NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

Cachar, Assam

B.Tech. VIth Sem

Subject Code: CS-321

Subject Name: Social Network Analysis Lab

Submitted By:

Name	: Subhojit Ghimire
Sch. Id.	: 1912160
Branch	: CSE – B

AIM: TO ANALYSE THE PERFORMANCE OF LINK PREDICTION METHODS MENTIONED IN THE PREVIOUS EXPERIMENT (EXP-8) IN TERMS OF PRECISION AND AREA UNDER THE RECEIVER OPERATING CHARACTERISTICS CURVE (AUC ROC).

THEORY:

1. **AUC-ROC Curve:**

$$\frac{n' + 0.5n''}{n}$$

where,

n is the number of times that we randomly pick a pair of links from missing links set and unconnected link set;
n′ is the number of times that the missing link got a higher score than unconnected link
n″ is the number of time when missing link equals unconnected link

2. **Precision:**

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$True\ Positive\ Rate = \frac{True\ Positives}{True\ Positive + False\ Negatives}$$

$$Specificity = \frac{True\ Negative}{True\ Negatives + False\ Postitives}$$

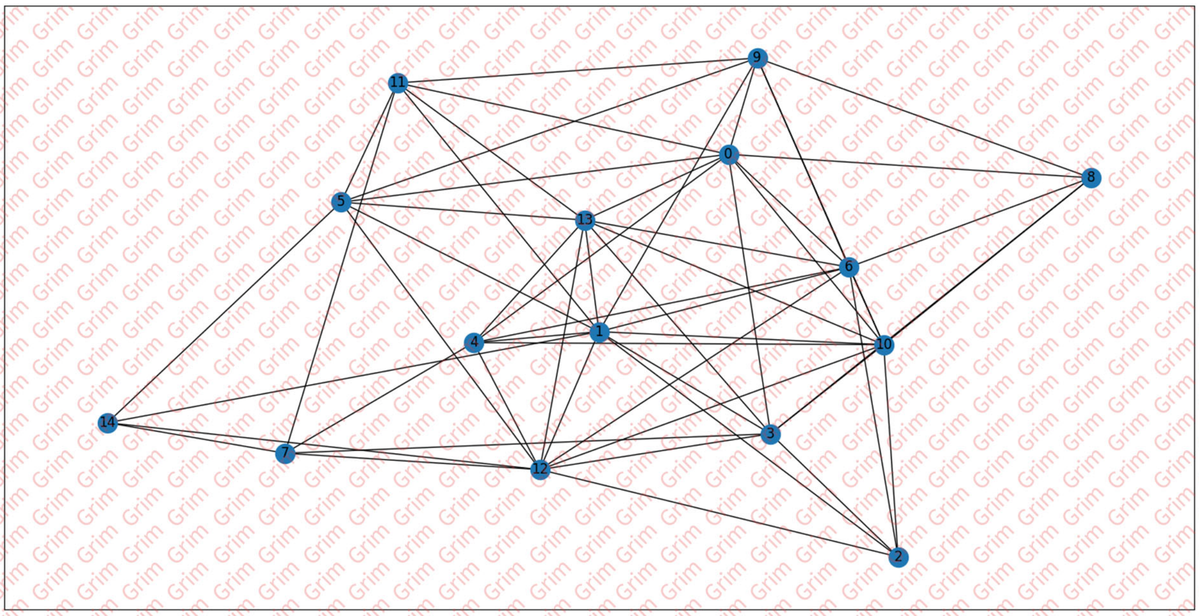
$$False\ Positive\ Rate\ (FPR) = \frac{False\ Positive}{True\ Negative + False\ Positive}$$

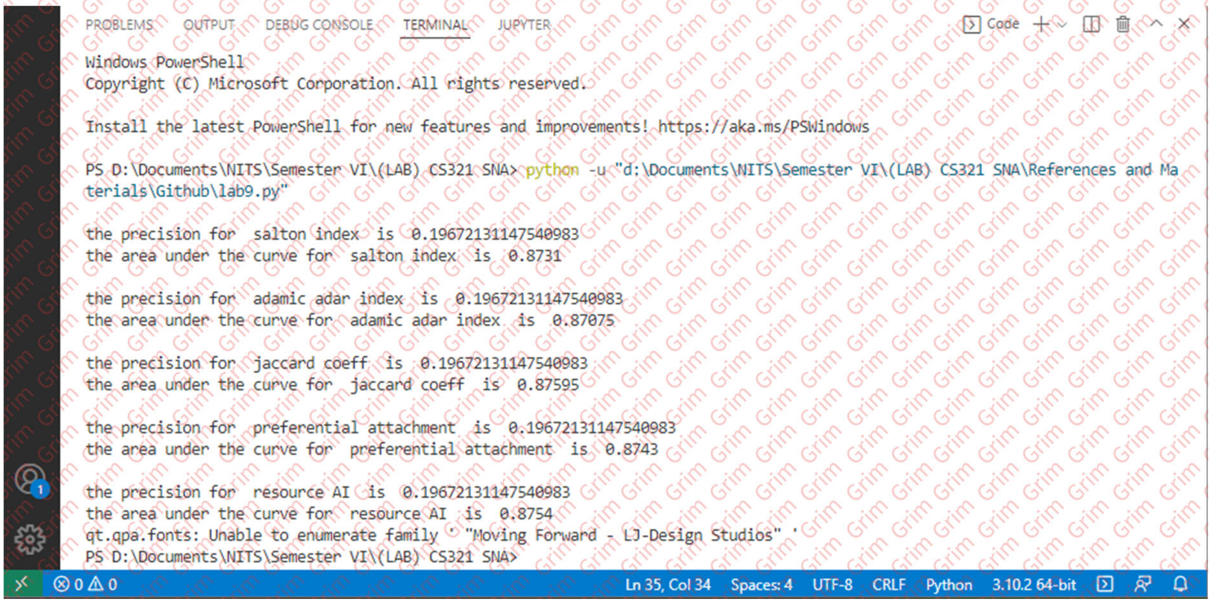
CODE:

```
from operator import index
import networkx as nx
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import spatial

def saltonList (G):
    edges = nx.edges(G)
    n = G.number_of_nodes()
    list = [[]]
    for i in range (n-1):
        list += [[]]
    for i in range (n):
        for j in edges:
            if (j[0] == i):
                list [i].append (j[1])
            if (j[1] == i):
                list [i].append (j[0])
    _list = [[]]
    var = 0
    for i in range (n):
        for j in range (i+1, n):
            if j not in list[i]:
                _list[var].append (i)
                _list[var].append (j)
                var += 1
                _list += [[]]
    _list.remove ([])

    adjMatrix = [[]]
    var = 0
    for i in range (n):
        for j in range (n):
            if j not in list [i]:
                adjMatrix[var].append (0)
            else:
                adjMatrix[var].append (1)
        var += 1
        adjMatrix += [[]]
    adjMatrix.remove ([])

    var = 0
    for i in _list:
        cosineSimilarity = 1 - spatial.distance.cosine (adjMatrix [i[0]],
adjMatrix [i[1]])
        _list[var].append(cosineSimilarity)
```

```
            var += 1
        return _list

    def precisionAUC (df, index, signal):
        t1 = int (len (df) * 0.8)
        t2 = len (df) - t1
        train = df [:][:t1]
        test = df [:][t1:t2+t1]

        G = nx.Graph ()
        G = nx.from_pandas_edgelist (train, 'from', 'to')

        match signal:
            case 0: preds = saltonList (G)
            case 1: preds = nx.adamic_adar_index (G)
            case 2: preds = nx.jaccard_coefficient (G)
            case 3: preds = nx.preferential_attachment (G)
            case 4: preds = nx.resource_allocation_index (G)

        preds = sorted (preds, reverse = True, key = lambda x: x [2])
        preds = preds [:10000]
        predictedLinks = []
        for i, j, k in preds:
            predictedLinks.append ([i, j])

        l1, l2 = [], []
        x1, x2 = len (train), len (test)
        for i in range (0, x1 - 1):
            l1.append ([train ['from'][i], train ['to'][i]])

        for i in range (x1, x1 + x2):
            l2.append ([test ['from'][i], test ['to'][i]])

        trainSet = set (tuple (i) for i in l1)
        testSet = set (tuple (i) for i in l2)
        predictedLinksSet = set (tuple (i) for i in predictedLinks)

        TP = predictedLinksSet.intersection (testSet)
        FP = predictedLinksSet.difference (TP)
        FN = testSet.difference (TP)

        tpLen = len (TP)
        fpLen = len (FP)
        fnLen = len (FN)

        precision = tpLen / (tpLen + fpLen)
        print ("\nthe precision for ", index," is ", precision)
        nn, n1, n2 = 0., 0., 0.
        for i in range (0, 10000):
            nn += 1
            tt, ff = np.random.randint (0, tpLen), np.random.randint (0, fpLen)
            if preds [tt][2] > preds [ff][2]:
                n1 += 1
            if preds [tt][2] == preds[ff][2]:
                n2 += 1
        AUC = (n1 + 0.5 * n2) / nn
        print ("the area under the curve for ", index," is ", AUC)

    G = nx.erdos_renyi_graph (15, p = 0.5)
    edges = nx.edges (G)
    f = open ('tempEdgeList.txt', 'w')
    for i in edges:
        f.write (str (i [0]))
        f.write (' ')
        f.write (str (i [1]))
        f.write ('\n')
    f.close ()
    df = pd.read_csv("tempEdgeList.txt",delimiter = ' ',header = None)
    df.columns = ['from','to']

    precisionAUC (df, "salton index", 0)
    precisionAUC (df, "adamic adar index", 1)
    precisionAUC (df, "jaccard coeff", 2)
    precisionAUC (df, "preferential attachment", 3)
    precisionAUC (df, "resource AI", 4)

    plt.figure ()
    nx.draw_networkx (G, with_labels = True)
    plt.show ()
```

OUTPUT AND OBSERVATIONS:





```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Documents\NITS\Semester VI\(LAB) CS321 SNA> python -u "d:\Documents\NITS\Semester VI\(LAB) CS321 SNA\References and Ma
terials\Github\lab9.py"

the precision for  salton index  is  0.19672131147540983
the area under the curve for  salton index  is  0.8731

the precision for  adamic adar index  is  0.19672131147540983
the area under the curve for  adamic adar index  is  0.87075

the precision for  jaccard coeff  is  0.19672131147540983
the area under the curve for  jaccard coeff  is  0.87595

the precision for  preferential attachment  is  0.19672131147540983
the area under the curve for  preferential attachment  is  0.8743

the precision for  resource AI  is  0.19672131147540983
the area under the curve for  resource AI  is  0.8754
qt.qpa.fonts: Unable to enumerate family ' "Moving Forward - LJ-Design Studios" '
PS D:\Documents\NITS\Semester VI\(LAB) CS321 SNA>
```