Social network Graph Link Prediction - Facebook Challenge

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
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 [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clien
t_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.co
m&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2
F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.co
m%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.re
adonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&respon
se_type=code (https://accounts.google.com/o/oauth2/auth?client_id=9473189898
03-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=
urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleap
is.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:
..........
Mounted at /content/drive
```

In [0]:

```
#reading
from pandas import read_hdf
df_final_train = read_hdf('drive//My Drive//fb/data//data//fea_sample//storage_sample_stage
df_final_test = read_hdf('drive//My Drive//fb/data//data//fea_sample//storage_sample_stage4
```

In [4]:

```
train_graph=nx.read_edgelist('drive//My Drive//fb/data//data/after_eda/train_pos_after_eda.
print(nx.info(train_graph))
```

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

Adding Preferential Attachment feature

In [0]:

```
def prefential_attachment_followers(a,b):
    try:
       if len(set(train_graph.predecessors(a))) == 0  | len(set(train_graph.predecessors(b
           return 0
       pref_attach = (len(set(train_graph.predecessors(a))*len(set(train_graph.predecessor

    except:
      return 0

    return pref_attach
```

In [0]:

```
def prefential_attachment_followees(a,b):
    try:
       if len(set(train_graph.successors(a))) == 0  | len(set(train_graph.successors(b)))
           return 0
       pref_attach = (len(set(train_graph.successors(a))*len(set(train_graph.successors(b)

    except:
      return 0

    return pref_attach
```

In [0]:

```
df_final_train['prefential_followers'] = df_final_train.apply(lambda row:
                                        prefential_attachment_followers(row['source_node'],
                                        row['destination_node']),axis=1)
df_final_train['prefential_followees'] = df_final_train.apply(lambda row:
                                        prefential_attachment_followees(row['source_node'],
                                         row['destination_node']),axis=1)

df_final_test['prefential_followers'] = df_final_test.apply(lambda row:
                                        prefential_attachment_followers(row['source_node'],
                                        row['destination_node']),axis=1)
df_final_test['prefential_followees'] = df_final_test.apply(lambda row:
                                        prefential_attachment_followees(row['source_node'],
                                        row['destination_node']),axis=1)
```

Adding SVD-Dot feature

In [0]:

```
df_final_train['svd_u_dot'] = ((df_final_train['svd_u_s_1']*(df_final_train['svd_u_d_1']))+
                               (df_final_train['svd_u_s_2']*(df_final_train['svd_u_d_2']))
                               (df_final_train['svd_u_s_3']*(df_final_train['svd_u_d_3']))
                               (df_final_train['svd_u_s_4']*(df_final_train['svd_u_d_4']))
                               (df_final_train['svd_u_s_5']*(df_final_train['svd_u_d_5']))
                               (df_final_train['svd_u_s_6']*(df_final_train['svd_u_d_6']))

df_final_train['svd_v_dot'] = ((df_final_train['svd_v_s_1']*(df_final_train['svd_v_d_1']))+
                               (df_final_train['svd_v_s_2']*(df_final_train['svd_v_d_2']))
                               (df_final_train['svd_v_s_3']*(df_final_train['svd_v_d_3']))
                               (df_final_train['svd_v_s_4']*(df_final_train['svd_v_d_4']))
                               (df_final_train['svd_v_s_5']*(df_final_train['svd_v_d_5']))
                               (df_final_train['svd_v_s_6']*(df_final_train['svd_v_d_6']))
```

In [9]:

```
df_final_train.columns
```

Out[9]:

```
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',
       'prefential_followers', 'prefential_followees', 'svd_u_dot',
       'svd_v_dot'],
      dtype='object')
```

In [0]:

```
df_final_test['svd_u_dot'] = ((df_final_test['svd_u_s_1']*(df_final_test['svd_u_d_1']))+
                              (df_final_test['svd_u_s_2']*(df_final_test['svd_u_d_2']))+
                              (df_final_test['svd_u_s_3']*(df_final_test['svd_u_d_3']))+
                              (df_final_test['svd_u_s_4']*(df_final_test['svd_u_d_4']))+
                              (df_final_test['svd_u_s_5']*(df_final_test['svd_u_d_5']))+
                              (df_final_test['svd_u_s_6']*(df_final_test['svd_u_d_6'])))

df_final_test['svd_v_dot'] = ((df_final_test['svd_v_s_1']*(df_final_test['svd_v_d_1']))+
                              (df_final_test['svd_v_s_2']*(df_final_test['svd_v_d_2']))+
                              (df_final_test['svd_v_s_3']*(df_final_test['svd_v_d_3']))+
                              (df_final_test['svd_v_s_4']*(df_final_test['svd_v_d_4']))+
                              (df_final_test['svd_v_s_5']*(df_final_test['svd_v_d_5']))+
                              (df_final_test['svd_v_s_6']*(df_final_test['svd_v_d_6'])))
```

In [12]:

```
df_final_test.columns
```

Out[12]:

```
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',
       'prefential_followers', 'prefential_followees', 'svd_u_dot',
       'svd_v_dot'],
      dtype='object')
```

In [13]:

```
df_final_train.head()
```

Out[13]:

d_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	p
719702e-14	-1.355368e-12	4.675307e-13	1.128591e-06	6.616550e-14	9.771077e-13	4.159752e-14	
251737e-10	1.245101e-12	-1.636948e-10	-3.112650e-10	6.738902e-02	2.607801e-11	2.372904e-09	
365389e-19	-1.238370e-18	1.438175e-19	-1.852863e-19	-5.901864e-19	1.629341e-19	-2.572452e-19	
498061e-13	-9.818087e-10	3.454672e-11	5.213635e-08	9.595823e-13	3.047045e-10	1.246592e-13	
407670e-14	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	

In [0]:

```

```

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

Applying RandomForest Classifier

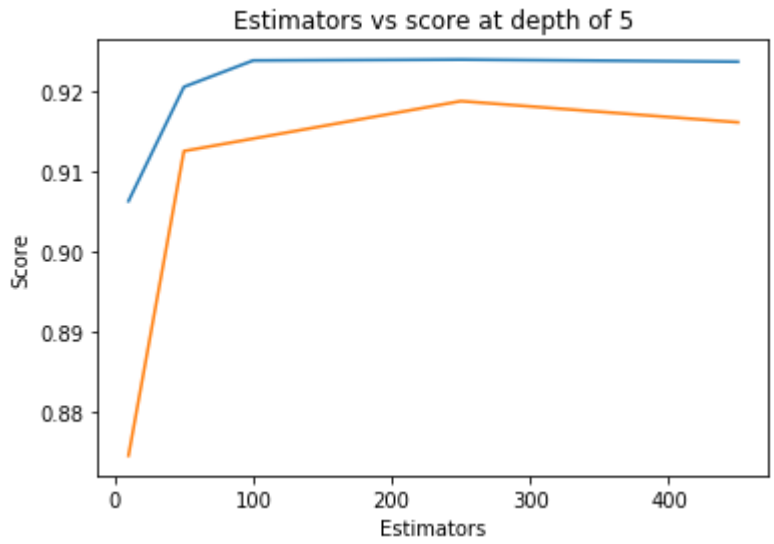
In [0]:

```
estimators = [10,50,100,250,450]
train_scores = []
test_scores = []
for i in estimators:
    clf = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='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,random_state=25,verbose
    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 0.9063252121775113 test Score 0.874560527800685
8
Estimators =  50 Train Score 0.9205725512208812 test Score 0.912565335563453
8
Estimators =  100 Train Score 0.9238690848446947 test Score 0.91411997141535
99
Estimators =  250 Train Score 0.9239789348046863 test Score 0.91880072326647
32
Estimators =  450 Train Score 0.9237190618658074 test Score 0.91615076858285
95
```

Out[6]:

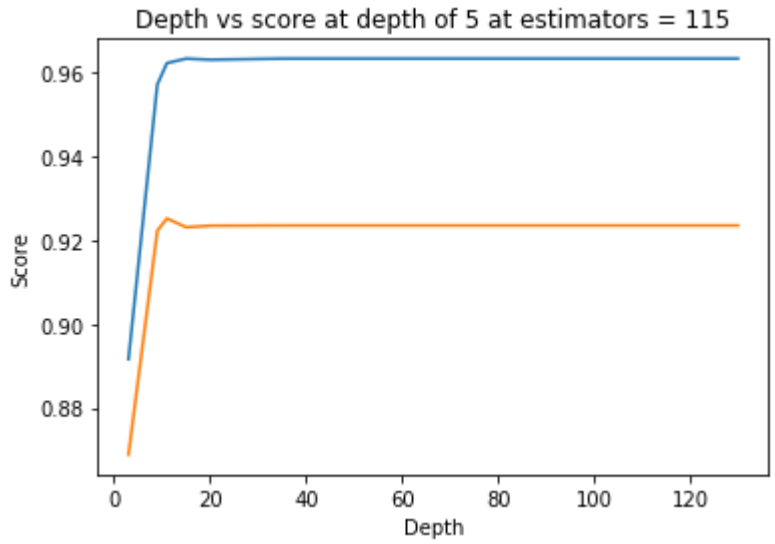
```
Text(0.5,1,'Estimators vs score at depth of 5')
```



In [0]:

```
depths = [3,9,11,15,20,35,50,70,130]
train_scores = []
test_scores = []
for i in depths:
    clf = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='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=115, n_jobs=-1,random_state=25,verbo
    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 =  3 Train Score 0.8916120853581238 test Score 0.8687934859875491
depth =  9 Train Score 0.9572226298198419 test Score 0.9222953031452904
depth =  11 Train Score 0.9623451340902863 test Score 0.9252318758281279
depth =  15 Train Score 0.9634267621927706 test Score 0.9231288356496615
depth =  20 Train Score 0.9631629153051491 test Score 0.9235051024711141
depth =  35 Train Score 0.9634333127085721 test Score 0.9235601652753184
depth =  50 Train Score 0.9634333127085721 test Score 0.9235601652753184
depth =  70 Train Score 0.9634333127085721 test Score 0.9235601652753184
depth =  130 Train Score 0.9634333127085721 test Score 0.9235601652753184
```



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()

rf_random = RandomizedSearchCV(clf, param_distributions=param_dist,
                                    n_iter=5,cv=10,scoring='f1',verbose = 30, n_jobs=-1)

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_)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=14, 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=121, n_jobs=-1,
            oob_score=False, random_state=25, verbose=0, warm_start=False)
```

In [0]:

```
clf = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=14, 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=121, n_jobs=-1,
            oob_score=False, random_state=25, verbose=0, warm_start=False)
```

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 [0]:

```
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 0.9652533106548414
Test f1 score 0.9241678239279553
```

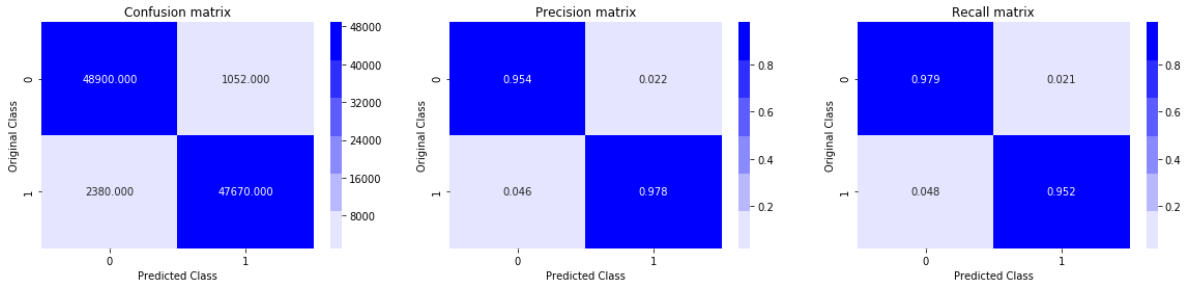
In [0]:

```
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("green")
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=la
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

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

    plt.show()
```

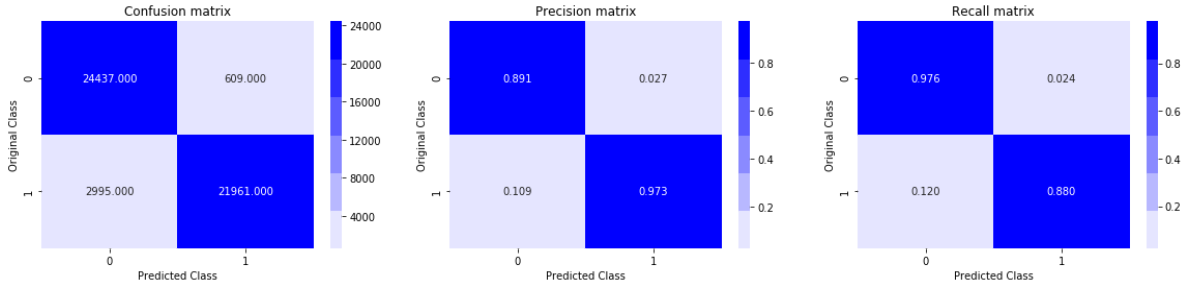
In [0]:

```
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
```

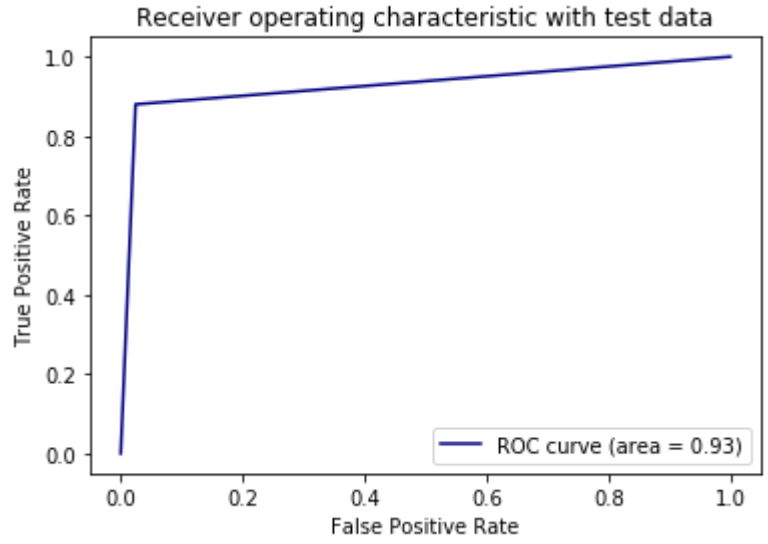



```
Test confusion_matrix
```



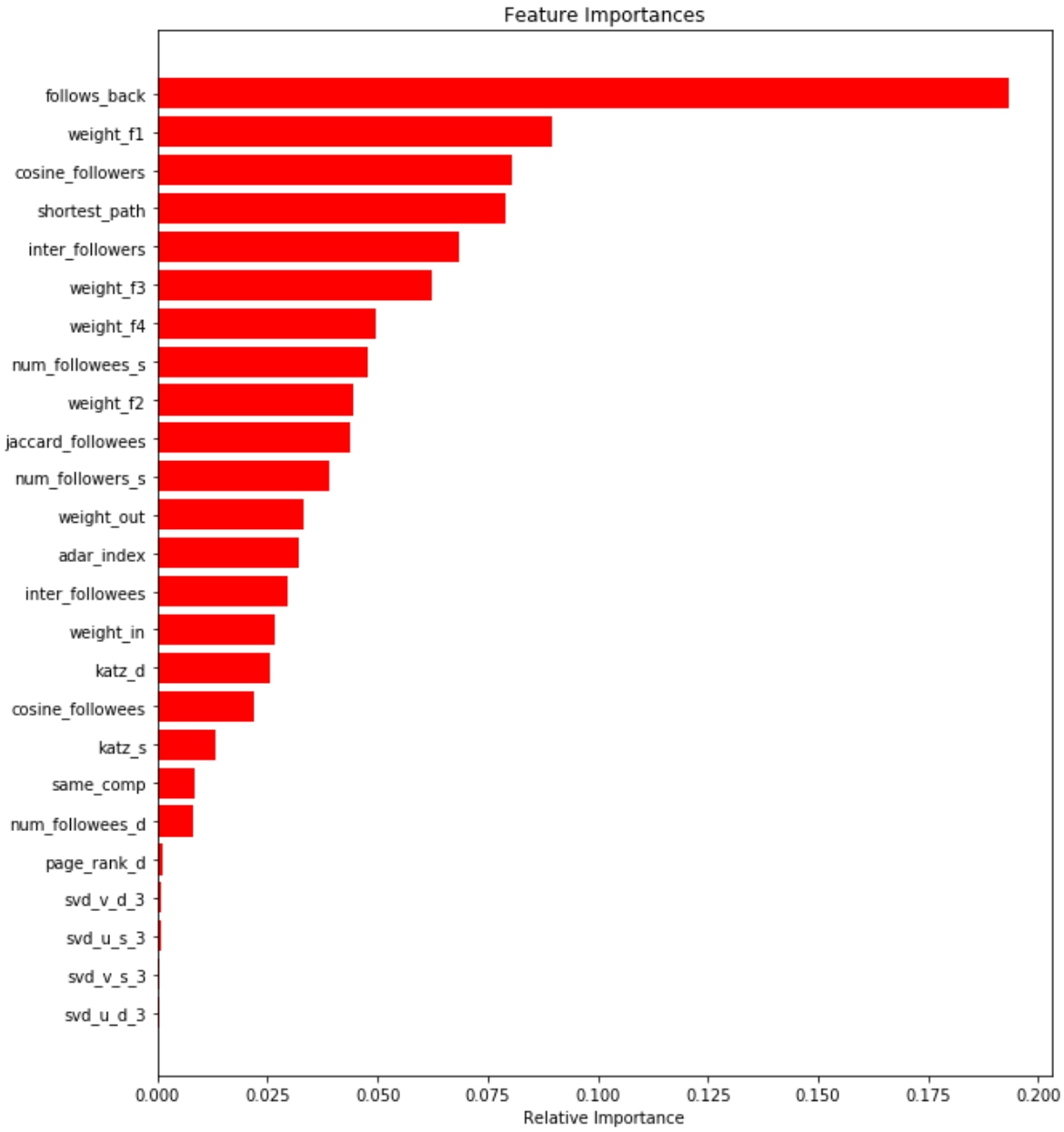
In [0]:

```
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 [0]:

```
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()
```



Applying XGBOOST Classifier

In [0]:

```
import xgboost as xgb
```

In [18]:

```
from sklearn.metrics import f1_score
from sklearn.model_selection import GridSearchCV


params = {"n_estimators":[5,10,20,35,50,75,100,150],
              "max_depth": [2,5,10,15,25]}

clf = xgb.XGBClassifier()

xg_grid = GridSearchCV(clf, param_grid=params, scoring='f1', verbose = 1, return_train_scor

xg_grid.fit(df_final_train,y_train)

train_error = xg_grid.cv_results_['mean_train_score']
cv_error = xg_grid.cv_results_['mean_test_score']

print('mean test scores',cv_error)
print('mean train scores',train_error)
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
```

```
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  6.7min
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
```

```
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executo
r.py:706: UserWarning: A worker stopped while some jobs were given to the ex
ecutor. This can be caused by a too short worker timeout or by a memory lea
k.
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed: 43.3min finished

mean test scores [0.9093935  0.91520883 0.91944168 0.9260257  0.94300771 0.9
5832148
 0.96438198 0.97150018 0.92818631 0.9301223  0.96118415 0.96817463
 0.97193336 0.97373327 0.97526257 0.97765709 0.97265926 0.97331634
 0.97473276 0.97598237 0.97656874 0.97802458 0.97908447 0.98060856
 0.97285756 0.97413794 0.97535537 0.97646277 0.97732781 0.97848266
 0.9791213  0.97974681 0.97267874 0.97407814 0.97555716 0.97657357
 0.97720709 0.97817292 0.97873396 0.97938576]
mean train scores [0.91019882 0.91563653 0.91996065 0.92644195 0.94388957 0.
95896703
 0.9649739  0.97185908 0.9283499  0.93014524 0.96154266 0.9683186
 0.97245427 0.97462561 0.9768479  0.98118547 0.97717446 0.97834334
 0.98039857 0.98240957 0.98350513 0.9869836  0.99225097 0.9987748
 0.9883703  0.99030902 0.99390048 0.9966334  0.99778277 0.99919519
 0.99994505 1.         0.99313963 0.99586417 0.99859558 0.99972532
 0.99991509 0.99998502 1.         1.        ]
```

In [19]:

```
print(xg_grid.best_estimator_)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=10,
              min_child_weight=1, missing=None, n_estimators=150, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

In [0]:

```
clf = xgb.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=10,
              min_child_weight=1, missing=None, n_estimators=150, n_jobs=-1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

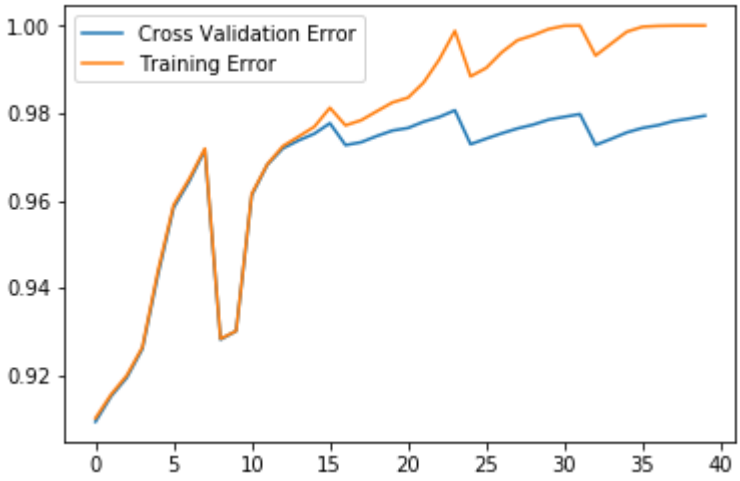
In [0]:

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

Error Plots

In [22]:

```
plt.plot(cv_error, label='Cross Validation Error')
plt.plot(train_error, label='Training Error')
plt.legend()
plt.show()
```



In [23]:

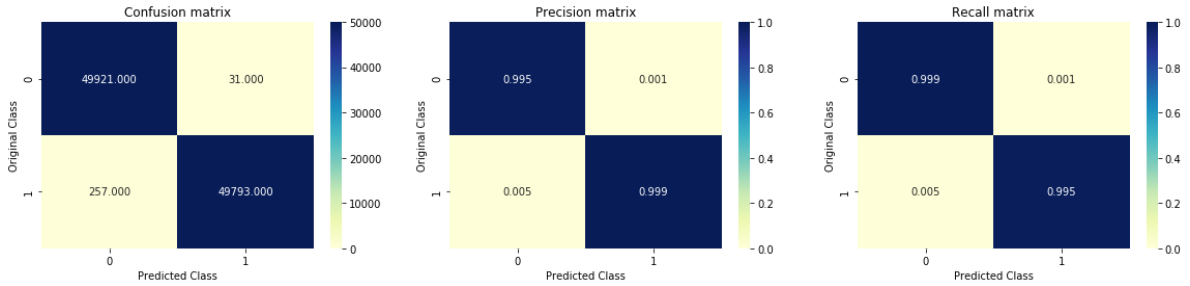
```
from sklearn.metrics import f1_score
print('Train f1 score',f1_score(y_train,y_train_pred))
print('Test f1 score',f1_score(y_test,y_test_pred))
```

```
Train f1 score 0.9971163666219436
Test f1 score 0.9257393116057572
```

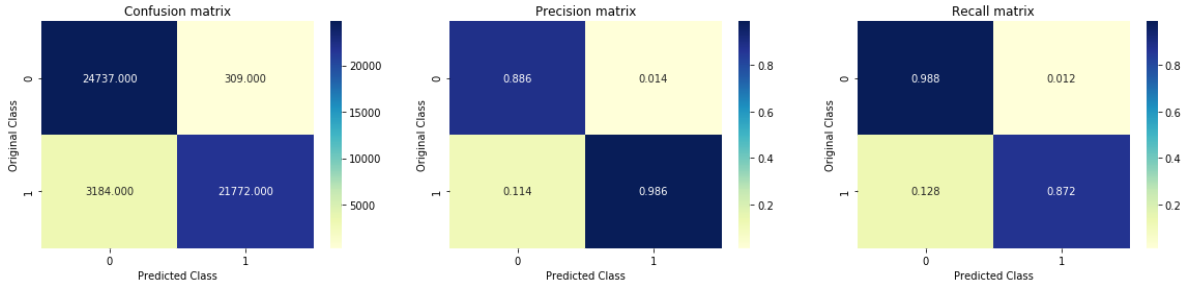
In [24]:

```
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

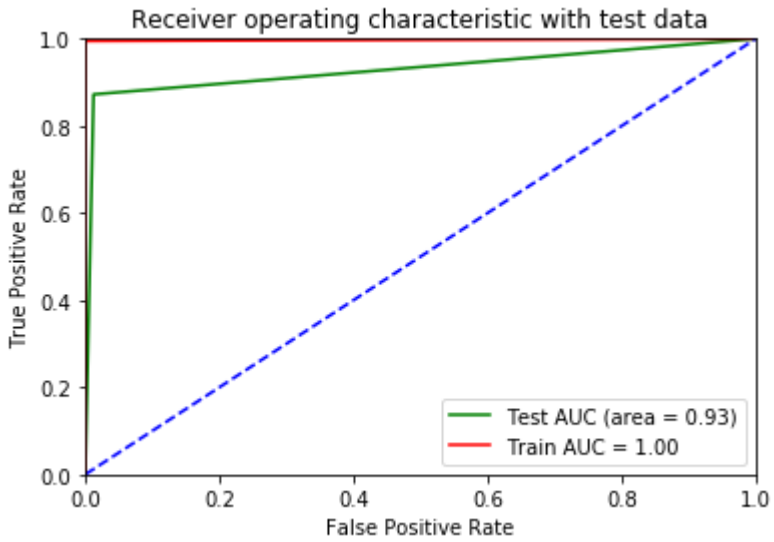


Test confusion_matrix



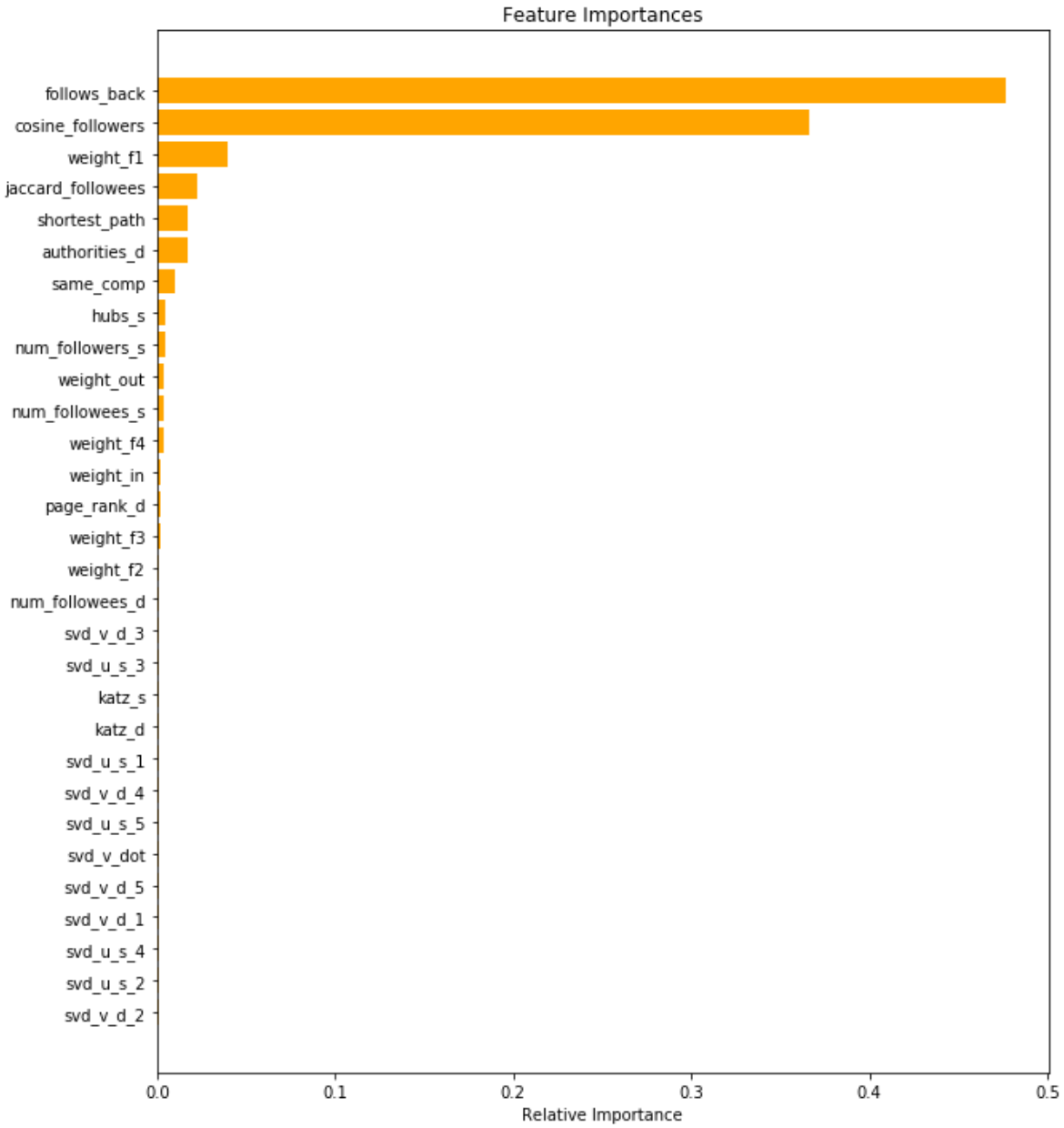
In [25]:

```
from sklearn.metrics import roc_curve, auc
fpr,tpr,ths = roc_curve(y_test,y_test_pred)
fpr2,tpr2,ths = roc_curve(y_train,y_train_pred)

auc_sc = auc(fpr, tpr)
auc_sc_train = auc(fpr2, tpr2)

plt.plot(fpr, tpr, color='green',label='Test AUC (area = %0.2f)' % auc_sc)
plt.plot(fpr2, tpr2, 'red', label = 'Train AUC = %0.2f' % auc_sc_train)

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.plot([0, 1], [0, 1],'b--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title('Receiver operating characteristic with test data')
plt.legend()
plt.show()
```



In [28]:

```
features = df_final_train.columns
importances = clf.feature_importances_
indices = (np.argsort(importances))[-30:]
plt.figure(figsize=(10,12))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='orange', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



- Clearly the features 'follows back' and 'cosine followers' are the most important among all others.

Conclusions:

In [0]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Algorithm", "Train AUC", "Test AUC"]

x.add_row(["Random Forest", 0.97, 0.93])
x.add_row(["XGBoost", 1.0, 0.93 ])


print(x)
```

```
+---------------+-----------+----------+
|   Algorithm   | Train AUC | Test AUC |
+---------------+-----------+----------+
| Random Forest |    0.97   |   0.93   |
|    XGBoost    |    1.0    |   0.93   |
+---------------+-----------+----------+
```

In [0]:

```

```