**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

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

**KHOA CÔNG NGHỆ THÔNG TIN**

**- - - 🙞 🕮** **🙜 - - -**

Icon

Description automatically generated

**BÁO CÁO**

MÔN: **NHẬP MÔN KHOA HỌC DỮ LIỆU**

CÂU 1: **Trình bày và xử lý bài toán Hồi quy tuyến tính dựa vào học máy bằng ngôn ngữ Pyhon**

**Giảng viên hướng dẫn: TS. Tân Hạnh**

**Sinh viên thực hiện:**

Nguyễn Trần Đức Thuận – N19DCCN203

**TP. HCM, 20/06/2023**

**HỒI QUY TUYẾN TÍNH**

## **Lý thuyết bài toán Hồi Quy Tuyến Tính:**

Bài toán hồi quy tuyến tính là bài toán tìm ra một phương trình tuyến tính dựa trên dữ liệu, để có thể dự đoán giá trị của biến phụ thuộc khi biến độc lập thay đổi.

Cụ thể, bài toán hồi quy tuyến tính sử dụng một tập hợp các giá trị của biến độc lập (x) và biến phụ thuộc (y) để tìm ra một phương trình tuyến tính có dạng:

y = βx+ α + ε

Trong đó:

* β là hệ số góc của đường hồi quy, cho biết mức độ tương quan giữa biến độc lập và biến phụ thuộc. Nó cho biết một đơn vị thay đổi trong biến độc lập sẽ dẫn đến thay đổi độ lớn bao nhiêu trong biến phụ thuộc.
* α là hệ số chặn của đường hồi quy, cho biết giá trị của biến phụ thuộc khi biến độc lập bằng 0.
* ε là sai số ngẫu nhiên, cho biết sự khác biệt giữa giá trị thực tế của biến phụ thuộc và giá trị được dự đoán bởi mô hình hồi quy.

Ví dụ:

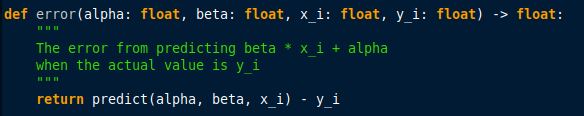
Giả sử ta đang điều tra mối quan hệ giữa số lượng bạn bè của người dùng DataSciencester và thời gian mà người dùng dành trên trang web mỗi ngày. Giả định rằng có nhiều bạn bè sẽ khiến cho người dùng dành nhiều thời gian hơn trên trang web.  
Bạn đươc yêu cầu xây dựng một mô hình mô tả mối quan hệ này. Và bạn đã tìm thấy mối quan hệ tuyến tính khá mạnh để bắt đầu là một mô hình tuyến tính.  
Bạn giả thuyết rằng có các hằng số α (alpha) và β (beta) sao cho:  
yi = βxi + α + εi  
trong đó y là số phút mà người dùng i dành trên trang web hàng ngày, x là số bạn bè mà người dùng i có, và ε là một thuật ngữ sai số đại diện cho việc có những yếu tố khác không được tính trong mô hình đơn giản này.

## **Xử lý bài toán Hồi quy tuyến tính dựa vào học máy bằng ngôn ngữ Pyhon:**



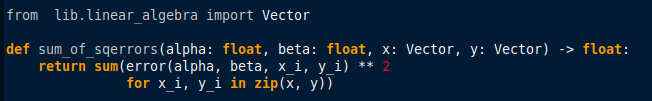
Hàm **predict** nhận ba đối số đầu vào là **alpha**, **beta** và **x\_i**, và trả về giá trị của biểu thức **beta \* x\_i + alpha**.

Hàm này là một mô hình hồi quy tuyến tính đơn giản, trong đó **alpha** là hệ số chặn (intercept) và **beta** là hệ số góc (slope) của đường thẳng hồi quy. Khi có một giá trị đầu vào **x\_i**, hàm **predict** sẽ tính toán giá trị dự đoán của biến phụ thuộc dựa trên đường thẳng hồi quy được xác định bởi **alpha** và **beta**.



Hàm **error** nhận vào bốn đối số đầu vào là **alpha**, **beta**, **x\_i** và **y\_i**, và trả về giá trị sai số **(error)** giữa giá trị dự đoán và giá trị thực tế.

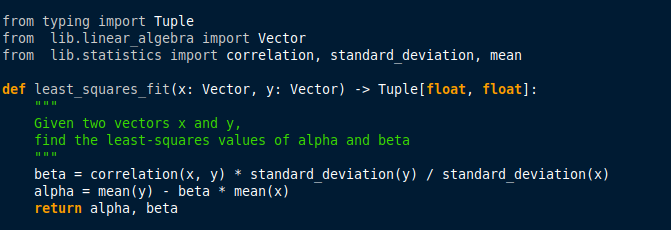
Hàm này dựa trên hàm **predict** đã được định nghĩa trước đó để tính toán giá trị dự đoán **predict(alpha, beta, x\_i)** và trừ giá trị thực tế **y\_i** để tính toán sai số. Khi mô hình dự đoán giá trị **predict(alpha, beta, x\_i)** gần với giá trị thực tế **y\_i**, thì giá trị của hàm **error** sẽ gần với 0. Tuy nhiên, khi mô hình dự đoán không chính xác, thì giá trị của hàm **error** sẽ có giá trị tương ứng với mức độ sai số của mô hình.



Hàm **sum\_of\_sqerrors** nhận vào bốn đối số đầu vào là **alpha**, **beta**, **x**, và **y**, trong đó **x** và **y** là các vector chứa các giá trị đầu vào và giá trị đầu ra tương ứng. Hàm này tính tổng bình phương của các sai số giữa giá trị dự đoán và giá trị thực tế cho các cặp **(x\_i, y\_i)** trong các vector **x** và **y**.

Để tính tổng này, hàm sử dụng hàm **error** đã được định nghĩa trước đó để tính toán sai số giữa giá trị dự đoán và giá trị thực tế cho từng cặp **(x\_i, y\_i)** trong các vector **x** và **y**. Sau đó, hàm tính tổng bình phương của các sai số này bằng cách sử dụng hàm sum trên một danh sách các giá trị bình phương của sai số.

Hàm này có thể được sử dụng để tính toán tổng sai số bình phương của mô hình hồi quy tuyến tính, trong đó mục tiêu là tìm giá trị **alpha** và **beta** sao cho tổng sai số bình phương là nhỏ nhất.

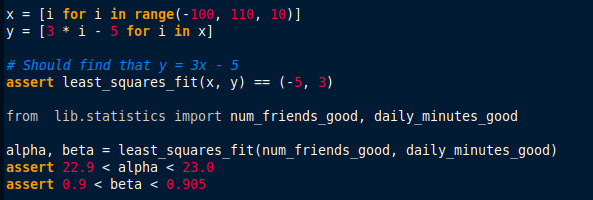


Hàm **least\_squares\_fit** nhận vào hai đối số đầu vào là **x** và **y**, là các vector chứa các giá trị đầu vào và đầu ra tương ứng, và trả về một tuple chứa giá trị **alpha** và **beta** tương ứng với giá trị hệ số chặn và hệ số góc của đường thẳng hồi quy tuyến tính được tính bằng phương pháp bình phương tối thiểu (least squares).

Để tính giá trị **beta**, hàm sử dụng hàm **correlation** và **standard\_deviation** để tính toán hệ số tương quan giữa **x** và **y** và độ lệch chuẩn của **y** và **x**. Sau đó, giá trị **beta** sẽ được tính bằng tích của hệ số tương quan và độ lệch chuẩn của **y**, chia cho độ lệch chuẩn của **x**.

Để tính giá trị **alpha**, hàm sử dụng giá trị trung bình của **y** và **x** đã tính bằng hàm **mean**, và giá trị **beta** đã tính được ở trên.

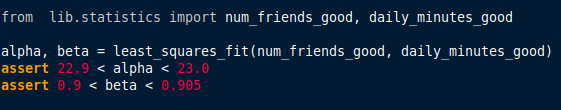
Hàm này có thể được sử dụng để tìm giá trị **alpha** và **beta** tối ưu cho mô hình hồi quy tuyến tính, với mục tiêu là giảm thiểu tổng sai số bình phương giữa giá trị dự đoán và giá trị thực tế.



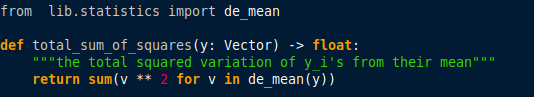
Mã này kiểm tra tính đúng đắn của hàm **least\_squares\_fit** bằng cách sử dụng các giá trị đầu vào và đầu ra được chỉ định cùng với một số giá trị mong đợi.

Hàm **least\_squares\_fit** được áp dụng cho các giá trị **x** và **y** được tạo ra bằng cách lặp qua các giá trị từ -100 đến 100 với bước nhảy là 10 và tính giá trị đầu ra bằng cách nhân giá trị **x** với 3, trừ đi 5. Kết quả được mong đợi của hàm **least\_squares\_fit** với các giá trị đầu vào này là (-5, 3), tương ứng với giá trị hệ số chặn và hệ số góc của đường thẳng hồi quy tuyến tính.

Áp dụng vào dữ liệu như sau:

  
Đoạn mã này sử dụng hàm **least\_squares\_fit** để tính toán giá trị **alpha** và **beta** cho hai vector **num\_friends\_good** và **daily\_minutes\_good**, biểu thị số lượng bạn bè và thời gian trung bình hàng ngày.

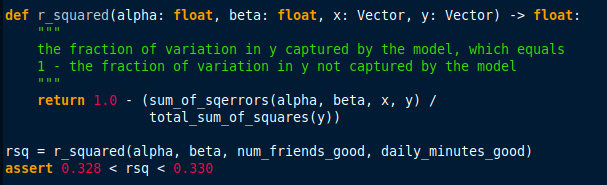
Sau khi tính được giá trị **alpha** và **beta**, đoạn mã sử dụng các lệnh **assert** để kiểm tra rằng giá trị **alpha** nằm trong khoảng từ 22.9 đến 23.0 và giá trị **beta** nằm trong khoảng từ 0.9 đến 0.905. Nếu các giá trị này không nằm trong khoảng mong đợi, đoạn mã sẽ tạo ra một **AssertionError** và báo lỗi.



Hàm **total\_sum\_of\_squares** tính tổng các bình phương độ lệch giữa từng giá trị trong vector **y** và giá trị trung bình của vector **y**. Đây còn được gọi là tổng bình phương phương sai **(total sum of squares)**.

Hàm sử dụng hàm **de\_mean** để chuẩn hoá giá trị trong vector **y** bằng cách trừ giá trị trung bình của vector **y** khỏi từng giá trị trong vector. Sau đó, hàm sử dụng hàm **sum** để tính tổng các bình phương của các giá trị đã được chuẩn hoá này.

Hàm này thường được sử dụng trong việc tính toán hệ số xác định (coefficient of determination) của mô hình hồi quy tuyến tính, là một phương pháp để đánh giá mức độ phù hợp của mô hình đối với dữ liệu. Hệ số xác định được tính bằng tổng bình phương phương sai giữa giá trị thực tế và giá trị trung bình của vector **y** (tổng bình phương sai của mô hình không có biến giải thích), chia cho tổng bình phương phương sai giữa giá trị thực tế và giá trị trung bình của vector **y** (tổng bình phương sai của mô hình tối đa). Hệ số xác định này có giá trị từ 0 đến 1, với giá trị càng gần 1 thì mô hình càng phù hợp với dữ liệu.

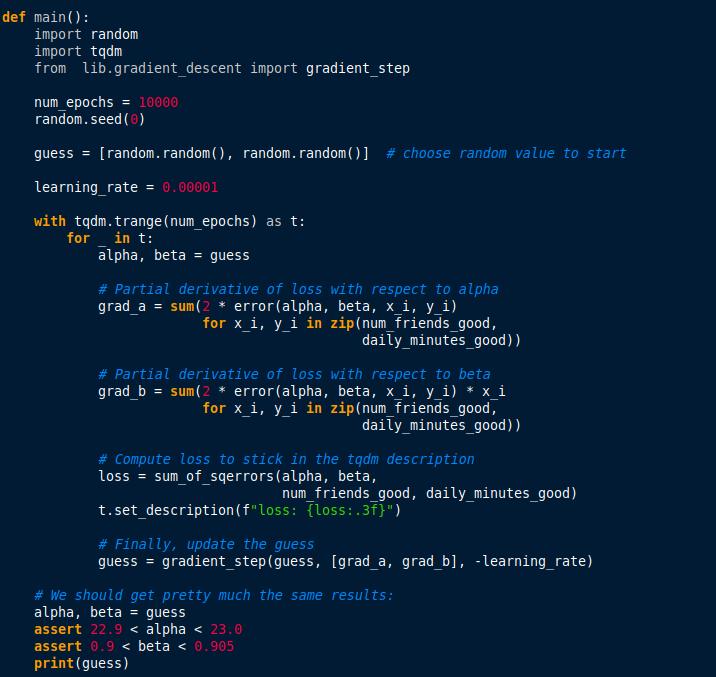


Hàm **r\_squared** tính toán hệ số xác định (coefficient of determination) của mô hình hồi quy tuyến tính, biểu thị mức độ phù hợp của mô hình với dữ liệu. Hệ số xác định được tính bằng tỷ lệ giữa phần trăm bình phương phương sai đã được giải thích bởi mô hình.

Để tính toán hệ số xác định, hàm sử dụng hàm **sum\_of\_sqerrors** để tính toán tổng sai số bình phương giữa giá trị dự đoán và giá trị thực tế cho các cặp **(x\_i, y\_i)** trong các vector **x** và **y**. Sau đó, hàm sử dụng hàm **total\_sum\_of\_squares** để tính tổng bình phương phương sai của giá trị thực tế so với giá trị trung bình của **y**. Cuối cùng, hàm trả về giá trị 1 trừ đi tỷ lệ giữa tổng sai số bình phương không được giải thích bởi mô hình và tổng bình phương phương sai tối đa.

Đoạn mã này sử dụng hàm **r\_squared** để tính toán hệ số xác định cho mô hình hồi quy tuyến tính được tính toán trước đó với vector **num\_friends\_good** và **daily\_minutes\_good**. Sau đó, đoạn mã sử dụng các lệnh **assert** để kiểm tra rằng giá trị hệ số xác định nằm trong khoảng từ 0.328 đến 0.330.

**2.** **Phương pháp sử dụng Gradient Descent:**



Hàm mainbao gồm một số đoạn mã để thực hiện việc huấn luyện mô hình hồi quy tuyến tính trên các dữ liệu về số lượng bạn bè và thời gianlướt web trung bình hàng ngày.

Đầu tiên, hàm khởi tạo một số tham số cho quá trình huấn luyện, bao gồm số lượng vòng lặp **(epochs)**, giá trị ban đầu của các tham số **alpha** và **beta** (được chọn ngẫu nhiên), và tốc độ học (learning rate).

Sau đó, hàm sử dụng một vòng lặp để thực hiện huấn luyện. Trong mỗi vòng lặp, hàm tính toán đạo hàm riêng của hàm mất mát theo **alpha** và **beta**, sử dụng các giá trị hiện tại của **alpha** và **beta** và dữ liệu huấn luyện. Sau đó, hàm cập nhật giá trị của **alpha** và **beta** bằng cách sử dụng hàm **gradient\_step**, một hàm được định nghĩa trong tệp **gradient\_descent.py**.

Cuối cùng, hàm kiểm tra rằng giá trị của **alpha** và **beta** đã được cập nhật có giá trị nằm trong khoảng mong đợi, và in ra kết quả của **alpha** và **beta**. Nếu các giá trị này không nằm trong khoảng mong đợi, hàm sẽ tạo ra một **AssertionError** và báo lỗi.