大數據分析
期末報告

M11007412

劉錕笙

問題描述

- 數據呈現與視覺化：

  在Covid-19的資料庫中，絕大多數的應用都在於統計疫情分佈與趨勢。因此如何呈現畫面讓使用者一目了然便非常重要。

- 未來趨勢預測：

  在Covid-19內的重要應用，如何使用對的預測模型是值得討論的。

新內容

- 數據呈現與視覺化：

  目前看到的方法多數使用靜止的圖像呈現，在此使用了Animation的方式製作了圖片。

- 未來趨勢預測：

  比較了多種辨識模型 :

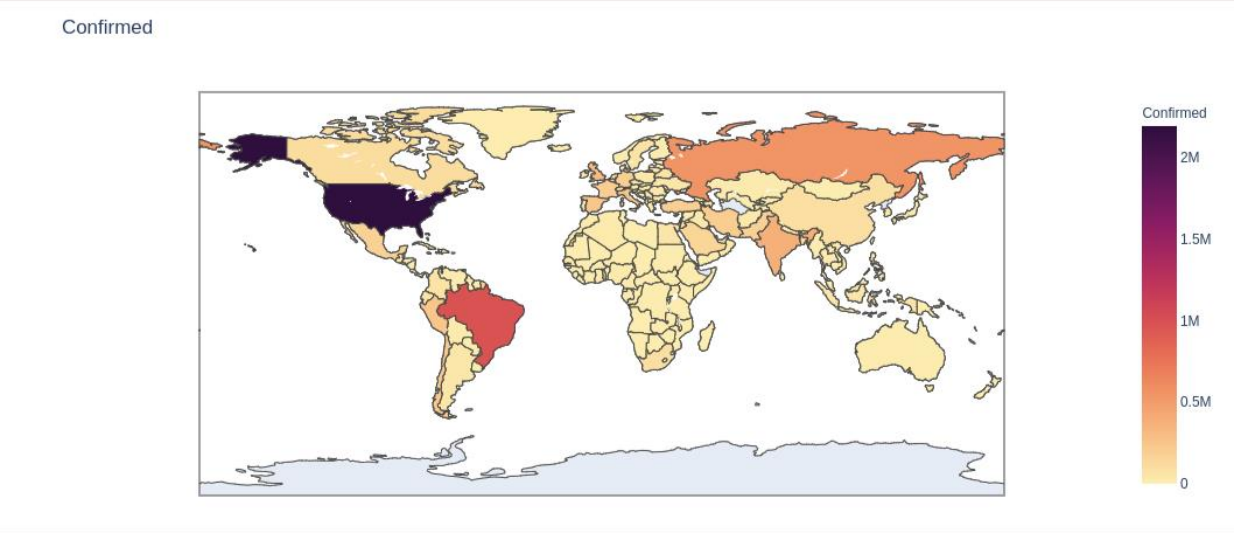
  Naive Approach, Moving Average, Holt Linear, ARIMA, And Prophet。

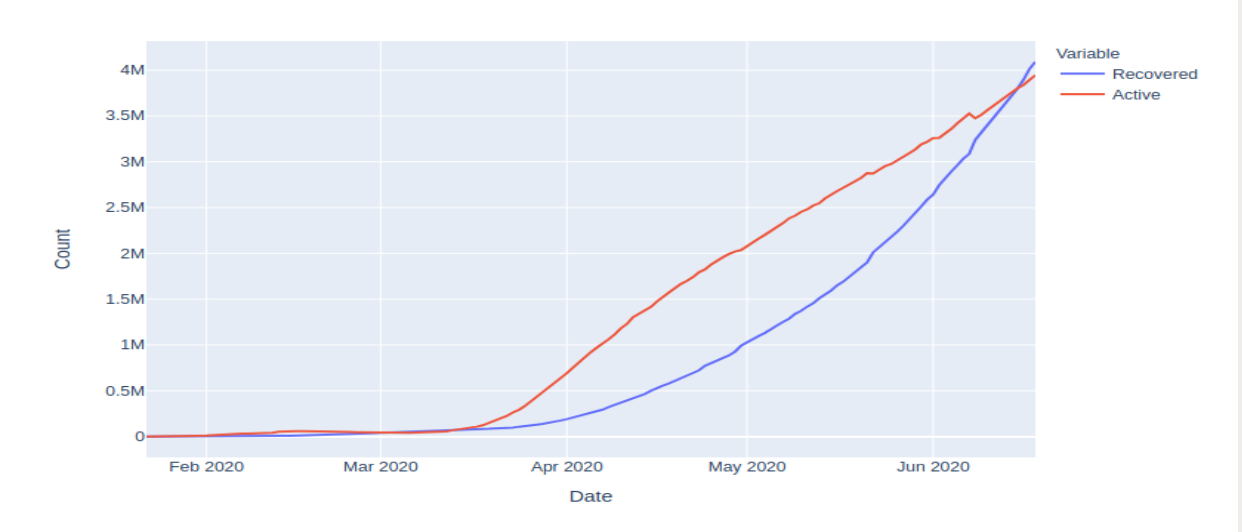
厲害指令

- Dataframe
- Animation
- Plotly
- plotly.graph_objects
- Statsmodels

視覺化：

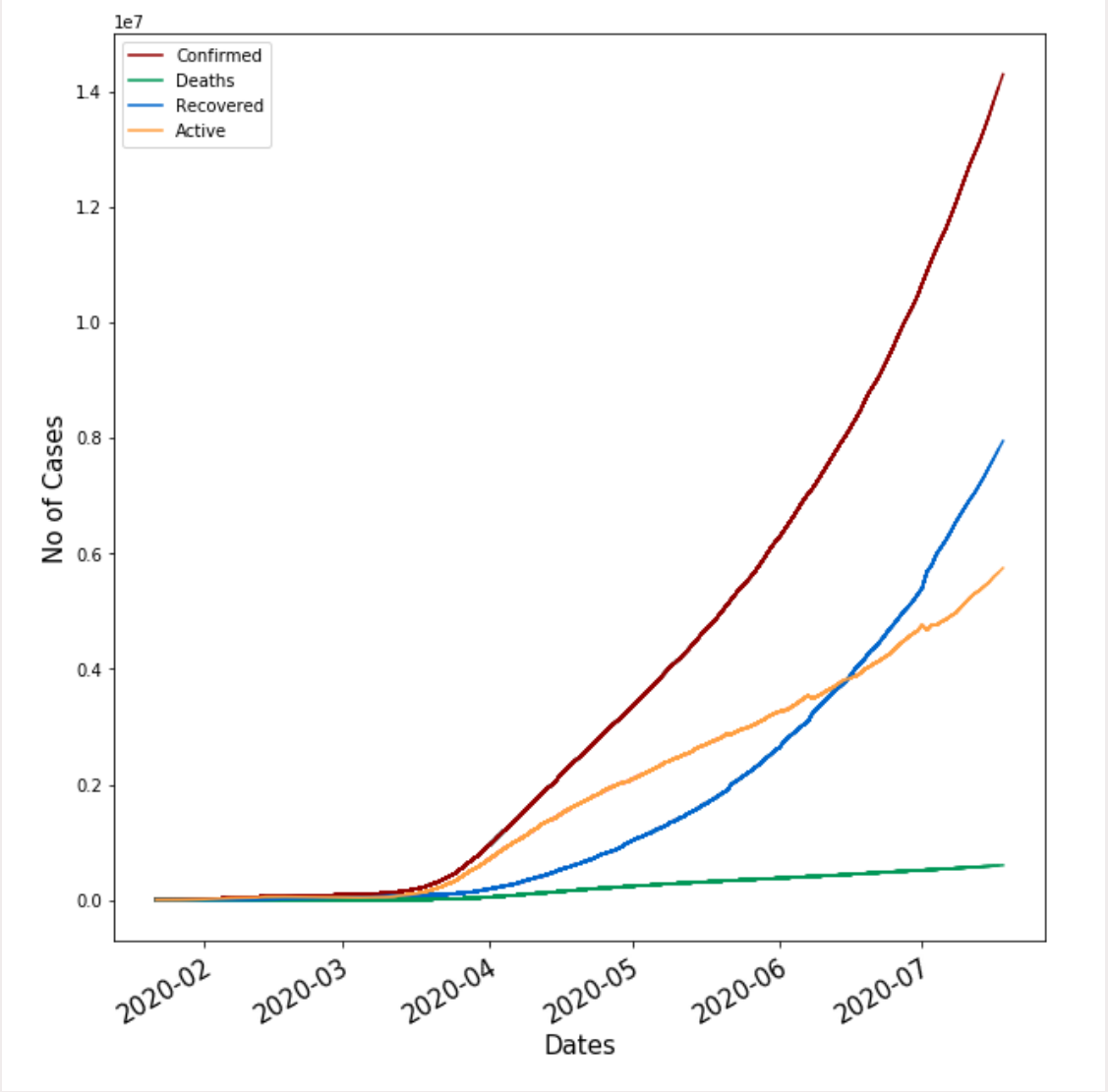
在期中的survey中我列舉了幾個許多人常用的分析呈現（如右圖）。讓使用者可以直觀的得知訊息，且不需要有專業的分析知識。



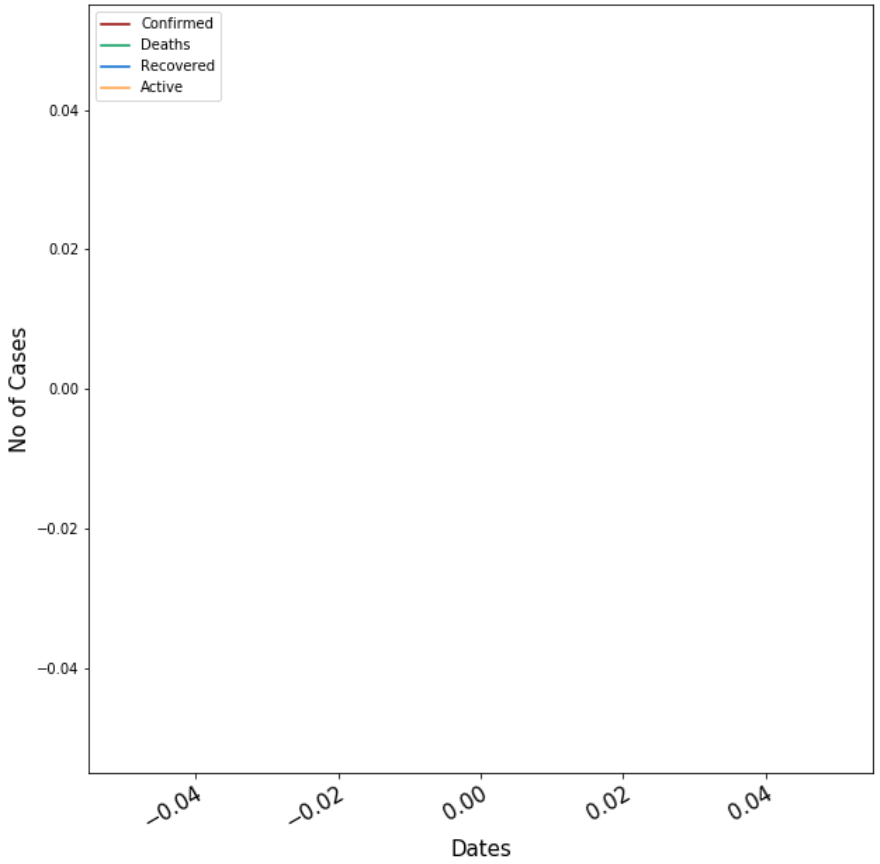


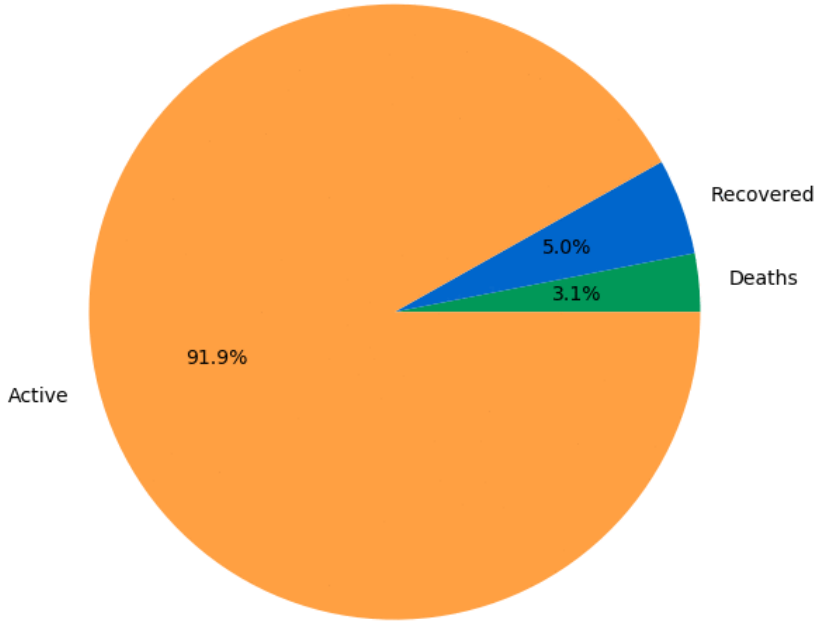
視覺化：

若是今天的畫面再複雜一點，便會使得畫面不再直觀。如右圖，我們無法一眼看出4條曲線的相對關係。同時也無法得知早期的數據走勢。



Python Library--Animation





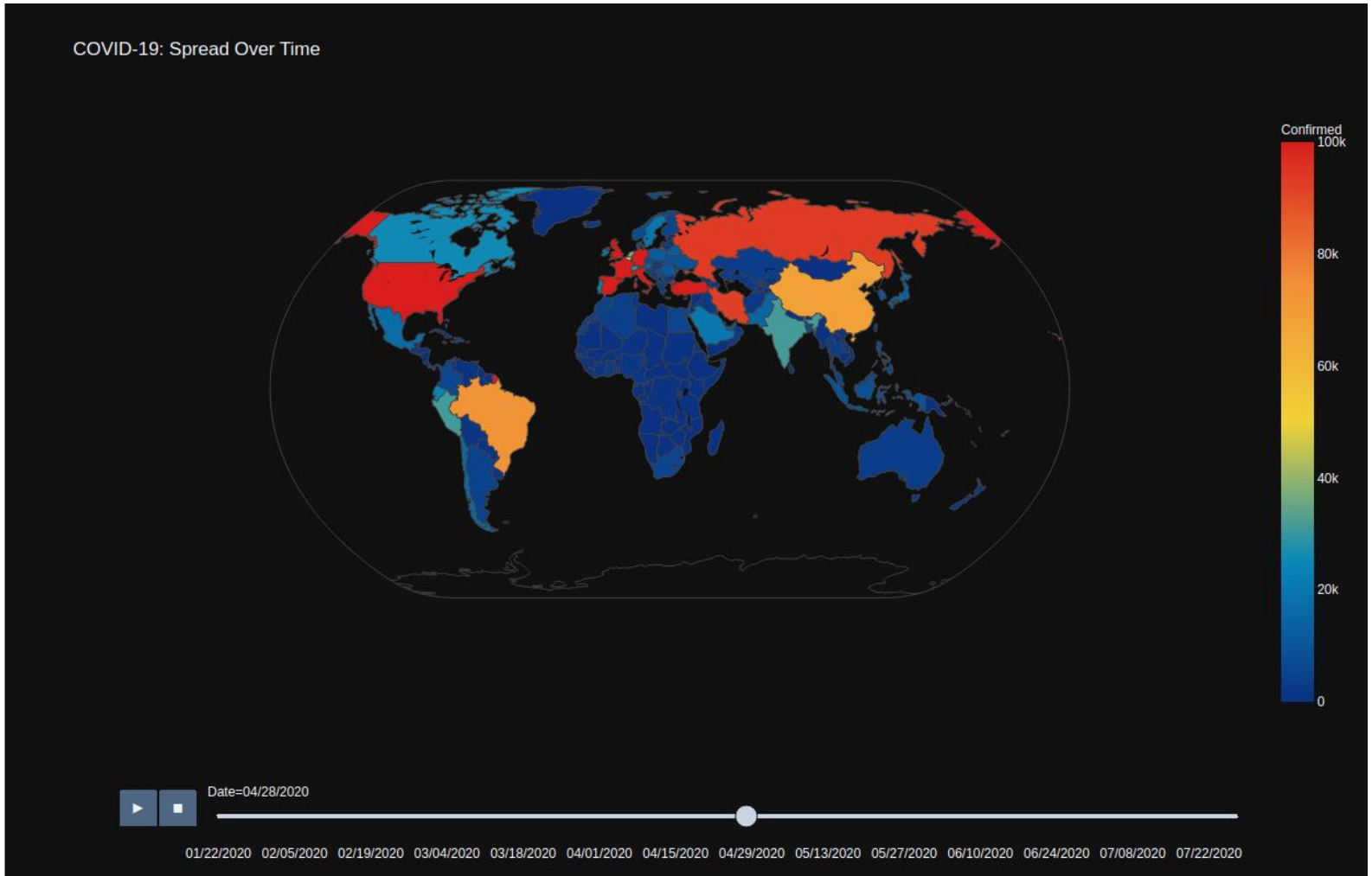
Matplotlib.Animation

```
def animate(i):
    plt.legend(["Confirmed","Deaths","Recovered","Active"],loc="upper left")
    line = plt.plot(day_wise[:i]["Date"],day_wise[["Confirmed","Deaths","Recovered","Active"]][:i].values)
    for i in range(0,4):
        line[i].set_color(color[i])
    return line

myAnimation = ani.FuncAnimation(fig, animate,frames=np.arange(0,180,1),interval=10, blit=True,repeat=False)
```

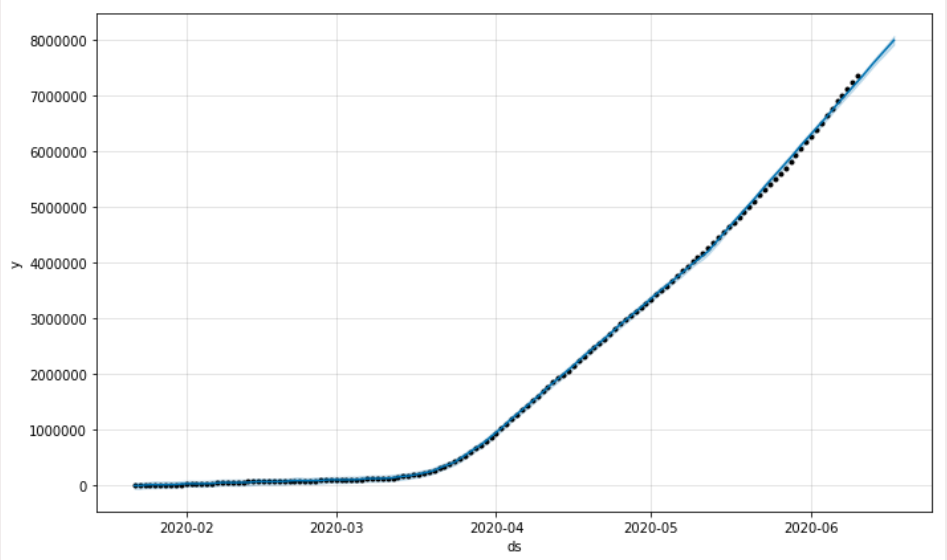
```
class matplotlib.animation.FuncAnimation(fig, func, frames=None, init_func=None,
fargs=None, save_count=None, *, cache_frame_data=True, **kwargs) ¶          [source]
```

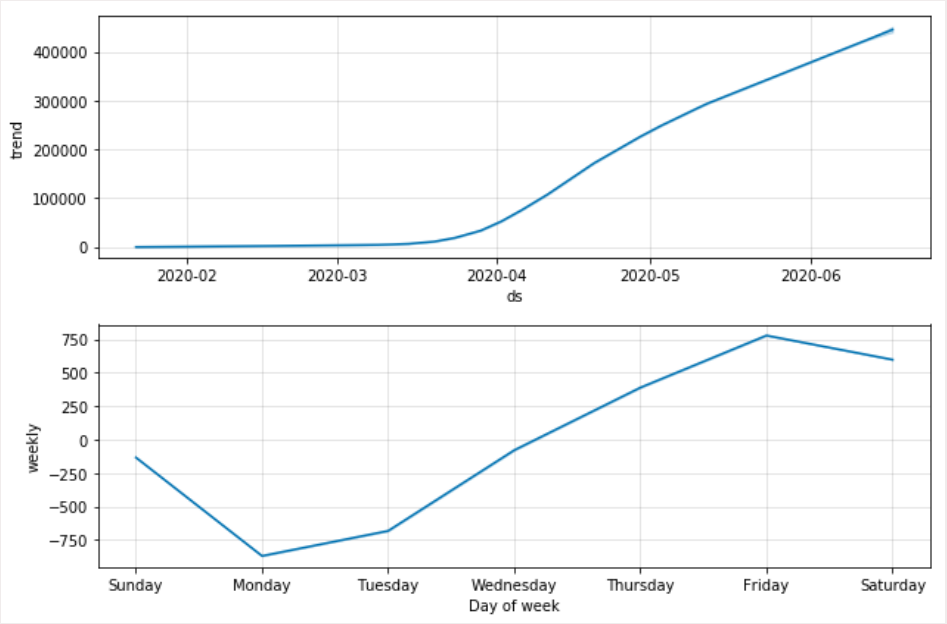
Python Library--Plotly



趨勢預測：

在期中的部份我展示了來預測的一個Library-Prophet，其結果如右圖。在我的報告中也指出了期短期預測的缺點。





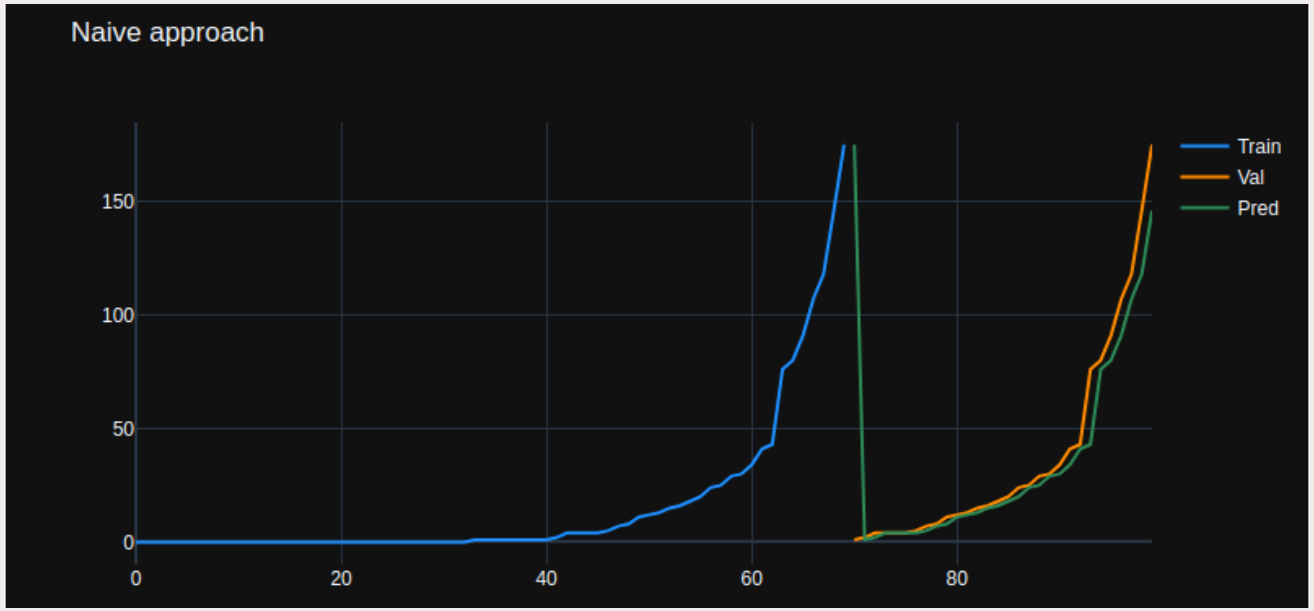
Naive Approach:

$$\hat{y}_{t+1} = y_t$$

此方法將上一階段的結果作為下一次的預測值，如右圖可看到其預測的結果是不準確的

```
predictions = []
for i in range(len(val_dataset.columns)):
    if i == 0:
        predictions.append(train_dataset[train_dataset.columns[-1]].values)
    else:
        predictions.append(val_dataset[val_dataset.columns[i-1]].values)

predictions = np.transpose(np.array([row.tolist() for row in predictions]))
error_naive = np.linalg.norm(predictions[:3] - val_dataset.values[:3])/len(predictions[0])
```



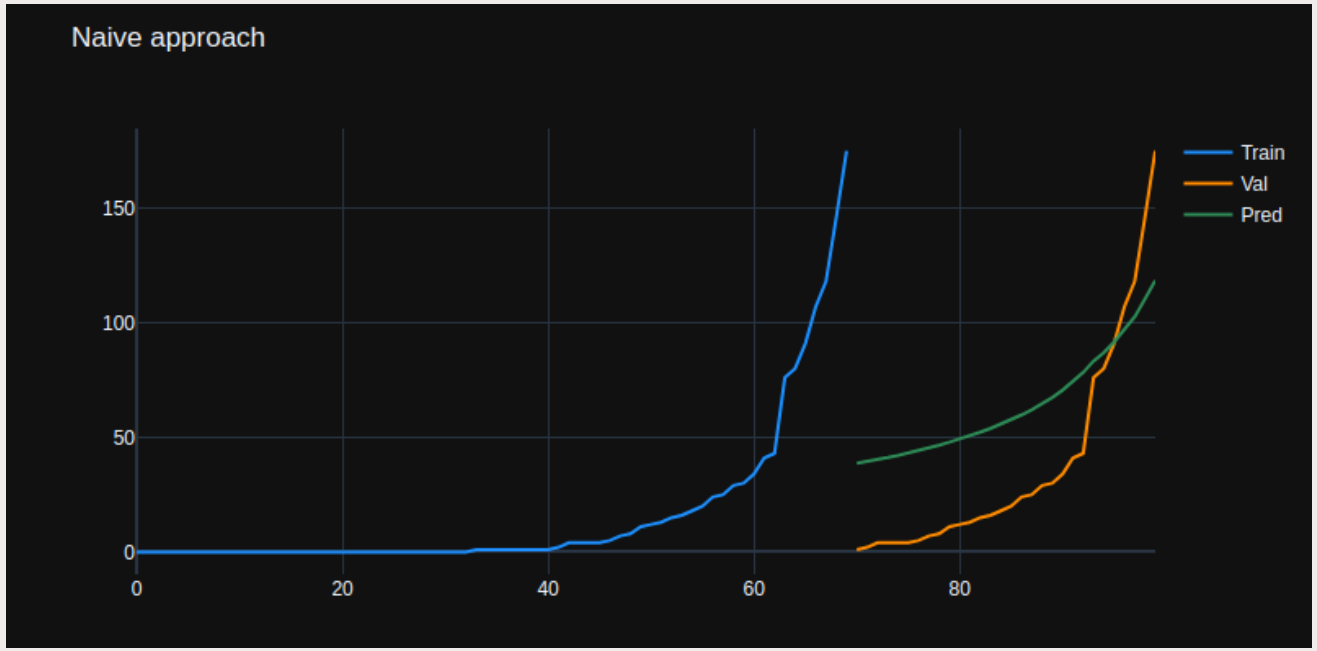
Moving Average:

$$\hat{y}_{t+1} = \frac{1}{30} \cdot \sum_{t-30}^{t} y_n$$

此方法的預測結果事前30個狀態的平均值。此方法的預測僅優於Naive Approach一點而已。

```
predictions = []
for i in range(len(val_dataset.columns)):
    if i == 0:
        predictions.append(np.mean(train_dataset[train_dataset.columns[-30:]].values, axis=1))
    if i < 31 and i > 0:
        predictions.append(0.5 * (np.mean(train_dataset[train_dataset.columns[-30+i:]].values, axis=1) + \
                                  np.mean(predictions[:i], axis=0)))
    if i > 31:
        predictions.append(np.mean([predictions[:i]], axis=1))

predictions = np.transpose(np.array([row.tolist() for row in predictions]))
error_avg = np.linalg.norm(predictions[:3] - val_dataset.values[:3])/len(predictions[0])
```



Holt Linear:

$$\hat{y}_{t+h} = l_t + h \cdot b_t$$

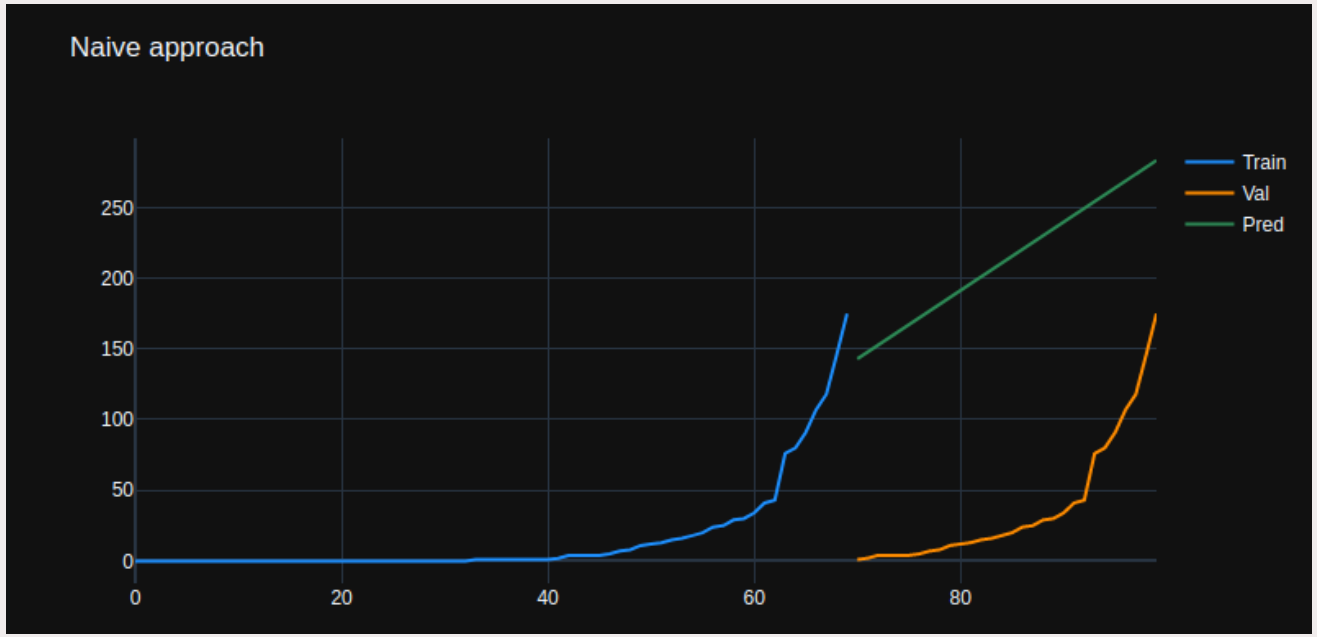
$$l_t = a \cdot y_t + (1 - \alpha) \cdot (l_{t-1} + b_{t-1})$$

$$b_t = \beta \cdot (l_t - l_{t-1}) + (1 - \beta) \cdot b_{t-1}$$

此方法主要是找出一條直線來作為未來的預測。其中的α和β為任意常數，借由此函式求出的l_t和b_t作為預測線的斜率和位移，繪製出預測的線段。

```
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt

predictions = []
for row in tqdm(train_dataset[train_dataset.columns[-30:]].values[:3]):
    fit = Holt(row).fit(smoothing_level = 0.3, smoothing_slope = 0.01)
    predictions.append(fit.forecast(30))
predictions = np.array(predictions).reshape((-1, 30))
error_holt = np.linalg.norm(predictions - val_dataset.values[:len(predictions)])/len(predictions[0])
```



ARIMA:

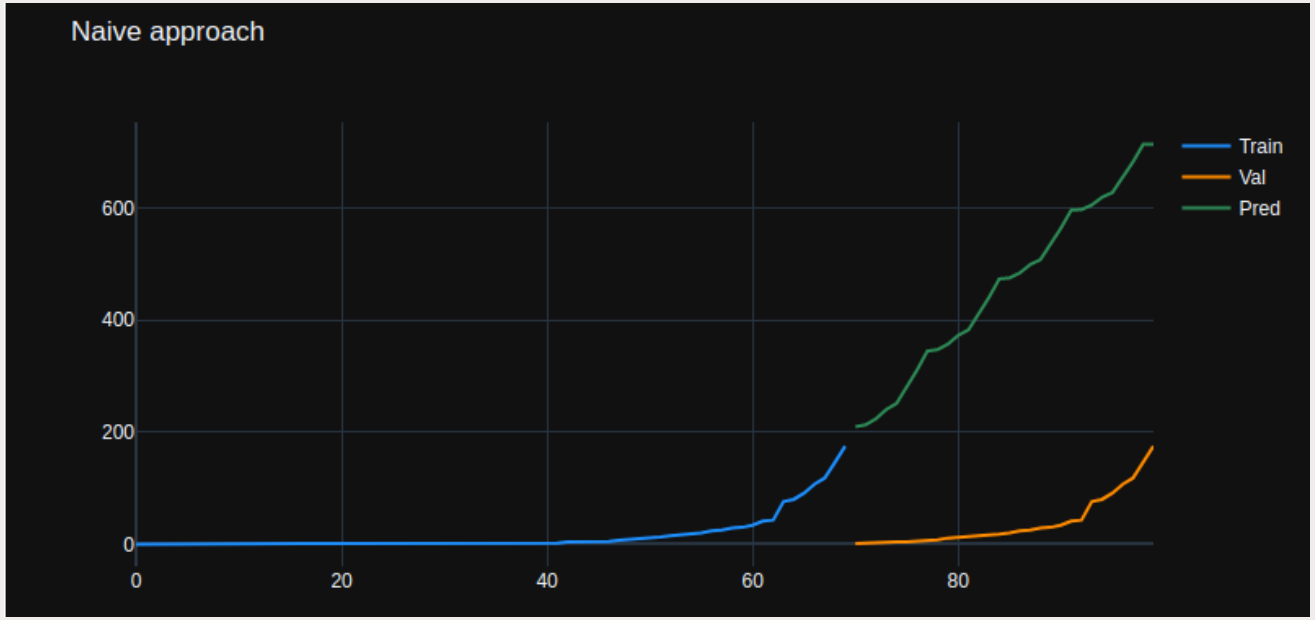
$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_0 Y_0 + \epsilon_t$$

$$Y_{t-1} = \beta_1 Y_{t-2} + \beta_2 Y_{t-3} + .. + \beta_0 Y_0 + \epsilon_{t-1}$$

此方法類似於泰勒級數的展開去模擬出原函數的多項式。因此在預測上有良好的結果。

```
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARIMA

predictions = []
for row in tqdm(train_dataset[train_dataset.columns[-30:]].values[:3]):
    fit = sm.tsa.statespace.SARIMAX(row, seasonal_order=(0, 1, 1, 7)).fit()
    predictions.append(fit.forecast(30))
predictions = np.array(predictions).reshape((-1, 30))
error_arima = np.linalg.norm(predictions[:3] - val_dataset.values[:3])/len(predictions[0])
```



結論

- 我認為將靜止的圖片轉換為動態的呈現方式，的確讓使用者在觀看數據時能更加生動。
- 使用不同Animation 函式庫，比較不同資料庫的執行時間和使用難易度。
- 比較預測模型，未來可以分析不同模型在不同情況(日、週、月)下的適應度。

END