Social network Graph Link Prediction - Facebook Challenge

```
#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
```

```
!pip install gdown
```

```
    Requirement already satisfied: gdown in /usr/local/lib/python3.7/dist-packages (4.4.0)
    Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from gdown) (3.7.0)
    Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from gdown) (4.64.0)
    Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-packages (from gdown) (4.6.3)
    Requirement already satisfied: requests[socks] in /usr/local/lib/python3.7/dist-packages (from gdown) (2.23.0)
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from gdown) (1.15.0)
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown) (20
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown) (2.10)
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown) (3.0
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from reques
    Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown)
```

```
# !gdown --folder https://drive.google.com/drive/folders/1c50Q5RcmdpMYj1jCPc3ShOE2y4G8G2ez
```

```
    Retrieving folder list
    Retrieving folder 1pEBmIl1tbuwYrsfeUv2KDz8a2gR0Fjxe data
    Retrieving folder 1ONG0P5YAMlkzyKB7VugPKtLs9fXcdmRx after_eda
    Processing file 1XNxqMI6qyXYhi2XVVRIxqD11hKgh2oja missing_edges_final.p
    Processing file 1_KN7S8zfHdrkRjRYOEtBxBVq8JrGxPXD test_after_eda.csv
    Processing file 1yynFkCz80RcbTgPylw8cdZ37qEsfxmGV test_neg_after_eda.csv
    Processing file 1yldY87HoO-XyfCJqyFqyi7QKeJkhMXgB test_pos_after_eda.csv
    Processing file 1lcxzVZ0-MkPmoH3lS35Q8rRfrecKSXb1 train_after_eda.csv
    Processing file 1qNhcor22jE5gJ_NfKATx_18oKYOKPy7_ train_neg_after_eda.csv
    Processing file 1XLHsIRXKLx9TA9nuC1SS7JDkLyRVmo69 train_pos_after_eda.csv
    Processing file 1c5omWa9D1b4iQ28tiDfMs4wglhKKLA9X train_woheader.csv
    Retrieving folder 1qYtDPghLMT6rv3xd7NmQUSUKWwCS5375 fea_sample
    Processing file 1YVVHZvqfopWwLeAdIu5-oHGocS7CmCZR hits.p
    Processing file 1Xp7QmNsdVF6BN0IumLGlfUOyr3IAJyjn katz.p
    Processing file 1hp-5BFw9xK1WmovBW17T5POa1BbGSk9N page_rank.p
    Processing file 1pIO_nOg9XU0WUD10brRvrgyUXbY5gqMs storage_sample_stage1.h5
    Processing file 10qJ04GRcaDxc16gmJXb8rpGPmlyys7E2 storage_sample_stage2.h5
    Processing file 10M-HFdUMelv6HT6m4RacJJWbNjQ_PkjC storage_sample_stage3.h5
    Processing file 1fDJptlCFEWNV5UNGPc4geTykgFI3PDCV storage_sample_stage4.h5
    Processing file 1H6qybuXr8i_USWu3k3ulXEOurc-SElUh test_y.csv
    Processing file 19mviN_yeJIfakb4kU5NfKdQlOQtaQ-kH train_y.csv
    Processing file 1l1adJnTgeHULVuoLdTRyqi25-z6shBSL train.csv
    Processing file 1kVpz3lays4hw-tzyQmjfZ4YuhGzbV2ca FB_EDA.ipynb
    Processing file 1rQgKqdUvwGvHpWUYCs1b9sIRGYgsQJT4 FB_featurization.ipynb
    Processing file 1HI7l4Bxi-Mi5TzeQmiShPDhs34F2LDFN FB_Models.ipynb
    Retrieving folder list completed
    Building directory structure
    Building directory structure completed
    Downloading...
    From: https://drive.google.com/uc?id=1XNxqMI6qyXYhi2XVVRIxqD11hKgh2oja
    To: /content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/after_eda/missing_edges_fin
    100% 150M/150M [00:01<00:00, 136MB/s]
    Downloading...
    From: https://drive.google.com/uc?id=1_KN7S8zfHdrkRjRYOEtBxBVq8JrGxPXD
    To: /content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/after_eda/test_after_eda.cs
    100% 59.7M/59.7M [00:00<00:00, 153MB/s]
    Downloading...
    From: https://drive.google.com/uc?id=1yynFkCz80RcbTgPylw8cdZ37qEsfxmGV
    To: /content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/after_eda/test_neg_after_ed
    100% 29.8M/29.8M [00:00<00:00, 71.8MB/s]
    Downloading...
    From: https://drive.google.com/uc?id=1yldY87HoO-XyfCJqyFqyi7QKeJkhMXgB
```

```
100% 29.8M/29.8M [00:00<00:00, 54.3MB/s]
Downloading...
From: https://drive.google.com/uc?id=1lcxzVZ0-MkPmoH31S35Q8rRfrecKSXb1
To: /content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/after_eda/train_after_eda.c
100% 239M/239M [00:02<00:00, 84.2MB/s]
Downloading...
From: https://drive.google.com/uc?id=1qNhcor22jE5gJ_NfKATx_18oKYOKPy7_
To: /content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/after_eda/train_neg_after_e
100% 119M/119M [00:01<00:00, 82.6MB/s]
Downloading...
From: https://drive.google.com/uc?id=1XLHsIRXKLx9TA9nuC1SS7JDkLyRVmo69
To: /content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/after_eda/train_pos_after_e
100% 119M/119M [00:01<00:00, 89.2MB/s]
Downloading...
From: https://drive.google.com/uc?id=1c5omWa9D1b4iQ28tiDfMs4wglhKKLA9X
```

```
# !wget --header="Host: doc-0o-bk-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Apple
```

```
--2022-05-20 12:09:49--  https://doc-0o-bk-docs.googleusercontent.com/docs/securesc/nss2f5s2soorprev6d4t4qp3n5ekp9nh/evl2j2j4
Resolving doc-0o-bk-docs.googleusercontent.com (doc-0o-bk-docs.googleusercontent.com)... 108.177.12.132, 2607:f8b0:400c:c08::
Connecting to doc-0o-bk-docs.googleusercontent.com (doc-0o-bk-docs.googleusercontent.com)|108.177.12.132|:443... connected.
HTTP request sent, awaiting response... 403 Forbidden
2022-05-20 12:09:49 ERROR 403: Forbidden.
```

```
! pwd
%cd /content/Facebook/data/fea_sample/Facebook/data/fea_sample
```

```
/content/Facebook/data/fea_sample
/content/Facebook/data/fea_sample/Facebook/data/fea_sample
```

```
#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')
```

```
df_final_train.columns
```

```
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_index',
       'follows_back', 'same_comp', 'shortest_path', 'weight_in', 'weight_out',
       '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')
```

```
type(df_final_train)
```

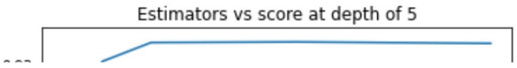
```
pandas.core.frame.DataFrame
```

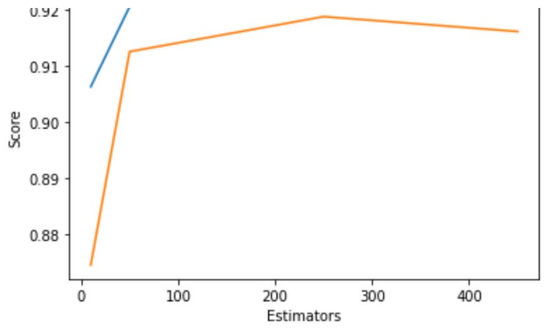
```
y_train = df_final_train.indicator_link
y_test = df_final_test.indicator_link
```

```
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)
```

```
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=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')
```

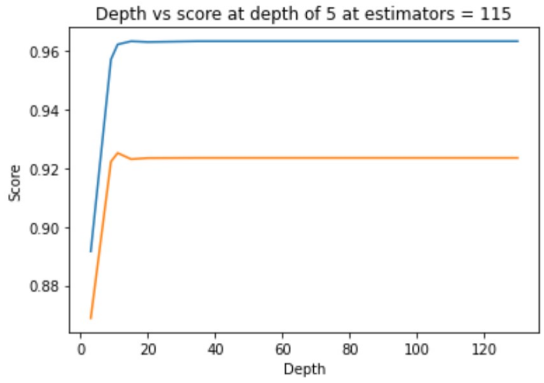
```
Estimators =  10 Train Score 0.9063252121775113 test Score 0.8745605278006858
Estimators =  50 Train Score 0.9205725512208812 test Score 0.9125653355634538
Estimators =  100 Train Score 0.9238690848446947 test Score 0.9141199714153599
Estimators =  250 Train Score 0.9239789348046863 test Score 0.9188007232664732
Estimators =  450 Train Score 0.9237190618658074 test Score 0.9161507685828595
Text(0.5, 1.0, 'Estimators vs score at depth of 5')
```





```
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,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 =  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
```



```
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, return_train_score=True)
# https://stackoverflow.com/questions/57136676/sklearn-model-selection-gridsearchcv-is-throwing-keyerror-mean-train-score
rf_random.fit(df_final_train,y_train)
```

```
RandomizedSearchCV(cv=10,
                   estimator=RandomForestClassifier(n_jobs=-1, random_state=25),
                   n_iter=5,
                   param_distributions={'max_depth': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7f2d909f9350>,
                                        'min_samples_leaf': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7f2d90a6
                                        'min_samples_split': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7f2d909
                                        'n_estimators': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7f2d909e991
                   random_state=25, return_train_score=True, scoring='f1')
```

```
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 [0.96225042 0.96215492 0.9605708  0.96194014 0.96330005]
mean train scores [0.96294922 0.96266735 0.96115674 0.96263457 0.96430539]
```

```
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)
```

```
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)
```

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

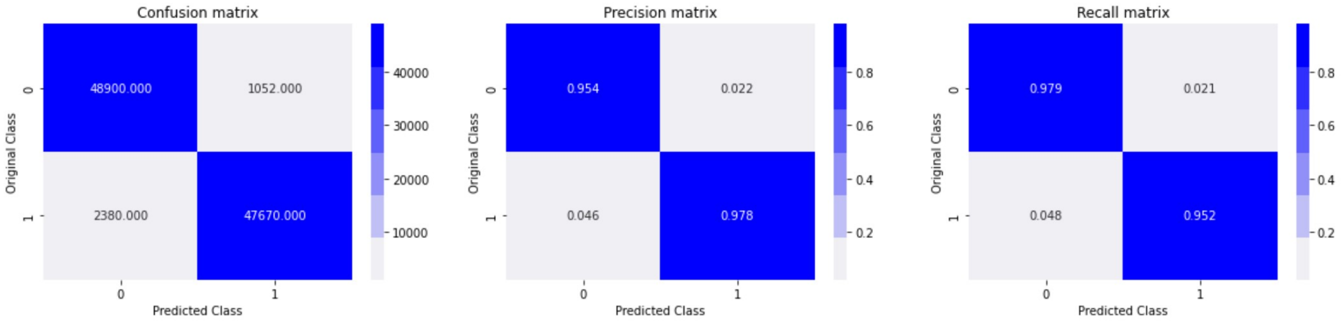
```
from sklearn.metrics import f1_score
print('Train f1 score',f1_score(y_train,y_train_pred))
print('Test f1 score',f1_score(y_test,y_test_pred))
```

```
Train f1 score 0.9652533106548414
Test f1 score 0.9241678239279553
```

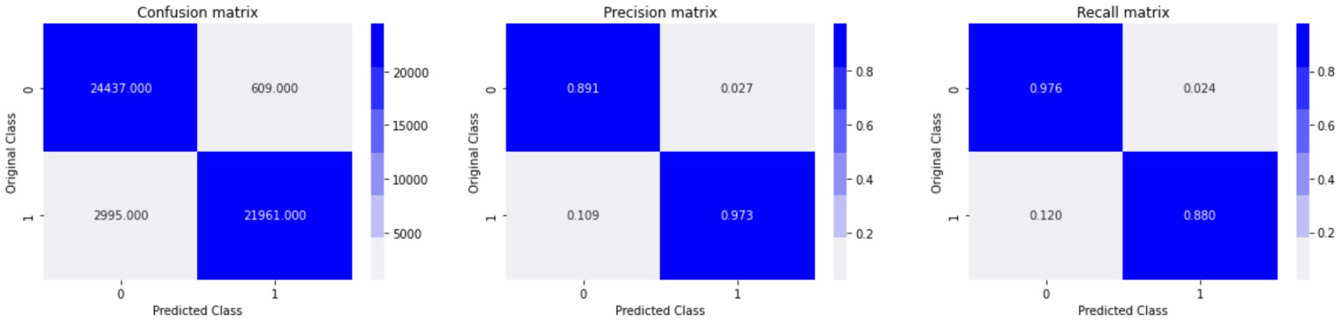
```
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)

    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()
```

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

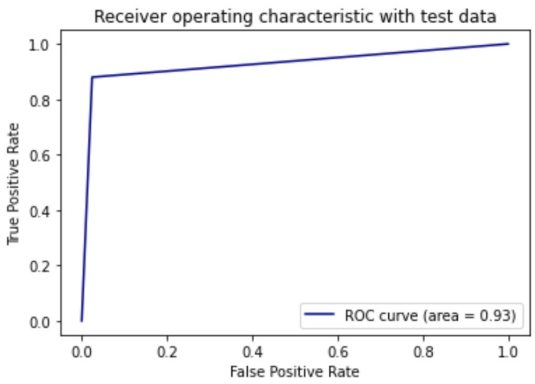
```
Train confusion_matrix
```



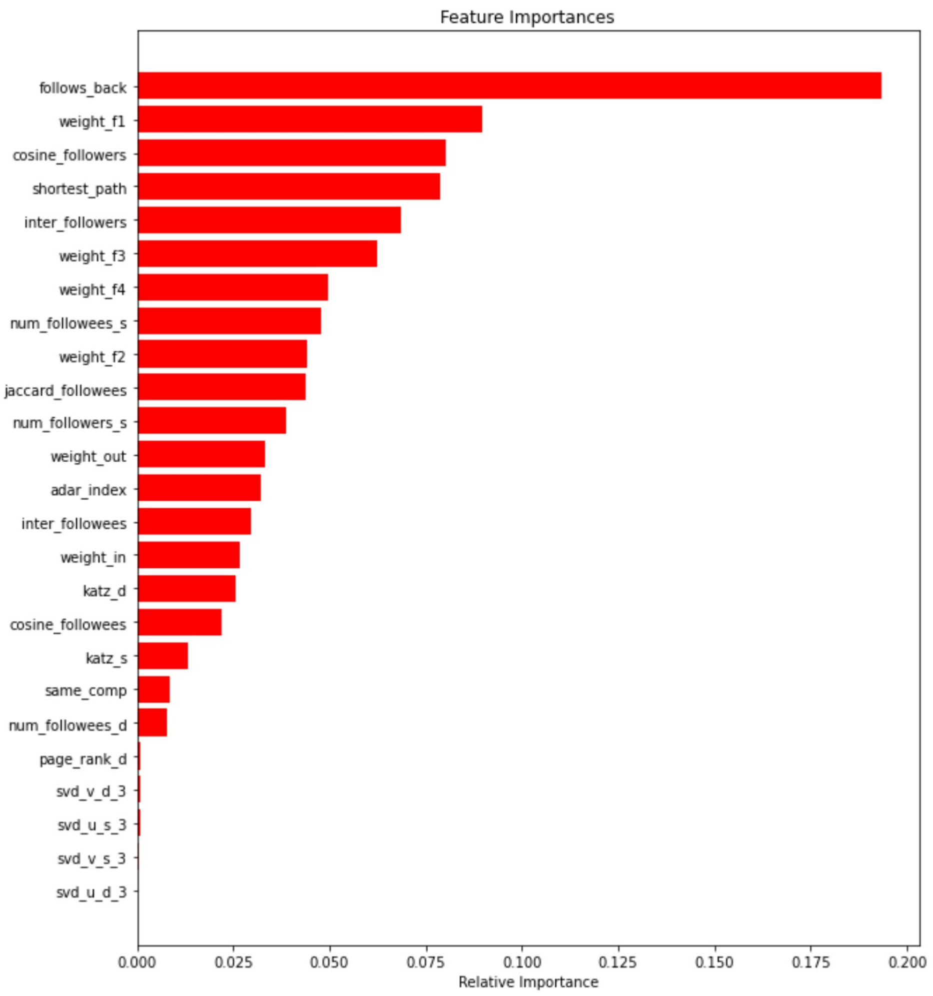
```
Test confusion_matrix
```



```
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()
```



```
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/
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
3. Tune hyperparameters for XG boost with all these features and check the error metric.

```
from google.colab import drive
from google.colab import drive
drive.mount('/content/gdrive/')
! pwd
%cd /content/gdrive/My\ Drive/Assignments\ AAIC/Assignment\ 17\ FB\ Friend\ Recommendation
```

```
Drive already mounted at /content/gdrive/; to attempt to forcibly remount, call drive.mount("/content/gdrive/", force_remount
/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation
/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation
```

```
if os.path.isfile('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/after_eda/train_p
    train_graph=nx.read_edgelist('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Facebook/data/af
    print(nx.info(train_graph))
else:
    print("please run the FB_EDA.ipynb or download the files from drive")
```

```
DiGraph with 1780722 nodes and 7550015 edges
```

```
#reading
from pandas import read_hdf
!pwd
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')
```

```
/content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation
```

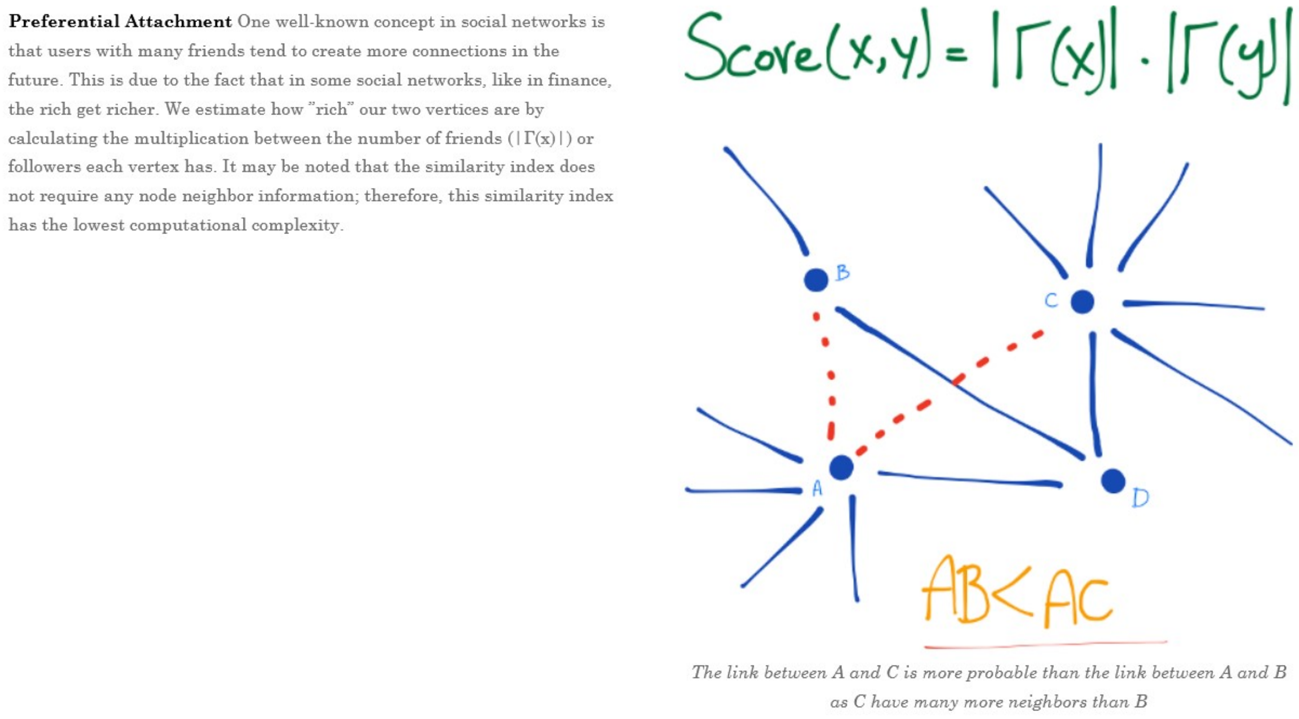
```
df_final_train.columns
```

```
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_index',
       'follows_back', 'same_comp', 'shortest_path', 'weight_in', 'weight_out',
       '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')
```

Note:- It seems num_follower_d feature is missing from the storage_sample_stage4, so we'll have to add it.

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

1. Adding Preferential Attachment



Preferential Attachment One well-known concept in social networks is that users with many friends tend to create more connections in the future. This is due to the fact that in some social networks, like in finance, the rich get richer. We estimate how "rich" our two vertices are by calculating the multiplication between the number of friends (|Γ(x)|) or followers each vertex has. It may be noted that the similarity index does not require any node neighbor information; therefore, this similarity index has the lowest computational complexity.

The link between A and C is more probable than the link between A and B as C have many more neighbors than B

```
def compute_features_stage5(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=[]
    # See if node in Train or Test
    for i,row in df_final.iterrows():
        try:
            s1=set(train_graph.predecessors(row['source_node'])) # source followers
            s2=set(train_graph.successors(row['source_node'])) # people following source
        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_d
```

```
df_final_train.head()
```

	source_node	destination_node	indicator_link	jaccard_followers	jaccard_followees	cosine_followers	cosine_followees	nu
0	273084	1505602	1	0	0.000000	0.000000	0.000000	
1	832016	1543415	1	0	0.187135	0.028382	0.343828	
2	1325247	760242	1	0	0.369565	0.156957	0.566038	
3	1368400	1006992	1	0	0.000000	0.000000	0.000000	
4	140165	1708748	1	0	0.000000	0.000000	0.000000	

5 rows × 54 columns

```
df_final_train['num_followers_d']= compute_features_stage5(df_final_train)
df_final_test['num_followers_d'] = compute_features_stage5(df_final_test)
```

```
! pwd
%cd  /content/gdrive/My·Drive/Assignments·AAIC/Assignment·17·FB·Friend·Recommendation/
```

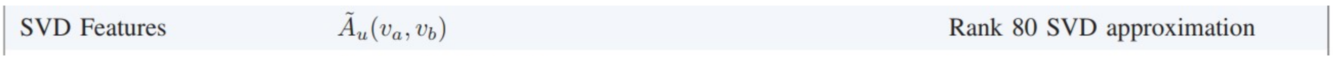
```
/content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation
/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation
```

```
# Create a file storage sample Stage 5 and add Two New Features ie Preferential Attachment and svd_dot
if not os.path.isfile('storage_sample_stage5.h5'):
    #mapping Preferential Attachment on train ie followers of source * followers of destination
    df_final_train['preferential'] = df_final_train.apply(lambda row: row['num_followers_s']* row['num_followers_d'],axis=1)
    #mapping adar index on test
    df_final_test['preferential'] = df_final_test.apply(lambda row: row['num_followers_s']*row['num_followers_d'],axis=1)

    #----------------------------------------------------------------------------------------------------

    hdf = HDFStore('storage_sample_stage5.h5')
    hdf.put('train_df',df_final_train, format='table', data_columns=True)
    hdf.put('test_df',df_final_test, format='table', data_columns=True)
    hdf.close()
else: # directly read a file
    df_final_train = read_hdf('storage_sample_stage5.h5', 'train_df',mode='r')
    df_final_test = read_hdf('storage_sample_stage5.h5', 'test_df',mode='r')
```

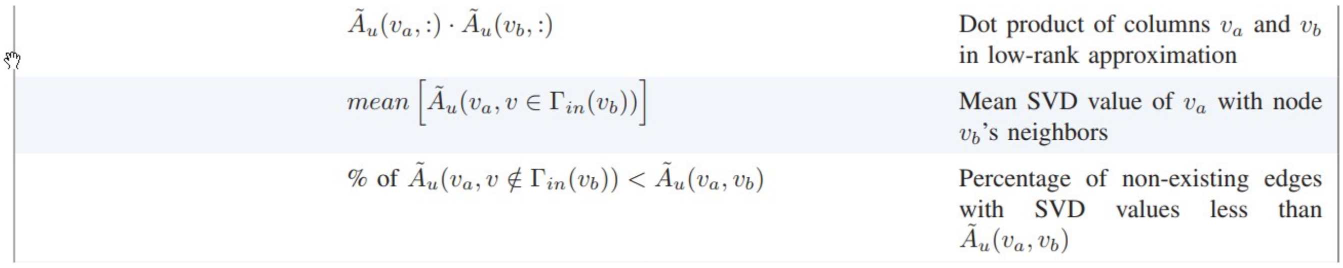
```
df_final_train.columns
```

```
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_index',
       'follows_back', 'same_comp', 'shortest_path', 'weight_in', 'weight_out',
       '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',
       'num_followers_d', 'preferential'],
      dtype='object')
```

Add svd_dot Feature



SVD Features	$\tilde{A}_u(v_a, v_b)$	Rank 80 SVD approximation



$\tilde{A}_u(v_a,:)\cdot\tilde{A}_u(v_b,:)$	Dot product of columns v_a and v_b in low-rank approximation
$mean\left[\tilde{A}_u(v_a, v\in\Gamma_{in}(v_b))\right]$	Mean SVD value of v_a with node v_b's neighbors
$\%$ of $\tilde{A}_u(v_a, v\notin\Gamma_{in}(v_b)) < \tilde{A}_u(v_a,v_b)$	Percentage of non-existing edges with SVD values less than $\tilde{A}_u(v_a,v_b)$

```
#mapping svd_dot on train
# [[ ]] is used to get Multiple rows https://stackoverflow.com/questions/49629992/how-to-select-multiple-rows-in-a-pandas-column-to

df_final_train['svd_dot_u']=df_final_train.apply(lambda row:np.dot(row[['svd_u_s_1','svd_u_s_2','svd_u_s_3', 'svd_u_s_4', 'svd_u_s_
                                                                     row[['svd_u_d_1', 'svd_u_d_2', 'svd_u_d_3', 'svd_u_d_4', 'svd_u
df_final_train['svd_dot_v']=df_final_train.apply(lambda row:np.dot(row[['svd_v_s_1','svd_v_s_2', 'svd_v_s_3', 'svd_v_s_4', 'svd_v_s
                                                                     row[['svd_v_d_1', 'svd_v_d_2', 'svd_v_d_3', 'svd_v_d_4', 'svd_v

#----------------------------------------------------------------------------------------------------------------------------------

#mapping svd_dot on test
df_final_test['svd_dot_u']=df_final_test.apply(lambda row:np.dot(row[['svd_u_s_1', 'svd_u_s_2','svd_u_s_3', 'svd_u_s_4', 'svd_u_s_5
                                                                   row[['svd_u_d_1', 'svd_u_d_2', 'svd_u_d_3', 'svd_u_d_4', 'svd_u_d
df_final_test['svd_dot_v']=df_final_test.apply(lambda row:np.dot(row[['svd_v_s_1', 'svd_v_s_2', 'svd_v_s_3', 'svd_v_s_4', 'svd_v_s_
                                                                   row[['svd_v_d_1', 'svd_v_d_2', 'svd_v_d_3', 'svd_v_d_4', 'svd_v_d
```

```
df_final_train.columns
```

```
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_index',
       'follows_back', 'same_comp', 'shortest_path', 'weight_in', 'weight_out',
       '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',
       'num_followers_d', 'preferential', 'svd_dot_u', 'svd_dot_v'],
      dtype='object')
```

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

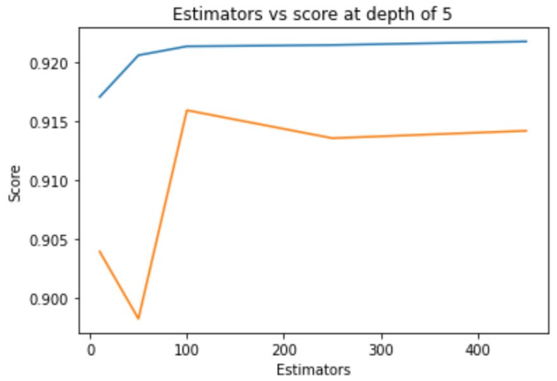
```
#As running these files takes a lot of time, so storing them in joblib file
import joblib
# create an iterator object with write permission - model.pkl
with open('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/df_final_train', 'wb') as files:
    joblib.dump(df_final_train, files)
with open('//content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/df_final_test', 'wb') as files:
    joblib.dump(df_final_test, files)
```

```
#As running these files takes a lot of time, so storing them in joblib file
import joblib
with open('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/df_final_train' , 'rb') as f:
    df_final_train = joblib.load(f)
with open('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/df_final_test' , 'rb') as f:
    df_final_test = joblib.load(f)
```

```
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)
```

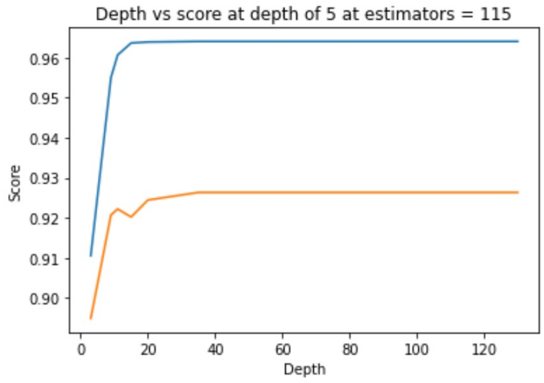
```
estimators = [10,50,100,250,450]
train_scores = []
test_scores = []
for i in estimators:
    clf = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0,
            min_samples_leaf=52, min_samples_split=120,
            min_weight_fraction_leaf=0.0, n_estimators=i, n_jobs=-1,random_state=25,verbose=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 0.9171013764062598 test Score 0.9039679681376184
Estimators =  50 Train Score 0.9206451345068084 test Score 0.8982537834691502
Estimators =  100 Train Score 0.9214043507746883 test Score 0.9159700835899691
Estimators =  250 Train Score 0.9215133283195456 test Score 0.9135973763874874
Estimators =  450 Train Score 0.9218142277612814 test Score 0.9142255892255893
Estimators =  450 Train Score 0.9218142277812814 test Score 0.9142259892259893
Text(0.5, 1.0, 'Estimators vs score at depth of 5')
```



```
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,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 =  3 Train Score 0.9105867070794544 test Score 0.8949359416473341
depth =  9 Train Score 0.9549922928512365 test Score 0.9207133058984912
depth =  11 Train Score 0.9606396939886264 test Score 0.9222154132240137
depth =  15 Train Score 0.9636245665970519 test Score 0.92020961879807 28
depth =  20 Train Score 0.9638757048395602 test Score 0.924450809831715
depth =  35 Train Score 0.9640206060114388 test Score 0.926338816204619
depth =  50 Train Score 0.9640206060114388 test Score 0.926338816204619
depth =  70 Train Score 0.9640206060114388 test Score 0.926338816204619
depth =  130 Train Score 0.9640206060114388 test Score 0.926338816204619
```



```
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=5,scoring='f1',random_state=25, return_train_score=True)
# https://stackoverflow.com/questions/57136676/sklearn-model-selection-gridsearchcv-is-throwing-keyerror-mean-train-score
rf_random.fit(df_final_train,y_train)
```

```
RandomizedSearchCV(cv=5,
                   estimator=RandomForestClassifier(n_jobs=-1, random_state=25),
                   n_iter=5,
                   param_distributions={'max_depth': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7fe6544590d0>,
                                        'min_samples_leaf': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7fe65442
                                        'min_samples_split': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7fe6547
                                        'n_estimators': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7fe654459650
                   random_state=25, return_train_score=True, scoring='f1')
```

```
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 [0.96230888 0.96147798 0.95999725 0.96116579 0.96358315]
     mean train scores [0.96310402 0.96222188 0.96046293 0.96187631 0.96451912]
```

```
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)
```

```
rf_random = 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)
```

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

```
! pwd
%cd /content/gdrive/My\ Drive/Assignments\ AAIC/Assignment\ 17\ FB\ Friend \Recommendation
```

```
     /content/gdrive/MyDrive/Assignments AAIC/Assignment 17 FB Friend Recommendation
     /content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation
```

```
#As running these files takes a lot of time, so storing them in joblib file
import joblib

with open('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Final/rf_random', 'wb') as files:
    pickle.dump(rf_random, files)
```

```
#As running these files takes a lot of time, so storing them in joblib file
import joblib
with open('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Final/rf_random', 'rb') as f:
    rf_random = joblib.load(f)
```

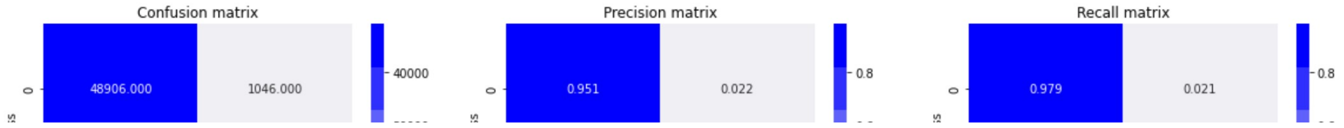
```
from sklearn.metrics import f1_score
print('Train f1 score',f1_score(y_train,y_train_pred))
print('Test f1 score',f1_score(y_test,y_test_pred))
```

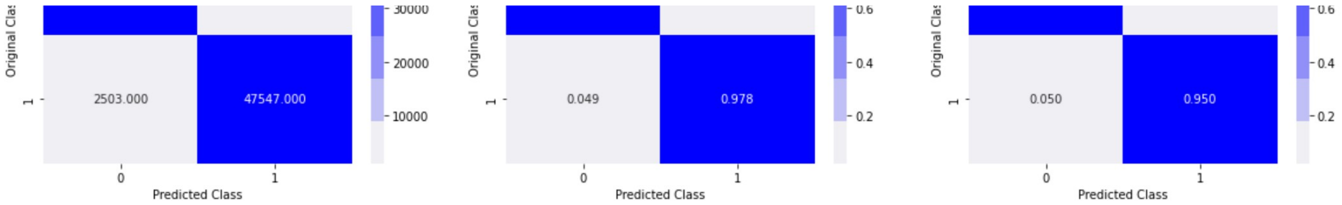
```
     Train f1 score 0.964021775493446
     Test f1 score 0.9236902050113895
```

```
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()
```

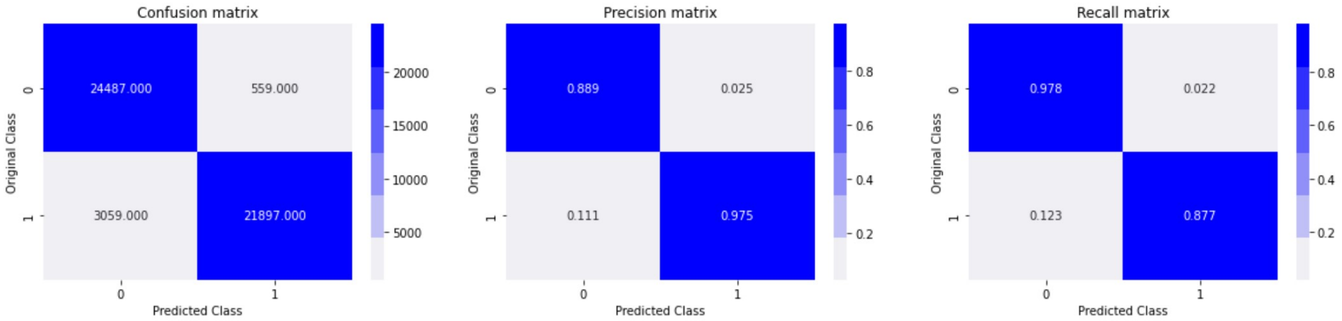
```
print('Train confusion_matrix')
plot_confusion_matrix(y_train,y_train_pred)
print('Test confusion_matrix')
plot_confusion_matrix(y_test,y_test_pred)
```

```
     Train confusion_matrix
```

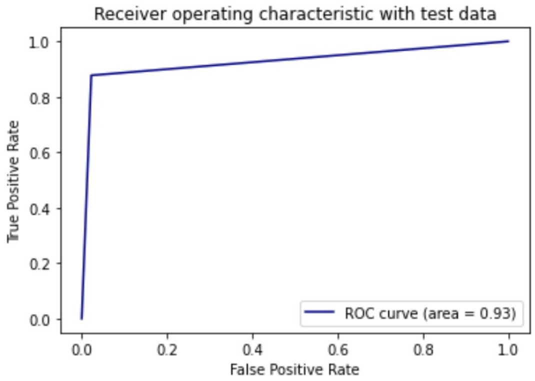




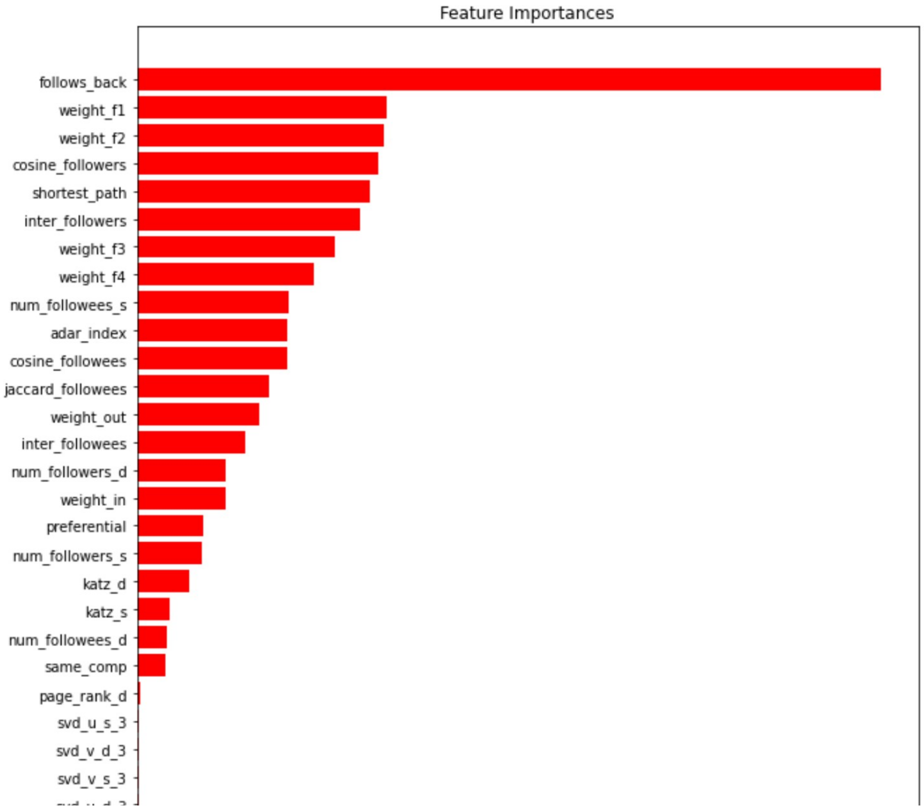
```
Test confusion_matrix
```

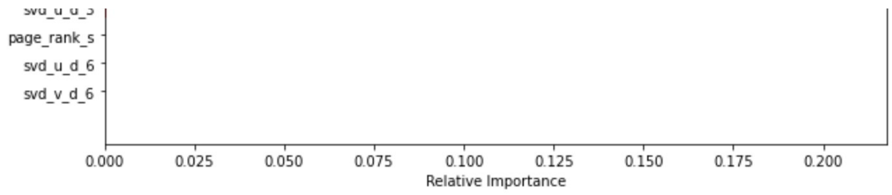


```
from sklearn.metrics import roc_curve, auc
fpr,tpr,ths = roc_curve(y_test,y_test_pred)
auc_sc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='navy',label='ROC curve (area = %0.2f)' % auc_sc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic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



```
features = df_final_train.columns
importances = rf_random.feature_importances_
indices = (np.argsort(importances))[-30:]
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 Model

```
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import f1_score,make_scorer


xgb_model = xgb.XGBClassifier() # Default params
params= {"max_depth": [1, 2, 4, 6], "n_estimators": [5, 20,50]}
gbdt = RandomizedSearchCV(xgb_model,params, n_iter=5, scoring='f1', cv=3, return_train_score=True, n_jobs=-1) # running 3 fold cros
gbdt.fit(df_final_train,y_train) #fitting
```

```
RandomizedSearchCV(cv=3, estimator=XGBClassifier(), n_iter=5, n_jobs=-1,
                   param_distributions={'max_depth': [1, 2, 4, 6],
                                        'n_estimators': [5, 20, 50]},
                   return_train_score=True, scoring='f1')
```

```
print('mean test scores',gbdt.cv_results_['mean_test_score'])
print('mean train scores',gbdt.cv_results_['mean_train_score'])
```

```
mean test scores [0.92064807 0.9212666  0.90939362 0.93393896 0.91944166]
mean train scores [0.92060962 0.92122149 0.91019882 0.93484829 0.91996065]
```

```
print(gbdt.best_estimator_)
```

```
XGBClassifier(max_depth=4, n_estimators=20)
```

```
gbdt = xgb.XGBClassifier(base_score=0.5, max_depth=10, min_child_weight=1, missing=None, n_estimators=109,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0, silent=True, subsample=1)

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

```
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 0.9913109863286155
Test f1 score 0.9279925319302413
```

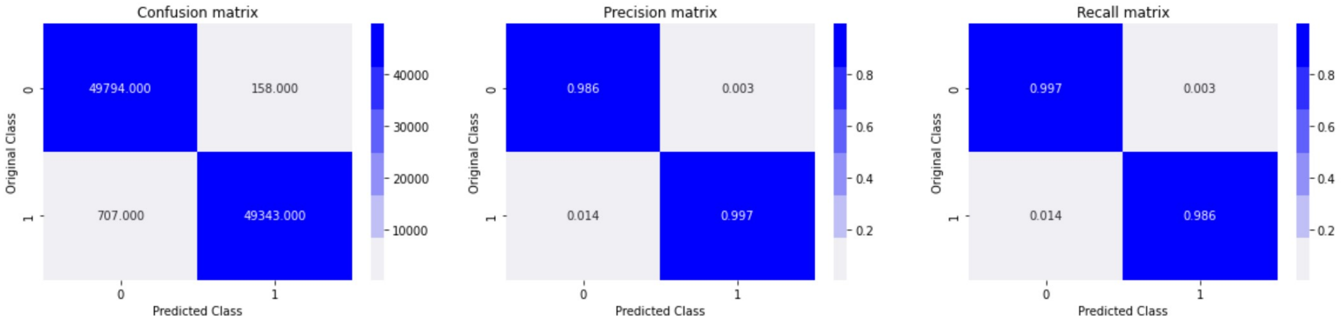
```
#As running these files takes a lot of time, so storing them in joblib file
import joblib

with open('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Final/gbdt', 'wb') as files:
    pickle.dump(gbdt, files)
```

```
#As running these files takes a lot of time, so storing them in joblib file
import joblib
with open('/content/gdrive/My Drive/Assignments AAIC/Assignment 17 FB Friend Recommendation/Final/gbdt', 'rb') as f:
    gbdt = joblib.load(f)
```

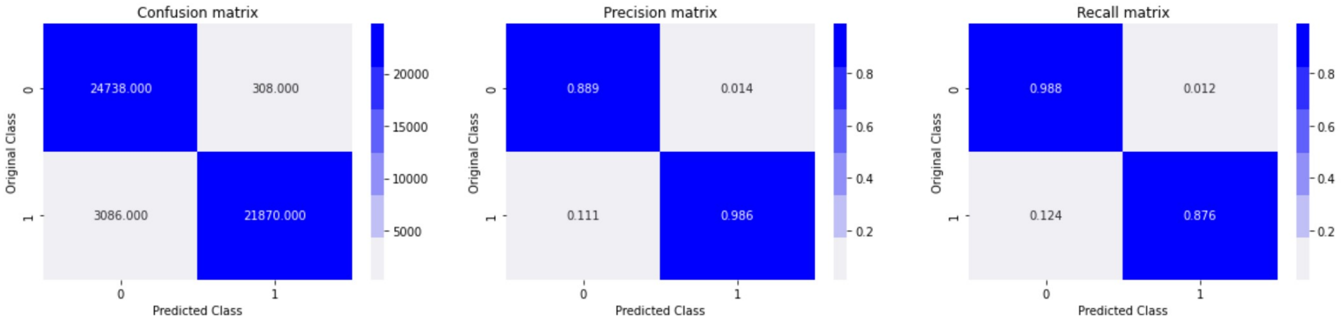
```
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)

    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()
```

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

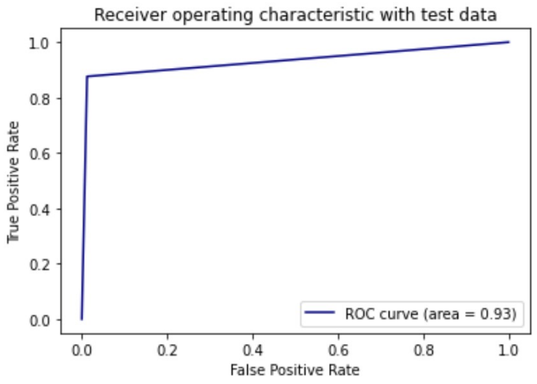
```
Train confusion_matrix
```



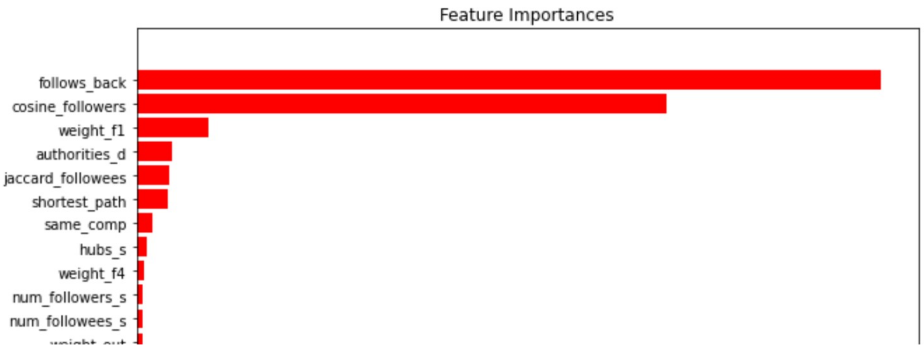
```
Test confusion_matrix
```

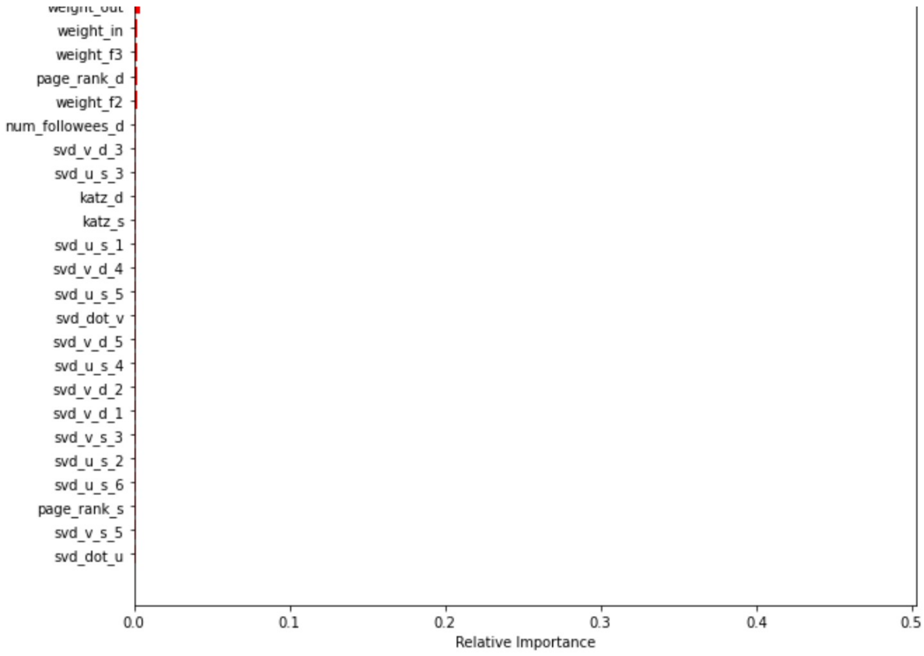


```
from sklearn.metrics import roc_curve, auc
fpr,tpr,ths = roc_curve(y_test,y_test_pred)
auc_sc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='navy',label='ROC curve (area = %0.2f)' % auc_sc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic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



```
features = df_final_train.columns
importances = gbdt.feature_importances_
indices = (np.argsort(importances))[-35:]
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()
```





3. Summary:-

```
from prettytable import PrettyTable
from prettytable import ALL as ALL
table=PrettyTable(hrules=ALL)
table.field_names = [ "Set.No","Model", "With or without Preferential and SVD_dot", "Hyper Parameters", "Test-AUC"]  # http://zetco
table.add_row([1,"Random Forest", "Without Preferential and SVD_dot", "max_depth =14 , n_estimator=121", 0.9241678239279553])
table.add_row([2, "Random Forest", "With Preferential and SVD_dot" ,"max_depth =14 , n_estimator=121", 0.9236902050113895])
table.add_row([3, "XGBoost", "With Preferential and SVD_dot" ,"max_depth =6 , n_estimator=5", 0.9279925319302413])

print(table)
```

```
+--------+---------------+------------------------------------------+----------------------------------+--------------------+
| Set.No |     Model     | With or without Preferential and SVD_dot |         Hyper Parameters         |      Test-AUC      |
+--------+---------------+------------------------------------------+----------------------------------+--------------------+
|   1    | Random Forest |     Without Preferential and SVD_dot     | max_depth =14 , n_estimator=121 | 0.9241678239279553 |
+--------+---------------+------------------------------------------+----------------------------------+--------------------+
|   2    | Random Forest |      With Preferential and SVD_dot       | max_depth =14 , n_estimator=121 | 0.9236902050113895 |
+--------+---------------+------------------------------------------+----------------------------------+--------------------+
|   3    |    XGBoost    |      With Preferential and SVD_dot       |   max_depth =6 , n_estimator=5   | 0.9279925319302413 |
+--------+---------------+------------------------------------------+----------------------------------+--------------------+
```

4. Conclusion:-

Adding Preferential Attachments, Svd_dot and applying XGBoost Model won't result in any substantial difference in the Performance of the Model