目錄

一、專案介紹

1. 背景與問題

隨著太陽能成為全國主要的再生能源供應來源，其發電的不穩定性對電網的穩定運作構成挑戰。本競賽以花東地區的微氣候數據為基礎，旨在透過預測太陽能板的發電瓦數，提升智慧電網的預測精準度，並促進參賽者對再生能源的理解與應用。

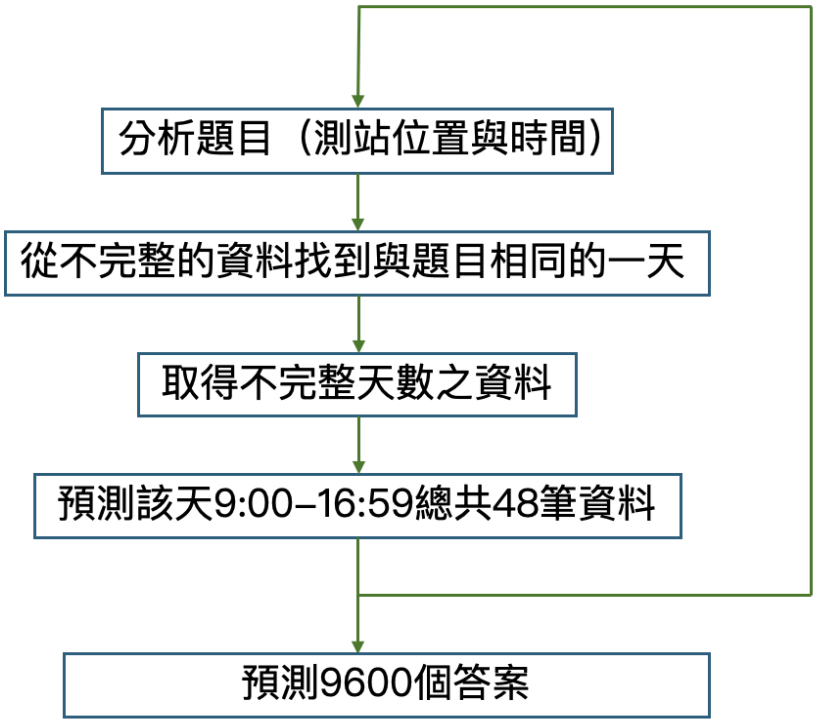
2. 資料集描述

資料集包含 17 個太陽能監測地點的微氣候數據，特徵變數包含 DateTime,WindSpeed(m/s),Pressure(hpa),Temperature(°C),Humidity(%),Sunlight(Lux),Power(mW)，標籤為發電量Power(mW)，每份資料以 CSV 格式呈現。數據具有時間序列特性，但可能因感測器異常或其他因素導致部分缺失，也有某些天 9:00 之後的數據被移除以作為預測題目。

3. 競賽目標

需根據歷史微氣候數據預測指定裝置在特定日期與時段內的每 10 分鐘平均發電瓦數。評分以參賽者預測值與正確答案的誤差絕對值加總為依據，誤差越小排名越高。

4. 分析步驟



二、資料前處理

1. 天氣資料切割

使用主辦方所提供之「天氣小助手」切割所需之資料。

a. Complete Average Data (AVGdata)

切割出9:00至16:59之完整天氣資料，並將資料以10分鐘一次取平均值到小數點後第二位。

b. Incomplete Average Data (IcompleteAVGdata)

將不完整的天氣資料（非整天之資料）以10分鐘一次取平均值到小數點後第二位。

2. 手動加入標題

天氣小助手輸出的資料會缺少標題，需用記事本手動加入標題。

3. 資料讀取

a. 套件使用

```
import numpy as np
import pandas as pd
import os
```

b. 讀取檔案

使用 `pd.read_csv()` 讀取兩份天氣資料檔案，分別為 `AvgDATA_03.csv` 和 `AvgDATA_08.csv`。這裡只使用了兩組測站的資料，若使用全部測站的資料合併應該可以得到更好的結果。

c. 合併資料

將`AvgDATA_03.csv`、`AvgDATA_08.csv`兩個 DataFrame合併。

d. 選擇目標欄位

從合併後的資料中選擇 `Power(mW)` 欄位，只留下發電量當特徵。

e. 建立訓練資料

`X_train`:每筆輸入資料包含i-12筆資料
`Y_train`:對應輸入資料後的第 i 筆資料

三、模型建構與訓練

1. LSTM簡介

LSTM(長短期記憶網路)是一種特殊類型的循環神經網路，用來解決傳統 RNN 在處理長序列資料時的梯度消失問題。LSTM 相較於傳統的 RNN，具有三個關鍵的內部結構：遺忘門、輸入門和輸出門。這些門控結構使 LSTM 能夠在處理長序列時有效地保留和忘記訊息，進而克服了 RNN 的梯度消失問題。

2. 建立 LSTM 模型

a. 套件使用

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout
from tensorflow.keras.models import load_model
```

b. 模型訓練

I. 初始化模型

`regressor = Sequential()`

- `Sequential()`是 Keras 中的一種模型，表示模型由一層一層地堆疊起來，適用於簡單的順序結構。

II. 添加 LSTM 層

`regressor.add(LSTM(units=128, return_sequences=True, input_shape=(X_train.shape[1], 1)))`

- `LSTM`：這是 LSTM 層的定義，用來處理時間序列資料。這一層有 128 個單元(units)。
- `return_sequences=True`：表示這一層會返回每個時間步的輸出，不是最後一個時間步的輸出。
- `input_shape=(X_train.shape[1], 1)`：這告訴模型輸入數據的形狀。`X_train.shape[1]` 是每個樣本的時間步數，`1` 是每個時間步的特徵數(這裡只有瓦數一個特徵)。

`regressor.add(LSTM(units=64))`

- 這層 LSTM 的單元數設定為 64，並且跟前一層不同的是 `return_sequences=False`，表示這一層只返回最終的隱藏狀態，並不返回每一個時間步的輸出。

III. 加入Dropout層

`regressor.add(Dropout(0.2))`

- `Dropout(0.2)`:在訓練過程中，會隨機丟棄 20%的神經元，這有助於提高模型的泛化能力。

IV. 加入輸出層

```
regressor.add(Dense(units=1))
```

- `Dense(units=1)`:輸出只有 1 個單元，表示模型的預測結果。

V. 編譯模型

```
regressor.compile(optimizer='adam', loss='mean_squared_error')
```

- `loss='mean_squared_error'`:選擇MSE作為損失函數，這對回歸問題（預測數值）來說是常見的選擇。

VI. 訓練模型

```
regressor.fit(X_train, y_train, epochs=100, batch_size=128)
```

- `epochs=100`:訓練 100 次（每次迭代所有訓練數據）。
- `batch_size=128`:每次訓練時，使用 128 筆資料進行更新。

VII. 儲存模型

```
from datetime import datetime
NowDateTime =
datetime.now().strftime("%Y-%m-%dT%H_%M_%SZ")
regressor.save('WheatherLSTM_'+NowDateTime+'.h5')
```

- `datetime.now().strftime()`:生成當前的時間用來命名模型文件。

3. 進行預測

a. 讀取需要預測的天數（讀取題目）

```
SourceData = pd.read_csv('upload(no answer).csv', encoding='utf-8')
target = ['序號']
EXquestion = SourceData[target].values
```

- 載入主辦單位給的 `upload(no answer).csv`檔案。
  `target = ['序號']` 表示從資料中選取 "序號" 欄位作為需要預測的對象。

b. 判斷需要預測的天數、測站位置（判斷題目）

I. 初始化參數

```
inputs = []
PredictOutput = []
count = 0
```

- `PredictOutput` 用來存放每次的預測結果。
- `count` 控制循環，每次處理 48 個資料（一天的預測資料）。

II. 使用迴圈判斷測站位置

```
while(count < len(EXquestion)):
    print('count : ', count)
    LocationCode = int(EXquestion[count])
    strLocationCode = str(LocationCode)[-2:]
    if LocationCode < 10:
        strLocationCode = '0' + str(LocationCode)
```

- `LocationCode` 變數從 `EXquestion[count]` 取得，並根據 `LocationCode` 生成對應的資料檔案名。

III. 讀取對應位置的資料

- 根據每個 `LocationCode` 讀取對應的訓練資料檔案，`ReferTitle` 和 `ReferData` 分別儲存資料中的序號和發電量數值。

IV. 從資料中選擇相同日期的資料

c. 用迴圈來進行預測

```
for i in range(ForecastNum):
    if i > 0:
        inputs = np.append(inputs, PredictOutput[i-1])

    X_test = []
    X_test.append(inputs[0 + i : LookBackNum + i])

    # Reshaping
    NewTest = np.array(X_test)
    NewTest = np.reshape(NewTest, (NewTest.shape[0],
NewTest.shape[1], 1))
    predicted = regressor.predict(NewTest)
    PredictOutput.append(round(predicted[0, 0], 2))
count += 48
```

- 這個迴圈用來進行逐步的預測。每次預測時會使用前 12 筆(`LookBackNum`)資料，並且每次預測的結果會被加到 `inputs` 中，用來作為下一次預測的參考資料。
- 每次預測會產生一個新的值，並將該預測結果加入到 `PredictOutput` 中。
- 每次預測 48 個資料(一天的預測)，完成後 `count` 會增加 48，處理下一批資料。

d. 儲存結果

```
df = pd.DataFrame(PredictOutput, columns=['答案'])
df.to_csv('output.csv', index=False)
print('Output CSV File Saved')
```

- 使用 `pd.DataFrame` 將預測結果轉換成 DataFrame，並使用 `to_csv` 將預測結果保存到 `output.csv` 中。

4. 手動調整資料以符合比賽上傳格式

輸出的`output.csv`只有答案，沒有題目。需要手動把答案複製貼上回`upload(no answer).csv`才能上傳。

四、評分標準

將每一筆預測資料與正確資料相比並計算誤差(取絕對值)，並將所有誤差加總在一起，以加總誤差值為排名依據，加總誤差越少排名越高。

$$\text{總分數} = \sum_{i=1}^{n} |A_i - S_i| \text{。}$$

五、結果與討論

1. 預測結果

Private score：2071845.76
排名：248

2. 討論

Private score：2071845.76顯示此模型未能充分捕捉數據中的某些關鍵特徵或模式。以下是幾點我們認為可以改善的部分。

a. 使用更多不同測站的AVGdata來建構模型：

我們的模型只使用了3、8兩個測站的資料來訓練模型，而不同測站之間的發電量與時間的關係可能會大相徑庭，因此使用更多或是所有測站的資料來尋練模型會大大提升模型的預測能力。

b. 加入其他特徵：

我們的模型只使用了發電量作為特徵，忽略了其他環境因素（例如亮度、風速、溫度）。這些變數可能影響太陽能板發電的效率，加入這些個徵可能可以為模型提供更強的預測基礎。

六、結論

本次研究中，我們使用 LSTM 模型對太陽能發電量進行預測，Private score 為 2071845.76，排名第 248。雖然結果具備一定的參考價值，但也揭示了模型在數據利用與特徵選擇上的不足。未來的改進方向包括：

1. 使用更多測站的數據，讓模型得到更多不同測站的資訊。
2. 納入其他環境特徵，讓模型捕捉環境特徵對發電效率的影響。

透過這些改進，我們期待模型能提升預測能力與競賽表現。

七、附錄

a. 完整程式碼(github)
   https://github.com/mellamochiao/datascienceCP2

b. 參考連結

   i. LSTM https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

   ii. 競賽網站 https://tbrain.trendmicro.com.tw/Competitions/Details/36

   iii. 氣候小助手https://www.youtube.com/watch?v=8Wvqr-vIh3A&t=724s