**CN**

**import** numpy **as** np  
**import** random  
**def** pja(noexist,result,c): *# 计算AUC的函数#* point = 0 *#point为得到的分数* cp = [] *#存储每次得到的分数* **for** x **in** range(0,len(c)-1):  
 **for** y **in** range(0,len(result)-1):  
 **if** int(c[x][0])==result[y][0]:  
 **if** int(c[x][1])==result[y][1]:  
 cp.append(result[y][2])  
 **if** int(c[x][0])==result[y][1]:  
 **if** int(c[x][1])==result[y][0]:  
 cp.append(result[y][2]) *#从result中得到测试集的边的CN值* **for** q **in** range(0,len(cp)-1):  
 x1=random.randint(0,len(noexist)-1) *#生成随机数，随机选择一条不存在的边* np=noexist[x1][2]  
 **if** cp[q]>np:  
 point=point+1  
 **if** cp[q]==np:  
 point=point+0.5  
 auc=point/len(cp) *#得到分数计算CN* **return** auc  
  
**def** pre(noexist,result,g):  
 p=0 *#p保存预测准确的数目* **for** n **in** range(0,len(result)-1):  
 **for** m **in** range(0,len(g)-1):  
 **if** result[n][0]==int(g[m][0]):  
 **if** result[n][1]==int(g[m][1]):  
 g[m].append(result[n][2])  
 **if** result[n][1]==int(g[m][0]):  
 **if** result[n][0]==int(g[m][1]):  
 g[m].append(result[n][2]) *#从result中得到测试集的边的CN值* noexist.append(g) *#将测试集的边加到noexist里* **for** x **in** range(0,len(noexist)-2):  
 **for** y **in** range(x+1,len(noexist)-1):  
 **if** noexist[x][2]<noexist[y][2]:  
 jh=noexist[x]  
 noexist[x]=noexist[y]  
 noexist[y]=jh *#给noexist里的边按CN值排序* l=len(g) *#选取前L个，暂时以测试集长度为L* **for** w **in** range(0,l-1):  
 **for** s **in** range (0,l-1):  
 **if** int(noexist[w][0])==int(g[s][0]):  
 **if** int(noexist[w][1] )== int(g[s][1]):  
 p=p+1  
 **if** int(noexist[w][0])==int(g[s][1]):  
 **if** int(noexist[w][1]) == int(g[s][0]):  
 p=p+1 *#如果前L个在测试集里，p+1* precision=p/l *#计算precision* **return** precision  
  
  
  
result=[] *#存储CN结果，形式为1,2,1和2的CN，三个数组成一个列表，每个列表存储在大列表result中*a=[] *#读取文件存放的列表#*c=[] *#读取测试集文件到C*x={} *#存储某个节点的邻居*cn=0  
i=0 *#列表x的循环变量***with** open(**'测试集.txt'**, **'r'**) **as** fp:  
 d = fp.readlines()  
 **for** line **in** d:  
 t3 = line.strip(**'\n'**)  
 t4 = t3.split(**'\t'**)  
 c.append((t4))  
**with** open(**'训练集.txt'**,**'r'**) **as** f:  
 b=f.readlines()  
 **for** line **in** b:  
 t1=line.strip(**'\n'**)  
 t2=t1.split(**'\t'**)  
 a.append((t2))  
 v=input(**"输入节点总数:"**)  
 s=int(v) *#读取训练集文件到a中，s为数字最大的节点* **for** j **in** range(1,s-1):  
 **for** k **in** range(j+1,s): *#依次求CN12,CN13...CN1s；CN23,CN24......CN(s-1)s* **for** m **in** range(0, len(a)-1):  
 **if** str(j)== a[m][0]:  
 **if not** a[m][1] **in** x :  
 x[i]=a[m][1]  
 i=i+1  
 **elif** str(j)==a[m][1]:  
 **if not** a[m][0] **in** x:  
 x[i] = a[m][0]  
 i =i+ 1 *#记录j的邻居* **for** n **in** range(0,len(a)-1):  
 **for** i **in** range(0,len(x)-1):  
 y=**''  
 if** a[n][0] == x[i] :  
 y=a[n][1]  
 **elif** a[n][1] == x[i]:  
 y=a[n][0]  
 **if** y==str(k):  
 cn=cn+1 *#判断j的邻居里有没有k，得到CN* print(**"cn of"** ,j, **"and"**, k, **"is"**, cn)  
 ls=[j,k,cn]  
 result.append(ls) *#将每组CN存储在result里* cn=0  
 i=0  
 x.clear() *#循环变量清零* noexist = [] *#存储不存在的边* r=0  
 **for** i **in** range(1, s - 1):  
 **for** j **in** range(i + 1, s):  
 ju=0 *#利用ju判断某条连边是否存在于训练集中* **for** m **in** range(0, len(a) - 1):  
 **if** str(i) **in** a[m]:  
 **if** str(j) **in** a[m]:  
 ju=1 *#如果连边存在于训练集，则ju=1,否则为0* **break  
 for** u **in** range(0,len(c)-1):  
 **if** str(i) **in** c[u]:  
 **if** str(j) **in** c[u]:  
 ju = 1 *# 如果连边存在于测试集，则ju=1,否则为0* **break  
 if** ju==0:  
 lsh = [i, j,result[r][2]]  
 noexist.append(lsh) *#存储ju==0的边（不存在的边）* r=r+1  
auc=pja(noexist,result,c) *#计算AUC*print(**'auc='**,auc)  
*preci=pre(noexist,result,c) #计算PRECISION  
print('precision=',preci)*

*JC*

**import** numpy **as** np  
**import** random  
**def** pja(noexist,result,c): *# 计算AUC的函数#* point = 0 *#point为得到的分数* cp = [] *#存储每次得到的分数* **for** x **in** range(0,len(c)-1):  
 **for** y **in** range(0,len(result)-1):  
 **if** int(c[x][0])==result[y][0]:  
 **if** int(c[x][1])==result[y][1]:  
 cp.append(result[y][2])  
 **if** int(c[x][0])==result[y][1]:  
 **if** int(c[x][1])==result[y][0]:  
 cp.append(result[y][2]) *#从result中得到测试集的边的CN值* **for** q **in** range(0,len(cp)-1):  
 x1=random.randint(0,len(noexist)-1) *#生成随机数，随机选择一条不存在的边* np=noexist[x1][2]  
 **if** cp[q]>np:  
 point=point+1  
 **if** cp[q]==np:  
 point=point+0.5  
 auc=point/len(cp) *#得到分数计算CN* **return** auc  
  
**def** pre(noexist,result,g):  
 p=0 *#p保存预测准确的数目* **for** n **in** range(0,len(result)-1):  
 **for** m **in** range(0,len(g)-1):  
 **if** result[n][0]==int(g[m][0]):  
 **if** result[n][1]==int(g[m][1]):  
 g[m].append(result[n][2])  
 **if** result[n][1]==int(g[m][0]):  
 **if** result[n][0]==int(g[m][1]):  
 g[m].append(result[n][2]) *#从result中得到测试集的边的CN值* noexist.append(g) *#将测试集的边加到noexist里* **for** x **in** range(0,len(noexist)-2):  
 **for** y **in** range(x+1,len(noexist)-1):  
 **if** noexist[x][2]<noexist[y][2]:  
 jh=noexist[x]  
 noexist[x]=noexist[y]  
 noexist[y]=jh *#给noexist里的边按CN值排序* l=len(g) *#选取前L个，暂时以测试集长度为L* **for** w **in** range(0,l-1):  
 **for** s **in** range (0,l-1):  
 **if** int(noexist[w][0])==int(g[s][0]):  
 **if** int(noexist[w][1] )== int(g[s][1]):  
 p=p+1  
 **if** int(noexist[w][0])==int(g[s][1]):  
 **if** int(noexist[w][1]) == int(g[s][0]):  
 p=p+1 *#如果前L个在测试集里，p+1* precision=p/l *#计算precision* **return** precision  
  
  
  
result=[] *#存储CN结果，形式为1,2,1和2的CN，三个数组成一个列表，每个列表存储在大列表result中*a=[] *#读取文件存放的列表#*c=[] *#读取测试集文件到C*x={}  
z={} *#存储某个节点的邻居*cn=0  
i=0  
i2=0 *#列表x的循环变量***with** open(**'测试集.txt'**, **'r'**) **as** fp:  
 d = fp.readlines()  
 **for** line **in** d:  
 t3 = line.strip(**'\n'**)  
 t4 = t3.split(**'\t'**)  
 c.append((t4))  
**with** open(**'训练集.txt'**,**'r'**) **as** f:  
 b=f.readlines()  
 **for** line **in** b:  
 t1=line.strip(**'\n'**)  
 t2=t1.split(**'\t'**)  
 a.append((t2))  
 v=input(**"输入节点总数:"**)  
 s=int(v) *#读取训练集文件到a中，s为数字最大的节点* **for** j **in** range(1,s-1):  
 **for** k **in** range(j+1,s): *#依次求CN12,CN13...CN1s；CN23,CN24......CN(s-1)s* **for** m **in** range(0, len(a)-1):  
 **if** str(j)== a[m][0]:  
 **if not** a[m][1] **in** x :  
 x[i]=a[m][1]  
 i=i+1  
 **elif** str(j)==a[m][1]:  
 **if not** a[m][0] **in** x:  
 x[i] = a[m][0]  
 i =i+ 1 *#记录j的邻居* **for** o **in** range(0, len(a)-1):  
 **if** str(k)== a[o][0]:  
 **if not** a[o][1] **in** z :  
 z[i2]=a[o][1]  
 i2=i2+1  
 **elif** str(k)==a[o][1]:  
 **if not** a[o][0] **in** z:  
 z[i2] = a[o][0]  
 i2 =i2+ 1  
 **for** n **in** range(0,len(a)-1):  
 **for** i **in** range(0,len(x)-1):  
 y=**''  
 if** a[n][0] == x[i] :  
 y=a[n][1]  
 **elif** a[n][1] == x[i]:  
 y=a[n][0]  
 **if** y==str(k):  
 cn=cn+1 *#判断j的邻居里有没有k，得到CN* **if**(len(x)+len(z)-cn)==0:  
 jc=0  
 **else**:  
 jc=cn/(len(x)+len(z)-cn)  
 print(**"JC of"**, j, **"and"**, k, **"is"**, jc)  
 ls=[j,k,jc]  
 result.append(ls) *#将每组JC存储在result里* cn=0  
 jc=0  
 i=0  
 i2=0  
 x.clear()  
 z.clear() *#循环变量清零* noexist = [] *#存储不存在的边* r=0  
 **for** i **in** range(1, s - 1):  
 **for** j **in** range(i + 1, s):  
 ju=0 *#利用ju判断某条连边是否存在于训练集中* **for** m **in** range(0, len(a) - 1):  
 **if** str(i) **in** a[m]:  
 **if** str(j) **in** a[m]:  
 ju=1 *#如果连边存在于训练集，则ju=1,否则为0* **break  
 for** u **in** range(0,len(c)-1):  
 **if** str(i) **in** c[u]:  
 **if** str(j) **in** c[u]:  
 ju = 1 *# 如果连边存在于测试集，则ju=1,否则为0* **break  
 if** ju==0:  
 lsh = [i, j,result[r][2]]  
 noexist.append(lsh) *#存储ju==0的边（不存在的边）* r=r+1  
auc=pja(noexist,result,c) *#计算AUC*print(**'auc='**,auc)  
preci=pre(noexist,result,c) *#计算PRECISION*print(**'precision='**,preci)