Social network Graph Link Prediction - Facebook Challenge

In [2]:

```
#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
from pandas import HDFStore,DataFrame
from pandas import read_hdf
from scipy.sparse.linalg import svds, eigs
import gc
from tqdm import tqdm
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
```

In [4]:

```
#reading
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')
```

In [5]:

```
df_final_train.columns
```

Out[5]:

```
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_inde
x',
       'follows_back', 'same_comp', 'shortest_path', 'weight_in', 'weight_ou
t',
       'weight_f1', 'weight_f2', 'weight_f3', 'weight_f4', 'page_rank_s',
       'page_rank_d', 'katz_s', 'katz_d', 'hubs_s', 'hubs_d', 'authorities_
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'],
      dtype='object')
```

In [8]:

```
df_final_train.head()
```

Out[8]:

num_followees_s	num_followees_d	inter_followers	inter_followees	adar_index	...	svd_v_s_3
15	8	0	0	0.000000	...	1.983691e-06
61	142	11	32	16.362912	...	-6.236048e-11
41	22	26	17	10.991826	...	-2.380564e-19
5	7	0	0	0.000000	...	6.058498e-11
11	3	0	0	0.000000	...	1.197283e-07

In [6]:

```
y_train = df_final_train.indicator_link
y_test = df_final_test.indicator_link
```

In [7]:

```
df_final_train.drop(['source_node', 'destination_node','indicator_link'],axis=1,inplace=Tru
df_final_test.drop(['source_node', 'destination_node','indicator_link'],axis=1,inplace=True
```

In [10]:

```
# Adding Preferential attachment for followers

df_final_train['Preferential_attachment']=[i*j*k for i, j, k  in zip(df_final_train['num_fo
df_final_test['Preferential_attachment']=[i*j*k for i, j, k  in zip(df_final_test['num_foll
```

In [11]:

```
df_final_train['Preferential_attachment'].values
```

Out[11]:

```
array([   720, 814228,  25256, ...,      0,      0,      0], dtype=int64)
```

In [12]:

```
df_final_test['Preferential_attachment'].values
```

Out[12]:

```
array([ 756,  323, 1440, ...,    0,    0,  540], dtype=int64)
```

In [18]:

```
# Adding svd_dot feature

df_final_train['svd_dot_u']=[np.dot(i,j) for i, j  in zip(df_final_train[['svd_u_s_1', 'svd
df_final_test['svd_dot_u']=[np.dot(i,j) for i, j  in zip(df_final_test[['svd_u_s_1', 'svd_u

df_final_train['svd_dot_v']=[np.dot(i,j) for i, j  in zip(df_final_train[['svd_v_s_1', 'svd
df_final_test['svd_dot_v']=[np.dot(i,j) for i, j  in zip(df_final_test[['svd_v_s_1', 'svd_v
```

In [20]:

```
df_final_train['svd_dot_u'].values
```

Out[20]:

```
array([1.11495785e-11, 3.19281225e-03, 1.78750258e-35, ...,
       4.15849064e-22, 4.30096716e-30, 1.02639899e-26])
```

In [21]:

```
df_final_test['svd_dot_v'].values
```

Out[21]:

```
array([2.07480755e-17, 1.18837644e-17, 3.90488508e-12, ...,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00])
```

In [32]:

```
df_final_train.columns
```

Out[32]:

```
Index(['jaccard_followers', 'jaccard_followees', 'cosine_followers',
       'cosine_followees', 'num_followers_s', 'num_followees_s',
       'num_followees_d', 'inter_followers', 'inter_followees', 'adar_inde
x',
       'follows_back', 'same_comp', 'shortest_path', 'weight_in', 'weight_ou
t',
       'weight_f1', 'weight_f2', 'weight_f3', 'weight_f4', 'page_rank_s',
       'page_rank_d', 'katz_s', 'katz_d', 'hubs_s', 'hubs_d', 'authorities_
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',
       'Preferential_attachment', 'svd_dot_u', 'svd_dot_v'],
      dtype='object')
```

In [19]:

```
X_train = df_final_train
X_test = df_final_test
```

In [22]:

```
X_train.shape, X_test.shape
```

Out[22]:

```
((100002, 54), (50002, 54))
```

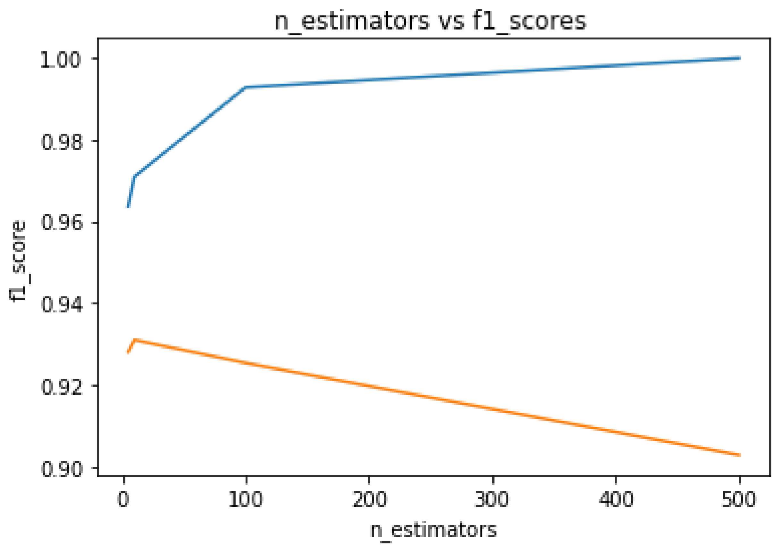
In [26]:

```
n_estimators=[5, 10, 100, 500]
train_scores = []
test_scores = []
for i in n_estimators:
    x_clf=xgb.XGBClassifier(n_estimators=i,nthread=-1, eval_metric = 'logloss')
    x_clf.fit(X_train,y_train)
    train_score = f1_score(y_train,x_clf.predict(X_train))
    test_score = f1_score(y_test,x_clf.predict(X_test))
    train_scores.append(train_score)
    test_scores.append(test_score)
    print('estimators = ',i,'Train Score',train_score,'test Score',test_score)
```

```
estimators =  5 Train Score 0.9636921057758364 test Score 0.928062602334829
estimators =  10 Train Score 0.9709757164511633 test Score 0.931000889491295
7
estimators =  100 Train Score 0.9928712513911588 test Score 0.92538011695906
43
estimators =  500 Train Score 1.0 test Score 0.902857391002258
```

In [27]:

```
plt.plot(n_estimators,train_scores,label='Train Score')
plt.plot(n_estimators,test_scores,label='Test Score')
plt.xlabel('n_estimators')
plt.ylabel('f1_score')
plt.title('n_estimators vs f1_scores ')
plt.show()

best_n_estimators = np.argmax(test_scores)
print("best n_estimators:" , n_estimators[best_n_estimators])
clf= xgb.XGBClassifier(n_estimators= n_estimators[best_n_estimators],nthread=-1, eval_metri
clf.fit(X_train,y_train)

y_train_pred = clf.predict(X_train)
y_test_pred = clf.predict(X_test)
train_score = f1_score(y_train,clf.predict(X_train))
test_score = f1_score(y_test,clf.predict(X_test))
print("train f1-score: ", train_score, "test f1-score : ", test_score)
```




```
best n_estimators: 10
train f1-score:  0.9709757164511633 test f1-score :  0.9310008894912957
```

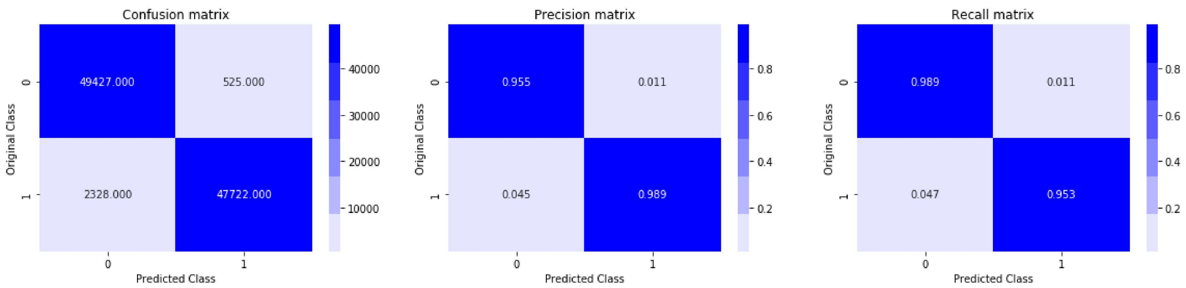
In [28]:

```
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()
```

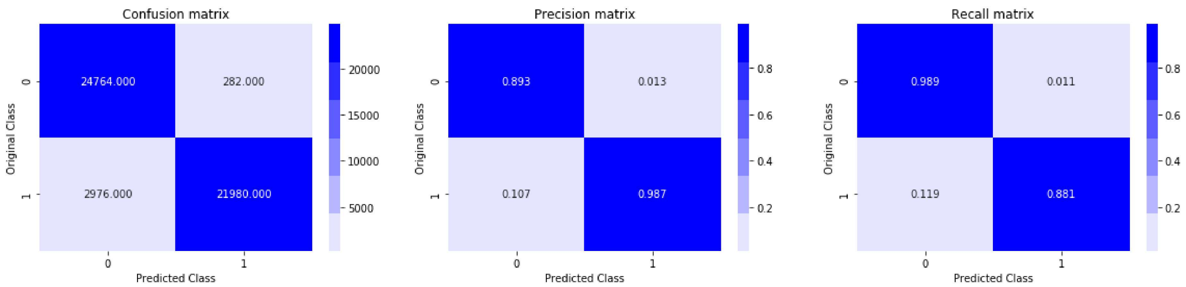
In [29]:

```
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
```

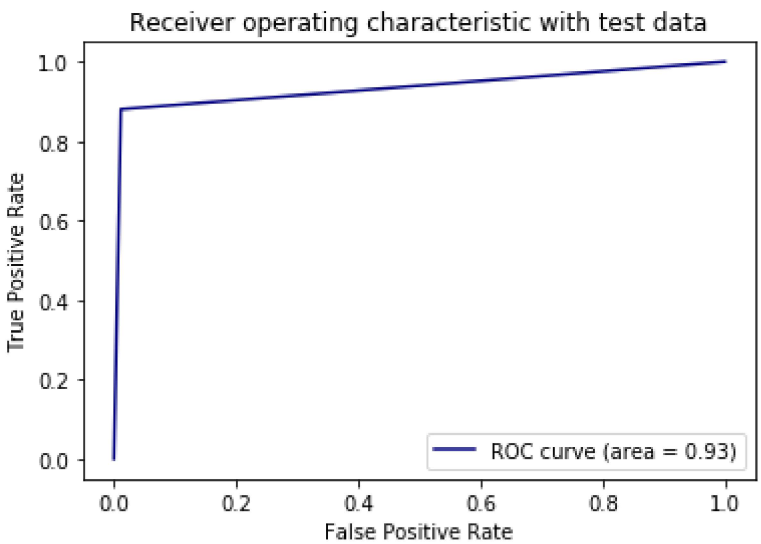



```
Test confusion_matrix
```



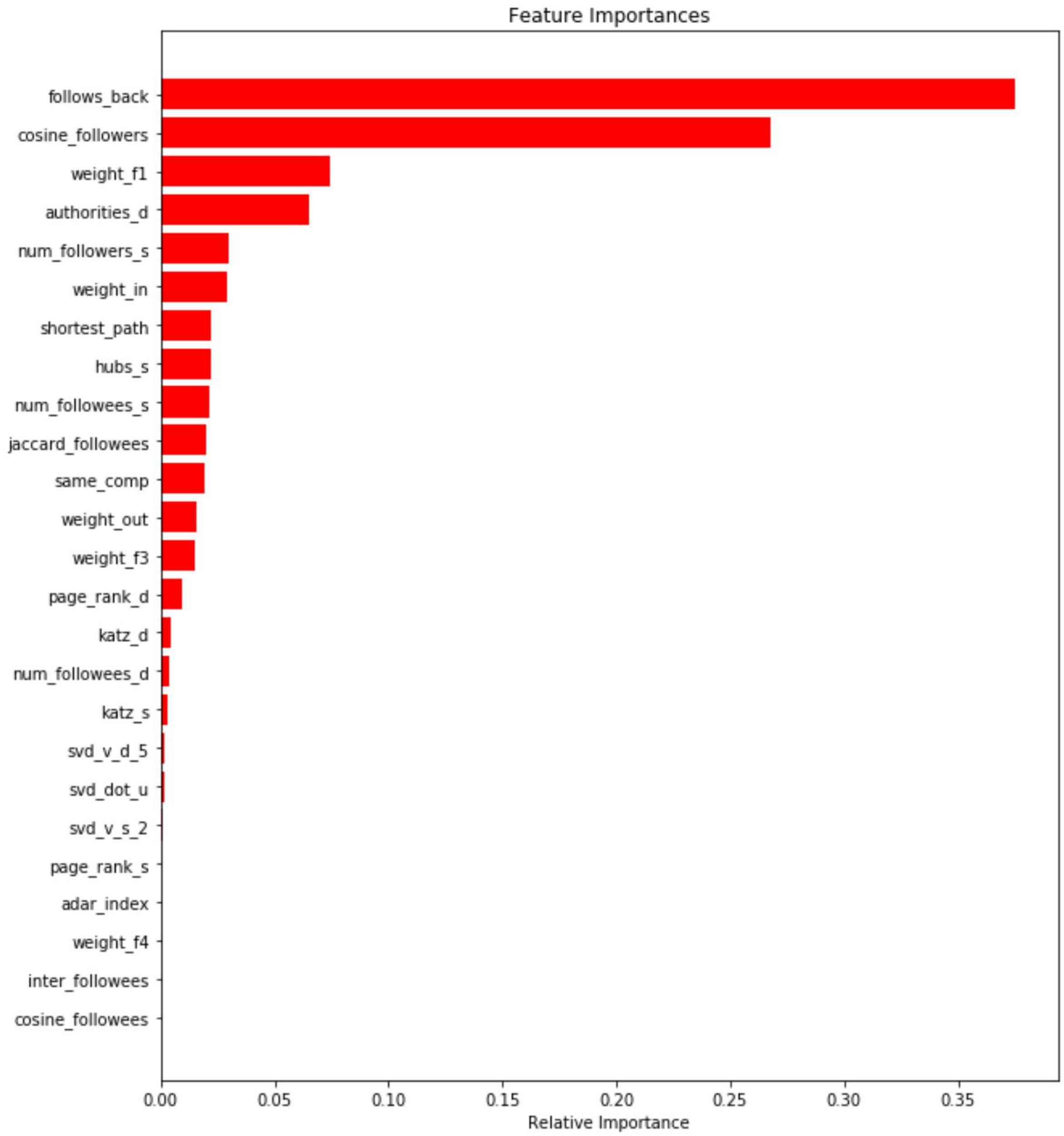
In [30]:

```
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 [31]:

```
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='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



Assignments:

1. Add another feature called Preferential Attachment with followers and followees data of vertex. you can check about Preferential Attachment in below link [http://be.amazd.com/link-prediction/ (http://be.amazd.com/link-prediction/)](http://be.amazd.com/link-prediction/)

2. Add feature called svd_dot. you can calculate svd_dot as Dot product between sourse node svd and destination node svd features. you can read about this in below pdf https://storage.googleapis.com/kaggle-forum-message-attachments/2594/supervised_link_prediction.pdf (https://storage.googleapis.com/kaggle-forum-message-attachments/2594/supervised_link_prediction.pdf)
3. Tune hyperparameters for XG boost with all these features and check the error metric.