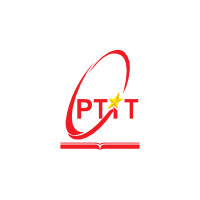
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**Bài tập lớn Nhập môn trí tuệ nhân tạo**

**Nhóm 21: Dự đoán điểm thi học sinh**

|  |  |
| --- | --- |
| **Giảng viên** | **: Nguyễn Mai Trang** |
| **Sinh viên thực hiện** | **: Vũ Phương Thanh** |
| **Mã sinh viên** | **: B20DCCN638** |
| **Gmail** | **: vpthanh1302@gmail.com** |
| **Điện thoại** | **: 0859490555** |
| **Nhóm** | **: 05** |

***Hà Nội – 2023***

**I) Giới thiệu đề tài**

1) Giới thiệu

Đề tài "Dự đoán điểm thi học sinh" là một nghiên cứu nhằm phân tích dữ liệu điểm thi của học sinh dựa vào số giờ học tập. Sử dụng các phương pháp phân tích thống kê và học máy để đưa ra dự đoán về điểm thi dựa trên thời gian học tập

2) Mục đích

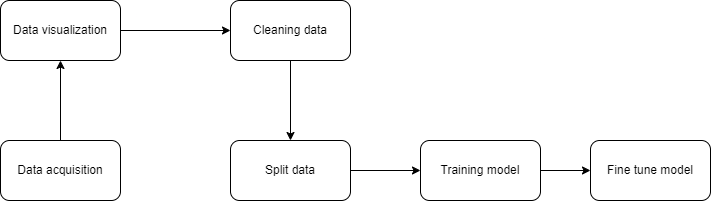
Các kết quả phân tích sẽ giúp cho những người quản lý giáo dục và giáo viên có thể đưa ra các quyết định hợp lý nhằm nâng cao chất lượng giảng dạy và hỗ trợ học sinh đạt được thành tích tốt hơn.

**II) Phương pháp nghiên cứu**

1) Áp dụng machine learning vào nghiên cứu

Trong dự án dự đoán điểm học sinh, việc áp dụng machine learning giúp cho việc dự đoán kết quả học tập của học sinh dựa trên các đặc trưng được thu thập từ quá trình học tập của họ trở nên chính xác hơn. Nhờ vào khả năng học tập từ dữ liệu của các thuật toán machine learning, ta có thể dễ dàng tìm ra những đặc trưng quan trọng ảnh hưởng đến kết quả học tập của học sinh. Từ đó, ta có thể tập trung vào việc thu thập và lưu trữ những đặc trưng này, giúp cho quá trình phân tích và dự đoán trở nên hiệu quả hơn.

2) Các bước tiến hành dự án



Bước 1: Thu thập dữ liệu

Cần thu thập dữ liệu về điểm số của học sinh. Dữ liệu này có thể được thu thập từ các nguồn khác nhau, chẳng hạn như cơ sở dữ liệu của trường hoặc các trang web chia sẻ dữ liệu.

Bước 2: Trực quan hóa dữ liệu

Biểu diễn dữ liệu dưới dạng hình ảnh hoặc đồ thị để giúp hiểu rõ hơn về dữ liệu và tìm ra các mẫu, xu hướng hoặc thông tin quan trọng. Thông qua việc sử dụng các công cụ trực quan hóa dữ liệu như biểu đồ, bản đồ, hình ảnh, người sử dụng có thể dễ dàng phân tích và hiểu các dữ liệu phức tạp, đồng thời giúp truyền tải thông tin dễ dàng hơn cho người đọc.

Bước 3: Làm sạch dữ liệu

Làm sạch dữ liệu quá trình xử lý dữ liệu để loại bỏ hoặc sửa chữa các giá trị dữ liệu không hợp lệ, thiếu sót hoặc không chính xác. Mục tiêu của việc làm sạch dữ liệu là tạo ra một tập dữ liệu đáng tin cậy và thích hợp cho quá trình huấn luyện và kiểm tra mô hình ML.

Bước 4: Chia tập dữ liệu

Chia tập dữ liệu là quá trình chia dữ liệu thành các tập con riêng biệt để sử dụng trong quá trình huấn luyện và đánh giá mô hình. Đây là một bước quan trọng để đảm bảo mô hình được đánh giá một cách khách quan và có khả năng tổng quát hóa cho dữ liệu mới.

Bước 5: Huấn luyện mô hình

Huấn luyện mô hình là quá trình huấn luyện mô hình máy học bằng cách cung cấp cho nó dữ liệu huấn luyện để học cách ánh xạ từ dữ liệu đầu vào đến dữ liệu đầu ra dự đoán.

Bước 6: Điều chỉnh mô hình

Điều chỉnh mô hình là quá trình điều chỉnh lại mô hình đã được huấn luyện sẵn trên một tập dữ liệu nhỏ hoặc tương tự để cải thiện hiệu suất hoặc đáp ứng nhu cầu cụ thể.

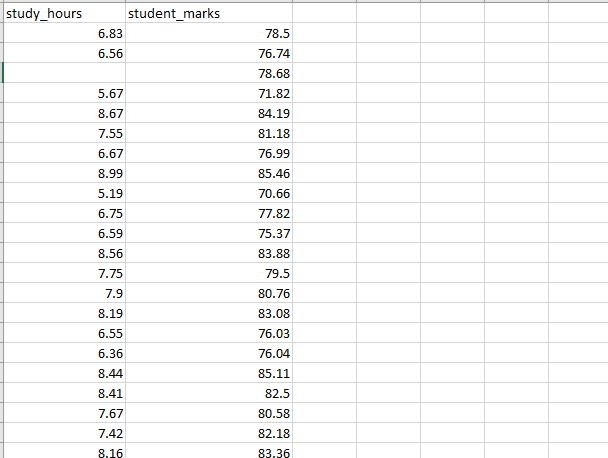
3) Mô hình học máy áp dụng

a) Linear Regressor

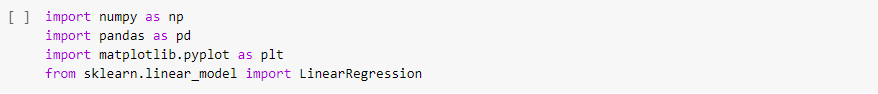
Linear Regression là một thuật toán học máy phân loại trong lĩnh vực học có giám sát (supervised learning). Nó được sử dụng để dự đoán một biến phụ thuộc liên tục dựa trên một hoặc nhiều biến độc lập. Trong Linear Regression, mục tiêu là tìm ra một mô hình tuyến tính (linear model) tốt nhất để xấp xỉ quan hệ giữa các biến đầu vào và biến đầu ra.

**III) Cài đặt chương trình**

File dữ liệu:



Các thư viện python sử dụng để thực hiện đề tài:

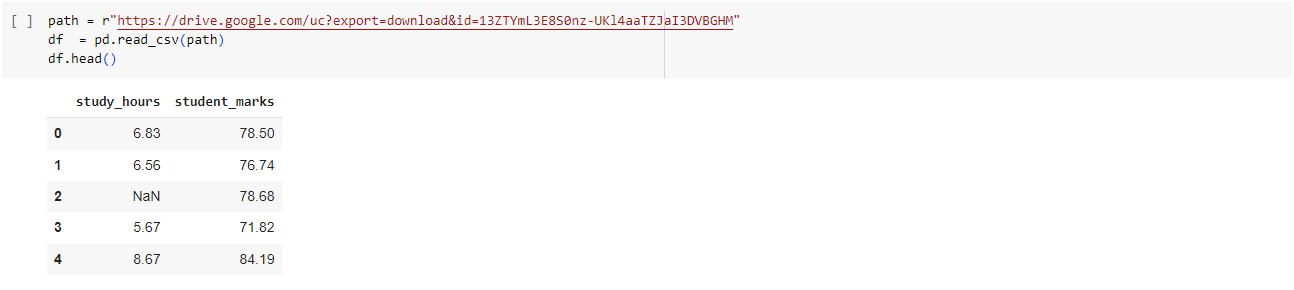


**numpy** : thư viện tính toán số học, đặc biệt là để làm việc với các mảng và ma trận

**pandas** : thư viện để làm việc với dữ liệu trong các bảng và các cột

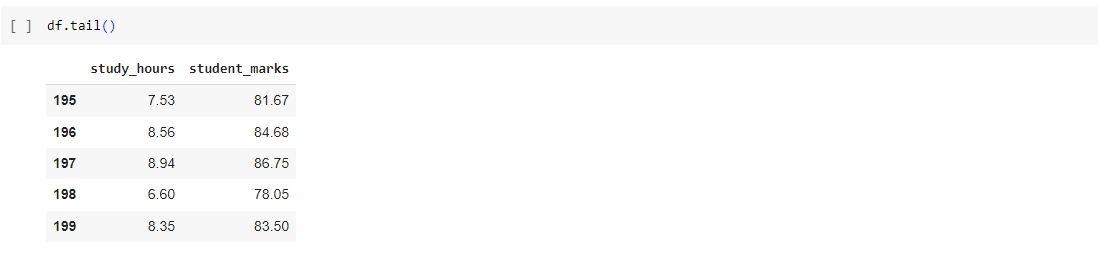
**matplotlib** : thư viện đồ họa để vẽ các biểu đồ (plots) như đường, histogram, scatter, bar, ..

**sklearn.linear\_model:** là một module trong thư viện scikit-learn (hay còn gọi là sklearn) để xử lý mô hình hồi quy tuyến tính (linear regression). Nó cung cấp các lớp và hàm để thực hiện hồi quy tuyến tính và các biến thể của nó.

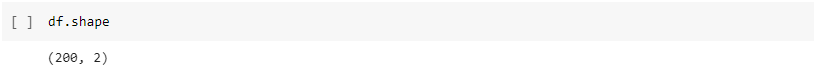


Ta sử dụng biến Path để trỏ đến file CSV trên google driver, rồi sau đó sử dụng thư viện pandas dữ liệu file CSV vào 1 dataframe.

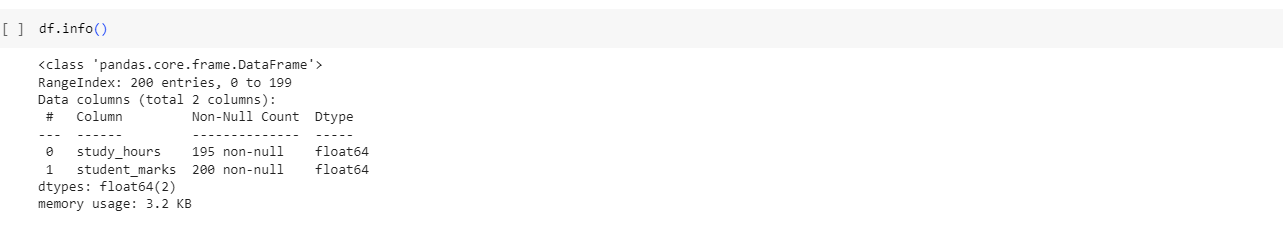
Câu lệnh df.head() dùng để hiển thị một số dòng đầu tiên của dataframe. Nếu không truyền gì vào thì câu lệnh sẽ hiển thị ra 5 dòng.



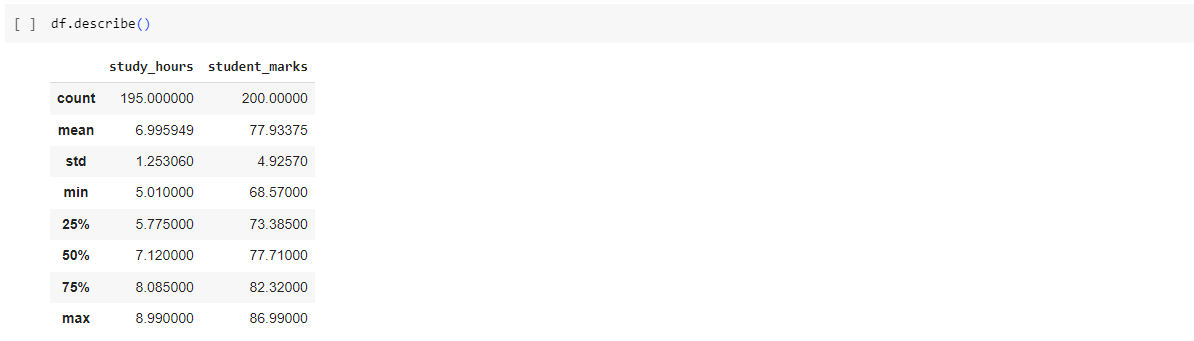
Df.tail() dùng để hiển thị một số dòng cuối của dữ liệu, nếu không truyền tham số nào vào thì mặc định sẽ in ra 5 dòng cuối.



Câu lệnh df.shape được sử dụng để trả về kích thước của DataFrame df dưới dạng tuple(số hàng, số cột)

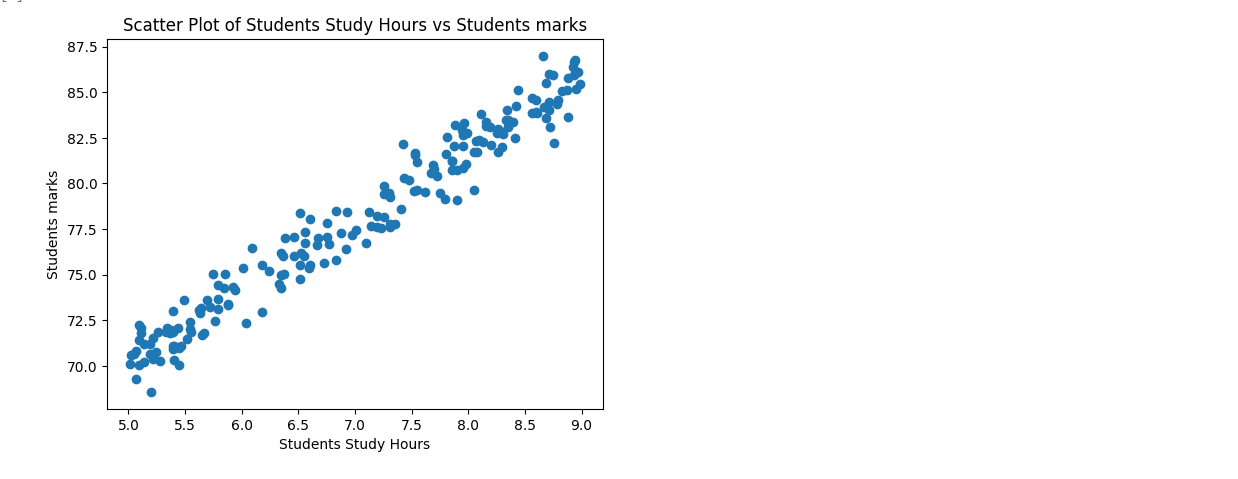


Câu lệnh df.info() được sử dụng để hiển thị thông tin tổng quan về DataFrame df, bao gồm số lượng dòng, số lượng cột, các loại dữ liệu của các cột, số lượng giá trị non-null và thông tin về việc sử dụng bộ nhớ.



Câu lệnh df.describe() được sử dụng để tạo một tóm tắt thống kê của DataFrame df. Nó trả về các thống kê mô tả về các cột số học của DataFrame như giá trị trung bình, độ lệch chuẩn, giá trị tối thiểu, giá trị tứ phân vị và giá trị tối đa.

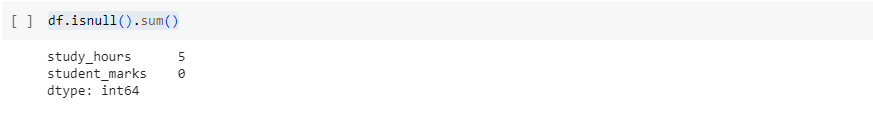




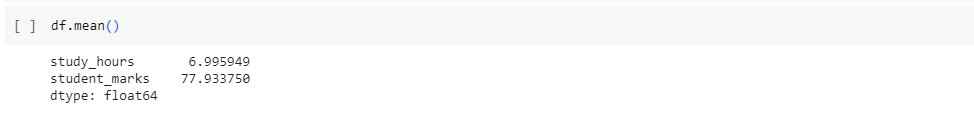
Câu lệnh plt.scatter(x=df.study\_hours, y=df.student\_marks) sẽ tạo ra một biểu đồ scatter plot với trục x được đại diện bởi cột study\_hours trong DataFrame df và trục y được đại diện bởi cột student\_marks. Mỗi điểm trên biểu đồ biểu thị một cặp giá trị (study\_hours, student\_marks) tương ứng của mỗi mẫu trong DataFrame.

Ta nhận thấy rằng Students marks = Students Study Hours \* m + c

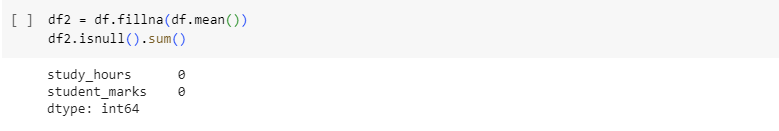
Hay y = m\*x + c



Câu lệnh df.isnull().sum() sẽ trả về một Series chứa thông tin về số lượng giá trị null trong mỗi cột của DataFrame df.

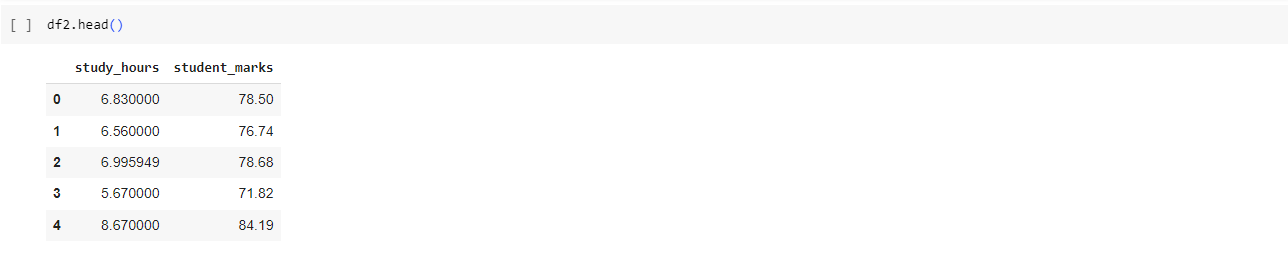


df.mean() sẽ tính giá trị trung bình của mỗi cột trong DataFrame.

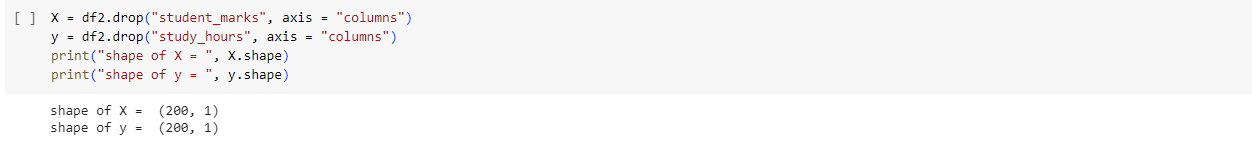


Câu lệnh df2 = df.fillna(df.mean()) có nghĩa là chúng ta thay giá trị NaN ( Not a number ) bằng giá trị trung bình của cột đó, rồi sau đó chúng ta gán dataframe mới vào df2.

Sử dụng câu lệnh df2.isnull().sum() để kiểm tra xem trong cột có giá trị nào có giá trị null hay không



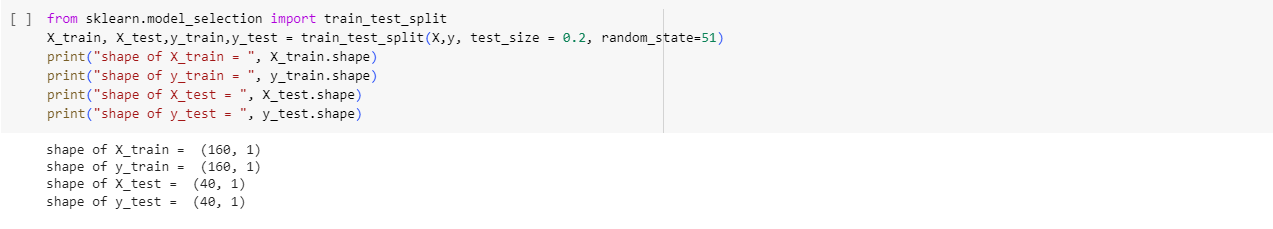
Kiểm tra 5 dòng đầu của df2



X = df2.drop(“student\_marks”, axis =”columns”) nghĩa là tạo 1 dataframe mới X từ df2 nhưng loại bỏ đi cột student\_marks.

y = df2.drop(“study\_hours”, axis =”columns”) nghĩa là tạo 1 dataframe mới X từ df2 nhưng loại bỏ đi cột student\_marks.

Sau đó in ra shape của 2 dataframe này



Câu lệnh trên sử dụng train\_test\_split từ module sklearn.model\_selection để chia dữ liệu thành hai phần: tập huấn luyện (training set) và tập kiểm tra (test set), để thực hiện quá trình huấn luyện và đánh giá mô hình.

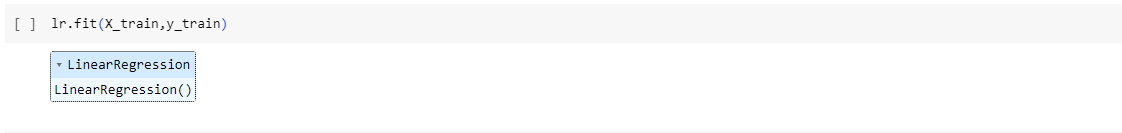
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=51): Câu lệnh này thực hiện việc chia dữ liệu thành các tập huấn luyện và kiểm tra. Cụ thể, tập dữ liệu X và y được chia thành X\_train, X\_test, y\_train, y\_test. Tham số test\_size=0.2 xác định tỷ lệ dữ liệu sẽ được chia cho tập kiểm tra, ở đây là 20% (0.2) của dữ liệu ban đầu. Tham số random\_state=51 xác định một giá trị để kiểm soát quá trình chia dữ liệu và đảm bảo kết quả chia dữ liệu như nhau mỗi khi chạy lại chương trình.

Sau khi thực hiện câu lệnh trên, X\_train và y\_train sẽ là tập huấn luyện, X\_test và y\_test sẽ là tập kiểm tra. Tập huấn luyện sẽ chứa 80% dữ liệu ban đầu, và tập kiểm tra sẽ chứa 20% dữ liệu ban đầu.

Tiếp theo, câu lệnh print("shape of X\_train = ", X\_train.shape) và các câu lệnh tương tự được sử dụng để in ra kích thước (số hàng, số cột) của X\_train, y\_train, X\_test, và y\_test.



Câu lệnh lr = LinearRegression() tạo ra một đối tượng LinearRegression trong module sklearn.linear\_model của thư viện scikit-learn. Đối tượng LinearRegression được sử dụng để thực hiện huấn luyện và dự đoán trong mô hình hồi quy tuyến tính.

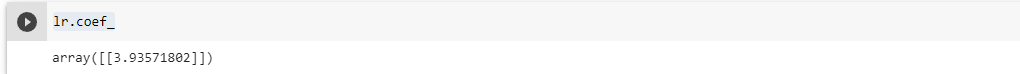


Câu lệnh lr.fit(X\_train, y\_train) được sử dụng để huấn luyện mô hình hồi quy tuyến tính (lr) trên tập dữ liệu huấn luyện (X\_train, y\_train).

Khi bạn gọi fit() trên một đối tượng LinearRegression, nó thực hiện quá trình tối ưu hóa mô hình hồi quy tuyến tính bằng cách điều chỉnh các hệ số và sai số của mô hình để tìm ra một phương trình tuyến tính phù hợp nhất với dữ liệu huấn luyện.

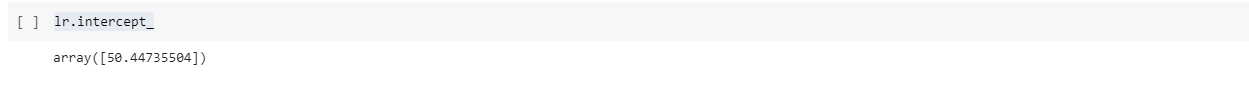
Trong quá trình huấn luyện, mô hình hồi quy tuyến tính cố gắng tìm ra các hệ số sao cho tổng bình phương sai số giữa giá trị dự đoán và giá trị thực tế trong tập huấn luyện (X\_train, y\_train) là nhỏ nhất. Quá trình này sử dụng phương pháp tối ưu hóa, chẳng hạn như phương pháp bình phương tối tiểu (least squares), để đạt được mục tiêu này.

Sau khi quá trình huấn luyện hoàn tất, mô hình hồi quy tuyến tính (lr) sẽ có các hệ số và sai số đã được điều chỉnh dựa trên dữ liệu huấn luyện. Mô hình này sau đó có thể được sử dụng để dự đoán giá trị cho dữ liệu mới bằng cách sử dụng phương thức predict().



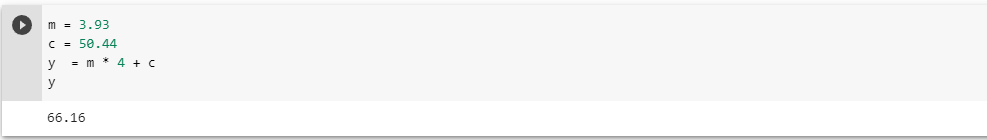
lr.coef\_ là một thuộc tính của đối tượng LinearRegression trong scikit-learn, được sử dụng để truy cập các hệ số của mô hình hồi quy tuyến tính sau khi đã được huấn luyện.

Thuộc tính coef\_ trả về một mảng 1 chiều chứa các hệ số tương ứng với các đặc trưng (features) trong mô hình hồi quy tuyến tính. Các hệ số này biểu diễn mức độ ảnh hưởng của từng đặc trưng đến biến phụ thuộc (target variable) trong mô hình.



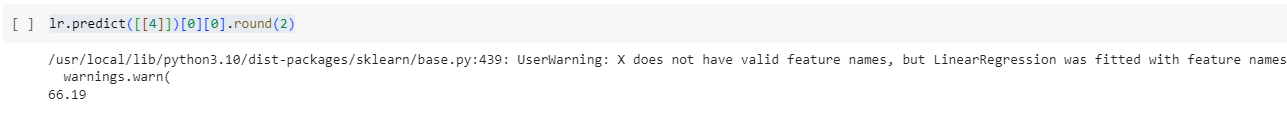
Câu lệnh lr.intercept\_ là một thuộc tính của đối tượng LinearRegression trong scikit-learn. Nó được sử dụng để truy cập hệ số góc (intercept) của mô hình hồi quy tuyến tính sau khi đã được huấn luyện.

Hệ số góc là hệ số độc lập (intercept) trong phương trình hồi quy tuyến tính, đại diện cho giá trị dự đoán của biến phụ thuộc khi tất cả các đặc trưng đều bằng 0.



Câu lệnh y = m \* 4 + c được sử dụng để tính toán giá trị của biến y dựa trên các giá trị m, c và 4.

Trong công thức này, m là hệ số góc (slope) và c là hệ số giao (intercept) trong một mô hình tuyến tính. Cụ thể, m đại diện cho độ dốc của đường thẳng và c đại diện cho điểm cắt trục y (giá trị của y khi x = 0).

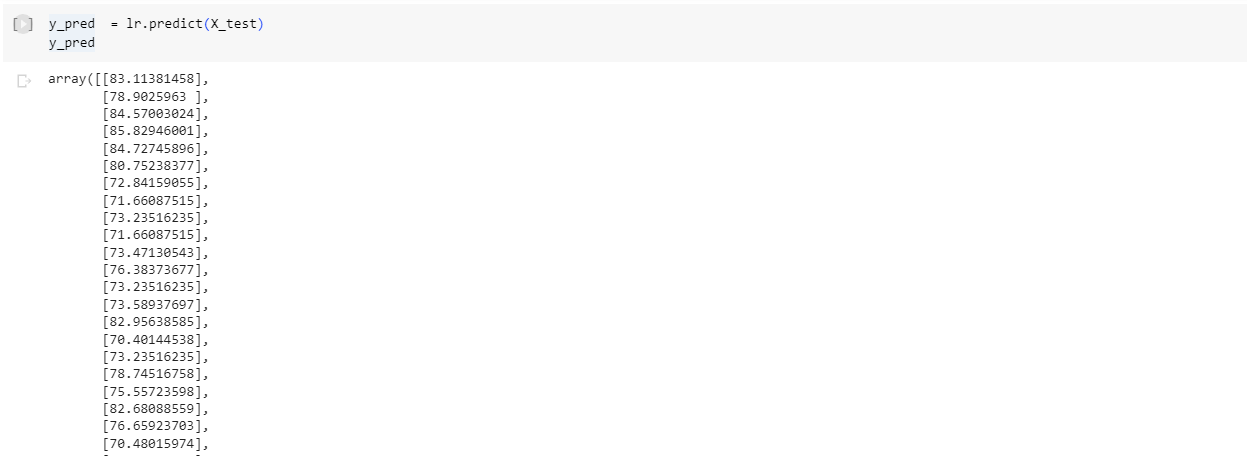


Câu lệnh lr.predict([[4]])[0][0].round(2) được sử dụng để thực hiện dự đoán giá trị của biến phụ thuộc (y) dựa trên mô hình hồi quy tuyến tính (lr) đã được huấn luyện trước đó.

Trong câu lệnh này, lr.predict([[4]]) được sử dụng để dự đoán giá trị của biến phụ thuộc (y) dựa trên giá trị đặc trưng (x) là 4. Điều này có nghĩa là ta đang cung cấp giá trị 4 làm đầu vào và mô hình hồi quy tuyến tính sẽ dự đoán giá trị của biến phụ thuộc tương ứng.

Kết quả trả về từ câu lệnh lr.predict([[4]]) là một mảng 2 chiều, có dạng [[predicted\_value]]. Do đó, để truy cập giá trị dự đoán, chúng ta sử dụng [0][0].

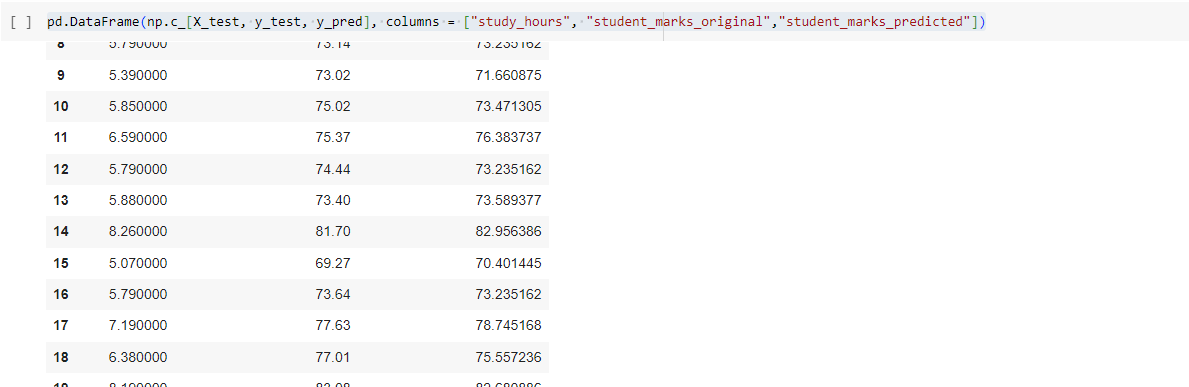
Cuối cùng, .round(2) được sử dụng để làm tròn giá trị dự đoán đến 2 chữ số thập phân. Câu lệnh lr.predict([[4]])[0][0].round(2) giúp ta dự đoán giá trị của biến phụ thuộc (y) dựa trên mô hình hồi quy tuyến tính đã được huấn luyện và đầu vào là giá trị đặc trưng (x) là 4.



Câu lệnh y\_pred = lr.predict(X\_test) được sử dụng để dự đoán giá trị của biến phụ thuộc (y) trên tập dữ liệu kiểm thử (X\_test) bằng mô hình hồi quy tuyến tính đã được huấn luyện (lr).

Trong câu lệnh này, X\_test là tập dữ liệu đặc trưng của tập kiểm thử, và lr.predict(X\_test) được sử dụng để dự đoán giá trị của biến phụ thuộc tương ứng.

Kết quả trả về từ câu lệnh lr.predict(X\_test) là một mảng chứa các giá trị dự đoán cho biến phụ thuộc trên tập kiểm thử. Các giá trị này tương ứng với các điểm dữ liệu trong tập kiểm thử X\_test.

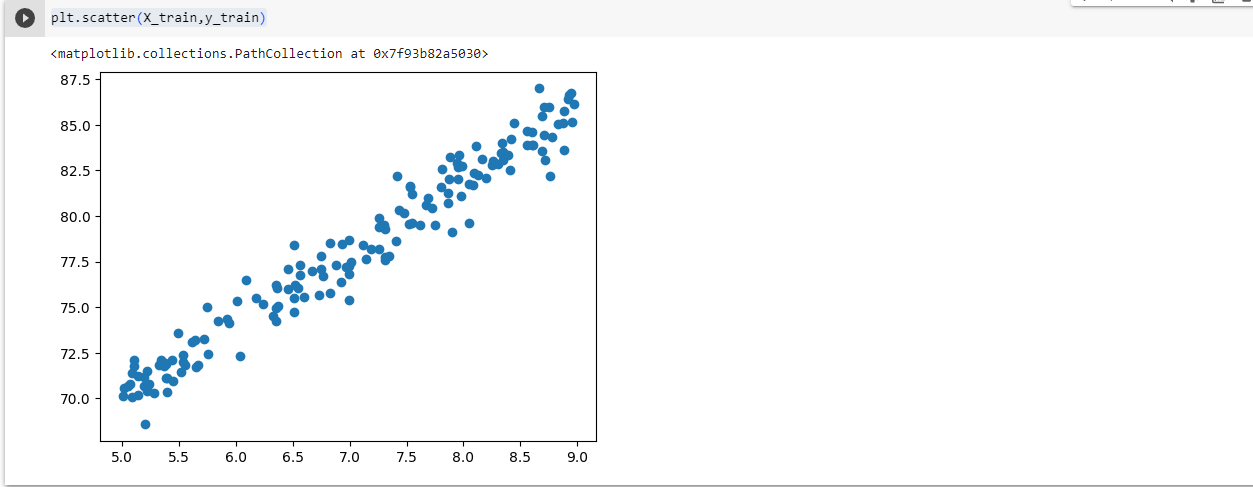


Câu lệnh pd.DataFrame(np.c\_[X\_test, y\_test, y\_pred], columns=["study\_hours", "student\_marks\_original", "student\_marks\_predicted"]) được sử dụng để tạo một DataFrame (bảng dữ liệu) để hiển thị các giá trị đặc trưng (X\_test), giá trị thực tế của biến phụ thuộc (y\_test) và giá trị dự đoán của biến phụ thuộc (y\_pred).

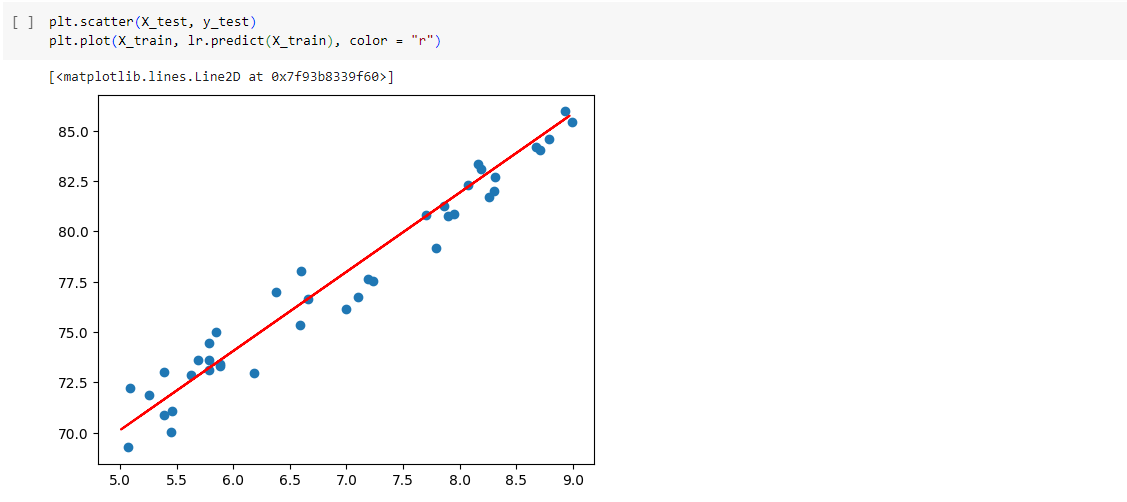
np.c\_[X\_test, y\_test, y\_pred] được sử dụng để ghép các mảng X\_test, y\_test và y\_pred cùng chiều theo cột để tạo thành một mảng 2D mới. Điều này cho phép các giá trị đặc trưng, giá trị thực tế và giá trị dự đoán được sắp xếp cùng hàng trong mảng mới.

columns=["study\_hours", "student\_marks\_original", "student\_marks\_predicted"] được sử dụng để đặt tên cho các cột trong DataFrame tương ứng với các thông tin tương ứng.

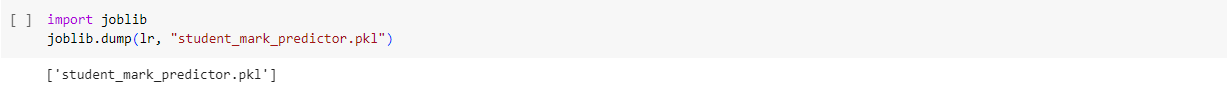
Kết quả của câu lệnh này là một DataFrame có các cột là "study\_hours" (giá trị đặc trưng), "student\_marks\_original" (giá trị thực tế của biến phụ thuộc) và "student\_marks\_predicted" (giá trị dự đoán của biến phụ thuộc). Các hàng của DataFrame tương ứng với các điểm dữ liệu trong tập kiểm thử, trong đó mỗi hàng chứa các giá trị tương ứng với đặc trưng, giá trị thực tế và giá trị dự đoán.



Câu lệnh plt.scatter(X\_train, y\_train) được sử dụng để tạo biểu đồ scatter plot (biểu đồ phân tán) giữa các giá trị đặc trưng (X\_train) và giá trị thực tế của biến phụ thuộc (y\_train).

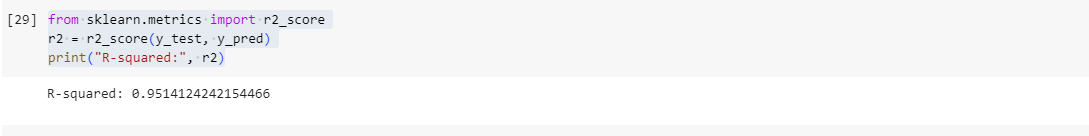


Câu lệnh plt.scatter(X\_test, y\_test) được sử dụng để vẽ biểu đồ scatter plot (biểu đồ phân tán) của các giá trị đặc trưng trong tập kiểm thử (X\_test) và giá trị thực tế của biến phụ thuộc (y\_test).



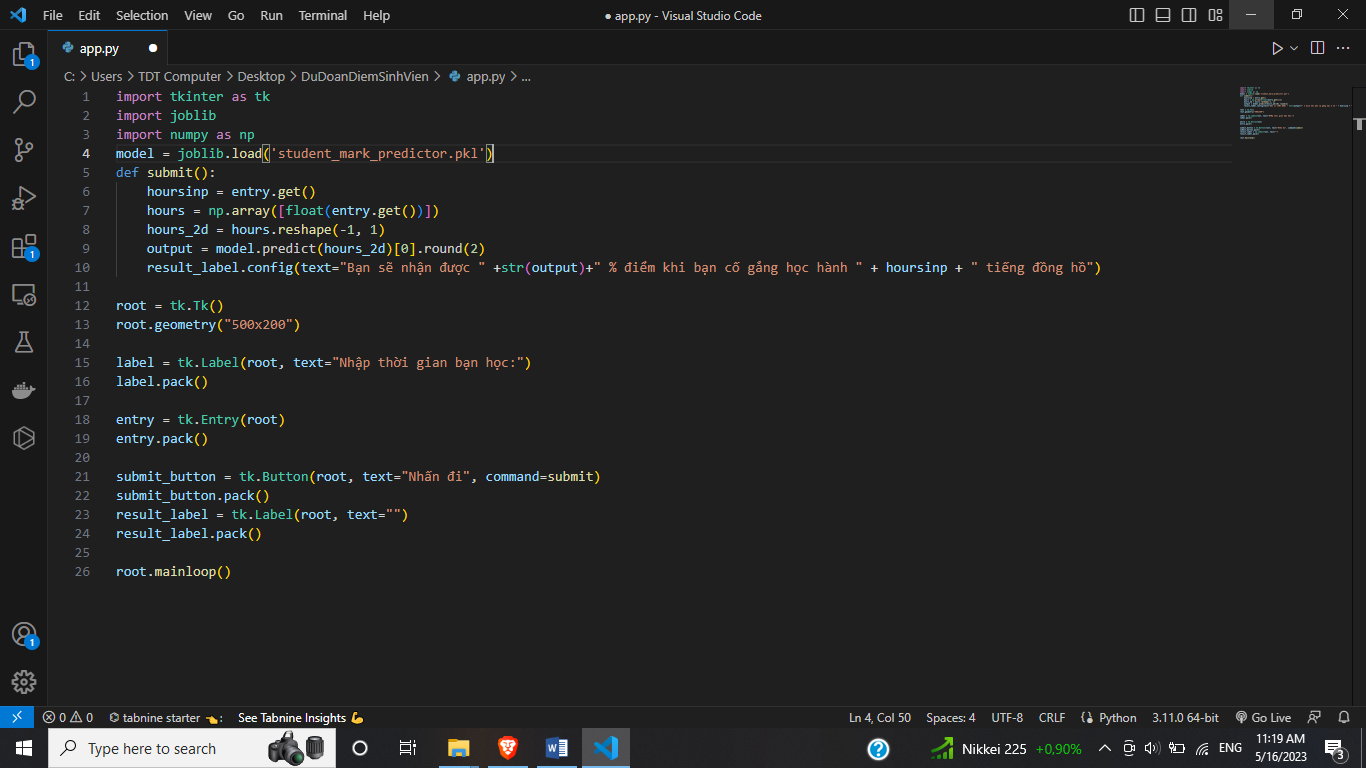
Câu lệnh joblib.dump(lr, "student\_mark\_predictor.pkl") được sử dụng để lưu trữ (serialize) mô hình hồi quy tuyến tính (lr) vào một tệp có tên là "student\_mark\_predictor.pkl" bằng cách sử dụng module joblib trong thư viện scikit-learn.

Tính toán độ chính xác:



Độ chính xác: 95%

Sau khi tải xong file student\_mark\_predictor.pkl, ta chỉ việc gọi file này trong giao application để sử dụng mô hình dữ đoán.



**Tài liệu tham khảo**

1. Machine learning project| | Student exam mark prediction using python| | AK |

<https://www.youtube.com/watch?v=lN8AsMRiKP4>

2. https://chat.openai.com/