雲端計算平台實務
期末報告

第九組
組員 ：
資訊四甲 10527124 邱正皓
資訊四甲 10527112 周志鴻
資訊四甲 10527111 施育暘

主題：股票分析
使用模組：
Microsoft Azure：Python 與 Azure Notebooks 的機器學習服務介紹

主題說明與系統架構介紹:

在以機器學習服務進行資料分析的領域中，Python 已經成為了主要的語言。 了解如何利用 Python 和 Jupyter Notebooks 中的相關程式庫，在 Azure Notebooks 上執行來預測模式及識別趨勢。

製作動機：

Azure 在做數據處理以及資料分析上非常強大有很多套件可以使用，像是這次的主題範例就是做天氣預測以及飛機誤點預測，我們覺得非常酷也很實用，所以我們也想利用 Azure 的資源來做分析股市趨勢，能讓我們發大財。

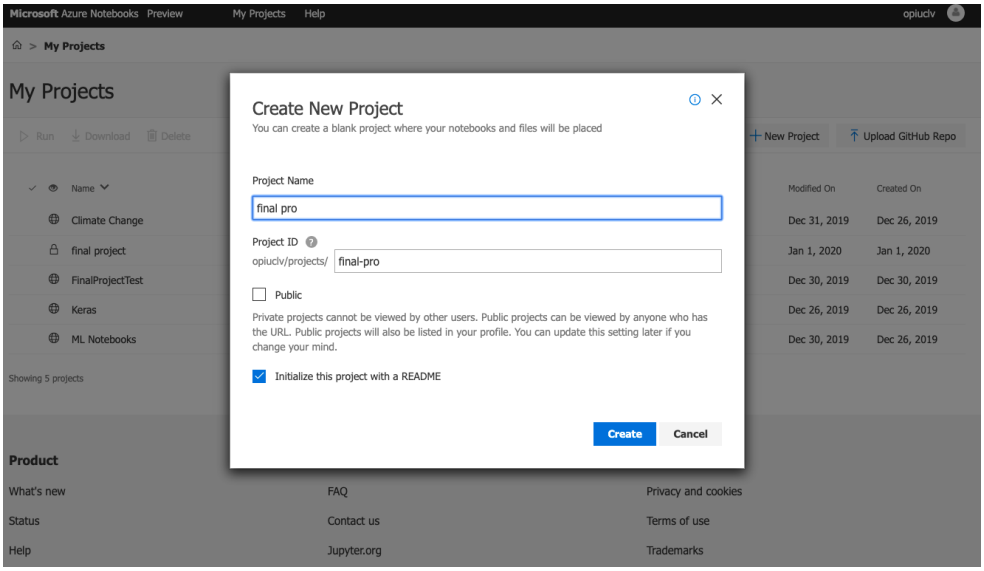
資料抓取 ：

有別於 Azure 範例自己抓取所需資料，使用爬蟲爬取台灣證券交易所台新金(2887)資料，並使用 json 格式解析轉存成.csv 檔方便分析。

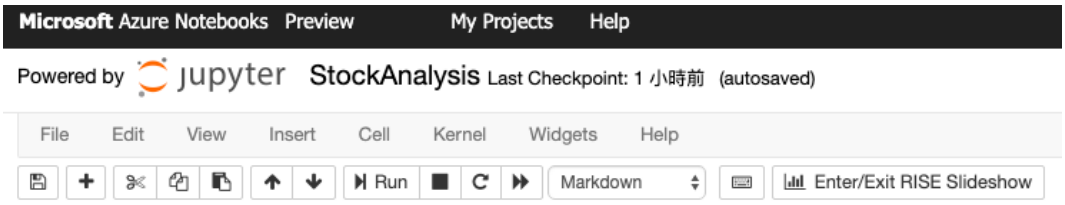
```
import datetime
import requests
import json
import csv

def dateRange(start, end, step=1, format="%Y-%m-%d"):  #設定datarange函式
    strptime, strftime = datetime.datetime.strptime, datetime.datetime.strftime
    days = (strptime(end, format) - strptime(start, format)).days
    return [strftime(strptime(start, format) + datetime.timedelta(i), format) for i in range(0, days, step)]


# ***********************************************************************************************************************

#搜尋日期範圍
date = dateRange("2017-11-01", "2017-12-31") # 設定時間範圍

testnum =  date[0]+date[1]+date[2]+date[3]+date[5]+'.'+date[8]+date[9] # 先給testnum初始 **因為不要"-"符號

for d in range(len(date)): #用迴圈跑日期

    #搜尋日期值
    day = date[d][0]+date[d][1]+date[d][2]+date[d][3]+date[d][5]+date[d][6]+date[d][8]+date[d][9] # 給day初始值 **因為不要

    if testnum[5] != day[5] :  # 一個月只要抓一次

        url_twse = 'http://www.twse.com.tw/exchangeReport/STOCK_DAY?response=json&date='+day+'&stockNo=2887&_=153874659

        res = requests.get(url_twse) # get網頁內容
        s = json.loads(res.text) # 把json格式轉給python

        #outputfile = open(r'台灣證券市場(台新金)'+day[5]+'月'+'.csv', 'w', newline='' )
        outputfile = open(r'Taishin(2887)'+ '201'+ day[3] + day[4] + day[5] +'.csv', 'w', newline='' )
        outputwriter = csv.writer(outputfile)
        outputwriter.writerow(s['title'])
        outputwriter.writerow(s['fields'])
        for data in (s['data']):
            outputwriter.writerow(data)

        outputfile.close()

    testnum =  date[d][0]+date[d][1]+date[d][2]+date[d][3]+date[d][5]+date[d][6]+date[d][8]+date[d][9]
```

方法實作:

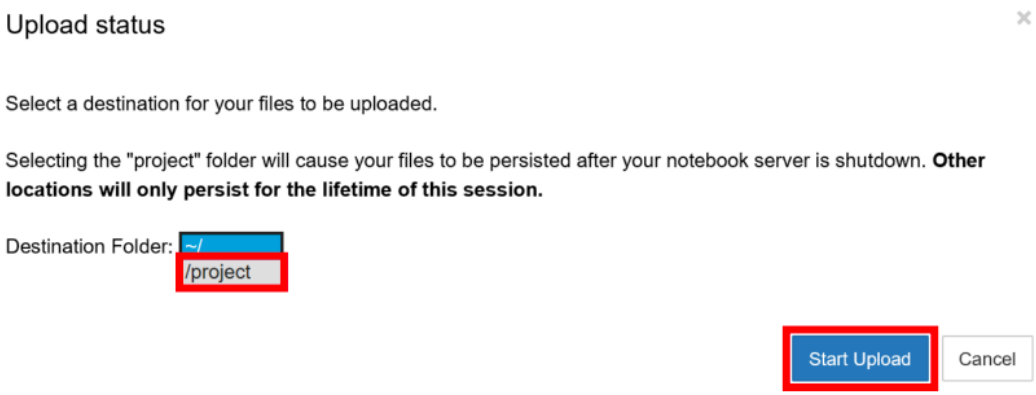
1. 建立 Azure NoteBook Project



2. 使用.ipynb 檔，開啟 jupyter 實作



3. upload 資料到 NoteBook



4. improt 所需的資料庫

```
import matplotlib.pyplot as plt
import os
import math
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import seaborn as sns; sns.set()
```

5. 利用 Pandas 做資料預處理

先抓取 2017~2018 年份每一個月的交易資料作為訓練集，再抓取 2019 年份的交易資料作為測試集。

```
def get_BaseFiles() : #2017 2018年份
    # 指定要列出所有檔案的目錄
    mypath = r"/Users/opiuclv/Desktop/Taishin2887Base"

    # 取得所有檔案與子目錄名稱
    Basefiles = listdir(mypath)  #遍歷整份檔案
    Basefiles.pop(5) #在run的時候會跑出一個DS.Store隱藏暫存檔 要把它踢掉
    Basefiles.sort(key=lambda x:int(x[14:19])) #排序檔案名稱

    for i in range(len(Basefiles)): # 使用loop集合資料
        #content = content.append(files[i])
        if i == 0:
            content = pd.read_csv("/Users/opiuclv/Desktop/Taishin2887Base/" + Basefiles[i]).iloc[1:,:]
        else:
            content = np.vstack((content, pd.read_csv("/Users/opiuclv/Desktop/Taishin2887Base/" + Basefiles[i]).iloc[1:,:]))


    content = pd.DataFrame(content) #將集合好的資料存成csv方便分析
    content.to_csv("TaishinBasefile.csv",index = False)
    return Basefiles

def get_file(): #2019年份
    # 指定要列出所有檔案的目錄
    mypath = r"/Users/opiuclv/Desktop/Taishin2887"

    # 取得所有檔案與子目錄名稱
    files = listdir(mypath)  #遍歷整份檔案
    files.pop(2) #在run的時候會跑出一個DS.Store隱藏暫存檔 要把它踢掉
    files.sort(key=lambda x:int(x[13:15])) #排序檔案名稱

    for i in range(len(files)): # 使用loop集合資料
        #content = content.append(files[i])
        if i == 0:
            content = pd.read_csv("/Users/opiuclv/Desktop/Taishin2887/" + files[i]).iloc[1:,:]
        else:
            content = np.vstack((content, pd.read_csv("/Users/opiuclv/Desktop/Taishin2887/" + files[i]).iloc[1:,:]))


    content = pd.DataFrame(content) #將集合好的資料存成csv方便分析
    content.to_csv("Taishin.csv",index = False)
    return files

files = get_file()
Basefiles = get_BaseFiles()
```

6. 檢查資料格式並使用 shape()函式來列出數量

```
priceBase = pd.read_csv("TaishinBasefile.csv")
priceBase.head()
```

	Date	Trading volume	Transaction amount	Opening price	Highest price	Lowest price	Closing price	Price difference	Number of transactions	9	...	23	24	25	26	27	28	29	30	31	32
0	106/01/03	5,850,782	68,921,876	11.8	11.80	11.70	11.80	0.00	1,298	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	106/01/04	9,525,453	112,374,148	11.8	11.85	11.75	11.80	0.00	2,502	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	106/01/05	15,407,062	182,765,274	11.8	11.90	11.75	11.90	+0.10	3,181	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	106/01/06	12,181,976	144,985,562	11.9	11.95	11.85	11.95	+0.05	2,582	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	106/01/09	10,336,371	122,780,097	11.9	11.95	11.85	11.85	-0.10	2,785	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 33 columns

```
price = pd.read_csv("Taishin.csv")
price.head()
```

	Date	Trading volume	Transaction amount	Opening price	Highest price	Lowest price	Closing price	Price difference	Number of transactions	9	...	23	24	25	26	27	28	29	30	31	32
0	108/01/02	13,078,054	171,175,897	13.10	13.15	13.05	13.10	+0.05	3,272	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	108/01/03	9,814,590	128,683,491	13.10	13.15	13.05	13.10	0.00	2,807	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	108/01/04	13,902,019	181,112,213	13.05	13.10	12.95	13.00	-0.10	3,642	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	108/01/07	18,835,228	249,170,023	13.15	13.35	13.10	13.35	+0.35	4,170	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	108/01/08	10,514,947	139,025,306	13.30	13.30	13.15	13.25	-0.10	3,479	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 33 columns

```
In [5]: priceBase.shape
Out[5]: (472, 33)

In [6]: price.shape
Out[6]: (199, 33)
```

7. 檢查資料是否有遺漏值

檢查遺漏值是為了避免出現在分析資料時沒有資料而產生錯誤，進而導致無法辨識分析資料而讓我們建立的圖表出現問題。

```
In [7]: priceBase.isnull().values.any()
Out[7]: True

In [8]: price.isnull().values.any()
Out[8]: True
```

8. 找出遺漏值位置

這裡使用兩個方法，一個是能確認每個分類是否有遺漏值，另一個是計算出每個分類出現遺漏值的數量，也方便我們來做擷取資料與補足缺失的資料。

```
priceBase.isnull().any()
```

Date	False
Trading volume	False
Transaction amount	False
Opening price	False
Highest price	False
Lowest price	False
Closing price	False
Price difference	False
Number of transactions	False
9	True
10	True
11	True
12	True
13	True
14	True
15	True
16	True
17	True
18	True
19	True
20	True
21	True
22	True
23	True
24	True
25	True
26	True
27	True
28	True
29	True
30	True
31	True
32	True

dtype: bool

```
priceBase.isnull().sum()
```

Date	0
Trading volume	0
Transaction amount	0
Opening price	0
Highest price	0
Lowest price	0
Closing price	0
Price difference	0
Number of transactions	0
9	472
10	472
11	472
12	472
13	472
14	472
15	472
16	472
17	472
18	472
19	472
20	472
21	472
22	472
23	472
24	472
25	472
26	472
27	472
28	472
29	472
30	472
31	472
32	472

dtype: int64

```
price.isnull().any()
```

Date	False
Trading volume	False
Transaction amount	False
Opening price	False
Highest price	False
Lowest price	False
Closing price	False
Price difference	False
Number of transactions	False
9	True
10	True
11	True
12	True
13	True
14	True
15	True
16	True
17	True
18	True
19	True
20	True
21	True
22	True
23	True
24	True
25	True
26	True
27	True
28	True
29	True
30	True
31	True
32	True

dtype: bool

```
price.isnull().sum()
```

Date	0
Trading volume	0
Transaction amount	0
Opening price	0
Highest price	0
Lowest price	0
Closing price	0
Price difference	0
Number of transactions	0
9	199
10	199
11	199
12	199
13	199
14	199
15	199
16	199
17	199
18	199
19	199
20	199
21	199
22	199
23	199
24	199
25	199
26	199
27	199
28	199
29	199
30	199
31	199
32	199

dtype: int64

9. 擷取需要的資料

因為在讀取我們合併好的資料時，會發現在後面有許多沒有資料的空白，那麼這裡我們是選擇切出我們需要的分類資料。

```
priceBase = priceBase[["Date", "Trading volume", "Transaction amount", "Opening price", "Highest price", "Lowest price", "Closing price", "Price difference", "Number of transactions" ]]
priceBase.head()
```

	Date	Trading volume	Transaction amount	Opening price	Highest price	Lowest price	Closing price	Price difference	Number of transactions
0	106/01/03	5,850,782	68,921,876	11.8	11.80	11.70	11.80	0.00	1,298
1	106/01/04	9,525,453	112,374,148	11.8	11.85	11.75	11.80	0.00	2,502
2	106/01/05	15,407,062	182,765,274	11.8	11.90	11.75	11.90	+0.10	3,181
3	106/01/06	12,181,976	144,985,562	11.9	11.95	11.85	11.95	+0.05	2,582
4	106/01/09	10,336,371	122,780,097	11.9	11.95	11.85	11.85	-0.10	2,785

```
price = price[["Date", "Trading volume", "Transaction amount", "Opening price", "Highest price", "Lowest price", "Closing price", "Price difference", "Number of transactions" ]]
price.head()
```

	Date	Trading volume	Transaction amount	Opening price	Highest price	Lowest price	Closing price	Price difference	Number of transactions
0	108/01/02	13,078,054	171,175,897	13.10	13.15	13.05	13.10	+0.05	3,272
1	108/01/03	9,814,590	128,683,491	13.10	13.15	13.05	13.10	0.00	2,807
2	108/01/04	13,902,019	181,112,213	13.05	13.10	12.95	13.00	-0.10	3,642
3	108/01/07	18,835,228	249,170,023	13.15	13.35	13.10	13.35	+0.35	4,170
4	108/01/08	10,514,947	139,025,306	13.30	13.30	13.15	13.25	-0.10	3,479

10. 切取單獨的分段

這裡我們也選用另一個方式來取我們需要的一個的分類資料，而我們取出的是股票的收盤價格來做處理。

選取需要分析的區塊

```
In [13]: priceBase = priceBase.iloc[:,6]
         priceBase.head()
```

```
Out[13]: 0    11.80
         1    11.80
         2    11.90
         3    11.95
         4    11.85
         Name: Closing price, dtype: float64
```

```
In [14]: price = price.iloc[:,6]
         price.head()
```

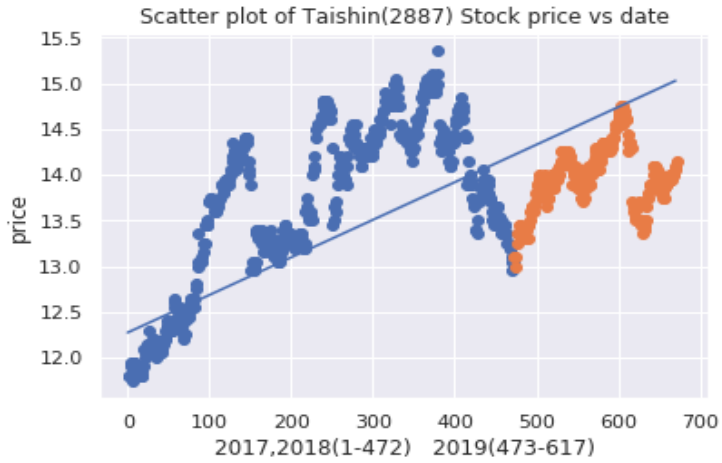
```
Out[14]: 0    13.10
         1    13.10
         2    13.00
         3    13.35
         4    13.25
         Name: Closing price, dtype: float64
```

11. 畫出散佈圖並使用函數來繪出斜率線

這裡我們所繪製的斜率線是採用整體的樣本數與樣本的值來去做繪製。

```
# Creates a linear regression from the data points
m,b = np.polyfit(dateBaseInt, priceBaseInt, 1)

# This is a simple y = mx + b line function
def f(x):
    return m*x + b

plt.scatter(dateBase, priceBase)
plt.scatter(date, price)
plt.plot(f(np.hstack((dateBase, date))))
plt.title('Scatter plot of Taishin(2887) Stock price vs date')
plt.xlabel('2017,2018(1-472)   2019(473-617)', fontsize=12)
plt.ylabel('price', fontsize=12)


print(' y = {0} * x + {1}'.format(m,b))
plt.show()
```

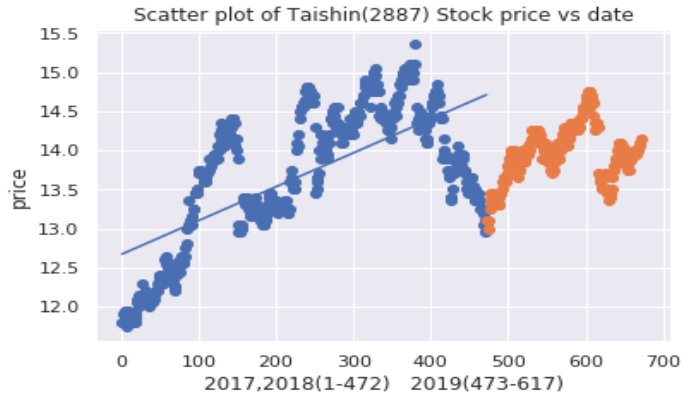
```
y = 0.004110102732026916 * x + 12.269486127604443
```



12. 使用 Scikit Learn 執行線性迴歸

這裡我們使用 Scikit Learn 來幫助我們繪製斜率線，也不用自己來撰寫所需要的線性函式，而 Scikit Learn 其實也提供了不同類型迴歸線的函式，也方便我們製作機器學習模型與分析。

```
# Pick the Linear Regression model and instantiate it
model = LinearRegression(fit_intercept=True)

dateBase=np.array(dateBase) #轉成array而不是list不然無法fit
priceBase=np.array(priceBase)
dateBaseInt=np.array(dateBaseInt)

model.fit(dateBase[:, np.newaxis], priceBase)
mean_predicted = model.predict(dateBaseInt[:, np.newaxis])

plt.scatter(dateBase, priceBase)
plt.scatter(date, price)
plt.plot(dateBase, mean_predicted)
plt.title('Scatter plot of Taishin(2887) Stock price vs date')
plt.xlabel('2017,2018(1-472)   2019(473-617)', fontsize=12)
plt.ylabel('price', fontsize=12)
plt.show()

print(' y = {0} * x + {1}'.format(model.coef_[0], model.intercept_))
```

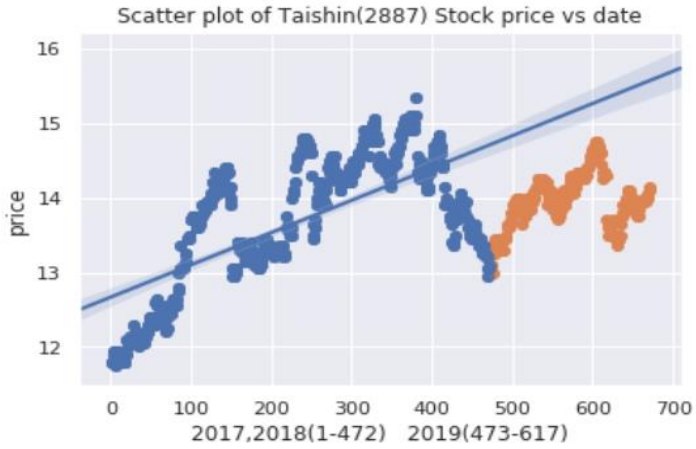



```
y = 0.0043266916571624734 * x + 12.66794505019972
```

13. 以 Seaborn 分析資料

這裡我們使用 Seaborn 繪圖來讓我們迴歸線多了一個預測區塊，讓我們可以來去分析資料可能會分布在該預測區塊中。而在 2019 年的資料與我們從 2017~2018 資料模擬預測出來的迴歸線，2019 的資料似乎低於了我們預測它該有的成長。

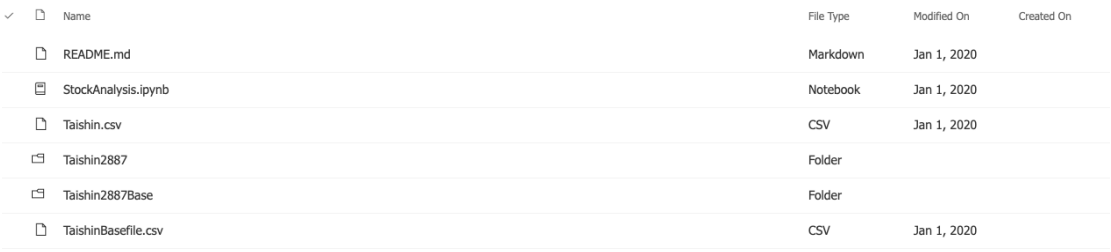
```
plt.scatter(dateBase, priceBase)
plt.scatter(date, price)
#plt.plot(f(np.hstack((dateBase, date))))
#plt.plot(dateBase, mean_predicted)
plt.title('Scatter plot of Taishin(2887) Stock price vs date')
plt.xlabel('2017,2018(1-472)   2019(473-617)', fontsize=12)
plt.ylabel('price', fontsize=12)
#plt.xticks(rotation=120) #x軸項目顯示斜的
#plt.xlim(0, 25) #顯示幾筆數目
#plt.yticks(rotation=120)
sns.regplot(dateBase, priceBase)
plt.show()

print(' y = {0} * x + {1}'.format(model.coef_[0], model.intercept_))
```




```
y = 0.0043266916571624734 * x + 12.66794505019972
```

雲屬性 ：

有別於在 local 端開發，使用 Azure notebook 線上存取資料(如下圖)，不僅不容易丟失資料還可以更清楚的紀錄日期與進度，更可以設定 Public 與他人協作，再利用 jupyter 做開發可以切換 Markdown 等等的語言去做輔助。



Name	File Type	Modified On	Created On
README.md	Markdown	Jan 1, 2020	
StockAnalysis.ipynb	Notebook	Jan 1, 2020	
Taishin.csv	CSV	Jan 1, 2020	
Taishin2887	Folder		
Taishin2887Base	Folder		
TaishinBasefile.csv	CSV	Jan 1, 2020	

問題討論 ：

1. 我們發現迴歸分析太單純其實不太適合拿來分析股票資訊

   因為即使選擇台新金這支相對穩定的股票，依然跌宕的太明顯，需要多層的神經網路來分析才比較不會出現預測錯誤得問題。

2. 爬蟲資料會有資料缺失導致無法準確預測

   台灣證券交易所的資料會有缺失，在預測之前應當先補齊資料再做預測會才不會發現有跨多日但是當成單日成長過多的學習問題。

3. 以日來做時間軸如果遇到跨年月會有非線性分析上的問題

   我們的資料單位是以日來做呈現，但是在做迴歸分析的時候日期跨越年月會有非線性的情況導致無法分析，所以我們改成使用數字的方式來做數量的計算雖然不太適合但是這樣就可以做出簡單的分析。

反饋 ：

Microsoft Azure 在學習上有非常多的資源可以使用，說明也寫得夠清楚，但是 Azure 重在大數據的計算與雲端資源的存取，如機器學習或是資料分析就有很多教學實作，但是相對於 AWS 少了許多線上環境的開發，例如：虛擬實境開發...等等。