大数据分析实践实验报告

实验一 数据采样方法实践

一、实验目标

利用 Pandas 库实现多种数据采样和过滤的方法

二、实验环境

python3，jupyter notebook

三、实验过程

[1]:
```
import pandas as pd
from pandas import DataFrame
import numpy as np

primitive_data = pd.read_csv(r"C:\Users\边鑫磊\Desktop\data.csv", encoding='gbk')
primitive_data
```

[1]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth
0	47	71	通辽	一般节点	1756	585	北京	网络核心	49636052613	1.000000e+11
1	47	74	通辽	一般节点	1756	776	北京	网络核心	50056871412	1.000000e+11
2	47	240	通辽	一般节点	1756	802	北京	网络核心	49453581081	1.000000e+11
3	47	241	通辽	一般节点	1997	464	天津	网络核心	49733361585	1.000000e+11
4	47	242	通辽	一般节点	474	672	哈尔滨	一般节点	50492573662	1.000000e+11
...	...	...	...	...	...	...	...	...	...	...
1113	1129	546	上海	网络核心	2050	502	石家庄	网络核心	48731433404	1.000000e+11
1114	1129	514	上海	网络核心	2473	946	吉林	一般节点	50060666120	1.000000e+11
1115	36036	499	长春	一般节点	1257	178	上海	网络核心	50545082113	1.000000e+11
1116	36422	346	天津	网络核心	1997	41	天津	网络核心	50628787089	1.000000e+11
1117	2701	619	大连	网络核心	2549	1070	沈阳	网络核心	48753971761	1.000000e+11

1118 rows × 10 columns

导入库&导入数据

[2]:
```
primitive_data_1=primitive_data.dropna(how='any')
primitive_data_1
```

[2]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth
0	47	71	通辽	一般节点	1756	585	北京	网络核心	49636052613	1.000000e+11
1	47	74	通辽	一般节点	1756	776	北京	网络核心	50056871412	1.000000e+11
2	47	240	通辽	一般节点	1756	802	北京	网络核心	49453581081	1.000000e+11
3	47	241	通辽	一般节点	1997	464	天津	网络核心	49733361585	1.000000e+11
4	47	242	通辽	一般节点	474	672	哈尔滨	一般节点	50492573662	1.000000e+11
...	...	...	...	...	...	...	...	...	...	...
1113	1129	546	上海	网络核心	2050	502	石家庄	网络核心	48731433404	1.000000e+11
1114	1129	514	上海	网络核心	2473	946	吉林	一般节点	50060666120	1.000000e+11
1115	36036	499	长春	一般节点	1257	178	上海	网络核心	50545082113	1.000000e+11
1116	36422	346	天津	网络核心	1997	41	天津	网络核心	50628787089	1.000000e+11
1117	2701	619	大连	网络核心	2549	1070	沈阳	网络核心	48753971761	1.000000e+11

1118 rows × 10 columns

删除多余的空行（采用 dropna 方法并指定参数为 any 删除多余的空行）

[3]:
```
data_before_filter=primitive_data_1
data_after_filter_1=data_before_filter.loc[data_before_filter["traffic"]!=0]
data_after_filter_2=data_after_filter_1.loc[data_after_filter_1["from_level"]=='一般节点']
data_after_filter_2
```

[3]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth
0	47	71	通辽	一般节点	1756	585	北京	网络核心	49636052613	1.000000e+11
1	47	74	通辽	一般节点	1756	776	北京	网络核心	50056871412	1.000000e+11
2	47	240	通辽	一般节点	1756	802	北京	网络核心	49453581081	1.000000e+11
3	47	241	通辽	一般节点	1997	464	天津	网络核心	49733361585	1.000000e+11
4	47	242	通辽	一般节点	474	672	哈尔滨	一般节点	50492573662	1.000000e+11
...	...	...	...	...	...	...	...	...	...	...
1097	2473	1460	吉林	一般节点	591	586	绥化	一般节点	48409925693	1.000000e+11
1103	36036	18	长春	一般节点	3443	650	青岛	网络核心	48663350759	1.000000e+11
1104	63	6	通辽	一般节点	36036	20	长春	一般节点	50355678076	1.000000e+11
1107	36036	52	长春	一般节点	1129	171	上海	网络核心	49345226162	1.000000e+11
1115	36036	499	长春	一般节点	1257	178	上海	网络核心	50545082113	1.000000e+11

550 rows × 10 columns

进行过滤（得到 traffic != 0 且 from_level =一般节点的数据）

(一) 加权抽样

[8]:
```
data_before_sample=data_after_filter_2
columns=data_before_sample.columns
weight_sample=data_before_sample.copy()
weight_sample['weight']=0
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

weight_sample_finish=weight_sample.sample(n=50,weights='weight')
#data_before_sample=data_before_sample[columns]
weight_sample_finish
```

[8]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth	weight
343	180	30	呼和浩特	一般节点	474	1228	哈尔滨	一般节点	50284244775	1.000000e+11	1
443	787	52	玉溪	一般节点	2360	215	太原	网络核心	49322809158	1.000000e+11	5
587	96	141	呼和浩特	一般节点	3213	246	重庆	网络核心	47941844052	1.000000e+11	5
54	96	159	呼和浩特	一般节点	2360	266	太原	网络核心	51625089370	1.000000e+11	5
68	180	30	呼和浩特	一般节点	235	1661	北京	网络核心	49596659754	1.000000e+11	5
307	63	282	通辽	一般节点	1756	18	北京	网络核心	49252024885	1.000000e+11	5
157	591	1106	绥化	一般节点	36036	939	长春	一般节点	50954337724	1.000000e+11	1
341	180	26	呼和浩特	一般节点	1756	796	北京	网络核心	48797633450	1.000000e+11	5
993	36036	18	长春	一般节点	2194	450	唐山	网络核心	49826827167	1.000000e+11	5
406	474	1473	哈尔滨	一般节点	36422	394	天津	网络核心	48378712039	1.000000e+11	5
46	96	135	呼和浩特	一般节点	2549	836	沈阳	网络核心	48380335564	1.000000e+11	5

加权抽样是一种基于概率的抽样方法，其核心思想是根据样本的特定属性或重要性赋予不同的抽样权重，使具有某些特征的样本被抽中的概率相应提高。适用于总体中不同子群体重要性不同的情况，通过调整权重可以确保重要群体在样本中有足够的代表性

具体实现时，首先创建一个权重列，通过条件判断为'to_level'字段的不同值分配不同的权重值（一般节点权重为 1，网络核心权重为 5）。然后使用 Pandas 的 sample()方法，指定 weights 参数为刚创建的权重列，实现按权重比例抽取 50 个样本

(二) 随机抽样

[10]:
```
random_sample=data_before_sample
random_sample_finish=random_sample.sample(n=50)
random_sample_finish=random_sample_finish[columns]
random_sample_finish
```

[10]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth
379	474	473	哈尔滨	一般节点	474	1374	哈尔滨	一般节点	50592340509	1.000000e+11
33	63	286	通辽	一般节点	180	52	呼和浩特	一般节点	49725190236	1.000000e+11
20	63	54	通辽	一般节点	235	100	北京	网络核心	49256234165	1.000000e+11
292	63	6	通辽	一般节点	2841	418	郑州	网络核心	51392218854	1.000000e+11
773	2473	762	吉林	一般节点	180	84	呼和浩特	一般节点	49702910101	1.000000e+11
291	47	427	通辽	一般节点	3227	766	济南	网络核心	49389391674	1.000000e+11
535	47	259	通辽	一般节点	2841	341	郑州	网络核心	51012708275	1.000000e+11
58	96	383	呼和浩特	一般节点	5242	783	西安	网络核心	50609333179	1.000000e+11
341	180	26	呼和浩特	一般节点	1756	796	北京	网络核心	48797633450	1.000000e+11
1073	47	417	通辽	一般节点	1756	1029	北京	网络核心	49459363742	1.000000e+11
371	474	360	哈尔滨	一般节点	3227	530	济南	网络核心	49027966353	1.000000e+11
305	63	232	通辽	一般节点	2549	1066	沈阳	网络核心	49269663214	1.000000e+11
828	47	314	通辽	一般节点	474	1470	哈尔滨	一般节点	50910415109	1.000000e+11
348	180	50	呼和浩特	一般节点	96	460	呼和浩特	一般节点	49427879868	1.000000e+11
493	47	251	通辽	一般节点	63	286	通辽	一般节点	49254299513	1.000000e+11
284	47	252	通辽	一般节点	5058	118	南宁	一般节点	49295040137	1.000000e+11
566	591	586	绥化	一般节点	1129	514	上海	网络核心	49995510331	1.000000e+11

随机抽样是最基本也是最简单的概率抽样方法，其核心原则是总体中的每一个样本都有完全相等且独立的被抽中机会。这种方法基于随机数生成原理，完全排除人为干预和主观选择，能够保证样本的无偏性和代表性

代码的实现比较简单，直接调用 Pandas 的 sample()方法，仅需指定需要抽取的样本数量 n=50，不需要设置任何权重、分层或其他参数。Pandas 内部使用随机数生成器来实现真正的随机选择，确保每个样本被抽中的概率完全相同

（三）分层抽样

[11]:
```
ybjd=data_before_sample.loc[data_before_sample['to_level']=='一般节点']
wlhx=data_before_sample.loc[data_before_sample['to_level']=='网络核心']
after_sample=pd.concat([ybjd.sample(17),wlhx.sample(33)])
after_sample
```

[11]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth
622	180	20	呼和浩特	一般节点	36036	499	长春	一般节点	49636788433	1.000000e+11
828	47	314	通辽	一般节点	474	1470	哈尔滨	一般节点	50910415109	1.000000e+11
86	180	226	呼和浩特	一般节点	36036	20	长春	一般节点	49248544673	1.000000e+11
48	96	141	呼和浩特	一般节点	474	422	哈尔滨	一般节点	49429192047	1.000000e+11
111	474	673	哈尔滨	一般节点	2473	799	吉林	一般节点	48852033101	1.000000e+11
912	47	242	通辽	一般节点	47	242	通辽	一般节点	49820071586	1.000000e+11
834	180	264	呼和浩特	一般节点	591	19	绥化	一般节点	50578150343	1.000000e+11
541	63	6	通辽	一般节点	2473	1043	吉林	一般节点	48954016072	1.000000e+11
387	474	677	哈尔滨	一般节点	474	672	哈尔滨	一般节点	50850714694	1.000000e+11
604	96	134	呼和浩特	一般节点	2473	1460	吉林	一般节点	49201392181	1.000000e+11
19	63	12	通辽	一般节点	180	252	呼和浩特	一般节点	49290094443	1.000000e+11
1057	47	243	通辽	一般节点	2473	769	吉林	一般节点	49117847542	1.000000e+11
732	96	141	呼和浩特	一般节点	36036	499	长春	一般节点	47474335913	1.000000e+11
775	96	134	呼和浩特	一般节点	180	98	呼和浩特	一般节点	51993612239	1.000000e+11
980	4360	472	南京	一般节点	63	286	通辽	一般节点	49837582425	1.000000e+11
530	47	249	通辽	一般节点	2473	799	吉林	一般节点	49803820036	1.000000e+11
25	63	70	通辽	一般节点	180	264	呼和浩特	一般节点	49630519508	1.000000e+11

分层抽样是一种先将总体按照某个重要特征划分为若干个互不重叠的子总体（即为层），然后在每个层内独立进行随机抽样的方法。这种方法的优势在于能够确保每个层都在最终样本中有适当的代表性，适用于总体内部差异较大、分布不均匀的情况

实现时，首先根据'to_level'字段将数据分为"一般节点"和"网络核心"两个层。然后按照预先确定的比例（一般节点 17 个，网络核心 33 个）分别在每个层内进行随机抽样。最后使用 pd.concat()方法将两个层的抽样结果合并，形成最终的 50 个样本

（四）系统抽样

[19]:
```
# 系统抽样
def systematic_sampling(data, n):
    k = len(data) // n
    start = np.random.randint(0, k)
    indices = range(start, len(data), k)
    systematic_sample = data.iloc[indices[:n]]
    return systematic_sample

systematic_sample_finish = systematic_sampling(data_before_sample, 50)
systematic_sample_finish = systematic_sample_finish[columns]
systematic_sample_finish
```

[19]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth
5	47	243	通辽	一般节点	96	124	呼和浩特	一般节点	49942713747	1.000000e+11
16	47	427	通辽	一般节点	1997	213	天津	网络核心	48476552334	1.000000e+11
27	63	224	通辽	一般节点	180	20	呼和浩特	一般节点	48761650539	1.000000e+11
38	96	111	呼和浩特	一般节点	3227	468	济南	网络核心	48575689938	1.000000e+11
49	96	152	呼和浩特	一般节点	47	314	通辽	一般节点	51981076188	1.000000e+11
60	96	399	呼和浩特	一般节点	1756	1117	北京	网络核心	50243694923	1.000000e+11
71	180	38	呼和浩特	一般节点	1385	2778	广州	网络核心	51626689601	1.000000e+11
82	180	205	呼和浩特	一般节点	1536	47	鄂尔多斯	网络核心	49925756570	1.000000e+11
93	180	276	呼和浩特	一般节点	36272	235	太原	网络核心	51775514286	1.000000e+11
108	474	670	哈尔滨	一般节点	2841	483	郑州	网络核心	50632622266	1.000000e+11
119	474	1246	哈尔滨	一般节点	3227	705	济南	网络核心	50954049859	1.000000e+11
130	474	1470	哈尔滨	一般节点	2473	1460	吉林	一般节点	49047884661	1.000000e+11
141	591	60	绥化	一般节点	36422	446	天津	网络核心	48726793145	1.000000e+11

系统抽样是一种将总体样本按某种顺序排列后，按照固定间隔抽取样本的方法。其基本步骤是：先计算抽样间隔（总体大小除以所需样本量），然后随机确定一个起始点，最后从起始点开始每隔固定间隔抽取一个样本。这种方法当总体很大时，能够保证样本在总体中的均匀分布，避免样本过于集中

在具体实现时，首先计算抽样间隔 k（总体样本数除以 50），然后使用 np.random.randint()随机生成一个介于 0 到 k-1 之间的起始点。接着生成一个从起始点开始、间隔为 k 的索引序列，使用这些索引从原始数据中选取对应的样本。最后确保只取前 50 个样本，防止在间隔计算有余数时抽取过多样本

（五）整群抽样

[14]:
```
# 整群抽样
def cluster_sampling(data, cluster_column, n_clusters):
    clusters = data[cluster_column].unique()
    selected_clusters = np.random.choice(clusters, n_clusters, replace=False)
    cluster_sample = data[data[cluster_column].isin(selected_clusters)]
    return cluster_sample

#抽取2个群
cluster_sample_finish = cluster_sampling(data_before_sample, 'to_city', 2)
cluster_sample_finish = cluster_sample_finish[columns]
cluster_sample_finish
```

[14]:

	from_dev	from_port	from_city	from_level	to_dev	to_port	to_city	to_level	traffic	bandwidth
43	96	124	呼和浩特	一般节点	47	243	通辽	一般节点	49986988230	1.000000e+11
45	96	134	呼和浩特	一般节点	47	252	通辽	一般节点	49416652053	1.000000e+11
49	96	152	呼和浩特	一般节点	47	314	通辽	一般节点	51981076188	1.000000e+11
59	96	391	呼和浩特	一般节点	47	417	通辽	一般节点	51570663870	1.000000e+11
65	180	20	呼和浩特	一般节点	63	224	通辽	一般节点	50551711152	1.000000e+11
74	180	52	呼和浩特	一般节点	63	286	通辽	一般节点	49155371449	1.000000e+11
87	180	252	呼和浩特	一般节点	63	12	通辽	一般节点	49137975001	1.000000e+11
91	180	264	呼和浩特	一般节点	63	70	通辽	一般节点	50106121660	1.000000e+11
110	474	672	哈尔滨	一般节点	47	242	通辽	一般节点	51555817613	1.000000e+11
137	591	23	绥化	一般节点	63	6	通辽	一般节点	49462640634	1.000000e+11
167	787	51	玉溪	一般节点	4561	1033	成都	网络核心	51033155364	1.000000e+11
170	787	60	玉溪	一般节点	4561	1025	成都	网络核心	49992676292	1.000000e+11
177	787	325	玉溪	一般节点	4561	1087	成都	网络核心	48864832885	1.000000e+11

整群抽样是一种先将总体划分为若干个自然的群组（集群），然后随机抽取部分群组，并对抽中群组内的所有个体进行全面调查的抽样方法。这种方法适用于总体自然形成群组结构的情况，适合大规模调查

具体实现时，首先使用 unique()方法获取'to_city'字段的所有唯一值，这些值自然形成了不同的群组。然后使用 np.random.choice()随机选择指定数量的群组（这里选择 2 个）。最后使用 isin()方法筛选出属于这些被选中群组的所有样本（群组大小的差异会影响最终样本的数量）

四、实验总结

本实验通过实践 Pandas 库的多种数据采样和过滤方法，成功实现了对网络节点数据的处理与分析。实验中首先清洗了数据，消除了空值干扰，随后通过条件过滤精准提取了有效数据。在此基础上，我们实现了加权抽样、随机抽样、分层抽样、系统抽样和整群抽样五种抽样方法，每种方法都体现了不同的抽样理念和应用场景

通过本次实验，不仅初步掌握了 Pandas 库在数据采样方面的功能，更深入理解了不同抽样方法的适用条件和实际效果。加权抽样能突出重要群体，分层抽样保证子群体代表性，系统抽样实现均匀分布，整群抽样保持群体完整性，这些方法为后续的数据分析工作做好了基础，体现了数据采样在实际数据处理中的关键作用