

Regression_Practice_Answer_1

August 4, 2023

1 Bài thực hành 1

1.1 Vấn đề

Dự đoán khả năng tiến triển của bệnh tiểu đường thông qua các chỉ số sinh lý của cơ thể.

1.2 Thông tin dữ liệu:

- Số lượng mẫu: 442 (thông tin từ 442 bệnh nhân)
- Số lượng thuộc tính: Thông tin các thuộc tính (10 cột giá trị đầu tiên): Age(tuổi), Sex (giới tính), Body mass index (chỉ số khối cơ thể), Average blood pressure(huyết áp trung bình), S1, S2, S3, S4, S5, S6 (sáu phép đo huyết thanh khác).
- Mục tiêu: Cột 11, chỉ số đánh giá mức độ tiến triển của bệnh 1 năm sau khi điều trị.

! Chú ý: Dữ liệu thông tin thuộc tính đã được chuẩn hoá

Xem thêm thông tin về nguồn dữ liệu tại: (<https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html>)

2 Hướng giải quyết

Giả sử rằng khả năng tiến triển của bệnh tiểu đường (ký hiệu: y) là đại lượng phụ thuộc tuyến tính vào các thông tin sinh lý của bệnh nhân như các thuộc tính đã mô tả ở trên (tuổi, giới tính, chỉ số khối, ... - ký hiệu: x_1, x_2, \dots, x_n) :

$$y = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

Mục tiêu: Tìm được bộ trọng số $[w_0, w_1, \dots, w_n]$ biểu diễn mối quan hệ này.

3 Các bước tiến hành

3.1 Thư viện sử dụng

- matplotlib: phục vụ vẽ các đồ thị
- numpy: tính toán các phép biến đổi trên ma trận / vector
- math: thực hiện một số hàm tính toán
- pandas: phục vụ chuyển đổi trên dữ liệu dạng bảng
- scikit-learn: (sklearn) thư viện hỗ trợ xây dựng các mô hình học máy, các hàm training và testing.

```
[1]: !pip install pandas
```

Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)

```
[2]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math

from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

3.2 Đọc dữ liệu

Dữ liệu về bệnh tiểu đường được hỗ trợ bởi sklearn, đọc dữ liệu thông qua hàm `datasets.load_diabetes()`

Xem thêm các bộ dữ liệu khác tại <https://scikit-learn.org/stable/datasets/index.html#toy-datasets>. https://scikit-learn.org/stable/datasets/toy_dataset.html

Dữ liệu nhận về ở dạng object với các thành phần thuộc tính:

- data: ma trận 2 chiều (442x10) - các thông tin bệnh nhân được chuẩn hoá về dạng số thực.
- target: mảng các số thực (442,) - chỉ số tiến triển của bệnh tiểu đường.

```
[3]: # lay du lieu diabetes - du lieu ve benh tieu duong
diabetes = datasets.load_diabetes()
print("Số chiều dữ liệu input: ", diabetes.data.shape)
print("Kiểu dữ liệu input: ", type(diabetes.data))
print("Số chiều dữ liệu target: ", diabetes.target.shape)
print("Kiểu dữ liệu target: ", type(diabetes.target))
print()

print("5 mẫu dữ liệu đầu tiên:")
print("input: ", diabetes.data[:5])
print("target: ", diabetes.target[:5])
```

Số chiều dữ liệu input: (442, 10)
Kiểu dữ liệu input: <class 'numpy.ndarray'>
Số chiều dữ liệu target: (442,)
Kiểu dữ liệu target: <class 'numpy.ndarray'>

5 mẫu dữ liệu đầu tiên:

```
input: [[ 0.03807591  0.05068012  0.06169621  0.02187239 -0.0442235
-0.03482076
-0.04340085 -0.00259226  0.01990749 -0.01764613]
[-0.00188202 -0.04464164 -0.05147406 -0.02632753 -0.00844872 -0.01916334
 0.07441156 -0.03949338 -0.06833155 -0.09220405]
[ 0.08529891  0.05068012  0.04445121 -0.00567042 -0.04559945 -0.03419447
-0.03235593 -0.00259226  0.00286131 -0.02593034]
[-0.08906294 -0.04464164 -0.01159501 -0.03665608  0.01219057  0.02499059
-0.03603757  0.03430886  0.02268774 -0.00936191]
[ 0.00538306 -0.04464164 -0.03638469  0.02187239  0.00393485  0.01559614
 0.00814208 -0.00259226 -0.03198764 -0.04664087]]
target: [151.  75. 141. 206. 135.]
```

Chia dữ liệu làm 2 phần training 362 mẫu và testing 80 mẫu

```
[4]: # cat nhỏ du lieu, lay 1 phan cho qua trinh thu nghiem,
# chia train test cac mau du lieu
# diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes_X = diabetes.data

diabetes_X_train = diabetes_X[:361]
diabetes_y_train = diabetes.target[:361]

diabetes_X_test = diabetes_X[362:]
diabetes_y_test = diabetes.target[362:]
```

3.3 Xây dựng mô hình Regression sử dụng Sklearn

Thử nghiệm xây dựng mô hình hồi quy (Linear Regression / Ridge Regression) để học được bộ tham số

- [Linear Regression](#) `linear_model.LinearRegression()`
- [Ridge Regression](#) `linear_model.Ridge()`

```
[5]: # Xay dung model su dung sklearn
regr = linear_model.LinearRegression()
```

```
[6]: ##### exercise #####
# Yêu cầu: Cài đặt mô hình Ridge Regression với alpha = 0.1
# Gợi ý: xem hướng dẫn tại https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.Ridge.html
#####

regr_ridge = linear_model.Ridge(alpha=0.1)
```

3.4 Training mô hình

Sử dụng Dữ liệu đã được chia ở bước trước đó để thực hiện training model.

=> Tìm được bộ trọng số $[w_0, w_1, \dots, w_n]$

```
[7]: # Huấn luyện mô hình Linear Regression
regr.fit(diabetes_X_train, diabetes_y_train)
print("[w1, ... w_n] = ", regr.coef_)
print("w0 = ", regr.intercept_)
```

```
[w1, ... w_n] = [-1.12982324e-02 -2.49782541e+02  5.18802202e+02
2.97220454e+02
-6.39802236e+02  3.56330510e+02  2.77834385e+01  1.46962789e+02
 6.90474996e+02  1.05716702e+02]
w0 = 152.56041961097782
```

```
[8]: ##### exercise #####
# Yêu cầu: Huấn luyện mô hình Ridge Regression và in ra các trọng số w0, w1, ...
# ↪, wn của mô hình
# Gợi ý: xem hướng dẫn tại https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.Ridge.html
#####
regr_ridge.fit(diabetes_X_train, diabetes_y_train)
print("[w1, ... w_n] = ", regr_ridge.coef_)
print("w0 = ", regr_ridge.intercept_)
```

```
[w1, ... w_n] = [ 10.51523842 -215.21946642  477.04576811  275.18547584
-64.84522547
-82.84458659 -194.22465786  112.99561167  427.19001456  121.18397114]
w0 = 152.57401415344637
```

```
[9]: ##### exercise #####
# Yêu cầu: tính giá trị dự đoán của mô hình trên mẫu đầu tiên của tập test và
# ↪ so sánh với kết quả của thư viện
# Gợi ý: sử dụng công thức  $y = w_0 + w_1*x_1 + w_1*x_2 + \dots + w_n*x_n$ 
#####
# Dự đoán thử cho trường hợp đầu tiên

# Giá trị đúng
print("Giá trị true: ", diabetes_y_test[0])

# Dự đoán cho mô hình Linear Regression sử dụng hàm dự đoán của thư viện
y_pred_linear = regr.predict(diabetes_X_test[0:1])
print("Giá trị dự đoán cho mô hình linear regression: ", y_pred_linear)

# Viết code tính và in kết quả dự đoán cho mô hình Linear Regression sử dụng
# ↪ công thức tại đây
y_pred_linear_0 = sum(regr.coef_*diabetes_X_test[0])+regr.intercept_
print("Giá trị dự đoán cho mô hình linear regression theo công thức:",
# ↪ y_pred_linear_0)
```

```

#Dự đoán cho mô hình Ridge Regression sử dụng hàm dự đoán của thư viện
y_pred_ridge = regr_ridge.predict(diabetes_X_test[0:1])
print("Gia tri du doan cho mô hình ridge regression: ", y_pred_ridge)

#Viết code tính và in kết quả dự đoán cho mô hình Ridge Regression sử dụng công
  ↳thức tại đây
y_pred_ridge_0 = sum(regr_ridge.coef_*diabetes_X_test[0])+regr_ridge.intercept_
print("Gia tri du doan cho mô hình ridge regression theo công thức:",
  ↳y_pred_ridge_0)

#####

```

Gia tri true: 321.0

Gia tri du doan cho mô hình linear regression: [234.35947872]

Gia tri du doan cho mô hình linear regression theo công thức: 234.35947872322686

Gia tri du doan cho mô hình ridge regression: [226.73474004]

Gia tri du doan cho mô hình ridge regression theo công thức: 226.73474004055703

3.5 Dự đoán các mẫu dữ liệu trong tập test

```

[10]: # Thực hiện suy diễn sau khi huấn luyện
diabetes_y_pred = regr.predict(diabetes_X_test)
pd.DataFrame(data=np.array([diabetes_y_test, diabetes_y_pred,
                             abs(diabetes_y_test - diabetes_y_pred)]).T,
             columns=["Thực tế", "Dự đoán", "Lệch"])

# pd.DataFrame(data=np.array([diabetes_y_test, diabetes_y_pred,
#                               abs(diabetes_y_test - diabetes_y_pred)]),
#               index=["Thực tế", "Dự đoán", "Lệch"])

```

```

[10]:
   Thực tế  Dự đoán  Lệch
0    321.0  234.359479  86.640521
1     58.0  163.999748 105.999748
2    262.0  163.520115  98.479885
3    206.0  167.189446  38.810554
4    233.0  254.806697  21.806697
..     ...      ...      ...
75   178.0  191.710701  13.710701
76   104.0  104.626354   0.626354
77   132.0  122.693494   9.306506
78   220.0  210.454911   9.545089
79    57.0   54.392009   2.607991

```

[80 rows x 3 columns]

3.6 Đánh giá

Sử dụng độ đo RMSE tính căn bậc 2 của trung bình bình phương lỗi. $\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2}$.

- Lỗi càng nhỏ càng thể hiện mô hình có khả năng học và dự đoán hiệu quả
- Như thế nào là nhỏ ?

```
[11]: # Giá trị RMSE của mô hình Linear Regression
math.sqrt(mean_squared_error(diabetes_y_test, diabetes_y_pred))
```

```
[11]: 51.53921127468042
```

```
[12]: ##### exercise #####
# Yêu cầu: đánh giá độ đo RMSE của mô hình Ridge Regression với các hằng số phạt khác nhau, in ra kết quả.
# Gợi ý: Các bước làm:
# - Lặp theo danh sách các hằng số phạt
# - Dùng các mô hình Ridge Regression với mỗi hằng số phạt tương ứng
# - Huấn luyện các mô hình và dự đoán
# - Tính RMSE tương ứng
#####
#Các giá trị hằng số phạt cho trước
_lambda = [0, 0.0001, 0.01, 0.04, 0.05, 0.06, 0.1, 0.5, 1, 5, 10, 20]

for a_lambda in _lambda:
    regression_regr = linear_model.Ridge(alpha = a_lambda, max_iter=1000, tol=1e-4)
    regression_regr.fit(diabetes_X_train, diabetes_y_train)
    diabetes_y_pred_redge = regression_regr.predict(diabetes_X_test)
    print('Lambda = ' + str(a_lambda) + '; RMSE = ' + str(math.sqrt(mean_squared_error(diabetes_y_test, diabetes_y_pred_redge))))
```

```
Lambda = 0; RMSE = 51.539211274680454
Lambda = 0.0001; RMSE = 51.54635240923982
Lambda = 0.01; RMSE = 51.8886606978561
Lambda = 0.04; RMSE = 52.11211596712348
Lambda = 0.05; RMSE = 52.15254636221311
Lambda = 0.06; RMSE = 52.19077570036307
Lambda = 0.1; RMSE = 52.347597926523825
Lambda = 0.5; RMSE = 54.613321511474574
Lambda = 1; RMSE = 57.382574887617025
Lambda = 5; RMSE = 67.24133680169886
Lambda = 10; RMSE = 71.18738262709283
Lambda = 20; RMSE = 74.05735936735674
```

```
[13]: !pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-
```

packages (0.12.2)
 Requirement already satisfied: numpy!=1.24.0,>=1.17 in
 /usr/local/lib/python3.10/dist-packages (from seaborn) (1.22.4)
 Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-
 packages (from seaborn) (1.5.3)
 Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in
 /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
 Requirement already satisfied: contourpy>=1.0.1 in
 /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
 (1.1.0)
 Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
 packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
 Requirement already satisfied: fonttools>=4.22.0 in
 /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
 (4.41.1)
 Requirement already satisfied: kiwisolver>=1.0.1 in
 /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
 (1.4.4)
 Requirement already satisfied: packaging>=20.0 in
 /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
 (23.1)
 Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
 packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
 Requirement already satisfied: pyparsing>=2.3.1 in
 /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
 (3.1.0)
 Requirement already satisfied: python-dateutil>=2.7 in
 /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
 (2.8.2)
 Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
 packages (from pandas>=0.25->seaborn) (2022.7.1)
 Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
 packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)

3.6.1 Vẽ biểu đồ phân phối cho chỉ số thực tế

```
[14]: import seaborn as sns
sns.distplot(diabetes_y_test)
pd.DataFrame(data=diabetes_y_test, columns=["values"]).describe()
```

<ipython-input-14-e34a5db57d2a>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

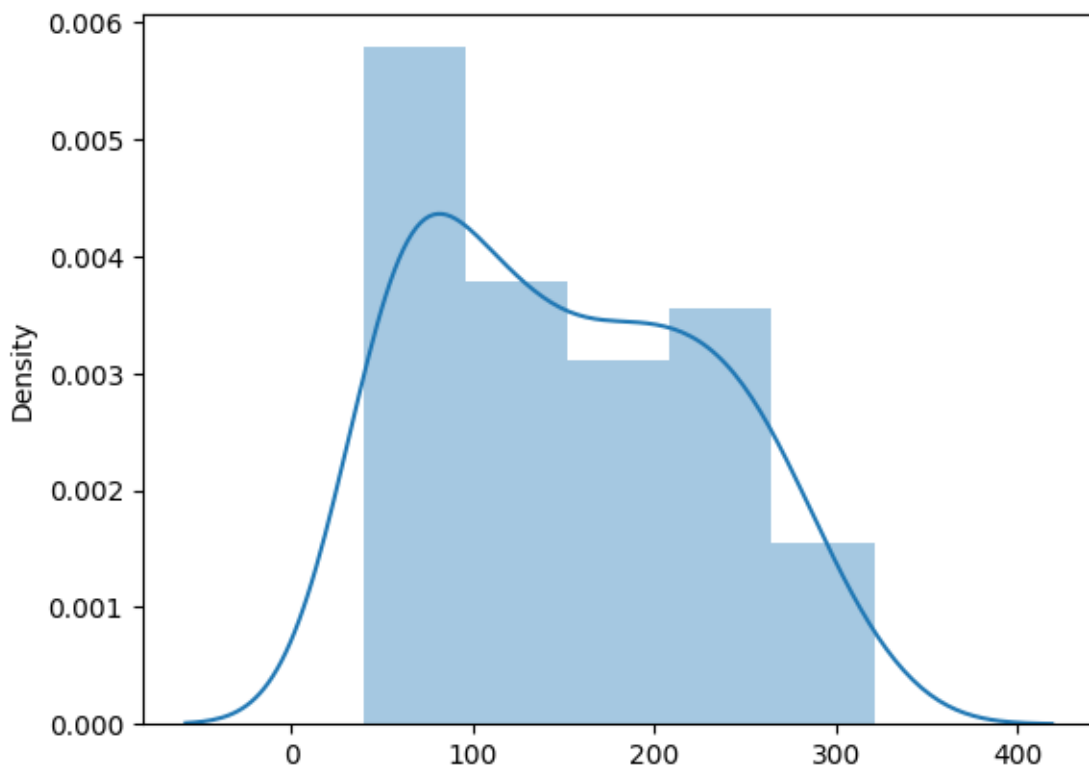
Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(diabetes_y_test)
```

```
[14]:          values
count    80.00000
mean     152.38750
std       78.46994
min       40.00000
25%       72.00000
50%      140.00000
75%      217.50000
max      321.00000
```



3.6.2 Vẽ biểu đồ phân phối cho chỉ số dự đoán của mô hình linear regression

```
[15]: ##### exercise #####
# Yêu cầu: Tính các chỉ số thống kê và vẽ biểu đồ phân phối của chỉ số dự đoán
# bằng mô hình Linear Regression, quan sát và nhận xét
# Gợi ý: sử dụng sns và pd
#####
sns.distplot(diabetes_y_pred)
```



```
pd.DataFrame(data=diabetes_y_pred, columns=["values"]).describe()
```

<ipython-input-15-46981ec08c70>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

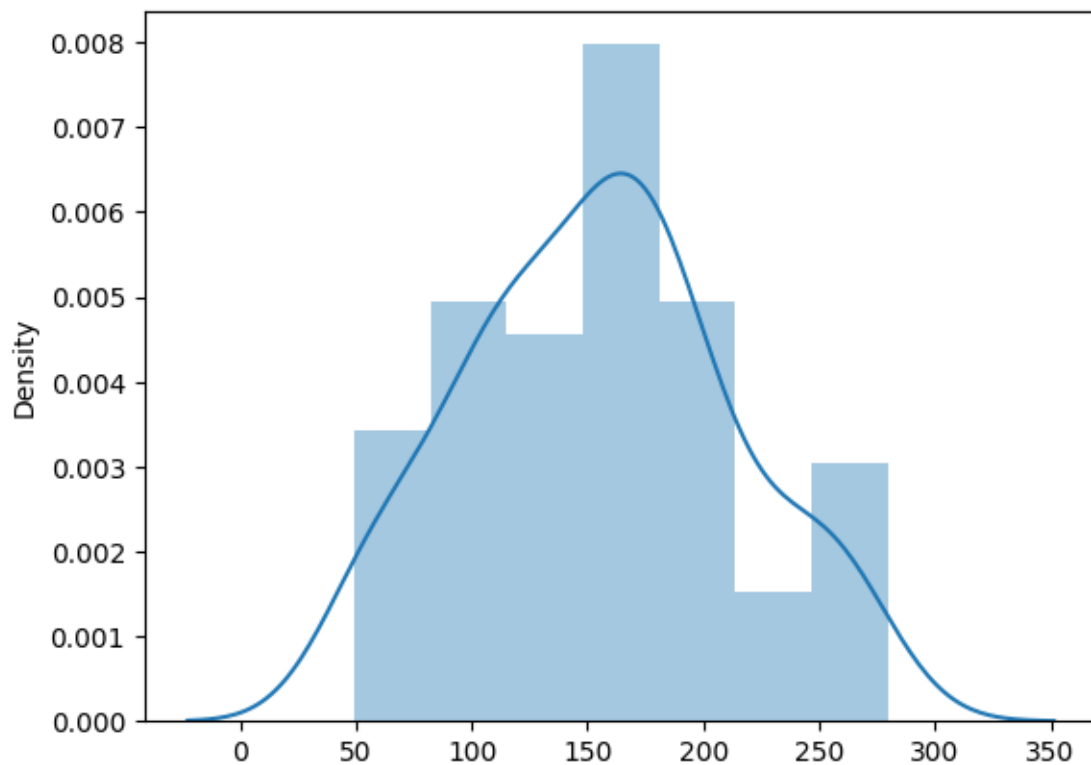
Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(diabetes_y_pred)
```

```
[15]:
```

	values
count	80.000000
mean	155.501049
std	57.511599
min	49.193721
25%	112.399159
50%	161.143223
75%	191.047878
max	279.598577



3.6.3 Vẽ biểu đồ so sánh kết quả dự đoán và thực tế

```
[16]: import matplotlib.pyplot as plt

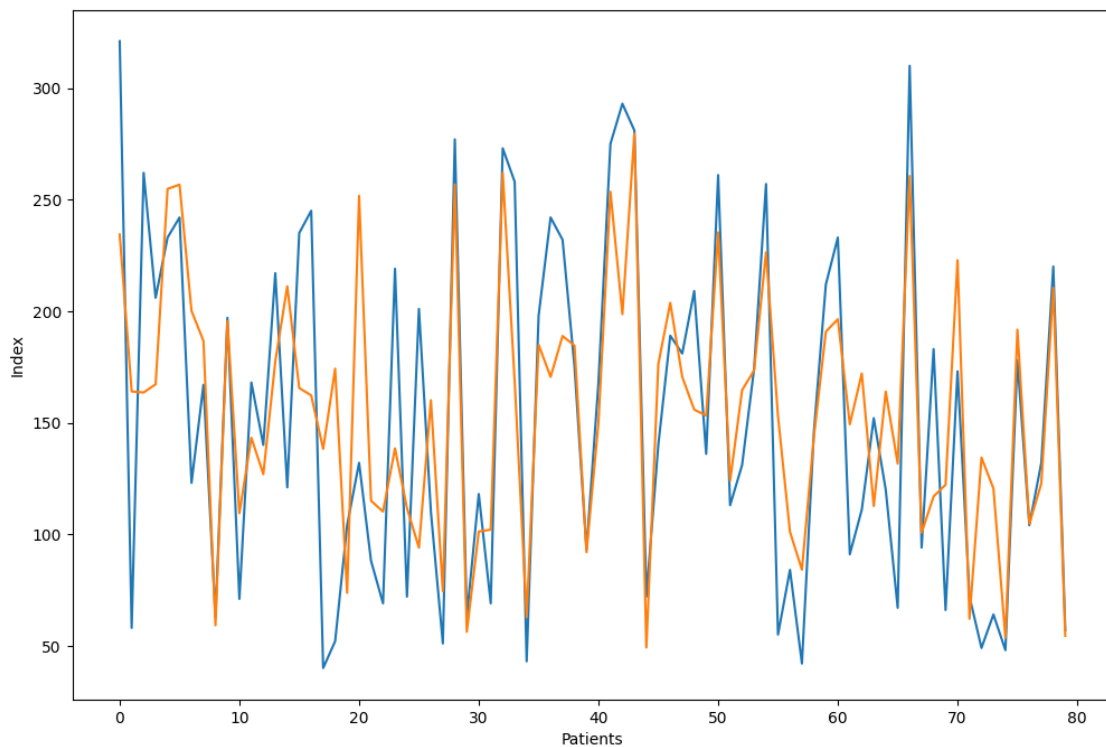
plt.figure(figsize=(12,8))

plt.plot(diabetes_y_test)
plt.plot(diabetes_y_pred)

plt.xlabel('Patients')

plt.ylabel('Index')

# function to show the plot
plt.show()
```



[16]: