X-Village 作業二　陳長明

為何選擇分析這些資料？

市場上常有人說：投資股票九成會賠錢，亦或是散戶都是被坑殺的，因此試著透過股票盤後交易資訊找出重要資訊，例如找出重要的變數，接著將變數透過線性迴歸模型去做預測。

壹、　資料蒐集

由於我們是要做股票分析，因此我們資料來源是從台灣證券交易所中的盤後資訊取得，收盤資訊有許多資料進行其中股票資訊不包含權證、牛熊證、可展延牛熊證。

貳、　資料處理

```
import requests
import pandas as pd
import matplotlib as plt
%matplotlib inline
from io import StringIO

# 將指定日期的股價抓下來，存成csv
response = requests.get('http://www.tse.com.tw/exchangeReport/MI_INDEX?response=csv&date=20180727&type=ALLBUT0999&_=153285233

lines = response.text.split('\n')
newlines = []
for line in lines:
    if len(line.split('",')) == 17:

        newlines.append(line)

df = pd.read_csv(StringIO("\n".join(newlines).replace('=','')))
df = df.astype(str)                     #將list轉為str

def f(s):                               ##################################################################
    return s.str.replace(',','')        #透過這三行讓type為str的表格 將原先表格裡的數字有「,」的部分做轉換#
df = df.apply(f)                        ##################################################################

df = df.set_index('證券代號')

df = df.apply(lambda s: pd.to_numeric(s, errors='coerce'))
df = df[df.columns[df.isnull().sum() != len(df)]]

df.head(5)
```

這份報告主要困難在資料的前處理，首先我們先 import 所需要的套件，如上圖所示，如果直接將 response 印出來看起來相當雜亂，因此透過程式第十行則是將抓下來的資料進行切割，我們想要把它被切開並計算切開後字串被分割成幾個元素，因此透過用「",」切開每一行，看是否被切成 17 個，接著把它放入 newlines 裡，而程式中第十七行解讀順序則是依序為”\n”.join(newlines)，關於它簡例如下"*".join(['abc','efg','hij'])

印出來會變成 abc*efg*hij，然後在用 replace('=', '')把等號給刪除，最後則是用 StringIO 變成檔案，並用 pd.read_csv 來讀取檔案，而

```
29 df.head(5)
```

:

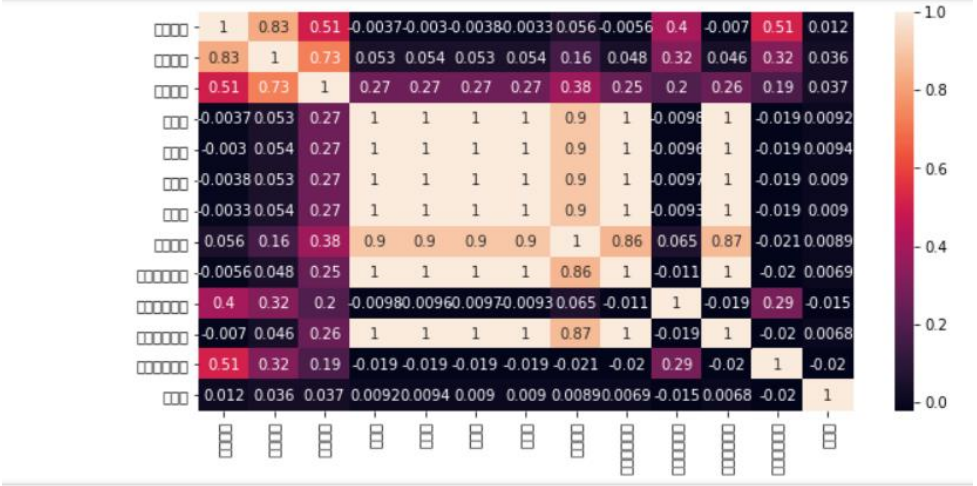
證券代號	成交股數	成交筆數	成交金額	開盤價	最高價	最低價	收盤價	漲跌價差	最後揭示買價	最後揭示買量	最後揭示賣價	最後揭示賣量	本益比
0050	5947118	1942	508730875	85.10	85.75	85.10	85.70	0.70	85.65	185	85.70	46	0.0
0051	17000	17	579410	34.00	34.15	34.00	34.15	0.26	34.12	21	34.20	1	0.0
0052	109500	19	5972049	54.10	54.65	54.10	54.55	0.55	54.40	1	54.65	1	0.0
0053	21000	15	786450	37.30	37.56	37.30	37.56	0.30	37.58	20	37.72	1	0.0
0054	53000	17	1330470	25.03	25.23	25.03	25.23	0.28	25.24	20	25.27	1	0.0

df.astype()則是可以把檔案轉成()中要變成的型態，最後對表格中一些資料做轉換（例如：Index 以證券代號為主，與刪除 NAN 值，最後把型態轉回數字）其結果如下。

因為後續要用到 Matplotlib 視覺化，中文標題還是會呈現亂碼，這使得我必須要把中文變數做轉換，以利後續資料視覺化的標題呈現。

（關於亂碼部分，已經找了好幾種方法也找助教討論過，還是無解，更改到後來還使得圖會跑版。）

```
In [36]: 1 sns.heatmap(df.corr(),annot=True,linewidths=0)
```




```
In [44]:
1  df.rename(columns={ df.columns[0]: "Trade Volume" }, inplace=True)
2  df.rename(columns={ df.columns[1]: "Transaction" }, inplace=True)
3  df.rename(columns={ df.columns[2]: "Trade Value" }, inplace=True)
4  df.rename(columns={ df.columns[3]: "Opening Price" }, inplace=True)
5  df.rename(columns={ df.columns[4]: "Highest Price" }, inplace=True)
6  df.rename(columns={ df.columns[5]: "Lowest Price" }, inplace=True)
7  df.rename(columns={ df.columns[6]: "Closing Price" }, inplace=True)
8  df.rename(columns={ df.columns[7]: "Change" }, inplace=True)
9  df.rename(columns={ df.columns[8]: "Last Best Bid Price" }, inplace=True)
10 df.rename(columns={ df.columns[9]: "Last Best Bid Volume" }, inplace=True)
11 df.rename(columns={ df.columns[10]: "Last Best Ask Price" }, inplace=True)
12 df.rename(columns={ df.columns[11]: "Last Best Ask Volume" }, inplace=True)
13 df.rename(columns={ df.columns[12]: "Price-Earning ratio" }, inplace=True)
14 df.head(5)
```

Out[44]:

證券代號	Trade Volume	Transaction	Trade Value	Opening Price	Highest Price	Lowest Price	Closing Price	Change	Last Best Bid Price	Last Best Bid Volume	Last Best Ask Price	Last Best Ask Volume	Price-Earning ratio
0050	5947118	1942	508730875	85.10	85.75	85.10	85.70	0.70	85.65	185	85.70	46	0.0
0051	17000	17	579410	34.00	34.15	34.00	34.15	0.26	34.12	21	34.20	1	0.0
0052	109500	19	5972049	54.10	54.65	54.10	54.55	0.55	54.40	1	54.65	1	0.0
0053	21000	15	786450	37.30	37.56	37.30	37.56	0.30	37.58	20	37.72	1	0.0
0054	53000	17	1330470	25.03	25.23	25.03	25.23	0.28	25.24	20	25.27	1	0.0

肆、 資料分析與呈現

我們透過表格可以篩選出我們想要的資訊，例如：收盤價／開盤價，其意義是代表股票的漲跌，又或者我們想關注收盤高於開盤 5%以上的個股。

```
close_open = df['Closing Price'] / df['Opening Price']
print(close_open.head(5))
```

```
證券代號
0050    1.007051
0051    1.004412
0052    1.008318
0053    1.006971
0054    1.007990
dtype: float64
```

```
# 選出 收盤 比 開盤 還要高 5% 以上的股票
df[close_open > 1.05].head(5)
```

證券代號	Trade Volume	Transaction	Trade Value	Opening Price	Highest Price	Lowest Price	Closing Price	Change	Last Best Bid Price	Last Best Bid Volume	Last Best Ask Price	Last Best Ask Volume	Price-Earning ratio
1256	375126	326	62882668	155.00	170.50	155.00	169.00	14.00	168.50	1	169.00	5	16.82
1472	99000	51	3360650	33.90	37.05	33.00	35.75	2.05	34.55	6	35.75	1	0.00
1539	5656029	2828	138186529	23.25	25.35	22.55	24.90	1.85	24.85	8	24.90	11	13.76
2338	25494654	11720	823500690	30.60	33.25	30.50	32.65	2.35	32.65	20	32.70	41	181.39
2413	6737391	3439	184193181	25.60	28.35	25.20	28.35	2.55	28.30	3	28.35	140	0.00

在此我想關注漲幅價差的個股，並且將它做排序，由高至低，其資訊與視覺化圖如下：

In [49]:
```
df.sort_values(['Change'],ascending=False).head(5)        #透過我們想關注的資訊做排序(以這為例是透過漲跌價差從大到小作排序)
```

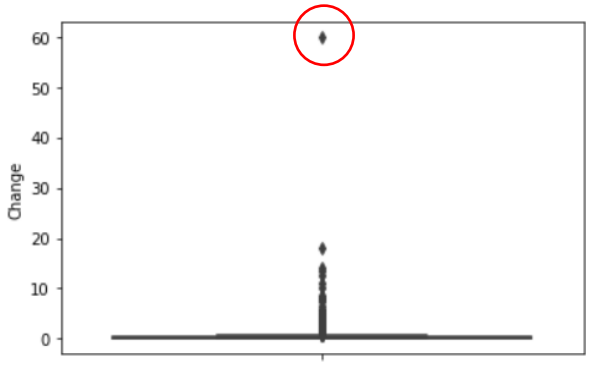
Out[49]:

證券代號	Trade Volume	Transaction	Trade Value	Opening Price	Highest Price	Lowest Price	Closing Price	Change	Last Best Bid Price	Last Best Bid Volume	Last Best Ask Price	Last Best Ask Volume	Price-Earning ratio
3008	265278	385	1388016160	5200.0	5265.0	5175.0	5260.0	60.0	5250.0	47	5265.0	3	28.10
2327	6194182	5309	5714291170	923.0	935.0	911.0	933.0	18.0	932.0	23	933.0	9	31.81
1256	375126	326	62882668	155.0	170.5	155.0	169.0	14.0	168.5	1	169.0	5	16.82
2439	15559887	7299	2348551758	143.5	152.5	143.5	152.5	13.5	152.5	3901	NaN	0	10.95
3406	4133270	3530	1990109220	480.0	488.0	474.0	488.0	12.5	487.5	4	488.0	45	39.35

In [50]:
```
1 import seaborn as sns
2 %matplotlib inline
3 sns.boxplot(y=df['Change'])                    #透過盒鬚圖可以更快看出最明顯的數值為何
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x2cd852426a0>

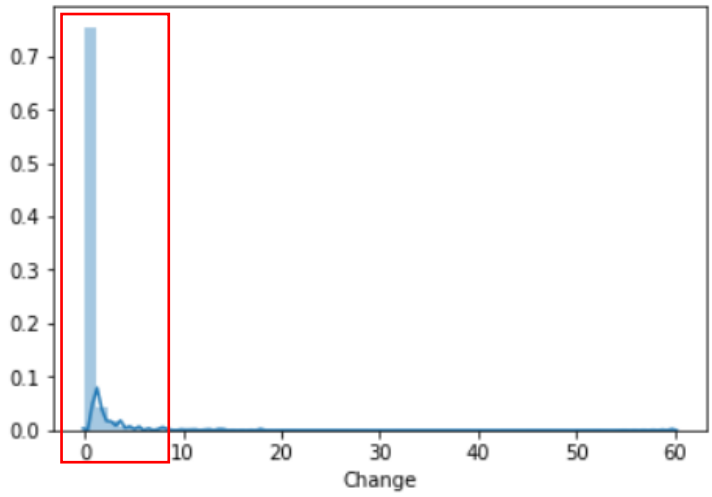
```
C:\Users\user\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1320: UserWarning:
i'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```



In [51]:
```
1 sns.distplot(df['Change'])  #利用distplot來看漲跌價差主要集中的區間
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x2cd85508208>

```
C:\Users\user\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1
i'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```



從關係圖中我們找出高相關性的變數，可以發現價差漲跌與開盤價、最高收盤、最低收盤、收盤價彼此之間互相有關聯，在此介紹主要變數其代表意義：

開盤價：是指開盤時(股票市場開始營業時)的股票價格。

收盤價：是指收盤時(股票市場關門打烊時)的股票價格。

最高開盤價：

股票的價格會隨著買賣雙方所議定的價格(也就是所謂的「成交價」)而有高低起伏所以在一天之中所產生的許多不同的成交價中所曾產生最高的成交價　就叫最高價。

最高收盤價：是指個股在交易日中結束的價格，目前深市是以最後三分鐘買賣撮合成交價作為收盤價格，這樣就可以比較客觀地反映市場價格的變動情況。

價差漲跌幅度：現行漲跌幅 7%限制，已實施多年，融監督管理委員會宣布自 104 年 6 月 1 日起，將漲跌幅度由 7%放寬為 10%。

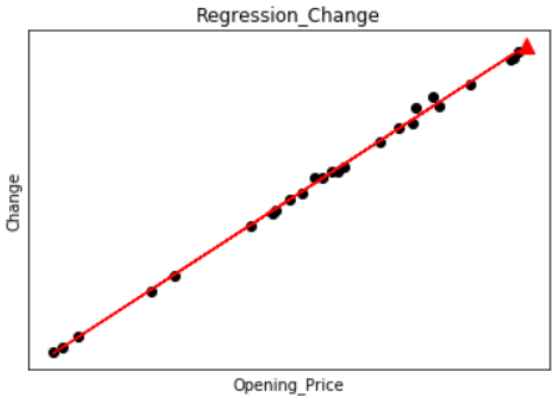
那為什麼我們要特別區分這五種價格？因為你可以很清楚的看到今天這檔股票的趨勢是怎麼走的，並且用你感興趣的價格去做預測(亦或是藉由相關圖找出高相關性的變數)，在此我用前 25 筆資料開盤價與最高價去對漲跌價差去作回歸分析，我們可以發現迴歸為$y = -0.05309621 + 1.01131488x$，在給定 X=10 的情況下，其漲跌價差約略 10%，其視覺化圖形如下:

```
plt.plot(to_be_predicted, predicted_Change, color = 'red', marker = '^', markersize = 10)
plt.xticks(())
plt.yticks(())
plt.xlabel('Opening_Price')
plt.ylabel('Change')
plt.title('Regression_Change')

plt.show()
```

```
係數為: [[1.01131488]]
截距項為: [-0.05309621]
迴歸式: y= [-0.05309621] + [[1.01131488]] x
預測的漲跌價差: [[10.06005263]]
```

```
C:\Users\user\Anaconda3\lib\site-packages\matplotlib\font_manager.py:1320: UserWarning: find
  (prop.get_family(), self.defaultFamily[fontext]))
```



可以發現因為我們所選取的變數是高相關性的，這使得我們迴歸線配適的很密合，為什麼不考慮多元迴歸呢？因為多元迴歸會因為挑選的變數越多，使得R^2越更大，也就是塞越多變數進去，其解釋力只會上升或持平，為了避免這種情況，這使得我考慮一般迴歸找出重要變數，即可做出預測。