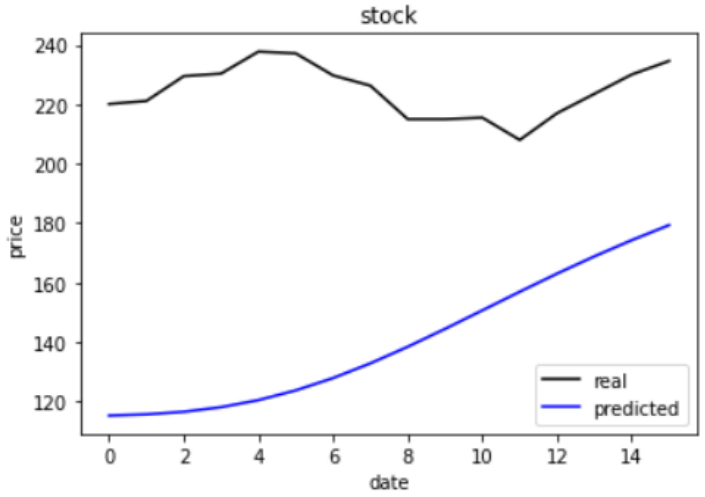
尋找股價最佳參數

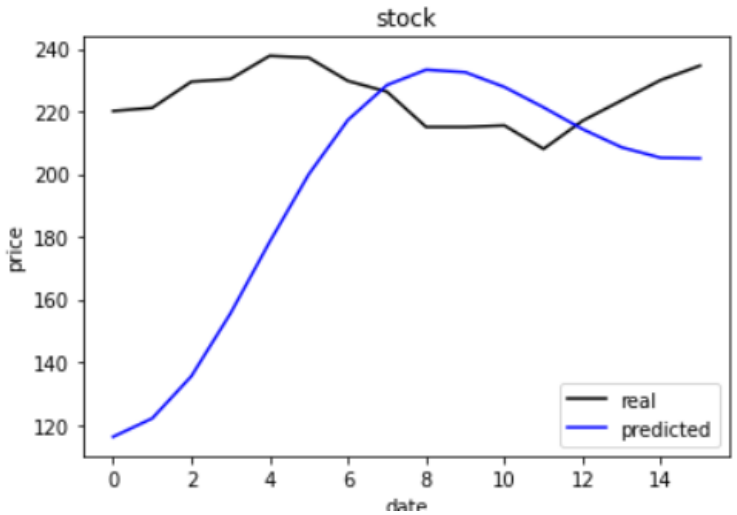
1. 原圖，所有參數皆和老師測試的一樣。



   - ✔ 結果：可以說是和原本的沒有什麼關係，但最後坡度的形狀有點類似。
   - ✔ 反思：因為回合數太少，所以準備下一步先改變回合數來觀測線條坡度的變化。

2. 更改**epochs**的數量：**30**回合

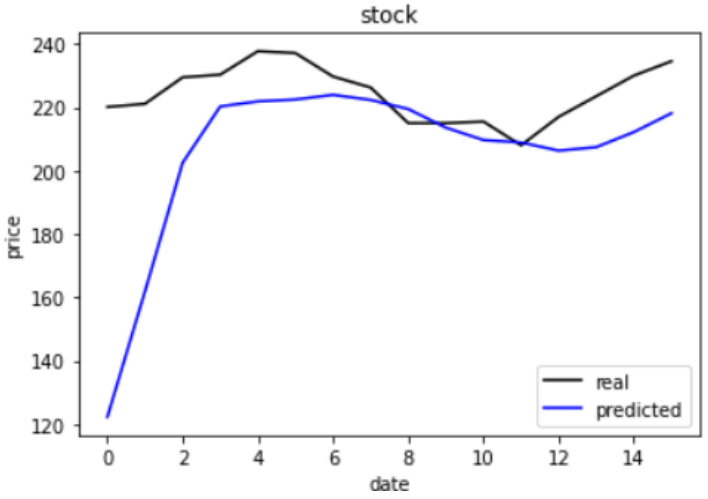
```
model.fit(x_train, y_train, epochs=30, batch_size=32)
```



   - ✔ 結果：稍微比原圖好了那麼一點點點，在6到12的過程中比較有變接近實際數字，但其他的差距還是相當大，坡度形狀的預測也不如預期。
   - ✔ 反思：30回合的可能也不算高，再進行一次比較多回合的預測，藉此來感受多次回合的訓練對於這樣的數據影響力有多少。

3. 更改**epoch**的數量：**80**回合

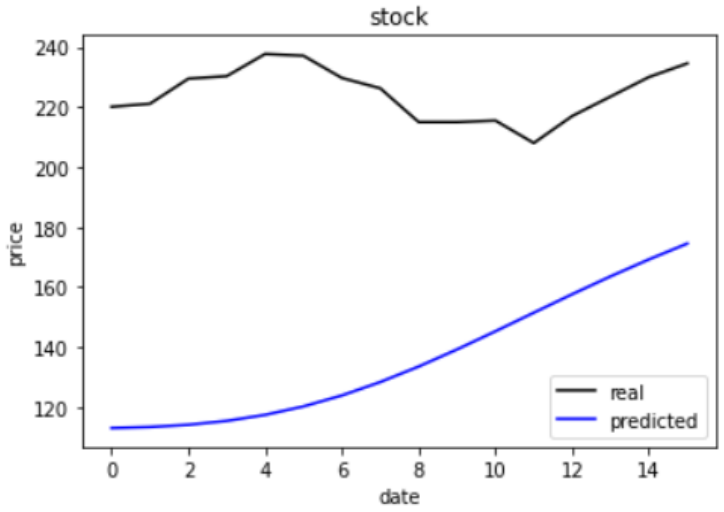
```
model.fit(x_train, y_train, epochs=80, batch_size=32)
```



- ✔ 結果:訓練80回的效果顯然明顯的上升了！不但在坡度的改變尚有相當一定的吻合，而且誤差也明顯地有下降。
- ✔ 反思:看來回合數對於訓練一個類神經網路來說真的是相當重要，回合數越多，數據也就更加準確，我想就老師上課所說的，loss函數有適合數字預測的、也有適合預測圖片的，來改這個試試。

4. 更改**loss**函數:**binary_crossentropy**，**epoch**調回**1**

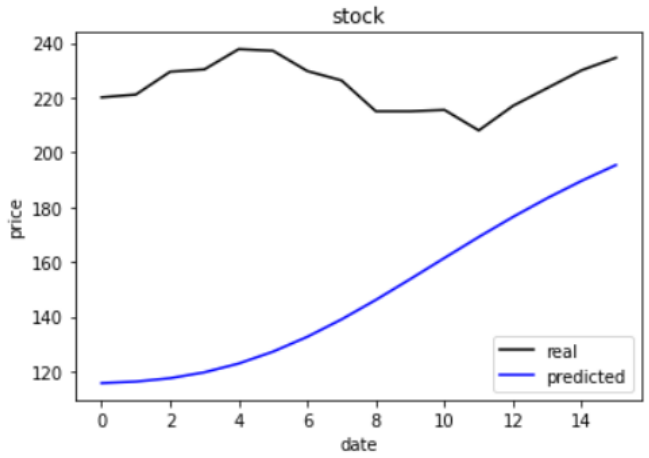
```
model.compile(optimizer='adam', loss='binary_crossentropy')
```



- ✔ 結果: 可以說是和原本的沒有什麼關係，但最後坡度的形狀有點類似。
- ✔ 反思:基本上epoch改回1的時候，不管損失函數設什麼，結果都蠻一致的，就是和原本的真實股價沒什麼區別，那來看看回合數30的時候會有什麼樣的改變。

5. 更改**loss**函數:**binary_crossentropy**，**epoch**調至**30**

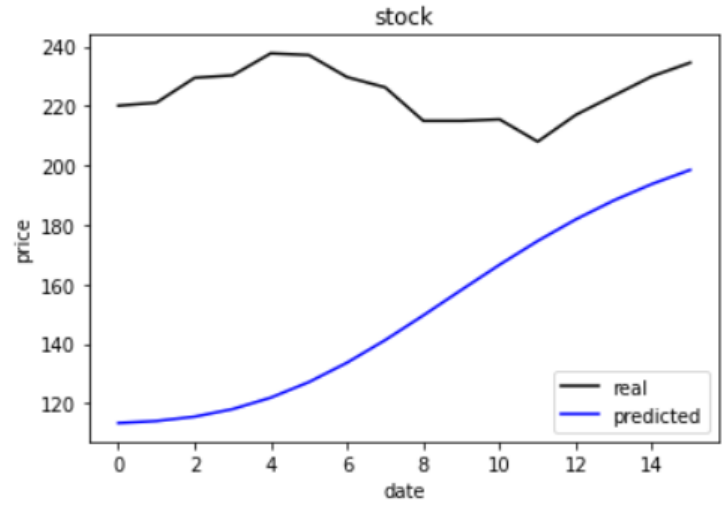
```
model.compile(optimizer='adam', loss='binary_crossentropy')
model.fit(x_train, y_train, epochs=30, batch_size=32)
```



- ✔ 結果：可以說是和原本的沒有什麼關係，但最後坡度的形狀有點類似。
- ✔ 反思：完全沒有因為回合數的增加而有更不一樣的曲線，在原本基礎上只更改回合數的情況之下，本來可以在日期6-12的時候有相對準確的估算，但更改了損失函數之後即使回合數增加了，還是像是回合數只有1的情況一樣，接下來會再試一次回合數增加，來看損失函數是binary_crossentropy的情況下，回合數再多是否都沒有用處。

6. 更改**loss**函數：**binary_crossentropy**，**epoch**調至**60**

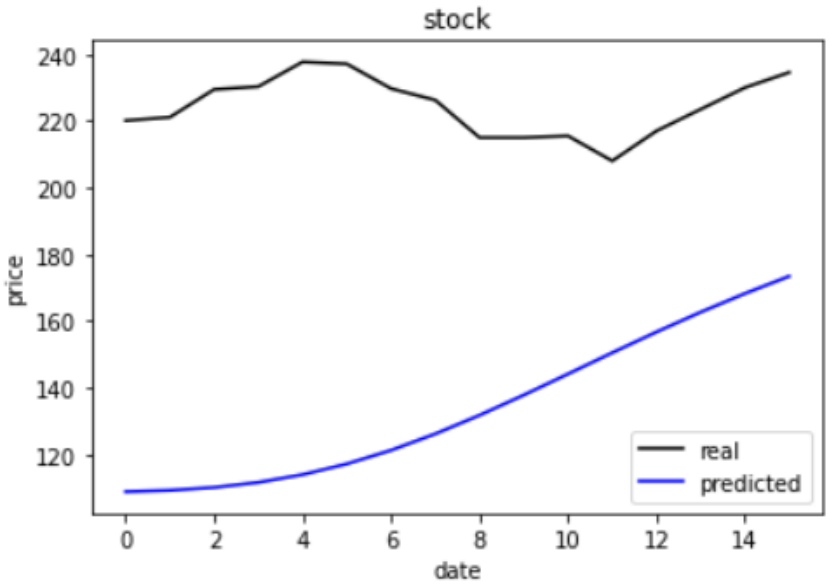
```
model.compile(optimizer='adam', loss='binary_crossentropy')
model.fit(x_train, y_train, epochs=60, batch_size=32)
```



- ✔ 結果：完全沒有變化！
- ✔ 反思：看來損失函數錯誤，不管回合數再多都是無用功呢。那接下來就把損失函數調回來，接著去改改看批次。

7. 損失函數改回原本的，更改**batch_size**：**16**，**epoch**設為**1**

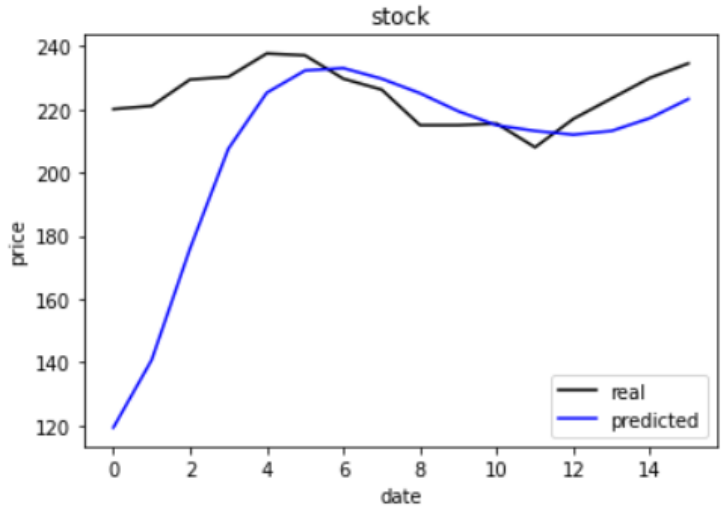
```
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(x_train,y_train,epochs=1,batch_size=16)
```



- ✔ 結果:和一開始的完全一樣，還看不出來batch_size對於這整體的預測是什麼影響。
- ✔ 反思:可能是因為回合數太少，所以還看不出有什麼變化，現在我把回合數調高一點看batch_size對於這整體的預測有什麼影響。

8. 更改**batch_size**:**16**，**epoch**設為**30**

```
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(x_train,y_train,epochs=30,batch_size=16)
```

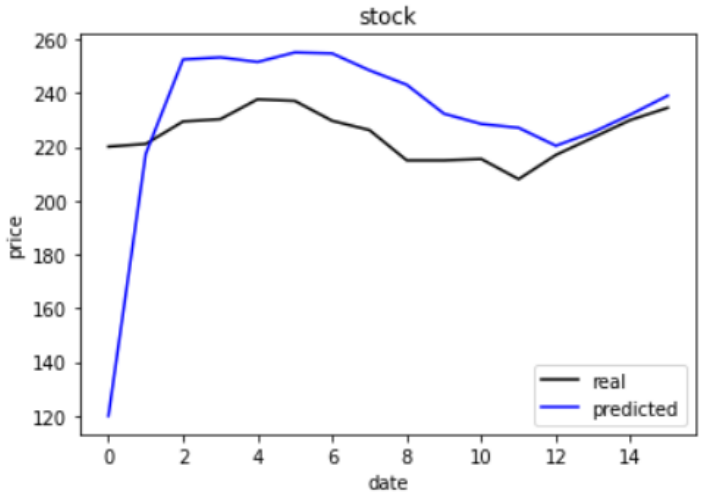


- ✔ 結果:和我的第二步驟相比起來，線條差不多，誤差也差不多。
- ✔ 反思:以上的結果看來還是epoch的影響最大，因為批次越小，它跑一回合的時間就會越久，但實際跑出來的東西和批次大的也沒有差很多，綜

上所述，我決定最後一步就是調整回合數到很大，來做能力範圍內最精準的預測。

9. 更改**batch_size**回**32**，**epoch**設為**200**

```
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(x_train,y_train,epochs=200,batch_size=32)
```



- ✔ 結果：整條曲線基本上位於real之上，但各區段的坡度非常相似！
- ✔ 反思：太驚奇了~~雖然照著我預測的去買可能還是會破產，但是它200回合基本上就可以把各區段的坡度模擬出來了，最後12-14之後的數據看起來也是誤差值很小。

10. 總結

測試過這個多次之後發現，其實現在的神經網路都有一個比較好的模板，而我要輸入的資料就可以依照資料的特性去套用較適合的模板，做出來的東西就會相對準確，而回合數的部分真的就是要靠電腦加油跑，越多次就越可以展現準確的數字，雖然最後調整完之後還是只有更改回合數而已，但觀察每一項參數造成的變化也是一件有趣的事情！