

資料科學導論 Final Project 報告

依據天氣影響之下周菜價預測平台

課程名稱：資料科學導論

授課老師：李政德

組別：Sorry my bad

組員：何寬羿(C34104032)、林業誠(E24105038)、曾柏誠(E24126717)

一、專案介紹

1. 專案動機



下大雨菜價又漲。(圖 / TVBS資料畫面)

在新聞上，我們常常能看到因為天氣因素，導致菜價浮動、上漲，但是我們一直沒有一個良好的預測菜價的平台；並且，在這學期的資料科學導論課程的競賽學習中，我們多次使用 **LSTM**，也了解到該模型適合用於這類型的預測。因此，有了想解決的問題，也有了有潛力的解決方法，我們決定在本學期的小組期末專案中製作一個能預測菜價的平台。

2. 專案目標

建立一個簡單的使用平台，讓使用者只需要輸入想要預測的日期，平台後端就算透過模型去預測下周菜價給使用者參考。

當然，預測的準確度是最重要的。已知多項因素會影響菜價，這表示預測菜價的困難度很高，但我們希望仍能有不錯的準確度，因此目標定位誤差在 20% 以內。

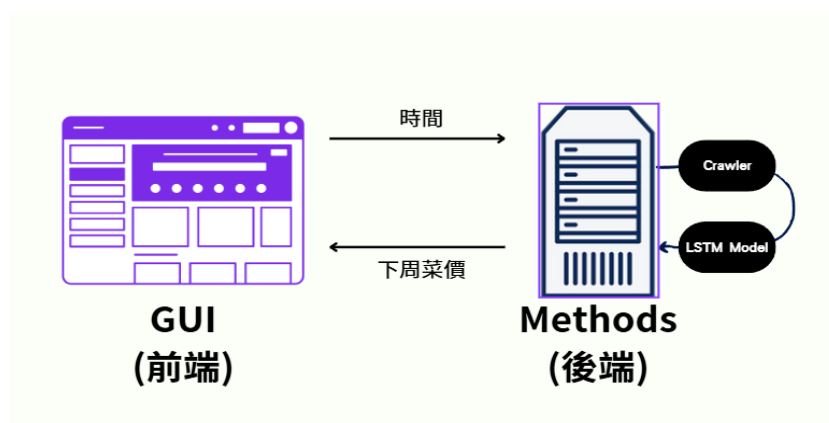
有了這樣的平台，對於需求者而言，能規劃採購時間、數量和食材選擇；對於供應者而言，能優化收成及銷售的計畫；對於政府而言，則能協助作出相關的政策。

3.專案整體架構

「菜價預測平台」分為兩個部分：

(1) GUI

由於時間因素，我們沒有時間搭設完整的前後端平台來讓用戶用以預測菜價。不過，我們製作了簡易的 GUI (模擬前端，詳細操作會在後面介紹)，使用者只需要輸入欲預測菜價的日期，平台會去呼叫方法爬取預測時需要的資訊及使用模型來預測(模擬後端)，以此模擬平台前後端的運行，如下圖。



(2) 模型及訓練

由於交易市場、菜種眾多，在有限的時間內，我們選擇預測「台北一」交易市場的「LF2 薤菜 小葉」平均菜價。(以下皆以薤菜/空心菜簡稱)

- Model：LSTM
- Input：前一周薤菜菜價、前一周全台平均降水量
- Output：下一周薤菜菜價
- Dataset：

我們主要使用 Selenium 來協助爬取需要的資料

Training/Validation/Test Dataset 為 7:2:1，分別使用 2015~2021、2022~2023、2024 年的資料

- 薤菜菜價(from 農業部-田邊好幫手)
- 天氣資料(from 交通部-中央氣象署)
- Evaluation Metrics：RMSE

4.現有研究比較與本組創新

	本組菜價預測平台	現有研究
模型	LSTM	
預測菜種	薤菜(空心菜)	甘藍

資料分析	主要針對天氣	主要針對交易量
特徵採納	天氣 + 近期菜價	僅天氣
Loss Function	RMSE	MSE
預測時間長度	7	5
是否有平台	設有 GUI (有拓展性之平台)	缺乏操作平台
結果呈現	視覺化呈現預測結果	

(整體而言，國內對於菜價預測的研究非常少，本專案具有一定的創新性和實用性)

二、資料收集

1. Selenium 爬蟲程式

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait, Select
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
```

我們在爬取菜價、天氣、颱風資料時，發現使用傳統的 BeautifulSoup 無法正確爬到我們需要的資料因此，我們決定使用 Selenium，透過模擬人在操作網頁的方式來爬取訓練及預測時所需的資料。

以下附上爬取颱風資料時的部分程式碼：

```
# 打開目標網站
url = "https://rdc28.cwa.gov.tw/TDB/public/typhoon_list/"
driver.get(url)

# 等待表格加載完成
time.sleep(5)

# 找到表格元素
try:
    table = driver.find_element(By.ID, 'typhoon_list_table')
    rows = table.find_elements(By.TAG_NAME, 'tr')

    # 提取表格內容
    data = []
    for i, row in enumerate(rows):
        cells = row.find_elements(By.TAG_NAME, 'th' if i == 0 else 'td')
        row_data = []
        for cell in cells:
            text = cell.text.strip()
            # 處理換行符號：颱風生命期間用 ~，颱風名稱移除換行
            if i > 0: # 跳過標題行
                if "颱風生命期間" in row_data: # 假設對應到特定欄位
                    text = text.replace("\n", "~")
                elif "颱風名稱" in row_data:
                    text = text.replace("\n", " ")
            row_data.append(text)
        if row_data:
            data.append(row_data)
```

以下為爬取到的歷年颱風資料(僅附上部分)：

	年份	颱風編號	颱風名稱	颱風生命期間	\	
0	2024	202426	帕布(PABUK)	2024-12-23 06:00~2024-12-25 00:00		
1	2024	202425	天兔(USAGI)	2024-11-11 18:00~2024-11-16 03:00		
2	2024	202424	萬宜(MAN-YI)	2024-11-09 06:00~2024-11-20 00:00		
3	2024	202423	桔梗(TORAJI)	2024-11-09 06:00~2024-11-14 06:00		
4	2024	202422	銀杏(YINKING)	2024-11-03 18:00~2024-11-12 06:00		
	颱風生命期間中心最低氣壓(hPa)		颱風生命期間中心最大風速(m/s)	颱風生命期間最大7級風暴風半徑(km)		\
0	1000		18	80		
1	945		45	150		
2	900		60	180		
3	965		38	150		
4	930		51	180		
	颱風生命期間最大10級風暴風半徑(km)		警報發布報數			
0	---		---			
1	50		19			
2	90		---			
3	50		---			
4	90		---			

2. 填補闕漏值

中央氣象署-每日雨量資料中有註明：若表格中雨量為"T"代表降水量小於 0.5mm，若雨量為"X"代表無紀錄值或儀器故障。

在農業部田邊好幫手-農產品交易行情中，也能發現有部分天數是查詢不到菜價的，如下圖的 2024 年 12 月 26 日就查無菜價資料。

關鍵字或代號查詢

交易日期區間

LAI

113/12/26

113/12/26

搜尋

列出全部

請勿輸入特殊字元（標點符號及空格）輸入名稱或代號，例如：鳳梨、B4

交易日期	類別	名稱	市場	上價	中價	下價	平均價	交易量
查無資料								

由於以上因素，我們爬下來的資料存在許多闕漏值。我們採用以下方法填補資料缺漏：

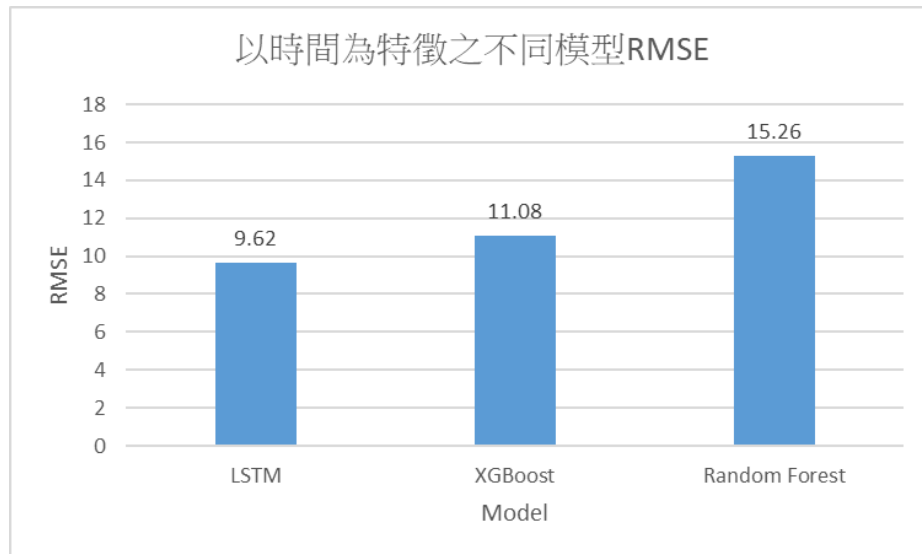
- (1) 對於降水量為"T"(小於 0.5mm)的數據，我們將其視為降水量 0.0mm
- (2) 對於降水量為"X"(無紀錄值或儀器故障)的數據以及無菜價紀錄之日期，我們使用內插法填補。

三、模型訓練

尋找最適合的模型

許多在價格預測上的研究都會選擇使用 LSTM，但是在資料科學導論課程的兩次競賽學習中我們發現 XGBoost 常常都有不錯的表現，因此也納入比較。

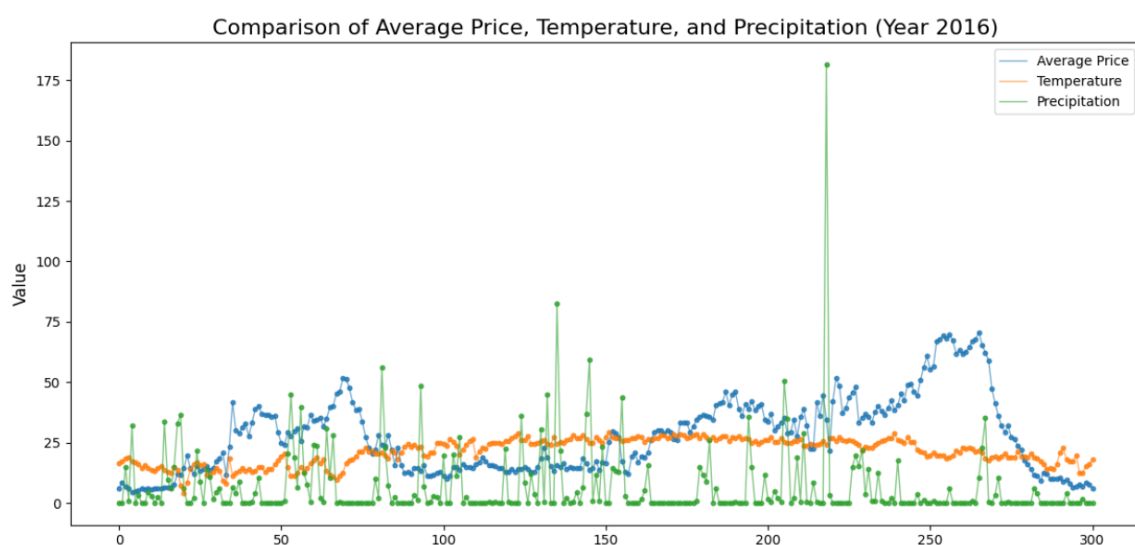
我們以簡單的機器學習模型 Random Forest 模型作為 Baseline，比較 LSTM、XGBoost、Random Forest 三種模型在相同 Dataset、訓練方式(皆以「時間」為唯一特徵)下在 Test Dataset 的 RMSE。從下圖可以得知，LSTM 在三者中具有最佳表現，因此我們選擇 LSTM 作為後續預測主力。



(皆為針對「甘藍」平均菜價之預測)

尋找適合的特徵

經過分析多種特徵對於菜價的影響後，我們發現降水量和溫度這兩個天氣特徵對於菜價的影響較大，因此初步鎖定這兩個因素做為特徵。此外，由於颱風通常會造成大雨，這與降雨量特徵重疊，因此我們後來選擇排除掉颱風特徵。



(上圖為高麗菜 2016 年的價格、溫度、降雨量折線圖)

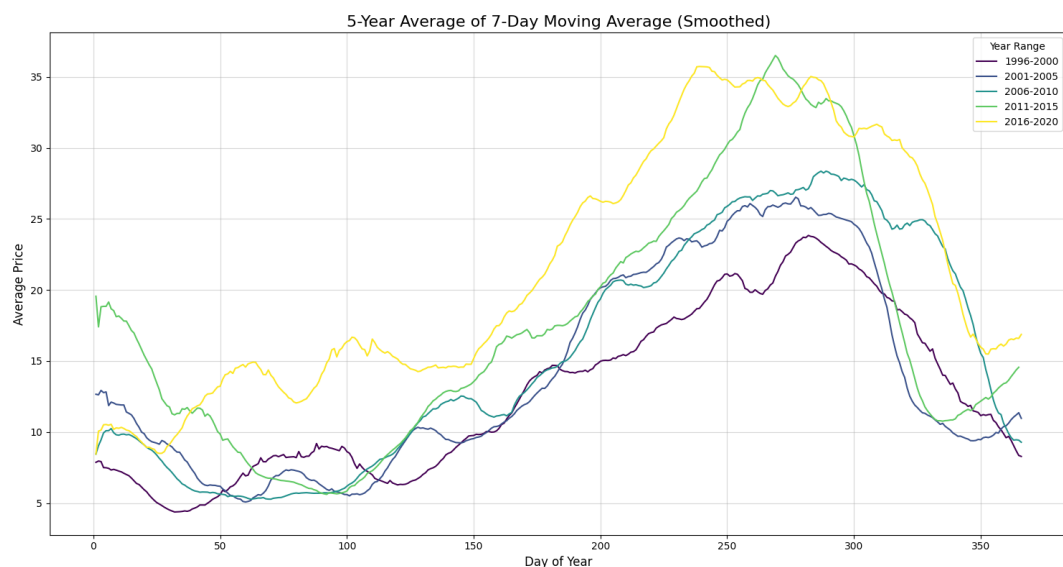
詳細分析、深入訓練

影響價格的變因

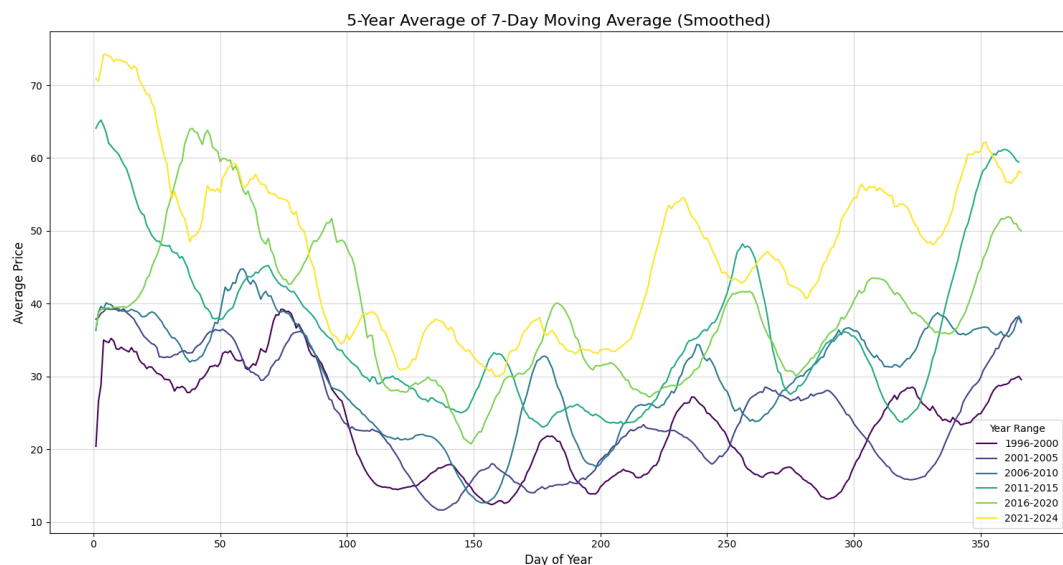
在將價格用圖表分析過後，我們發現影響價格的因素不單單只有颱風、寒流等因素，更包含諸多變因：

1、果菜種類的市場特性

蔬菜價格隨時間變化雖然聽似簡單，但其實內涵相當多的細節，例如是否為長年性的植物、是否為一年一獲或是多獲，產季與非產季的落差，是否擁有替代品種，都大幅影響菜價(而且在政策層面，部分蔬果會有農民搶種而引發的市場動盪)。



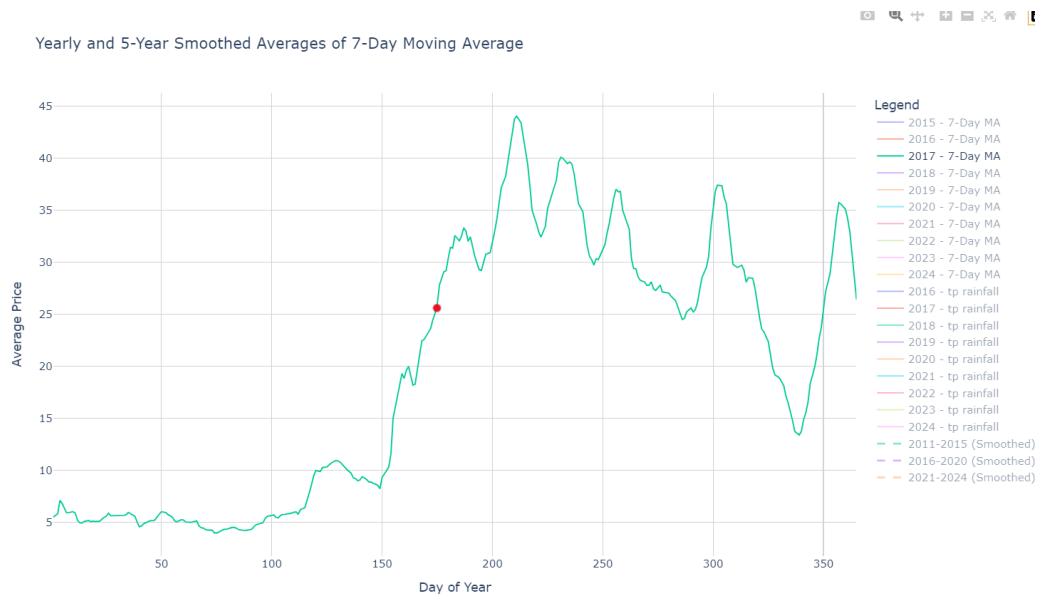
(上圖為高麗菜(品種:初秋)歷年價格，可以明顯看出在秋冬之際價格最高，春夏之際最低)



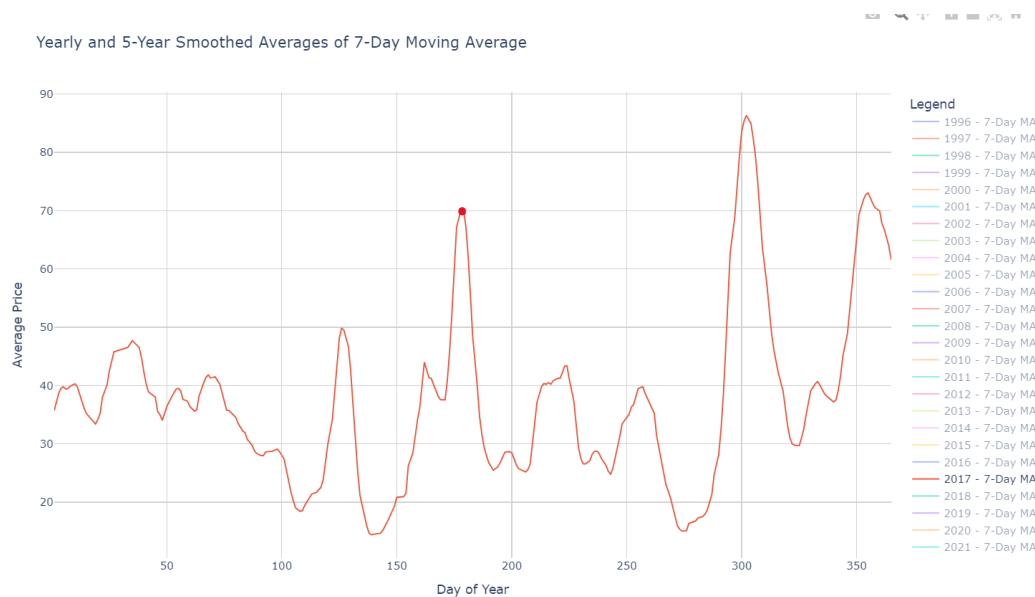
(而上圖為空心菜的歷年價格，可以看出部分除了春季些微較高外，無明顯趨勢)

2、植株受天災的影響程度

不同的植株受到天災的影響程度也相差甚大，例如空心菜、波菜等較脆弱的葉菜類受到風雨的影響也比高麗菜大(下圖為 2017 的菜價走勢，紅點標記處發生了接連的豪雨)。



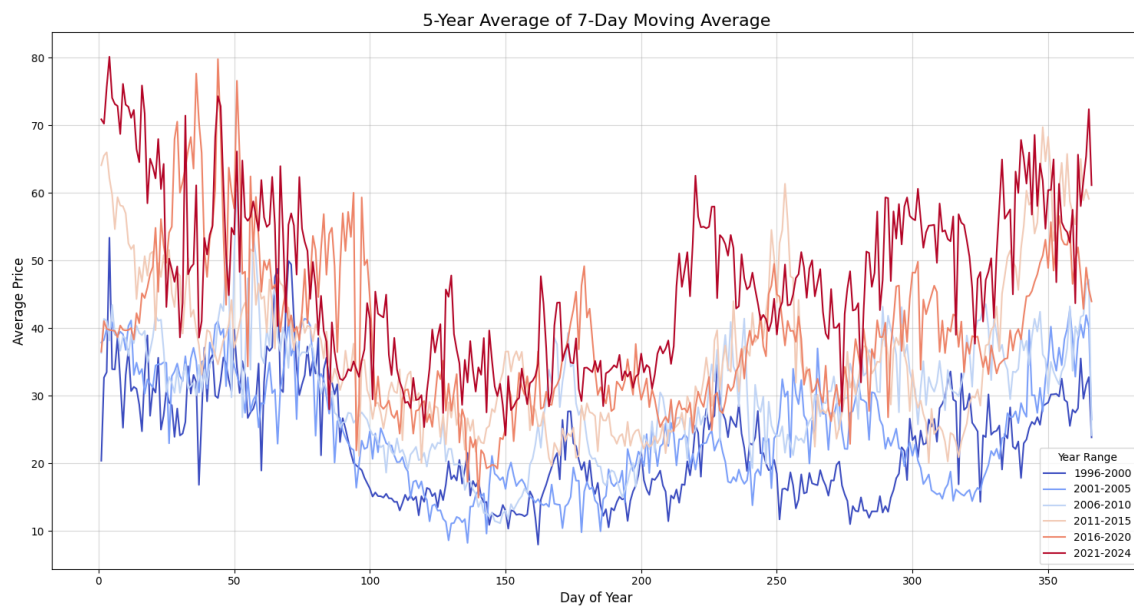
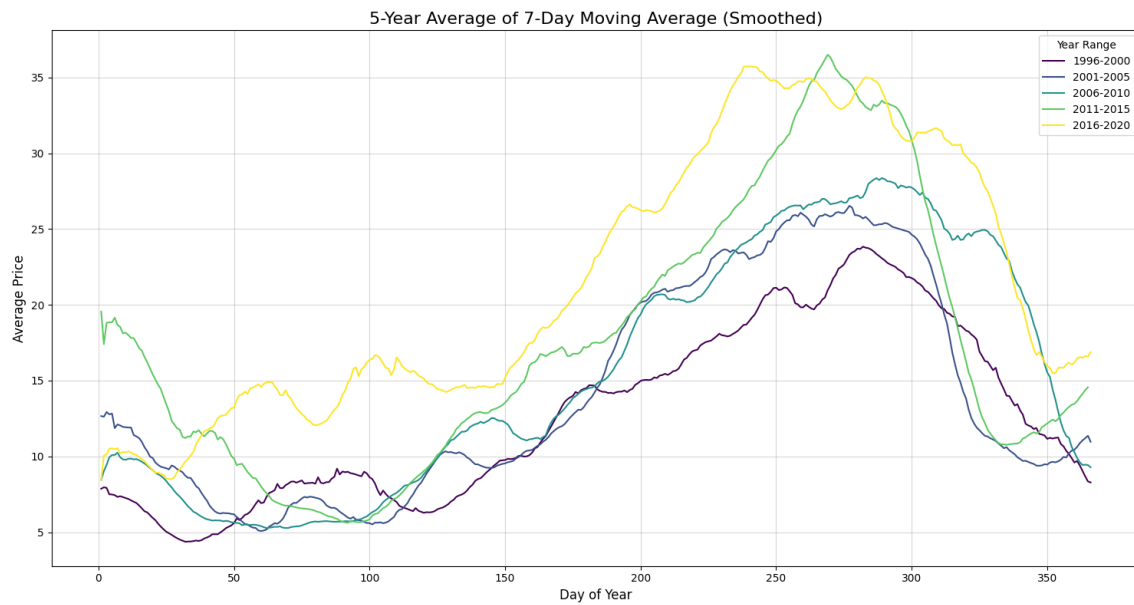
(我們從上圖(高麗菜價格)可以看到在豪雨時雖然有價格上漲，但無明顯變化)



(而從上圖(空心菜價格)可以看到在豪雨時價格上漲相當劇烈)

3、通膨

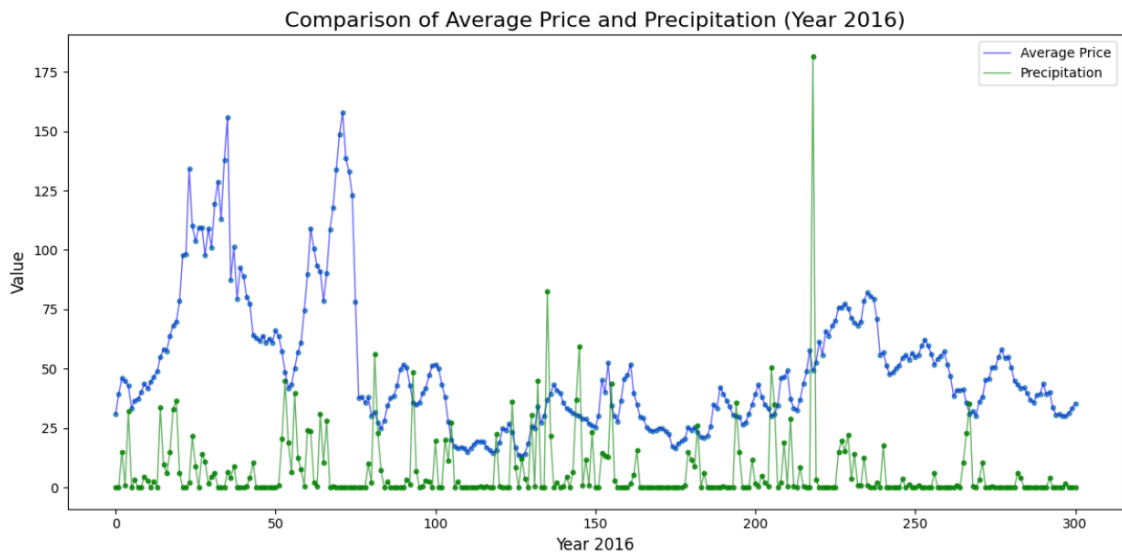
通膨造成的影響也相當明顯，不論任何種類、在 20 年來漲幅落在數成到一倍不等。



蔬菜種類選擇：空心菜

綜合以上考量，我們最後選擇將焦點放在空心菜價格的預測上，原因有以下：

1. 遇到風雨時，空心菜價格反應明顯
2. 空心菜做為台灣的日常青菜，一年四季市場也較為穩定
3. 空心菜的品種單一，不會有不同品種於不同季節互補的因素



(上圖為空心菜價格與降雨量的對比折線圖（2016 年）)

資料前處理

因此在這次的訓練前預處理的部分，我們選擇將「價格」整理成「與預期價格的相差」。

預期價格的算法

我們首先考量每年物價基準不同的因素，將歷年的價格用線性的方式調到相同的基準（讓他們每年的平均價格相同），之後將同月同日的資料取平均值(程式碼也附在下方)。

$$\text{期望價格} = \text{mean}(\text{adjust inflation}(\text{歷史價格}))$$

```
exceptet_price = df_ma.copy()
exceptet_price['weight'] = exceptet_price['year'].map(every_year_average_price)

def calculate_custom_weighted_average(group):
    # 計算基準值：價格除以各自權重後取均值
    baseline_mean = (group['avg_price_ma7'] / group['weight']).mean()
    # 根據每年權重調整值
    group['special_avg'] = baseline_mean * group['weight']
    return group[['year', 'day_of_year', 'special_avg']]

# 按 day_of_year 分組並計算
special_average_df = (
    exceptet_price.groupby('day_of_year')
    .apply(calculate_custom_weighted_average)
    .reset_index(drop=True)
)
```

但是因為歷史價格有部分的缺失且波動值極大，因此先將其作一次 7 天的平滑。

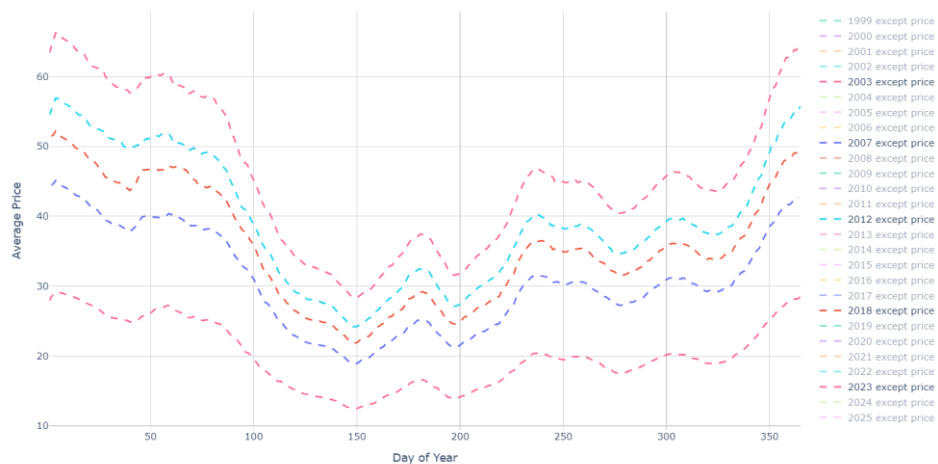
然而後續平均後的價格也仍因為波動依舊劇烈，所以在最後又疊加了一層 14 天的平滑，公式最後變成：(程式碼也附在下方)

$$\text{期望價格} = 14\text{daysmooth}\left(\text{mean}\left(7\text{daysmooth}\left(\text{adjust inflation}(\text{歷史價格})\right)\right)\right)$$

```
ma_dfs = []
for year, group in df_price.groupby('year'):
    group = calculate_moving_average(group, 'avg_price', 7)
    # group = calculate_moving_average(group, 'avg_price', 10)
    # group = calculate_moving_average(group, 'avg_price', 20)
    ma_dfs.append(group)

df_ma = pd.concat(ma_dfs)

special_average_df['smooth_14day'] = (
    special_average_df.groupby('year')['special_avg']
    .transform(lambda x: x.rolling(window=14, min_periods=1).mean())
)
```



(上圖為推算出的各年預期價格的曲線，我們可以看到在考量通膨後，預期價格也有所不同)

轉換價格的原因

之所以要大費周章的將價格換成「與預期價格的相差」是因為我們希望這樣可以減少輸入日期的干擾，讓模型只有「雨量」與「先前價格資訊」作為輸入，更可讓模型可以直接判斷現在為偏高或偏低的價格。

```

import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# 合併數據
main_feature, main_target = [], []
for i in yearlist[:-2]:
    data = pd.merge(price_dict[i], rain_dict[i], left_on='day_of_year', right_on='dayofyear', how='outer')
    data = data.sort_values('dayofyear').reset_index(drop=True)

    # 填補缺失值
    data['price_diff'] = data['price_diff'].fillna(method='ffill').fillna(method='bfill')
    data['Precipitation'] = data['Precipitation'].fillna(0)

    # 標準化數據
    # scaler = MinMaxScaler()
    # data[['price_diff', 'Precipitation']] = scaler.fit_transform(data[['price_diff', 'Precipitation']])

    # 創建序列數據
    sequence_length = 7
    features, targets = [], []
    for i in range(len(data) - sequence_length - 7):
        r_sequence = data['Precipitation'].iloc[i:i + sequence_length].values
        p_sequence = data['price_diff'].iloc[i:i + sequence_length].values
        target = data['price_diff'].iloc[i + sequence_length:i + sequence_length + 7].values
        features.append(np.concatenate([r_sequence, p_sequence]))
        targets.append(target)

    main_feature.extend(features)
    main_target.extend(targets)

features = np.array(main_feature)
targets = np.array(main_target)

train_size = int(len(features) * 0.8)
X_train, X_test = features[:train_size], features[train_size:]
y_train, y_test = targets[:train_size], targets[train_size:]

# 調整數據形狀
X_train = X_train.reshape((X_train.shape[0], sequence_length, -1))
X_test = X_test.reshape((X_test.shape[0], sequence_length, -1))

# 建立模型
model = Sequential([
    LSTM(64, return_sequences=False, input_shape=(sequence_length, X_train.shape[2])),
    Dense(32, activation='relu'),
    Dense(7) # 預測7天
])

model.compile(optimizer='adam', loss='mse')
model.summary()

# 訓練模型
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))

# 評估模型
loss = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}")

```

（上圖為前處理、訓練模型程式碼）

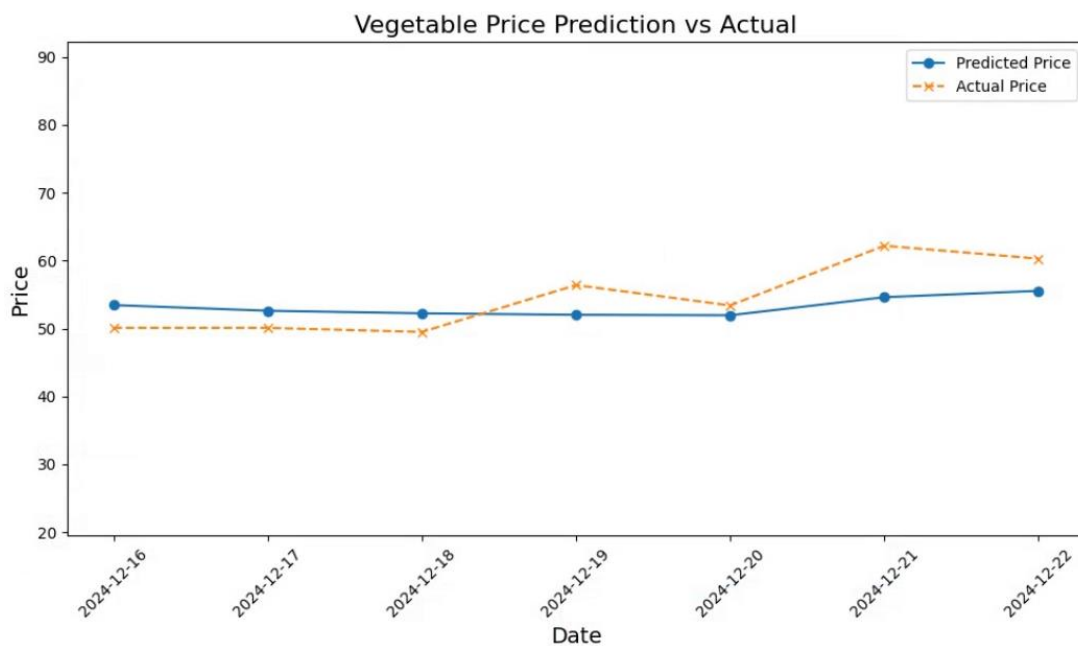
四、預測結果分析

```

Epoch 37/50
62/62 0s 4ms/step - loss: 68.0984 - val_loss: 134.3673
Epoch 38/50
62/62 0s 3ms/step - loss: 66.4690 - val_loss: 135.6890
Epoch 39/50
62/62 0s 3ms/step - loss: 62.8229 - val_loss: 142.1145
Epoch 40/50
62/62 0s 4ms/step - loss: 62.0584 - val_loss: 135.9332
Epoch 41/50
62/62 0s 3ms/step - loss: 59.9546 - val_loss: 137.8671
Epoch 42/50
62/62 0s 3ms/step - loss: 62.9439 - val_loss: 142.6102
Epoch 43/50
62/62 0s 3ms/step - loss: 58.4223 - val_loss: 136.5250
Epoch 44/50
62/62 0s 3ms/step - loss: 55.6084 - val_loss: 146.7313
Epoch 45/50
62/62 0s 4ms/step - loss: 55.0417 - val_loss: 142.5346
Epoch 46/50
62/62 0s 3ms/step - loss: 52.9411 - val_loss: 141.4600
Epoch 47/50
62/62 0s 3ms/step - loss: 54.2953 - val_loss: 138.1782
Epoch 48/50
62/62 0s 3ms/step - loss: 53.8569 - val_loss: 146.4849
Epoch 49/50
62/62 0s 4ms/step - loss: 53.7806 - val_loss: 145.6363
Epoch 50/50
62/62 0s 4ms/step - loss: 53.8230 - val_loss: 145.4563
16/16 0s 2ms/step - loss: 152.1919
Test Loss (mse): 145.45628356933594

```

(上圖為訓練模型時的結果)



最後模型在 Test Dataset 的 RMSE 是 12.06，對我們當初設立的目標而言，這是一個不錯的準度。測試不同預測日期時，能發現模型大多能預測到「價格大概的位置」，使得整體誤差不會太大。不過，目前的模型對於局部漲價與降價的量值仍不能做到很好的預測，因此，即便我們即便我們預測到有漲價的趨勢(如上圖 12/18~12/22)，但預測值的漲價幅度卻小於實際值，這是未來可以再去努力改善的地方。

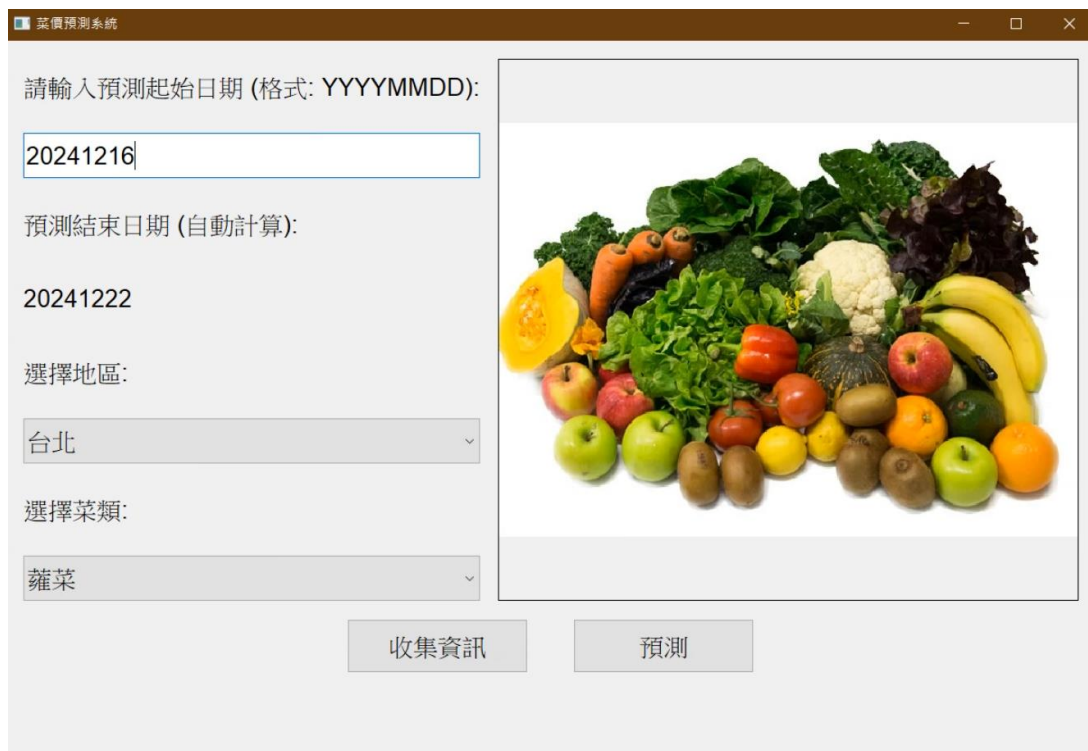
五、專案 Demo 及結果展示

專案介紹 Video 連結：<https://youtu.be/gojqVlS-tQg>

GUI 操作 Video 連結：<https://youtu.be/YydvjFqTknU>

GUI 操作：

- (1) 輸入欲預測之時間區段(7 天)
- (2) 選擇交易市場 (本專案目前只完成台北一交易市場)
- (3) 選擇欲預測之菜種 (本專案目前只完成薙菜之預測模型)
- (4) 點擊「收集今日資訊」，呼叫 Selenium 爬蟲程式收集過去 7 天菜價和過去 7 天全台灣降水量平均值，並做好資料前處理
- (5) 點擊「預測」，透過模型預測菜價，並視覺化預測結果



菜價預測系統

請輸入預測起始日期 (格式: YYYYMMDD):

20241216

預測結束日期 (自動計算):

20241222

選擇地區:

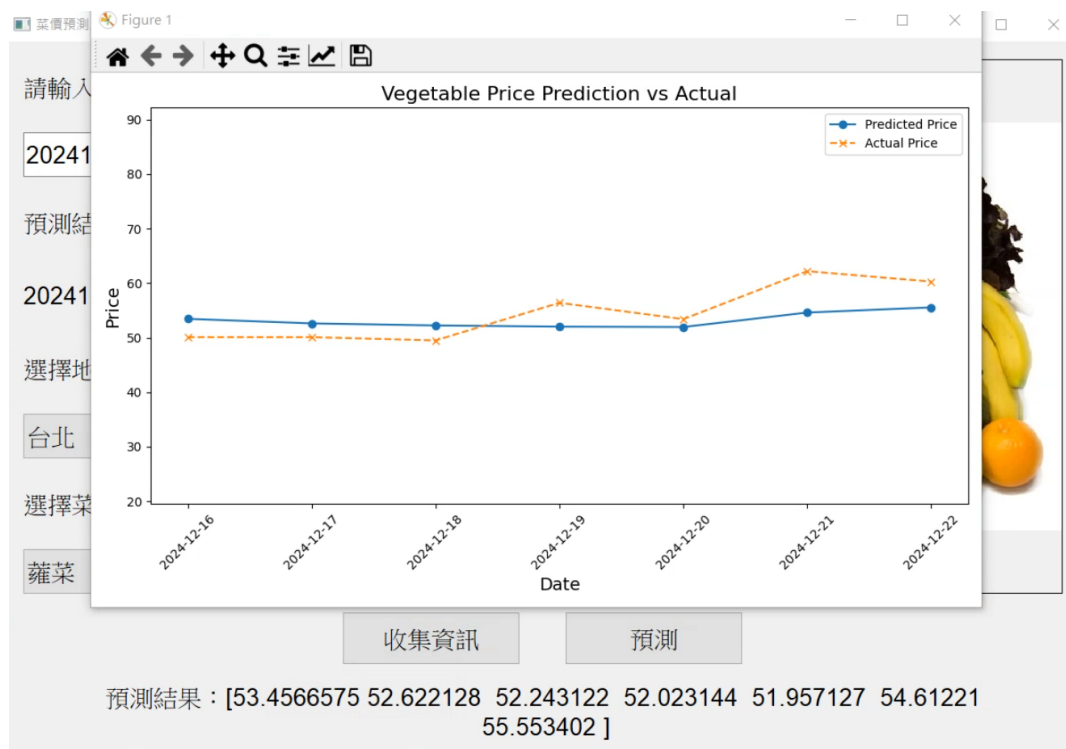
台北

選擇菜類:

薙菜

收集資訊 預測

(預測前之 GUI 展示圖)



(預測後之 GUI 展示圖)

六、專案遇到之困難和平台困境

- 影響菜價之因素眾多：
不只有天氣，政府相關政策、市場供需、通貨膨脹等因素常常都會影響模型的預測準確度，但採納過多不必要的特徵，可能也會產生雜訊。
- 交易市場和菜類眾多：
預測菜價的應用範圍若不能覆蓋整個台灣市場，若不能預測多種菜類，那這個平台的實用性就會降低，畢竟大家不會只吃同一種蔬菜。不過如果考量多個市場、多種蔬菜，一來會大幅增加模型訓練的複雜程度，一來可能也需要訓練多個模型，這樣在訓練和預測過程上皆會耗費大量時間。
- 菜價、天氣資料缺失：
我們在爬取資料的過程中，發現菜價平台並不會記錄每一天的交易資訊，有部分日期是沒有資料的；在降水量的資料收集中，也常發現因為儀器故障等因素，會有一些日期沒有降水量資料。由於我們的菜價預測模型會使用到過去一周的菜價擊降水量作為輸入，因此資料缺失可能會影響預測結果準確性。
- 預測時間不遠：
目前我們的平台僅能提供下一週的菜價預測，對於更長時間範圍的未來菜價尚無法準確預測，因此在預測時間的範圍上存在一定的限制。

七、困難之解決及未來展望

- 建立可拓展的平台：
雖然目前平台只能預測在特定市場的單一菜種，但我們設計平台時賦予它可拓展性。若是將來針對不同市場、不同菜種訓練了其他模型，只需放入我們的模型庫，便可被使用者透過平台調用。
- 針對不同模型進行深度優化：
當前的模型架構雖然能夠在目前的訓練方式上取得較好的表現，但並非所有的模型都經過深度優化。我們可以嘗試針對其他模型進行更全面的調整，包括特徵的重新選擇、超參數的細化調整等(儘管這樣的調整方式需要耗費大量的時間與資源)。這樣一來，在充分理解數據分布與模型表現的基礎上，應該能進一步提升整體準確率與穩定性。
或許在不同菜種的預測上，不同模型的表現也不一，因此有待研究。
- 如同結果分析所說，模型對於漲價和降價"幅度"的預測訓練仍待加強。

八、主要檔案上傳說明

data/：訓練所需要用到的 datasets

output/：存放訓練好的預測模型、Test dataset 預測輸出

main.py：GUI，用於呼叫爬蟲程式及使用模型

crawler.py：爬取訓練用的 datasets 及上一周菜價&全台降雨

lstm_薤菜.ipynb：訓練模型

九、組員的分工/貢獻

組員名稱	貢獻
電機系116曾柏誠	1.詳細分析資料，題目 2. 訓練&測試模型 3.報告製作
資訊系114何寬羿	1.使用網路爬蟲爬取資料 2.訓練&測試模型 3. 報告製作
資訊系114林業誠	1. 建構UI圖像化使用者介面 2.訓練&測試模型 3. 影片製作