ساخت داده و طبقه بندی آن با استفاده از الگوریتم

(LINEAR REGRESSION)

نام درس :

پروژه

نام دانشجو:

محمد رضا شیخ پور

نام استاد :

استاد امین دهقان

دی 1402

دانشگاه شهید چمران کرمان

من در این پروژه با بعضی از کتابخانه های پایتون اشنا شدم .

Numpy – pands – scikitlern

با این کتابخانه ها و طرز استفاده و فایده های ان اشنا شدم در ادامه با این موارد بیشتر اشنا میشویم .

در کل این کتابخانه ها برای ریاضیات پایتون استفاده میشود .

1 Numpy ------------------------------------

2. pands ---------------------------------

3. scikitlern ------------------------------

4. code----------------------------

5. ---------------------------------منابع

Numpy

یک کتابخانه محاسبات عددی قدرتمند در پایتون است با استفاده از این کتاب خانه امکان انجام عملیات ماتریسی و عملیات ارایه ای . عملیات ریاضی سخت و پیچیده و تبدیل های داده در پایتون به اسان و سادگی فراهم میکند . ویژگی بعد نامپای این است که میتواند بر روی آرایه ها و ماتریس چند بعدی کار میکند و سریع کارایی بالایی از ارایه استفاده میکند .

نامپای توسط جامعه علمی توسعه پیدا کرده . تا جایی این کتاب خانه پیشرفت کرده که این کتاب خانه به عنوان اصلی ترین و کلیدترین کتاب خانه در پوشش داده های عددی و عملیات ماتریسی در پایتون محسوب میشود .

(نامپای) در چه حوزه هایی Numpay کتاب خانه

مور استفاده قرار میگیرد ؟

: کتاب خانه نامپای در حوزه هایی مانند

علوم داده :

نامپای به عنوان یکی کتاب خانه های کلیدی در علوم داده ها است این کتاب خانه قابلیت محاسباتی ارایه . عملیات سریع روی داده های عددی . الگوریتم و عملیات ماتریسی و .... کاربرد بسیار دیگری دارد

پردازش تصویر و بینایی ماشین :

در نامپای حوزه پردازش تصویر و بینایی ماشین هم قابلیت دارد مانند عملیات تصاویر دیجیتال مثل تغییر اندازه . تبدیلات فوریه و پردازش سریع تصویر فراهم میکند

رشته زمانی و سیگنال ها :

در نامپای به عنوان ابزار پر کابرد برای پردازش سیگنال ها و رشته های زمانی مورد استفاده قرار میگیرد

شبیه سازی محاسبات عددی :

در نامپای به عنوان یکی از پر کاربرد ترین و محبوب ترین ابزار های برای شبیه سازی و محاسبات عددی در فیزیک ریاضیات و مهندسی مورد استفاده قرار میگیرد

علوم طبیعی :

نامپای در علوم طبیعی برای عملیات ماتریسی . حل معادلات و تجزیه و تحلیل داده های طبیعی استفاده میشود

مهندسی :

در نامپای در حوزه مهندسی کاربرد زیادی دارد

به عنوان مثال برای محاسبات ریاضی، سیگنال‌های صوتی، پردازش سیگنال، سیستم‌های کنترل، رمزنگاری و دیگر حوزه‌های مهندسی نیز استفاده می‌شود. این کتابخانه قابلیت‌های قدرتمندی برای انجام محاسبات پیچیده در حوزه مهندسی استفاده میشود

بهینه سازی :

در نامپای ابزارهایی برای بهینه‌سازی محاسباتی ارائه می‌دهد و می‌تواند در حل مسائل بهینه‌سازی مورد استفاده قرار بگیرد

یادگیری ماشین :

در نامپای عنوان یکی از پایه‌های اصلی برای انجام الگوریتم‌های یادگیری ماشین استفاده می‌شود

ویژه سازی ماتریس :

مکان ویژه‌سازی و حل معادلات و مسائل مربوط به ماتریس‌ها را فراهم می‌کند.

رسم نمودار ها و تجسم داده ها :

 با همکاری با کتابخانه‌های تجسمی نظیر Matplotlib، NumPy می‌تواند داده‌ها را به شکل نمودارها و تجسم‌های مختلف نمایش دهد.

نکته: این حوزه‌ها فقط مثال‌هایی از کاربردهای NumPy در صنایع مختلف هستند و کتابخانه NumPy ممکن است در بسیاری از برنامه‌ها و حوزه‌های دیگر نیز استفاده شود

پایه‌ای برای بسیاری از کتابخانه‌ها و ابزارهای داده‌اندازی و یادگیری ماشین در پایتون محسوب می‌شود.

**مزایای کتابخانه نامپای**

از مزایای یک کتابخانه قدرتمند برای محاسبات علمی در پایتون است.

* این عملیات سریع و کارآمد را روی آرایه‌ها و
* ماتریس‌های چندبعدی ارائه می‌دهد

.

* همچنین بسیار بهینه شده است و بسیاری از توابع مفید برای جبر خطی، تبدیل فوریه و تولید اعداد تصادفی دارد

**معایب کتابخانه نامپای**

برای مبتدیان ممکن است دشوار باشد، زیرا به درک بالایی از جبر خطی و عملیات ماتریس نیاز دارد

علاوه بر این، عملکرد آن می‌تواند در هنگام برخورد با مجموعه داده‌های بزرگ کند شود

Pands

کتابخانه پانداس یکی ازقدرتمند ترین و پر استفاده ترین در زبان برنامه نویسی و پایتون میباشد . این کتابخانه برای کار با داده های تراز کشیده و تحلیل داده و همچنین این کتابخانه ابزار های توابع برای مدیریت و تبدیل داده و برای کار با داده ها در قالب جدولی مناسب و قدرتمند است

با استفاده از پانداس میتوان داده های تازه خواند شده را به سادگی به صورت جدولی سازماندهی کرد .

و همچنین میتوان داده ها را فیلتر و تحلیل و توابع محاسباتی روی داده ها اعمال میکنند .

پانداس همچنین این قابلیت دارد که نتایج به صورت گرافیکی به نمایش میگذارد .

پانداس قابلیت اتصال و ترکیب داده ها از منابع مختلف فراهم میکند .

به عنوان مثال با پانداس فراخوانی داده ها از منابع مختلف مانند اکسل و فایل

و هچنین بانک های اطلاعاتی و CSV

ارتباط با وب سرویس ها داده ها را وارد میکنند .

پانداس ابزار های برای پردازش داده های بزرگتر و مقایسه دیتا فرم (قالب جدولی) مدیریت مفهمومی نواقص در داده ها تجزیه و تحلیل داده ها زمانی فراهم میکند .

کتابخانه پانداس کاربرد های زیادی در تحلیل داده ها و کار با داده های تراز کشیده دارد در ادامه با کاربرد های این کتابخانه اشنا میشویم

با استفاده از پانداس تحلیل داده ها :.1

داده ها رو به صورت جدولی سازماندهی کرد و همچنین میتوان عملیات های مختلفی مانند فیلترکردن، ترتیب‌بندی، تجزیه و تحلیل آماری، ترسیم نمودارها و ایجاد گزارش‌های تحلیلی روی داده‌ها انجام داد.

2. پردازش داده های بزرگ :

پانداس قابلیت‌های قدرتمندی برای کار با داده‌های بزرگ و پربار فراهم کرده. به کمک ابزارهایی مانند تقسیم و یکپارچه ‌سازی دیتافریم‌ها، تجمیع داده‌ها و انجام محاسبات موازی، می‌توانید با داده‌های بزرگ و پیچیده رو به رو شوید.

3. پردازش داده های زمانی :

پانداس ابزارهای قدرتمندی برای کار با داده‌های زمانی فراهم کرده. با استفاده از این ابزارها، می‌توانید داده‌های زمانی را مدیریت، تجزیه و تحلیل کنید و الگوهای زمانی را شناسایی کنید.

4. تمیز کردن و پیش‌پردازش داده‌ها :

پانداس ابزارهایی را برای تمیز کردن و پیش‌پردازش داده‌ها فراهم کرده. با استفاده از این ابزارها می‌توانید داده‌های ناقص، تکراری یا ناهماهنگ را شناسایی و اصلاح کنید.

5. ادغام داده‌ها :

با اپانداس می‌توانید داده‌ها را از منابع مختلف مانند فایل‌های CSV، Excel و بانک‌های اطلاعاتی وارد کرد و آن‌ها را با هم ادغام کرد.

6. تحلیل و پردازش داده‌های مالی :

پانداس برای تحلیل و پردازش داده‌های مالی نیز بسیار پرکاربرد است. این کتابخانه ابزارهایی را برای محاسبه عملیات مالی مانند محاسبه بازده‌ی سهام، محاسبه معدل سودآوری، محاسبه ریسک و مقایسه سرمایه‌گذاری‌ها فراهم برای کاربرانش فراهم کرده است

7. پردازش داده‌های مرتبط با علوم داده :

پانداس در علوم داده و تحلیل آن‌ها، از جمله ماشین لرنینگ و هوش مصنوعی، استفاده می‌شود و برای تحلیل و پردازش داده‌های مرتبط با علوم داده بسیار حیاتی هستند

این ها خلاصه ای از کاربرد کتابخانه پانداس است به طوری کلی میتوان گفت پانداس برای یک ابزار قدرتمند و کامل برای تحلیل، پردازش و مدیریت داده‌ها است و در زمینه‌های مختلفی از تحلیل داده تا علوم داده و مالی مورد استفاده قرار می‌گیرد

Scikitlern

کتابخانه سایسیک لرن یکی از بهترین و محبوب ترین کتابخانه های برای یادگیری ماشین و تحلیل داده ها در زبان برنامه نویسی پایتون میباشد.

کتابخانه سایسیک لرن این کتابخانه ابزارها و الگوریتم‌های متنوعی در اختیار توسعه دهندگان قرار میدهد .

این کتابخانه قابلیت‌هایی برای پیاده‌سازی و استفاده آسان الگوریتم‌ها، پیش‌پردازش داده‌ها، استخراج و انتخاب ویژگی‌ها، ارزیابی مدل‌ها و تحلیل داده‌ها را برای توسعه دهندگان فراهم میکند .

با استفاده از سایسیک لرن میتوان الگوریتم‌های یادگیری ماشین مانند درخت تصمیم، ماشین بردار پشتیبان، شبکه عصبی، خوشه ‌بندی و رگرسیون را پیاده‌سازی و اجرا کنید. این کتابخانه همچنین ابزارهای مرتبط با تقسیم داده، ارزیابی مدل، تنظیم پارامتر و تجزیه و تحلیل داده‌ها را نیز برای کاربران و توسعه دهندگان فراهم می‌کند .

به طور خلاصه میتوان گفت با استفاده از سایکیت لرن الگوریتم‌های یادگیری ماشین را پیاده‌سازی کرد و به راحتی آن‌ها را بر روی داده‌های خود آموزش دهید و پیش‌بینی‌ها و تحلیل‌های مورد نیاز خود را انجام دهید.

در ادامه با کاربرد های کتابخانه سایسک لرن اشنا میشویم

کاربرد های سایسیک لرن بسیار گسترده هستند ما با چند تا از کاربرد های این کتابخانه اشنا میشویم :

1. طبقه‌ بندی : سایسیک لرن ابزارها و الگوریتم‌های مختلفی را برای طبقه‌بندی داده‌ها فراهم می‌کند. این الگوریتم‌ها شامل الگوریتم‌های معروفی مانند درخت تصمیم، نزدیک‌ترین همسایه، ماشین بردار پشتیبان (SVM) شبکه‌های عصبی و بیشتر هستند .

2. خوشه بندی : سایسیک لرن قابلیت خوشه‌بندی داده‌ها را فراهم می‌کند. این الگوریتم‌ها به کمک روش‌های مختلفی مانند خوشه‌بندی کمینه فاصله، خوشه‌بندی سلسله مراتبی و خوشه‌بندی طیفی، داده‌ها را به گروه‌های مشابه تقسیم می‌کنند

3. رگرسیون : سایسک لرن ابزارها و الگوریتم‌های متنوعی را برای تخمین مقادیر پیوسته بر اساس داده‌ها فراهم می‌کند. این الگوریتم‌ها شامل رگرسیون خطی، رگرسیون لجستیک، شبکه‌های عصبی و سایر روش‌های تخمین مقادیر پیوسته هستند .

4. تشخیص نقاط داده : ابزارها و الگوریتم‌های متنوعی را برای تشخیص نقاط داده نظیر نقاط پرت، نقاط خارج از محدوده عادی و نقاط نادر فراهم می‌کند. این الگوریتم‌ها به کمک روش‌های مانند کمترین مربعات و کمترین مجذورات خارج از محدوده عادی، نقاط نادر را تشخیص می‌دهند

5. استخراج ویژگی‌ها : قابلیت استخراج ویژگی‌های مختلف از داده‌ها را فراهم می‌کند. این ویژگی‌ها شامل تبدیل‌های مختلف مانند PCA، LDA و تبدیل‌های موجک است

6. ارزیابی مدل‌ها :

ابزارها و معیارهای متنوعی را برای ارزیابی عملکرد مدل‌های یادگیری ماشین فراهم می‌کند. این معیارها شامل دقت، فراخوانی، برچسب‌دهی F1 و سایر معیارهای ارزیابی هستند

7. تقسیم داده به دسته‌های آموزش و آزمون : امکان تقسیم داده‌ها به دسته‌های آموزش و آزمون را فراهم می‌کند تا بتوانید مدل‌های خود را بر روی داده‌های آموزش آموزش داده و سپس با استفاده از داده‌های آزمون عملکرد آن‌ها را ارزیابی کنید

8. ارزیابی مدل‌ها : عملکرد مدل‌های یادگیری ماشین خود را با استفاده از معیارهای مختلفی مانند دقت، صحت و

ارزیابی کنید

تعمیم پذیری مدل ها : امکان تعمیم‌پذیری مدل‌های یادگیری ماشین را فراهم می‌کند. به عبارت دیگر، شما می‌توانید مدل‌های خود را بر روی داده‌های جدید و ناشناخته اعمال کنید

اینها برخی از کاربرد های سایسیک لرن بود. سایسیک لرن در حوزه‌های علمی و صنعتی هم مورد استفاده قرار میگیرد

**import** numpy **as** np

**import** pandas **as** pa

**import** skeearn **as** sk

nampay cod

**import** numpy **as** np

arr **=** np**.**array([1, 2, 3, 4, 5])

print(arr)

[1 2 3 4 5]

In [2]:

**import** numpy **as** np

arr **=** np**.**array([1, 2, 3, 4, 5])

print(arr)

print(type(arr))

[1 2 3 4 5]

<class 'numpy.ndarray'>

In [3]:

**import** numpy **as** np

arr **=** np**.**array(42)

print(arr)

42

In [4]:

**import** numpy **as** np

arr **=** np**.**array([[1, 2, 3], [4, 5, 6]])

print(arr)

[[1 2 3]

[4 5 6]]

In [5]:

**import** numpy **as** np

arr **=** np**.**array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(arr)

[[[1 2 3]

[4 5 6]]

[[1 2 3]

[4 5 6]]]

In [6]:

**import** numpy **as** np

a **=** np**.**array(42)

b **=** np**.**array([1, 2, 3, 4, 5])

c **=** np**.**array([[1, 2, 3], [4, 5, 6]])

d **=** np**.**array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a**.**ndim)

print(b**.**ndim)

print(c**.**ndim)

print(d**.**ndim)

0

1

2

3

In [7]:

arr **=** np**.**array([1, 2, 3, 4], ndmin**=**5)

print(arr)

print('number of dimensions :', arr**.**ndim)

[[[[[1 2 3 4]]]]]

number of dimensions : 5

In [8]:

arr **=** np**.**array([1, 2, 3, 4])

print(arr[2] **+** arr[3])

7

In [9]:

arr **=** np**.**array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])

2nd element on 1st row: 2

In [10]:

arr **=** np**.**array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[1, 3])

2nd element on 1st row: 9

In [14]:

arr **=** np**.**array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])

6

In [15]:

arr **=** np**.**array([[1,2,3,4,5], [6,7,8,9,10]])

print('Last element from 2nd dim: ', arr[1, **-**1])

Last element from 2nd dim: 10

In [16]:

arr **=** np**.**array([[1,2,3,4,5], [6,7,8,9,10]])

print('Last element from 2nd dim: ', arr[1, **-**3])

Last element from 2nd dim: 8

In [18]:

arr **=** np**.**array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:6])

[2 3 4 5 6]

In [ ]:

In [19]:

arr **=** np**.**array([1, 2, 3, 4, 5, 6, 7])

print(arr[**-**1:**-**5])

[]

In [21]:

arr **=** np**.**array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:**-**5])

[2]

In [22]:

arr **=** np**.**array([1, 2, 3, 4, 5, 6, 7])

print(arr[::2])

[1 3 5 7]

**import** matplotlib.pyplot **as** plt

x **=** [5,7,8,7,2,17,2,9,4,11,12,9,6]

y **=** [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt**.**scatter(x, y)

plt**.**show()

In [5]:

**from** scipy **import** stats

x **=** [5,7,8,7,2,17,2,9,4,11,12,9,6]

y **=** [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std\_err **=** stats**.**linregress(x, y)

**def** myfunc(x):

**return** slope **\*** x **+** intercept

mymodel **=** list(map(myfunc, x))

plt**.**scatter(x, y)

plt**.**plot(x, mymodel)

plt**.**show()

In [6]:

**from** scipy **import** stats

x **=** [5,7,8,7,2,17,2,9,4,11,12,9,6]

y **=** [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std\_err **=** stats**.**linregress(x, y)

print(r)

-0.7585915243761551

In [7]:

**from** scipy **import** stats

x **=** [5,7,8,7,2,17,2,9,4,11,12,9,6]

y **=** [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std\_err **=** stats**.**linregress(x, y)

**def** myfunc(x):

**return** slope **\*** x **+** intercept

speed **=** myfunc(10)

print(speed)

85.59308314937454

In [9]:

**from** scipy **import** stats

x **=** [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]

y **=** [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

slope, intercept, r, p, std\_err **=** stats**.**linregress(x, y)

**def** myfunc(x):

**return** slope **\*** x **+** intercept

mymodel **=** list(map(myfunc, x))

plt**.**scatter(x, y)

plt**.**plot(x, mymodel)

plt**.**show()

In [10]:

x **=** [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]

y **=** [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

plt**.**scatter(x, y)

plt**.**show()

In [11]:

**import** numpy

**import** matplotlib.pyplot **as** plt

x **=** [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]

y **=** [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel **=** numpy**.**poly1d(numpy**.**polyfit(x, y, 3))

myline **=** numpy**.**linspace(1, 22, 100)

plt**.**scatter(x, y)

plt**.**plot(myline, mymodel(myline))

plt**.**show()

In [12]:

**import** numpy

**from** sklearn.metrics **import** r2\_score

x **=** [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]

y **=** [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel **=** numpy**.**poly1d(numpy**.**polyfit(x, y, 3))

speed **=** mymodel(17)

print(speed)

88.87331269697975

In [13]:

**import** numpy

**import** matplotlib.pyplot **as** plt

x **=** [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]

y **=** [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

mymodel **=** numpy**.**poly1d(numpy**.**polyfit(x, y, 3))

myline **=** numpy**.**linspace(2, 95, 100)

plt**.**scatter(x, y)

plt**.**plot(myline, mymodel(myline))

plt**.**show()

In [14]:

**import** numpy

**import** matplotlib.pyplot **as** plt

numpy**.**random**.**seed(2)

x **=** numpy**.**random**.**normal(3, 1, 100)

y **=** numpy**.**random**.**normal(150, 40, 100) **/** x

plt**.**scatter(x, y)

plt**.**show()

In [16]:

**import** numpy

**import** matplotlib.pyplot **as** plt

numpy**.**random**.**seed(2)

x **=** numpy**.**random**.**normal(3, 1, 100)

y **=** numpy**.**random**.**normal(150, 40, 100) **/** x

train\_x **=** x[:80]

train\_y **=** y[:80]

test\_x **=** x[80:]

test\_y **=** y[80:]

mymodel **=** numpy**.**poly1d(numpy**.**polyfit(train\_x, train\_y, 4))

myline **=** numpy**.**linspace(0, 6, 100)

plt**.**scatter(train\_x, train\_y)

plt**.**plot(myline, mymodel(myline))

plt**.**show()

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.neighbors **import** KNeighborsClassifier

x **=** np**.**random**.**rand(5000, 5000)

y **=** np**.**random**.**randint(2, size**=**5000)

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.25)

knn **=** KNeighborsClassifier(n\_neighbors**=**10)

knn**.**fit(x\_train, y\_train)

y\_pred **=** knn**.**predict(x\_test)

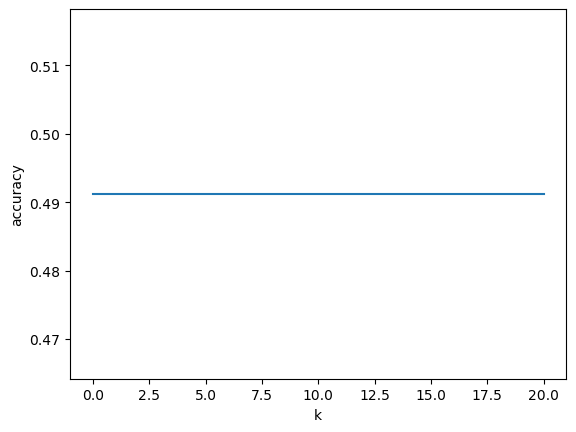
accuracy **=** np**.**mean(y\_pred **==** y\_test)

plt**.**plot([0, 20], [accuracy, accuracy])

plt**.**xlabel('k')

plt**.**ylabel('accuracy')

plt**.**show()



import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

X = np.random.rand(5000, 5000)

Y = np.random.randint(2, size=5000)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

scaler = MinMaxScaler()

X\_train\_normalized = scaler.fit\_transform(X\_train)

X\_test\_normalized = scaler.transform(X\_test)

model = LinearRegression()

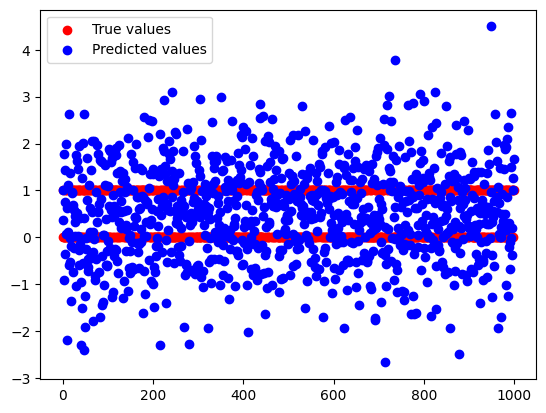
model.fit(X\_train\_normalized, Y\_train)

plt.scatter(range(len(Y\_test)), Y\_test, color='red', label='True values')

plt.scatter(range(len(Y\_test)), model.predict(X\_test\_normalized), color='blue', label='Predicted values')

plt.legend()

plt.show()



منابع

<https://maktabkhooneh.org/mag/python-libraries/>

https://maktabkhooneh.org/mag/python-libraries/.

<https://t.me/Gpt4Creator_bot>

<https://t.me/ChatGPT_BIAbot>