

# Covid-19 Pandemic Prediction

# Dataset introduction

# Covid-19 Pandemic Prediction

- 2019 年 12 月 31 日，世衛組織收到關於中國湖北省武漢市多起肺炎病例的警報。
- 該病毒與任何其他已知病毒都不相符
  - 這引起了人們的關注，因為當病毒是新病毒時，我們不知道它會對人們造成什麼影響。
- 透過分析過去這段時間的疫情資料，可以為我們提供更多對於新冠病毒的瞭解，掌握目前趨勢，甚至進行本國疫情之預測
- 資料來自：  
<https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset>

# Dataset

- 資料來源：約翰霍普金斯大學（Johns Hopkins） Github repository (as csv files)
- 306429 筆資料, 8 個 columns:
  - Sno - Serial number
  - ObservationDate - Date of the observation in MM/DD/YYYY
  - Province/State - Province or state of the observation (Could be empty when missing)
  - Country/Region - Country of observation
  - Last Update - Time in UTC at which the row is updated for the given province or country.
  - Confirmed - **Cumulative number** of confirmed cases till that date
  - Deaths - **Cumulative number** of of deaths till that date
  - Recovered - **Cumulative number** of recovered cases till that date

# Data Preprocessing & Visualization

# 資料預處理

## 1. Standardizing the date-time column

➔ column "ObservationDate" to date time

```
[ ] df['ObservationDate'] = pd.to_datetime(df['ObservationDate'])
```

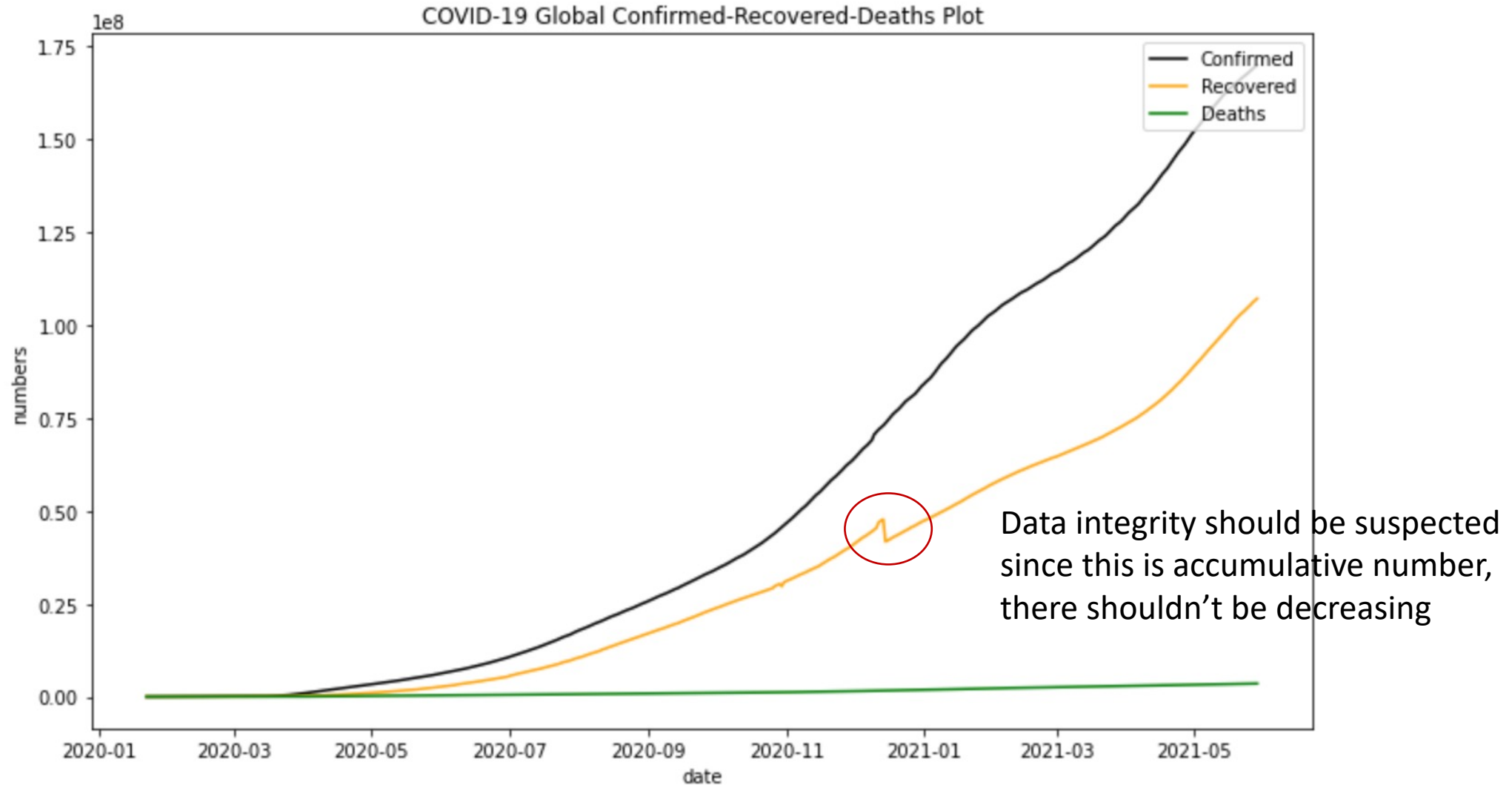
## 2. Group by Observation Date & sum up all the case count through the countries

```
preprocessed_df = df.groupby(df['ObservationDate'].dt.date)['Confirmed', 'Deaths', 'Recovered'].agg(['sum']) #算總和
```

➔ after grouping: 306429 rows × 8 columns ➔ 494 rows × 3 columns

## 3. Data Visualization

# COVID-19 Pandemic Global Trend Plot



# Model Training & Prediction



# Part 2: COVID-19 台灣疫情預測

- We choose Taiwan as the country to do the prediction

```
[ ] x = df[df['Country/Region']=='Taiwan'] #選要做分析的國家
```

- Do the preprocessing as while dealing with the global data:

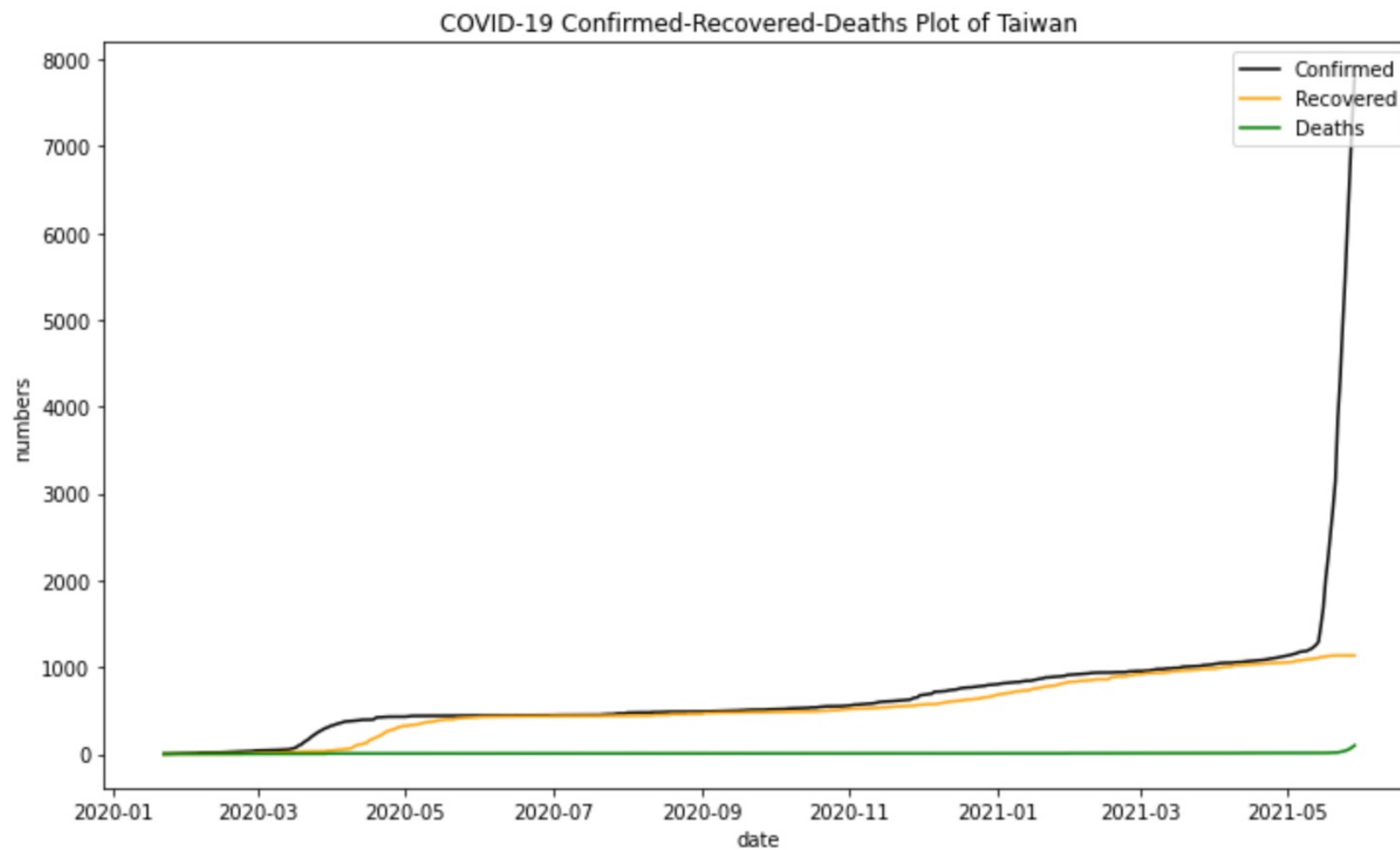
- Standardizing the date-time
- Group by & sum up
- Data visualization

```
[ ] preprocessed_x
```

	Confirmed	Deaths	Recovered
	sum	sum	sum
ObservationDate			
2020-01-22	1.0	0.0	0.0
2020-01-23	1.0	0.0	0.0
2020-01-24	3.0	0.0	0.0
2020-01-25	3.0	0.0	0.0
2020-01-26	4.0	0.0	0.0
...	...	...	...
2021-05-25	5456.0	35.0	1133.0
2021-05-26	6091.0	46.0	1133.0
2021-05-27	6761.0	59.0	1133.0
2021-05-28	7315.0	78.0	1133.0
2021-05-29	7806.0	99.0	1133.0

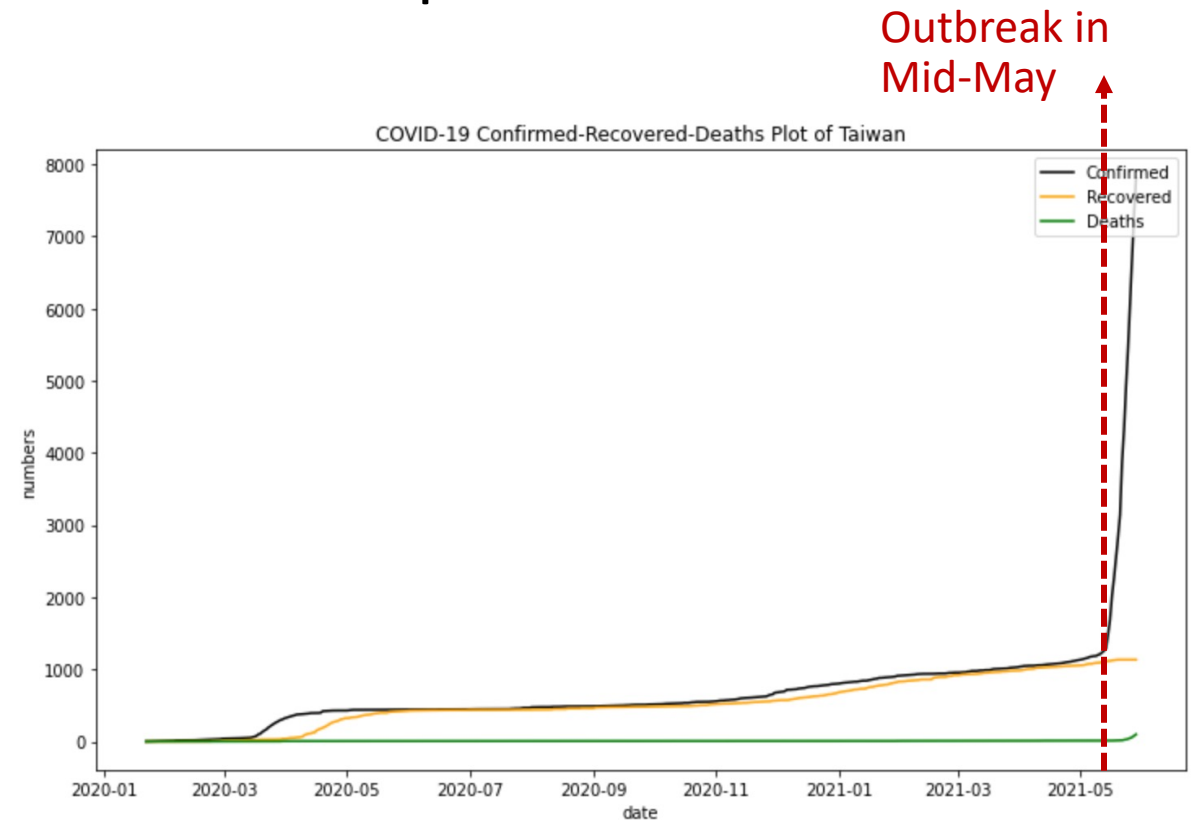
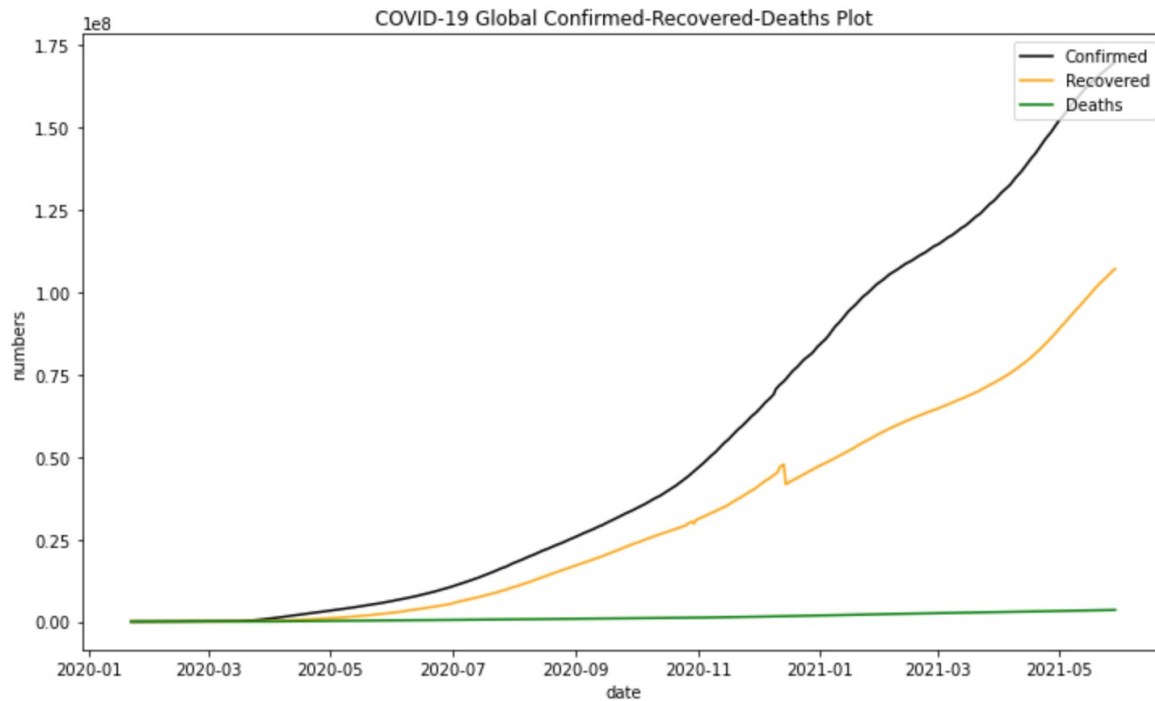
494 rows x 3 columns

# COVID-19 台灣趨勢累計圖



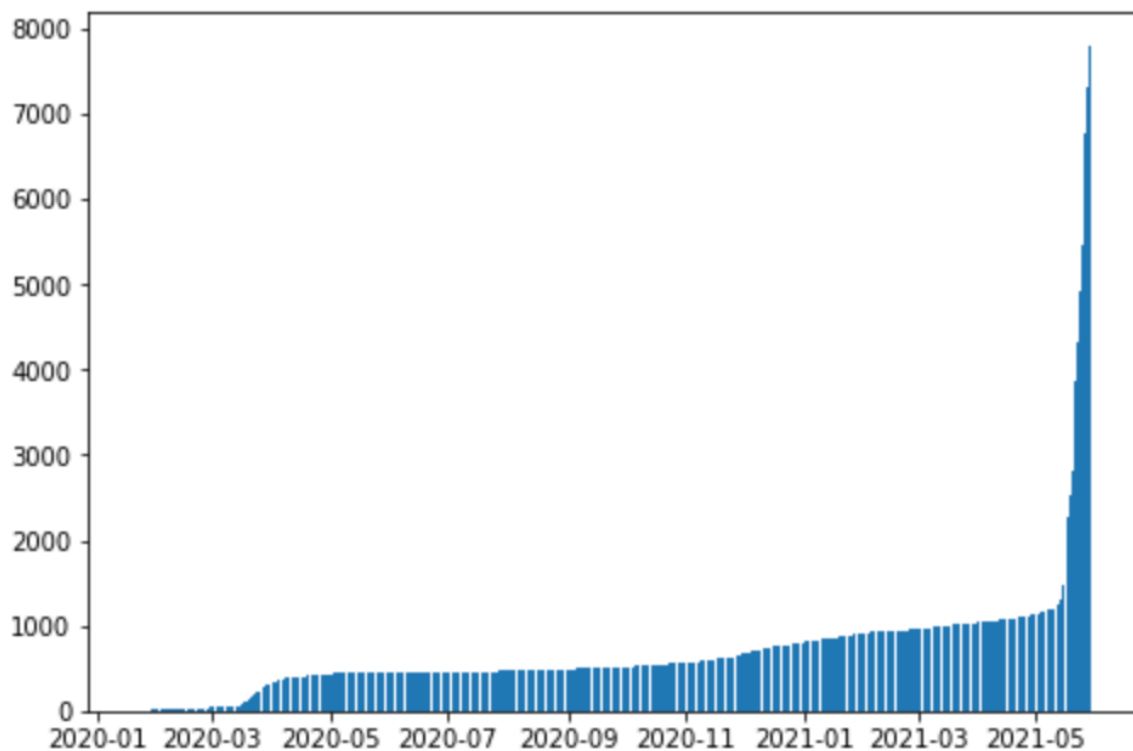
# COVID-19 全球趨勢與台灣趨勢之比較

\*\*\* note: the order of magnitude is different in 2 plots

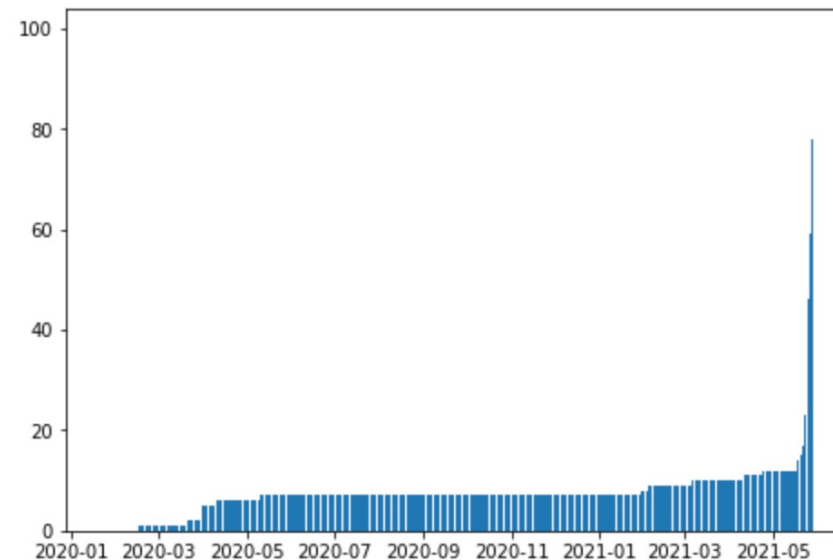


# 台灣案件數統計圖

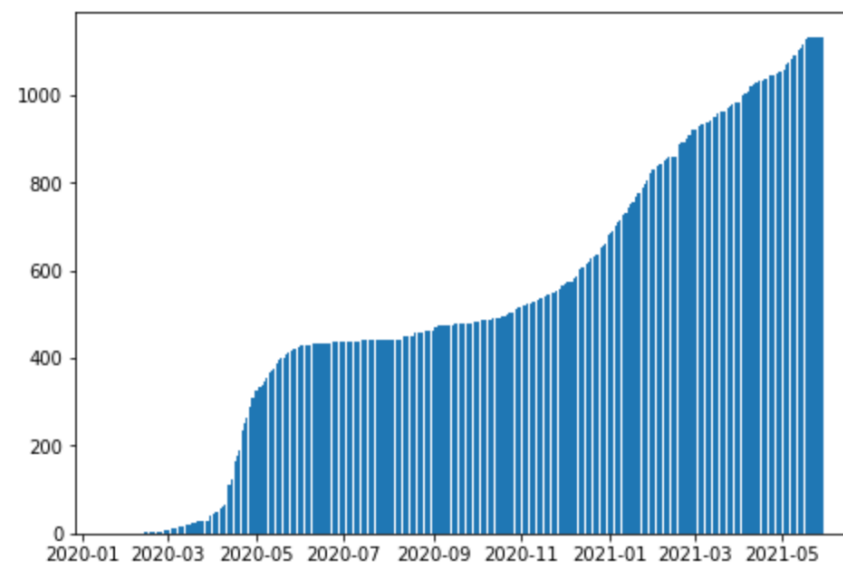
- 台灣確診數累計圖



- 台灣死亡數累計圖



- 台灣復原數累計圖

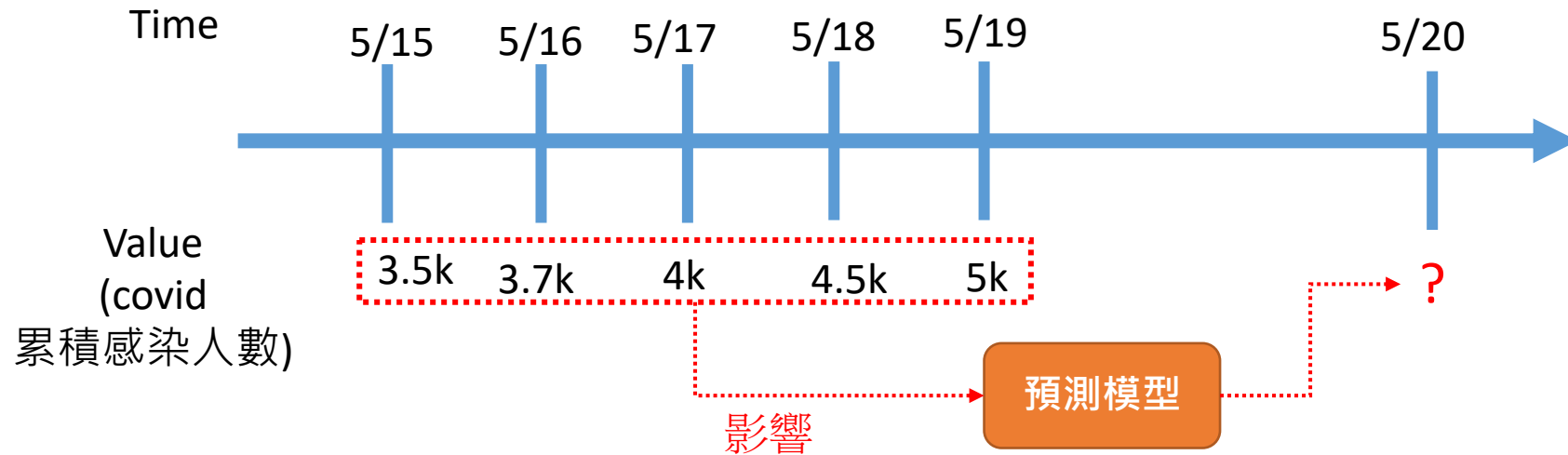


# LSTM

以預測covid每日確診累積人數為例

# 使用LSTM的動機

- 欲處理的資料**具有時序上**的關係
  - 例如, 當我們打算預測5/20的累積感染人數時, 5/20之前的感染人數(5/19,18...) 其實應該是會對5/20的預測有所影響的。



# 對比: 使用CNN or MLP 處理非時序性資料

- **Task1 年收入預測**

- 50000筆資料, 每筆資料代表了一個人的資訊
- 每筆訓練資料內容為: 性別、出生地、人種、職業、年齡...
- 訓練label: 0 = 年收入小於\$50k, label:1 =年收入大於\$50k
- 任務目標是:
  - 給定一筆個人資訊 (性別、出生地、人種、職業、年齡), 預測其年收入大於or小於\$50k

- **Model: MLP**

- 對於這種任務而言, 資料與資料之間就不具時序性。模型預測過程中, 上一筆資料的預測結果並不會影響到這一筆資料的預測結果
  - 上一個人是否年收入 > \$50k 並不影響現在這個人的預測結果

# 對比: 使用CNN or MLP 處理非時序性資料

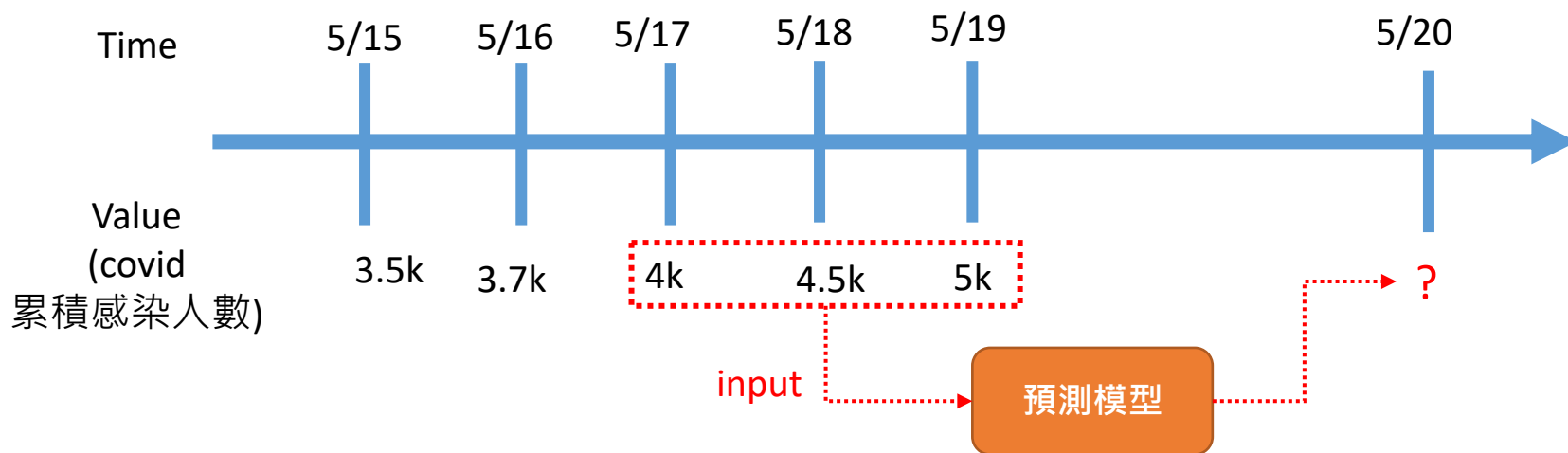
- Task2 影像分類

- 20000筆動物影像資料
  - 訓練label 為影像所屬的動物類別
  - 任務目標是:
    - 給定一張影像，進行所屬動物類別的判斷
  - Model: CNN
- 對於這種任務而言，資料與資料之間就不具時序性。模型預測過程中，上一張影像的判斷結果並不會影響到這一張影像的判斷結果



# Look back

- 前述提到了資料具有時序性，LSTM如何讓前幾筆資料能夠對現在這筆資料的預測產生影響？
- 首先從LSTM 中**look back**的概念開始：
  - Look back: 意即對於預測這筆資料，你打算往前看多少筆資料？
  - Ex: Look back = 3, 代表對於預測5/20 感染人數，我打算往前看3筆資料來幫助我進行這一次的預測。

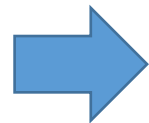


# Look back

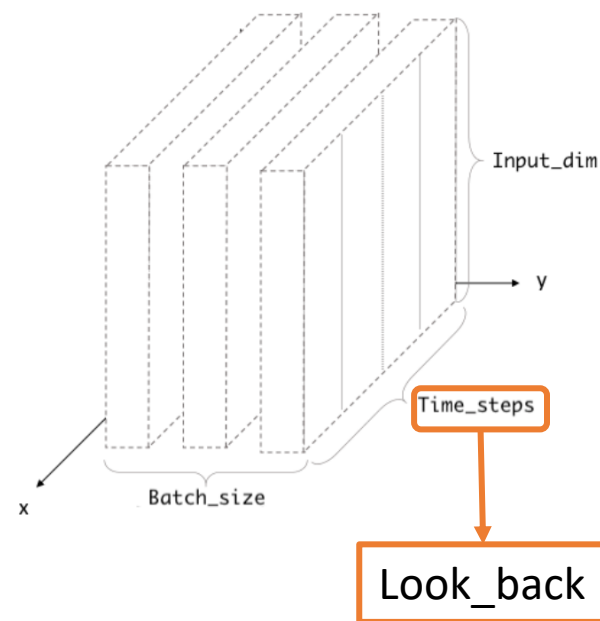
- 承接LSTM中look back (ex, look\_back = 3)的想法，我們可以把資料進行預處理
- 將每三筆 (look\_back size) 資料包起來當作一筆模型的訓練資料。

各日累積確診數

```
array([[ 1.,  1.,  3.,  3.,  4.,  5.,  8.,  8.,  9., 10., 10.,
        10., 10., 11., 11., 16., 16., 17., 18., 18., 18., 18.,
        18., 18., 18., 20., 22., 22., 23., 24., 26., 26., 28.,
        30., 31., 32., 32., 34., 39., 40., 41., 42., 42., 44.,
        45., 45., 45., 45., 47., 48., 49., 50., 53., 59., 67.,
        77., 100., 108., 135., 153., 169., 195., 215., 235., 252., 267.,
        283., 298., 306., 322., 329., 339., 348., 355., 363., 373., 376.,
        379., 380., 382., 385., 388., 393., 393., 395., 395., 395., 398.,
        420., 422., 425., 426., 427., 428., 429., 429., 429., 429., 429.,
        429., 429., 432., 436., 438., 438., 439., 440., 440., 440., 440.,
        440., 440., 440., 440., 440., 440., 440., 440., 440., 440.,
        441., 441., 441., 441., 441., 441., 441., 442., 442., 442., 443.,
        443., 443., 443., 443., 443., 443., 443., 443., 443., 443.,
        443., 443., 445., 445., 445., 446., 446., 446., 446., 446., 446.,
        446., 447., 447., 447., 447., 447., 447., 448., 449., 449.,
        449., 449., 449., 449., 449., 451., 451., 451., 451., 451.,
        451., 451., 451., 451., 451., 455., 455., 458., 458., 458.,
        462., 467., 467., 467., 467., 474., 475., 474., 476., 476., 477.,
        477., 479., 477., 477., 480., 481., 481., 481., 482., 484., 485.,
        486., 486., 486., 487., 487., 487., 487., 487., 487., 487., 487.,
        488., 488., 488., 488., 489., 489., 490., 492., 493., 494., 495.,
        495., 496., 498., 498., 498., 499., 499., 500., 503., 503., 506.,
        507., 509., 509., 509., 509., 510., 510., 510., 513., 513., 514.,
        515., 517., 517., 517., 518., 521., 523., 524., 527., 527., 527.,
        529., 530., 530., 531., 535., 535., 535., 540., 543., 544., 548.,
        548., 550., 550., 550., 550., 550., 553., 554., 555., 558., 563.,
        567., 568., 569., 573., 573., 577., 578., 580., 584., 589., 597.,
        600., 602., 603.]])
```

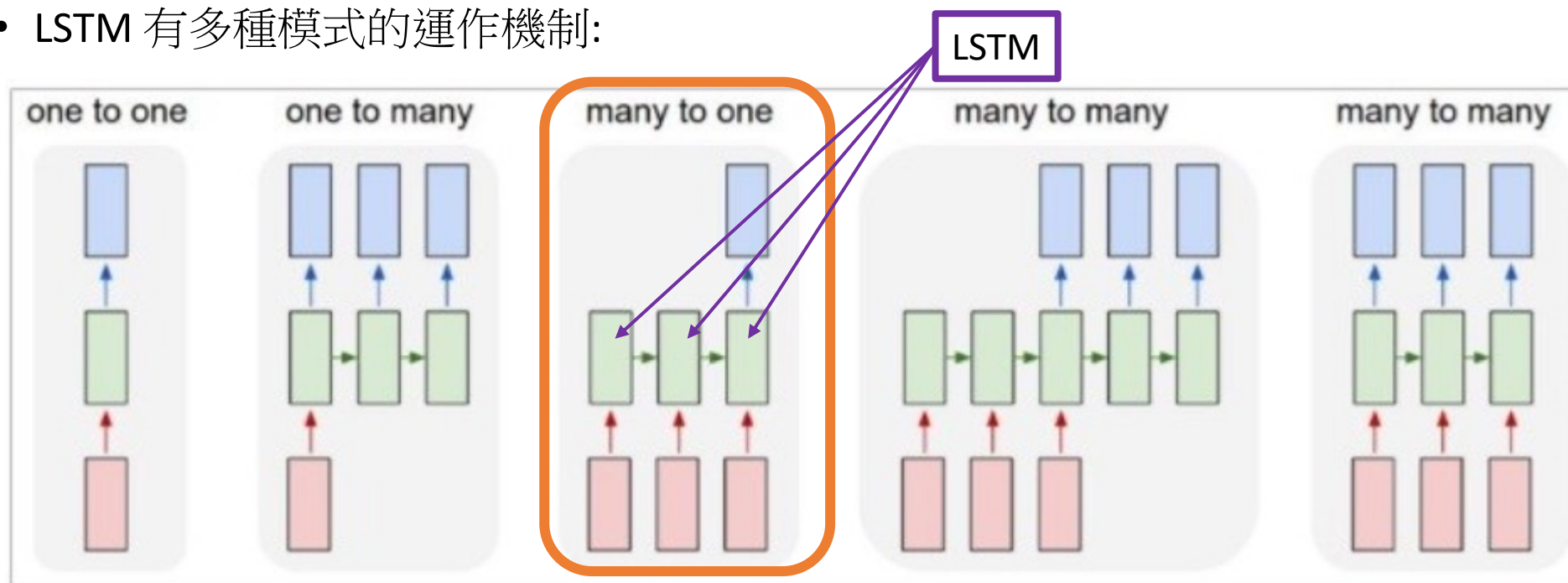


```
array([[ 1.,  1.,  3.],
       [ 1.,  3.,  3.],
       [ 3.,  3.,  4.],
       [ 3.,  4.,  5.],
       [ 4.,  5.,  8.],
       [ 5.,  8.,  8.],
       [ 8.,  8.,  9.],
       [ 8.,  9., 10.],
       [ 9., 10., 10.],
       [10., 10., 10.],
       [10., 10., 10.],
       [10., 10., 11.],
       [10., 11., 11.],
       [11., 11., 16.],
       [11., 16., 16.],
       [16., 16., 17.],
       [16., 17., 18.],
       [17., 18., 18.],
       [18., 18., 18.],
       [18., 18., 18.],
       [18., 18., 18.],
       [18., 18., 18.],
       [18., 18., 20.],
       [18., 20., 22.],
       [20., 22., 22.],
       [22., 22., 23.],
       [22., 23., 24.],
       [23., 24., 26.],
       [24., 26., 26.],
       [26., 26., 28.],
       [26., 28., 30.],
       [28., 30., 31.],
       [30., 31., 32.],
       [31., 32., 32.],
       [32., 32., 34.],
       [32., 34., 39.],
       [34., 39., 40.],
       [39., 40., 41.],
       [40., 41., 42.],
       [41., 42., 42.],
       [42., 42., 44.],
       [42., 44., 45.]])
```



# LSTM 架構

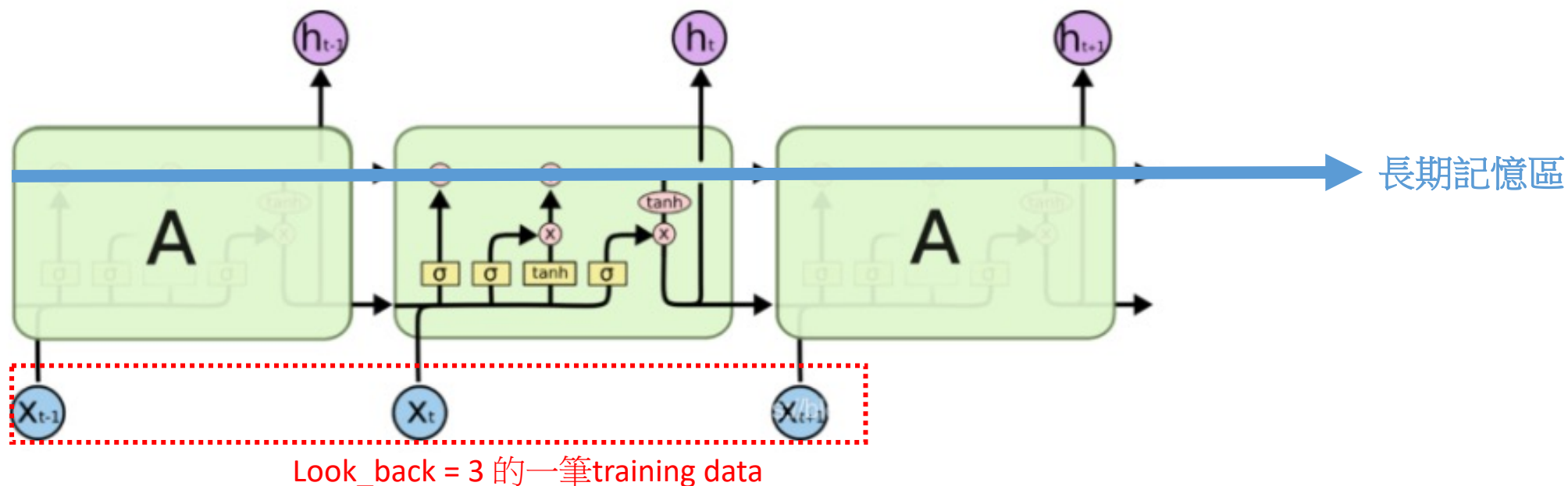
- 根據look back概念處理完訓練資料後，我們就可以來看 LSTM 的運作機制
- LSTM 有多種模式的運作機制：



- 端看訓練資料以及label的維度來決定，以預測covid累積確診人數為例，訓練資料3維(look back = 3) 而預測維度一維(某單日累積確診數)，LSTM 是採取上述 **Many-to-one** 的模式。

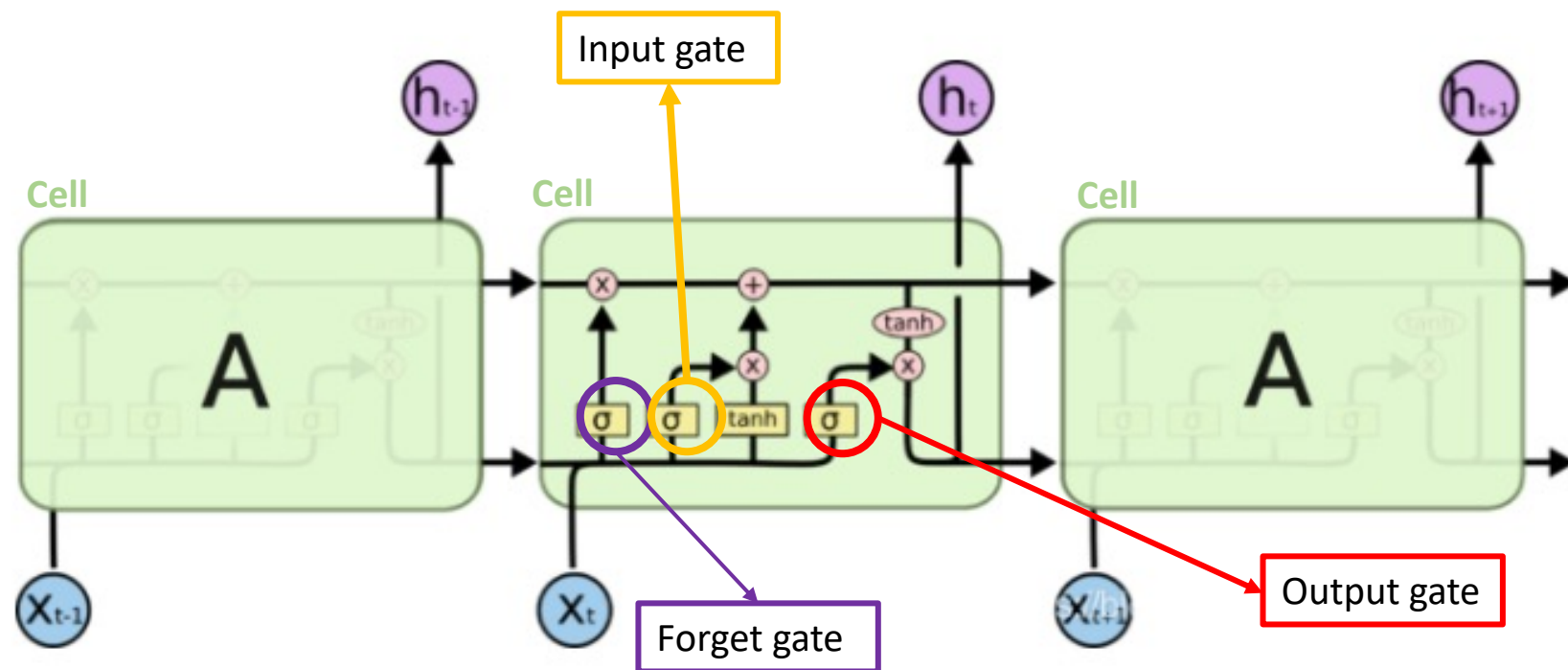
# LSTM 架構

- LSTM 如何建立不同資料間的關係以及彼此之間怎麼傳遞訊息
- 長期記憶區就像是個輸送帶一樣，訊息在上面進行輸入、修改、傳送的动作，以此來達到不同資料之間訊息的交流以及保存。



# LSTM 架構

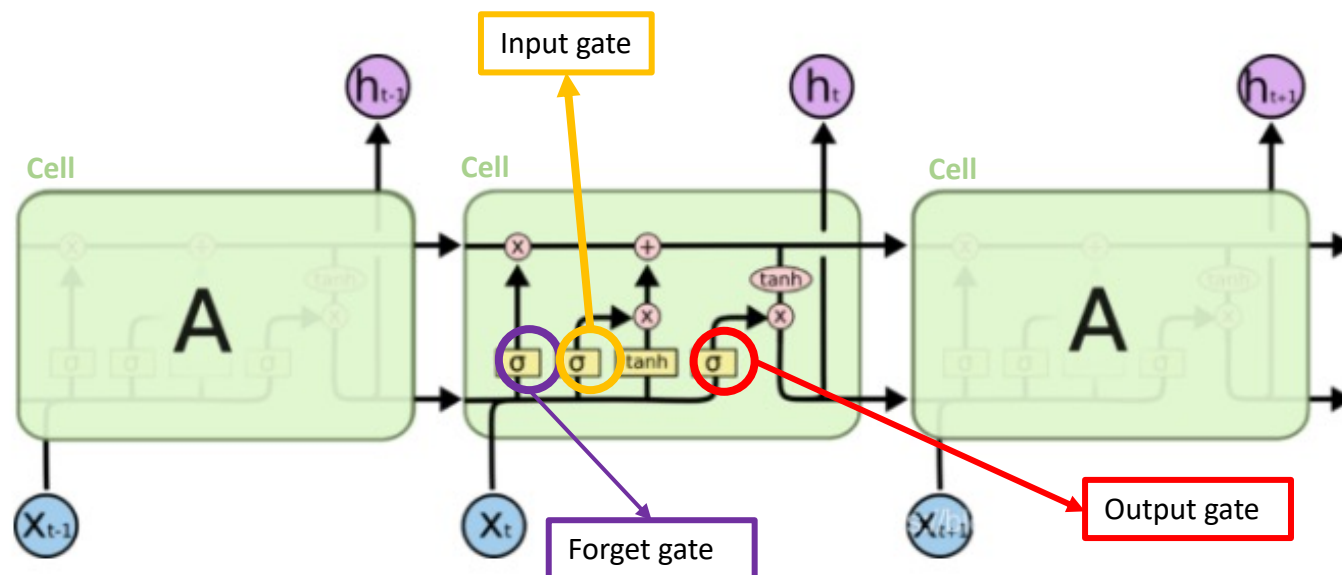
- 利用三個gate, 決定來自過去的訊息要保留多少、遺忘多少、如何使用過去訊息來與當下訊息進行計算
- Forget gate, Input gate, Output gate



# LSTM 架構

- 三個gate

- Forget gate : 累積至此的資訊，要忘記多少後再送入長期記憶區?
- Input gate: 累積至此的資訊要使用多少來進行一個cell的計算後，再送入長期記憶區?
- Output gate: 累積至此的資訊要留多少進入下一個cell?



# LSTM 架構

- 藉由上述 LSTM 的設計架構，我們就能讓 look back 範圍內的資料能夠充分的交流，最後再以彙整的資訊進行最後的 output。
- LSTM 最後 output 出來的 feature 會再當作後續 MLP 的 input，直到最後輸出一維的預測累積感染人數。

```
model = Sequential()
# look_back = 3
model.add(LSTM(20, input_shape=(look_back, 1), activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1))
#model.compile(loss='mean_squared_error', optimizer='adam')
model.compile(loss='MSE', optimizer='adam')
history = model.fit(trainX, trainY, epochs=100, batch_size=8)
```

# 實驗結果

- 對於預測累積確診、康復、死亡人數，我們都各自建立的model，並從486筆資料中切出296筆作為訓練資料，190筆為驗證資料

```
model = Sequential()
# look_back = 3
model.add(LSTM(20, input_shape=(look_back, 1), activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1))
#model.compile(loss='mean_squared_error', optimizer='adam')
model.compile(loss='MSE', optimizer='adam')
history = model.fit(trainX, trainY, epochs=100,
```

```
[ ] # split into train and test sets
    train_size = 300
    test_size = len(y_confirmed) - train_size
    trainX, testX = X[0:train_size], X[train_size:]
    trainY, testY = y_confirmed[0:train_size], y_confirmed[train_size:]
```

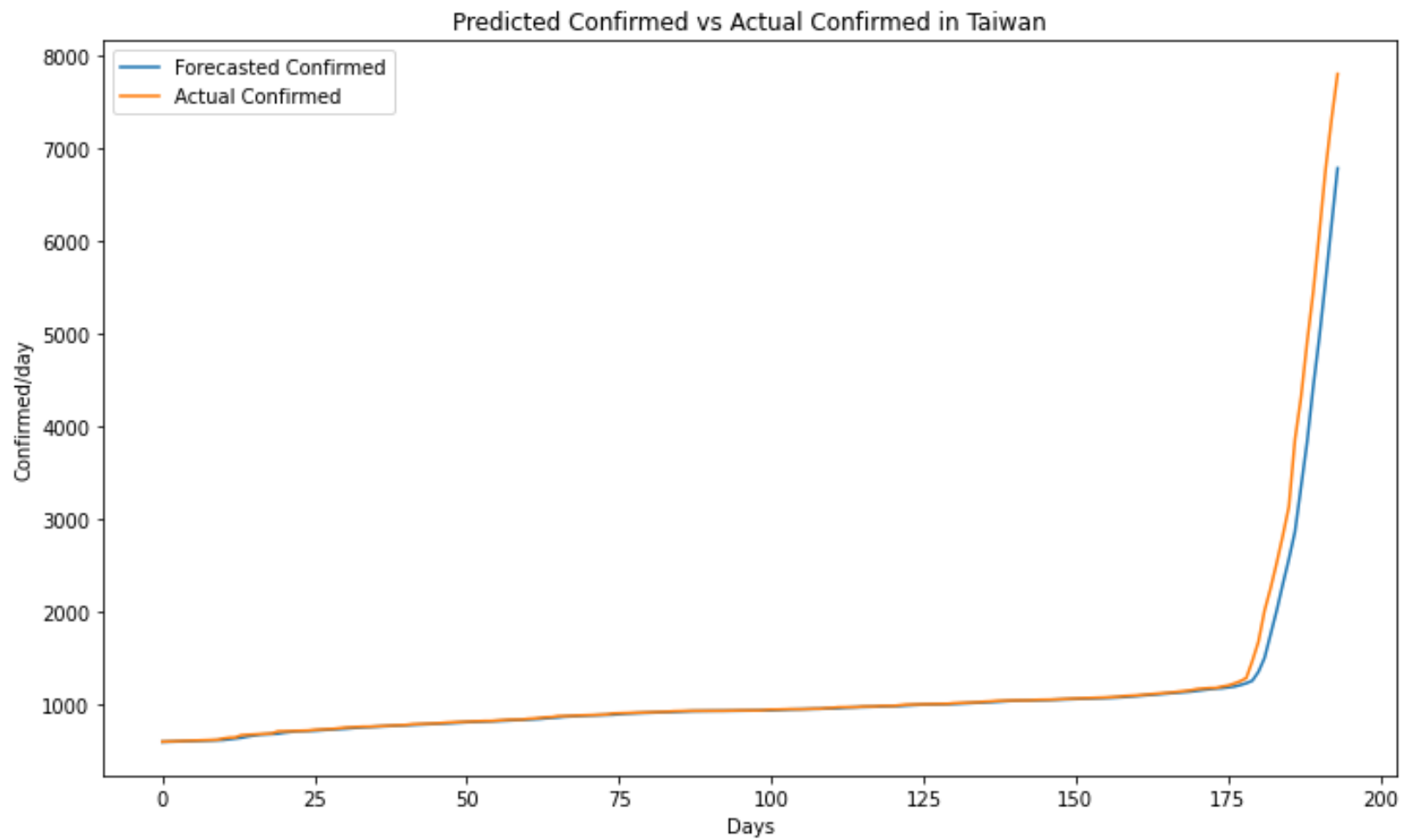
```
[ ] look_back = 3
    trainX, trainY = create_dataset(trainY, look_back)
    testX, testY = create_dataset(testY, look_back)
```

```
[ ] print(trainX.shape, trainY.shape, testX.shape, testY.shape)
```

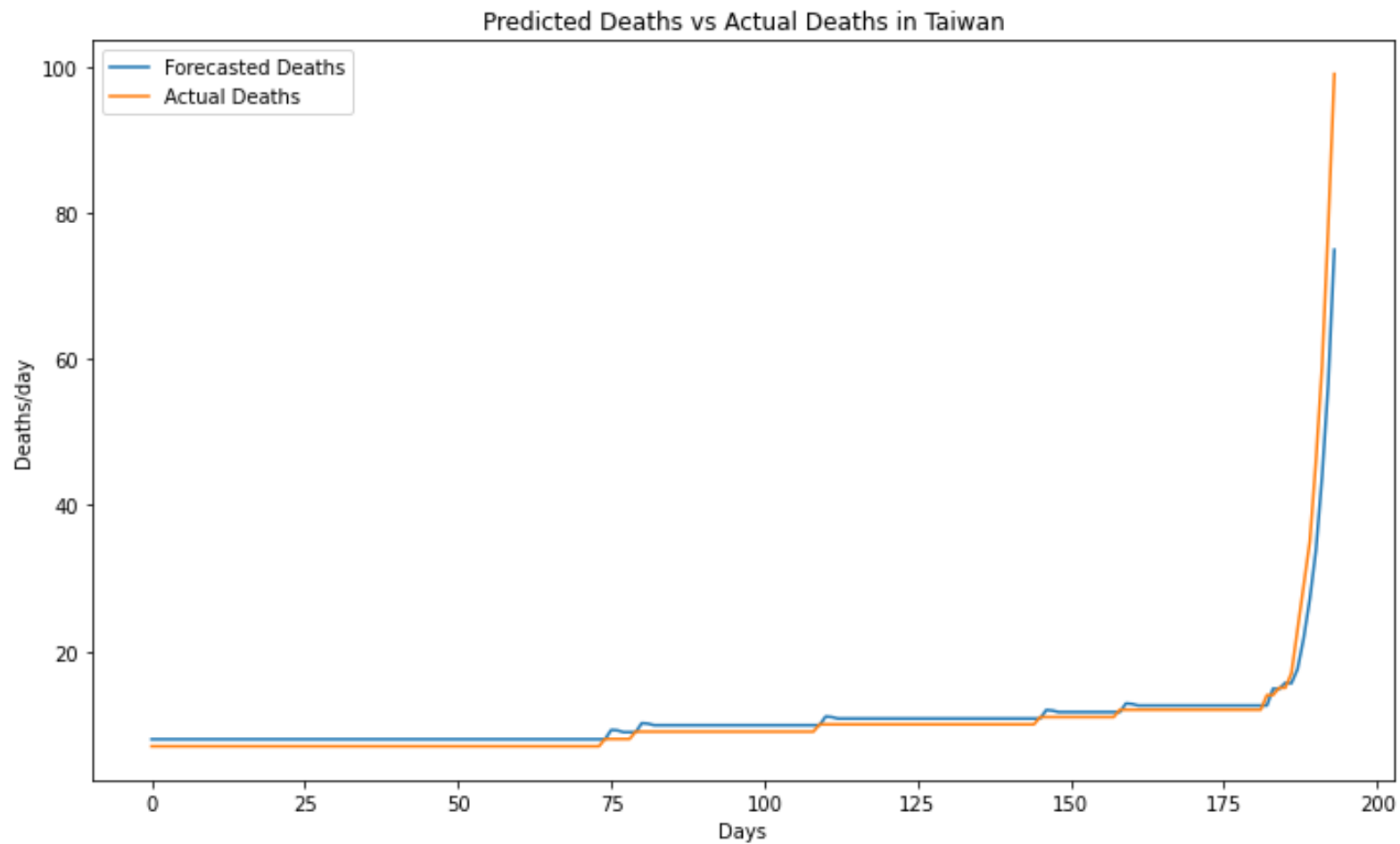
```
(296, 3) (296,) (190, 3) (190,)
```



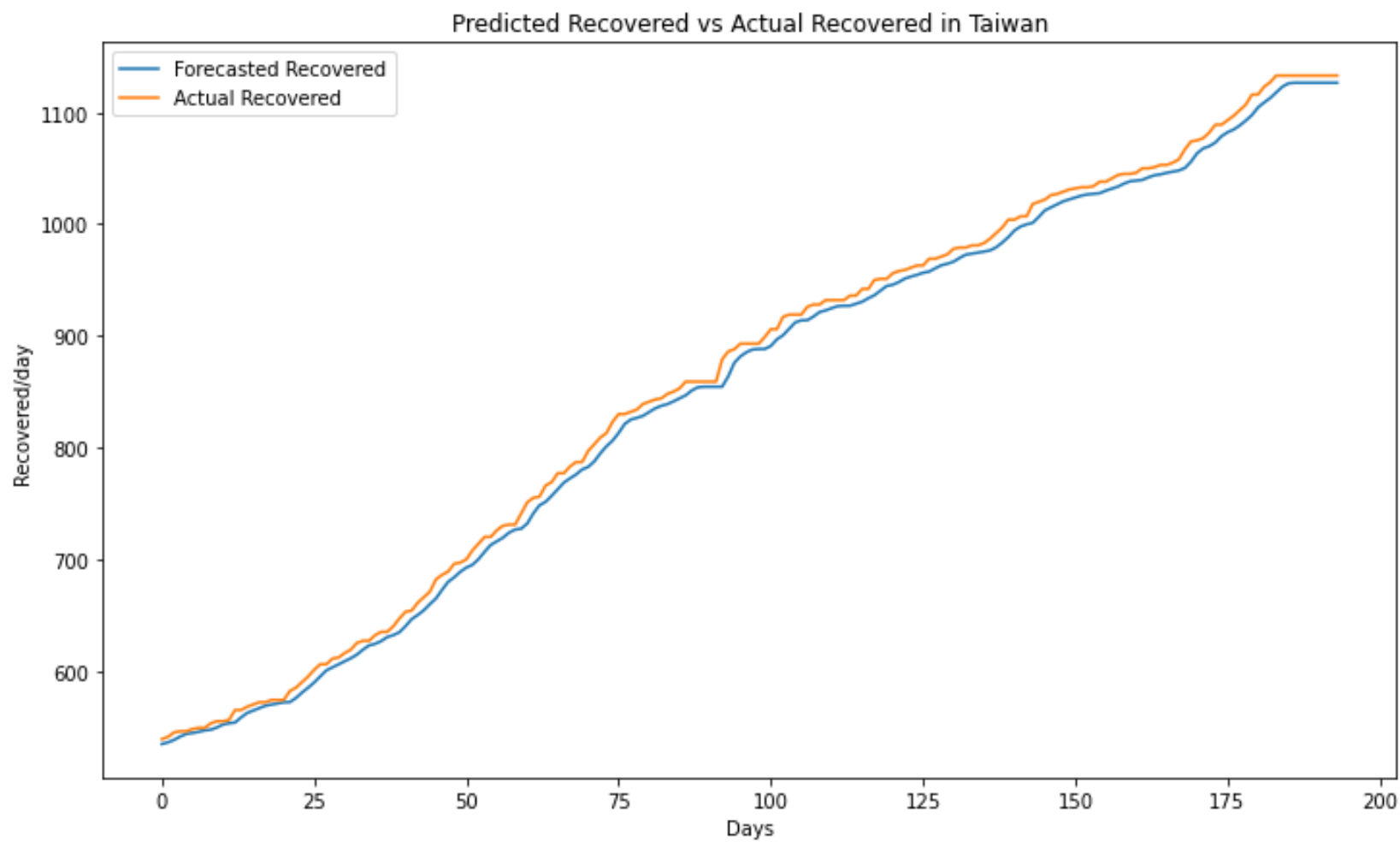
# 實驗結果



# 實驗結果



# 實驗結果



# 實驗結果

- 上述實驗結果顯示，利用**LSTM + MLP** 的模型架構，能夠很好的擬合於訓練資料並且預測出合理的未來趨勢