Lab01

October 27, 2022

Họ tên:

MSSV:

Cách làm bài

Bạn sẽ làm trực tiếp trên file notebook này. Đầu tiên, bạn điền họ tên và MSSV vào phần đầu file ở bên trên. Trong file, bạn làm bài ở những chỗ có ghi là:

```
# YOUR CODE HERE
raise NotImplementedError()
```

hoặc đối với những phần code không bắt buộc thì là:

```
# YOUR CODE HERE (OPTION)
```

hoặc đối với markdown cell thì là:

```
YOUR ANSWER HERE
```

Tất nhiên, khi làm thì bạn xóa dòng `raise NotImplementedError()` đi. Đối những phần yêu cầu code thì thường ở ngay phía dưới sẽ có một (hoặc một số) cell chứa các bộ test để giúp bạn biết đã code đúng hay chưa; nếu chạy cell này không có lỗi gì thì có nghĩa là qua được các bộ test. Trong một số trường hợp, các bộ test có thể sẽ không đầy đủ; nghĩa là, nếu không qua được test thì là code sai, nhưng nếu qua được test thì chưa chắc đã đúng.

Trong khi làm bài, bạn có thể cho in ra màn hình, tạo thêm các cell để test. Nhưng khi nộp bài thì bạn xóa hoặc comment các câu lệnh in ra màn hình. Bạn lưu ý không được tự tiện xóa các cell hay sửa code của các cell có sẵn (trừ những chỗ được phép sửa như đã nói ở trên). Bên cạnh đó, các bạn có thể tự mình tạo thêm các hàm phụ trợ nhằm mục tiêu giải quyết bài toán

Trong khi làm bài, thường xuyên `Ctrl + S` để lưu lại bài làm của bạn, tránh mất mát thông tin.

*Nên nhớ mục tiêu chính ở đây là học, học một cách chân thật. Bạn có thể thảo luận ý tưởng với bạn khác cũng như tham khảo các nguồn trên mạng, nhưng sau cùng code và bài làm phải là của bạn, dựa trên sự hiểu thật sự của bạn. Khi tham khảo các nguồn trên mạng thì bạn cần ghi rõ nguồn trong bài làm. Bạn không được tham khảo bài làm của các bạn năm trước (vì nếu làm vậy thì bao giờ bạn mới có thể tự mình suy nghĩ để giải quyết vấn đề); sau khi kết thúc môn học, bạn cũng không được đưa bài làm cho các bạn khóa sau hoặc public bài làm trên Github (vì nếu làm vậy thì sẽ ảnh hưởng tới việc học của các bạn khóa sau).

Trong trường hợp bạn vi phạm những điều mình nói ở trên thì mình sẽ đề nghị giảng viên lý thuyết cho 0 điểm môn học.

Cách nộp bài

Khi chấm bài, đầu tiên mình sẽ chọn `Kernel - Restart & Run All`, để restart và chạy tất cả các cell trong notebook của bạn; do đó, trước khi nộp bài, bạn nên chạy thử `Kernel - Restart & Run All` để đảm bảo mọi chuyện diễn ra đúng như mong đợi.

Sau đó, bạn tạo thư mục nộp bài theo cấu trúc sau: - Thư mục `MSSV` (vd, nếu bạn có MSSV là 1234567 thì bạn đặt tên thư mục là `1234567`) - File `Lab01.ipynb` (không cần nộp các file khác)

Cuối cùng, bạn nén thư mục `MSSV` này lại và nộp ở link trên moodle. Đuôi của file nén phải là .zip (chứ không được .rar hay gì khác).

Bạn lưu ý tuân thủ chính xác qui định nộp bài ở trên.

Mục tiêu bài tập

Sinh viên có khả năng hiểu và cài đặt lại các hàm bổ trợ cho việc tiền xử lý dữ liệu

Mô tả yêu cầu

Sinh viên sẽ đọc vào một file dữ liệu có định dạng `.csv` (các giá trị thiếu sẽ được ký hiệu bằng chuỗi rỗng), sau đó sẽ thực hiện lần lượt các yêu cầu liên quan đến tiền xử lý dữ liệu

0.1 Import

[ ]:
```
import math
import csv
```

0.2 Thu thập dữ liệu

Các bạn sẽ tải file điểm thi THPT 2019 tại đây

0.3 Phần 1: Tiền xử lý dữ liệu (7.5đ)

Lưu ý:

Sinh viên không được sử dụng các hàm có sẵn của pandas hay numpy mà cần phải tự cài đặt lại

Mô tả dữ liệu theo thứ tự cột:

```
id :                int
Hoa :               float
Li :                float
Ma_mon_ngoai_ngu :  str
Ngoai_ngu :         float
Sinh :              float
Toan :              float
Van :               float
```

Tuy nhiên, đọc file bằng thư viện csv sẽ chuyển tất cả các giá trị về kiểu `str`, ta sẽ sẽ ép kiểu các giá trị số (trừ các giá trị rỗng) về theo kiểu dữ liệu phù hợp

Đọc file dữ liệu

[ ]:
```
def data_reader(path):
    csv_reader = csv.reader(open(path, 'r'), delimiter=',')
```

```
    # Skip header
    data = []
    header = csv_reader
    for i,row in enumerate(csv_reader):
        if i == 0:
            header = row
            continue
        convert_list = []
        for i, element in enumerate(row):
            if element == '':
                convert_list.append(str(element))
                continue
            if i == 0:
                convert_list.append(int(element))
            elif i == 3:
                convert_list.append(str(element))
            else:
                convert_list.append(float(element))
        data.append(convert_list)
    return header, data
```

[ ]:
```
header, data = data_reader('diemthi2019.csv')
```

[ ]:
```
print("header: ", header)
print("So dong: ", len(data))
```

[ ]:
```
data[23:33]
```

0.3.1 1) Liệt kê các cột thiếu dữ liệu (0.5đ)

Hàm `checkNaN_col()` sẽ trả về `list` chứa tên cột nếu như cột bị thiếu dữ liệu

[ ]:
```
def checkNaN_col(data, header):
    """Hàm này sẽ liệt kê các cột có chứa dữ liệu có dạng chuỗi rỗng

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị
        header <list>: list chứa tên của các cột trong file dữ liệu

    Returns:
        <list>: danh sách chứa tên các cột có chứa giá trị chuỗi rỗng
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
assert len(checkNaN_col(data, header)) == 7
```

Hàm `checkNaN_row()` sẽ trả về `list` chứa index của các dòng có tồn tại giá trị rỗng

[ ]:
```
def checkNaN_row(data):
    """Hàm này sẽ đếm số dòng có chứa dữ liệu có dạng chuỗi rỗng

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list>: danh sách chứa index của các dòng có tồn tại giá trị chuỗi rỗng
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
row_have_NaN = checkNaN_row(data)
assert len(row_have_NaN) == 17753
```

0.3.2 2) Điền giá trị thiếu (1.5đ)

Điền giá trị thiếu bằng phương pháp constant (điền giá trị 0), mean, median vào giá trị thiếu cho cột numeric, và mode cho cột categorical. Đối với các thuộc tính numeric, giá trị điền vào sẽ là giá trị `float`, đối với thuộc tính categorical, giá trị điền vào sẽ là giá trị `str`

Lưu ý: khi tính mean, median hay mode các bạn cần bỏ qua giá trị bị thiếu. Khi thao tác trên các cột dạng float, kết quả phải được làm tròn ở hai chữ số bằng hàm `round()`

Hàm tính mean/median/mode của các cột tương ứng trong dữ liệu sẽ trả về một danh sách, chứa các giá trị mean/median/mode của các cột, và sẽ được hàm `fill_data()` sử dụng danh sách đó để điền vào giá trị thiếu

[ ]:
```
def sum_data(data):
    """Hàm này dùng để tính tổng theo các cột có dạng numeric

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list>, list chứa các giá trị tổng theo cột
        <list>: list chứa số lượng giá trị khác giá trị chuỗi rỗng
    """
    # Hàm này sẽ được dùng bổ trợ để tính sum và số lượng mẫu không bị thiếu␣
↪của từng cột thuộc tính (trừ cột id)
    # YOUR CODE HERE (OPTION)
    pass
```

[ ]:
```python
def mean(data):
    """Hàm này dùng để tính trung bình theo các cột có dạng numeric

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
 ↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list>: list chứa các giá trị trung bình theo từng cột
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```python
assert mean(data) == [5.34, 5.56, 4.76, 4.67, 6.8, 5.62]
```

[ ]:
```python
def median(data):
    """Hàm này dùng để tính trung bình theo các cột có dạng numeric

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
 ↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list>: list chứa các giá trị trung vị theo từng cột
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```python
assert median(data) == [5.5, 5.75, 4.4, 4.5, 7.0, 5.75]
```

[ ]:
```python
def find_distinct_value(data):
    """Hàm này sẽ có nhiệm vụ tìm và trả về danh sách chứa các giá trị có thể␣
 ↪có của cột 'Ma_mon_ngoai_ngu'

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
 ↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list>: list đã được sắp xếp, chứa các giá trị chuỗi có thể có của cột␣
 ↪'Ma_mon_ngoai_ngu'
    """
    # # Chỉ cần quan tâm đến cột 'Ma_mon_ngoai_ngu' và tìm ra các giá trị có␣
 ↪thể có
    # YOUR CODE HERE
    raise NotImplementedError()
```

Thực hiện đếm số lần xuất hiện của các giá trị đã được tìm và chọn ra giá trị xuất hiện nhiều lần nhất, nếu có nhiều giá trị cùng xuất hiện nhiều lần, thì chọn giá trị nhỏ hơn theo bảng chữ cái

[ ]:
```
def mode(data):
    """Hàm này sẽ có mục tiêu tìm ra giá trị chuỗi xuất hiện nhiều lần nhất. Để␣
↪đơn giản, sinh viên chỉ cần quan đến cột 'Ma_mon_ngoai_ngu'

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        str: giá trị chuỗi xuất hiện nhiều lần nhất trong cột␣
↪'Ma_mon_ngoai_ngu', nếu có nhiều hơn một kết quả, chọn chuỗi nhỏ hơn theo␣
↪bảng chữ cái
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

Hàm `fill_data` sẽ cho phép tùy chọn điền vào giá trị thiếu của cột dạng numeric bằng giá trị 0 hoặc `mean` hoặc `median`, còn cột có dạng categorical sẽ mặc định được fill bằng mode

[ ]:
```
def fill_data(data, type_on_numeric = 'mean'):
    """Hàm này có nhiệm vụ điền dữ liệu bị thiếu

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị
        type_on_numeric (str, optional): phương pháp điền dữ liệu cho cột␣
↪numeric (const/mean/median)

    Returns:
        <list<list>>: dữ liệu sau khi đã được điền vào các giá trị thiếu, dữ␣
↪liệu là list của các dòng, mỗi dòng là một list chứa các giá trị
    """
    if type_on_numeric == 'const':
        # YOUR CODE HERE
        raise NotImplementedError()
    elif type_on_numeric == 'mean':
        # YOUR CODE HERE
        raise NotImplementedError()
    elif type_on_numeric == 'median':
        # YOUR CODE HERE
        raise NotImplementedError()
```

[ ]:
```
data_filled_const = fill_data(data, type_on_numeric = 'const')
data_filled_mean = fill_data(data, type_on_numeric = 'mean')
```

```
data_filled_median = fill_data(data, type_on_numeric = 'median')
```

[ ]:
```
assert data_filled_const[20] == [20, 3.0, 0, 'N1', 0, 5.0, 4.8, 0]
assert data_filled_const[25] == [25, 3.75, 3.5, 'N1', 0, 0, 6.8, 0]
```

[ ]:
```
assert data_filled_mean[20] == [20, 3.0, 5.56, 'N1', 4.76, 5.0, 4.8, 5.62]
assert data_filled_mean[25] == [25, 3.75, 3.5, 'N1', 4.76, 4.67, 6.8, 5.62]
```

[ ]:
```
assert data_filled_median[20] == [20, 3.0, 5.75, 'N1', 4.4, 5.0, 4.8, 5.75]
assert data_filled_median[25] == [25, 3.75, 3.5, 'N1', 4.4, 4.5, 6.8, 5.75]
```

0.3.3 3) Xóa các dòng và cột mang số lượng của giá trị thiếu lớn hơn một ngưỡng cho trước (0.5đ)

Hàm `filter_missing_row()` sẽ nhận vào dữ liệu và tỉ lệ ngưỡng `ratio` giữa số lượng dữ liệu thiếu và tổng số lượng thuộc tính. Lấy ví dụ, nếu `ratio=0.5` thì các hàng có số lượng giá trị thiếu lớn hơn 50% sẽ bị loại bỏ khỏi tập dữ liệu

[ ]:
```
def filter_missing_row(data, ratio = 0.5):
    """Hàm này có nhiệm vụ xóa đi các dòng có tỉ lệ dữ liệu bị thiếu LỚN HƠN␣
↪HOẶC BẰNG một ngưỡng cho trước

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị
        ratio (float, optional): Tỉ lệ ngưỡng mà dữ liệu cho phép thiếu

    Returns:
        <list<list>>: dữ liệu sau khi đã được xóa bớt các dòng thiếu lượng lớn␣
↪dữ liệu, dữ liệu là list của các dòng, mỗi dòng là một list chứa các giá trị
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
assert len(filter_missing_row(data, 0.3)) == 285581
```

Hàm `filter_missing_col()` sẽ nhận vào dữ liệu và tỉ lệ ngưỡng `ratio` giữa số lượng dữ liệu thiếu và tổng số lượng mẫu. Lấy ví dụ, nếu `ratio=0.5` thì các thuộc tính (các cột) có số lượng giá trị thiếu lớn hơn 50% so với số lượng mẫu của tập dữ liệu sẽ bị loại bỏ

[ ]:
```
def filter_missing_col(data, header, ratio = 0.5):
    """Hàm này có nhiệm vụ xóa đi các cột có tỉ lệ dữ liệu bị thiếu LỚN HƠN␣
↪HOẶC BẰNG một ngưỡng cho trước

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị
```

```
        ratio (float, optional): Tỉ lệ ngưỡng mà dữ liệu cho phép thiếu

    Returns:
        <list>: header đã được cập nhật thông qua việc loại bỏ đi bớt các cột␣
↪không đủ điều kiện (nếu có loại bỏ)
        <list<list>>: dữ liệu sau khi đã được xóa bớt các cột thiếu lượng lớn␣
↪dữ liệu, dữ liệu là list của các dòng, mỗi dòng là một list chứa các giá trị
    """

    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
new_header, new_data = filter_missing_col(data, header, 0.05)
```

[ ]:
```
assert new_header == ['id', 'Hoa', 'Li', 'Sinh', 'Toan', 'Van']
# Không tính cột id
assert len(new_data[0]) - 1 == 5
```

0.3.4 4) Xóa các mẫu bị trùng lặp (0.5đ)

Kiểm tra, xác định các dòng bị trùng dữ liệu và xóa các dòng đó.

Với mỗi dòng, so sánh với các dòng dữ liệu bên dưới để kiểm tra sự trùng lặp, các dòng sẽ được xem là trùng lặp nếu tất cả các thuộc tính giống nhau.

Gợi ý: với mỗi dòng dữ liệu tổng hợp lại thành một chuỗi `str` nào đó rồi đưa vào `set`, sau đó dùng `set` để kiểm tra xem dữ liệu đã xuất hiện chưa.

[ ]:
```
def remove_duplicate(data):
    """Hàm này sẽ thực hiện loại bỏ các dòng bị trùng lặp VỀ MẶT DỮ LIỆU (không␣
↪xét cột id)

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list<list>>: dữ liệu sau khi đã được xóa bớt các dòng trùng lặp, dữ␣
↪liệu là list của các dòng, mỗi dòng là một list chứa các giá trị
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
assert len(remove_duplicate(data)) == 292762
```

0.3.5 5) Loại bỏ bớt giá trị cực hạn của các cột numeric (1đ)

Sinh viên sẽ sử dụng luật 2 sigma (95% dữ liệu sẽ thuộc đoạn $[\mu \pm 2 * \sigma]$) và 3 sigma (99.7% dữ liệu sẽ thuộc đoạn $[\mu \pm 3 * \sigma]$) để loại bỏ các mẫu có giá trị mang giá trị không bình thường.

Hàm `std()` sẽ trả về danh sách chứa các giá trị độ lệch chuẩn của các cột có dạng numeric

Hàm `remove_outlier()` sẽ loại bỏ các nhiều bằng các sử dụng lại kết quả của hàm `std()` và hàm `mean()`

Lưu ý: Các số float cần phải được làm tròn về 2 chữ số bằng hàm `round()`

[ ]:
```
def std(data):
    """Hàm này dùng để tính độ lệch chuẩn theo các cột có dạng numeric

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list>: list chứa các giá trị độ lệch chuẩn theo từng cột
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
assert std(data) == [1.59, 1.62, 1.78, 1.25, 1.39, 1.11]
```

Hàm `remove_outlier()` sẽ nhận vào `data` và số `k`-sigma mà chúng ta muốn chọn ra từ dữ liệu, dòng dữ liệu sẽ được giữ lại nếu:

$$(\mu_i - k * \sigma_i) \leq value_i \leq (\mu + k * \sigma_i)$$

Với $\sigma_i, value_i$ lần lượt là giá trị của cột i và độ lệch chuẩn của cột i

[ ]:
```
from tqdm import tqdm

def remove_outlier(data, k):
    """Hàm này sẽ loại bỏ các dòng chứa ít nhất một giá trị không thuộc đoạn␣
↪cho phép ứng với mean và std của từng cột

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị
        k <float>: số lượng sigma được dùng để tính khoảng

    Returns:
        <list<list>>: dữ liệu sau khi đã được xóa bớt các dòng có chứa giá trị␣
↪không thuộc đoạn dữ liệu cho phép, dữ liệu là list của các dòng, mỗi dòng là␣
↪một list chứa các giá trị
    """
```

```
    # YOUR CODE HERE
    raise NotImplementedError()
```

Do hàm này có khá nhiều phép so sánh nên mình sẽ chọn ra 1000 mẫu đầu tiên của dữ liệu để kiểm tra hàm

[ ]:
```
sampled_data = data[0:1000]

assert len(remove_outlier(sampled_data, k = 2)) == 810
```

0.3.6 6) Tính điểm tổ hợp môn (1.5đ)

Sinh viên cần tạo thêm các cột mới chứa điểm tổ hợp các môn theo thứ tự lần lượt [`A`, `A1`, `A2`, `B`, `D`]. Cụ thể, sinh viên cần tính điểm tổ hợp khối `A`, khối `A1`, khối `A2`, khối `B` và khối `D` theo cách tính.

Cột `A` = `Toan` + `Li` + `Hoa`

Cột `A1` = `Toan` + `Li` + `Ngoai_ngu`

Cột `A2` = `Toan` + `Hoa` + `Ngoai_ngu`

Cột `B` = `Toan` + `Hoa` + `Sinh`

Cột `D` = `Toan` + `Van` + `Ngoai_ngu`

Lưu ý: khối `A1`, khối `A2` chỉ được tính khi `Ma_mon_ngoai_ngu = N1`, nếu `Ma_mon_ngoai_ngu != N1`, thì cột `A1` và `A2` sẽ có giá trị 0

Xét ví dụ:

- Cho header

['id', 'Hoa', 'Li', 'Ma_mon_ngoai_ngu', 'Ngoai_ngu', 'Sinh', 'Toan', 'Van']

- Xét mẫu dữ liệu

[0, 4.5, 8.25, 'N1', 8.0, 6.0, 8.6, 6.17]

- Header sẽ được xây dựng thành

['id', 'Hoa', 'Li', 'Ma_mon_ngoai_ngu', 'Ngoai_ngu', 'Sinh', 'Toan', 'Van', `'A'`, `'A1'`, `'A2'`, `'B'`, `'D'`]

- Dữ liệu xây dựng thành

[0, 4.5, 8.25, N1, 8.0, 6.0, 8.6, 6.17, `A`, `A1`, `A2`, `B`, `D`]

- Và bằng

[0, 4.5, 8.25, N1, 8.0, 6.0, 8.6, 6.17, `21.35`, `24.85`, `21.1`, `19.1`, `22.77`]

Sinh viên sử dụng dữ liệu đã được điền đầy đủ giá trị bằng giá trị 0 (data_filled_const) để tính tổng, vì thí sinh không dự thi đồng nghĩa với điểm bằng 0 khi xét tổ hợp môn. Điểm số sẽ được làm tròn tối đa 2 chữ số thập phân bằng hàm `round()`

```
[ ]: # Sinh viên uncomment đoạn code này để đọc vào dữ liệu đã được điền thiếu có␣
      ↪sẵn nếu không làm được câu 2
     # header, data_filled_const = data_reader('data_filled_const.csv')
     # assert data_filled_const[20] == [20, 3.0, 0, 'N1', 0, 5.0, 4.8, 0]
```

```
[ ]: def calculate_combination(data, header):
         """Hàm này sẽ có nhiệm vụ xây dựng nên các thuộc tính mới cho dữ liệu, cụ␣
      ↪thể là tính điểm tổ hợp các khối liên quan

         Args:
             data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
      ↪vào, mỗi dòng sẽ là một list các giá trị
             header <list>: list chứa tên của các cột thuộc tính


         Returns:
             <list>: header đã được cập nhật thêm tên của các cột thuộc tính mới
             <list<list>>: dữ liệu sau khi đã được xóa bớt các dòng có chứa giá trị␣
      ↪không thuộc đoạn dữ liệu cho phép, dữ liệu là list của các dòng, mỗi dòng là␣
      ↪một list chứa các giá trị
         """
         # YOUR CODE HERE
         raise NotImplementedError()
```

```
[ ]: new_header, new_data = calculate_combination(data_filled_const, header)
```

```
[ ]: assert new_header == ['id', 'Hoa', 'Li', 'Ma_mon_ngoai_ngu', 'Ngoai_ngu',␣
      ↪'Sinh', 'Toan', 'Van', 'A', 'A1', 'A2', 'B', 'D']
     assert new_data[0] == [0, 4.5, 8.25, 'N1', 8.0, 6.0, 8.6, 6.17, 21.35, 24.85,␣
      ↪21.1, 19.1, 22.77]
```

0.3.7 7) Chuẩn hóa dữ liệu (1đ)

Sinh viên cần chuẩn hóa các cột dạng numeric theo phương pháp min-max và Z-score

Sử dụng kết quả dữ liệu tìm được ở câu 2 để làm input cho hàm chuẩn hóa

```
[ ]: # Sinh viên uncomment đoạn code này để đọc vào dữ liệu đã được điền thiếu có␣
      ↪sẵn nếu không làm được câu 2
     # header, data_filled_const = data_reader('data_filled_const.csv')
     # assert data_filled_const[20] == [20, 3.0, 0, 'N1', 0, 5.0, 4.8, 0]
```

```
[ ]: def min_max_value(data):
         """Hàm này sẽ có nhiệm vụ tìm các giá trị min và max theo từng cột thuộc␣
      ↪tính của dữ liệu đầu vào

         Args:
```

```
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list>: list chứa các giá trị min ứng với từng cột thuộc tính có dạng␣
↪numeric
        <list>: list chứa các giá trị max ứng với từng cột thuộc tính có dạng␣
↪numeric
    """
    # YOUR CODE HERE (OPTIONAL)
    pass
```

[ ]:
```
def min_max_scaler(data):
    """Hàm này sẽ có nhiệm vụ chuẩn hóa dữ liệu theo phương pháp min-max

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list<list>>: dữ liệu sau khi đã được chuẩn hóa theo phương pháp␣
↪min-max, dữ liệu là list của các dòng, mỗi dòng là một list chứa các giá trị
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
def standard_scaler(data):
    """Hàm này sẽ có nhiệm vụ chuẩn hóa dữ liệu theo phương pháp z-score. Sử␣
↪dụng lại hàm mean() và std() nếu có thể.

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị

    Returns:
        <list<list>>: dữ liệu sau khi đã được chuẩn hóa theo phương pháp␣
↪z-score, dữ liệu là list của các dòng, mỗi dòng là một list chứa các giá trị
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```

[ ]:
```
min_max_normalized_data = min_max_scaler(data_filled_const)
standard_normalized_data = standard_scaler(data_filled_const)
```

[ ]:
```
assert min_max_normalized_data[0] == [0, -0.5, 1.58, 'N1', 1.72, 1.02, 1.29, 0.
↪5]
```

```
assert standard_normalized_data[0] == [0, 0.45, 0.82, 'N1', 0.8, 0.6, 0.86, 0.
↪65]
```

0.3.8 8) Mã hóa one-hot (1đ)

Biến đổi cột `Ma_mon_ngoai_ngu` (cột dạng categorical) theo phương pháp mã hóa one-hot

Xét ví dụ:

- Cho header

['id', 'Hoa', 'Li', 'Ma_mon_ngoai_ngu', 'Ngoai_ngu', 'Sinh', 'Toan', 'Van']

- Xét mẫu dữ liệu

[0, 4.5, 8.25, 'N1', 8.0, 6.0, 8.6, 6.17]

- Header được xây dựng thành

['id', 'Hoa', 'Li', 'Ngoai_ngu', 'Sinh', 'Toan', 'Van', `'N1'`, `'N2'`, `'N3'`, `'N4'`, `'N5'`, `'N6'`]

- Và dữ liệu bằng

[0, 4.5, 8.25, 8.0, 6.0, 8.6, 6.17, `1`, `0`, `0`, `0`, `0`, `0`]

Lưu ý: Sinh viên cần tự cài đặt hàm tìm các giá trị có thể có của một cột dạng categorical và lưu các giá trị của cột đó bằng một danh sách đã được sắp xếp tăng dần theo bảng chữ cái (nếu không thực hiện được câu 2)

Sử dụng kết quả dữ liệu tìm được ở câu 2 để làm input

[ ]:
```
# Sinh viên uncomment đoạn code này để đọc vào dữ liệu đã được điền thiếu có␣
↪sẵn nếu không làm được câu 2
# header, data_filled_const = data_reader('data_filled_const.csv')
# assert data_filled_const[20] == [20, 3.0, 0, 'N1', 0, 5.0, 4.8, 0]
```

[ ]:
```
def OneHotEncoder(data, header):
    """Hàm này sẽ có nhiệm mã hóa one-hot dữ liệu trên cột 'Ma_mon_ngoai_ngu'

    Args:
        data <list<list>>: list của các dòng dữ liệu trong file dữ liệu đầu␣
↪vào, mỗi dòng sẽ là một list các giá trị
        header <list>: list chứa tên của các cột thuộc tính


    Returns:
        <list>: header đã được cập nhật thêm tên của các cột thuộc tính mới và␣
↪đã loại bỏ đi cột 'Ma_mon_ngoai_ngu'
        <list<list>>: dữ liệu sau khi đã được mã hóa one-hot, dữ liệu là list␣
↪của các dòng, mỗi dòng là một list chứa các giá trị
    """
    # YOUR CODE HERE
```

```
    raise NotImplementedError()
```

[ ]:
```
new_header, new_data = OneHotEncoder(data_filled_const, header)
```

[ ]:
```
assert new_data[0] == [0, 4.5, 8.25, 8.0, 6.0, 8.6, 6.17, 1, 0, 0, 0, 0, 0]
```

0.4 Phần 2: Làm quen với numpy và pandas (2.5đ)

Đa số các yêu cầu trên đề được hỗ trợ bởi pandas và numpy.

Sang đến phần này yêu cầu sinh viên tính độ tương quan giữa các cặp thuộc tính dạng numeric.

Phần này cho phép sinh viên sử làm quen và sử dụng pandas để tính toán, và trực quan, tuy nhiên phần nào mình yêu cầu cài đặt thì sinh viên không được sử dụng thư viện có sẵn.

Phần này sinh viên có thể tạo thêm cell nếu cần thiết

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

[ ]:
```
df = pd.read_csv('data_filled_const.csv', index_col = 'id')
df
```

0.4.1 9) Tính độ tương quan giữa các cột thuộc tính (1đ)

Sinh viên thực hiện tính độ tương quan (correlation) giữa các cặp thuộc tính có dạng numeric, và trực quan hóa theo biểu đồ heat map

Tham khảo:

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

https://seaborn.pydata.org/generated/seaborn.heatmap.html

Sau đó sử dụng thông tin về độ tương quan để kiểm tra giả thuyết :

`Thí sinh học toán tốt sẽ có xu hướng học lý hoặc học hóa tốt`

[ ]:

[ ]:

0.4.2 10) Trực quan hóa (1.5đ)

Thực hiện chọn và trực quan hóa phân phối điểm của các môn có trong dữ liệu học và nêu nhận xét về phân bố điểm thi.

Riêng đối với môn ngoại ngữ, sinh viên chọn riêng ra theo từng `Ma_mon_ngoai_ngu` và trực quan hóa theo từng mã môn ngoại ngữ riêng (N1, N2, ... N6) và nêu nhận xét.

[ ]:

[ ]:

[ ]:

[ ]:

1 Tài liệu tham khảo cho sinh viên

1. Slide lý thuyết
2. Textbook: J. Han and M. Kamber: Data Mining, Concepts and Techniques, Second Edition - Chapter 2: Getting to Know Your Data & Chapter 3: Data Preprocessing

Mọi thắc mắc các bạn gửi mail cho người hướng dẫn thực hành phụ trách: **Kiều Vũ Minh Đức** (kvmduc3@gmail.com).

[ ]: