

# PBC期末專案

## Trading Strategy 投資策略機器人

### 第三組

譚庚豪(R10741056)

姜舜馨(R10741005)

劉政儒(R10741040)

林政諺(R10741038)

劉昱呈(R10741025)

# 目錄

- ◆ 專案目標
- ◆ 主要功能與介紹
- ◆ 三大功能的處理程序
- ◆ 串連LINEBot方式
- ◆ 總架設圖
- ◆ 專案心得
- ◆ 專案未來展望

# 專案目標

提供給對不同交易策略有一定程度了解的投資人  
一個快速針對標的來選擇「交易策略」的機器人  
並同時提供該標的相關「基本資訊」  
來減少投資人研究、計算、挑選交易策略的時間

# 主要功能

## 基本面

### 基本資訊

所屬產業  
EPS  
PE ratio  
業務佔比  
等等..

### 新聞

提供該標的最新的  
新聞資訊

## 交易策略

提供該標的針對不同交易策略的報酬率，並與持有到期的報酬率做比較

五大交易策略：  
布林通道、KD、RSI、SMA、MACD

# 基本資訊

## 基本資訊涵蓋

- 公司名稱
- 所屬產業
- 經營項目
- 股價淨值比
- 現金殖利率
- 本益比
- 營收比重

## 爬取

- 以 `request.BeautifulSoup` 抓取以下網頁資訊



- 爬取網頁內容後，利用 `find_all()` 找尋相對應資訊，並抓出該資訊，

## 輸出

- 將結果存為文字輸出，供機器人使用

### 範例

公司名稱: 聯電

產業: 半導體

經營項目: 積體電路各種半導體相關零組件

股價淨值比: 1.73

近5年平均現金殖利率: 4.40%

本益比(同業平均): 7.32 (80.03)

營收比重: 晶圓95.30%、其他4.70% (2022年)

# 相關新聞

## 爬取

- 以`urllib.request`抓取以下網頁資訊
- 爬取網頁內容後，以`json.loads`將`json`格式之資料轉為`Python`字典格式

## 輸出

- 由爬取後儲存資料之字典中取出「十則」新聞連結與新聞標題之資訊，供機器人使用

## 輸出範例

2023/05/20 台積電布局日本、三金控法說、美股13F報告 本周大事回顧  
<https://news.cnyes.com/news/id/5186297>  
2023/05/19 訊芯-KY改選董事 蔣尚義入列候選名單  
<https://news.cnyes.com/news/id/5185624>  
2023/05/19 〈房產〉迎未來五年4.5萬就業人口 高雄套房交易價量齊揚  
<https://news.cnyes.com/news/id/5185522>  
2023/05/19 台股獨家預測：全新主流趨勢為投資者帶來巨大機會！  
<https://news.cnyes.com/news/id/5184100>

# 交易策略

## 爬取

- 利用**yfinance**模組獲取標的物每日的交易資料，包含**OHLC**、調整後**收盤價**及**交易量**等



## 處理

- 準備五項策略，包含布林通道、**KD**指標、**RSI**指標、均線策略、**MACD**指標
- 個別計算買進及賣出的指標及訊號 ( **pandas**、**numpy** )
- 個別最適化策略的參數 ( **for loop** )

## 輸出

- 策略的最適化參數、最適化策略的報酬率及買進並持有的報酬率
- 範例  
最佳天線: 11  
買進持有策略: 26.9244%  
KD策略: 59.6811%

# 串接 LINEBot : 驗證

## 機器人的身分證密碼

- CHANNEL\_ACCESS\_TOKEN 是一種用於驗證和授權的金鑰，用於識別 Line Bot 應用程式並與 LINE 平台進行溝通
- CHANNEL\_SECRET 是在串接 Line Bot 時使用的另一個金鑰，用於驗證和加密訊息以確保安全性

## 進行Messaging API的驗證

- LineBotApi : 利用 CHANNEL\_ACCESS\_TOKEN 驗證
- WebhookParser : 利用 CHANNEL\_SECRET 進行驗證

## 示意圖

```
CHANNEL_ACCESS_TOKEN = "WFG0hWW581sHWFBKDQVjb+X3HxDjI3VIT  
CHANNEL_SECRET = "c6586ddc26e80da8ee220f7945ae73aa"  
line_bot_api = LineBotApi(CHANNEL_ACCESS_TOKEN)  
parser = WebhookParser(CHANNEL_SECRET)
```



# 串 LINEBot 函式

## 串 LINEBot 函式

- 把所有情況條列在 `callback` 函式，偵錯是否有在基本函、交易策略的內容，有就往下層的 `if` 迴圈做判斷

## 程式碼

```
def callback(request):  
    if request.method == 'POST':  
        signature = request.META['HTTP_X_LINE_SIGNATURE']  
        body = request.body.decode('utf-8')  
  
        try:  
            events = parser.parse(body, signature) # 傳入的事件  
        except InvalidSignatureError:  
            return HttpResponseForbidden()  
        except LineBotApiError:  
            return HttpResponseBadRequest()  
  
        for event in events:  
            if isinstance(event, MessageEvent): # 如果有訊息事件  
                if isinstance(event.message, TextMessage):  
                    mtext = event.message.text
```

# 串 LINEBot 呼叫函式：交易策略

## 交易策略

- 判斷使用者輸入中有哪一種策略執行相對應的函式運算
- 以布林通道為例，將使用者輸入的文字轉為計算策略中的所需的變數後呼叫以建立的模型計算，最後吐出結果

```
if mtext == "交易策略":  
    line_bot_api.reply_message(  
        event.reply_token,  
        TextSendMessage(text=RES_TS))  
  
elif "布林通道" in mtext:  
    strategy, stock_id, days, capital, short, long = \  
        mtext.split(" ")
```

利用api擷取使用者輸入「交易策略」

```
# process the data types  
days = int(days)  
capital = int(capital)  
short = int(short)  
long = int(long)  
end_day = dt.datetime.today()
```

轉換輸入格式

```
# call the model  
boll = BollingerModel(stock_id, end_day, days, capital)  
best_window, best_dr, best_sr = boll.optimizer(short, long)  
best_dr = round(100 * best_dr, 4)  
best_sr = round(100 * best_sr, 4)
```

利用建好的模型函式計算

```
res_msg = f"最佳天線: {best_window}\n買進持有策略: {best_dr}%\n布林通道策略: {best_sr}%"  
line_bot_api.reply_message(  
    event.reply_token,  
    TextSendMessage(text=res_msg))
```

輸出格式

利用api丟回line

# 串 LINEBot 呼叫函式：基本函

## 基本函

- 判斷使用者輸入的功能為基本函中為：

- 1.基本資訊
- 2.新聞

若非上述兩者或交易策略  
則回歸起點

```
elif mtext == "基本函":
    res_msg = "(1)欲查詢基本資訊請輸入:\n <股票代號> <基本資訊>\n(2)欲查詢新聞請輸入:\n <股票代號> <新聞>\n(e.g. 2330 基本資訊)"
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=res_msg))

elif "基本資訊" in mtext:
    stock_id = str(mtext.split(" ")[0])
    basic_info = scrawl_info(stock_id)
    # if len(basic_info) > 279:
    #     basic_info = basic_info[:279]
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=basic_info))

elif "新聞" in mtext:
    stock_id = str(mtext.split(" ")[0])
    ns = str(news(stock_id))
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text=ns))

else:
    line_bot_api.reply_message(
        event.reply_token,
        TextSendMessage(text="(1)欲查詢公司基本資訊請輸入: 基本函\n(2)欲使用交易策略請輸入: 交易策略"))
```

基本函 要求資訊

要求資訊=基本資訊  
呼叫已建好爬蟲函式並輸出

要求資訊=新聞  
呼叫已建好爬蟲函式並輸出

非以上需求則重回起點互動

# 架設圖



# Heroku & LINEBot 設置

- 1 在 LINE developer 中創建機器人
- 2 透過 create channel 來建立 messaging API
- 3 取得 channel secret 以及 channel access token
- 4 在 Heroku 創建 APP
- 5 將我們的專案透過 Heroku CLI + Git 的方式部署上去
- 6 透過Git 語法來版本控制並上傳至Heroku
- 7 成功將Bot部署到Heroku上，並取得固定Domain
- 8 將其更新到我們的LINE Channel Webhook URL

**LINE Developers**

 **Messaging API**



TradingStrategy



Heroku Git

Use Heroku CLI

# 使用套件與工具

## 使用套件

Django	typing_extensions
line-bot-sdk	gunicorn
DateTime	whitenoise
numpy	pyfolio
pandas	requests
pandas-datareader	bs4
Yfinance	regex
matplotlib	lxml
	urllib3

## 工具

版本控制



編寫功能



# 專案心得

## 全新的學習

- 第一次爬蟲，花了非常多時間上網查教學和摸索
- 接觸到全新的程式碼，也遇到JSON格式等等
- 第一次串接機器人，也慢慢學習和嘗試，並詢問身邊朋友

## 解決困難

- 在專案的分工上我們進行了較長的討論，後來順離達成共識
- 每次 Debug 的過程都非常辛苦
- 要看懂別人的code也花費很大一番功夫

## 享受成就感

- 一開始並沒有想到真的能做出一個機器人
- 以為無法做出這麼多個交易策略
- 完成的時候成就感非常大
- Debug成功也很有成就感

# 未來展望

## 使用者優化

- 建立line的功能選單，可點選不同功能
- 股票名稱檢索採廣泛比對，不限定用代號
- 將基本資訊量化數值採動態
- 優化交易策略演算速度



開啟 LINE Bank



常見問題



優惠活動

## 新功能

- 增加更多基本資料供點選：法人買賣統計
- 將基本資訊增加圖片，如K線圖
- 將交易策略動態計算出的圖表呈現在line對話中
- 更彈性的交易策略篩選比對：投資組合、策略組合
- 增加更多非個股的金融服務

