山东大学计算机科学与技术学院

大数据分析实践课程实验报告

学号：202300130178	姓名：刘爽	班级：23 数据
实验题目：数据采样方法实践		
实验学时：4	实验日期：2025.9.18	
实验目标： 利用 Pandas 库实现多种数据采样和过滤的方法		
实验环境： python3.9，jupyter notebook		
实验过程： ① 库的导入与数据的读入		

```
import pandas as pd
import numpy as np
# 直接使用GBK编码（中文Windows系统常用）
data = pd.read_csv( filepath_or_buffer: "data.csv", encoding='gbk')
```

	from_dev	from_port	from_city	...	to_level	traffic	bandwidth
0	47	71	通辽	...	网络核心	49636052613	1.000000e+11
1	47	74	通辽	...	网络核心	50056871412	1.000000e+11
2	47	240	通辽	...	网络核心	49453581081	1.000000e+11
3	47	241	通辽	...	网络核心	49733361585	1.000000e+11
4	47	242	通辽	...	一般节点	50492573662	1.000000e+11
...	...	...	...	...	...	...	...
1113	1129	546	上海	...	网络核心	48731433404	1.000000e+11
1114	1129	514	上海	...	一般节点	50060666120	1.000000e+11
1115	36036	499	长春	...	网络核心	50545082113	1.000000e+11
1116	36422	346	天津	...	网络核心	50628787089	1.000000e+11
1117	2701	619	大连	...	网络核心	48753971761	1.000000e+11

[1118 rows x 10 columns]

（2）删除多余的空行并过滤掉 traffic 不等于 0 且 from_level=一般节点的数据

```
#过滤空行
data_cleaned=data.dropna(how='any')
#过滤traffic！=0 f一般
filtered_data1 = data_cleaned.loc[(data_cleaned['traffic'] != 0) ]
filtered_data=filtered_data1.loc[(data_cleaned['from_level'] == '一般节点')]
```

```
#显示清理空行的数据
print(data_cleaned)
#
print(filtered_data)
```

删除空行：

	from_dev	from_port	from_city	...	to_level	traffic	bandwidth
0	47	71	通辽	...	网络核心	49636052613	1.000000e+11
1	47	74	通辽	...	网络核心	50056871412	1.000000e+11
2	47	240	通辽	...	网络核心	49453581081	1.000000e+11
3	47	241	通辽	...	网络核心	49733361585	1.000000e+11
4	47	242	通辽	...	一般节点	50492573662	1.000000e+11
...	...	...	...	...	...	...	...
1113	1129	546	上海	...	网络核心	48731433404	1.000000e+11
1114	1129	514	上海	...	一般节点	50060666120	1.000000e+11
1115	36036	499	长春	...	网络核心	50545082113	1.000000e+11
1116	36422	346	天津	...	网络核心	50628787089	1.000000e+11
1117	2701	619	大连	...	网络核心	48753971761	1.000000e+11

[1118 rows x 10 columns]

过滤 traffic 不等于 0 且 from_level=一般节点的数据：

	from_dev	from_port	from_city	...	to_level	traffic	bandwidth
0	47	71	通辽	...	网络核心	49636052613	1.000000e+11
1	47	74	通辽	...	网络核心	50056871412	1.000000e+11
2	47	240	通辽	...	网络核心	49453581081	1.000000e+11
3	47	241	通辽	...	网络核心	49733361585	1.000000e+11
4	47	242	通辽	...	一般节点	50492573662	1.000000e+11
...	...	...	...	...	...	...	...
1097	2473	1460	吉林	...	一般节点	48409925693	1.000000e+11
1103	36036	18	长春	...	网络核心	48663350759	1.000000e+11
1104	63	6	通辽	...	一般节点	50355678076	1.000000e+11
1107	36036	52	长春	...	网络核心	49345226162	1.000000e+11
1115	36036	499	长春	...	网络核心	50545082113	1.000000e+11

[550 rows x 10 columns]

（3）对数据进行抽样

采取不同的采样方式采取 50 个样本并比较采样结果

① 加权采样：to_level 的值为一般节点与网络核心的权重之比为 1 ： 5

```python
#加权抽象
data_before_sample=filtered_data.copy()
columns=data_before_sample.columns
weight_sample=data_before_sample.copy()
#添加权重行
weight_sample['weight']=0
#设置权重
for i in weight_sample.index:
    if weight_sample.at[i,'to_level']=='一般节点':
        weight=1
    else:
        weight=5
    weight_sample.at[i,'weight']=weight
```

采取 50 个样本并比较采样结果

```python
#抽取50个样本
weight_sample_finish=weight_sample.sample(n=50,weights='weight')
weight_sample_finish=weight_sample_finish[columns]
print(weight_sample_finish)
```

```
324         98        152      呼和浩特  ...     网络核心  49863987888  1.000000e+11
304         63        230        通辽  ...     网络核心  50504074996  1.000000e+11
902         96        141      呼和浩特  ...     网络核心  51273380868  1.000000e+11
609         96        391      呼和浩特  ...     网络核心  48978587445  1.000000e+11
424        591        560        绥化  ...     网络核心  48754882922  1.000000e+11
587         96        141      呼和浩特  ...     网络核心  47941844052  1.000000e+11
121        474       1269       哈尔滨  ...     网络核心  50312177853  1.000000e+11
1005     36036        499        长春  ...     网络核心  49116324777  1.000000e+11
80         180        200      呼和浩特  ...     网络核心  51884294458  1.000000e+11
81         180        202      呼和浩特  ...     网络核心  49867223584  1.000000e+11
1103     36036         18        长春  ...     网络核心  48663350759  1.000000e+11
1107     36036         52        长春  ...     网络核心  49345226162  1.000000e+11
15          47        425        通辽  ...     网络核心  50796899329  1.000000e+11

[50 rows x 10 columns]
```

② 随机抽样

```python
#随机抽样
random_sample=filtered_data
random_sample_finish=random_sample.sample(n=50)
random_sample_finish=random_sample_finish[columns]
print(random_sample_finish)
```

```
160         591        1258          绥化  ...      一般节点  50322958171  1.000000e+11
344         180          34       呼和浩特  ...      网络核心  50352242512  1.000000e+11
324          96         152       呼和浩特  ...      网络核心  49665987866  1.000000e+11
330          96         336       呼和浩特  ...      网络核心  51277669375  1.000000e+11
165         591        1290          绥化  ...      网络核心  49758461056  1.000000e+11
340         180          20       呼和浩特  ...      网络核心  51392475128  1.000000e+11
308          63         286          通辽  ...      一般节点  50067368970  1.000000e+11
384         474         671         哈尔滨  ...      网络核心  51647234796  1.000000e+11
368         180         276       呼和浩特  ...      网络核心  51651922009  1.000000e+11
103         474         472         哈尔滨  ...      网络核心  49236653925  1.000000e+11
47           96         136       呼和浩特  ...      网络核心  49292630301  1.000000e+11
66          180          26       呼和浩特  ...      网络核心  51023900961  1.000000e+11
806         180          20       呼和浩特  ...      一般节点  50581993828  1.000000e+11
72          180          42       呼和浩特  ...      一般节点  49293665157  1.000000e+11
390         474         683         哈尔滨  ...      一般节点  50437152432  1.000000e+11

[50 rows x 10 columns]
```

③ 分层抽样：根据 to_level 的值进行分层采样，根据比例一般节点抽 17 个，网络核心抽 33 个

```
#分层抽样 一般节点17 网络核心节点33
ybjd=filtered_data.loc[filtered_data['to_level']=='一般节点']
wlhx=filtered_data.loc[filtered_data['to_level']=='网络核心']
fc_sample=pd.concat([ybjd.sample(17),wlhx.sample(33)])
print(fc_sample)
```

```
537          47         314          通辽  ...      网络核心  49136293957  1.000000e+11
29           63         230          通辽  ...      网络核心  50037668767  1.000000e+11
669          63         286          通辽  ...      网络核心  50318390185  1.000000e+11
75          180          84       呼和浩特  ...      网络核心  49100967003  1.000000e+11
341         180          26       呼和浩特  ...      网络核心  48797633450  1.000000e+11
587          96         141       呼和浩特  ...      网络核心  47941844052  1.000000e+11
8            47         251          通辽  ...      网络核心  50755299504  1.000000e+11
533          47         252          通辽  ...      网络核心  52135271000  1.000000e+11
1059         47         252          通辽  ...      网络核心  50358481161  1.000000e+11
430         591        1082          绥化  ...      网络核心  49355162407  1.000000e+11
351         180          90       呼和浩特  ...      网络核心  49446475351  1.000000e+11

[50 rows x 10 columns]
```

④ 系统抽样，等距抽样：

固定间隔从有序排列的总体中抽取样本，核心步骤如下：
排序总体：将总体中的所有个体按某种顺序（如自然顺序、编号顺序）排列；
计算间隔：根据总体规模 N 和样本量 n，计算抽样间隔 $k=N//n$（即每隔 k 个个体抽一个样本）；

随机起点：在第 1 个到第 k 个个体中随机选择一个作为起始点；
抽取样本：从起始点开始，每隔 k 个个体抽取一个，直到抽满 n 个样本。

```python
# 系统抽样（等距抽样）
1 usage
def systematic_sampling(data, n):
    N = len(data)
    k = N // n  # 计算抽样间隔
    # 随机选择起始点
    start = np.random.randint( low: 0, k)
    # 选择样本索引
    indices = [start + i * k for i in range(n) if (start + i * k) < N]
    return data.iloc[indices]

# 进行系统抽样
systematic_sample = systematic_sampling(data_before_sample, n: 50)
systematic_sample = systematic_sample[columns]
print(systematic_sample)
```

420	591	100	绥化	...	网络核心	51157112955	1.000000e+1
431	591	1104	绥化	...	网络核心	49411244329	1.000000e+1
442	787	51	玉溪	...	网络核心	50594027588	1.000000e+1
453	787	326	玉溪	...	一般节点	51285240797	1.000000e+1
492	47	250	通辽	...	网络核心	49014089485	1.000000e+1
531	47	250	通辽	...	网络核心	48844966451	1.000000e+1
542	63	10	通辽	...	一般节点	49716409605	1.000000e+1
553	63	230	通辽	...	网络核心	50530328860	1.000000e+1
564	96	117	呼和浩特	...	网络核心	49468205759	1.000000e+
604	96	134	呼和浩特	...	一般节点	49201392181	1.000000e+
640	47	252	通辽	...	一般节点	48030989242	1.000000e+1
686	63	70	通辽	...	网络核心	50424129643	1.000000e+1
728	2473	946	吉林	...	网络核心	52184126133	1.000000e+1
770	474	672	哈尔滨	...	一般节点	51263599555	1.000000e+
799	180	52	呼和浩特	...	一般节点	49553070694	1.000000e+
836	180	20	呼和浩特	...	一般节点	49701796126	1.000000e+
888	36036	20	长春	...	网络核心	48987594976	1.000000e+1
931	4069	1205	宁波	...	网络核心	52060473597	1.000000e+1
979	2473	1043	吉林	...	一般节点	49176857434	1.000000e+1
1021	2473	762	吉林	...	网络核心	47991126091	1.000000e+1
1059	47	252	通辽	...	网络核心	50358481161	1.000000e+1
1107	36036	52	长春	...	网络核心	49345226162	1.000000e+1

[50 rows x 10 columns]

⑤ 整群抽样：
整群抽样将总体划分为若干个群（Cluster），每个群包含若干个体，然后随机抽取部分群，对选中的群内所有个体进行调查，核心步骤如下：
划分群：将总体按某种规则（如地理区域、时间区间等）分成若干个互不重叠的群；

随机抽群：从所有群中随机选择 n 个群；
全群调查：对选中的每个群内的所有个体进行抽样

```
 整群抽样（假设按某个字段分组抽样）
ge
cluster_sampling(data, n, cluster_column):
# 获取所有唯一的群
clusters = data[cluster_column].unique()

# 随机选择n个群
selected_clusters = np.random.choice(clusters, size=min(n, len(clusters)), replace=False)

# 选择这些群的所有数据
cluster_sample = data[data[cluster_column].isin(selected_clusters)]

return cluster_sample
```

本实验中按照 to_level 列进行整群抽样

```
# 按'to_level'列进行整群抽样
data_with_clusters = data_before_sample.copy()
data_with_clusters['to_level'] = data_with_clusters.index // 10  # 每10行一个群

cluster_sample = cluster_sampling(data_with_clusters, n: 5, cluster_column: 'to_level')  # 使用
cluster_sample = cluster_sample[columns]

print("按'to_level'列整群抽样结果:")
print(cluster_sample)
```

```
按'to_level'列整群抽样结果:
     from_dev  from_port from_city  ... to_level      traffic     bandwidth
50         96        155      呼和浩特  ...        5  51538493830  1.000000e+11
51         96        156      呼和浩特  ...        5  50654404568  1.000000e+11
52         96        157      呼和浩特  ...        5  50096366926  1.000000e+11
53         96        158      呼和浩特  ...        5  51342500152  1.000000e+11
54         96        159      呼和浩特  ...        5  51625089370  1.000000e+11
55         96        336      呼和浩特  ...        5  51600306541  1.000000e+11
56         96        346      呼和浩特  ...        5  47759033178  1.000000e+11
57         96        379      呼和浩特  ...        5  49400869697  1.000000e+11
58         96        383      呼和浩特  ...        5  50609333179  1.000000e+11
59         96        391      呼和浩特  ...        5  51570663870  1.000000e+11
90        180        260      呼和浩特  ...        9  48006842653  1.000000e+11
91        180        264      呼和浩特  ...        9  50106121660  1.000000e+11
92        180        272      呼和浩特  ...        9  52854391127  1.000000e+11
93        180        276      呼和浩特  ...        9  51775514286  1.000000e+11
94        180        485      呼和浩特  ...        9  52460156321  1.000000e+11
95        474        359       哈尔滨   ...        9  51299508559  1.000000e+11
```

结论与体会：

（一）数据预处理的重要性

实验初期的数据清洗步骤（删除空行、过滤无效数据）是保证抽样质量的基础。原始数据中存在的空行和不符合条件的记录（如 traffic=0、from_level 非 "一般节点"）若不处理，会直接影响抽样结果的准确性。通过 dropna 方法和条件过滤，得到了干净、有效的分析数据集，为后续抽样工作奠定了良好基础。

（二）不同抽样方法的效果对比

1. **加权抽样**：通过为 "网络核心" 节点赋予 5 倍于 "一般节点" 的权重，最终样本中 "网络核心" 节点的占比显著提高，能够突出重点关注对象的特征。这种方法适合需要强化特定类别样本代表性的场景，能按照预设权重比例获取样本。
2. **随机抽样**：完全基于概率的抽样方式，样本中 "一般节点" 和 "网络核心" 节点的比例与总体分布基本一致，具有无偏性特点。该方法实现简单，在没有特殊研究目标时，能较好地反映总体的真实情况。
3. **分层抽样**：按照预设比例（一般节点 17 个，网络核心 33 个）从不同层中分别抽样，再将结果组合。这种方法保证了每层样本的代表性，能精确控制各层样本量，适合总体中不同类别差异较大的情况。
4. **系统抽样与整群抽样**：系统抽样通过固定间隔抽样，样本在总体中分布均匀；整群抽样将数据按索引划分为若干群，抽取整群作为样本，效率较高。两种方法各有侧重，系统抽样适合总体有序且无周期性的数据，整群抽样适合可按自然群体划分的数据。