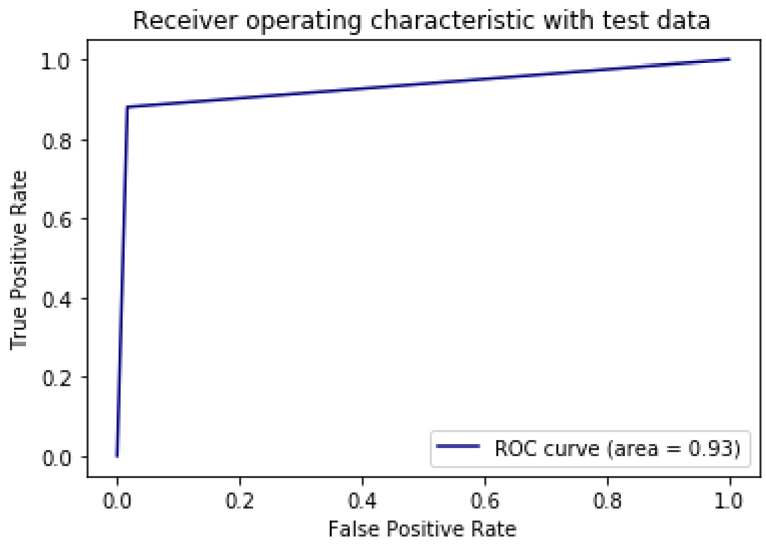
```
Train confusion_matrix
```




```
Test confusion_matrix
```

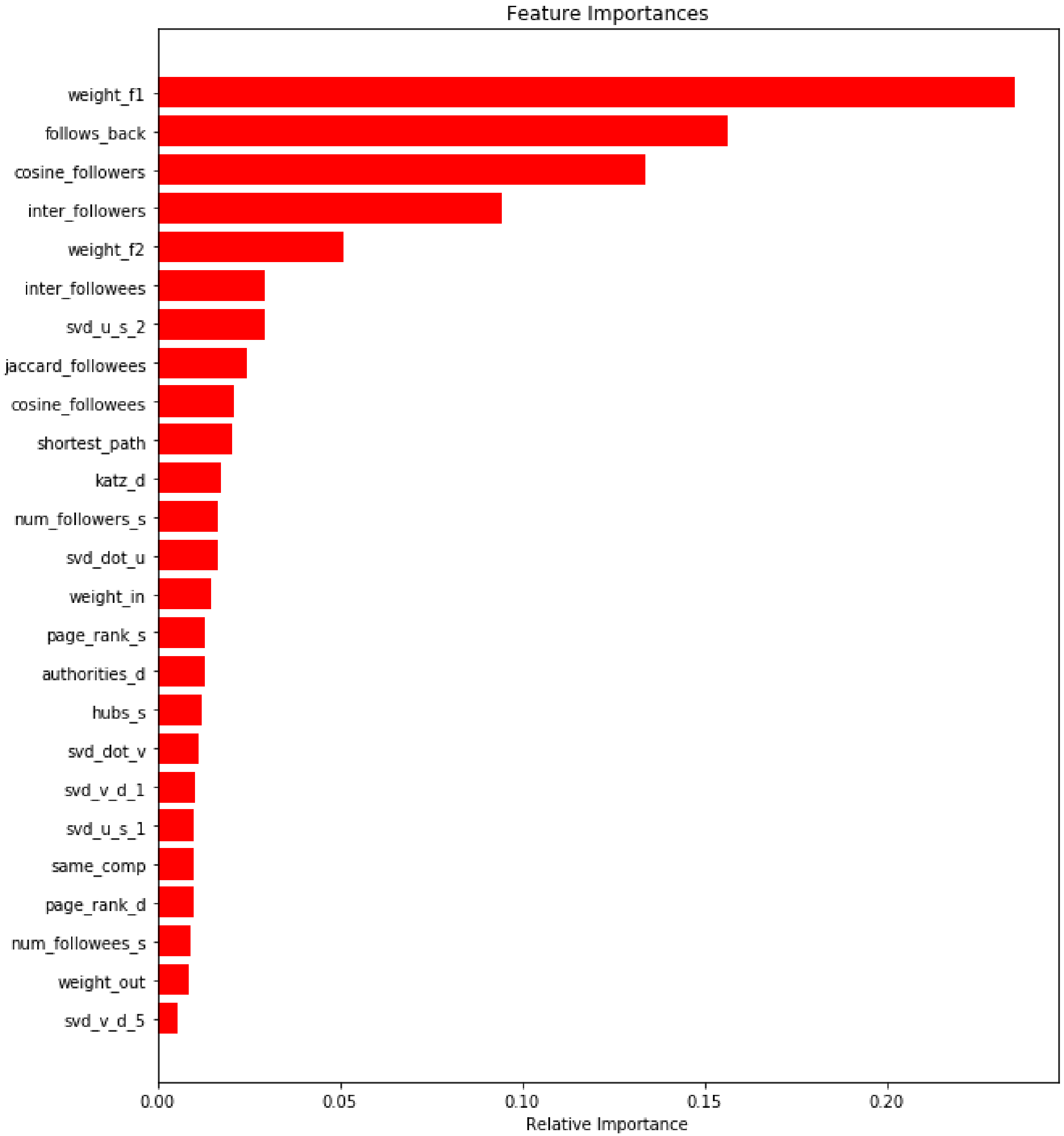


In [94]:
```python
from sklearn.metrics import roc_curve, auc
fpr,tpr,ths = roc_curve(y_test,y_test_pred)
auc_sc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='navy',label='ROC curve (area = %0.2f)' % auc_
sc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic with test data')
plt.legend()
plt.show()
```



In [95]:

```
features = df_final_train.columns
importances = clf.feature_importances_
indices = (np.argsort(importances))[-25:]
plt.figure(figsize=(10,12))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='r', align='c
enter')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```


Feature Importances
weight_f1
follows_back
cosine_followers
inter_followers
weight_f2
inter_followees
svd_u_s_2
jaccard_followees
cosine_followees
shortest_path
katz_d
num_followers_s
svd_dot_u
weight_in
page_rank_s
authorities_d
hubs_s
svd_dot_v
svd_v_d_1
svd_u_s_1
same_comp
page_rank_d
num_followees_s
weight_out
svd_v_d_5
0.00
0.05
0.10
0.15
0.20
Relative Importance

Conclusion

5.4 Adding new set of features

1) Collected the data

2) imported the libraries

3) performed EDA to analyse number of followers and followees

4) labeled 1 to the data points given where when one id follows other id

5) labeled 0 to the data point where the path distance is more than 2 to balance the data from the created graph

6) Feature Enginerring is applied on the data

we will create these each of these features for both train and test data points

1. Weight Features
   - weight of incoming edges
   - weight of outgoing edges
   - weight of incoming edges + weight of outgoing edges
   - weight of incoming edges * weight of outgoing edges
   - 2*weight of incoming edges + weight of outgoing edges
   - weight of incoming edges + 2*weight of outgoing edges
2. Page Ranking of source
3. Page Ranking of dest
4. katz of source
5. katz of dest
6. hubs of source
7. hubs of dest
8. authorities_s of source
9. authorities_s of dest
10. adar index

11. is following back
12. belongs to same weakly connect components
13. shortest path between source and destination
14. jaccard_followers
15. jaccard_followees
16. cosine_followers
17. cosine_followees
18. num_followers_s
19. num_followees_s
20. num_followers_d
21. num_followees_d
22. inter_followers
23. inter_followees
24. Parential_followers_d
25. Parential_followees_d
26. SVD_followers
27. svd_followees

8) Splitted the data into train dataset an test datatset

9) Applied RandomforestClassifier and XGBClassifier and the results are shown above using FI score and accuracy of 92.8%