中油無鉛汽油預測

CPC Unleaded Gasoline Price Forecasting

資工四 410521206 葉士誠

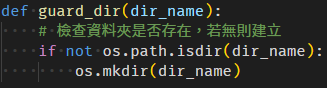
# 動機與目的：

在現今科技發達的社會中，汽油是我們日常生活中，需頻繁使用的一種能源之一，它主要被用於交通工具中提供引擎動力。而它的價格大約一個禮拜公告一次變動，因此有時候想省錢的我都會在想下禮拜的油價是否會漲，若有的話就要在這禮拜先把油加滿，不過如果會跌的話可以下禮拜再加就可以了！因此我想到可以去常去加油的加油站公司—中油(CPC)的網站抓取他們油價的歷史紀錄，這樣我就可以大概預測到下禮拜的油價漲跌趨勢，並決定是否要在這禮拜把油加滿，亦或是等到下禮拜再加！

# 套件與框架：

* Python
  + os
  + requests
  + numpy
  + pandas
  + requests
  + beautifulsoup
  + json
  + matplotlib
  + scikit-learn
  + webbrowser
* HTML
  + Vue.js
  + Chart.js
  + Vuetify.js

# 程式說明：

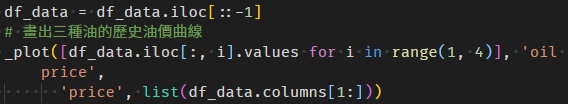
 在最開始的時候我是先檢查我需要的資料夾是否存在，若不存在則建立，都檢查完畢後才正式開始我的主要執行程式。

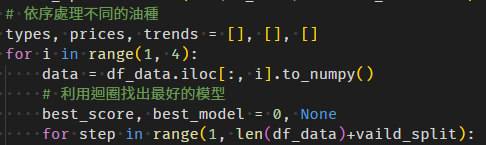
首先，我在載入資料前，我會先決定是否要重抓資料？在我的策略中，因為我要的資料有分成：

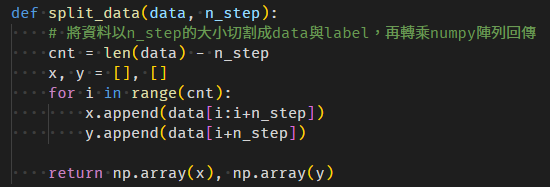
1. 今年的歷史資料(2020以前)
2. 今年前的歷史資料(2019以後)

所以我是每次都會抓今年的資料來更新到最新資訊，而之前的歷史資料則是在遺失時才重新抓，這樣可以才不會因為舊歷史資料龐大的資料量拉長程式執行的時間。接著我就將所有資料合併成一個資料集後，儲存並回傳。

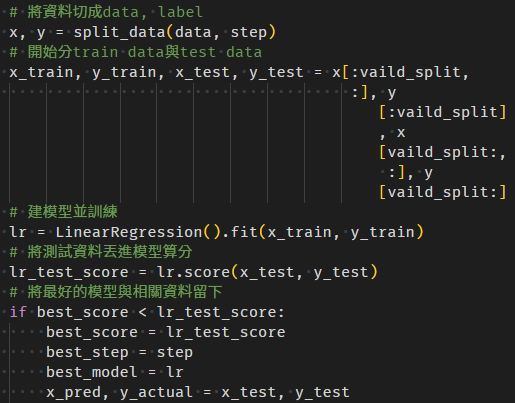
至於抓資料的部分，我是先對網頁取得連線，若請求不成功的話則回傳狀態並終止程式。若可以成功連到網頁的話，先抓取出網頁原碼並分析，再找出每列的名稱以便等等抓資料使用！然後會先計算到底有多少筆資料，接著依序將其以id得到相對應的資料，並記錄下來。而當抓到資料為空時，先轉成nan然後等最後都爬完再一次drop所有nan。在所有需要的資料都抓完、處理完後，就存檔並且回傳資料。

然而，因為資料是從新到舊，但我希望的則是舊到新，這樣才方便預測到下一次新的資料。此外，我抓取的日期是string，不是datetime，且在資料的後半段某幾筆資料中，他們日期也不單純只是的日期而已，還有包含中文字與時間，而我並不需要使用到datetime型態的日期，因此我是直接反轉整個資料，這樣就不需要浪費資源在這部份了！另外，為了方便觀察，我還有把三種油價的走勢曲線畫在同一張途中並儲存。

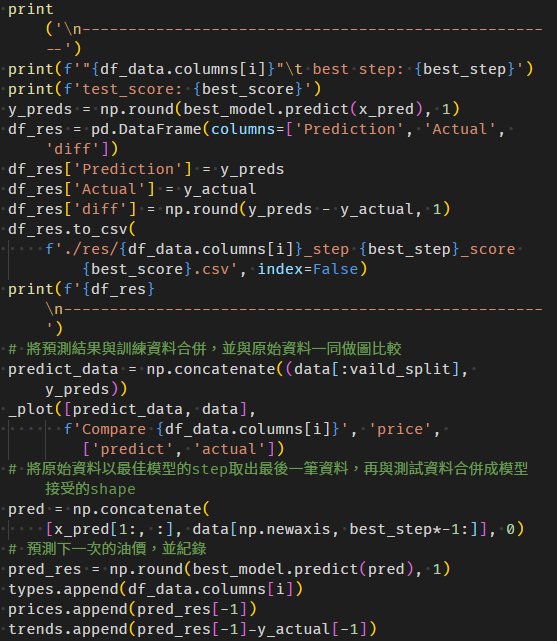
接者，我就開始處理我的資料，我是將不同油種依序處理並訓練。然後利用迴圈將訓練完最後的模型及相關資料存下來，等待處理。



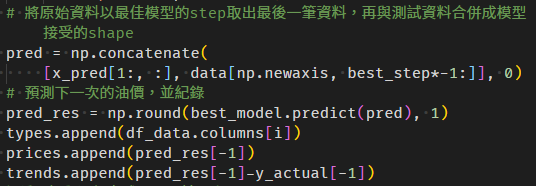
而我會先以時間序的方式將所有資料切成同個step的data與對應的label，然後轉成numpy陣列回傳。



等切完資料與標籤後，我就以9:1的比例將資料分成訓練資料與測試資料，而因為我的資料量算蠻少的，因此只將其取出10%來做等等的驗證。而利用以訓練資料訓練好的模型來算出測試資料的分數來挑選出分數最高的模型當作最終模型。



再來就是將最終模型拿來使用，預測測試資料的結果，並且將其與原始資料做比較，再畫圖視覺化，較方便直接觀察出差異，最後再將資料與圖存檔。而為了要視覺化，比需先將預測出來的資料與前麼的原始資料做合併，然後以不同顏色的方式與原始資料化在同一張圖上，這麼一來就可以輕鬆觀察到差異了。

最後，再將原始最後一筆資料以剛開始丟進模型訓練的shape預測下一次的油價為多少。但也因為shape的關係，因此輸出的部份我們只需要取得最後一筆資料即可。最後會記錄下來，等待所有資料齊全後，再一併存檔並印出。

而UI的部分我是用架網站的方式呈現，而我的方法是先寫出一個html的模板，然後將資料轉成json的形式，並且在模板中特定的字串做替換，將預設字串換成我的資料，然後存成一個新的html。最後再利用webbrowser這個套件自動幫我開啟我的網頁。

至於html模板的部分我是直接套用一些現有的框架，然後寫一點js幫我以框架的需求填入對應的資料。而我也有再自己的ip位置開一個port來架我的[網站](http://134.208.3.147:54147)。

# 開發過程：

我整個系統的開發主要包含：

1. **爬蟲**
2. **資料預處理**
3. **分析、訓練**
4. **預測**
5. **架設網頁**
6. **爬蟲：**

我計畫是先到**台灣中油股份有限公司**抓取[**國內油氣歷史價格**](https://www.cpc.com.tw/cp.aspx?n=92)，然後使用歷史資料來做分析並與預測下一星期的油價漲跌。

剛開始，我研究完它的網頁架構後，就馬上寫爬蟲程式，想把表格對應的值都抓下來，但執行後卻抓不到，經過一番折騰後，我才發現因為它並不是把整個表格寫在現有的網頁架構中，而是透過連結將表格放上去，因此我沒辦法直接將資料抓下來，而是要透過另外的連結去做爬蟲才能抓到我要的歷史資料。

而在抓取資料的過程中，也遇到了一些小問題，像我是利用id來抓各個資料，而每個id前面都一樣，只需改變它是第幾列、哪一個油價就好，但92的那行卻不一樣，因此我需要另外處理！而因為我想要的只有無鉛汽油的資料，因此其他資料就不需要了，但這樣的話因為其他不需要的資料可能是在沒有無鉛汽油的日期，所以在只抓汽油資料的時候變成可能有時會抓到一個空的字串，因此我是將其替換成nan，然後最後再一次把有nan的列欄位刪掉，而因為三種無鉛汽油的調價日期幾乎都是同一天，所以可以不用擔心會有將原就存在的資料丟失的問題。

另外，每次當我要讀取資料進行下一步預處理之前，我都會先去檢查資料是否存在，若沒有的話要重新抓取，不過這只是對比較舊的資料而已，若是新的資料，會是每執行一次就抓一次，因為我需要更新到最新的資料，至於較舊的資料因為不會再被更新且資料量龐大，所以只需在資料遺失時重新抓取即可，這樣能加快程式整體的執行時間。

1. **資料預處理:**

我將抓下來的資料利用時間序的方式將它分成資料與標籤，然後再設定訓練資料與測試資

料的比例並將其分開，這樣就能確保等等的驗證是正確的，並且比較不會有過擬合的問題！

1. **分析、訓練:**

而我只使用中油的歷史油價去做預測是因為這些價格雖然是跟很多因素有關，像是WTI原油油價等等，但我覺得直接單純拿歷史油價去預估可能會比較準確，畢竟這些價錢已經有把很多因素考慮進去，CPC才公告這個價格。

至於訓練的方式我是利用多項式回歸，因為油價其實有蠻多人為因素干擾的關係，因此若使用太複雜的訓練方式可能結果還是與多項式回歸的結果一致，沒有比較好。另外，我還使用迴圈找出哪個模型的測試分數較高，然後就拿來做預測。因為訓練的時候並無測試的資料，因此我覺得當測試資料丟進去模型後，準確度越高的模型，可信度越高！

1. **預測:**

我是以測試分數最好的timestep去取出原始資料的最後一筆資料，然後將同個timestep的測試資料與原始的最後一筆資料合併成模型接受的shape後，丟進去做預測，這樣預測結果的最後一筆資料就是下一次的公告價格。

而我會這樣子取資料然後再預測是因為原本模型就是以經過處理後的shape去訓練，所以我們必須以一樣的shape去預測才行。此外，測試資料實際上只包含到原始資料的倒數第二筆資料，因為最後一筆需要拿來當最後一筆測試資料的答案。

1. **架設網頁:**

網頁的部分我主要是利用Vue.js和Vuetify.js來建立我的html範本，然後利用python讀取範本，將所有資料放進去，像是各個油種的歷史油價和預測結果，再另外存成新的html，然後利用webbrowser的套件將這個新的html打開。

此外，因為實際上我只需要知道下次油價是會漲還是跌，所以我在預測的地方是設計成若預測比最後一次的價錢高才顯示上升的紅色顏文字，不變的話則是灰色的向右顏文字，而價格下跌的話就是向下的綠色顏文字。而我還有特別讓不同油種的icon顏色不一樣，就像是我們平時加油時，看到油槍的顏色一樣。

另外，我還利用了Chart.js製作了線性圖表，而這圖表示各油種的歷史紀錄，只需要將游標移至點上就是顯示出日期與價錢的資訊。

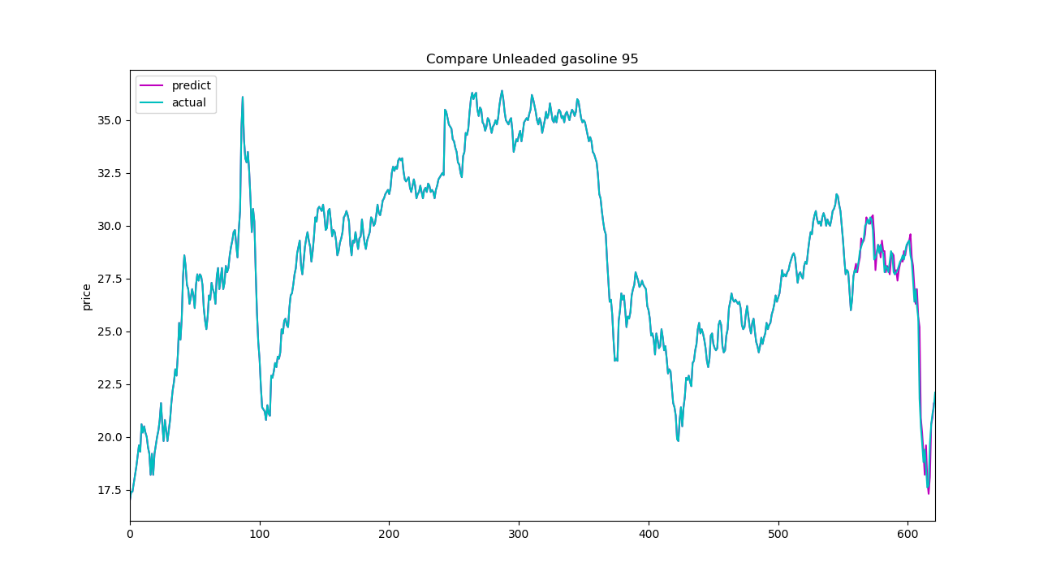
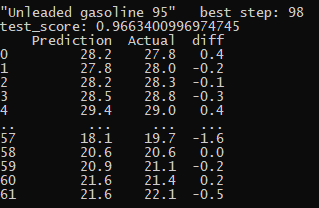
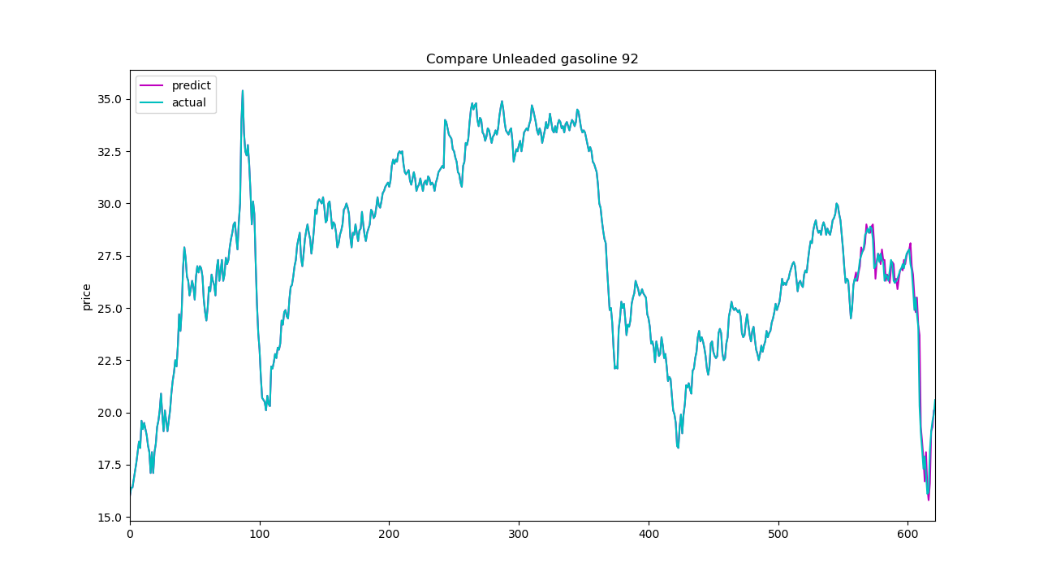
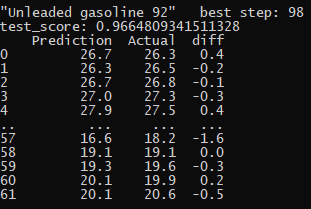
# 網頁功能：

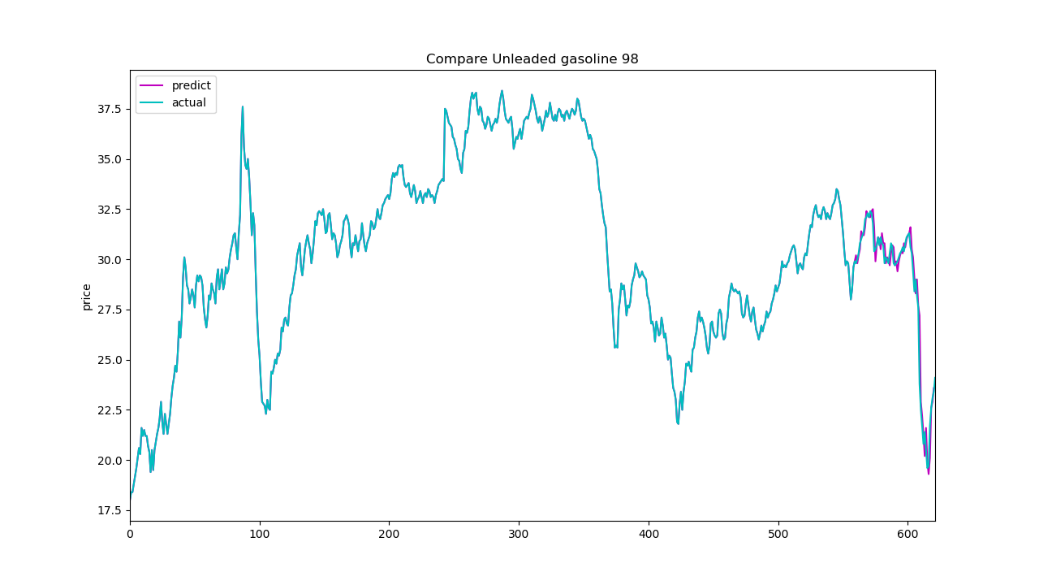
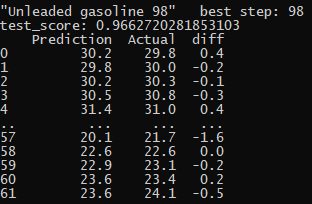
我的網頁可以在左側切換分頁，展示出對應的資訊，像是表格或圖表。而在每個油種中，還可以透過上面的search來找尋符合關鍵字的資料，這樣的話雖然資料很多，但我們還是能快速地找尋到我們要的資料。此外，圖表也類似的功能，只需將游標移到圖表上，他就會自動顯示對應的資訊！

# 成果展示：

1. **資料處理結果：**

這些是不同油種經過訓練與測試後，所得到最好的timestep、測試的分數，以及預測結果和實際結果的資料與差異。





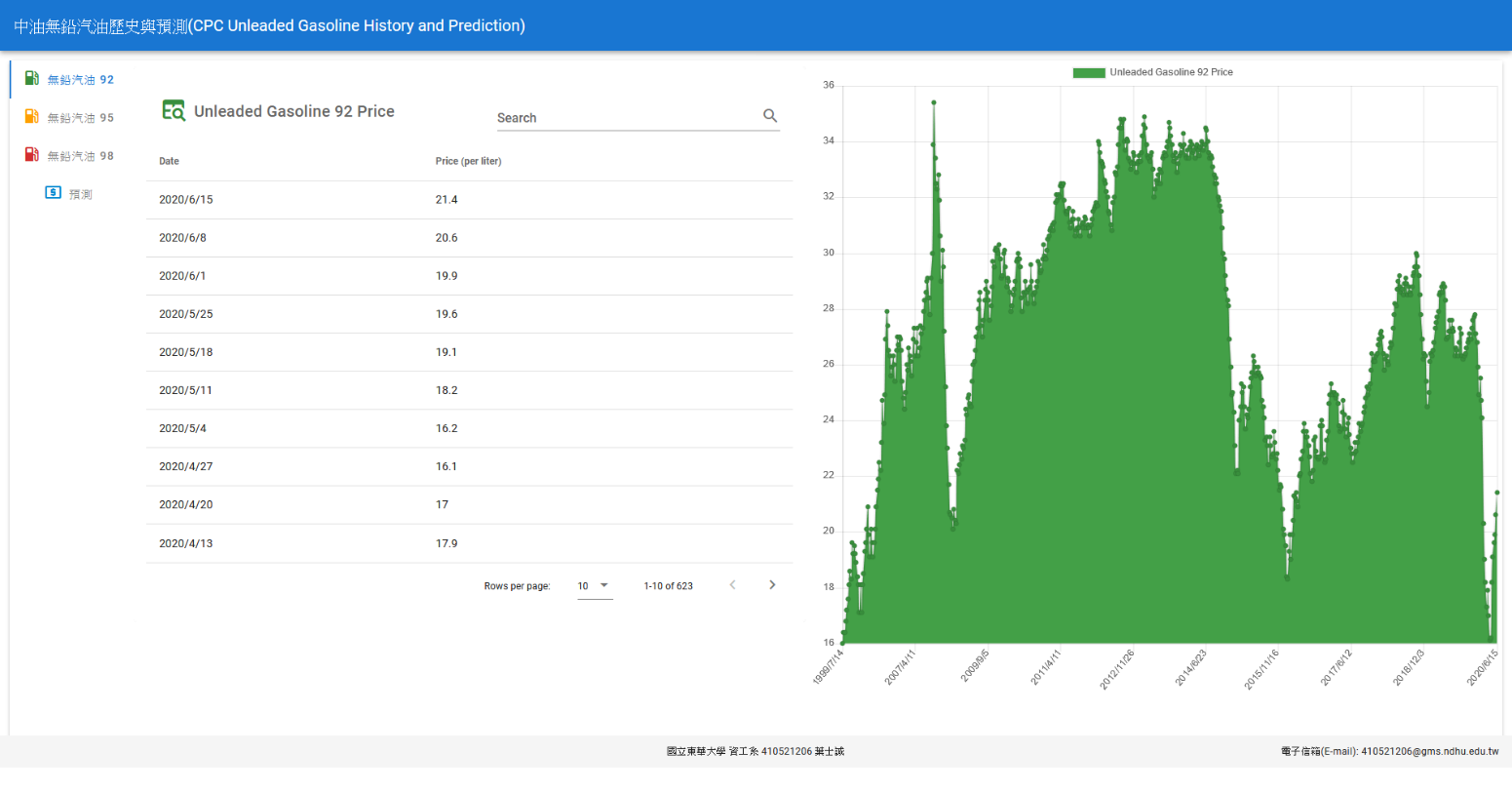
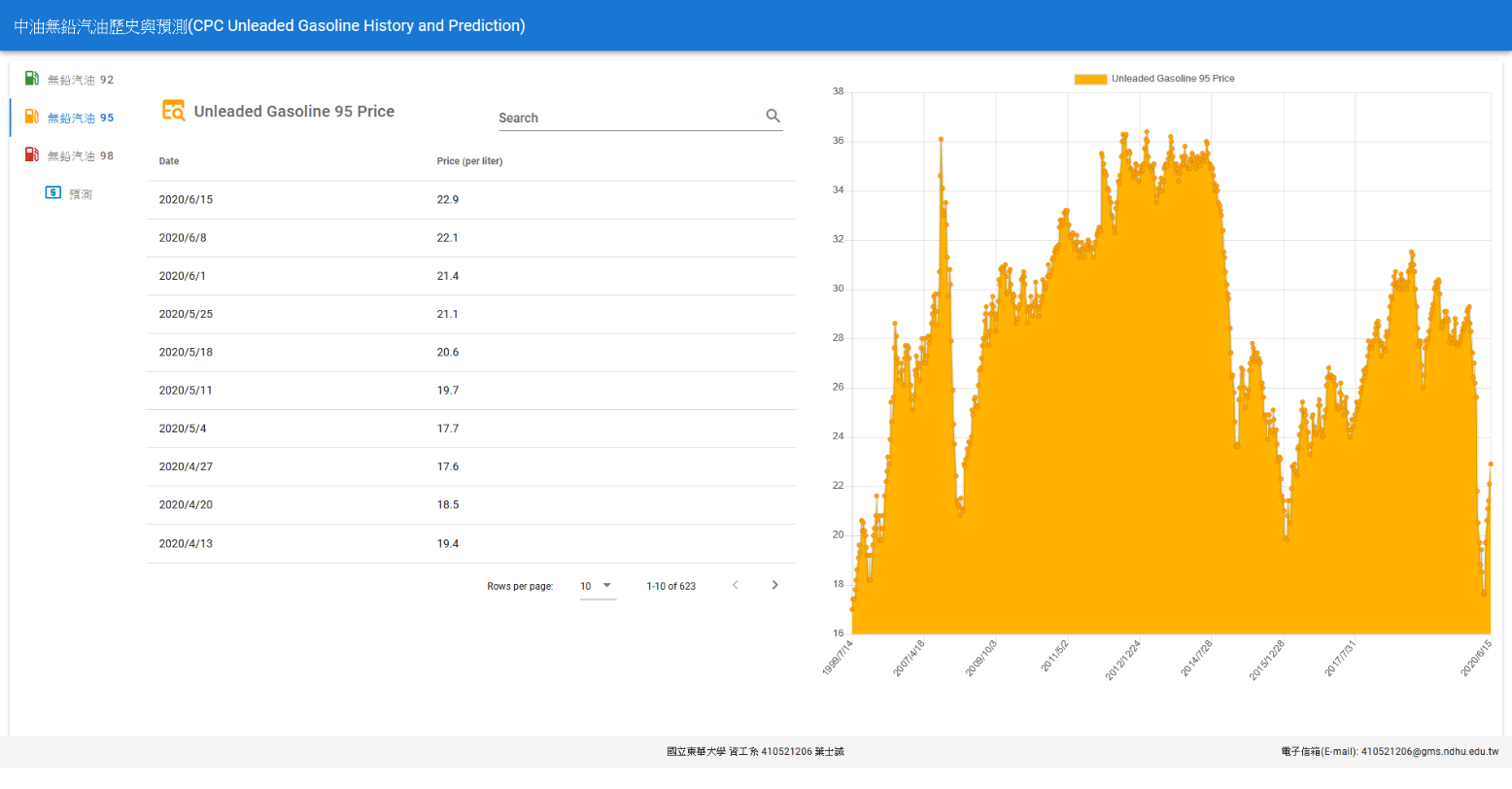
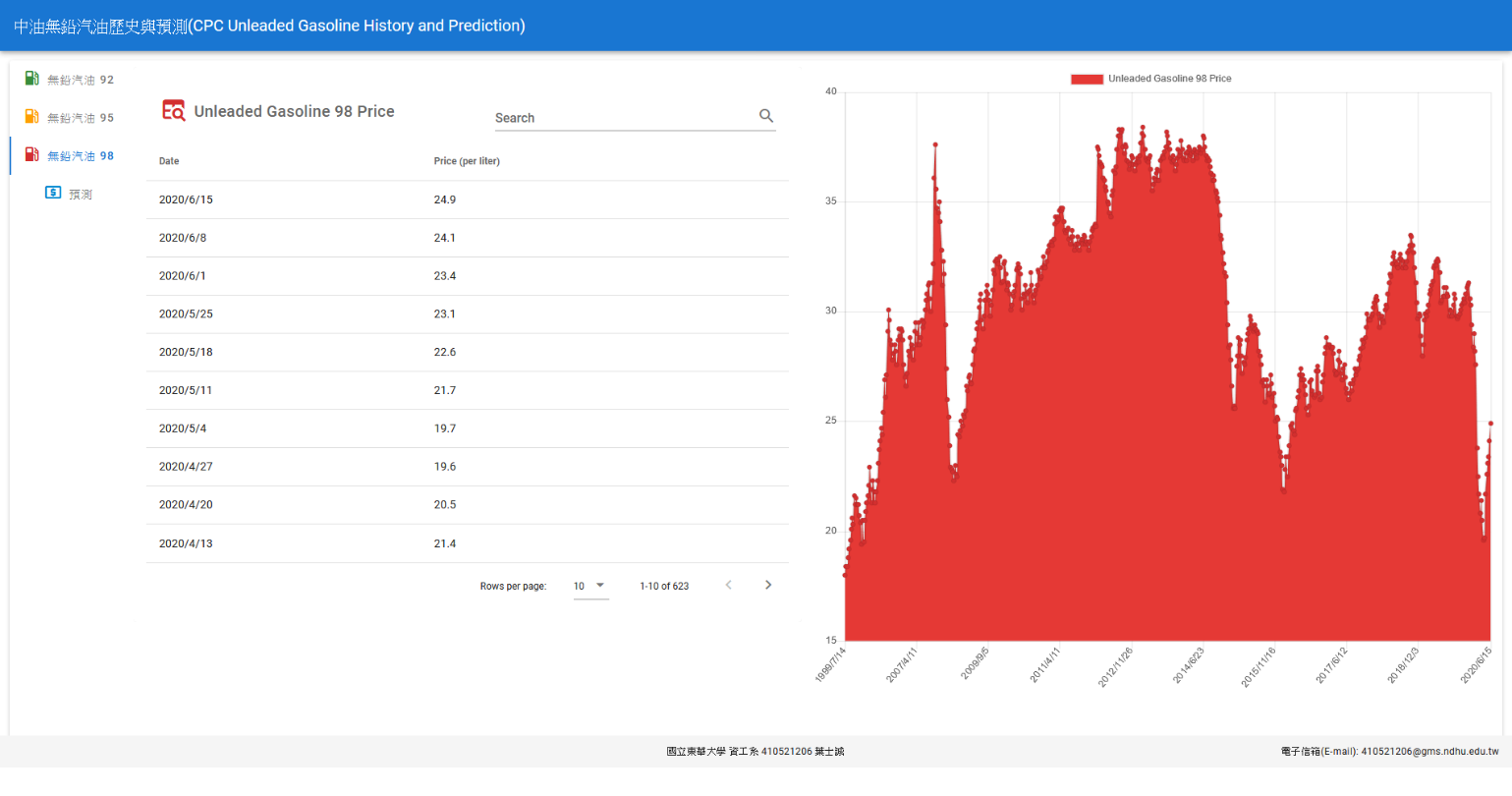
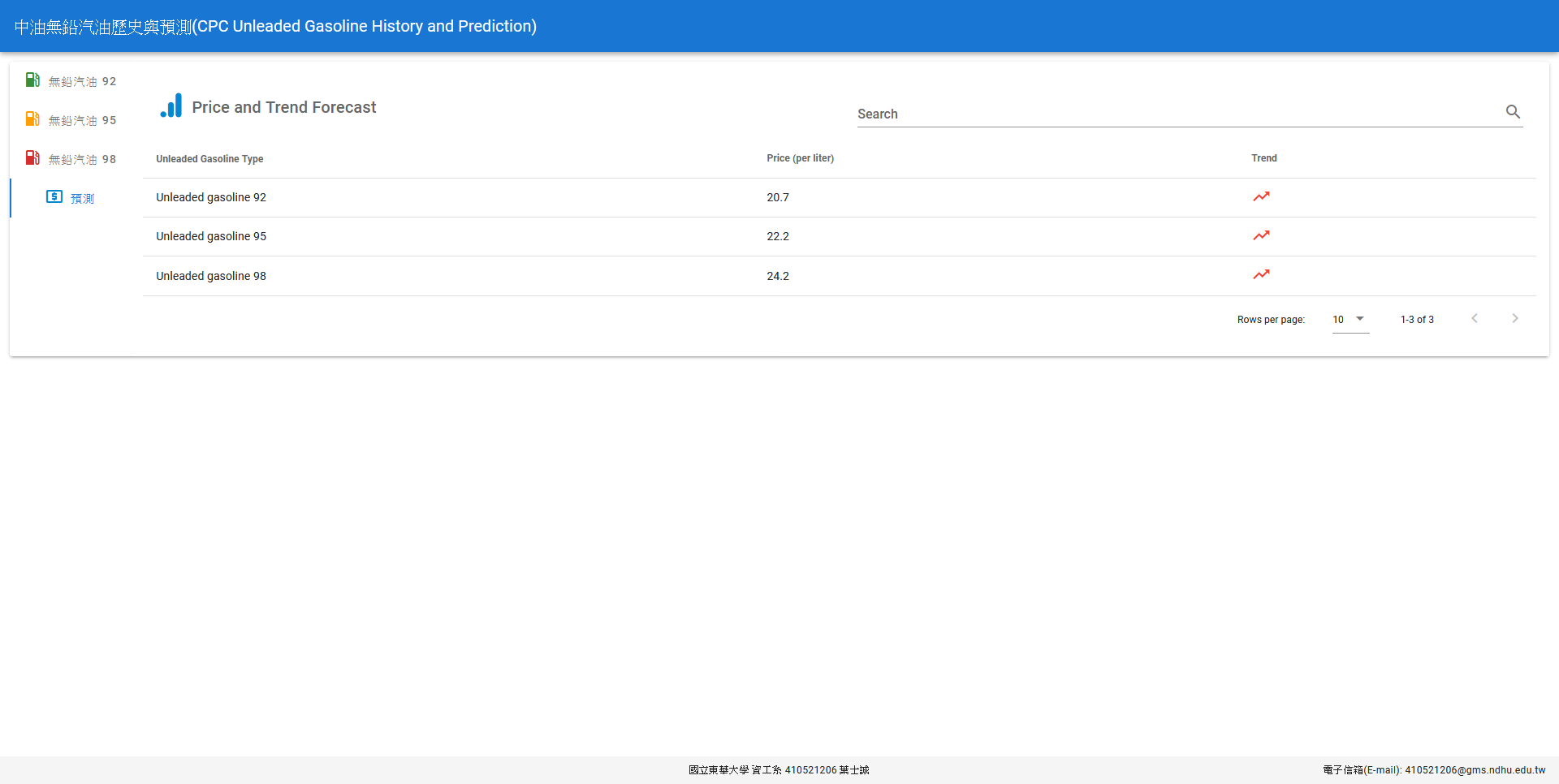
根據上面的圖可以發現，測試預測結果與實際結果十分接近，而各個的測試分數也都高達約97%，所以我覺得該模型預測出來的可信度其實滿高的。另外，因為我的資料量其實不算多，因此只切了10%的測試資料出來。

而這是最後預測結果與最後一筆資料的差。



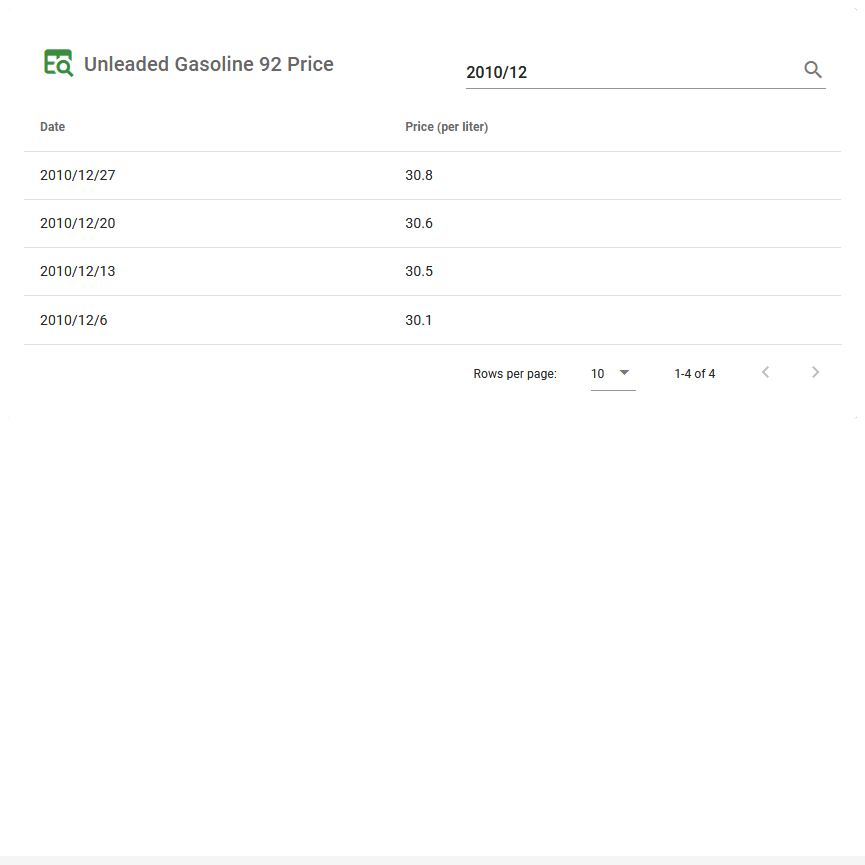
1. **分頁切換：**

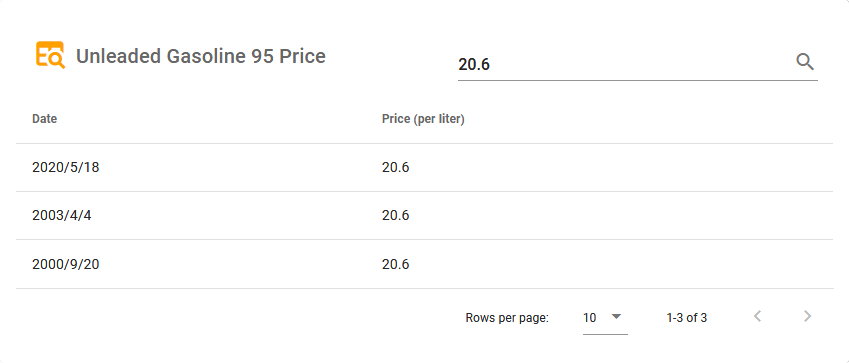
 下面的圖就是屆時跑完python後，會自動彈出的一個網頁。而他可以利用左邊的分頁選項來選擇要切換到哪個分頁，並且顯示出相對應的內容。

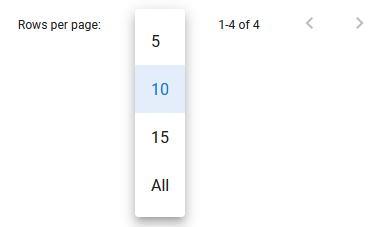
1. **搜尋欄：**

而在每個分頁中都可以利用右圖的搜尋欄來快速地搜索想找的關鍵字。

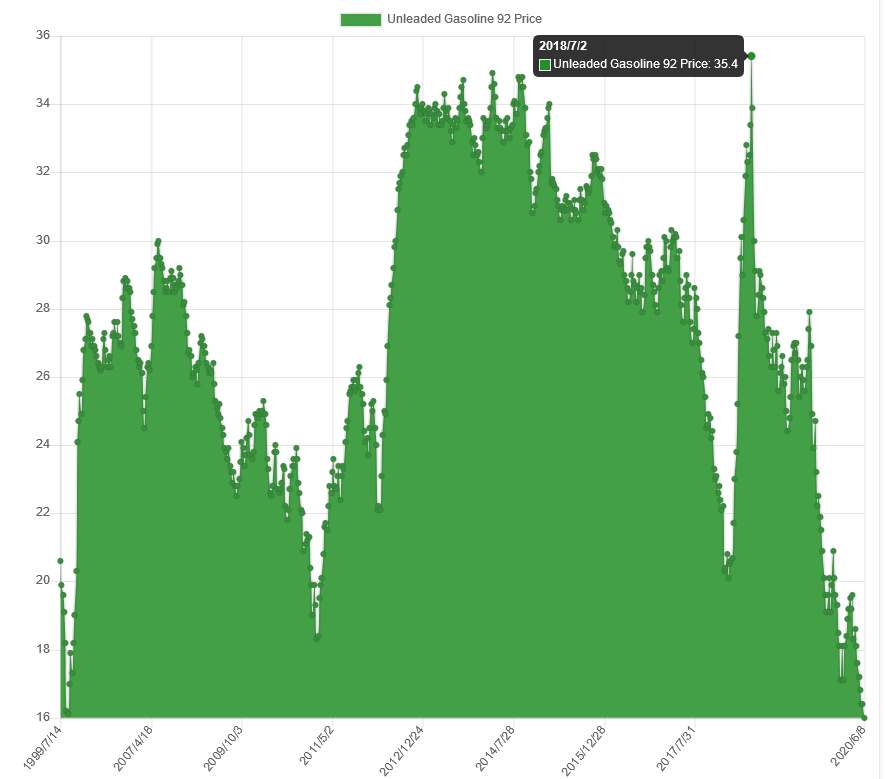




1. **顯示資料的限制：**

我們可以藉由這個設定來決定我們一次要看多少筆資料，若還有其他資料沒有顯示出來，則可以透過右邊的左右按鈕來接著查找下一頁，而旁邊有最大頁數與目前頁數可供使用者觀看。

1. **圖表快速查看資訊：**

****將游標一到點上就可以快速地知道該點的所有資訊。