

期末報告:

以 Neural Network 去回歸進出口與加  
權股價指數的關係

作者: John Blue

# 目標:

利用 python 的機器學習套件 sklearn，建構 Neural Network 作迴歸 (regression) 預測，回歸目標為台灣每個月進出口 4 個維度的資訊以及台灣加權股價指數收盤指數一個月平均值的映射關係。



有待回歸的目標(左進出口，右加權指數)

# 套件說明:

Requests 負責請求伺服器連線並取得原始資訊。

BeautifulSoup 負責將 html 的超文本解析出來，這樣一來我們就可以針對超文本進行進一步的操作，例如尋找某 label: `<table> <tr> <td> ...` 並將其中包含的文字萃取出來。

Numpy 是 python 語言的數字處理的相關資源庫。

Matplotlib 是 python 資料視覺化的相關資源庫。

sklearn 是 python 實作 machine learning 的相關資源庫，我會用其中的 MLPRegressor 對台灣每個月進出口 4 個維度的資訊以及台灣加權股價指數收盤指數一個月平均值進行回歸預測。

Pandas 是 python 資料的處理的相關資源庫，相當於 excel 的功能，可以輸入與匯出.csv 檔案。

**第一階段**我們利用 Requests 以及 BeautifulSoup 向兩個政府公開的網址抓取資料，利用 `numpy.array()` 建成 numpy 資料格式並以 numpy 做一些基礎運算，再將所有資料彙整並以 pandas 輸出。

**第二階段**首先藉由 pandas 將第一階段獲取的資料讀進，之後開始回歸運算。回歸運算的第一步會將資料藉由 sklearn 下的 `train_test_split` 去做分割，分割成訓練資料跟測試資料，再來用 sklearn 下的 MLPRegressor 去做類神經運算並帶入測試資料進行預測，最後再以 matplotlib 進行出圖。

# Neural Network model 說明：

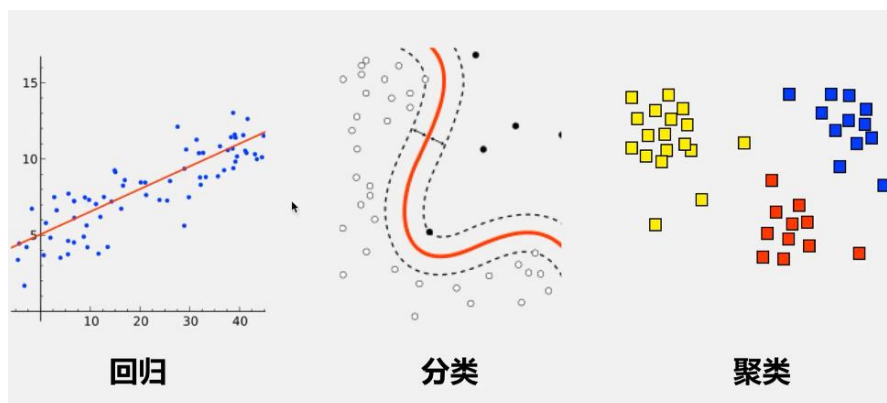
人工神經網絡（ANN）是模仿動物大腦的生物神經網絡計算系統。這樣的系統通常通過考慮示例來“學習”執行任務，而不用特定於任務的規則進行編程。例如，在圖像識別中，他們可能會通過分析已被手動標記為“貓”或“沒有貓”，並使用結果識別其他圖像中的貓來學會識別包含貓的圖像。

人工神經網絡基於稱為人工神經元的連接單元或節點的集合，這些單元或節點可以對生物腦中的神經元進行鬆散建模。每連接都像生物大腦中的突觸一樣，可以將信號傳輸到其他神經元。接收信號的人工神經元隨後對其進行處理，並可以向與之相連的神經元發出信號。

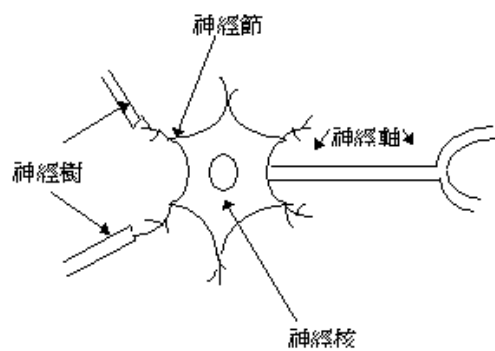
在 ANN 實現中，連接處的“信號”為實數，每神經元的輸出通過其輸入之和的某些非線性函數來計算。神經元和邊緣通常具有隨著學習進行而調整的權重。權重增加或減小連接處信號的強度。通常，神經元聚集成層。不同的層可以對它們的輸入執行不同的變換。

人工神經網絡方法的最初目標是以與人腦相同的方式解決問題。但是，隨著時間的流逝，人們的注意力轉移到執行特定任務上。人工神經網絡已用於多種任務，包括計算機視覺，語音識別，機器翻譯，社交網絡過濾，棋盤遊戲和視頻遊戲，醫療診斷，甚至用於傳統上被認為是人類專有的活動，例如繪畫。

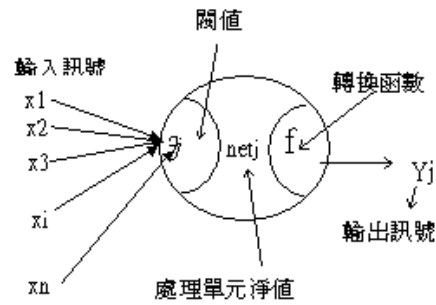
ANN 在目前市面上的使用策略有兩種，一是做分類，二是做回歸，兩者的區別微妙，分類是將輸入以及有人為標註的輸出一併灌進類神經模型進行訓練標註的輸出值必定為離散數值;不同於分類，回歸則是在訓練一種連續的映射關係，灌進模型的輸入輸出都會是連續數值。



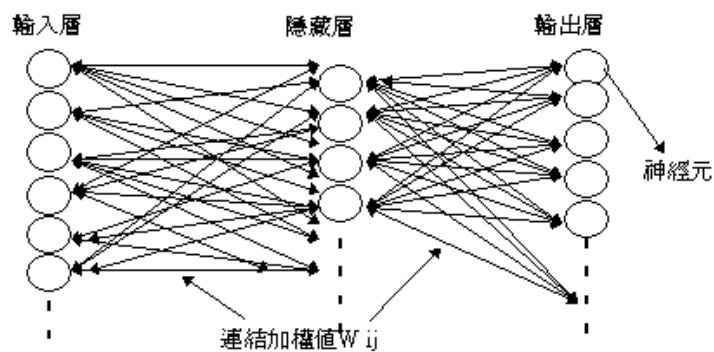
回歸，分類，聚類 的比較示意圖



圖一 生物神經元模型



圖二 人工神經元模型



圖三 倒傳遞類神經網路架構

每一層神經元間都有聯繫關係的權重

類神經模型示意圖

## 數據說明：

本次主題將會分成兩個階段，第一階段為從網站上抓取資料，並輸出成.csv 以讓第二階段能有直接的資料去不斷試驗訓練模型的參數要怎麼調整。第二階段的輸入資料則從低一階段的輸出.csv 所取得，經過一些轉換讓套件中的模型得以訓練。第一階段的程式碼在 `Getdata.py`，第二階段的程式碼在 `Rundata.py`。

## 第一階段:抓取的數據說明：

第一階段中，我們利用 `Requests` 以及 `BeautifulSoup` 向兩個政府公開的網址抓取資料

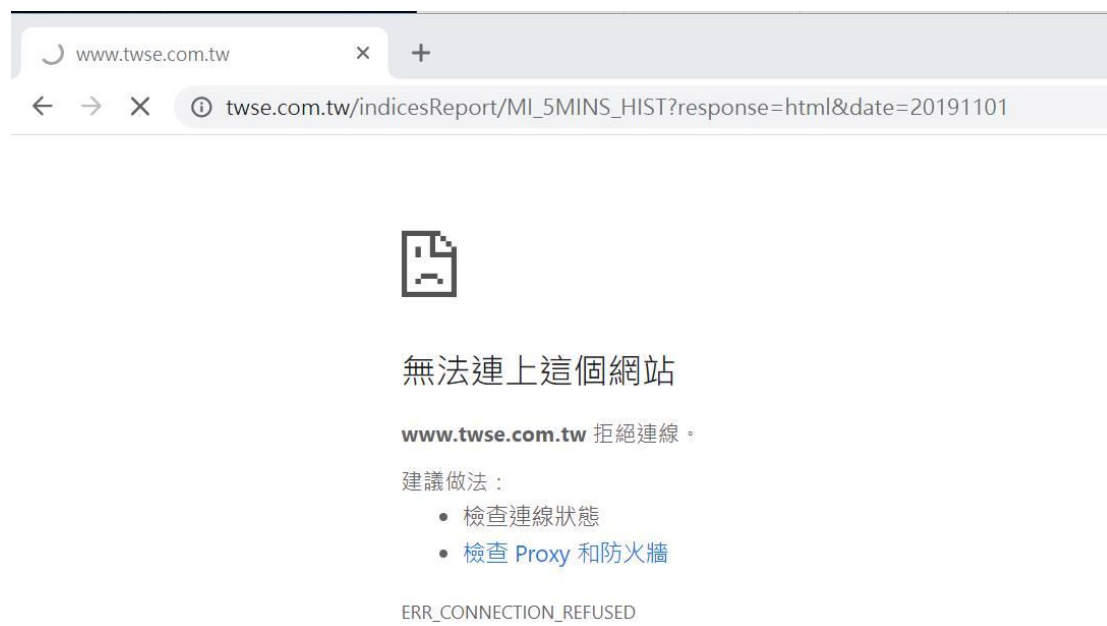
(加權股價指數加權指數

[https://www.twse.com.tw/zh/page/trading/indices/MI\\_5MINS\\_HIST.html](https://www.twse.com.tw/zh/page/trading/indices/MI_5MINS_HIST.html)，

進出口 4 項資料

<http://web02.mof.gov.tw/njswww/WebProxy.aspx?sys=100&funid=defjsptgl>)。

加權股價指數加權指數的 url 都是單筆的資訊，例如說前面提到的網址進去之後可以自由選取所需要的年份與月份，而我要的是 1999~2018 年 1~12 月的資訊，所以我會用迴圈不斷的稍微更新 url 去獲取資料。但是有一個極大的困難點，這種瞬間獲取資料的策略將會被政府公開資料拒絕存取(request 會告訴我們此網站目前拒絕存取，如下圖，而被擋住之後會有 1 個小時的時間會拒絕存取。而解決方法很機械式，就是使用時間延遲 44 秒 time(44)，使得獲取資料的頻率降低。這樣就能騙過伺服器，只是我設的秒數 44 秒 time(44)，而我的資料有 240 讀取次數，這樣跑的時間大約是 4 個小時左右。



### 政府公開資料拒絕存取畫面

加權股價指數加權指數的資料一次為一個月，標題分為" 日期:", " 開盤指數:", " 最高指數:", " 最低指數:", " 收盤指數:" 而我以收盤指數為標準，將約 30 筆的資訊做平均值，視為當月分的加權股價指數。

	A	B	C	D	E
1	108年12月	發行量加權股價指數歷史資料			
2	日期	開盤指數	最高指數	最低指數	收盤指數
3	108/12/02	11,509.94	11,524.85	11,454.38	11,502.83
4	108/12/03	11,473.32	11,531.58	11,460.06	11,531.58
5	108/12/04	11,511.82	11,513.83	11,457.43	11,510.47
6	108/12/05	11,546.75	11,604.39	11,546.75	11,594.65
7	108/12/06	11,639.80	11,657.65	11,577.83	11,609.64
8	108/12/09	11,635.47	11,678.36	11,631.03	11,660.77
9	108/12/10	11,647.78	11,649.73	11,607.26	11,627.84
10	108/12/11	11,635.07	11,700.77	11,622.58	11,700.77
11	108/12/12	11,766.98	11,875.98	11,766.98	11,836.42
12	108/12/13	11,937.90	11,990.78	11,913.02	11,927.73

加權股價指數加權指數某月的資料

進出口 4 項資料則是進去網在之後自行點選所需的資訊，再進行爬蟲。本次題目點取的項目是，貿易指數(左側欄)->總指數->點選 88 年 1 月 至 107 年 12 月(統計期)->點選 數量指數出口+數量指數進口+單位價值指數出口+單位價值指數進口(項目別) ->查詢，所得的資料如下，進出口 4 項資料還會另外進行部分資料的去除動作，例如說 107 年,108 年,... 在讀取進來後會予以忽略。

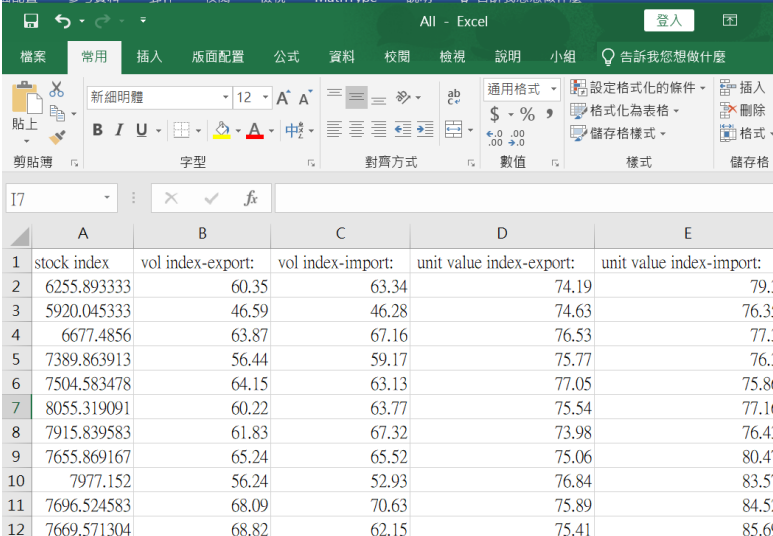
產生時間:108/12/26 10:35:46

## 貿易指數－總指數

	數量指數－出口	數量指數－進口	單位價值指數－出口	單位價值指數－進口
107年	110.82	107.15	100.99	108.51
107年 11月	111.62	102.09	102.08	112.86
107年 12月	115.49	107.77	101.08	110.22
108年 1月	112.39	121.87	99.20	107.85
108年 2月	84.49	68.88	98.79	111.05
108年 3月	118.52	117.45	98.84	108.53
108年 4月	106.21	104.14	98.99	110.20
108年 5月	114.72	104.99	99.20	110.65
108年 6月	119.41	112.13	99.19	111.00
108年 7月	118.32	113.27	98.49	109.03
108年 8月	121.56	104.67	99.04	110.91
108年 9月	118.82	116.82	98.52	108.41
108年 10月	123.00	117.07	97.04	107.34
108年 11月	120.91	113.88	96.10	105.70

## 進出口 4 項資料

在讀取資料資後之後，我利用 `numpy.array()` 建成 `numpy` 資料格式並以 `numpy` 做一些基礎運算，再將所有資料彙整並以 `pandas` 輸出 `All.csv` 輸出之 `.csv` 如下。分別中文英文對應，股價加權指數: stock index，數量指數出口: vol index-export，數量指數進口: vol index-import，單位價值指數出口: unit value index-export，單位價值指數進口: unit value index-import。



	A	B	C	D	E
1	stock index	vol index-export:	vol index-import:	unit value index-export:	unit value index-import:
2	6255.893333	60.35	63.34	74.19	79.3
3	5920.045333	46.59	46.28	74.63	76.35
4	6677.4856	63.87	67.16	76.53	77.3
5	7389.863913	56.44	59.17	75.77	76.3
6	7504.583478	64.15	63.13	77.05	75.86
7	8055.319091	60.22	63.77	75.54	77.16
8	7915.839583	61.83	67.32	73.98	76.43
9	7655.869167	65.24	65.52	75.06	80.47
10	7977.152	56.24	52.93	76.84	83.57
11	7696.524583	68.09	70.63	75.89	84.52
12	7669.571304	68.82	62.15	75.41	85.69

## 輸出之 All.csv

## 第二階段:訓練及測試數據說明：

第二階段中，首先藉由 `pandas` 將第一階段獲取的資料讀進，讀進的資料經作處理變成 `numpy` 資料格式 分別為進出口資料 `X` 加權股價指數資料 `Y`

X - NumPy array							Y - NumPy array	
	0	1	2	3			0	
0	60.35	63.34	74.19	79.3	0		6255.89	
1	46.59	46.28	74.63	76.35	1		5920.05	
2	63.87	67.16	76.53	77.3	2		6677.49	
3	56.44	59.17	75.77	76.3	3		7389.86	
4	64.15	63.13	77.05	75.86	4		7504.58	
5	60.22	63.77	75.54	77.16	5		8055.32	
6	61.83	67.32	73.98	76.43	6		7915.84	
7	65.24	65.52	75.06	80.47	7		7655.87	
8	56.24	52.93	76.84	83.57	8		7977.15	

進出口資料 `X` 以及加權股價指數資料 `Y`

取得資訊之後開始回歸運算，回歸運算的第一步會將資料藉由 `sklearn` 下的 `train_test_split` 去做分割，分割成訓練資料 `X_train`, `Y_train` 跟測試資料 `X_test`, `Y_test`。

X_train - NumPy array							Y_train - NumPy array	
	0	1	2	3			0	
0	82.31	82.48	105.07	108.78	0		7630.3	
1	77.09	77.09	99.73	108.63	1		6834.54	
2	52.8	48.66	104.72	101.57	2		4475.14	
3	67.98	78.86	105.22	99.65	3		5497.72	
4	82.08	90.53	109.28	113.14	4		8579.58	
5	67.21	79.27	107.29	100.75	5		5865.56	
6	86.04	92.2	112.56	120.96	6		7922.65	
7	70.68	77.5	104.26	101.34	7		6609.32	
8	82.07	77.02	110.35	122.58	8		5043.05	

訓練資料 `X_train`, `Y_train`



X_test - NumPy array							Y_test - NumPy array
	0	1	2	3			0
0	49.52	59.9	92.76	85.89	0		5440.06
1	61.83	67.32	73.98	76.43	1		7915.84
2	83.83	88.4	105.82	107.9	2		7029.4
3	68.33	71.36	111.73	119.53	3		7999.48
4	88.28	87.5	112.49	118.52	4		9069.11
5	99.85	91.67	107.56	119.11	5		8272.34
6	110.03	101.06	101.92	110.04	6		10986.5
7	75.4	86.81	103.94	101.01	7		6500.62
8	68.09	70.63	75.89	84.52	8		7696.52

測試資料 X\_test, Y\_test

再來用 sklearn 下的 MLPRegressor 去做類神經運算並帶入測試資料進行預測，資料分別為 X\_test, Y\_test, Pred。

Pred - NumPy array	
	0
0	5873.14
1	5654
2	7810.3
3	7545.49
4	8237.4
5	8445.14
6	8517.23
7	7437.82
8	6042.51

預測資料 Pred

## 準確性說明：

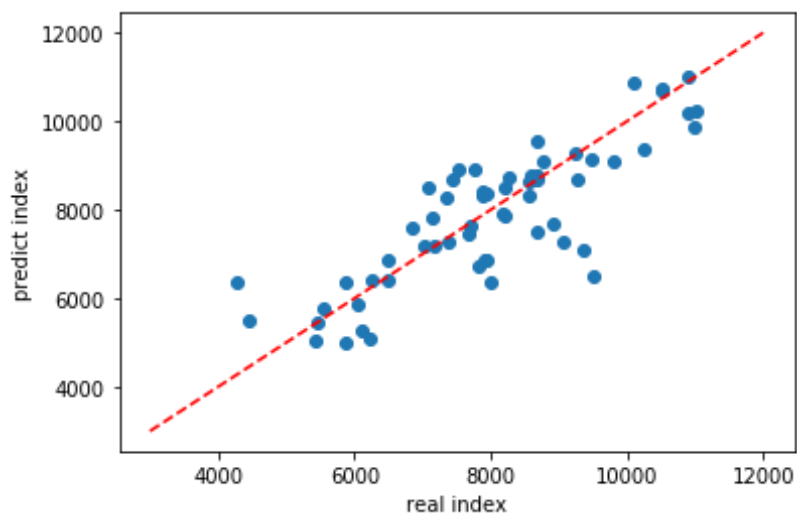
經過一些運算後，準確性說明如下，資料儲存在 Compare.csv。



B	C	D	E	F	G	H
x[1]	x[2]	x[3]	x[4]	y from real function (A)	y from prediction (B)	relative error (B-A)/A
49.52	59.9	92.76	85.89	5440.0625	5873.144618	0.079609769
61.83	67.32	73.98	76.43	7915.839583	5653.996347	-0.285736366
83.83	88.4	105.82	107.9	7029.3985	7810.295985	0.111090228
68.33	71.36	111.73	119.53	7999.480714	7545.494087	-0.056752012
88.28	87.5	112.49	118.52	9069.111667	8237.397932	-0.091708402
99.85	91.67	107.56	119.11	8272.343182	8445.138781	0.02088835
110.03	101.06	101.92	110.04	10986.51	8517.226073	-0.22475599

準確性說明示意圖

以下為回歸差異圖，虛線為對稱線，每點座標都由實際值以及預測值組成，離虛線則表示準確，反之則不理想。



回歸差異圖

## 程式說明:

### Getdata

這幾行 import 所需的套件

```
import time
```

```
import requests
```

```
from bs4 import BeautifulSoup as soup
```

這幾行抓取股票加權指數

```
# get data one url by one
```

```
def get_IEX(url_IEX):
```

# 由 requests 提出網路的請求並提交表單，將結果儲存到變數 page 中

```

page = requests.get(url_IEX)
# 將網頁原始碼丟給 BeautifulSoup 解析
#print(page.text)
html = soup(page.text, "html.parser")
# for storage
data = []
data_IEX = []
col_IEX = [" 日期:", " 開盤指數:", " 最高指數:", " 最低指數:", " 收盤指數:"]
# get table
Time_table = html.find("table")
# get tr from table
Time_table = Time_table.find_all("tr")
for i in range(len(Time_table)):
    # skip first 2 lines
    if i < 2:
        continue
    # tmp for data
    tmp = []
    # get td from every tr(runs)
    td = Time_table[i].find_all("td")
    for j in range(len(td)):
        if j == 0:
            a = td[j].text
            tmp.append(a)
            continue
        # get text in a in each td(td[i])
        a = td[j].text.split(',')
        a1 = a[1].split('.')
        # append tmp for data
        tmp.append(1000 * float(a[0]) + float(a1[0]) + 0.01 * float(a1[1]))
    # append data
    data_IEX.append(tmp)
    data.append(tmp[4])
# show
#print(len(data_IEX))
# return
return sum(data) / len(data)
# for storage

```

```

data_IEX = []
mon = [ '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12' ]
#for i in range(1999,2019):
for i in range(1999,2000):
    for j in range(0,12):
        url_IEX =
'https://www.twse.com.tw/indicesReport/MI_5MINS_HIST?response=html&date=' +
str(i) + mon[j] + '01'
        data_IEX.append([ get_IEX(url_IEX) ])
        # 每次睡 44 秒 以避免被認為是網路攻擊
        time.sleep( 44 )
    這幾行抓取進出口資料
# 選擇要爬蟲的網址
url_Port =
'http://web02.mof.gov.tw/njswww/WebProxy.aspx?sys=220&ym=8801&ymt=10712
&kind=21&type=1&funid=i8301&cycle=41&outmode=0&compmode=00&outkind=1
1&fldspc=2,4,&rdm=ej2lmneW"
# 由 requests 提出網路的請求並提交表單，將結果儲存到變數 page 中
page = requests.get(url_Port)
# 將網頁原始碼丟給 BeautifulSoup 解析
#print(page.text)
html = soup(page.text, "html.parser")
# for storage
data_Port = []
col_Port = [ " 數量指數-出口:", " 數量指數-進口:", " 單位價值指數-出口:", " 單位
價值指數-進口: " ]
# " 編號: " in th
#col_Port = [ " 編號:", " 數量指數-出口:", " 數量指數-進口:", " 單位價值指數-出
口:", " 單位價值指數-進口: " ]
# get table
Index_table = html.find("table", {"class": "tblcls"})
#Index_table = html.find("table", {"summary": "統計資料庫查詢結果網頁表格資料"
# get tr from table
Index_table = Index_table.find_all("tr")
# skip 88 ~ 107
skip = []
#for i in range(88,108):
for i in range(88,89):

```

```

        skip.append(str(i) + '年')
for i in range(len(Index_table)):
    # skip first line
    if i == 0:
        continue
    # skip 107 108, ... year
    ck = 0
    for j in range(len(skip)):
        if Index_table[i].find("th").text == skip[j]:
            ck = 1
            break
    if ck == 1:
        continue
    # tmp for data
    tmp = []
    # get td from every tr(runs)
    td = Index_table[i].find_all("td")
    for j in range(len(td)):
        # get text in a in each td(td[i])
        a = td[j].find("a").text
        # append tmp for data
        tmp.append(float(a))
    # append data
    data_Port.append(tmp)
print(len(data_Port))
    這幾行將股價加權指數以及進出口資料合併
# contensate
col = [" 收盤指數:", " 數量指數-出口:", " 數量指數-進口:", " 單位價值指數-出口:", " 單位價值指數-進口:"]
data = []
for i in range(len(data_IEX)):
    data.append( data_IEX[i] + data_Port[i] )
    這幾行輸出合併的資料回.csv
# using pandas to output
import pandas as pd
# dataframe
df = pd.DataFrame(data, columns = col)
df.to_csv('All.csv', index = False)

```

# Rundata

這幾行 import 所需的套件

```
import time
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
import matplotlib.pyplot as plt
```

這幾行讀取.csv 檔並轉呈 numpy 格式

```
# reading the data using pandas
Xi = pd.read_csv('All.csv')
X = []
X.append( Xi.values[:,1].flatten().tolist() )
X.append( Xi.values[:,2].flatten().tolist() )
X.append( Xi.values[:,3].flatten().tolist() )
X.append( Xi.values[:,4].flatten().tolist() )
X = np.transpose( np.array(X) )
Y = []
Y.append( Xi.values[:,0].flatten().tolist() )
Y = np.transpose( np.array(Y) )
```

這幾行是分割成訓練資料跟測試資料(參數 X\_train, X\_test, Y\_train, Y\_test)

```
# split data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size = 0.75, random_state
= 42, shuffle = True)
```

這幾行建構 MLPRegressor 運算 並預測

```
# MLP model
model = MLPRegressor(hidden_layer_sizes = (70,40,),
                      activation = 'relu',
                      solver = 'adam',
                      learning_rate = 'adaptive',
                      learning_rate_init = 0.001,#0.001
                      max_iter = 10000,
                      verbose = True, momentum = 0.9)

# fit (training)
model.fit(X_train, Y_train)

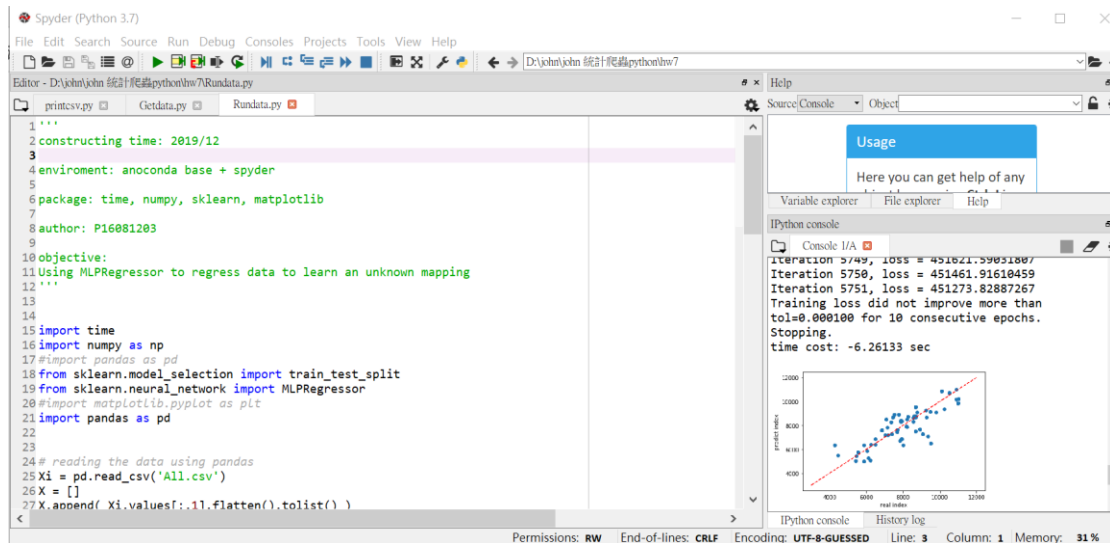
# predict and result
```

```

Pred = model.predict(X_test)
# we use abs(true value - predict value) to analyze
#result = [ Pred[i] - Y_test[i] for i in range(M_ts) ]
    這幾行計算跑了多久
start = time.time()
end = time.time()
timecost = start - end
print('time cost: %.5f sec' %timecost)
    誤差分析
# data construction
col = ['x[1]', 'x[2]', 'x[3]', 'x[4]', 'y from real function (A)', 'y from prediction (B)',
'relative error (B-A)/A']
data = []
for i in range(len(Y_test)):
    tmp = [ X_test[i][0], X_test[i][1], X_test[i][2], X_test[i][3], Y_test[i][0], Pred[i],
(Pred[i] - Y_test[i][0]) / Y_test[i][0] ]
    data.append(tmp)
    輸出.csv
# using pandas to output
import pandas as pd
# dataframe
df = pd.DataFrame(data, columns = col)
df.to_csv('Compare.csv')
#df.to_excel('compare.xlsx')
    使用 matplotlib 做 回歸差異圖
# using matplotlib
import matplotlib.pyplot as plt
# base line
y_b = np.linspace(3000,12000)# (,)之間的值
# show scatter
fig1, axs1 = plt.subplots()
axs1.plot(y_b, y_b, 'r--')
axs1.scatter(Y_test, Pred)
axs1.set_xlabel('real index')
axs1.set_ylabel('predict index')

```

# 執行畫面 Print Screen:



...

Iteration 5749, loss = 451621.59031807

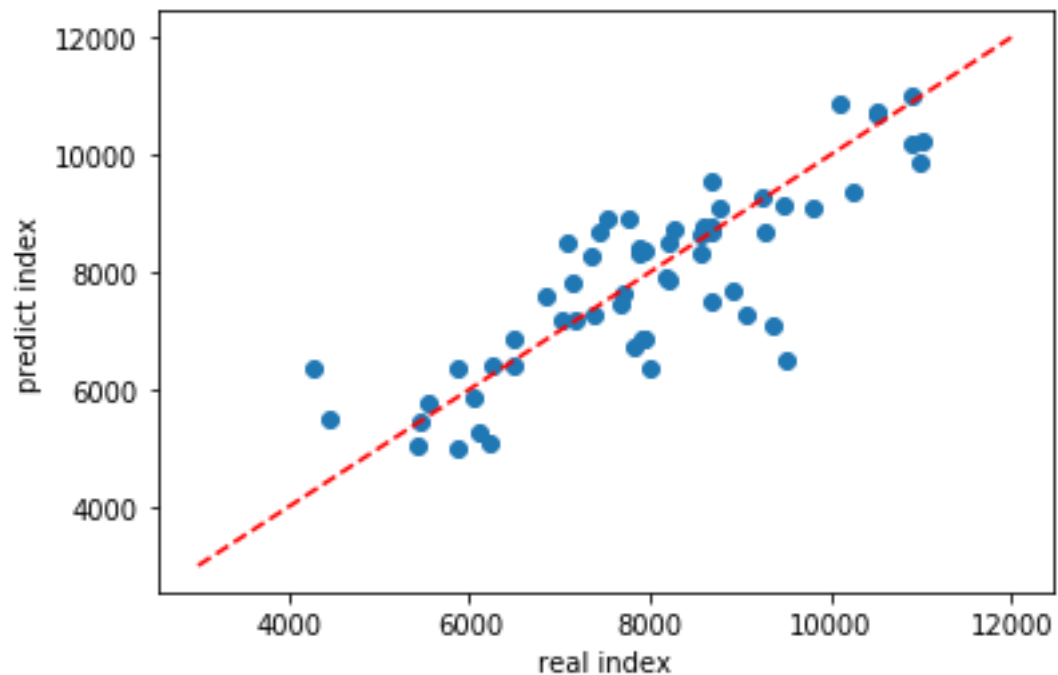
Iteration 5750, loss = 451461.91610459

Iteration 5751, loss = 451273.82887267

Training loss did not improve more than tol=0.000100 for 10 consecutive epochs.

Stopping.

time cost: -6.26133 sec





# 心得

訊號處理是快速發展的一門學理，由通訊理論和計算機科學交集 已被廣泛應用在多媒體、通訊、生物醫學、航太工程與電子工程等領域，是現代化社會和科技發展的重要技術其中之一。

這學期的科目介紹正是在訊號處理方面的抽象概念，目的為把抽象的理論--統計與機率--經過寫程式於以具體化，並用演算法計算最適合的統計參數。課程從基礎的統計方法開始，接著再推展到訊號處理，以及近幾年比較熱門的機器學習，因而本科目內容包含的基礎知識統計學科知識、程式使用---python、機器學習基礎介紹 Un-supervised learning 非監督學習以及 Supervised learning(監督學習)的分別、Classification(分類)以及 Regression(迴歸)彼此的差別，網路爬蟲的基礎使用、等等。經過一學期的學習，確實可以發現自己被磨練，而具備進階之規劃與科研能力。