**Phòng thí nghiệm số 1.**

**Phần 1. Hồi quy tuyến tính.**

Mục tiêu: củng cố kiến thức lý thuyết về hồi quy tuyến tính, rèn luyện kỹ năng thực hành xây dựng mô hình hồi quy tuyến tính.

Nhiệm vụ:

* Ghi nhớ và hiểu các khái niệm cơ bản của mô hình hồi quy tuyến tính;
* Tìm hiểu cách xây dựng mô hình hồi quy tuyến tính bằng phương pháp bình phương tối thiểu;
* Tìm hiểu cách sử dụng thư viện statsmodels ;
* Tìm hiểu cách xây dựng mô hình hồi quy tuyến tính bằng phương pháp giảm độ dốc;
* Tìm hiểu cách sử dụng thư viện scikit - tìm hiểu .

**Hồi quy tuyến tính. Ý tưởng**

Các hiện tượng kinh tế, như một quy luật, được xác định bởi một số lượng lớn các yếu tố tác động đồng thời và tích lũy. Về vấn đề này, nhiệm vụ nghiên cứu sự phụ thuộc của một biến phụ thuộc Y vào một số biến giải thích *X 1 , X 2 , ,X m thường phát sinh* . Vấn đề này được giải quyết bằng cách sử dụng ***phân tích hồi quy bội.***

Cho có một vectơ giá trị quan sát của m biến giải thích . Khi đó vectơ giá trị quan sát của biến giải thích Y có dạng: *y 1 , y 2 ,..., y p* và mô hình hồi quy tuyến tính có thể được biểu diễn dưới dạng chuẩn như sau:

*y tôi =* *β1 x i1 + β2 x i2 + β3 x i3 +…+ βm x im + u i , i = 1,…,p* (1)

Chúng ta giả sử *x i 1 = 1* , tức là *β1* - số hạng tự do, - sai số ngẫu nhiên, là sự khác biệt giữa giá trị thực và giá trị dự đoán.

Phương pháp bình phương tối thiểu

Ước tính bình phương nhỏ nhất là một vectơ gồm các tham số *( β1 , β2 ,… , βm )* giúp giảm thiểu tổng độ lệch bình phương của các giá trị quan sát được so với các giá trị được báo cáo, tức là.

.

Phương trình (1 ) có thể được viết dưới dạng ma trận như sau:

*Y = XB + U* ( 3),

Ở đâu

(4 ), (5), (6), (7)

Khi đó phương trình (2) ở dạng ma trận sẽ có dạng như sau:

*S = U T U = (Y - XB) T (Y - XB)*

Giải pháp của phương trình này là vectơ:



Đạo hàm của vectơ này đã được thảo luận chi tiết tại bài giảng.

thiểu trong python

Thư viện NumPy

NumPy là một thư viện dành cho ngôn ngữ lập trình Python cung cấp hỗ trợ để làm việc với các mảng và ma trận đa chiều, đồng thời bao gồm một số lượng lớn các hàm toán học để thực hiện các phép toán trên các mảng này. Đây là một trong những thư viện cốt lõi cho tính toán khoa học bằng Python và được sử dụng rộng rãi trong các lĩnh vực phân tích dữ liệu, học máy và tính toán số. np.array là một hàm được cung cấp bởi thư viện NumPy, được sử dụng để tạo các đối tượng ndarray. Nó lấy một chuỗi (chẳng hạn như danh sách hoặc bộ dữ liệu) làm đối số và trả về một mảng NumPy mới.

Ví dụ:

nhập khẩu có khối u BẰNG n.p.

#Tạo mảng một chiều

mảng\_1d = np . mảng ([ 1 , 2 , 3 , 4 , 5 ])

print ( "Mảng một chiều:" , array\_1d )

#Tạo mảng hai chiều

mảng\_2d = np . mảng ([[ 1 , 2 , 3 ], [ 4 , 5 , 6 ]])

in ( " Hai chiều mảng : \n " , mảng\_2d )

Các tính năng chính của NumPy :

* Mảng: NumPy cung cấp đối tượng ndarray , là một mảng đa chiều có kích thước cố định chứa các phần tử cùng loại. Điều này cho phép bạn lưu trữ và xử lý dữ liệu một cách hiệu quả. Để tạo một mảng
* Hoạt động mảng: NumPy hỗ trợ các hoạt động được vector hóa, cho phép bạn thực hiện các phép toán trên mảng mà không cần vòng lặp.
* Các hàm dành cho Đại số tuyến tính: NumPy bao gồm các hàm để thực hiện các phép toán đại số tuyến tính như nhân ma trận, tính định thức và giá trị riêng.
* Các hàm thống kê: Thư viện cung cấp các hàm tính toán các số liệu thống kê khác nhau như giá trị trung bình, trung vị, độ lệch chuẩn, v.v.
* Các chức năng thuận tiện để làm việc với dữ liệu: NumPy bao gồm các chức năng tạo số ngẫu nhiên, làm việc với các khoảng và chuyển đổi dữ liệu.

Để cài đặt thư viện, bạn cần viết vào bảng điều khiển:

pip cài đặt có khối u

Sau khi cài đặt, để sử dụng thư viện này, bạn phải nhập thư viện này:

nhập khẩu có khối u BẰNG n.p.

Chúng ta hãy xem việc sử dụng thư viện NumPy để xây dựng mô hình hồi quy tuyến tính bằng phương pháp bình phương tối thiểu.

1. Ở giai đoạn đầu tiên, chúng tôi sẽ tạo dữ liệu ban đầu:

np . ngẫu nhiên . hạt giống ( 0 ) # Để tái sản xuất

n\_mẫu = 100

n\_features = 3

# Tạo dữ liệu ngẫu nhiên

X = np . ngẫu nhiên . rand ( n\_samples , n\_features )

# Tạo hệ số cho mô hình tuyến tính

hệ số true = np . mảng ([ 1.5 , - 2.0 , 3.0 ])

# Tạo biến mục tiêu có thêm nhiễu

y = X @ hệ số true + np . ngẫu nhiên . bình thường ( 0 , 0.1 , n\_samples )

# Thêm một cột đơn vị cho thuật ngữ miễn phí (chặn)

X\_b = np . c\_ [ np . những cái (( n\_samples , 1 )), X ] # Thêm một cột gồm những cái

hàm np . ngẫu nhiên . Seed () trong thư viện NumPy được sử dụng để khởi tạo trình tạo số ngẫu nhiên. Cài đặt hạt giống cho phép bạn nhận được kết quả có thể lặp lại khi tạo số ngẫu nhiên. Điều này đặc biệt hữu ích trong nghiên cứu khoa học và phát triển thuật toán khi bạn cần lặp lại các thí nghiệm với cùng một dữ liệu ngẫu nhiên.

khi bạn đặt hạt giống bằng cách sử dụng np . ngẫu nhiên . hạt giống (0), bạn nắm bắt được trạng thái ban đầu của trình tạo số ngẫu nhiên. Điều này có nghĩa là mỗi khi bạn tạo các số ngẫu nhiên sau khi cài đặt tùy chọn này, bạn sẽ nhận được các số giống nhau.

np.random .rand(n\_samples, n\_features) được sử dụng để tạo mảng hai chiều (ma trận) gồm các số ngẫu nhiên được phân bố đồng đều trong phạm vi từ 0 đến 1.

Thông số:

n\_samples: số hàng trong mảng, tương ứng với số lượng đối tượng (hoặc quan sát).

n\_features: Số lượng cột trong mảng, tương ứng với số lượng tính năng (hoặc biến) cho mỗi tính năng.

Với hai dòng cuối cùng, chúng tôi tạo dữ liệu cho biến phụ thuộc y sao cho nó phụ thuộc tuyến tính vào X (hệ số phụ thuộc tuyến tính được đặt thành true\_cofactors, nhưng chúng tôi cũng thêm nhiễu phân phối chuẩn bằng cách sử dụng np.random .normal(0, 0.1 , n\_mẫu).

Trong NumPy, toán tử @ được sử dụng để thực hiện phép nhân ma trận. Nó được giới thiệu trong Python 3.5 và cho phép bạn thực hiện các phép nhân ma trận thuận tiện và dễ đọc hơn so với sử dụng hàm np.dot( ) hoặc phương thức .dot(), cũng cho phép bạn thực hiện phép nhân ma trận.

Để tính độ dịch chuyển (thuật ngữ tự do *β0* ) được tính đến trong phương trình của chúng ta, chúng ta thêm vào ma trận X một vectơ có độ dài X , bao gồm

1. Hãy triển khai phương pháp bình phương tối thiểu bằng thư viện NumPy :

#Phương pháp bình phương tối thiểu

# theta = (X\_b^T \* X\_ b)^ (-1) \* X\_b^T \* y

theta\_best = np . linal . inv ( X\_b.T @ X\_b ) @ X\_b.T @ y

#In kết quả

print ( "Hệ số hồi quy:" )

in ( theta\_best )

Kết quả ta thu được các hệ số:

[-0. 00497861, 1.45913248 , -2.00200434, 3.0253231 ], rất gần với sự phụ thuộc ban đầu của chúng tôi [1.5, -2.0, 3.0].

Khi xây dựng mô hình hồi quy tuyến tính, điều quan trọng là phải biết trước bản chất của các mối phụ thuộc trong dữ liệu. Một cách thuận tiện để thực hiện việc này là sử dụng trực quan hóa bằng biểu đồ phân tán. Để vẽ biểu đồ phân tán trong python, thật thuận tiện khi sử dụng thư viện trực quan matplotlib . Để tạo biểu đồ phân tán cho ví dụ của chúng tôi, bạn có thể sử dụng mã sau:

nhập khẩu matplotlib . pyplot BẰNG làm ơn

làm ơn . phân tán ( X [:, 1 ], y )

làm ơn . trình diễn ()

làm ơn . phân tán ( X [:, 2 ], y )

làm ơn . trình diễn ()

làm ơn . tán xạ ( X [:, 3 ], y )

làm ơn . trình diễn ()

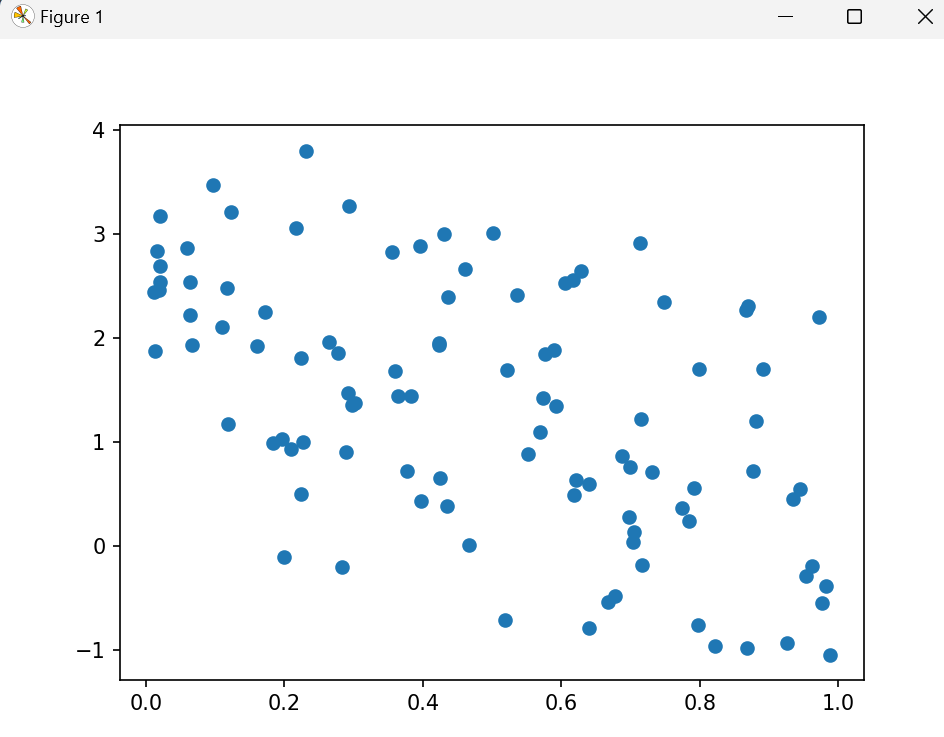
NumPy hai chiều được truy cập thông qua việc sử dụng các chỉ mục. Chỉ số đầu tiên có nghĩa là số hàng, chỉ số thứ hai là số cột.

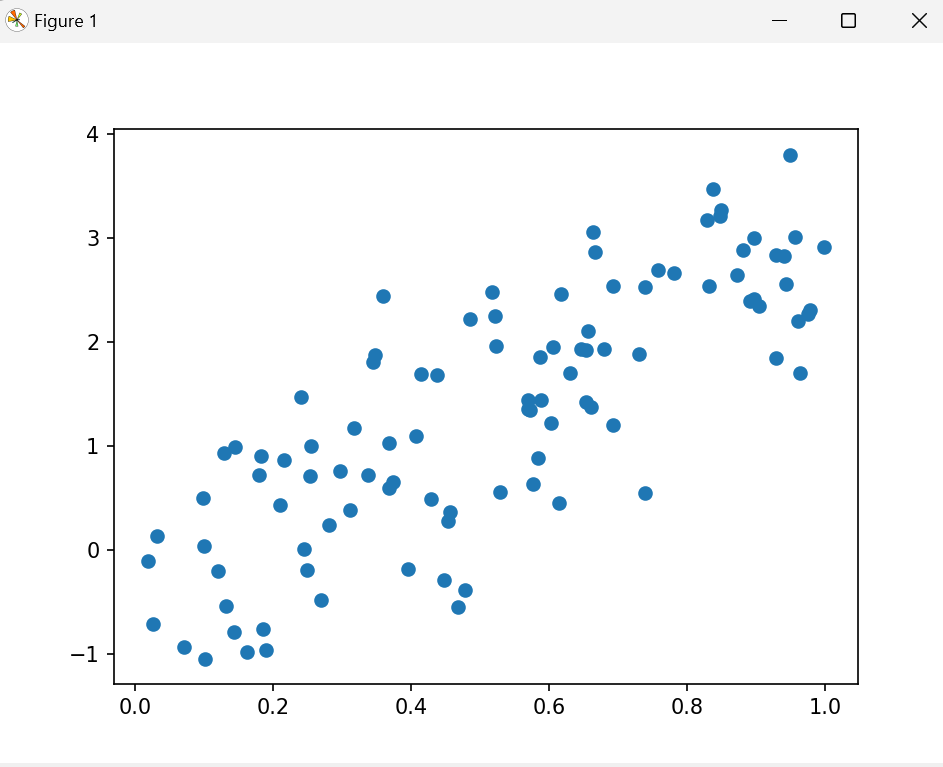
Nếu bạn muốn cắt theo cột hoặc hàng thì cú pháp là:

[ bắt đầu : dừng lại ]. Nếu bạn muốn tạo một lát cắt bắt đầu từ phần tử 0 (phần tử ở chỉ số 0), bạn không cần phải viết số 0. Tương tự, nếu bạn muốn tạo một lát cắt từ phần tử nào đó đến phần tử cuối cùng, bạn cũng không cần phải viết chỉ mục của phần tử cuối cùng.

Изображение выглядит как снимок экрана, диаграмма

Автоматически созданное описание





**Nhiệm vụ 1**

1. Nhập thư viện NumPy . Sử dụng thư viện này, tải tệp tuyến tính . csv và xuất ra bảng điều khiển. Để tải tệp vào mảng NumPy , hãy sử dụng hàm np.genfromtxt , theo sau là tên tệp trong ngoặc đơn. Tệp đã tải xuống phải nằm trong cùng thư mục với tập lệnh thực thi.

# Đang tải dữ liệu từ tệp CSV

dữ liệu = np . genfromtxt ( 'data.csv' , delimiter = ',' , Skip\_header = 1 )

tham số hàm np . genfromtxt ():

fname : tên file hoặc đường dẫn file. Trong ví dụ 'data.csv'

delimiter : ký tự dùng để phân tách các giá trị (mặc định là khoảng trắng).

Skip\_header : Số dòng cần bỏ qua ở đầu file (ví dụ: bỏ qua tiêu đề) .

dtype : kiểu dữ liệu cho mảng (mặc định float ).

1. Chia dữ liệu thành các biến phụ thuộc và độc lập bằng cách sử dụng các lát cắt theo cú pháp sau:

X = dữ liệu [:, : - 1 ]

y = dữ liệu [:, - 1 ]

chỉ số âm có nghĩa là phần tử cuối cùng .

1. Đối với dữ liệu đã tải, hãy xây dựng biểu đồ phân tán của y so với mỗi x . Rút ra kết luận về bản chất của sự phụ thuộc.
2. Thực hiện thuật toán hồi quy tuyến tính đã thảo luận ở trên. In kết quả ra bàn điều khiển.

**Sử dụng thư viện mô hình thống kê**

statsmodels là một thư viện dành cho Python cung cấp các lớp và hàm để ước tính mô hình thống kê, thực hiện kiểm tra thống kê và phân tích dữ liệu. Nó đặc biệt hữu ích cho phân tích thống kê và kinh tế lượng và thường được sử dụng kết hợp với các thư viện như NumPy và Pandas.

Các tính năng chính của mô hình thống kê:

* Đánh giá mô hình: mô hình thống kê hỗ trợ nhiều mô hình thống kê khác nhau, bao gồm hồi quy tuyến tính và phi tuyến tính, chuỗi thời gian, mô hình sinh tồn và nhiều mô hình khác.
* Kiểm tra thống kê: Thư viện cung cấp các chức năng để thực hiện các kiểm tra thống kê khác nhau như kiểm tra tính quy tắc, kiểm tra sự bằng nhau của phương sai, v.v.
* Phân tích chuỗi thời gian: mô hình thống kê bao gồm các công cụ để phân tích chuỗi thời gian, bao gồm ARIMA, SARIMA và các mô hình khác.
* Diễn giải kết quả: Thư viện cung cấp các phương pháp thuận tiện để lấy bản tóm tắt và diễn giải kết quả mô hình.

Nếu bạn chưa cài đặt thư viện statsmodels, bạn có thể cài đặt nó bằng pip:

mô hình thống kê cài đặt pip

Để nhập thư viện, chúng tôi viết:

nhập khẩu các mô hình thống kê API BẰNG sm

Để triển khai mô hình hồi quy tuyến tính bằng thư viện này, hãy xem xét ví dụ sau:

#Tạo mô hình

người mẫu = sm . OLS ( y , X ) #OLS - phương pháp bình phương tối thiểu

kết quả = người mẫu . vừa vặn () # Vừa vặn người mẫu

in ( results.summary ( ) )

# Nhận dự báo

dự đoán = results .predict ( X ) # Dự đoán bằng cách sử dụng cùng một dữ liệu

# Hiển thị 5 dự đoán đầu tiên

print ( " 5 dự đoán đầu tiên :" )

in ( dự đoán [ : 5 ])

Với model = sm.OLS( y, X), bạn tạo một đối tượng mô hình hồi quy tuyến tính bằng cách sử dụng Bình phương tối thiểu thông thường (OLS) từ thư viện mô hình thống kê.

Thông số:

y: Đây là biến phụ thuộc (vectơ của các giá trị bạn muốn dự đoán).

X: Đây là ma trận các biến độc lập (các yếu tố bạn dùng để dự đoán y). Điều quan trọng là X phải bao gồm một hằng số (một thuật ngữ giả) nếu bạn muốn mô hình tính đến nó.

model.fit( ) là một phương pháp huấn luyện (tính toán trực tiếp các hệ số mong muốn bằng phương pháp bình phương tối thiểu).

results.summary () – hiển thị kết quả cuối cùng cho mô hình kết quả.

Kết quả.tóm tắt () bao gồm :

* **Thông tin chung về mô hình:**
  + Sở. Biến: Biến phụ thuộc (ví dụ Y).
  + Model: Loại mô hình (ví dụ OLS).
  + Phương pháp: Phương pháp ước lượng (thường là Bình phương tối thiểu).
  + Date: Ngày và giờ thực hiện.
  + Thời gian: Thời gian thực hiện.
* **Thống kê mô hình:**
  + R bình phương: Hệ số xác định cho thấy mức độ biến thiên của một biến phụ thuộc được giải thích bởi các biến độc lập.
  + Điều chỉnh. R bình phương: Hệ số xác định được điều chỉnh có tính đến số lượng biến độc lập trong mô hình.
  + Thống kê F: Thống kê F để kiểm tra tầm quan trọng của mô hình tổng thể.
  + Prob (F-statistic): giá trị p cho thống kê F.
* **Hệ số mô hình:**

Đối với mỗi biến độc lập và số hạng tự do (hằng số), thông tin sau được hiển thị:

* coef: Hệ số ước tính.
* std err: Sai số chuẩn của các hệ số.
* t: Các giá trị thống kê t để kiểm tra ý nghĩa của các hệ số.
* P>| t|: giá trị p để kiểm tra tầm quan trọng của các hệ số.
* [0,025 0,975]: Khoảng tin cậy cho các hệ số.
* **Thống kê số dư:**
  + Omnibus: Kiểm tra tính quy phạm của phần dư.
  + Prob( Omnibus): giá trị p cho bài kiểm tra Omnibus.
  + Skew: Hệ số độ lệch của phần dư.
  + Kurtosis: (Kurtosis) một thước đo thống kê đo lường hình dạng của phân phối dữ liệu và chỉ ra mức độ mà các điểm của nó khác với các điểm của phân phối chuẩn.
  + Durbin-Watson: Kiểm tra sự tự tương quan của phần dư.
  + Jarque-Bera (JB): Kiểm tra tính quy phạm của phần dư.
  + Prob( JB): giá trị p cho phép thử Jarke-Bera.

Để có được dự đoán, phương thức results.predict (X) được sử dụng.

**Nhiệm vụ 2**

1. Đối với dữ liệu được sử dụng ở bước trước, hãy xây dựng mô hình hồi quy tuyến tính bằng thư viện statsmodels .
2. Tải xuống tệp thử nghiệm . csv và xây dựng dự báo cho những điểm này. So sánh kết quả với số liệu thực tế. Đưa ra kết luận về chất lượng và độ chính xác của mô hình.

**Hồi quy tuyến tính sử dụng** phương pháp giảm dần độ dốc

Ngoài phương pháp bình phương tối thiểu, bạn có thể sử dụng phương pháp giảm độ dốc để tìm hệ số hồi quy tuyến tính.

Giảm độ dốc là một thuật toán tối ưu hóa lặp lại được sử dụng để giảm thiểu hàm mất mát. Trong bối cảnh hồi quy tuyến tính, nó được sử dụng để tìm các giá trị tối ưu của các hệ số mô hình nhằm giảm thiểu sự khác biệt giữa giá trị dự đoán và giá trị thực tế của biến phụ thuộc.

Các bước cơ bản của việc giảm độ dốc trong hồi quy tuyến tính:

1. Khởi tạo: Giá trị ban đầu của các hệ số (thường là ngẫu nhiên hoặc bằng 0).
2. Chọn hàm mất mát: Trong hồi quy tuyến tính, sai số bình phương trung bình gốc (MSE) thường được sử dụng:

đâu là giá trị thực tế và đâu là giá trị dự đoán

1. Tính toán độ dốc: Độ dốc của hàm mất mát trên mỗi hệ số cho thấy hướng mà các hệ số cần được thay đổi để giảm sai số.
2. Tính toán độ dốc: Độ dốc của hàm mất mát trên mỗi hệ số cho thấy hướng mà các hệ số cần được thay đổi để giảm sai số.
3. Cập nhật tỷ lệ: Tỷ lệ được cập nhật theo công thức sau:

trong đó là giá trị hiện tại của tham số mô hình thứ j , là độ dốc của hàm mất mát và là tốc độ học (kích thước bước).

1. Lặp lại: Bước 3 và 4 được lặp lại cho đến khi đạt được sự hội tụ (ví dụ: khi những thay đổi trong hàm mất mát trở nên rất nhỏ).

Hãy thử triển khai logic này trong python . Giả sử chúng ta có một số dữ liệu:

X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

y = [52, 74, 79, 95, 115, 110, 129, 126, 147, 146, 156, 184]

Đối với họ, cần phải triển khai mô hình hồi quy tuyến tính sử dụng phương pháp giảm độ dốc. Hãy thực hiện logic này bằng cách sử dụng các hàm:

1. Hãy viết hàm tính gradient cho độ dịch chuyển:

chắc chắn get\_gradient\_at\_b ( x , y , b , a ) :

  N = len ( x )

  khác biệt = 0

  vì Tôi TRONG phạm vi ( N ):

    x\_val = x [ tôi ]

    y\_val = y [ tôi ]

    khác biệt += ( y\_val - (( Một \* x\_val ) + b ))

  b\_gradient = - ( 2 / N ) \* khác biệt

  trở lại b\_gradient

Giải thích chức năng

Thông số:

* x: mảng các giá trị biến độc lập.
* y: mảng giá trị biến phụ thuộc.
* b: giá trị hiện tại của số hạng tự do (hằng số) trong mô hình.
* a : Giá trị hiện tại của hệ số độ dốc (độ dốc) trong mô hình.

Tính toán độ dốc:

* N = len(x): xác định số lượng quan sát.
* diff: biến để tích lũy sự khác biệt giữa giá trị y thực tế và giá trị dự đoán, được tính theo công thức a x+b.
* Vòng lặp for đi qua tất cả các quan sát, tính toán sự khác biệt cho từng giá trị và tích lũy nó thành một khác biệt. Tiếp theo, độ dốc dọc theo b được tính toán. Trả về giá trị của gradient dọc theo b.

1. Hãy viết hàm tính gradient cho hệ số của x .

chắc chắn get\_gradient\_at\_a ( x , y , b , a ) :

  N = len ( x )

  khác biệt = 0

  vì Tôi TRONG phạm vi ( N ):

      x\_val = x [ tôi ]

      y\_val = y [ tôi ]

      khác biệt += x\_val \* ( y\_val - (( Một \* x\_val ) + b ))

  a\_gradient = - ( 2 / N ) \* khác biệt

  trở lại a\_gradient

Hàm get\_gradient\_at\_m được thiết kế để tính gradient của hàm loss đối với hệ số độ dốc a. Hãy xem cách nó hoạt động và sau đó tôi sẽ cung cấp một ví dụ hoàn chỉnh về cách sử dụng hàm này trong bối cảnh giảm độ dốc cho hồi quy tuyến tính.

Giải thích chức năng

Thông số:

* x: mảng các giá trị biến độc lập.
* y: mảng giá trị biến phụ thuộc.
* b: giá trị hiện tại của số hạng tự do (hằng số) trong mô hình.
* a: Giá trị hiện tại của hệ số độ dốc (độ dốc) trong mô hình.

Tính toán độ dốc:

* N = len(x): xác định số lượng quan sát.
* diff: biến để tích lũy sự khác biệt giữa giá trị y thực tế và giá trị dự đoán, được tính theo công thức a ⋅ x+b.
* Vòng lặp for đi qua tất cả các quan sát, tính toán sự khác biệt cho từng giá trị và tích lũy nó thành một khác biệt. Tiếp theo, chúng tôi tính toán độ dốc dọc theo a. Hàm in giá trị gradient và trả về nó.

1. Hãy xác định chức năng của một bước giảm độ dốc.

chắc chắn step\_gradient ( x , y , b\_current , a\_current , learning\_rate ):

    b\_gradient = get\_gradient\_at\_ b ( x , y , b\_current , a\_current )

    m\_gradient = get\_gradient\_at\_m ( x , y , b\_current , a\_current )

    b = b\_current - ( tỷ lệ học tập \* b\_gradient )

    Một = m\_current - ( tỷ lệ học tập \* a\_gradient )

    trả về [ b , a ]

Hàm step\_gradient được thiết kế để thực hiện một bước giảm độ dốc trong hồi quy tuyến tính. Nó cập nhật các giá trị của số hạng chặn b và hệ số độ dốc m dựa trên độ dốc được tính toán và tốc độ học được chỉ định. Hãy xem cách nó hoạt động và sau đó tôi sẽ cung cấp một ví dụ đầy đủ về cách sử dụng hàm này trong bối cảnh giảm độ dốc.

Giải thích chức năng

Thông số:

* x: mảng các giá trị biến độc lập.
* y: mảng giá trị biến phụ thuộc.
* b\_current: giá trị hiện tại của số hạng tự do (hằng số) trong mô hình.
* a \_current: Giá trị hiện tại của hệ số độ dốc (độ dốc) trong mô hình.
* learning\_rate: Tốc độ học, xác định mức độ cập nhật của các hệ số ở mỗi bước.

Tính toán độ dốc:

* b\_gradient: Được tính bằng hàm get\_gradient\_at\_b, trả về gradient của b.
* a \_gradient: được tính bằng hàm get\_gradient\_at\_ a , trả về gradient trên m.

Cập nhật tỷ lệ cược:

* Giá trị mới của b được tính bằng giá trị hiện tại của b trừ đi tích của tốc độ học và gradient trong b.
* Giá trị mới của a được tính theo cách tương tự.
* Hàm trả về một danh sách chứa các giá trị được cập nhật của b và a .

1. Hãy trực tiếp thực hiện giảm độ dốc.

chắc chắn gradient\_descent ( x , y , learning\_rate , num\_iteration ) :

  b = 0

  một = 0

  vì Tôi TRONG phạm vi ( num\_iteration ):

    b , a = step\_gradient ( x , y , b , a , learning\_rate )

  trở lại b , một

1. Hãy tính toán với dữ liệu đã cho:

b , một = gradient\_descent ( X , y , 0.01 , 1000 )

1. Hãy xây dựng một biểu đồ phân tán cho dữ liệu gốc và biểu đồ của đường thẳng thu được.

y = [ m \* x + b vì x TRONG tháng ]

làm ơn . cốt truyện ( tháng , doanh thu , "o" )

làm ơn . lô ( tháng , y )

làm ơn . trình diễn ()

Изображение выглядит как снимок экрана, текст, линия, График

Автоматически созданное описание

**Nhiệm vụ 3**

1. Đối với dữ liệu từ một tệp, hãy chọn một trong các dấu x để thực hiện dự đoán bằng mã bạn đã viết. Hình dung kết quả.

**Hồi quy tuyến tính sử dụng Scikit - tìm hiểu**

Scikit-learn là một thư viện máy học phổ biến bằng Python, cung cấp các công cụ đơn giản và mạnh mẽ để phân tích dữ liệu và xây dựng mô hình. Nó hỗ trợ nhiều thuật toán học máy, bao gồm phân loại, hồi quy, phân cụm và giảm kích thước, đồng thời cung cấp các công cụ để xử lý trước dữ liệu, đánh giá mô hình và lựa chọn tham số.

Nếu bạn chưa cài đặt thư viện, bạn có thể cài đặt nó bằng pip:

pip cài đặt scikit - tìm hiểu

Khi nhập thư viện, họ thường nhập lớp của một mô-đun cụ thể. Trong trường hợp hồi quy tuyến tính – LinearRegression.

từ sklearn.line \_model nhập LinearRegression

Tiếp theo, hãy xem việc triển khai ví dụ tương tự bằng cách sử dụng scikit - learn .

1. Hãy nhập các thư viện và đưa dữ liệu của chúng tôi vào một biểu mẫu phù hợp để sử dụng trong scikit - tìm hiểu :

nhập khẩu có khối u BẰNG n.p.

nhập khẩu matplotlib . pyplot BẰNG làm ơn

từ sklearn . model\_selection nhập khẩu tàu\_test\_split

từ sklearn . mô hình tuyến tính nhập khẩu Hồi quy tuyến tính

# Dữ liệu dưới dạng danh sách

X = [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 ] # Biến độc lập

y = [ 52 , 74 , 79 , 95 , 115 , 110 , 129 , 126 , 147 , 146 , 156 , 184 ] # Biến phụ thuộc

# Chuyển đổi danh sách thành mảng NumPy và thay đổi hình dạng của X

X = np . mảng ( X ). định hình lại ( - 1 , 1 ) # Chuyển đổi thành mảng hai chiều

y = np . mảng ( y )

Các lớp trong scikit - learn chứa các phương thức chỉ có thể xử lý mảng NumPy và Pandas DataFrame . Do đó, danh sách nguồn phải được chuyển sang loại np . mảng .

.reshape (-1, 1) thay đổi hình dạng (hoặc kích thước) của mảng. Trong trường hợp này, -1 có nghĩa là NumPy sẽ tự động xác định thứ nguyên của thứ nguyên đó dựa trên tổng số phần tử trong mảng và 1 biểu thị rằng chúng ta muốn mảng có một thứ nguyên với thứ nguyên 1. Việc chuyển đổi này là cần thiết vì Scikit -learn hy vọng rằng dữ liệu đầu vào (biến độc lập) sẽ ở dạng mảng hai chiều, trong đó mỗi hàng đại diện cho một ví dụ riêng biệt và mỗi cột đại diện cho một biến riêng biệt. Trong trường hợp hồi quy tuyến tính, nếu bạn chỉ có một biến độc lập, bạn cần biểu diễn nó dưới dạng mảng hai chiều với một cột. Nếu bạn có một số biến độc lập thì bạn không cần phải định hình lại .

1. Hãy tạo mô hình hồi quy tuyến tính:

# Tạo mô hình hồi quy tuyến tính

người mẫu = Hồi quy tuyến tính ( )

#đào tạo người mẫu

người mẫu . phù hợp ( X , y )

1. Để xây dựng dự báo, hãy sử dụng phương pháp dự đoán :

y\_pred = người mẫu . dự đoán ( X\_test )

**Nhiệm vụ 4**

1. Xây dựng mô hình hồi quy tuyến tính cho dữ liệu từ tệp được tải trong các bài tập trước. Kiểm tra các hệ số hồi quy thu được bằng cách sử dụng mô hình thống kê và thư viện Scikit - learn .

**Số liệu hồi quy tuyến tính trong Scikit - tìm hiểu**

Hệ số xác định *R* 2 (R bình phương) được tính bằng tỷ lệ biến thiên của biến phụ thuộc được giải thích bởi các biến độc lập trong mô hình. Scikit-learn sử dụng công thức sau để tính toán:

trong đó (tổng bình phương phần dư) là tổng bình phương chênh lệch giữa các giá trị thực tế ( và các giá trị dự đoán của mô hình ( ):

(tổng bình phương độ lệch so với giá trị trung bình) là tổng bình phương chênh lệch giữa giá trị thực tế ( ) và giá trị trung bình của biến phụ thuộc ( ):

**Phiên dịch**

* Nếu *R* 2 =1, điều này có nghĩa là mô hình giải thích được tất cả các biến thể trong dữ liệu.
* Nếu *R* 2 = 0, điều này có nghĩa là mô hình không giải thích bất kỳ biến thể nào và các dự đoán bằng giá trị trung bình của biến phụ thuộc.
* Giá trị *R* 2 âm có thể xảy ra nếu mô hình kém hơn giá trị trung bình đơn giản.

Để sử dụng số liệu này trong scikit - learn, bạn cần nhập phương thức tương ứng từ mô-đun số liệu :

từ sklearn . số liệu nhập khẩu r2\_score

Sau đó áp dụng cho dữ liệu tương ứng. Hãy tính toán cho dữ liệu của chúng tôi:

y\_pred = người mẫu . dự đoán ( X )

r2 = r2\_score ( y , y\_pred )

**Nhiệm vụ 5.**

Tính toán cho các mô hình R 2 được tích hợp trong scikit - tìm hiểu trước đó .

Kết luận

Là một phần của công việc này, những điều sau đây đã được xem xét:

1. một số phương pháp thay thế để xây dựng mô hình hồi quy tuyến tính;
2. Python khác nhau để thực hiện hồi quy tuyến tính;
3. sự khác biệt giữa các thư viện cho phép xây dựng mô hình hồi quy tuyến tính

**Làm việc độc lập**

1. Lấy và tải xuống tệp csv có dữ liệu bất động sản ( Nedvig . csv )
2. Xác định biến độc lập và biến phụ thuộc.
3. Xây dựng biểu đồ phân tán cho dữ liệu thu được.
4. Sử dụng thư viện statsmodel , xây dựng mô hình hồi quy tuyến tính. In thông tin về mô hình ra bàn điều khiển. Bạn có thể rút ra kết luận gì?
5. Sử dụng phương pháp dự đoán và tệp Nedvig\_test . csv tính toán dự báo cho dữ liệu.
6. Sử dụng thư viện Scikit - learn , xây dựng mô hình hồi quy tuyến tính.
7. Tính toán dự báo bằng phương pháp dự đoán và tệp Nedvig\_test . csv .
8. Tính toán R 2

Câu hỏi bảo mật:

1. Vấn đề hồi quy là gì?
2. Hồi quy tuyến tính là gì?
3. Hồi quy tuyến tính bội là gì?
4. Bản chất của phương pháp bình phương tối thiểu là gì?
5. Hạn chế của phương pháp bình phương tối thiểu