姓名	学号	Github 账号	分工	成绩
周芯宇	201592451	Nana1209	组长	
崔芳宇	201592091	roverCui	组员	
陈冠伊	201592479	crystalcgy	组员	
王柄楠	201592084	Dainy33	组员	

开源软件基础课程报告

报 告 题 目　基于群聚系数的链路消失预测

项 目 网 址　https://github.com/Nana1209/OpenSourceSoftware.git

完 成 日 期　2018/1/11

大连理工大学软件学院

目　　录

1 需求分析

1.1 问题背景

1.1.1 复杂网络链路预测

网络中的链路预测是指如何通过已知的网络节点以及网络结构等信息预测网络中尚未产生连边的两个节点之间产生链接的可能性[1]。这种预测既包含了对未知链接的预测也包含了对未来链接的预测。该问题的研究在理论和应用两个方面都具有重要的意义和价值。最近几年，基于网络结构的链路预测方法受到越来越多的关注。相比节点的属性信息而言，网络的结构更容易获得，也更加可靠。同时，该类方法对于结构相似的网络具有普适性，从而避免了对不同网络需要机器学习获得一些特定的参数组合。

1.1.2 衡量链路预测算法精确度的指标

衡量链路预测算法精确度的指标有 AUC、Precision 和 Ranking Score 共 3 种。三者侧重点不同。在这里我们主要关注偏重于从整体上衡量算法精确度的 AUC。AUC 可以理解为在测试集中的边的分数值高的概率，也就是说，每次随机从测试集汇中选取一条边与随机集中选取一条边与随机选择的不存在的边进行比较，如果测试集中的边的分数值大于不存在的边的分数值，就加 1 分；如果两个分数值相等，就加 0.5 分。

1.2 问题定义

聚集系数是表示一个图形中节点聚集程度的系数，证据显示，在现实的网络中，尤其是在特定的网络中，由于相对高密度连接点的关系，节点总是趋向于建立一组严密的组织关系。在现实世界的网络，这种可能性往往比两个节点之间随机设立了一个连接的平均概率更大；同理，两个节点间连接的消失也可用聚类系数来衡量。这种相互关系可以利用聚类系数进行量化表示。

$$cc_v = \frac{2n}{k(k-1)}$$

其中：k 表示节点 v 的所有相邻的节点的个数，即节点 v 的邻居。n 表示节点 v 的所有相邻节点之间相互连接的边的个数。

群聚系数是网络中所有节点的聚类系数的平均值。我们采用原网络群聚系数作为基准，删除某一对节点间连接后的网络的群聚系数与基准间的差值作为衡量指标。差值越大，表示该连接对网络的影响力越大；反之，越小，也就意味该连接最容易消失。

删除一条边后对所关联节点的度数即节点邻居数一定会有影响，但对节点所有相邻节点之间的相互连接的边的个数不一定会有影响，因此，群聚系数的差值可能为正也可能为负。

将每条边的衡量指标进行从小到大排序，排名越靠前链路消失的概率越大。

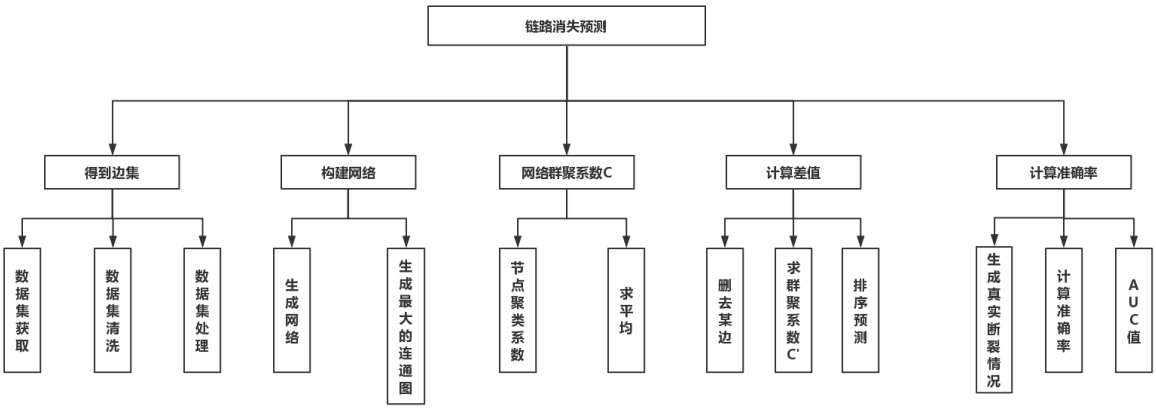
2 总体设计

2.1 任务划分

任务名称	资源
项目计划	周芯宇、崔芳宇、王柄楠、陈冠伊
需求分析	周芯宇、崔芳宇、王柄楠、陈冠伊
总体设计	周芯宇、王柄楠
测试计划编制	陈冠伊、崔芳宇
详细设计	周芯宇
获取边集	崔芳宇、王柄楠、陈冠伊
数据集获取	崔芳宇
数据集清洗	王柄楠
数据集处理	陈冠伊
构建网络	周芯宇、王柄楠、陈冠伊
生成网络	周芯宇
生成最大的连通图	王柄楠、陈冠伊
求网络群聚系数 C	周芯宇、崔芳宇、王柄楠
节点聚类系数	周芯宇
求平均	崔芳宇、王柄楠
计算差值	周芯宇、崔芳宇、陈冠伊
删除某边	崔芳宇、陈冠伊
求群聚系数 C	周芯宇
排序预测	周芯宇
计算准确率	周芯宇、崔芳宇、王柄楠、陈冠伊
生成真实断裂情况	周芯宇、崔芳宇
计算准确率	周芯宇、王柄楠
计算 AUC 值	周芯宇、陈冠伊

代码复核	王柄楠、陈冠伊
测试	周芯宇、崔芳宇、王柄楠、陈冠伊
编写项目报告	周芯宇、崔芳宇、王柄楠、陈冠伊

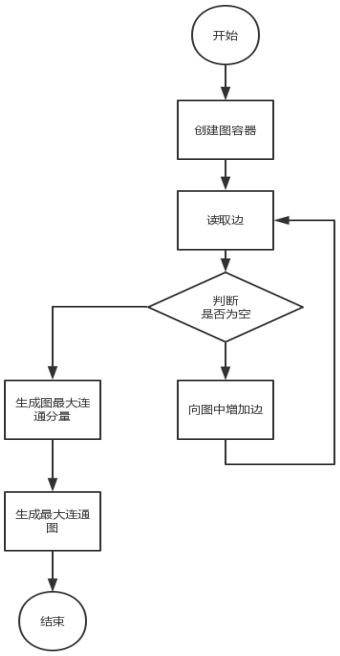
2.2 软件结构设计



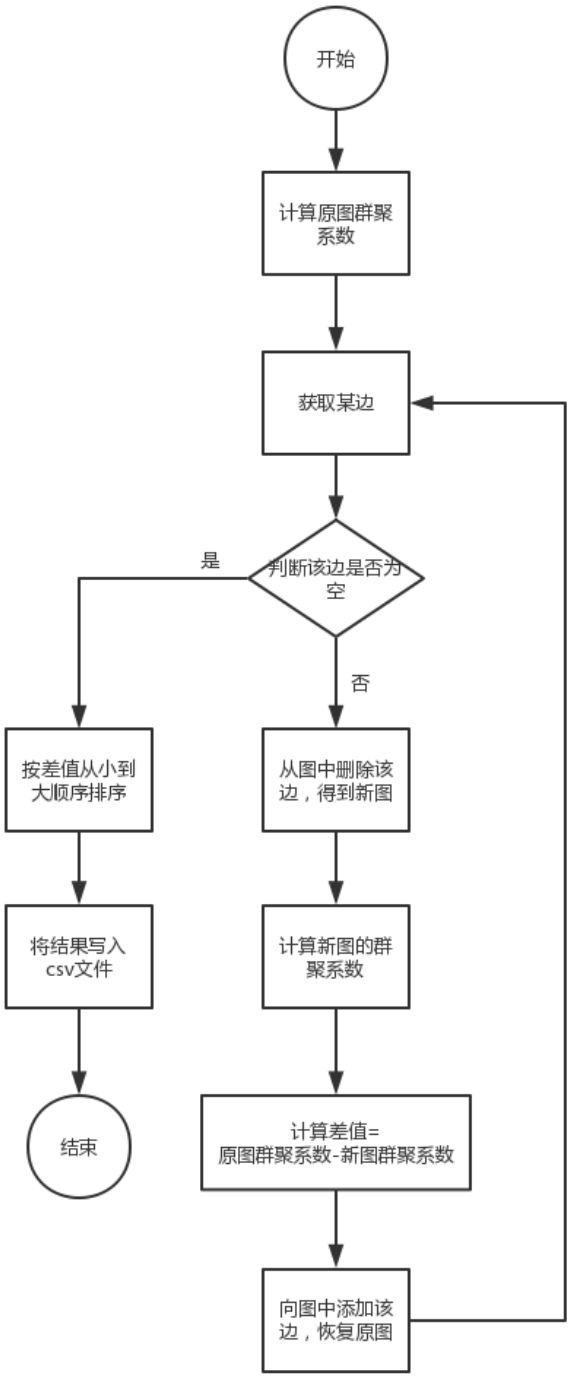
3 详细设计

本项目详细设计中各部分的程序流程图如下所示。

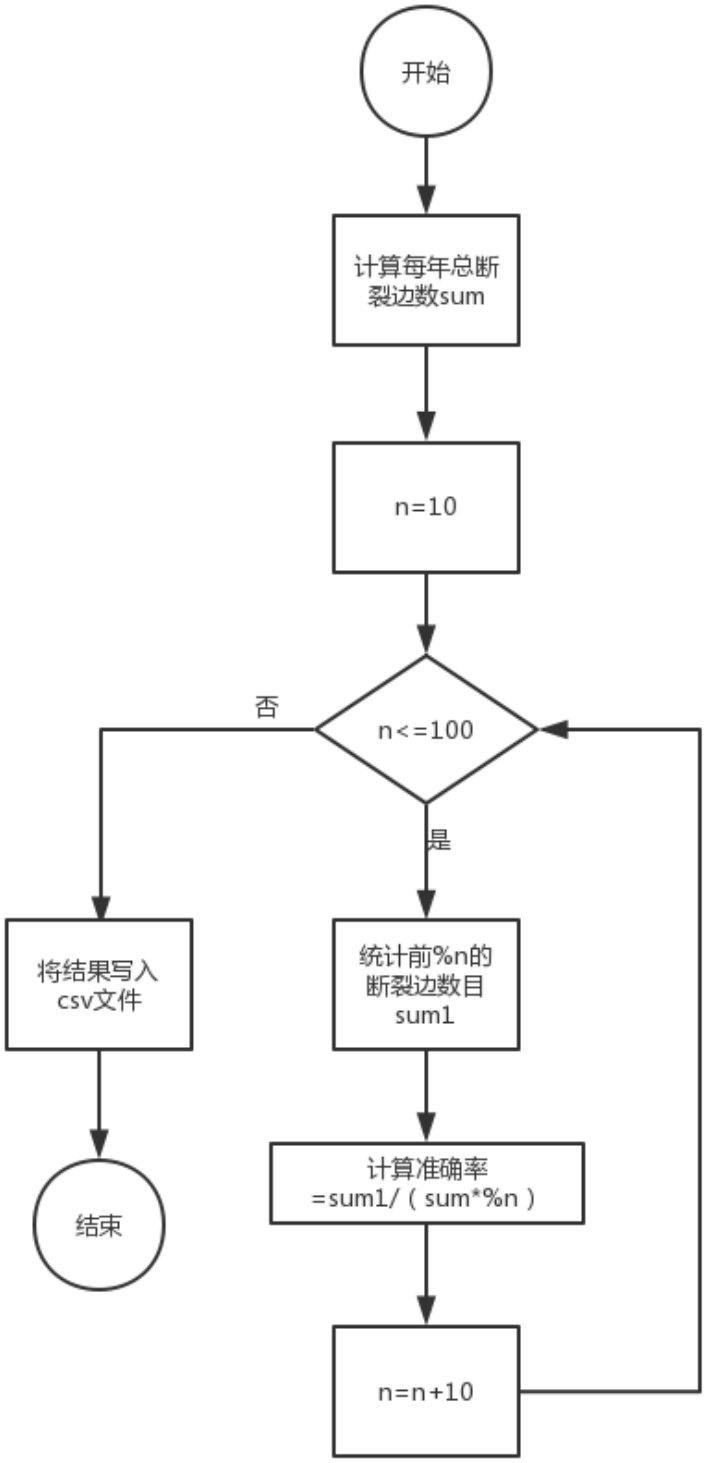
3.1 构建网络



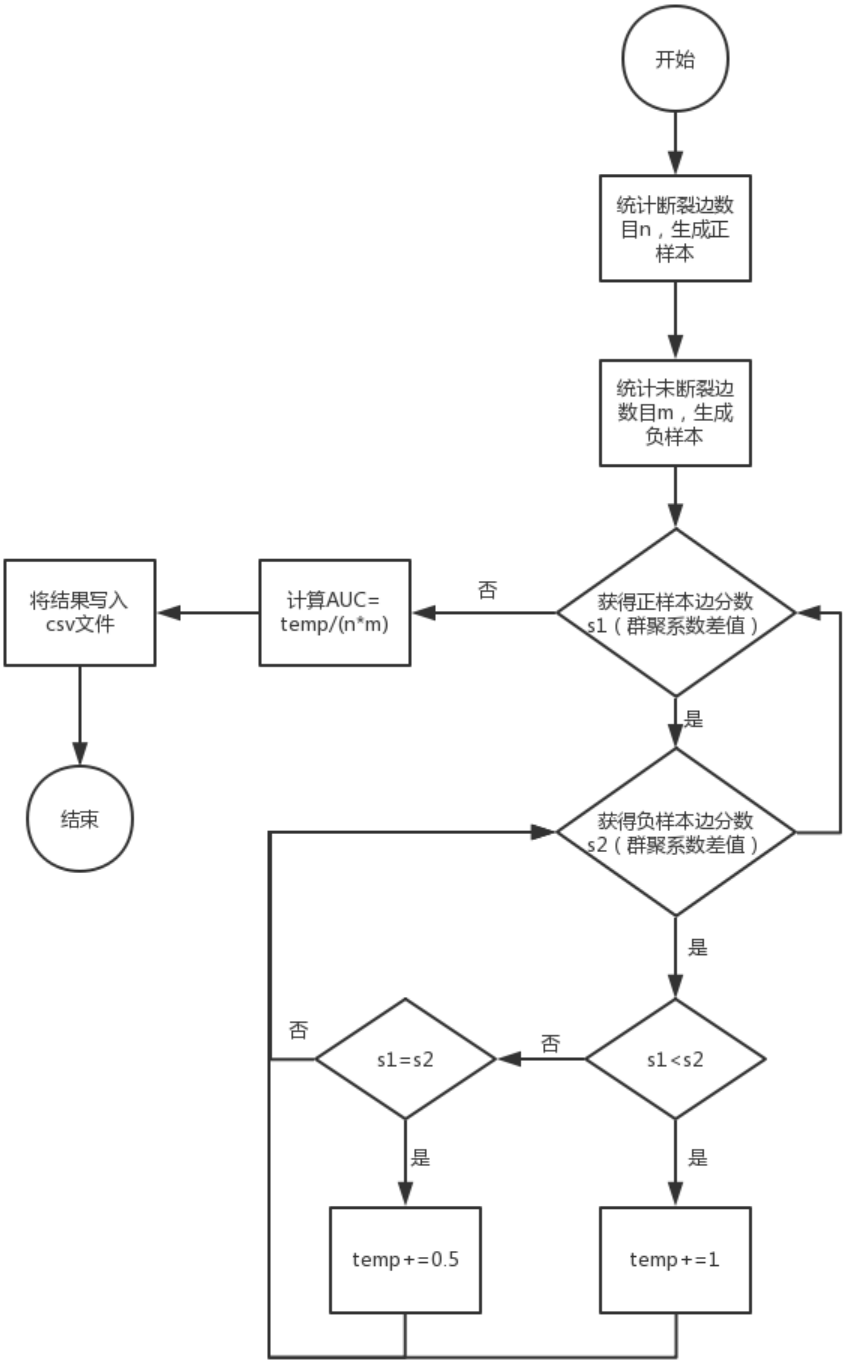
3.2 计算差值


开始
计算原图群聚系数
获取某边
判断该边是否为空
是
否
按差值从小到大顺序排序
从图中删除该边，得到新图
将结果写入csv文件
计算新图的群聚系数
结束
计算差值=
原图群聚系数-新图群聚系数
向图中添加该边，恢复原图

3.3 计算准确率


开始
计算每年总断裂边数sum
n=10
n<=100
否
是
将结果写入csv文件
统计前%n的断裂边数目sum1
结束
计算准确率=sum1/（sum*%n）
n=n+10

3.4 计算 AUC 值


开始
统计断裂边数目n，生成正样本
统计未断裂边数目m，生成负样本
获得正样本边分数s1（群聚系数差值）
否
计算AUC=temp/(n*m)
将结果写入csv文件
结束
是
获得负样本边分数s2（群聚系数差值）
是
s1<s2
否
s1=s2
否
是
temp+=1
是
temp+=0.5

4 实现

4.1 创建网络图

```
import csv  //引入 csv 库，对 csv 文件进行操作
import networkx as nx  //引入 networkx 库

Gw=nx.Graph()  //创建一个图容器（加权图）

csvReader=csv.reader(file('E:\dataProgram\datasets\Collaborations_PRA.csv',
'rb'))  //创建 csv 文件读取模块

#creat Graph
for row in csvReader:
    Gw.add_edge(row[0],row[1],weight=row[2])  //向图中添加节点，构建网络

#creat the largest connected component
largest_ccw = max(nx.connected_components(Gw), key=len)  //提取最大连通分量
subGw=Gw.subgraph(largest_ccw)  //生成最大连通图（下文用“子图”代替）

edge_list=subG.edges()
node_list=subG.nodes()

csvfile1=file('E:\dataProgram\datasets\PRB_AverageClusteringWeight.csv','wb
')  //生成写文件的 csv 容器
writer1=csv.writer(csvfile1)  //生成写文件句柄
```

4.2 求每条连接分数（群聚系数差值）

```
average=nx.average_clustering(subG,weight="weight")  //求出子图群聚系数
for i in range(0,len(edge_list)):
    #print edge_list[i]
    temp=edge_list[i]  //要删除的某边
    subG.remove_edge(*edge_list[i])  //从子图中删去该边
```

```
average1=average_clustering(subG,weight="weight");   //求删去该边的群聚系数
   writer1.writerow([temp[0],temp[1],nx.average1,average-average1])  //将数据和群聚系数差值写入文件
   subG.add_edge(*temp)  //向图中添加该边，恢复子图
```

4.3 获取真实连接断裂情况

```
r1=open('E:\dataProgram\datasets\AverageClustering1.csv','r')
csvReader1=csv.reader(r1)

r2=open('E:\dataProgram\datasets\Collaborations_PRA.csv','r')
csvReader2=csv.reader(r2)

addA={}
addB={}
w=open('E:\dataProgram\datasets\suBYearAverageClustering1.csv','wb')
writer1=csv.writer(w)  //获取真实连接情况
writer1.writerow(['source','target','averageClusteringCoefficient','sub','hasIn1Y','hasIn2Y','hasIn3Y','hasIn4Y','hasIn5Y','hasIn6Y','hasIn7Y','hasIn8Y','hasIn9Y','hasIn10Y','hasIn11Y','hasIn12Y','hasIn13Y','hasIn14Y','hasIn15Y','hasIn16Y','hasIn17Y','hasIn18Y','hasIn19Y','hasIn20Y'])
for m,row1 in enumerate(csvReader1):
   addA[m]=row1
for n,row2 in enumerate(csvReader2):
   addB[n]=row2
for i in range(len(addA)):
   for j in range(len(addB)):
   if (addA[i][0]==addB[j][0] and addA[i][1]==addB[j][1]) or (addA[i][1]==addB[j][0] and addA[i][0]==addB[j][1]):
       #print (1)
       in93=addB[j][3]
       in94=addB[j][4]+addB[j][3]
       in95=addB[j][5]+addB[j][4]+addB[j][3]
```

```
        in96=addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]
        in97=addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in98=addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in99=addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+a
ddB[j][3]

    in00=addB[j][10]+addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+
addB[j][4]+addB[j][3]

    in01=addB[j][11]+addB[j][10]+addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]
+addB[j][5]+addB[j][4]+addB[j][3]

    in02=addB[j][12]+addB[j][11]+addB[j][10]+addB[j][9]+addB[j][8]+addB[j][7
]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in03=addB[j][13]+addB[j][12]+addB[j][11]+addB[j][10]+addB[j][9]+addB[j][
8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in04=addB[j][14]+addB[j][13]+addB[j][12]+addB[j][11]+addB[j][10]+addB[j]
[9]+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in05=addB[j][15]+addB[j][14]+addB[j][13]+addB[j][12]+addB[j][11]+addB[j]
[10]+addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB
[j][3]

    in06=addB[j][16]+addB[j][15]+addB[j][14]+addB[j][13]+addB[j][12]+addB[j]
[11]+addB[j][10]+addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+add
B[j][4]+addB[j][3]

    in07=addB[j][17]+addB[j][16]+addB[j][15]+addB[j][14]+addB[j][13]+addB[j]
```

```
[12]+addB[j][11]+addB[j][10]+addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]+ad
dB[j][5]+addB[j][4]+addB[j][3]

    in08=addB[j][18]+addB[j][17]+addB[j][16]+addB[j][15]+addB[j][14]+addB[j]
[13]+addB[j][12]+addB[j][11]+addB[j][10]+addB[j][9]+addB[j][8]+addB[j][7]+a
ddB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in09=addB[j][19]+addB[j][18]+addB[j][17]+addB[j][16]+addB[j][15]+addB[j]
[14]+addB[j][13]+addB[j][12]+addB[j][11]+addB[j][10]+addB[j][9]+addB[j][8]+
addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in10=addB[j][20]+addB[j][19]+addB[j][18]+addB[j][17]+addB[j][16]+addB[j]
[15]+addB[j][14]+addB[j][13]+addB[j][12]+addB[j][11]+addB[j][10]+addB[j][9]
+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j][3]

    in11=addB[j][21]+addB[j][20]+addB[j][19]+addB[j][18]+addB[j][17]+addB[j]
[16]+addB[j][15]+addB[j][14]+addB[j][13]+addB[j][12]+addB[j][11]+addB[j][10
]+addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j][4]+addB[j]
[3]

    in12=addB[j][22]+addB[j][21]+addB[j][20]+addB[j][19]+addB[j][18]+addB[j]
[17]+addB[j][16]+addB[j][15]+addB[j][14]+addB[j][13]+addB[j][12]+addB[j][11
]+addB[j][10]+addB[j][9]+addB[j][8]+addB[j][7]+addB[j][6]+addB[j][5]+addB[j
][4]+addB[j][3]

    writer1.writerow([addA[i][0],addA[i][1],addA[i][2],addA[i][3],int(in93)>
0,int(in94)>0,int(in95)>0,int(in96)>0,int(in97)>0,int(in98)>0,int(in99)>0,i
nt(in00)>0,int(in01)>0,int(in02)>0,int(in03)>0,int(in04)>0,int(in05)>0,int(
in06)>0,int(in07)>0,int(in08)>0,int(in09)>0,int(in10)>0,int(in11)>0,int(in1
2)>0])
        Break
```

4.4 计算预测准确率

$$准确率 = \frac{前\%n中断裂边总数}{断裂边总数 * \%n}$$

```
//计算准确率
r1=open('E:\dataProgram\datasets\PRA_suBYearAverageClustering.csv','r')  //经排序后的边信息文件（差值从小到大排序）
csvReader1=csv.reader(r1)

addA={}

for m,row1 in enumerate(csvReader1):
    addA[m]=row1
sum=[]
real1=0
for j in range(4,24):  //计算某年链路断裂总数
    ss=0
    for i in range(len(addA)):
    if(addA[i][j]=='FALSE'):
        ss=ss+1
    sum.append(ss)
a=[]
for z in range(len(sum)):
    s=[]
    acc=[]
    r=[]
    for j in range(10):  //按照排序的前 10%，20%...100%，计算断裂边数占总边数的比例
    s.append(int(sum[z]*(0.1*(j+1))))
    real=0
    for i in range(s[j]):
        if(addA[i][4+z]=='FALSE'):
            real=real+1
```

```
    r.append(real)
    acc.append(float(r[j])/float(s[j]))
    print float(r[j])/float(s[j])
    a.append(acc)

w=open('E:\dataProgram\data\PRA_Accuracy.csv','wb')  //将准确率写入文件
writer1=csv.writer(w)
for i in range(20):
   writer1.writerow(a[i])
```

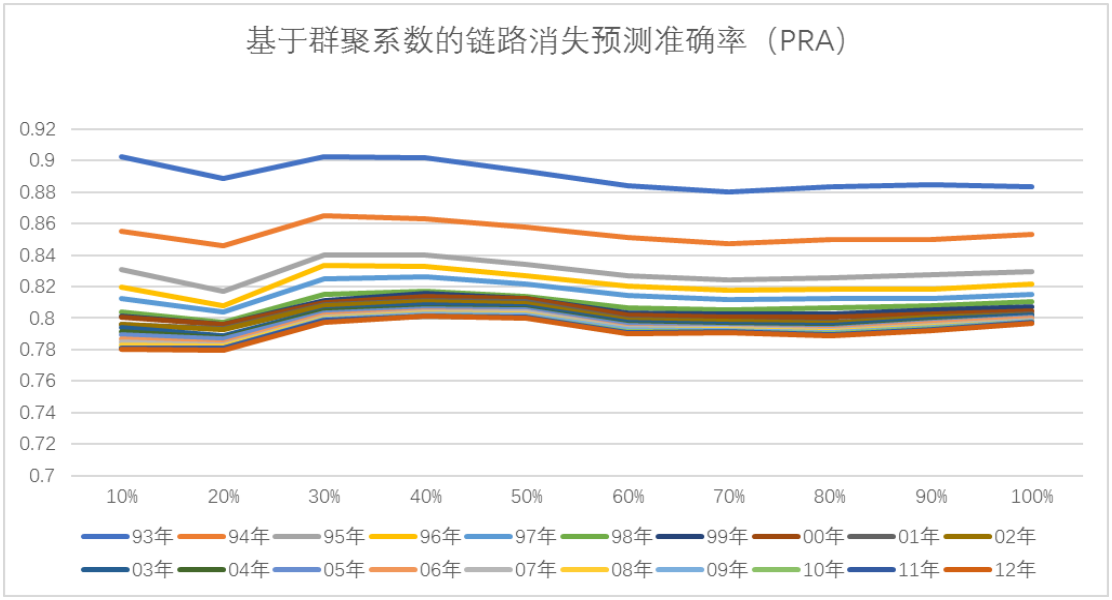
4.5 计算预测结果 AUC 值

```
//计算预测结果 AUC 值
r1=open('E:\dataProgram\data\DBLP_yearCluWU_order.csv','r')
csvReader1=csv.reader(r1)

csvfile1=file('E:\dataProgram\data\DBLP_AUC(WU).csv','wb')
writer1=csv.writer(csvfile1)

tableA={}

arrayF={}
arrayT={}

for m,row1 in enumerate(csvReader1):
   tableA[m]=row1
for k in range(20):
   m=0
   n=0
   for i in range(len(tableA)):  //统计断裂边数目与未断裂边数目，并生成正、负样本
    if tableA[i][4+k]=='FALSE':
       #print tableA[i][3]
```

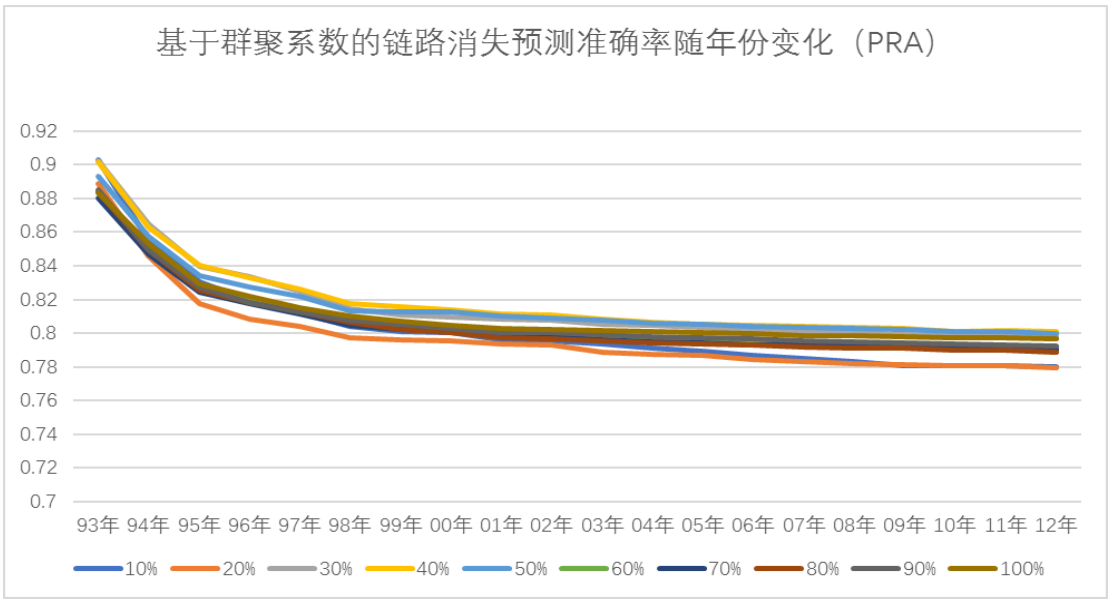
```
        arrayF[m]=tableA[i][3]
        m=m+1
    else:
        arrayT[n]=tableA[i][3]
        n=n+1
    tempa=0;
    auc=0;
    for i in range(m):
    for j in range(n):
        if arrayF[i]<arrayT[j]:  //如果正样本分数（群聚系数差值）小于负样本分数，频率取 1
            tempa=tempa+1
            #print 1
        if arrayF[i]==arrayT[j]:  //如果正样本分数（群聚系数差值）等于负样本分数，频率取 0.5
            tempa=tempa+0.5
            #print 11
    auc=tempa/(m*n)  //频率除以正负样本大小乘积，得到每年 AUC 值（AUC 值越大，预测越准确）
    writer1.writerow([93+k,auc])  //将每年的 AUC 值写入文件
    print auc
```

5 实验结果分析

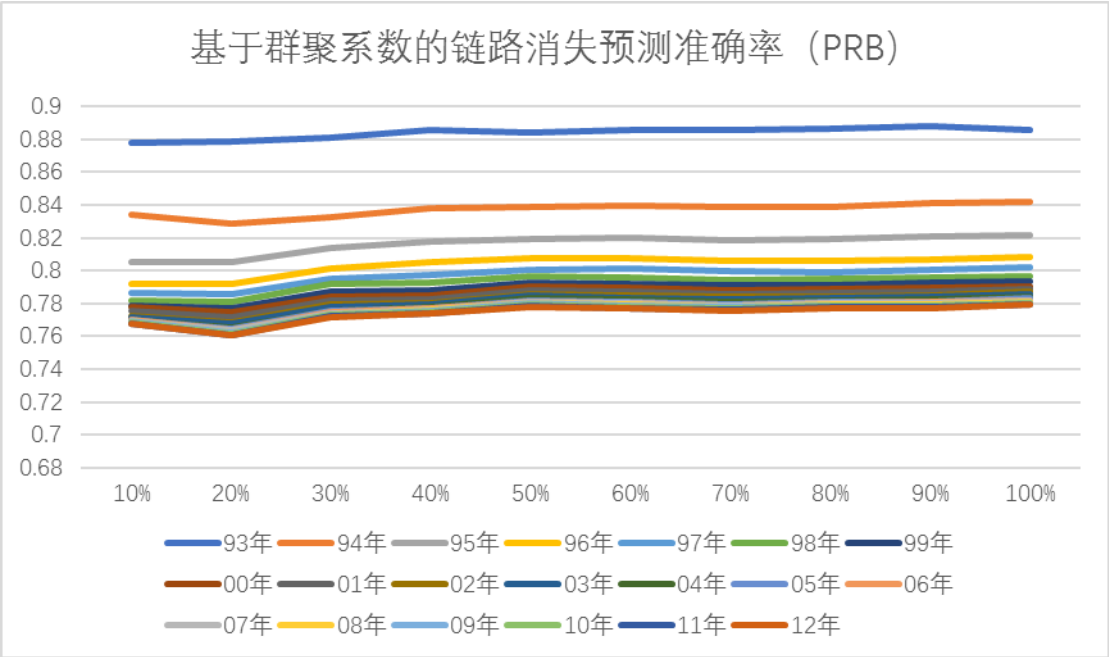
5.1 预测准确率分析



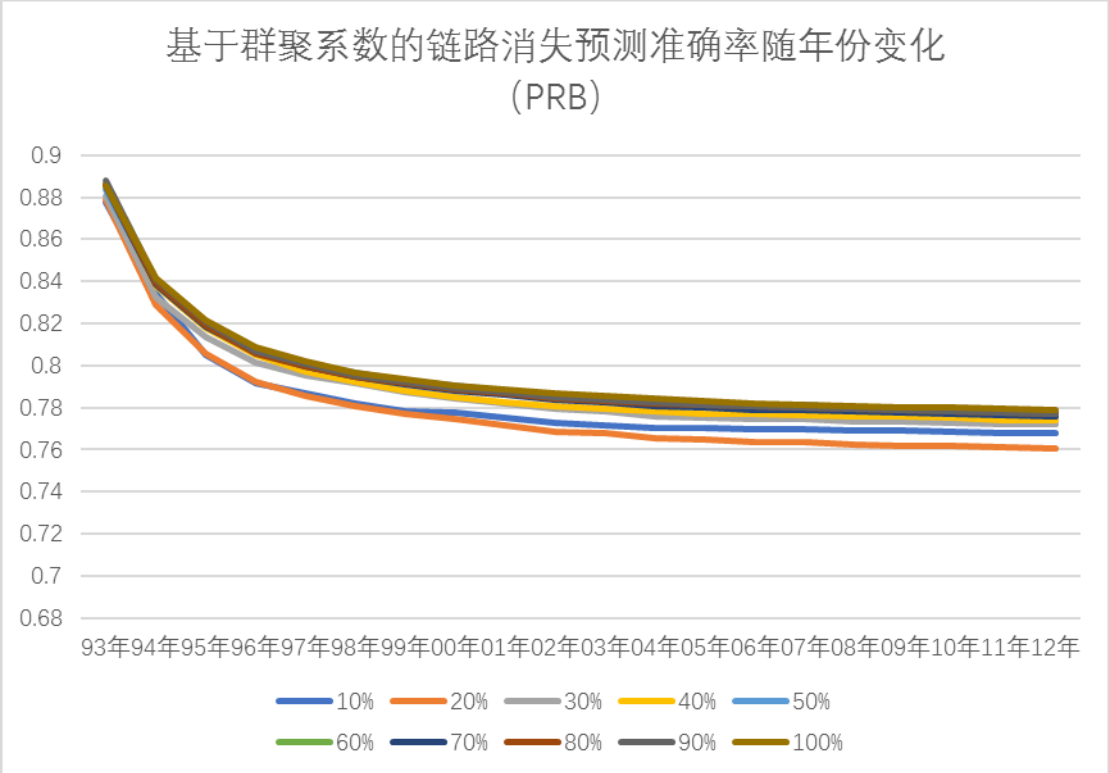
PRA 数据集基于群聚系数的链路消失预测准确率



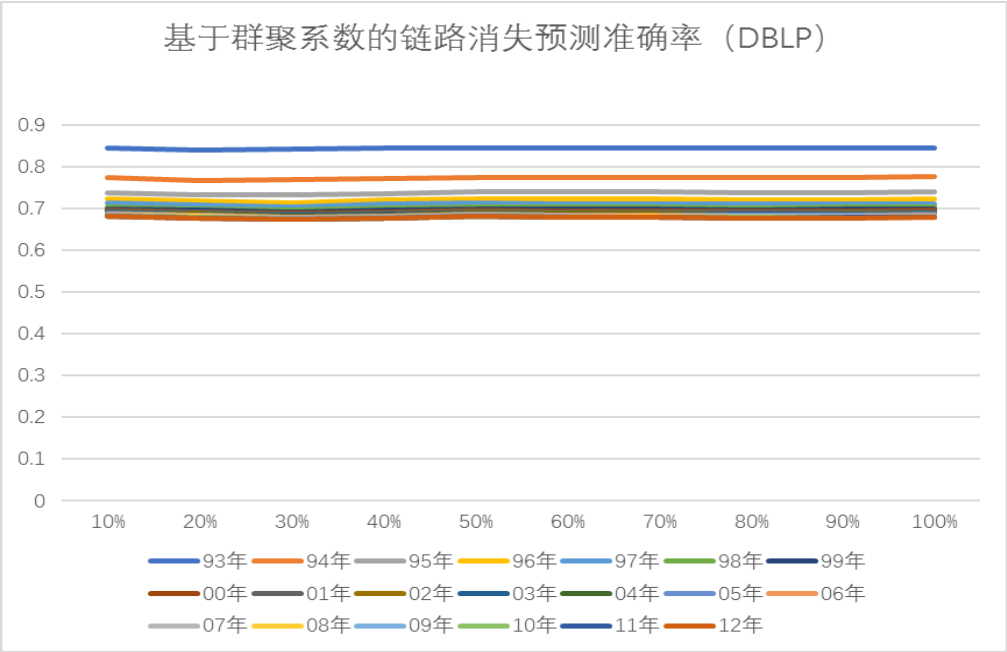
PRA 数据集基于群聚系数的链路消失预测随年份变化的准确率



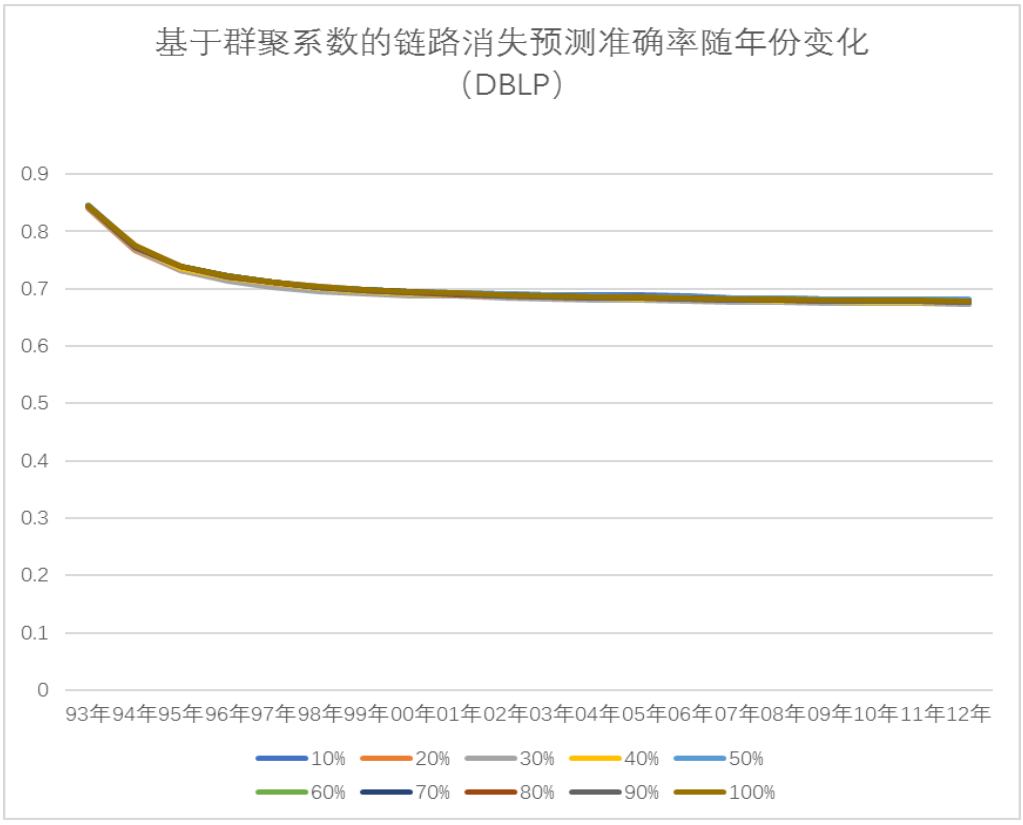
PRB 数据集基于群聚系数的链路消失预测准确率



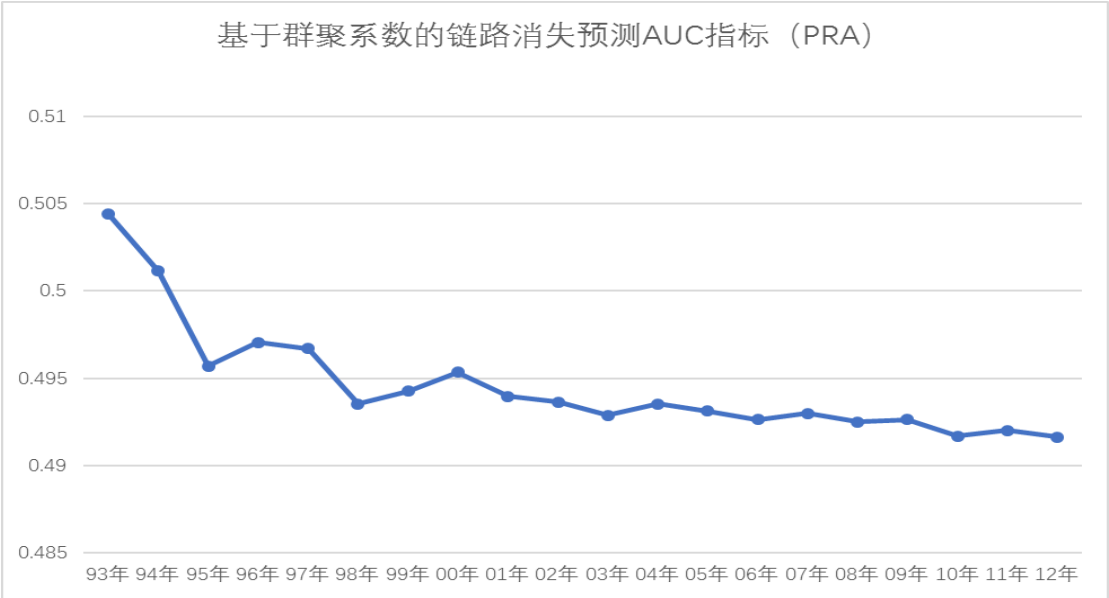
PRB 数据集基于群聚系数的链路消失预测随年份变化的准确率



DBLP 数据集基于群聚系数的链路消失预测准确率

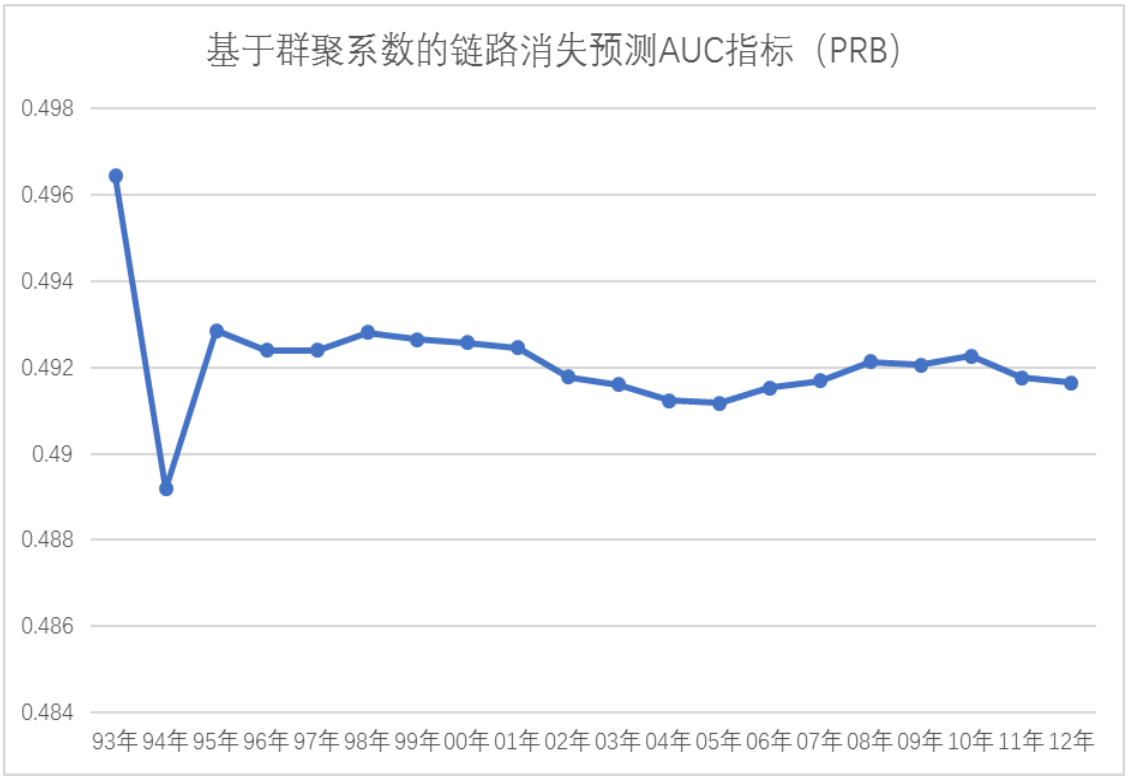


DBLP 数据集基于群聚系数的链路消失预测随年份变化的准确率

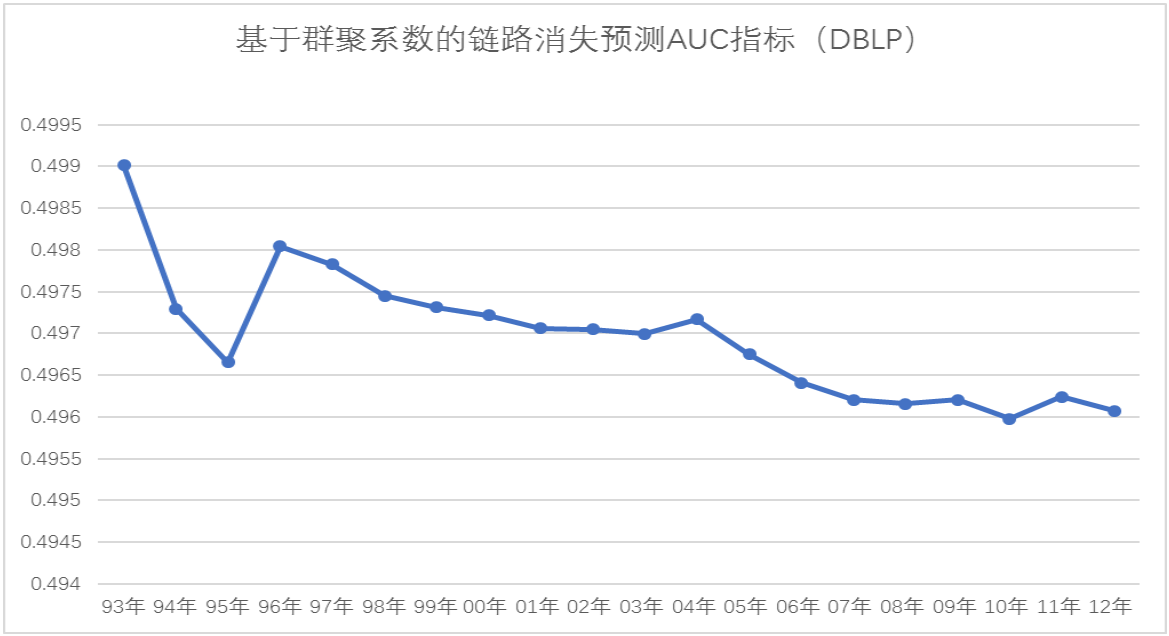


PRA 数据集基于群聚系数的链路消失预测 AUC 指标

5.2 基于 AUC 指标分析



PRB 数据集基于群聚系数的链路消失预测 AUC 指标



DBLP 数据集基于群聚系数的链路消失预测 AUC 指标

6 课程感悟

在选择开源软件基础这门课程之前，从来没听说过，这是第一次开课的课程。抱着好奇与疑惑的心态，和同学一起选择了这门课程。上课的第一天就发现，老师平易近人，亲和，会讲段子，整节课的氛围是活跃，轻松的。在之后的课程中，老师会耐心地问我们有没有听懂，贴心地问我们能不能看清 PPT，带领我们敲代码，熟悉操作，也会下发问卷调查学生对课程进度快慢，课程难度的看法及学生学习情况，从而更好地把握课程教学内容，使学生更好地学习。在学习的过程中，接触到了与 python 相关的编程以及开源软件仓库等专业知识，对数据爬取、过滤、分析、统计、可视化展示等都有了进一步的了解掌握，老师也给我们拓展了很多有用的课外知识。时代在不断的前进，软件技术也在不断地发展中。开源软件的发展相当的快速，作为一名专修软件工程的学生，我很开心选择了这门课程，让我学习扩充了知识，认识到不一样的软件技术，不一样的老师。

——陈冠伊

从这学期的《开源软件基础》课程中，我学到了 python 的基本用法，一些类库，以及简单的爬虫程序与数据分析的内容；在后续课程中学到了 git 的一些简单使用。这对日后的学习工作尤其是团队工作的展开提供了很大的帮助。任志磊老师认真负责有耐心，带领我们一起学习，照顾到零基础的学生，学习起来没有负担容易接受，还提供了很多学习资料以助我们深入学习。在使用 python 做数据分析的过程中，我体会到了这

门语言的简单便捷全面。从这次的课程设计中，我体会到了数据分析的乐趣以及团队协作的力量，收获颇多。整个课程学习过程中，我学习到了软件开源的精神。

——周芯宇

学完这门课给我的感受是，大学可能再也没有这么有意思的课了。听着段子跟着老师敲代码的感觉可以说是非常的棒了。通过这段时间的学习拓宽了我的视野，让我了解了跟开源软件相关的东西，让我对开源软件仓库有了更多的了解。在上这门课之前自己一直想学 Python，但是一直没勇气下手。这门课给了我学习 Python 的机会，同时让我学会了简单的爬虫。可以说是给我我一块学习数据分析的敲门砖，让我对以后的学习和编码有了更大的信心。基于 git 的团队合作也让我对现今流行的项目管理方式有了一定的了解。在本次项目中我们选择了复杂网络中的链路预测作为研究课题，让我对数据分析有了更加深刻的认识，同时也拓宽了自己的视野，不在局限于学校内自己的见识到的，也让我意识到了多接触开源项目的必要性。总之，这门课给我开启了一扇通往新世界的大门，相信自己将会在以后的学习中不断从中获益。

——崔芳宇

经过一个学期的简单学习，我了解到 Python 是一门比较容易学习的语言，因为它是非常高级的语言，比 C 和 C++这样的语言，还要高级几个层次，即使是一个没有任何基础的人都可以轻松学会。它不需要管理内存分配，不需要定义变量的类型即可使用，内置了很多数据类型直接使用，而不需要考虑怎么样创建这些类型，比如列表、字典、字符串这样高级的功能。

另外，用它写出的代码，可以直接运行，不需要进行编译的操作。还有一点，用它写出的代码非常短，事半而功倍。

而且，Python 是一门开发效率最高的语言，它比 C 有 6 倍的开发效率，简单来说，如果一个 C 开发人员工作 6 天，使用 Python 的开发人员只需要工作一天即可，意味着做 Python 开发人员可一周只上一天班。它比 C++有 2 倍的开发效率，它比 Java 和 C#也有 1.5 倍的开发效率。有这么高的开发效率，当然是用性能换来的代价，不过从目前硬件技术进步来看，目前的 CPU 计算能力普遍是过剩的，并且越来越多硬件成本降低，但人工的成本越来越贵。

无论是在 Windows 平台，还是 Linux 平台，都一样开发和调试。跨平台运行更加方便，如果没有使用平台差别的 API 接口，只要写一遍代码，就可以在 Windows 平台或 Linux 平台上运行。

Python 无论在商业上，还是教育上，都是免费使用，意味可以零成本进入学习它，使用它。Python 拥有众多功能完善的开发库可以使用。

众所周知，测试是软件开发里有相当大的工作量，比如模块测试，当开发人员把一个模块功能完成之后，需要测试这个模块是否正确，就需要搭建一堆测试代码，才可以验证的。这时，如果使用 C++或 Java 来写这些功能，显然没有使用 Python 来得快，从前面效率就可以看到。因此，通常就会变成这样的开发模式：发布的软件是使用 C++或 Java 开发，但测试的代码使用 Python 来开发。比如嵌入式系统涉及网络通讯方面，需要不断地向嵌入式系统发送网络数据和接收网络数据，就可以使用 Python 搭建一个测试环境出来，这样花费很少的时间，就可以对嵌入式系统进行验证，提高代码的质量，减少嵌入式系统与其它系统的调试时间，以及以后维护时间。

另外，通过使用 Python 语言编写众多的脚本，就可以提高自动化测试水平，每发布一个版本，就可以把以前的测试用例，全自动化测试一遍，这样会大大提高对软件快速发布的要求。

——王柄楠