Social network Graph Link Prediction - Facebook Challenge

In [1]:

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

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

import warnings
import networkx as nx
import pdb
import pickle
from pandas import HDFStore,DataFrame
from pandas import read_hdf
from scipy.sparse.linalg import svds, eigs
import gc
from tqdm import tqdm
jhkuk
from sklearn.metrics import f1_score
```

In [2]:

```
!wget --header="Host: doc-0o-bk-docs.googleusercontent.com" --header="User-Agent: Mozilla/5
```

```
'wget' is not recognized as an internal or external command,
operable program or batch file.
```

In [2]:

```
#reading
from pandas import read_hdf
df_final_train = read_hdf('E:/applied ai/module 6/case study 3 facebook/data/fea_sample/sto
                          'train_df',mode='r')
df_final_test = read_hdf('E:/applied ai/module 6/case study 3 facebook/data/fea_sample/stor
                         'test_df',mode='r')
```

In [3]:

```
df_final_train.columns
```

Out[3]:

```
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', 'prefer_score_followees', 'prefer_score_followers',
       'num_followers_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'],
      dtype='object')
```

In [4]:

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

In [5]:

```
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
```

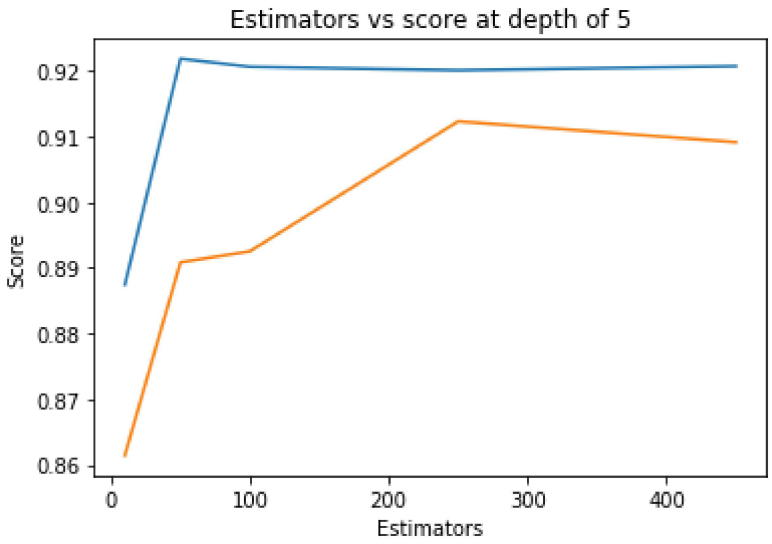
In [8]:

```
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_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 0.8874425563749064 test Score 0.861406690410120
9
Estimators =  50 Train Score 0.9218968212610735 test Score 0.890848763780370
3
Estimators =  100 Train Score 0.9206603488988985 test Score 0.89253133259442
41
Estimators =  250 Train Score 0.9201226635514018 test Score 0.91234624910792
99
Estimators =  450 Train Score 0.9207507820646508 test Score 0.90917512316308
06
```

Out[8]:

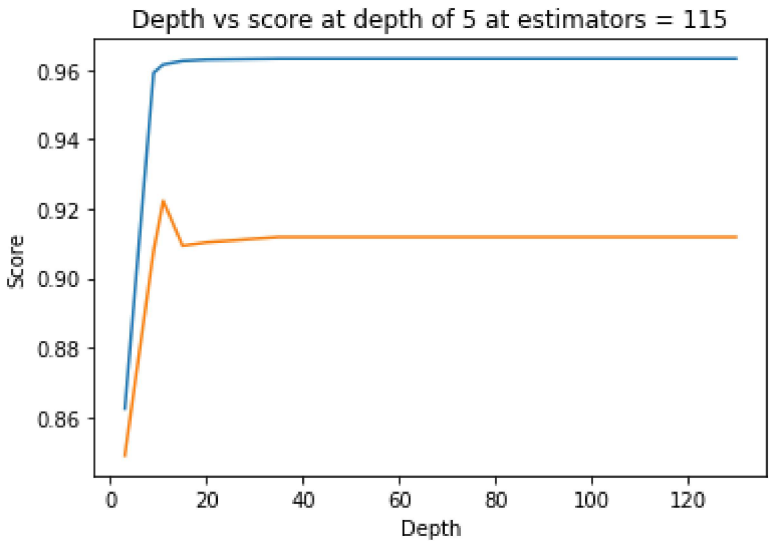
```
Text(0.5, 1.0, 'Estimators vs score at depth of 5')
```



In [9]:

```
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_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 0.8624922032751531 test Score 0.8489980967045188
depth =  9 Train Score 0.9591749557576127 test Score 0.9079691736225263
depth =  11 Train Score 0.961526367573818 test Score 0.9222376679725347
depth =  15 Train Score 0.9625684723067559 test Score 0.9093585968505953
depth =  20 Train Score 0.9629246631588553 test Score 0.9103386543338804
depth =  35 Train Score 0.9631949088992927 test Score 0.9118879928506075
depth =  50 Train Score 0.9631949088992927 test Score 0.9118879928506075
depth =  70 Train Score 0.9631949088992927 test Score 0.9118879928506075
depth =  130 Train Score 0.9631949088992927 test Score 0.9118879928506075
```



In [10]:

```
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_state=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'])
```

```
mean test scores [0.9621107  0.96176445 0.96083719 0.96150991 0.9627959 ]
```

In [11]:

```
print(rf_random.best_estimator_)
```

```
RandomForestClassifier(max_depth=14, min_samples_leaf=28, min_samples_split=
111,
                       n_estimators=121, n_jobs=-1, random_state=25)
```

In [12]:

```
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_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 [13]:

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

In [14]:

```
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 0.9640508281909711
Test f1 score 0.9110522493271244
```

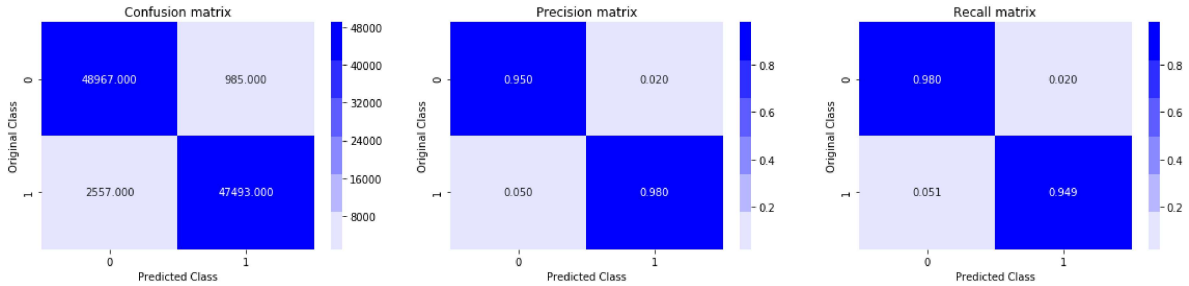
In [15]:

```
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()
```

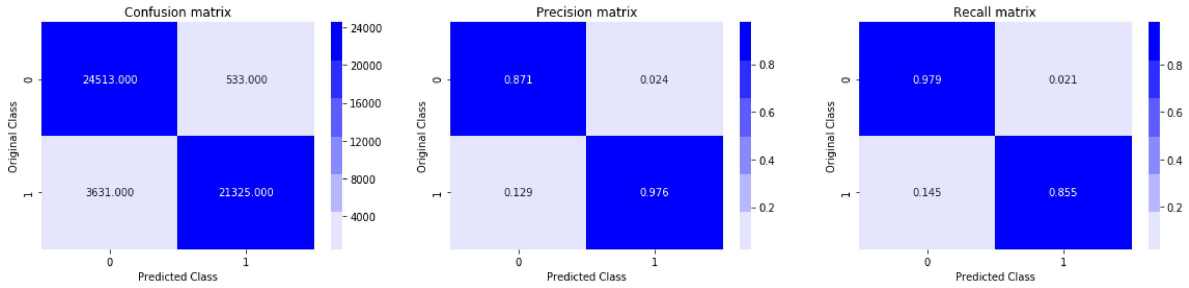
In [16]:

```
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
```

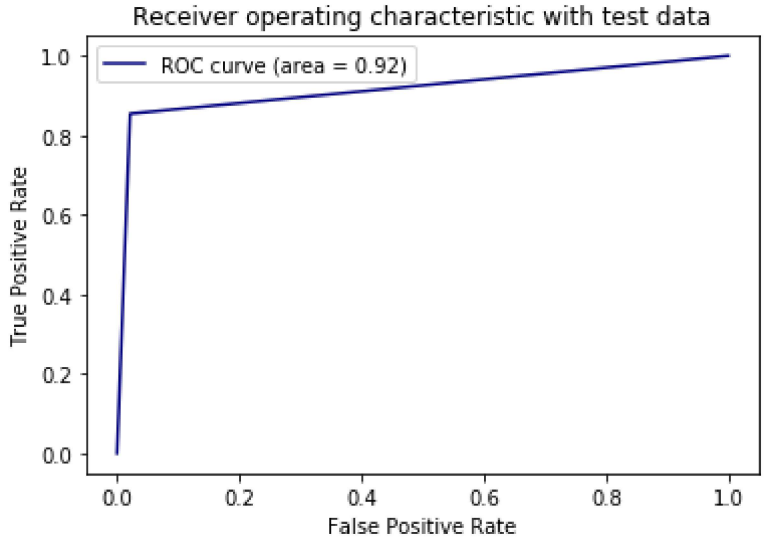



```
Test confusion_matrix
```



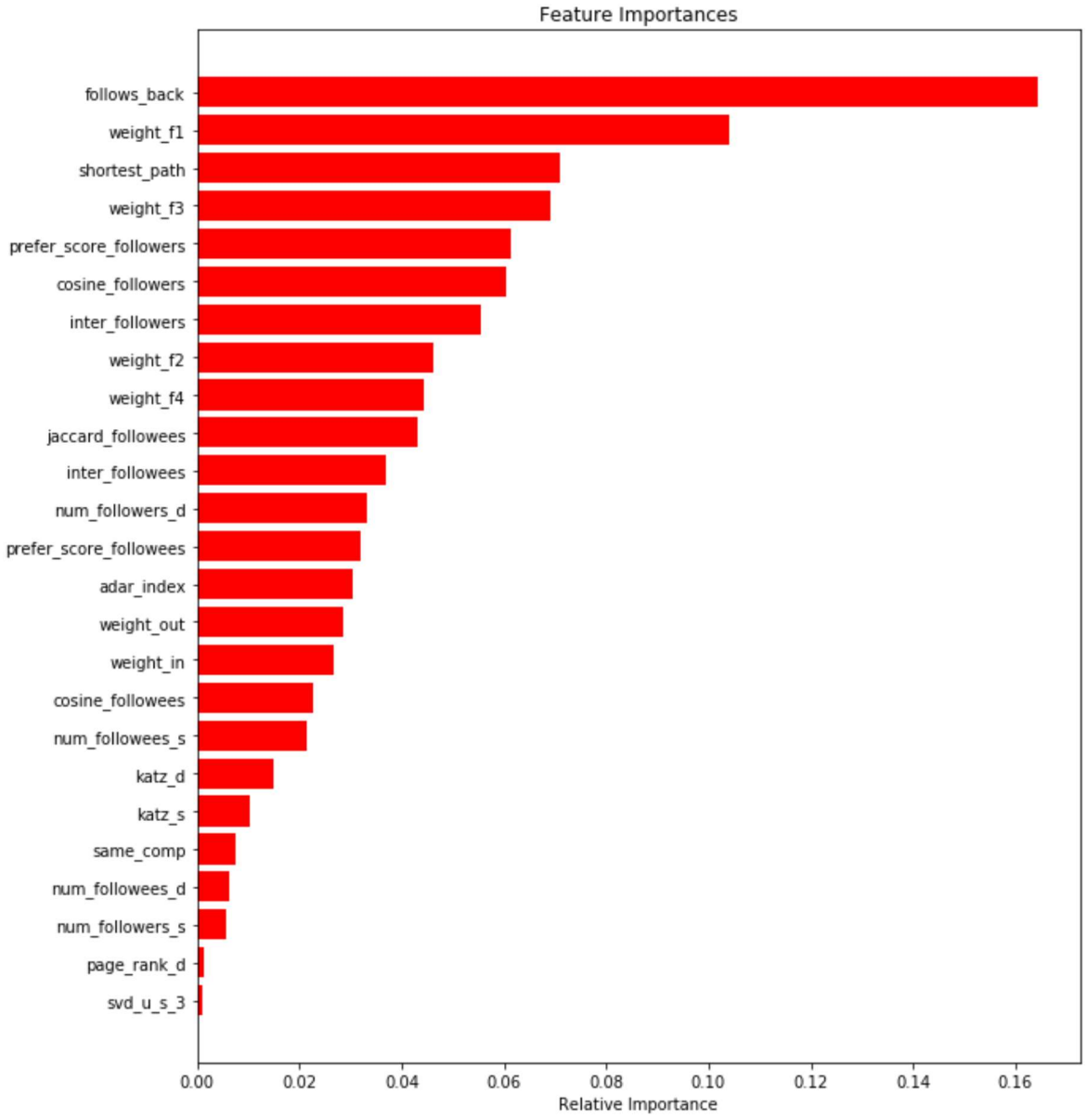
In [17]:

```
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 [18]:

```
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/)
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.

XG boost with all these features

In [6]:

```
from xgboost import XGBClassifier
```

In [22]:

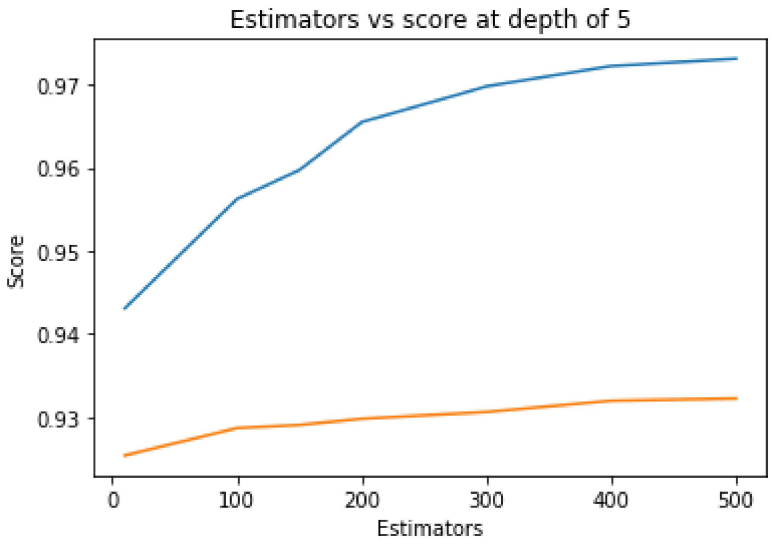
```
import warnings
warnings.filterwarnings("ignore")
estimators = [10,100,150,200,300,400,500]
train_scores = []
test_scores = []
for i in estimators:
    clf = XGBClassifier(base_score=0.5,booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.01, max_delta_step=0, max_depth=5t,
              min_child_weight=1,monotone_constraints='()',
              n_estimators=i, n_jobs=4, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
    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')
```

```
[11:06:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Estimators =  10 Train Score 0.9431101724155958 test Score 0.925428003505247
1
[11:06:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Estimators =  100 Train Score 0.9562459319564836 test Score 0.92871898626344
36
[11:06:38] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Estimators =  150 Train Score 0.9597189599043969 test Score 0.92909431392577
46
[11:07:26] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Estimators =  200 Train Score 0.9654861637410131 test Score 0.92983834993733
68
[11:08:36] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
```

```
Estimators =  300 Train Score 0.9697711668145972 test Score 0.93065228469509
09
[11:10:11] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Estimators =  400 Train Score 0.972210067872152 test Score 0.93200754764984
[11:12:19] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Estimators =  500 Train Score 0.9730965860004676 test Score 0.93227412350386
61
```

Out[22]:

```
Text(0.5, 1.0, 'Estimators vs score at depth of 5')
```

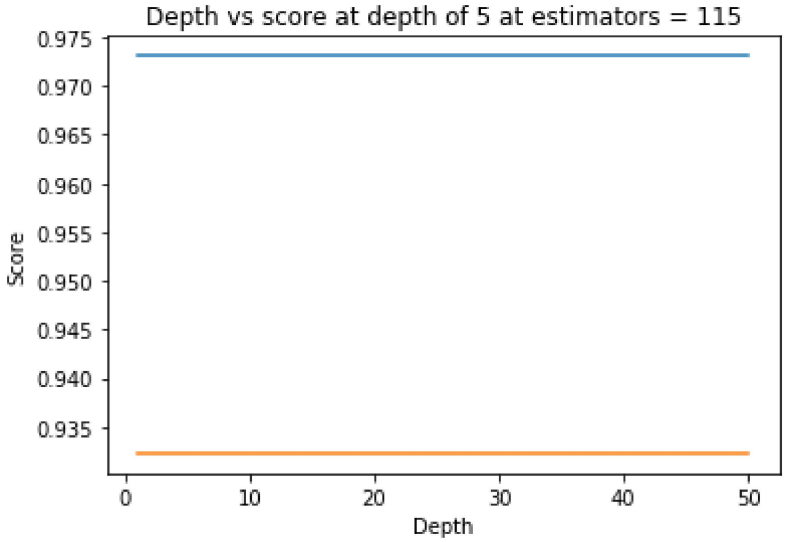


In [28]:

```
import warnings
warnings.filterwarnings("ignore")
depths = [1,5,10,15,25,50]
train_scores = []
test_scores = []
for i in depths:
    clf1 = XGBClassifier(base_score=0.5,booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.01, max_delta_step=0, max_depth=i,
              min_child_weight=1,monotone_constraints='()',
              n_estimators=500, n_jobs=4, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
    clf1.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 estimators = 500')
plt.show()
```

```
[12:11:42] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
depth =  1 Train Score 0.9730965860004676 test Score 0.9322741235038661
[12:12:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
depth =  5 Train Score 0.9730965860004676 test Score 0.9322741235038661
[12:14:44] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
depth =  10 Train Score 0.9730965860004676 test Score 0.9322741235038661
[12:18:52] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
depth =  15 Train Score 0.9730965860004676 test Score 0.9322741235038661
[12:24:14] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
depth =  25 Train Score 0.9730965860004676 test Score 0.9322741235038661
[12:31:13] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
```

```
_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
depth =  50 Train Score 0.9730965860004676 test Score 0.9322741235038661
```



In [54]:

```
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from scipy.stats import uniform
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
```

In [59]:

```
parameters = {"n_estimators":(505,510,515,520),
              "max_depth":(1,2,3,4,5)}

clf2 = XGBClassifier(random_state=25,n_jobs=-1,learning_rate=0.01)

xgbc = GridSearchCV(clf2, parameters,
                                    cv=3, scoring = 'roc_auc',n_jobs=-1)

xgbc.fit(df_final_train,y_train)
print('mean test scores',xgbc.cv_results_['mean_test_score'])
```

```
[18:43:25] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
mean test scores [0.97656796 0.97666072 0.9767313  0.97685624 0.99048833 0.9
905708
 0.99061858 0.99073994 0.99409991 0.99414056 0.99416987 0.99421443
 0.99559142 0.99562014 0.99564162 0.99566952 0.99628762 0.99631364
 0.99634371 0.99636756]
```

In [60]:

```
print(xgbc.best_estimator_)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=Fal
se,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.01, max_delta_step
=0,
              max_depth=5, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=520, n_jobs=-1,
              num_parallel_tree=1, predictor='auto', random_state=25,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [63]:

```
clf3 =XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.01, max_delta_step=0,
              max_depth=5, min_child_weight=1,
              monotone_constraints='()', n_estimators=520, n_jobs=-1,
              num_parallel_tree=1, predictor='auto', random_state=25,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [64]:

```
clf3.fit(df_final_train,y_train)
y_train_pred1 = clf3.predict(df_final_train)
y_test_pred1 = clf3.predict(df_final_test)
```

```
[19:06:06] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
```

In [65]:

```
print('Train f1 score',f1_score(y_train,y_train_pred1))
print('Test f1 score',f1_score(y_test,y_test_pred1))
```

```
Train f1 score 0.9722965847327006
Test f1 score 0.9317625200574277
```

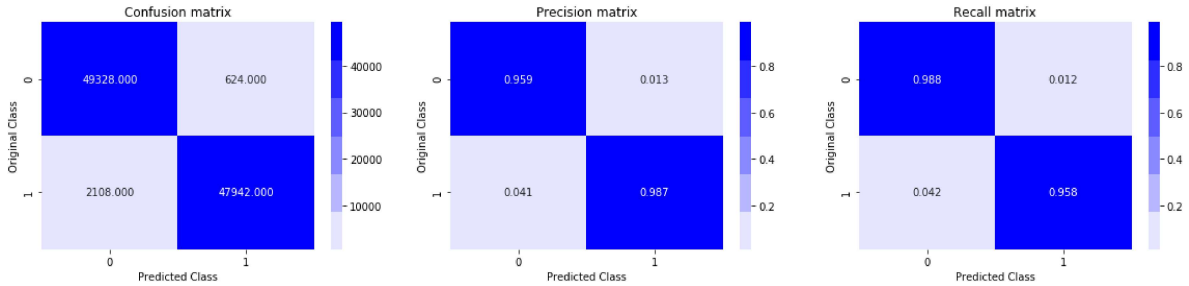
In [66]:

```
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()
```

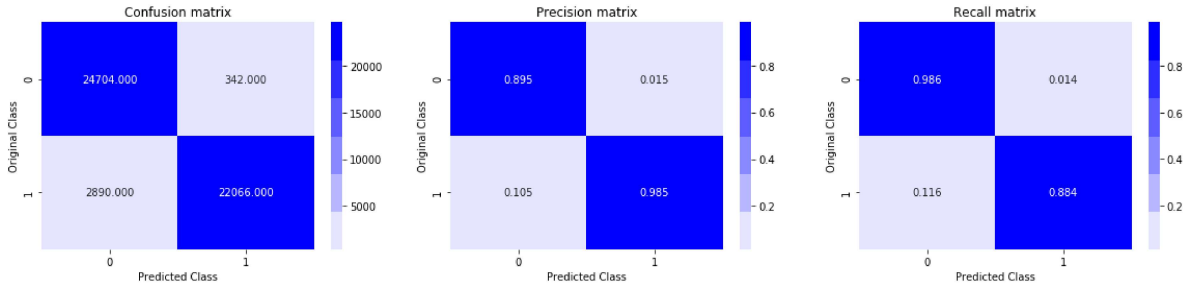
In [67]:

```
print('Train confusion_matrix')
plot_confusion_matrix(y_train,y_train_pred1)
print('Test confusion_matrix')
plot_confusion_matrix(y_test,y_test_pred1)
```

```
Train confusion_matrix
```

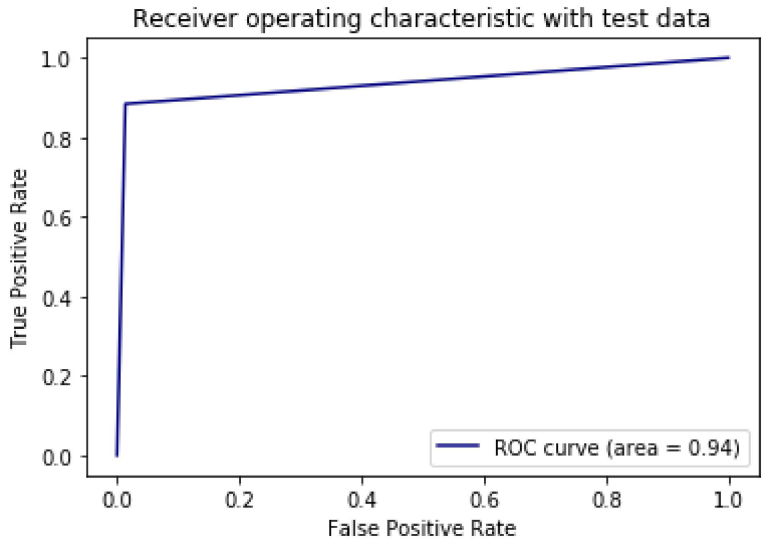



```
Test confusion_matrix
```



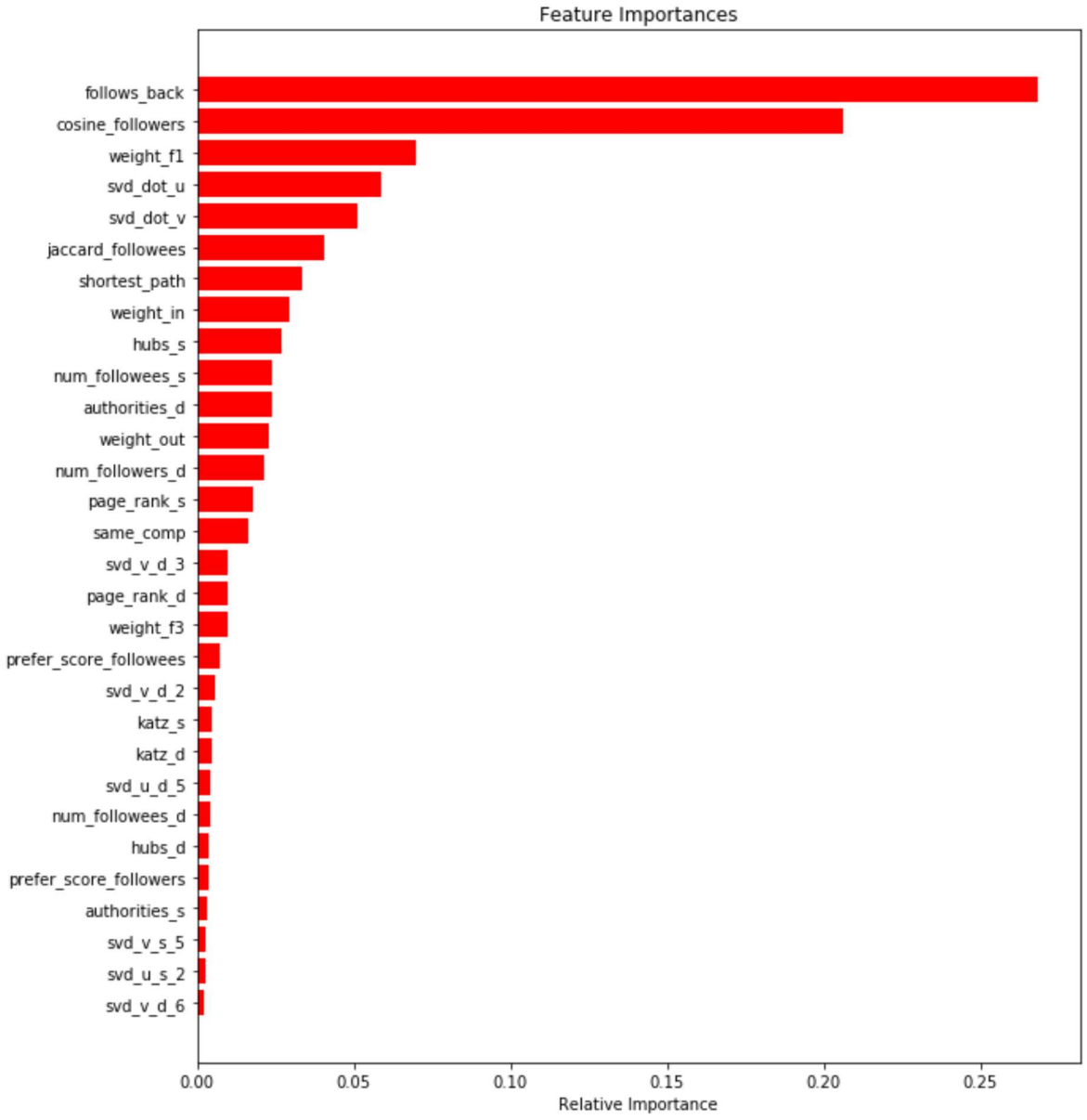
In [68]:

```
fpr1,tpr1,ths1 = roc_curve(y_test,y_test_pred1)
auc_sc1 = auc(fpr1, tpr1)
plt.plot(fpr1, tpr1, color='navy',label='ROC curve (area = %0.2f)' % auc_sc1)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic with test data')
plt.legend()
plt.show()
```



In [69]:

```
features = df_final_train.columns
importances_xgb = clf3.feature_importances_
indices = (np.argsort(importances_xgb))[-30:]
plt.figure(figsize=(10,12))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances_xgb[indices], color='r', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In [ ]:

In [ ]: