Practice_answer_1

September 2, 2021

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 ap 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: `x1, x2, .. x_n`) :

```
y = w0 + w1*x1 + w2*x2 + ... + w_n*x_n
```

Mục tiêu: Tìm được bộ trọng số `[w0, w1, ... w_n]` biểu diễn mối quan hệ này.

3 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.

[3]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\linhvn\miniconda3\lib\site-
packages (1.2.3)
Requirement already satisfied: python-dateutil>=2.7.3 in
c:\users\linhvn\miniconda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: numpy>=1.16.5 in
c:\users\linhvn\miniconda3\lib\site-packages (from pandas) (1.20.1)
Requirement already satisfied: pytz>=2017.3 in
c:\users\linhvn\miniconda3\lib\site-packages (from pandas) (2021.1)
Requirement already satisfied: six>=1.5 in c:\users\linhvn\miniconda3\lib\site-
packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
```

[1]:
```
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
```

4 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.

[2]:
```
# 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  0.02187235 -0.0442235
-0.03482076
  -0.04340085 -0.00259226  0.01990842 -0.01764613]
 [-0.00188202 -0.04464164 -0.05147406 -0.02632783 -0.00844872 -0.01916334
   0.07441156 -0.03949338 -0.06832974 -0.09220405]
 [ 0.08529891  0.05068012  0.04445121 -0.00567061 -0.04559945 -0.03419447
  -0.03235593 -0.00259226  0.00286377 -0.02593034]
 [-0.08906294 -0.04464164 -0.01159501 -0.03665645  0.01219057  0.02499059
  -0.03603757  0.03430886  0.02269202 -0.00936191]
 [ 0.00538306 -0.04464164 -0.03638469  0.02187235  0.00393485  0.01559614
   0.00814208 -0.00259226 -0.03199144 -0.04664087]]
target:  [151.  75. 141. 206. 135.]
```

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

[3]:
```
# cat nho 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:]
```

5 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()`

[4]:
```
# Xay dung model su dung sklearn
regr = linear_model.LinearRegression()
```

[5]:
```
##### 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)
```

6 Training mô hình

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

=> Tìm được bộ trọng số [w0, w1, ... w_n]

[6]:
```
# 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] =  [-1.40025874e-02 -2.49785764e+02  5.18797590e+02
2.97225806e+02
 -6.39824326e+02  3.56346572e+02  2.77927304e+01  1.46967414e+02
  6.90487255e+02  1.05714417e+02]
w0 =  152.56042805095646
```

[7]:
```
##### 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] =  [  10.51377484 -215.2216077   477.04290225  275.19004043
-64.84582447
  -82.84560954 -194.22489474  112.99649773  427.19288067  121.182733  ]
w0 =  152.57401459390456
```

[8]:
```
##### 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 = w0 + w1*x1 + w1*x2 + ... + w_n*x_n
########################
#Dự đoán thử cho trường hợp đầu tiên

#Giá trị đúng
print("Gia tri 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("Gia tri du doan 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("Gia tri du doan 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:  [234.36115876]
Gia tri du doan cho mô hình linear regression theo công thức: 234.3611587552754
Gia tri du doan cho mô hình ridge regression:  [226.73584699]
Gia tri du doan cho mô hình ridge regression theo công thức: 226.73584698688643
```

7 Dự đoán các mẫu dữ liệu trong tập test

[9]:
```
# 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"])
```

[9]:

	Thực tế	Dự đoán	Lệch
0	321.0	234.361159	86.638841
1	58.0	163.998469	105.998469
2	262.0	163.521039	98.478961
3	206.0	167.189777	38.810223
4	233.0	254.808171	21.808171
..	…	…	…
75	178.0	191.711158	13.711158
76	104.0	104.622355	0.622355
77	132.0	122.691780	9.308220
78	220.0	210.455070	9.544930
79	57.0	54.393113	2.606887

```
[80 rows x 3 columns]
```

8 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 ?

[10]:
```
# Giá trị RMSE của mô hình Linear Regression
math.sqrt(mean_squared_error(diabetes_y_test, diabetes_y_pred))
```

[10]:
```
51.539230657101335
```

[11]:
```
##### 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
# - Dựng 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 = 51.539230657101385
Lambda = 0.0001; RMSE = 51.54637136803664
Lambda = 0.01; RMSE = 51.888661243305464
Lambda = 0.04; RMSE = 52.112100345430115
Lambda = 0.05; RMSE = 52.15252744411004
Lambda = 0.06; RMSE = 52.19075386078626
Lambda = 0.1; RMSE = 52.34756662293576
Lambda = 0.5; RMSE = 54.61325570600304
Lambda = 1; RMSE = 57.38250521480497
Lambda = 5; RMSE = 67.24129454526758
Lambda = 10; RMSE = 71.18735549546312
```

```
Lambda = 20; RMSE = 74.05734361149904
```

[16]:
```
!pip install seaborn
```

```
Collecting seaborn
  Downloading seaborn-0.11.1-py3-none-any.whl (285 kB)
Requirement already satisfied: numpy>=1.15 in
c:\users\linhvn\miniconda3\lib\site-packages (from seaborn) (1.20.1)
Requirement already satisfied: scipy>=1.0 in
c:\users\linhvn\miniconda3\lib\site-packages (from seaborn) (1.6.1)
Requirement already satisfied: pandas>=0.23 in
c:\users\linhvn\miniconda3\lib\site-packages (from seaborn) (1.2.3)
Requirement already satisfied: matplotlib>=2.2 in
c:\users\linhvn\miniconda3\lib\site-packages (from seaborn) (3.3.4)
Requirement already satisfied: pytz>=2017.3 in
c:\users\linhvn\miniconda3\lib\site-packages (from pandas>=0.23->seaborn)
(2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
c:\users\linhvn\miniconda3\lib\site-packages (from pandas>=0.23->seaborn)
(2.8.1)
Requirement already satisfied: cycler>=0.10 in
c:\users\linhvn\miniconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(0.10.0)
Requirement already satisfied: pillow>=6.2.0 in
c:\users\linhvn\miniconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(8.1.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\linhvn\miniconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\linhvn\miniconda3\lib\site-packages (from matplotlib>=2.2->seaborn)
(2.4.7)
Requirement already satisfied: six>=1.5 in c:\users\linhvn\miniconda3\lib\site-
packages (from python-dateutil>=2.7.3->pandas>=0.23->seaborn) (1.15.0)
Installing collected packages: seaborn
Successfully installed seaborn-0.11.1
```

8.1 Vẽ biểu đồ phân phối cho chỉ số thực tế

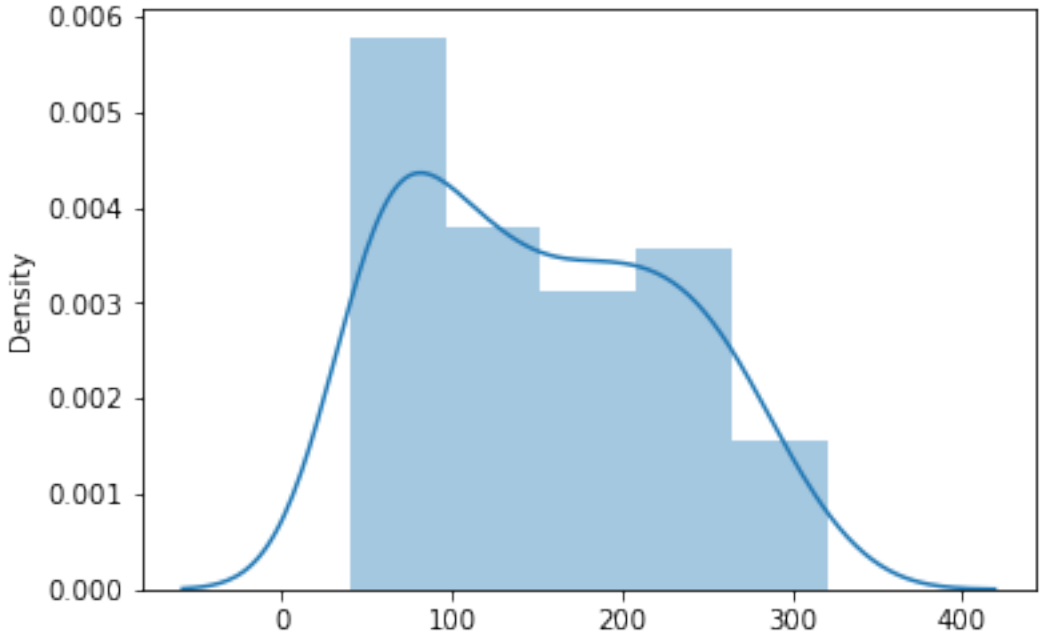
[12]:
```
import seaborn as sns
sns.distplot(diabetes_y_test)
pd.DataFrame(data=diabetes_y_test, columns=["values"]).describe()
```

```
D:\ProgramingStudy\Python\Anaconda\lib\site-
packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
```

```
  warnings.warn(msg, FutureWarning)
```

[12]:

	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



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

[13]:

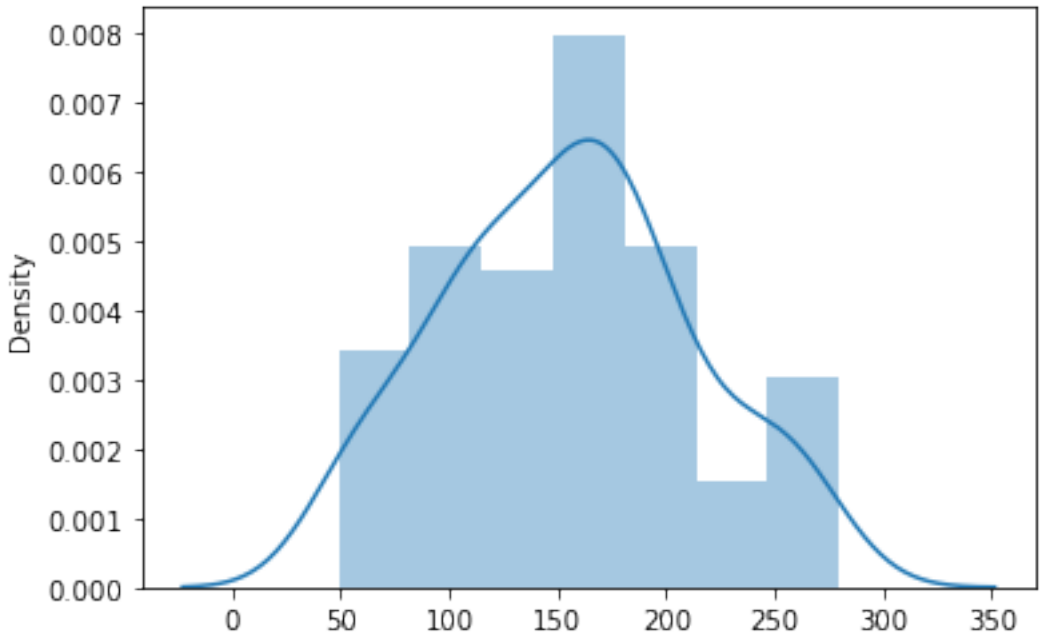
```
##### exercise #####
# Yêu cầu: Tính các chỉ số thống kê và vẽ biểu đồ phân phối của chỉ số dự đoán␣
 ↪bằng mô hình Linear Regression, quan sát và nhận xét
# Gợi ý: sử dụng sns và pd
#######################
sns.distplot(diabetes_y_pred)
pd.DataFrame(data=diabetes_y_pred, columns=["values"]).describe()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

```
D:\ProgramingStudy\Python\Anaconda\lib\site-
packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
```

```
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

[13]:

	values
count	80.000000
mean	155.501064
std	57.511992
min	49.190655
25%	112.400481
50%	161.144060
75%	191.047286
max	279.600249

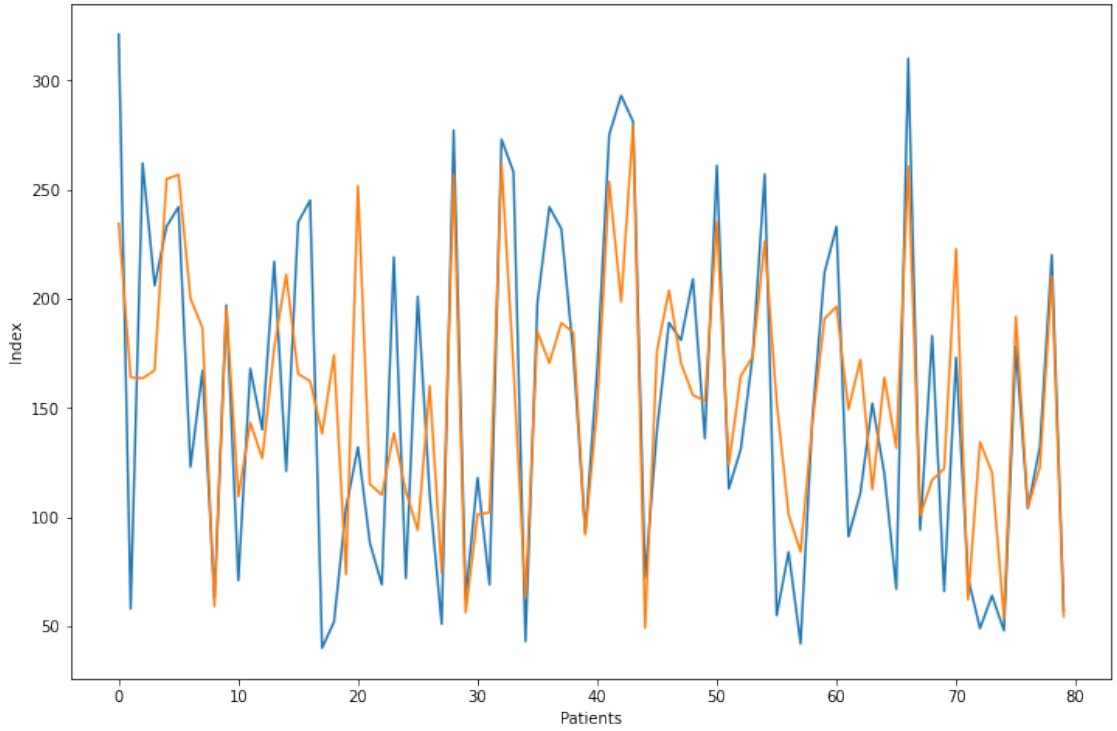


8.3 Vẽ biểu đồ so sánh kết quả dự đoán và thực tế

[33]:

```
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()
```



[ ]: