

利用新聞建立出隱含波動度指標 (News Volatility Index)

第 11-5 組

財金碩二 r07723067 黃偉倫 財金碩一 r08723006 溫舜元

財金三 b06703009 吳培瑜 會計三 b06702089 陳毅芸

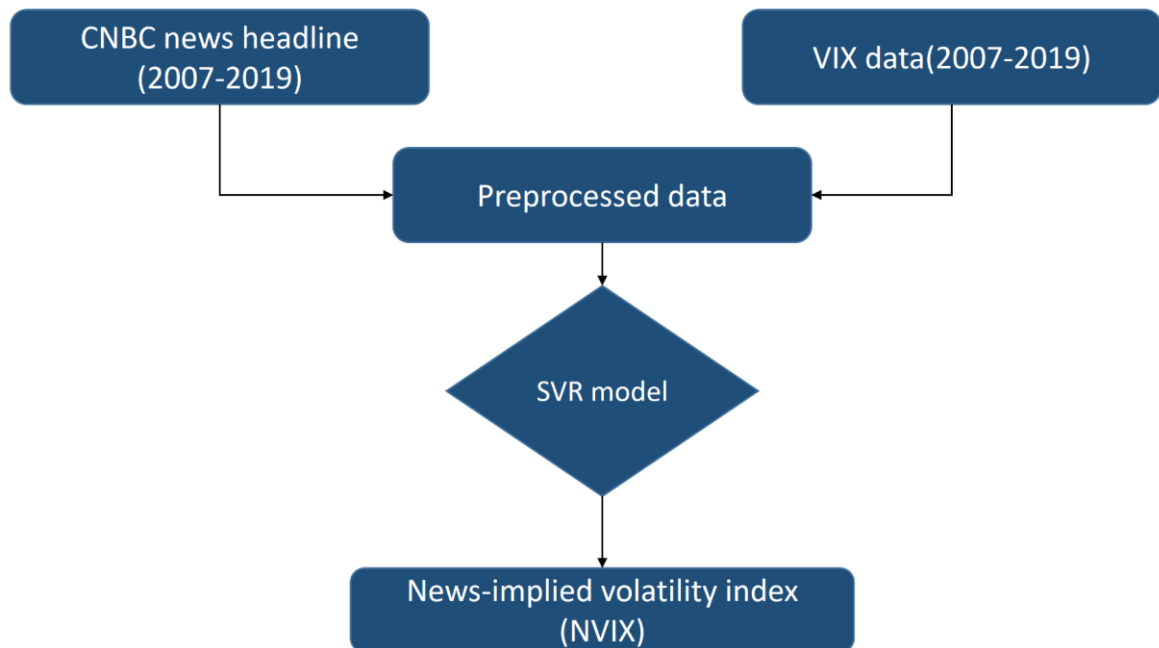
- 專案目標：使用 C N B C 新聞建立出隱含波動度指標 (N V I X)

VIX 指數是用以反映 S&P 500 指數期貨的波動程度，透過 S&P 500 index option 的隱含波動度計算而來，可以反映市場上投資人的情緒。

NVIX 指數是透過新聞的文字內容來衡量市場上投資人的情緒，而建構而成的波動度指標，是基於這樣的假設：

Business press word choice provides a good and stable reflection of average investor's concerns

- 專案架構:



一、資料蒐集- CNBC 網頁爬蟲

我們利用 python 中的 selenium 作為爬蟲的工具。Selenium 是一種自動化測試工具，因為與網頁的交互能力極佳，可以模擬登錄、滑鼠點擊、鍵盤輸入等，所以適合抓取動態頁面的資料數據，而此特性很適合爬取新聞，因為爬取時需要不停點擊與退出並尋找下一篇新聞，而它的缺點則是爬取速度較慢，因為需要等待整個頁面加載完成後才能去爬取頁面元素，接著利用 selenium 的定位元素方法，如 find_elements_by_xpath()、find_elements_by_name() 等等去尋找所需要的資訊。我們一開始是使用是從 CNBC 的 site map 依序點擊進入年、月、日，並抓取全部新聞標題後，再返回前一個頁面並進入下一天，若為當月最後一天則退回該年的月份列表，若為該年最後一個月則退回年份列表。依此循環抓取近 11 年來每日的新聞標題，並在每日新聞標題之前和之後分別輸入年月日和分隔符號。

後來我們發現 CNBC 的網址都非常有規律，如 2019/1/1 發布的數篇新聞皆在 <https://www.cnbc.com/site-map/2019/January/1> 這個網址，其他天的網址也是依此規則，因此我們將程式碼改為用三個迴圈寫出符合形式的年月日，並讓網頁直接 get 到特定日期的網址，再依據 class 定位到要抓取的資訊，抓取完後就直接 get 到下一天的網址，這樣可以省去許多點擊跟退回的步驟及時間，讓程式碼更為簡潔。

另外我們同時也下載 yahoo finance 的 VIX 歷史資料，為後續做準備
資料蒐集程式碼如下：

```
from selenium import webdriver
browser = webdriver.Safari(executable_path='/usr/bin/safaridriver')
browser.get('https://www.cnbc.com/site-map/')
yy = 0
m1st = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']
d1st = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
for y in range(11):
    yy += 1
    mm = 0
    for m in range(12):
        mm += 1
        dd = 0
        for d in range(d1st[mm-1]):
            dd += 1
            browser.get('https://www.cnbc.com/site-map/%d/%s/%d/' % (2020-yy, m1st[mm-1], dd))
            articlelst = browser.find_elements_by_class_name('SiteMapArticleList-link')
            articlelst = [i.text for i in articlelst]
            with open('CNBC_TITLE.txt', 'a') as f:
                f.write(str(2020 - yy) + '/' + str(mm) + '/' + str(dd) + '\n')
                for i in articlelst:
                    f.write(i)
                    f.write('\n==\n')
browser.quit()
```

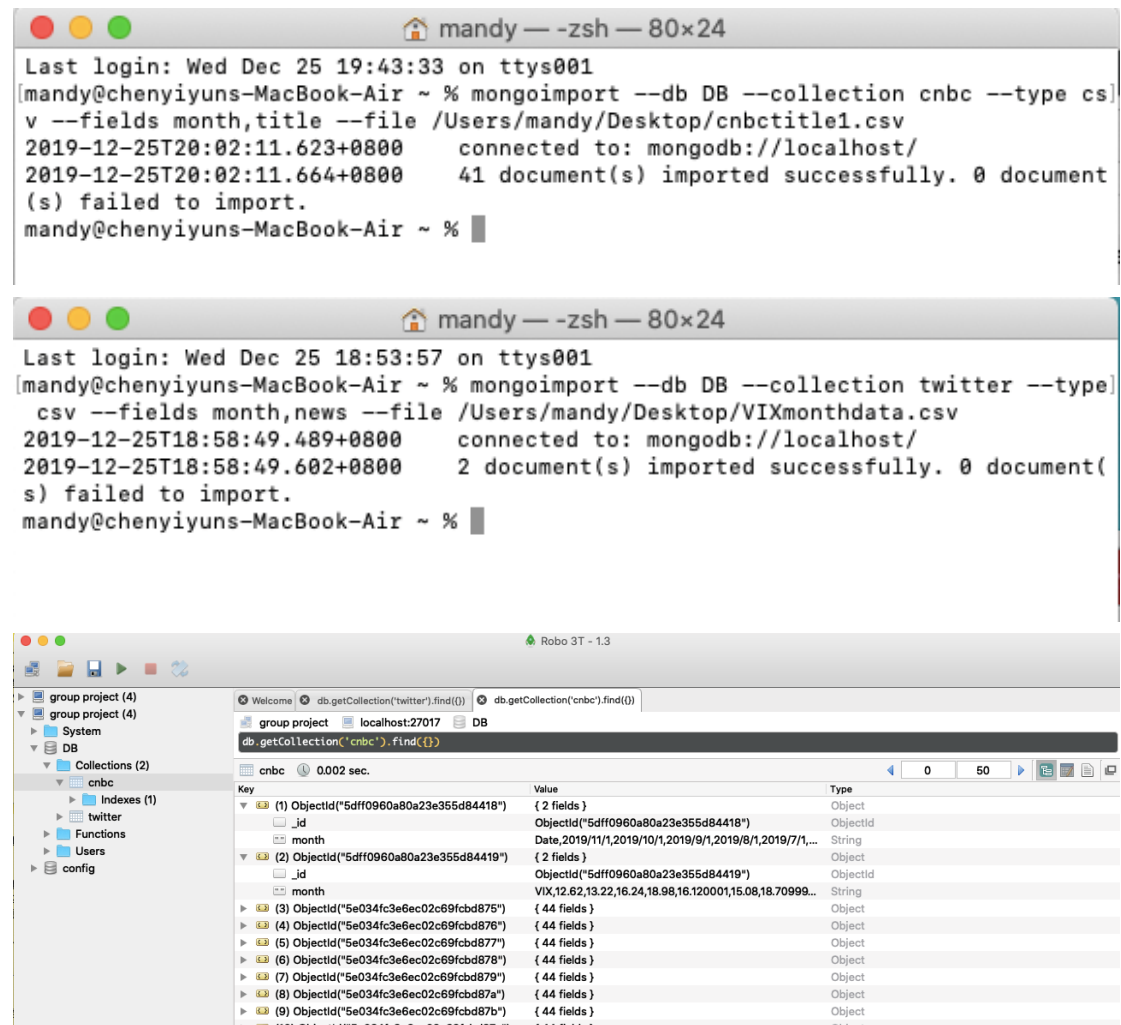
二、資料儲存

MongoDB 是非結構式資料庫的一種，儲存類型為 Documents，一般會使用 Json 格式儲存。爬蟲下來的資料最簡單的方式為存成 txt 檔，然而資料讀取上可能會有一些問題；或者可以存成 Json 檔，但是資料量龐大時，在查詢方面可能會有效率上的疑慮，而 MongoDB 提供了很好的解決方法，且能夠快速地更新。

在使用上，我們已本機端的安裝為主，與其他程式比較不同的是，安裝結束後需要啟

動 server 才算成功安裝，所以程序上較複雜一些。另外還有安裝 MongoDB 的視覺操作頁面---Robo3T，以利於直接在 collection 下創建資料夾儲存資料。

就匯入爬蟲抓取的 CNBC 頭條及 VIX month data 而言，先將存取下來的 txt 檔轉成 csv 檔，並藉由 mongoimport 匯入資料庫。



The image shows two terminal windows and a Robo3T interface. The first terminal window shows the command to import CNBC data into the 'cnbc' collection. The second terminal window shows the command to import VIX month data into the 'twitter' collection. The Robo3T interface shows the 'cnbc' collection in the 'DB' database, with a list of documents and their fields.

```
Last login: Wed Dec 25 19:43:33 on ttys001
[mandy@chenyiyuns-MacBook-Air ~ % mongoimport --db DB --collection cnbc --type csv
v --fields month,title --file /Users/mandy/Desktop/cnbctitle1.csv
2019-12-25T20:02:11.623+0800    connected to: mongod://localhost/
2019-12-25T20:02:11.664+0800    41 document(s) imported successfully. 0 document
(s) failed to import.
mandy@chenyiyuns-MacBook-Air ~ %
```

```
Last login: Wed Dec 25 18:53:57 on ttys001
[mandy@chenyiyuns-MacBook-Air ~ % mongoimport --db DB --collection twitter --type]
csv --fields month,news --file /Users/mandy/Desktop/VIXmonthdata.csv
2019-12-25T18:58:49.489+0800    connected to: mongod://localhost/
2019-12-25T18:58:49.602+0800    2 document(s) imported successfully. 0 document(
s) failed to import.
mandy@chenyiyuns-MacBook-Air ~ %
```

Robo 3T - 1.3

group project (4)

group project (4)

System

DB

Collections (2)

cnbc

Indexes (1)

twitter

Functions

Users

config

Welcome

db.getCollection("twitter").find()

db.getCollection("cnbc").find()

group project localhost:27017 DB

db.getCollection("cnbc").find()

cnbc 0.002 sec.

Key	Value	Type
(1) ObjectId("5dff0960a80a23e355d84418")	{ 2 fields }	Object
_id	ObjectId("5dff0960a80a23e355d84418")	ObjectId
month	Date,2019/11/1,2019/10/1,2019/9/1,2019/8/1,2019/7/1,...	String
(2) ObjectId("5dff0960a80a23e355d84419")	{ 2 fields }	Object
_id	ObjectId("5dff0960a80a23e355d84419")	ObjectId
month	VIX,12.62,13.22,16.24,18.98,16.120001,15.08,18.70999...	String
(3) ObjectId("5e034fc3e6ec02c69fcbd875")	{ 44 fields }	Object
(4) ObjectId("5e034fc3e6ec02c69fcbd876")	{ 44 fields }	Object
(5) ObjectId("5e034fc3e6ec02c69fcbd877")	{ 44 fields }	Object
(6) ObjectId("5e034fc3e6ec02c69fcbd878")	{ 44 fields }	Object
(7) ObjectId("5e034fc3e6ec02c69fcbd879")	{ 44 fields }	Object
(8) ObjectId("5e034fc3e6ec02c69fcbd87a")	{ 44 fields }	Object
(9) ObjectId("5e034fc3e6ec02c69fcbd87b")	{ 44 fields }	Object
(10) ObjectId("5e034fc3e6ec02c69fcbd87c")	{ 44 fields }	Object

事實上，本應是透過 python 去操作資料的匯入，但是，我們在爬蟲資料的儲存上格式為 txt 檔，而有資料存取上的問題。所以，學習將資料儲存成 Json 格式並利用其簡單清晰的優點是必要的。

三、資料處理

1. 有了 CNBC 資料之後，我們需要將這些資料進行預處理，這邊我們使用 NLTK 自然語言工具包將句子進行斷詞、將 stopword 如介係詞、代名詞等去除，且進行動詞、名詞的 lemmatization(如將 declining, declined 都還原回 decline，stocks、Stock 等還原成 stock)。程式碼如下：

```

17 def preprocess(text):
18     stop = set(stopwords.words("english"))
19     text = re.sub("[0-9\!%\[\]\,\.\'\'\|\-;\:\(\)\&\$\-\~\?\/]", "", text)
20     text = text.lower()
21     token_words = word_tokenize(text)
22     words = [word for word in token_words if word not in stop]
23     lemmatizer = WordNetLemmatizer()
24     text = [lemmatizer.lemmatize(t, pos='v') for t in words]
25     return text
26

```

2. 接著我們希望使用 word2vector 套件找出詞與詞之間的相關性，把相關性相近的字詞分為同一類並將其頻率加總。我們擁有兩個能夠幫助我們尋找相似詞的詞庫，一為 google 本身就訓練好得到的詞庫以及我們使用維基百科資料透過 word2vector 套件訓練出的詞庫。但是我們發現利用這些詞庫來找 CNBC 詞彙的相似詞似乎有點落差，且我們無法將 CNBC 的所有詞彙依該模型分成獨立的群集，因此我們單純改以論文上的方式，使用 unigram、bigram 的方式建立出詞庫，並將 156 個月中，總出現頻率過低(<10 次)以及過高(>150)刪除，整理出 ngram table，Table 及程式碼如下：

>>>Time Series (2019/12-2007/1)>>>>

1	words	freq2	freq3	freq4	freq5	freq6	freq7	freq8	freq9	freq10	freq11	freq12	freq13	freq14	freq15	fi
40	('save,')	6	5	5	9	5	4	3	7	6	6	6	6	9	5	
41	('billion,')	10	7	6	6	11	6	16	11	12	11	3	9	10	9	
42	('gain,')	3	2	3	2	3	3	3	6	4	4	5	2	4	7	
43	('plan,')	8	8	6	12	9	8	10	11	12	11	10	13	10	7	
44	('rate,')	4	2	11	5	12	15	2	4	4	2	4	5	1	4	
45	('earn,')	10	25	4	9	20	3	13	26	5	15	12	6	4	11	
46	('trade,')	24	33	24	39	21	28	39	8	20	21	19	19	28	22	
47	('deal,')	15	12	8	8	10	8	16	5	14	14	9	10	13	4	
48	('update,')	1	2	1	0	1	0	0	0	1	0	0	0	1	0	
49	('watch,')	8	13	4	6	6	7	3	0	1	1	2	3	3	4	
50	('market,')	27	30	28	29	17	26	23	21	16	18	24	34	35	37	
51	('one,')	11	12	7	10	8	7	5	8	7	4	12	8	10	10	
52	('employees,')	3	1	2	0	2	5	1	1	1	7	2	1	2	2	
53	('make,')	16	26	21	15	19	23	26	21	20	18	32	27	20	23	
54	('open,')	10	5	8	7	3	3	5	2	3	2	2	3	4	3	
55	('let,')	3	1	2	2	1	1	1	1	0	5	3	4	3	2	
56	('china,')	8	19	14	19	6	17	33	9	15	10	11	15	11	13	
57	('push,')	5	2	2	4	3	4	3	2	7	0	3	5	2	1	
58	('company,')	11	13	13	13	19	11	14	10	11	15	16	12	18	12	
59	('people,')	5	5	6	5	3	3	6	10	5	8	8	8	8	6	

```

27 def ngram_table(o):
28     for i in range(2,157):
29         with open('CNBC\\%s.txt'%i, 'r', encoding='utf-8') as f:
30             text = f.read()
31             tokens = preprocess(text)
32             bgs = list(ngrams(tokens, o))
33             fdist = nltk.FreqDist(bgs)
34
35             words=[]
36             freq=[]
37             for a,b in fdist.items():
38                 words.append(a)
39                 freq.append(b)
40
41             globals()['table%s'%i] =pd.DataFrame(list(zip(words,freq)),columns=['words', 'freqs'%i])
42             merge_freq = pd.merge(table2,table3)
43             for k in range(4,157):
44                 merge_freq = pd.merge(merge_freq,globals()['table%s'%k],how="outer")
45                 merge_freq.fillna(0,inplace =True)
46             return merge_freq

```

四、SVR 模型訓練:

使用 Python 的 sklearn 的 SVM 套件中的 SVR 套件來訓練預測 VIX 的模型

```
from sklearn import svm
import numpy as np
from sklearn.svm import SVR
import matplotlib.pyplot as plt
import pandas as pd
```

首先，先讀取前一個步驟所整理好的 n-gram table 與每月 VIX 數值的檔案

```
data = pd.read_csv('/Users/aaronhuang/Desktop/ngram_table.csv', index_col=0).iloc[:,0:].T.astype(float)
vix = pd.read_csv('/Users/aaronhuang/Desktop/VIX month data.csv', index_col=0).T
```

接著，再度對資料做一些調整與篩選：

1. 將資料由舊至新排序

```
data = data[::-1]
vix = vix[::-1]
```

2. 合併 VIX 與 n-gram table，將作為訓練資料（前 120 筆資料）的每個詞與 VIX

算 cor，再將 cor 小於零與 NaN 的字刪除

```
x_train_withVIX = data.iloc[:120,:1]
corr=x_train_withVIX.corr()['VIX']
corr.T

data.loc["corr"]=list(corr.values.reshape(-1))

droplist_1=list(data.loc["corr"][data.loc["corr"]<=0].index)
data=data.drop(droplist_1,axis=1)

data=data.dropna(axis=1)
```

3. 分割訓練資料與測試資料與對應的日期

```
x_test = data.iloc[120:156,1:].values
x_train = data.iloc[:120,1:].values

y_test = vix.iloc[120:156,:1].values
y_train = vix.iloc[:120,:1].values

date = data.index.values
date_test = date[120:156]
date_train = date[:120]
```

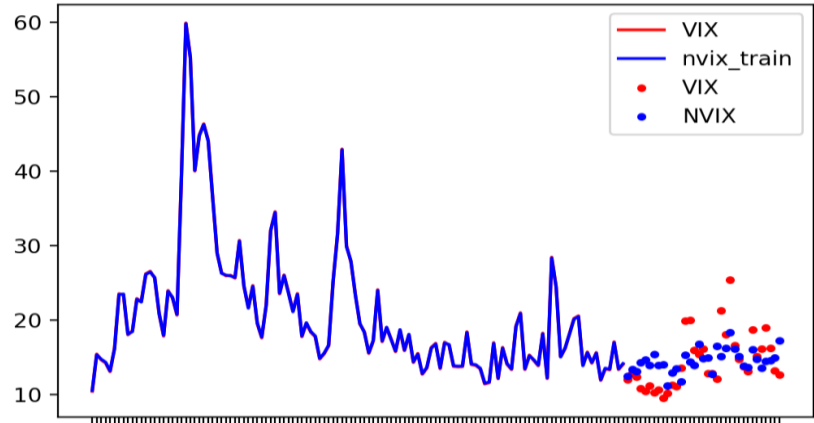
4. 使用訓練資料來訓練 SVR 模型

```
vix_train = svm.SVR(C=1e3, cache_size=2000, coef0=0.0, degree=3, epsilon=0.1,
                    gamma='auto', kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
vix_train.fit(x_train, y_train)
```

5. 用訓練好的模型，輸入測試資料來進行預測

```
vix_test = vix_train.predict(x_test)
```

6. 畫圖來檢視模型訓練的狀況與測試的預測表現



7. 查看模型的分數與參數的權重

```
vix_train.score(x_train, y_train)
```

```
vix_train.coef_
```

Train score	Test score
0.9999	0.2574

Rank	Word	Weight
1	recession	0.0490
2	Obama	0.0432
3	day	0.0426
4	September	0.0426
5	amid	0.0398
6	volatility	0.0395
7	buffett	0.0388
8	time	0.0385

2008/10 Top 10 contribution	
1	('credit',)
2	('bailout',)
3	('round',)
4	('day',)
5	('october',)
6	('buffett',)
7	('time',)
8	('mail',)
9	('pros', 'say')
10	('recession',)

根據我們訓練出的結果，從 Test Score 的結果看來還可以接受；且根據權重表來看，當新聞標題出現 Recession 時，VIX 會有最大幅度增加，符合邏輯。我們認為該專案需要再精進的部分為“將所有程式碼完整整合，連結雲端資料庫”、“詞庫需要隨時間部斷更新訓練，且多試幾種不同的方法比較效果”。