|  |  |  |
| --- | --- | --- |
| **Nhóm 10** | | |
| **STT** | **MSSV** | **HỌ VÀ TÊN** |
| 46 | 21057901 | Nguyễn Đa Nghiêm |
| 47 | 21000615 | Lê Hải Ngọc |
| 48 | 20000841 | Nguyễn Thái Ngọc |
| 49 | 21070711 | Lê Phước Nguyên |
| 50 | 21068021 | Phan Nguyễn Khôi Nguyên |

**Lấy mẫu ngẫu nhiên đơn giản (Simple Random Sampling):**

Đặc điểm chính: Mỗi đối tượng trong tổng thể có cơ hội bằng nhau để được chọn vào mẫu.

***Ưu điểm:***

* Dễ thực hiện và hiểu.
* Đảm bảo tính ngẫu nhiên và khả năng biểu diễn của mẫu.

***Nhược điểm:***

Không đảm bảo sự đại diện của mẫu đối với từng phân khúc trong tổng thể.

**Lấy mẫu có hệ thống (Systematic Sampling):**

Đặc điểm chính: Là lấy mẫu bằng cách chọn một điểm bắt đầu và điểm kết thúc, sau đó lần lượt chọn phần tử thứ k từ quần thể. Mỗi k đối tượng (ví dụ: mỗi 5 hoặc mỗi 10) được chọn vào mẫu sau một chu kỳ xác định.

***Ưu điểm:***

* Dễ thực hiện.
* Phù hợp khi có sự sắp xếp đều đặn trong tổng thể.

***Nhược điểm:***

Có thể tạo ra mẫu có đặc điểm đặc biệt nếu có mô hình trong dãy số được sắp xếp.

**Lấy mẫu phân tầng (Stratified Sampling):**

Đặc điểm chính: chia quần thể thành nhiều nhóm nhỏ, mỗi nhóm có cùng đặc tính, sau đó lấy mẫu bằng cách chọn ngẫu nhiên từ các nhóm nhỏ đó.

***Ưu điểm:***

* Đảm bảo đại diện của mẫu đối với từng phân khúc.
* Giảm sai số mẫu so với ngẫu nhiên đơn.

***Nhược điểm:***

Yêu cầu thông tin về tỷ lệ phân phối trong từng phân khúc.

**Phân khúc có trọng số (Weighted Random Sampling):**

Đặc điểm chính: Giống với phân khúc, nhưng mỗi phần tử trong mẫu có một trọng số khác nhau.

Ưu điểm:

* Cho phép xem xét mức độ quan trọng khác nhau của các phân khúc.

Nhược điểm:

Đòi hỏi thông tin về trọng số và có thể phức tạp hơn so với phân khúc thông thường.

**Lấy mẫu theo cụm (Cluster Sampling):**

Đặc điểm chính: Tổng thể được chia thành các cụm (clusters), và một số cụm được chọn để nghiên cứu. Tất cả các đối tượng trong cụm được bao gồm vào mẫu.

Ưu điểm:

* Dễ thực hiện, đặc biệt là khi tổng thể có sự phân cấp tự nhiên thành các nhóm lớn.
* Giảm chi phí so với việc chọn ngẫu nhiên từ toàn bộ tổng thể.

Nhược điểm:

Cụm có thể không đại diện nếu sự đồng nhất không đảm bảo.

**Giải thích code :**

**coffee\_ratings.sample(n=5, random\_state=19000113)**

Giải thích :

Hàm sample() được sử dụng để lấy mẫu ngẫu nhiên. Trong đó:

**+ n=5**: Đây là số lượng dòng được lấy mẫu, trong trường hợp này là 5 dòng.

**+ random\_state=19000113**: Đây là một seed number được sử dụng để tạo ra một loạt các số ngẫu nhiên. Sử dụng cùng một seed number sẽ đảm bảo rằng kết quả của quá trình lấy mẫu ngẫu nhiên sẽ không thay đổi mỗi khi chạy lại code. Điều này hữu ích trong việc tái tạo kết quả và làm cho kết quả của mẫu ngẫu nhiên được dự đoán được.

**sample\_size = 5** # kích thước mẫu muốn lấy

**pop\_size = len(coffee\_ratings)** # kích thước quần thể (tổng số dòng của bảng coffee\_ratings)

**print(pop\_size)**

**interval = pop\_size // sample\_size** # khoảng cách giữa các dòng được chọn ngẫu nhiên .Trong trường hợp này, nó được tính bằng cách chia tổng số dòng của bảng dữ liệu cho kích thước mẫu.

**print(interval)**

**coffee\_ratings.iloc[::interval]** # lấy mẫu ngẫu nhiên từ bảng coffee\_ratings với khoảng cách interval giữa các dòng được chọn ngẫu nhiên

**iloc**: Là một phương thức được sử dụng để trích xuất dữ liệu dựa trên chỉ số của dòng và cột.

**::interval**: Đây là cú pháp slicing trong Python. Trong trường hợp này, **::** có ý nghĩa là trích xuất từ đầu đến cuối của “**coffee\_ratings**” và **interval** được sử dụng như một bước nhảy.

Ví dụ, nếu **interval** là 2, thì code này sẽ lấy mỗi hàng thứ 2 từ dataframe hoặc series "coffee\_ratings", bắt đầu từ hàng đầu tiên. Điều này sẽ tạo ra một mẫu dữ liệu nhỏ từ dữ liệu gốc, dựa trên khoảng cách được xác định trước.

* Đây là cách để lấy mẫu có hệ thống (Systematic Sampling)

**The trouble with systematic sampling (Vấn đề khi sử dụng lấy mẫu có hệ thống)**

**coffee\_ratings\_with\_id = coffee\_ratings.reset\_index()**

# Khi chúng ta gọi phương thức **reset\_index()** trên **"coffee\_ratings"**, nó sẽ tạo ra một bản sao của DataFrame hoặc Series đó với một chỉ số mới. Chỉ số mới này sẽ bắt đầu từ 0 và tăng dần cho mỗi dòng. Bản sao mới này, thường được lưu trong biến **coffee\_ratings\_with\_id**

**coffee\_ratings\_with\_id.plot(x="index",y="Aftertaste",kind="scatter")** # vẽ biểu đồ scatter plot với trục x là index, trục y là Aftertaste.

**plt.show()**

Lưu ý : Trong một lần lấy mẫu dữ liệu từ một tập hợp lớn, "systematic sampling" đề cập đến việc chọn các mẫu một cách đều đặn, có quy tắc từ tập dữ liệu đó. Tuy nhiên, nếu trong tập dữ liệu có một mẫu hay một nhóm mẫu tập trung ở một vị trí cụ thể hoặc theo một quy luật nhất định, việc này có thể làm cho systematic sampling trở nên không đáng tin cậy.

**Making systematic sampling safe (Cách để giải quyết )**

**shuffled = coffee\_ratings.sample(frac=1)** # lấy mẫu ngẫu nhiên từ bảng coffee\_ratings (frac=1 là lấy toàn bộ dữ liệu)

**shuffled = shuffled.reset\_index(drop=True).reset\_index()**

**# reset\_index(drop=True)**: Đặt lại chỉ số của "shuffled" để bắt đầu từ 0 và loại bỏ chỉ số cũ. Tham số **drop=True** chỉ định rằng chỉ số cũ sẽ không được thêm vào DataFrame mới.

# **reset\_index():** Tạo một cột mới có tên là "index" trong DataFrame, chứa giá trị của chỉ số mới. Điều này làm cho chỉ số cũ trở thành một cột dữ liệu mới trong DataFrame.

* Có thể hiểu đơn giản là bỏ đi index cũ và tạo ra một index mới chạy từ 0.

**shuffled.plot(x="index",y="Aftertaste",kind="scatter")** # vẽ biểu đồ scatter plot với trục x là index, trục y là aftertaste của bảng shuffled

**plt.show()**

* Kết hợp shuffling rows (xáo trộn các hàng) với systematic sampling (lấy mẫu theo quy tắc), kết quả cuối cùng sẽ tương đương với simple random sampling (lấy mẫu ngẫu nhiên đơn giản).

**top\_counts = coffee\_ratings['Country of Origin'].value\_counts()** # đếm số lượng mẫu của từng quốc gia trong cột Country of Origin

**top\_counts.head(6)** # in ra 6 quốc gia có số lượng mẫu nhiều nhất

**top\_counted\_countries = ["Taiwan", "Guatemala", "Colombia", "Honduras", "Thailand", "Ethiopia"]** # danh sách 6 quốc gia có số lượng mẫu nhiều nhất

**top\_counted\_subset = coffee\_ratings['Country of Origin’].isin(top\_counted \_countries)** # nó sẽ tạo ra một mảng boolean với True nếu giá trị của cột ‘Country of Origin’ có nằm trong ‘top\_counted\_countries’ và ngược lại là False

**coffee\_ratings\_top = coffee\_ratings[top\_counted\_subset]** # Dòng này sử dụng Series boolean được tạo ra ở bước trước để chọn các dòng trong DataFrame "coffee\_ratings" mà cột 'Country of Origin' của chúng nằm trong danh sách "top\_counted\_countries".

**coffee\_ratings\_samp = coffee\_ratings\_top.sample(frac=0.1, random\_state = 2021)** # lấy mẫu ngẫu nhiên 10% từ bảng coffee\_ratings\_top với random\_state=2021

**coffee\_ratings\_samp['Country of Origin'].value\_counts(normalize=True)** # đếm số lượng mẫu của từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_samp với normalize=True để tính tỉ lệ phần trăm

**Proportional stratified sampling**

**coffee\_ratings\_strat=coffee\_ratings\_top.groupby('Country of**

**Origin').sample(frac=0.1,random\_state=2021)**

# lấy mẫu ngẫu nhiên 10% từ mỗi nhóm quốc gia trong cột Country of Origin của bảng coffee\_ratings\_top với random\_state=2021

**coffee\_ratings\_strat['Country of Origin'].value\_counts(normalize=True)**

# đếm số lượng mẫu của từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_strat với normalize=True để tính tỉ lệ phần trăm

**Equal counts stratified sampling**

**coffee\_ratings\_eq = coffee\_ratings\_top.groupby('Country of Origin').sample(n=15, random\_state=2021)** # lấy mẫu ngẫu nhiên 5 mẫu từ mỗi nhóm quốc gia trong cột **Country of Origin của bảng coffee\_ratings\_top với random\_state=2021**

**coffee\_ratings\_eq['Country of Origin'].value\_counts(normalize=True)** # đếm số lượng mẫu của từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_eq với normalize=True để tính tỉ lệ phần tram

**Weighted random sampling**

**coffee\_ratings\_weight = coffee\_ratings\_top**

**condition = coffee\_ratings\_weight['Country of Origin'] == 'Taiwan'** # tạo điều kiện lọc dữ liệu cho quốc gia Taiwan

**coffee\_ratings\_weight['weight'] = np.where(condition,2,1)** # thêm cột weight vào bảng coffee\_ratings\_weight với giá trị 2 nếu dòng thỏa mãn điều kiện condition, ngược lại giá trị 1

**coffee\_ratings\_weight = coffee\_ratings\_weight.sample(frac=0.1,weights='weight')** # lấy mẫu ngẫu nhiên 10% từ bảng coffee\_ratings\_weight với trọng số là cột weight

***Kết quả :***

**coffee\_ratings\_weight['Country of Origin'].value\_counts(normalize=True)**

# đếm số lượng mẫu của từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_weight với normalize=True để tính tỉ lệ phần trăm

**varieties\_pop = list(coffee\_ratings['Variety'].unique())** # lấy danh sách các giá trị duy nhất trong cột Variety của bảng coffee\_ratings

**Stage 1: Sampling for subgroups**

**import random**

**varieties\_samp = random.sample(varieties\_pop,k=3)** # lấy ngẫu nhiên 3 giá trị từ danh sách varieties\_pop

**Stage 2: sampling each group**

**varieties\_condition = coffee\_ratings['Variety'].isin(varieties\_samp)** # tạo điều kiện lọc dữ liệu cho các giá trị trong danh sách varieties\_samp

**coffee\_ratings\_cluster = coffee\_ratings[varieties\_condition]** # lấy các dòng thỏa mãn điều kiện varieties\_condition

**coffee\_ratings\_cluster['Variety']=coffee\_ratings\_cluster['Variety'].cat.remove\_unused\_categories()** # xóa các giá trị không sử dụng trong cột Variety

**coffee\_ratings\_cluster.groupby('Variety').sample(n=5,random\_state=2021)**

# lấy mẫu ngẫu nhiên 5 mẫu từ mỗi nhóm giá trị trong cột Variety của bảng coffee\_ratings\_cluster với random\_state=2021

**top\_counted\_countries = ["Taiwan", "Guatemala", "Colombia", "Honduras", "Thailand", "Ethiopia"]** # danh sách 6 quốc gia có số lượng mẫu nhiều nhất

**subset\_condition = coffee\_ratings['Country of Origin'].isin(top\_counted\_countries)** # tạo điều kiện lọc dữ liệu cho các quốc gia trong danh sách top\_counted\_countries

**coffee\_ratings\_top = coffee\_ratings[subset\_condition]** # lấy các dòng thỏa mãn điều kiện subset\_condition

**coffee\_ratings\_top.shape**

**coffee\_ratings\_srs = coffee\_ratings\_top.sample(frac=1/3, random\_state=2021)** # lấy mẫu ngẫu nhiên 1/3 từ bảng coffee\_ratings\_top với random\_state=2021

**coffee\_ratings\_srs.shape** # in ra kích thước của bảng coffee\_ratings\_srs

**Review of stratified sampling**

**coffee\_ratings\_strat = coffee\_ratings\_top.groupby('Country of Origin').sample(frac=1/3,random\_state=2021)**

# lấy mẫu ngẫu nhiên 1/3 từ mỗi nhóm quốc gia trong cột Country of Origin của bảng coffee\_ratings\_top với random\_state=2021

**coffee\_ratings\_strat.shape** # in ra kích thước của bảng coffee\_ratings\_strat

**Review of cluster sampling**

**import random**

**top\_countries\_samp = random.sample(top\_counted\_countries,k=2)** # lấy ngẫu nhiên 2 quốc gia từ danh sách top\_counted\_countries

**top\_condition = coffee\_ratings\_top['Country of Origin'].isin(top\_countries\_samp)** # tạo điều kiện lọc dữ liệu cho các quốc gia trong danh sách top\_countries\_sample

**coffee\_ratings\_cluster = coffee\_ratings\_top[top\_condition]** # lấy các dòng thỏa mãn điều kiện top\_condition

**coffee\_ratings\_cluster['Country of Origin'] = coffee\_ratings\_cluster['Country of Origin'].astype('category').cat.remove\_unused\_categories()** # xóa các giá trị không sử dụng trong cột Country of Origin

**coffee\_ratings\_cluster.groupby('Country of Origin'). Sample (n=len(coffee\_ ratings\_ top) //6)** # lấy mẫu ngẫu nhiên 1/6 từ mỗi nhóm quốc gia trong cột Country of Origin của bảng coffee\_ratings\_cluster

**coffee\_ratings\_cluster.shape** # in ra kích thước của bảng coffee\_ratings\_cluster

**Calculating mean cup points**

* **Population**

**coffee\_ratings\_top['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points trong bảng coffee\_ratings\_top

* **Simple randam sample**

**coffee\_ratings\_srs['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points trong bảng coffee\_ratings\_srs

* **Stratified sample**

**coffee\_ratings\_strat['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points trong bảng coffee\_ratings\_strat

* **Cluster sample**

**coffee\_ratings\_cluster['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points trong bảng coffee\_ratings\_cluster

**Mean cup points by country : stratified**

**coffee\_ratings\_top.groupby('Country of Origin')['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points theo từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_top

**coffee\_ratings\_srs.groupby('Country of Origin')['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points theo từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_srs

**coffee\_ratings\_strat.groupby('Country of Origin')['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points theo từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_strat

**coffee\_ratings\_cluster.groupby('Country of Origin')['Total Cup Points'].mean()** # tính giá trị trung bình của cột Total Cup Points theo từng quốc gia trong cột Country of Origin của bảng coffee\_ratings\_cluster