VIETNAM NATIONAL UNIVERSITY HCMC
UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

———————o0o———————

Project 3: Linear Regression

Subject: Applied Mathematics and Statistics

Lecturer: **Mr. Vũ Quốc Hoàng**
Mr. Nguyễn Văn Quang Huy
Mr. Nguyễn Ngọc Toàn
Ms. Phan Thị Phương Uyên

Class: **22CLC05**

Student: **22127119 – Hồ Phước Hoàn**

HCMC, 8/2024

Table of Contents

I. Thư viện sử dụng

Source code được viết bằng ngôn ngữ Python, thực hiện trên môi trường Jupyter Notebook. Trong Source code có sử dụng các thư viện cần thiết bên ngoài như:

- Numpy: thư viện dùng để tính toán trong thuật toán
- Matplotlib: thư viện dùng để hiển thị ảnh
- Pandas: thư viện giúp xử lí dữ liệu dạng bảng phân cách bằng dấu phẩy (csv: comma-separated values) và phân tích dữ liệu [2]
- Seaborn: thư viện trực quan hóa dữ liệu dựa trên matplotlib, cung cấp giao diện bậc cao trong việc vẽ đồ họa thống kê một cách mạnh mẽ [1]

II. Thực hiện

1) Mô tả hàm

a) Class $\underline{OLSLinearRegression()}$

Lý do sử dụng class này là để gom nhóm những chức năng phù hợp với mỗi mô hình hồi quy, mỗi khi gọi class này là một mô hình hồi quy khác nhau.

- Hàm $\underline{fit(self, X, y)}$

- Tham số đầu vào:
  + X: dữ liệu đầu vào trong mô hình hồi quy, (dữ liệu train) (np.array)
  + y: dữ liệu đầu ra trong mô hình hồi quy, (dữ liệu train) (np.array)
- Kết quả trả về: đối tượng của class $OLSLinearRegression$ (object)
- Mô tả:
  Hàm này dùng để khớp mô hình vào dữ liệu cho trước. Nó sử dụng phương pháp Ordinary Least Square để tìm trọng số tối ưu cho mô hình.
  Trong hàm này, ta tính vector trọng số w bằng cách áp dụng công thức: [3]

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

  Với $(X^T X)^{-1} X^T$: đây là giả nghịch đảo (pseudo inverse), để áp dụng công thức này ta dùng hàm *np.linalg.pinv(X)* để tính rồi nhân ma trận với y.

- Hàm $\underline{getWeights(self)}$

- Tham số đầu vào: **không có**
- Kết quả trả về: vector trọng số tối ưu (theo dạng cột) (np.array)
- Mô tả:
  Trả về vector trọng số tối ưu cho mô hình

- Hàm $\underline{predict(self, X)}$

- Tham số đầu vào:
  X: dữ liệu đầu vào trong mô hình hồi quy, (dữ liệu test) (np.array)
- Kết quả trả về: $y_{predict}$ là dữ liệu đầu ra dự đoán từ mô hình (np.array)
- Mô tả:
  Ta nhân 2 ma trận $X,\ w$ để lấy $y_{predict}$. Vì khi đó ta có X là dữ liệu đã có sẵn và w là

trọng số tối ưu vừa tìm được nên ta đã có sẵn các dữ liệu cho công thức hồi quy dạng như sau: [4]

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix},$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}.$$

Trong công thức ε_i là độ lỗi của từng đặc trưng (feature) trong mô hình so với tập test.

Trong hình bên dưới, ta thấy x_i^T sẽ tương ứng X mà ta đã có, ngoài ra β tương ứng với w, là trọng số tối ưu ta đã tính trước đó. Do đó, khi ta nhân 2 ma trận X, w, ta sẽ được kết quả $y_{predict}$ đúng theo công thức.

b) Hàm $\underline{MAE(y, y\ pred)}$

- Tham số đầu vào:
  + y: dữ liệu đầu ra (dữ liệu test) (np.array)
  + y_{pred}: dữ liệu đầu ra dự đoán khi dùng mô hình hồi quy với tập X_{test} (np.array)

- Kết quả trả về: trung bình độ lỗi tuyệt đối (Mean Absolute Error)
- Mô tả:
  Ta áp dụng theo công thức: [5]

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

  Trong đó y_i hoặc x_i có thể là y_{pred} hoặc y đều được. Công thức tính độ lỗi này giúp ta xác định độ chính xác của mô hình mà ta vừa train.
  Ở đây ta dùng trị tuyệt đối để không xuất hiện số âm để tránh xuất hiện kết quả không mong muốn. Ngoài ra, sử dụng attribute $ravel()$ để trải phẳng ma trận ra thành shape $[1, n]$ để đưa 2 ma trận trên đều về cùng shape 1 chiều và các giá trị của mỗi ma trận vẫn tương ứng với nhau đôi một. Điều này giúp việc tính toán broadcasting của numpy nhanh hơn và dễ kiểm soát hơn.

c) Hàm $\underline{linear\ regression\ model(models, y\ train\ shuffle, k = 5)}$

- Tham số đầu vào:
  + $models$: mảng các model mà ta muốn tính toán, trong mảng sẽ là các np.array lưu trữ các đặc trưng khác nhau của mỗi mô hình. Các np.array này đã được dùng np.hstack để nối cột 1 vào mảng các đặc trưng và được shuffle trước đó, có thể nói $models$ là mảng các $X_{train\_preprocess}$ (array)
  + $y_{train\_shuffle}$: dữ liệu đầu ra của tập train đã được shuffle (np.array)
  + k: số lượng fold muốn tách trong phương pháp k-fold cross validation (int). Mặc định là 5
- Kết quả trả về: mảng các số MAE của từng mô hình (array)
- Mô tả:
  Hàm trả về mảng các số MAE của từng mô hình, MAE ở đây chỉ là các phép so sánh trên tập dữ liệu các fold đã chia chứ không phải trên tập test thực tế.

  Đầu tiên ta lấy ra tập y_{fold} bằng cách tách mảng $y_{train\_shuffle}$ theo chiều dọc (cắt ngang). Vì mảng $y_{train\_shuffle}$ đang là kiểu dữ liệu Series của pandas nên ta cần chuyển kiểu dữ liệu sang np.array bằng hàm $to\_numpy()$ của pandas, vì hàm này trải phẳng mảng $y_{train\_shuffle}$ nên ta cần chuyển nó sang dạng vector cột bằng hàm $reshape(-1, 1)$ của numpy rồi cuối cùng mới tách mảng ra k fold khác nhau.

  $MAE_{features}$ là mảng dùng để lưu các số MAE của các mô hình sau khi đã chạy xong. Vào vòng lặp, ta thực hiện duyệt qua từng mô hình để tính cho từng mô hình riêng lẻ. Ta sẽ tạo mảng X_{fold} là mảng chứa k fold tương ứng với y_{fold}, và dùng 2 mảng lưu các fold này để vừa huấn luyện mô hình, vừa cho mô hình thi. Bên trong, ta sẽ duyệt k lần vì có k fold các tập dữ liệu train và test. Cứ mỗi lần duyệt tới fold nào thì fold đó sẽ là $fold_{test}$, còn $k - fold_{test}$ sẽ là các fold dùng cho việc huấn luyện trong vòng lặp đó. Với cách huấn luyện này thì sẽ tạo ra sự khách quan trong quá trình đánh giá mô hình, ngoài ra, việc thực hiện k fold cross validation mang lại hiệu

quả tốt cho việc xây dựng mô hình trong trường hợp dữ liệu không được dồi dào cho lắm [6]. Trong code, em thực hiện gán các tập test vào X_{test_i}, Y_{test_i} trước, sau đó pop tập dữ liệu vừa gán khỏi tập fold và lưu nó trong 2 biến tạm X_{temp_i}, Y_{temp_i} nhằm insert nó vào tập fold lại để cho các vòng lặp sau đó dùng. Sau đó em sẽ ghép các fold còn lại trong tập fold để thành 1 mảng hoàn chỉnh để làm tập train, rồi em lại insert tập bị pop vào index đã pop đi hồi nãy của tập fold.

Cuối cùng là em sẽ train mô hình với tập train X_{train_i}, Y_{train_i} bằng cách dùng hàm $fit()$ trong $OLSLinearRegression$, rồi tính ra dữ liệu dự đoán khi dùng tập X_{test_i} và test với tập y_{test_i} rồi gán vào mảng $MAE_{feature}$ là mảng lưu các giá trị MAE của mỗi lần tính trong k_{fold} để khi đã kết thúc việc train mô hình đó xong thì sẽ lấy trung bình cộng của các MAE đã tính đó và gán vào $MAE\_features$ là mảng sẽ xuất khỏi hàm.

2) Triển khai và đánh giá

2.1. Yêu cầu 1

- Trước khi thực hiện bất kì bài toán về machine learning nào, hay bài toán nào có dữ liệu thì ta cũng cần phải phân tích khám phá dữ liệu (Exploratory Data Analysis). Mục đích là để giúp chúng ta có cái nhìn đầu tiên về dữ liệu. Chúng ta cần có một cảm giác nhất định về những gì mình có trong tay trước khi có những chiến lược xây dựng mô hình. EDA giúp ta mường tượng được độ phức tạp của bài toán và vạch ra những bước đầu tiên cần làm [7].
- Ta sẽ cần phải khám phá các đặc trưng sau đây [7]:
  + Kích thước dữ liệu
  + Ý nghĩa từng trường dữ liệu
  + Kiểu dữ liệu của mỗi trường
  + Phân phối xác suất của từng trường
  + Mối tương quan giữa các trường dữ liệu

  Có được những điều này, ta sẽ dường như nắm chắc được ta cần làm gì trong bài toán để huấn luyện ra mô hình mong muốn.

  Với những mục đích trên, em đã thực hiện khám phá mô hình như sau.

- Trước tiên, em đã dùng hàm $describe()$ của pandas để xem các thông tin thống kê trong tập dữ liệu train đã cho. Hàm này giúp hiển thị các trường thông tin cơ bản trong từng đặc trưng của tập dữ liệu như:

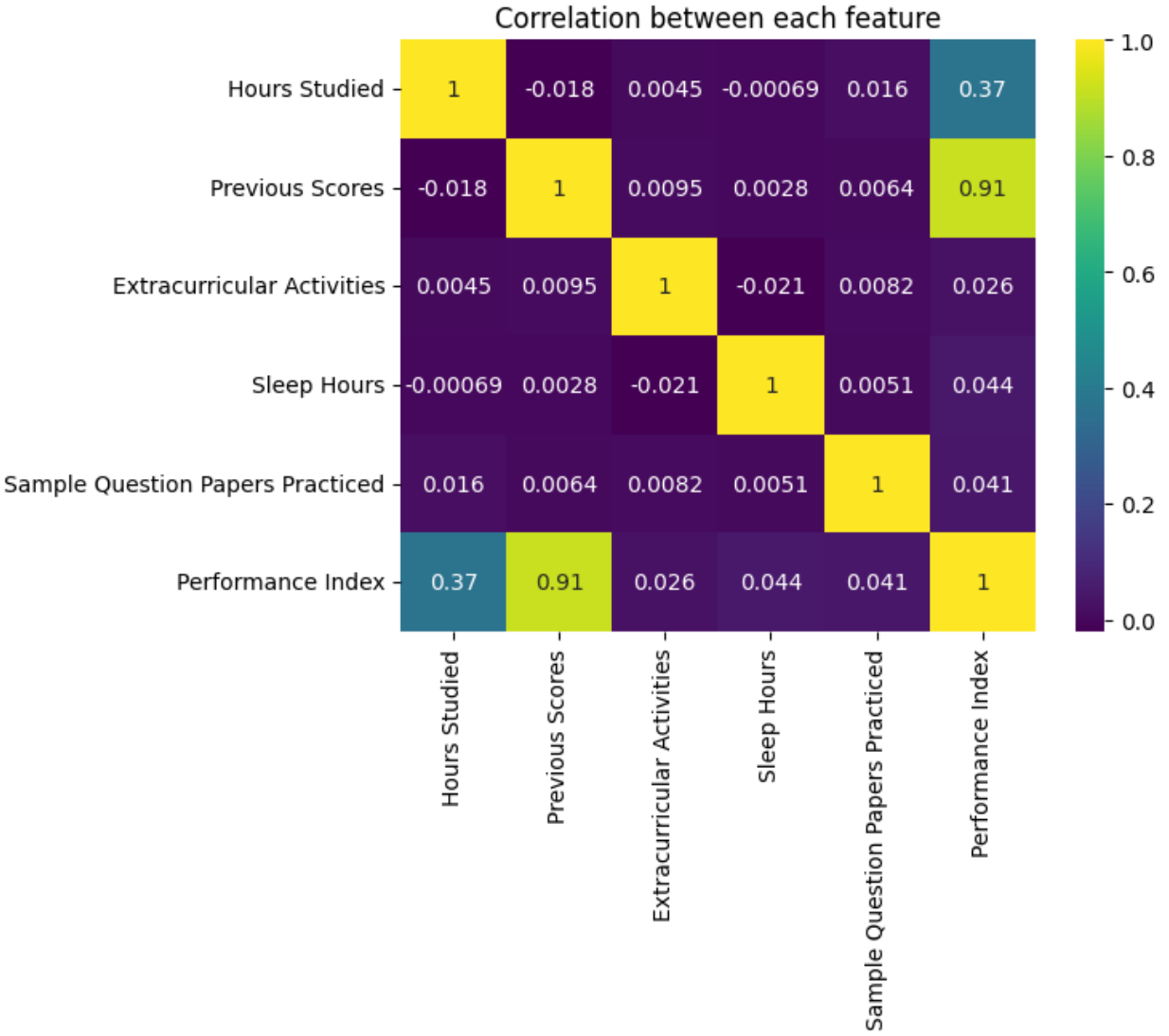
```
**** Description about each feature****
       Hours Studied  Previous Scores  Extracurricular Activities  \
count    9000.000000      9000.000000                 9000.000000
mean        4.976444        69.396111                    0.493667
std         2.594647        17.369957                    0.499988
min         1.000000        40.000000                    0.000000
25%         3.000000        54.000000                    0.000000
50%         5.000000        69.000000                    0.000000
75%         7.000000        85.000000                    1.000000
max         9.000000        99.000000                    1.000000

       Sleep Hours  Sample Question Papers Practiced  Performance Index
count  9000.000000                       9000.000000        9000.000000
mean      6.535556                          4.590889          55.136333
std       1.695533                          2.864570          19.187669
min       4.000000                          0.000000          10.000000
25%       5.000000                          2.000000          40.000000
50%       7.000000                          5.000000          55.000000
75%       8.000000                          7.000000          70.000000
max       9.000000                          9.000000         100.000000
```

  + count (số dòng xuất hiện của đặc trưng này)
  + mean (giá trị trung bình trong đặc trưng)
  + std (độ lệch chuẩn)
  + min (giá trị nhỏ nhất trong đặc trưng)
  + max (giá trị lớn nhất trong đặc trưng),
  + 25%: tứ phân vị thứ 1
  + 50%: tứ phân vị thứ 2
  + 75%: tứ phân vị thứ 3

  Trong đây đều là những thông tin cơ bản trong thống kê dữ liệu mà ta cần biết. Với việc dữ liệu cho trước đã làm sạch và được chuyển thành dạng số nên ta có thể thống kê nhanh được, nếu dữ liệu ban đầu ở dạng thô, ta cần phải có bước làm sạch dữ liệu trước khi khám phá dữ liệu.

  Bước 2, em dùng hàm $heatmap()$ trong thư viện seaborn để hiển thị tập dữ liệu train ban đầu sau khi được tương quan dữ liệu bằng hàm $corr()$ trong pandas. Tương quan dữ liệu giữa các đặc trưng trong tập dữ liệu giúp ta biết được những dữ liệu nào có mối liên hệ chặt chẽ với

nhau giúp ta dễ tham số hóa các đặc trưng đó trong mô hình ta cần làm. Khi dữ liệu được plot lên bằng $heatmap()$, và set annot = True: để hiển thị số lên từng pixels, cmap = "viridis": để

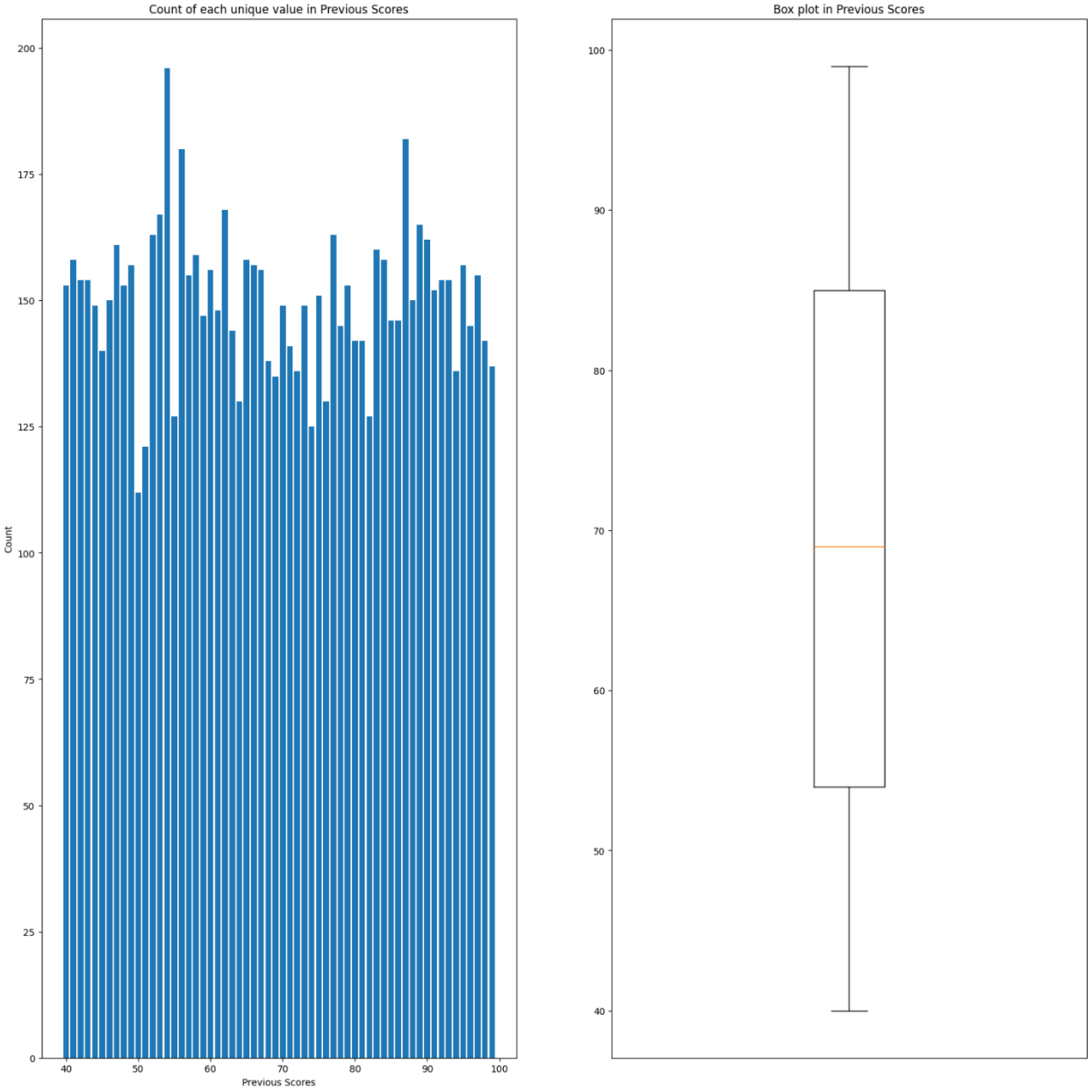


tô màu các pixels theo dạng viridis, để ý ta thấy các điểm màu vàng là các điểm màu thể hiện hoàn toàn tương quan với nhau và mang giá trị 1, ở đây với giá trị 1 thì chỉ có trường hợp là đặc trưng đó được so sánh với chính đặc trưng đó. Các số càng nhỏ, màu càng tím dần thì sẽ cho thấy sự tương quan giữa 2 đặc trưng càng ít dần. Đặc trưng mang giá trị kết quả là đặc trưng "Performance Index" nên ta sẽ chỉ tập trung vào các đặc trưng khác khi so sánh với đặc trưng "Performance Index". Trong hình, dễ dàng thấy được màu sáng nhất chỉ sau màu vàng là nằm ở "Previous Scores". Qua đó ta ngầm hiểu được đặc trưng "Previous Scores" có tương quan rất lớn với "Performance Index", có nghĩa là nếu ta plot dữ liệu của từng trường lên cùng một hình dưới dạng line, thì khả năng cao là đồ thị dạng line đó sẽ gần khớp với nhau. Do đó có thể thấy rằng trong mô hình hồi quy mà ta chuẩn bị làm, đặc trưng "Previous Scores" sẽ là một tham số cực kì quan trọng để xây dựng hàm hồi quy. Ngoài ra ta cũng thấy với "Hours Studied" thì cũng có sự tương quan nhất định với "Performance Index", dù không nhiều nhưng cũng có thể nói rằng "số giờ đã học" của mỗi học sinh cũng sẽ ảnh hưởng đến

“hiệu suất học tập” của học sinh đó, chỉ đứng sau “những điểm số đã thi trước đó”, nó cũng cho ta biết mô hình cũng nên tham khảo tới tham số đặc trưng “Hours Studied” này. Cuối cùng, điểm số thấp nhất mà ta thấy là “Extracurricular Activities”, điều này chứng tỏ gần như không có sự tương quan giữa đặc trưng này với “Performance Index”, và gần như ta sẽ không dùng nó trong việc huấn luyện mô hình và có thể xem xét bỏ đi đặc trưng này. Tương tự với “các hoạt động ngoại khóa” thì các đặc trưng “Sleep Hours”: “số giờ ngủ” và “Sample Question Papers Practiced”: “số bài kiểm tra thử đã làm” cũng có thể được xem xét bỏ đi vì sự tương quan quá thấp với đặc trưng kết quả. Và như vậy ta có thể biết rõ mình cần dùng những đặc trưng nào để xây dựng mô hình và những đặc trưng nào thì không nên dùng vì nếu dùng thì cũng chỉ tăng độ chính xác lên một ít nhưng lại mang đến hiệu suất tính toán giảm rõ rệt. Ngoài ra cũng có các phương pháp không bỏ đi đặc trưng mà chuẩn hóa (Normalization) đặc trưng đó sang miền dữ liệu khác để phục vụ cho mô hình tốt hơn.

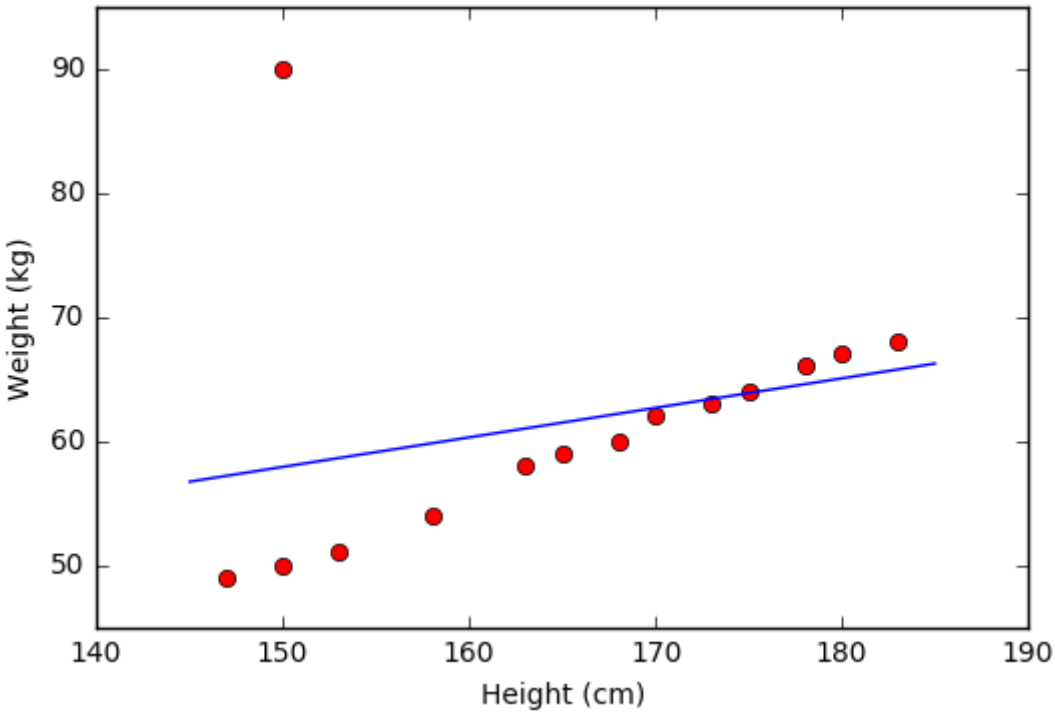
Tiếp theo, để trực quan hóa (visualization) các đặc trưng theo nhiều cách hơn thì em sẽ dùng 1 vòng lặp để chạy qua từng đặc trưng, sau đó em dùng hàm $value\_counts()$ để lấy ra số lần xuất hiện của 1 giá trị riêng biệt (unique) trong đặc trưng đó và dùng $sort\_index()$ để sort lại danh sách các giá trị unique đó theo index (ở đây index chính là giá trị unique đó, còn values: chính là số lần xuất hiện nên ta phải dùng $sort\_index()$ chứ không dùng $sort\_values()$). Em sẽ tạo ra 2 ảnh liền kề nhau trong một khung ảnh với $figsize = 20 * 20$ inches (khớp với khung ngang của vscode IDE) bằng hàm $subplots()$ của matplotlib. Vì hàm này trả về 2 kết quả, 1 là khung ảnh chứa 2 ảnh đó, 2 là trục chứa ảnh. Với trục chứa ảnh ($axes$) thì là mảng chứa thông tin của 2 ảnh. Nên với $index$ 0 sẽ là ảnh đầu tiên và $index$ 1 là ảnh thứ 2 từ trái qua. Ảnh bên trái em sẽ tạo ra một đồ thị dạng cột (bar chart) thể hiện số lần xuất hiện của mỗi unique value của đặc trưng tại vòng lặp đó. Em set tên ở trục x sẽ là tên đặc trưng và tên ở trục y sẽ là “Count”, và em cũng set title cho ảnh đó. Với ảnh thứ 2 thì em hiển thị nó theo đồ thị dạng hộp (boxplot) cho đặc trưng tại vòng lặp đó. Cuối cùng em sẽ show hình ảnh bằng hàm $plt.show()$ ngoài ra em còn show thêm độ thị dạng đường (line plot) thể hiện sự tương quan giữa đặc trưng tại vòng lặp đó với đặc trưng kết quả “Performance Index”. Trục x sẽ là tập dữ

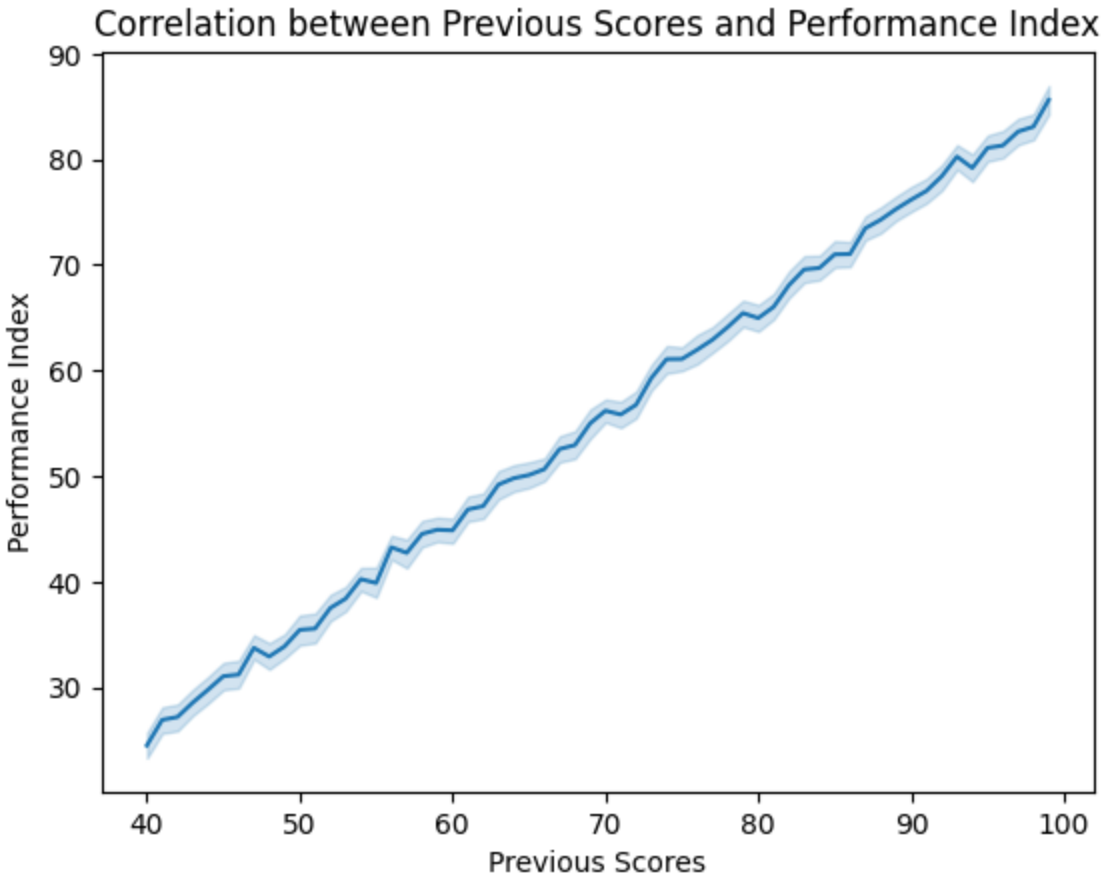
liệu của đặc trưng tại vòng lặp đó, trục y sẽ là tập đặc trưng "Performance Index". Trong $train.columns$ là hàm hiển thị các tên các đặc trưng trong tập dữ liệu và em lấy $train.columns[-1]$ tức là lấy tên đặc trưng cuối cùng đó là: "Performance Index".



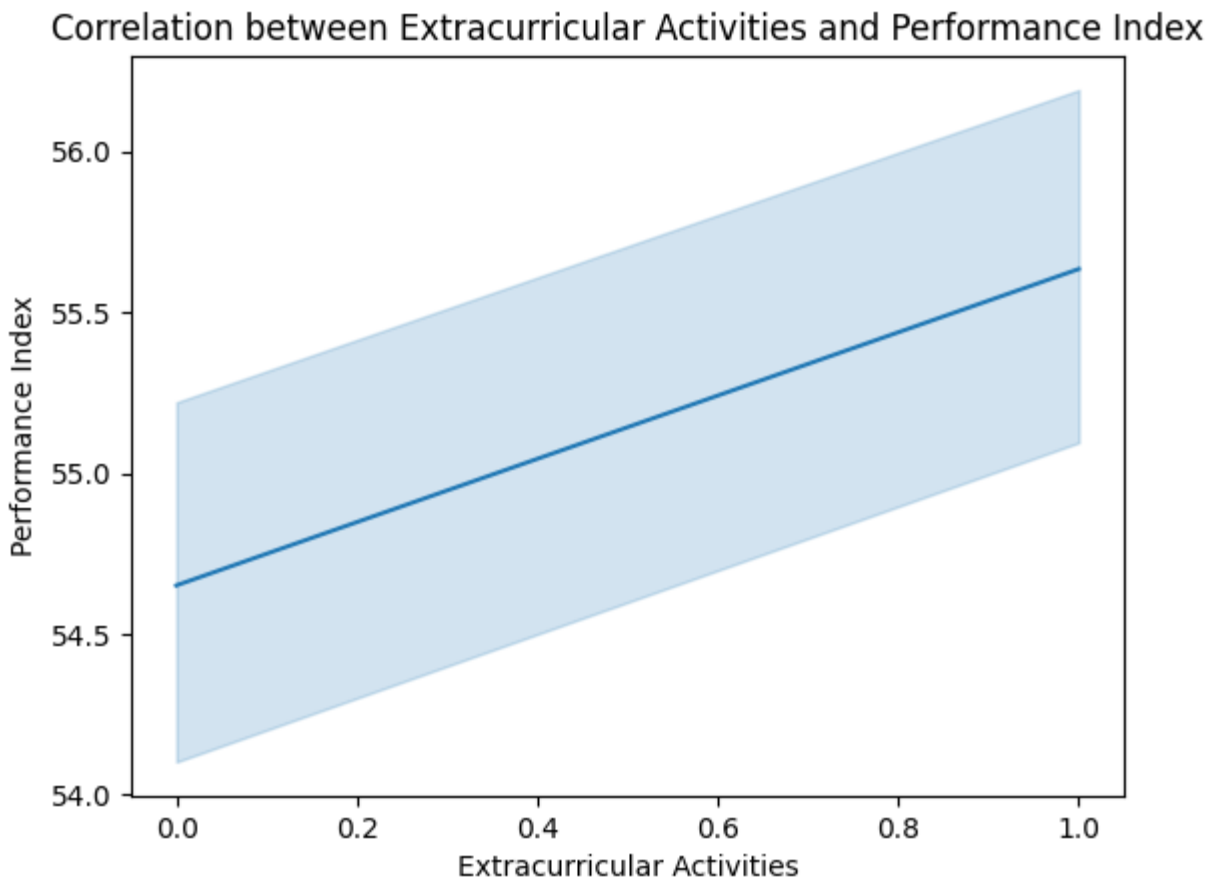
Đây là ảnh thể hiện bar chart và box plot cho đặc trưng "Previous Scores". Ta có thể thấy được một số thông tin là tập dữ liệu trên được phân phối khá đều (tần suất xuất hiện), và vùng có giá trị ở giữa là vùng xuất hiện nhiều nhất. Ngoài ra ta cũng thấy số lượng nhiều nhất là giá trị trong khoảng (53-56), còn bên đồ thị box plot thì trước tiên ta thấy dữ liệu không có điểm outlier nào. Vì trong thực tế nếu ta áp dụng các mô hình hồi quy cho dữ liệu có điểm outlier thì mô hình không có độ chính xác cao nữa.

Hạn chế của Linear Regression là nó rất nhạy cảm với nhiễu (sensitive to noise). Với ảnh trên, chỉ cần có một outlier thì tập dự đoán của mô hình sẽ bị lệch với tập test ngay. Vì vậy bước làm sạch dữ liệu luôn là bước đầu tiên và là bước cực kì quan trọng trước khi thực hiện xây dựng mô hình [8]





Đây là đồ thị dạng đường thể hiện sự tương quan giữa đặc trưng "Previous Scores" và đặc trưng kết quả "Performance Index", như đã nói ở trên vì đặc trưng "Previous Scores" có sự tương quan rất lớn với đặc trưng "Performance Index" nên với độ thị line này ta thấy dữ liệu được sắp xếp gần như theo 1 đường thẳng, vậy mô hình Linear Regression dùng đặc trưng này nhiều khả năng sẽ cho kết quả tốt. Ngoài ra để ý thấy tại mỗi giá trị cụ thể trong "Previous Scores", biên độ lệch của nó so với "Performance Index" cũng khá nhỏ, do đó đây chính là một yếu tố làm sự tương quan giữa 2 đặc trưng cao.



Đây là đồ thị dạng đường thể hiện sự tương quan giữa đặc trưng "Extracurricular Activities" và đặc trưng kết quả "Performance Index", như lúc nãy, ta thấy tương quan giữa 2 đặc trưng này là thấp nhất trong các đặc trưng còn lại nên trong hình này cũng là 1 minh chứng rõ ràng cho việc chệnh lệch đó. Cuối cùng, nãy giờ chúng ta thấy các hình ảnh đều có dạng đường thẳng mà không có dạng parabol hay dạng khác, nên ta sẽ nghĩ đến việc mô hình sẽ là mô hình hồi quy tuyến tính bậc nhất với các tham số là các đặc trưng của tập dữ liệu.

2.2. Yêu cầu 2

Trước tiên ta sẽ có bước preprocess cho dữ liệu X_{train}, X_{test} bằng cách ghép cột giá trị 1 vào trước các cột đặc trưng, ta thực hiện điều này để cho tham số w_0 của mô hình hồi quy. Ngoài ra, em còn tạo ra $features$: là mảng chứa các tên của các header trong tập dữ liệu, và $feature\_inshort$: là mảng chứa các ký tự in hoa đầu tiên của tên các đặc trưng. Để cho ra được $feature\_inshort$, em áp dụng list comprehension trong python, đầu tiên là duyệt các đặc trưng trong tập các đặc trưng $features$, với mỗi đặc trưng ta sẽ dùng hàm $split()$ để tách string ra thành mảng chứa từng từ, vì ta chỉ lấy 2 từ đầu tiên nên sẽ slicing nó $[0:2]$, ta dùng hàm $map()$ để giảm bớt phải ghi 1 vòng lặp trong từng từ, và cũng để map lấy từng kí tự đầu tiên trong từng từ mà nó map tới. Cuối cùng với những kí tự mà hàm map trả ra, vì nó không dính vào nhau nên ta sẽ dùng hàm ''.join() của python để nối 2 kí tự riêng biệt ấy thành một từ và trả về một list chứa các từ đã tách kí tự đầu của 2 từ đầu tiên xong.

a) Yêu cầu 2a

Với yêu cầu này ta sẽ xây dựng mô hình hồi quy với công thức sau:

$$y = w_0 + w_1 * HS + w_2 * PS + w_3 * EA + w_4 * SH + w_5 * SQ$$

Sau khi có $X_{train\_pre}, X_{test\_pre}$ thì ta sẽ áp dụng OLS vào để khớp mô hình với dữ liệu $[X_{train\_pre}, y_{train}]$, sau đó ta có được mô hình hồi quy với các trọng số tối ưu, ta sẽ xuất công thức hồi quy trên. Ở đây em mong muốn code mà không cần tự ghi những tên bên ngoài vào như "Performance Index" nên em luôn lấy các tên ra bằng các hàm có sẵn hoặc tự lấy ra. Ví dụ cho câu nói trên là "Performance Index" là từ cuối cùng trong header tập dữ liệu nên em sẽ lấy nó từ $train.columns[-1]$. Cũng như với việc xuất công thức, vì đã có sẵn mảng chứa các tên của các đặc trưng $features$ và mảng chứa chữ cái in hoa của 2 từ đầu tiên của các đặc trưng $features\_inshort$ nên em sẽ dùng nó để thay thế việc ghi từng tên riêng vào. Để xuất công thức với làm tròn 3 chữ số thập phân em dùng $:.3f$. Nhận thấy ở công thức hồi quy trên, tên của các đặc trưng nối với trọng số tối ưu đôi một và chỉ có trọng số đầu tiên không có tên tương ứng tới nên em sẽ ghi $weights[0]$ cho trọng số đầu tiên trong hàm string xuất ra ($weights$ là biến được gán từ $LR.getWeights()$). Tiếp theo để thực hiện ghép nối đôi một trọng số và tên đặc trưng thì em sẽ $zip(weights[1:], features\_inshort)$, weights[1:] là để lấy tất cả trọng số còn lại ngoại trừ trọng số đầu tiên đã in trong string. Sau đó cũng dùng list comprehension để duyệt qua từng từ ghép đó, và lấy ra giá trị trọng số làm tròn 3 chữ số thập phân và tên ngắn của đặc trưng đó tương ứng tại mỗi vòng lặp. Cuối cùng là dùng hàm $" + ".join()$ để nối từng từ ghép đó với nhau qua dấu " + ".

Sau đó em sẽ in giá trị MAE bằng cách dùng hàm $MAE()$ với 2 đối số (parameter) là: y_{pred} và y_{test}. Ta dùng hàm $predict()$ để tính ra y_{pred} dựa trên X_{test_pre}.

```
Student Performance = -33.969 + 2.852HS + 1.018PS + 0.604EA + 0.474SH + 0.192SQ
```

Công thức hồi quy sau khi xây dựng xong

```
MAE = 1.595648688476295
```

Giá trị MAE sau khi đã tính xong

Nhận xét:

Với mô hình trên thì cho ra kết quả MAE khá là khả quan khi giá trị MAE không quá cao. Tuy nhiên, mô hình lại sử dụng một lượng lớn các đặc trưng khiến hiệu suất training mô hình sẽ kém. Vì như ta đã thấy các đặc trưng như "Extracurricular Activities", "Sleep Hours", "Sample Question Papers Practiced" có sự tương quan rất rất thấp so với "Performance Index". Nên việc bỏ đi các đặc trưng đó nhưng kết quả không tệ đi bao nhiêu thì sẽ là một giải pháp mô hình hiệu quả hơn.

b) Yêu cầu 2b

Để xây dựng mô hình duy nhất một đặc trưng và áp dụng k-fold cross validation.

Đầu tiên, ta sẽ shuffle tập dữ liệu X_{train}, y_{train} bằng cách shuffle các index. Em tạo ra các index ứng với số dòng của tập train bằng hàm $np.arange()$, sau đó em dùng

$np.random.shuffle()$ để shuffle các index đó, tiếp theo em lấy $X_{train\_shuffle}, y_{train\_shuffle}$ bằng cách gán dùng $iloc(idx)$ và bỏ index đã shuffle vào để lấy ra tập $X_{train\_shuffle}, y_{train\_shuffle}$ đã shuffle từ tập X_{train}, y_{train}, $iloc()$ là hàm lấy ra dữ liệu dựa trên index [9]. Sau khi có tập shuffle thì em sẽ dùng hàm $np.hsplit()$ để tách các cột của $X_{train\_shuffle}$ ra thành từng cột dựa trên số cột của $X_{train\_shuffle}$ và gán vô $features\_arr$. Vì mỗi model sẽ là một đặc trưng khác nhau nên em tạo ra model bằng cách dùng list comprehension, duyệt qua từng đặc trưng trong $features\_arr$, và gán cột 1 vào đầu bằng $np.hstack()$ sau đó em gọi hàm $linear\_regression\_model(models, y\_train\_shuffle, k = 5)$ và trả về $MAE\_features$. Sau đó để in ra bảng giống yêu cầu của cô, em dùng $pd.Dataframe()$ để đưa dictionary chứa header của bảng và nội dung bảng ứng theo cột vào, chỉ số $index$ là để thể hiện số thứ tự trong bảng.

Để lấy ra mô hình tốt nhất thì em dùng np.argmin() để lấy ra index của giá trị nhỏ nhất trong $MAE\_features$ sau đó xây dựng mô hình với đặc trưng tốt nhất, để lấy dữ liệu của đặc trưng tốt nhất và cột 1 thì em dùng slicing $[:, [0, best\_model\_idx + 1]]$, để lấy ra tất cả dòng của dữ liệu, còn cột thì lấy ra cột 0: cột có giá trị 1 và cột $best\_model\_idx + 1$: là cột của đặc trưng "Previous Scores". Cuối cùng là gọi hàm $MAE()$ để tính giá trị MAE của y_{pred} và y_{test} rồi em cho hiển thị giá trị MAE và công thức hồi quy của đặc trưng tốt nhất.

	Mô hình với 1 đặc trưng	MAE
1	Hours Studied	15.448240
2	Previous Scores	6.618202
3	Extracurricular Activities	16.197720
4	Sleep Hours	16.187498
5	Sample Question Papers Practiced	16.185434

Hiển thị giá trị MAE ứng với từng mô hình (MAE tính trên tập train dựa trên k-fold cross validation)

```
MAE = 6.544277293452511
Student Performance = -14.989 + 1.011*PS
```

Giá trị MAE mô hình có đặc trưng tốt nhất đưa ra

Nhận xét:

Ta thấy với nhận xét ban đầu khi quan sát thống kê và trực quan hóa dữ liệu bằng hình ảnh thì kết quả cho ra của mô hình hiện tại giống hoàn toàn với nhận xét đó, khi mà "Previous Scores" cho ra kết quả tốt nhất rồi đến "Hours Studied", cuối cùng là 3 mô hình tệ nhất "Extracurricular Activities", "Sleep Hours", "Sample Question Papers Practiced"

Với việc xây dựng từng mô hình với từng đặc trưng duy nhất khác nhau thì mô hình sẽ rất kém. Mặc dù với đặc trưng "Previous Scores" là đặc trưng tốt nhất và trước khi làm thì ta hy vọng

nếu chỉ với 1 đặc trưng duy nhất này xây dựng mô hình thì kết quả sẽ tốt, nhưng thực tế thì kết quả cũng không tốt mấy khi $MAE \approx 6.544$, quá cách biệt so với mô hình ở câu a. Với điều đó ta sẽ hy vọng cần thêm đặc trưng hỗ trợ và có thể tăng độ mạnh của đặc trưng bằng cách bình phương hoặc nhân với một đặc trưng khác để cho ra mô hình tốt hơn ở câu c.

c) Yêu cầu 2c

Với việc chỉ có 1 đặc trưng thì kết quả quá kém, hoặc có nhiều đặc trưng thì kết quả tốt nhưng hiệu suất hoạt động lại không tốt. Ta sẽ nghĩ đến một số mô hình có thể mang đến hiệu quả cao hơn và hiệu suất lại tốt hơn khi áp dụng một số phương pháp để tăng hiệu quả sử dụng đặc trưng tốt nhất.

Mô hình 1: $y = w_0 + w_1 * HS + w_2 * PS$

Đây là mô hình sử dụng 2 đặc trưng là "Hours Studied", "Previous Scores" vì biết 2 đặc trưng này mang tính tương quan với đặc trưng kết quả "Performance Index" khá cao, mặc dù với đặc trưng "Hours Studied" cho kết quả tương quan không quá cao nhưng để bổ sung thêm độ hiệu quả cho mô hình thì em sẽ dùng thử 2 đặc trưng tốt nhất xem sao, vì em cũng để ý rằng khi cộng nhiều đặc trưng vào thì kết quả sẽ tốt hơn nhiều. Ngoài ra nếu nó kém quá nhiều so với mô hình 5 đặc trưng thì đây có lẽ sẽ là mô hình hiệu quả bởi kết quả lẫn hiệu suất nó mang lại.

Mô hình 2: $y = w_0 + w_2 * PS^2$

Đây là mô hình sử dụng mô hình hồi quy tuyến tính bậc 2 và chỉ sử dụng 1 đặc trưng là "Previous Scores". Mặc dù đã biết sự tương quan giữa các đặc trưng với đặc trưng kết quả có hình dáng đường thẳng tức phương trình bậc nhất, được hiển thị ở phần khám phá dữ liệu ở câu 1. Tuy nhiên để thử nghiệm xem thực tế có đúng là sự tương quan là phương trình bậc 1 không thì em sẽ thử với phương trình bậc 2 để xem nếu cho ra kết quả tệ hơn thì quá trình EDA của em đã chính xác, còn nếu không thì có thể EDA của em bị sai ở đâu đó. Và lý do em bình phương đặc trưng "Previous Scores" là vì đặc trưng này là đặc trưng tốt nhất và em muốn tăng độ mạnh của đặc trưng bằng cách bình phương nó lên, ngoài ra cũng là để thử nghiệm với hồi quy bậc 2.

Mô hình 3: $y = w_0 + w_1 * HS * PS$

Với mô hình này, em vẫn tiếp tục sử dụng 2 đặc trưng là "Hours Studied", "Previous Scores", lý do thì như em đã đưa ra ở trên, là vì 2 đặc trưng này là tốt nhất. Ở mô hình này, em quyết định thử nhân 2 đặc trưng thay vì bình phương đặc trưng tốt nhất, để tạo ra 1 đặc trưng mới là 2 đặc trưng nhân nhau. Để kiểm nghiệm xem với mô hình 2 đặc trưng khác nhau nhân nhau, thay vì bình phương thì sẽ cho ra kết quả có cải thiện hơn so với mô hình chỉ 1 đặc trưng duy nhất.

Trong cả 3 mô hình trên, em quyết định không sử dụng cả 3 đặc trưng còn lại vì tính tương quan với đặc trưng kết quả quá kém.

Thực hiện:

Đầu tiên em tạo 1 mảng $models_X$ chứa 3 mô hình trên tương tự như câu 2b là em sẽ ghép cột 1 vào trước các đặc trưng, và em sử dụng $features\_arr$ là mảng chứa từng np.array là các đặc

trưng khác nhau. Sau đó là tương tự như câu 2b em chạy hàm $linear\_regression\_model(models_X, y_{train\_shuffle}, k = 5)$ để lấy ra $MAE\_features$

em tạo bảng như cô trình bày để hiển thị các giá trị MAE của các mô hình và in nó ra, rồi chọn mô hình tốt nhất, train mô hình tốt nhất với tập train rồi test mô hình với tập test rồi in giá trị MAE cuối cùng sau khi so sánh với tập test và hiển thị công thức hồi quy.

	Mô hình	MAE
1	Sử dụng 2 đặc trưng (Hours Studied, Previous S...	1.816697
2	Sử dụng 1 đặc trưng (bình phương đặc trưng Pre...	6.767018
3	Sử dụng 2 đặc trưng (Hours Studied, Previous S...	11.086185

Hiển thị giá trị MAE ứng với từng mô hình (MAE tính trên tập train dựa trên k-fold cross validation)

```
MAE = 1.83943639765299
Student Performance = -29.747 + 2.856HS + 1.018PS
```

Giá trị MAE mô hình có đặc trưng tốt nhất đưa ra

Nhận xét:

Đúng với suy nghĩ sau khi thực hiện EDA và sau khi thực hiện 2 mô hình trước đó thì với chỉ 2 đặc trưng tốt nhất mô hình đã có kết quả gần tốt bằng mô hình sử dụng 5 đặc trưng như câu 2a, ngoài ra nó lại còn tối ưu về mặt hiệu suất hơn. Mặc dù mô hình chỉ 1 đặc trưng tốt nhất có kết quả khá tệ, nhưng chỉ cần thêm 1 đặc trưng tốt thứ 2 thì mô hình đã cho ra kết quả tốt hơn rất nhiều so với mô hình 1 đặc trưng tốt nhất đó. Và em nghĩ nếu có thêm 1 hay 2 đặc trưng khác cộng vào thêm thì mô hình cũng sẽ tốt hơn nhưng chắc chắn tốt lên không nhiều vì ta có thể thấy cách biệt của mô hình với 5 đặc trưng so với 2 đặc trưng chỉ là ≈ 0.24, là một khoảng quá nhỏ để đánh đổi mặt hiệu suất. Về tổng thể, chỉ với 2 đặc trưng tốt nhất đã có thể tạo ra một mô hình hồi quy tốt về mặt hiệu suất lẫn kết quả.

Khi nhìn vào mô hình hồi quy bậc 2, ta không ngạc nhiên khi kết quả của nó khá tệ vì ta biết là sự tương quan giữa các đặc trưng với đặc trưng kết quả là phương trình bậc 1 (dạng đường thẳng) nên với phương trình hồi quy bậc 2 (dạng parabol) thì sai lệch về độ chính xác là điều tất yếu, nhưng cũng là để thử nghiệm với việc EDA ban đầu có chính xác hay không.

Mặc dù mô hình 2 kém nhưng mô hình 3 lại kém hơn, với chủ đích là dùng đặc trưng tốt thứ 2 nhân với đặc trưng tốt nhất để tăng độ mạnh của đặc trưng tốt nhất nhưng kết quả cho ra lại trái ngược hoàn toàn. Có lẽ việc tạo đặc trưng mới này đã mang lại một dữ liệu không tương quan quá nhiều với đặc trưng kết quả "Performance Index". Và vì như vậy nên ta sẽ không thể nhân 2 đặc trưng với nhau để cải thiện mô hình nếu chưa xử lí dữ liệu cho các đặc trưng trước khi xây dựng. Có thể có một số phương pháp để xử lí lại dữ liệu đặc trưng nếu phát hiện nó kém, ví dụ như chuẩn hóa đặc trưng (Normalization), …

Tài liệu tham khảo

[1] Seaborn, "seaborn: statistical data visualization," [Online]. Available: https://seaborn.pydata.org/. [Accessed 16 8 2024].

[2] wikipedia, "pandas (software)," [Online]. Available: https://en.wikipedia.org/wiki/Pandas_(software). [Accessed 16 8 2024].

[3] wikipedia, "Ordinary least squares," [Online]. Available: https://en.wikipedia.org/wiki/Ordinary_least_squares. [Accessed 16 8 2024].

[4] wikipedia, "Linear regression," [Online]. Available: https://en.wikipedia.org/wiki/Linear_regression. [Accessed 16 8 2024].

[5] wikipedia, "Mean absolute error," [Online]. Available: https://en.wikipedia.org/wiki/Mean_absolute_error. [Accessed 16 8 2024].

[6] "k-Fold Cross-Validation là gì?," [Online]. Available: https://trituenhantao.io/kien-thuc/gioi-thieu-ve-k-fold-cross-validation/. [Accessed 16 8 2024].

[7] V. H. Tiệp, "Mục đích của EDA," [Online]. Available: https://machinelearningcoban.com/tabml_book/ch_data_processing/eda_purpose.html. [Accessed 16 8 2024].

[8] V. H. Tiệp, "Bài 3: Linear Regression," [Online]. Available: https://machinelearningcoban.com/2016/12/28/linearregression/. [Accessed 16 8 2024].

[9] "pandas.DataFrame.iloc," [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html. [Accessed 16 8 2024].

[10] "Normalization (statistics)," [Online]. Available: https://en.wikipedia.org/wiki/Normalization_(statistics). [Accessed 16 8 2024].