Machine Learning 2021 FALL

Final Project

G-Research Crypto Forecasting

組別：第 11 組

組員：

R10921098 楊仁傑

R10921057 鄭凱鴻

B06505003 陳祐融

Catalogue 目錄

Description 題目概述

隨著加密貨幣的重要性日益增加，其交易量也大幅上升，然而當中參雜許多的因素—包括物理或心理因素、理性或者不理性行為等等，使得價格波動劇烈，如果我們能精準預測其未來的走勢並以此作為投資與否的標準就可以在最小化風險的情況下最大化收益，這也使其變成現今人們是廣泛探討的課題之一。

由於加密貨幣的趨勢大幅受到資料的不確定性、投資人的行為、不同種類間的相互關聯、國內與國際市場趨勢等的影響，相關預測工作的難度也大幅提高，演算法本身不僅要能排除不相關資料與雜訊的影響，也要能從瞬息萬變的走勢中精準作出判斷。

過往的文獻資料已經有許多類似的預測模型，其大多藉由當下的資料產生一連續的移動平均再由此作進一步的預測，然而這一方面也意味著會將一部分的測試資料作為訓練用資料而與原本定義上的預測有所出入。為了避免這個現象，我們會將過往蒐集的數據提供給 model 作訓練以產生未來幾分鐘的資料再將其取平均作為最後的預測。此外，我們也會分析與探討幣種間的權重對模型造成的影響以及模型對於不同幣種的訓練校果。

Change Title 題目更動

在 Kaggle 上遇到的問題：

1. Kaggle 上的分數沒辦法作為衡量的標準

由於用來測試的資料(2021/06/01-2021/09/21)已經包含在給定的檔案中，所以只要將這部分的資料上傳正確率就可以達到 0.999，因此沒辦法但從分數判斷模型的表現。

2. 將移動平均作為訓練的資料，在上傳時會產生 error

我們在多次的繳交中發現，如果以 log 處理一段時間的資料，雖然可以成功產生預測，但在上傳時有很高的機率會跑超過九小時而會顯示 runtime error，即使對 nan 值補 0 也沒辦法解決，後來在查詢時推測有可能是規定用來上傳的 API 陷入無窮迴圈或計算上產生錯誤而導致的現象。

3. 需要使用官方所提供的虛擬環境

在繳交相關成績時，需要使用官方所提供的 gresearch_crypto.make_env()，那就需要使用到 Kaggle 的筆記本，效率比起自己的電腦而言有一段性能落差。同時也沒辦法叫好的評估各個模型之間的對比。

題目更動：

由於上述 Kaggle 上遇到的問題，我們在經過討論以後決定在期末報告上調整題目如下。

1. 由於 Kaggle 上的測試資料包含在訓練資料中，並無法達到驗證模型準確度的效果，因此我們決定將原本的訓練資料，依照 Train 70%, Valid 15%, Test 15%的比例切割，這樣就沒有測試資料包含在訓練資料中的問題
2. 由於不用在 Kaggle 上比準確率，所以為了簡化分析，我們將問題簡化為：只看預測資料與真實資料的 MAE、RMSE，同樣也可以表示模型的準確率。

Introduction 理論介紹

MA(Moving Average，移動平均線)

MA 在技術分析中常常被用來輔助觀察對趨勢的判斷，是以道·瓊斯的「平均成本概念」為理論基礎，採用統計學中「移動平均」的原理，將一段時期內的股票價格平均值連成曲線以代表過去一段時間裡的平均成交價格，主要目的是用於判斷趨勢，預期市場當前及未來有可能的走勢。

計算方式：

將 N 個時段的收盤價加總後取平均，即可得到第 N 個時段的算數平均線。

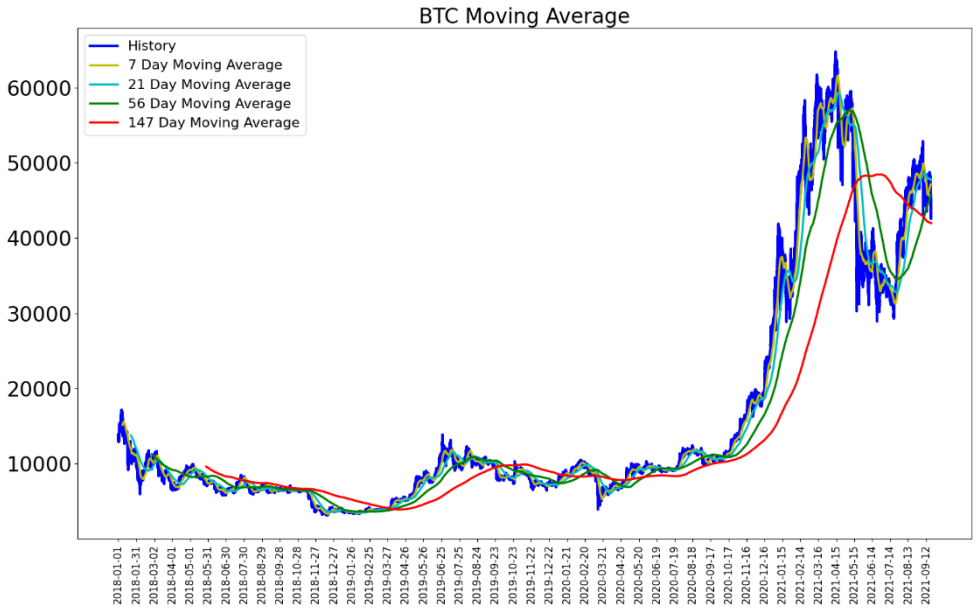

圖 3.1 BTC MA 趨勢圖

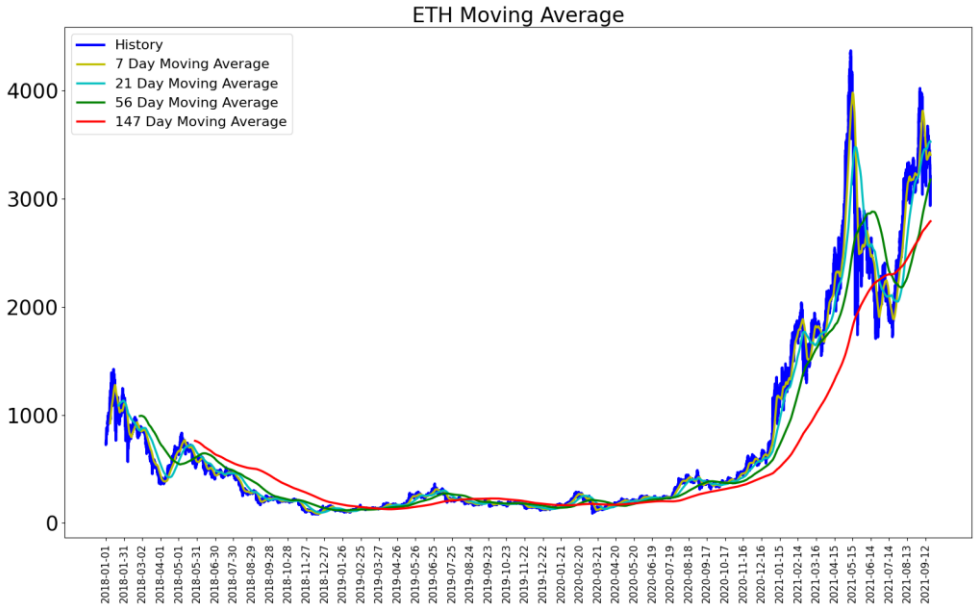

圖 3.2 ETH MA 趨勢圖

從圖 3.1 和圖 3.2 中得知使用 MA 可以預測相關趨勢，並且搭配不同時段的 MA 可以衍伸出"突破"、"死亡交叉"、"黃金交叉"等等相關技術指標，但是由於 MA 是一種過去市場平均價格的結果並視所有值都為等價貢獻，屬於落後指標，進場訊號較模糊而無法反應時間的重要性，不適用上下行情整理的盤型，因此我們接下來會提出其他預測價格的機器學習模型。

RNN(Recurrent Neural Network, 循環神經網路)

RNN 是一種以序列數據為輸入資料，按照序列的傳遞方向進行遞迴並且所有節點依照順序連接的神經網路。

其中較為常見的 RNN 有 Bi-RNN(Bidirectional RNN, 雙線循環神經網路)和 LSTM(Long Short-Term Memory networks, 長短期記憶網路)。而 LSTM 能夠記住需要長時間記憶的，忘記不重要的信息；而不像普通的 RNN 那樣只能有一種記憶疊加方式。對很多需要"長期記憶"的任務來說，較為適合使用 LSTM。

RNN 進行預測（或回歸）的時候，不僅要考慮到當前時刻的輸入，還要考慮上一個時刻的輸入。也就是說，預測的結果不僅與當前狀態有關，還與上一個時刻的狀態有關。與我們在上個小節提到的 MA 有類似的概念，因此在本次報告中我們將會探討以 LSTM 作為模型的不同結果。

ARIMA(Auto Regressive Integrated Moving Average)

ARIMA 是一種基於時間序列歷史值和歷史值上的預測誤差來對當前做預測的模型。其中 ARIMA(p, d, q)稱為差分自回歸移動平均模型， p 為自回歸項，d 為時間序列成為平穩時所作的差分次數，q 為移動平均項數。

ARIMA 模型的設計想法是：將預測對象隨時間推移而行的的數據序列視為一個隨機序列，使用一定的數學模型來表達這個序列。一旦可以成功辨識後，就可以使用時間序列的過去值和現在值來預測未來值。

Preprocessing 資料前處理

Data Analysis 資料分析

從 Kaggle 上的 G-Research Crypto Forecasting 的資料中下載”train.csv”檔案 (https://www.kaggle.com/c/g-research-crypto-forecasting)，裡面包含了各個幣種的相關資訊。首先，我們先來觀察一下資料：

	timestamp	Asset_ID	Count	Open	High	Low	Close	Volume	VWAP	Target
0	1514764860	2	40.0	2376.580000	2399.5000	2357.1400	2374.590000	19.233005	2373.116392	-0.004218
1	1514764860	0	5.0	8.530000	8.5300	8.5300	8.530000	78.380000	8.530000	-0.014399
2	1514764860	1	229.0	13835.194000	14013.8000	13666.1100	13850.176000	31.550062	13827.062093	-0.014643
3	1514764860	5	32.0	7.659600	7.6596	7.6567	7.657600	6626.713370	7.657713	-0.013922
4	1514764860	7	5.0	25.920000	25.9200	25.8740	25.877000	121.087310	25.891363	-0.008264
5	1514764860	6	173.0	738.302500	746.0000	732.5100	738.507500	335.987856	738.839291	-0.004809
6	1514764860	9	167.0	225.330000	227.7800	222.9800	225.206667	411.896642	225.197944	-0.009791
7	1514764860	11	7.0	329.090000	329.8800	329.0900	329.460000	6.635710	329.454118	NaN
8	1514764920	2	53.0	2374.553333	2400.9000	2354.2000	2372.286667	24.050259	2371.434498	-0.004079
9	1514764920	0	7.0	8.530000	8.5300	8.5145	8.514500	71.390000	8.520215	-0.015875

圖 4.1 ”train.csv”的部分資料

由於訓練集的資料中，所有的幣種的資料放在一起，而每個幣種的平均價值不同，我們可以很直覺的判斷，直接將這些幣種放在一起訓練，肯定會得不到好的預測結果。因此我們先將各個幣種的資料分開，得到下表描述的資料集

Asset_ID	Asset_Name	Asset_Shape
0	**Binance Coin (BNB)**	(1942619, 10)
1	**Bitcoin (BTC)**	(1956282, 10)
2	Bitcoin Cash	(1953537, 10)
3	**Cardano(ADA)**	(1791867, 10)
4	Dogecoin	(1156866, 10)
5	EOS.IO	(1955140, 10)
6	**Ethereum (ETH)**	(1956200, 10)
7	Ethereum Classic	(1951127, 10)
8	IOTA	(1592071, 10)
9	Litecoin	(1956030, 10)
10	Maker	(670497, 10)
11	Monero	(1701261, 10)
12	Stellar	(1778749, 10)
13	TRON	(1942619, 10)

表 4.1 各幣種對應的 ID 與資料筆數

我們使用 panadas 套件去讀取"train.csv"，並且根據 Asset_ID 提取 BTC, ETH, BNB, ADA 四種加密貨幣資訊，作為後續的使用。

```
BNB = train[train['Asset_ID'] == 0]
BTC = train[train['Asset_ID'] == 1]
ADA = train[train['Asset_ID'] == 3]
ETH = train[train['Asset_ID'] == 6]

BNB.to_csv(f'ID_0_BNB.csv', index=None)
BTC.to_csv(f'ID_1_BTC.csv', index=None)
ADA.to_csv(f'ID_3_ADA.csv', index=None)
ETH.to_csv(f'ID_6_ETH.csv', index=None)
```

圖 4.2 根據 Asset_ID 提取對應加密貨幣資訊

Dataset Split 劃分資料集

如先前題目更動中所敘述的，我們將原本的訓練資料，依照 Train 70%, Valid 15%, Test 15%的比例進行分割，其中 Train 用於模型訓練，Valid 用於選擇最佳驗證模型，Test 用於評估模型效能。

```
# 70% train, 15% valid, 15% test
point_1 = int(BTC.shape[0] * 0.7)
point_2 = int(BTC.shape[0] * 0.85)

train = BTC_Close[:point_1 + time_stamp]
valid = BTC_Close[point_1 - time_stamp:point_2 + time_stamp]
test = BTC_Close[point_2 - time_stamp:]
```

圖 4.3 分割資料集的程式碼

MinMaxScaler 最大最小標準化

資料的標準化可以使模型有以下兩個優勢，一是提升模型的收斂速度，將資料標準化，能減少收斂時間；二是提高模型的精準度，可讓每個特徵值對結果做出相近程度的貢獻。

在本次報告中我們所採用的標準化方式是最小值最大值正規化，將資料等比例縮放到 [0, 1] 區間中，由公式(圖 4.4)的原理進行轉換。

$$X_{nom} = \frac{X - X_{min}}{X_{max} - X_{min}} \in [0,1]$$

圖 4.4 MinMaxScaler 的計算方式

```
# Zoom to between [0, 1].
scaler = MinMaxScaler(feature_range=(0, 1))
# train data
train_data = scaler.fit_transform(train)
# valid data
valid_data = scaler.fit_transform(valid)
# test data
test_data = scaler.fit_transform(test)
```

圖 4.5 標準化資料(MinMaxScaler)的程式碼

Model 模型架構

PRIMA

使用 PRIMA.auto_arima()對 BTC 搜尋最佳的(p, d, q)設定，如圖 5.1 所示。BTC 中最佳的(p, d, q)=(4, 1, 0)。但將最佳(p, d, q)代入模型中後，對於 PRIMA 的預測效率而言，由於 BTC 的資料筆數過多，因此會花費大量的時間在預測上面(圖 5.2)，經過評估後，決定本次報告中不使用 PRIMA 模型。

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=14438761.813, Time=75.39 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=14443974.688, Time=16.33 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=14440068.307, Time=20.42 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=14439863.461, Time=48.11 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=14443974.520, Time=8.13 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=14438885.450, Time=70.74 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=14438869.939, Time=88.94 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=14438739.414, Time=86.54 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=14438818.031, Time=91.21 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=14438742.224, Time=179.69 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=14438744.046, Time=416.95 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=14438741.215, Time=120.97 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=14438729.623, Time=180.34 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AIC=14438728.083, Time=114.06 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=14438820.447, Time=43.56 sec
 ARIMA(5,1,0)(0,0,0)[0] intercept   : AIC=14438729.088, Time=134.54 sec
 ARIMA(5,1,1)(0,0,0)[0] intercept   : AIC=14438730.935, Time=217.61 sec
 ARIMA(4,1,0)(0,0,0)[0]             : AIC=14438727.834, Time=40.39 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=14438820.225, Time=16.28 sec
 ARIMA(5,1,0)(0,0,0)[0]             : AIC=14438728.841, Time=51.30 sec
 ARIMA(4,1,1)(0,0,0)[0]             : AIC=14438729.374, Time=71.01 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=14438817.874, Time=61.66 sec
 ARIMA(5,1,1)(0,0,0)[0]             : AIC=14438730.688, Time=83.18 sec

Best model:  ARIMA(4,1,0)(0,0,0)[0]
Total fit time: 2237.406 seconds
```

圖 5.1 使用 PRIMA.auto_arima()搜尋 BTC 中的最佳(p, d, q)

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)]
  0%|          | 2/293443 [01:56<4761:43:43, 58.42s/it]
```

圖 5.2 使用 PRIMA 模型所需的預測時間

RNN, LSTM

我們採用類似移動平均的概念，利用前 m 天的收盤價來預測後 n 天的價格，此處 n>1，表示往後預測 n 天並加總取平均作為當下的預測結果。而在兩個模型中，設置了 4 層網絡，第一層、第二層為 RNN/LSTM 層(維度；64)，第三層為 RNN/LSTM 層(維度；32)，第四層為 Dropout 層(0.01，用於防止過擬合)，第四層

為全連接層(神經元個數為 n，用於預測未來 n 筆資料)，optimizer 使用 Adam，並且 lr_rate = 0.01, epoch = 50, batch_size = 1024。

此外，我們也設定了兩個回呼函數幫助我們可以取得更好的模型，一是只儲存最佳 val_loss 的模型作為使用，此設定可以幫助我們使用最佳 val_loss 的訓練模型；二則是設定動態學習率，若當在訓練過程中模型經過三個回合都沒有取得更好的 val_loss，則就會將學習率縮小 0.1 倍，最小值為 1e-7。

```
# create model
model = Sequential()
model.add(SimpleRNN(units=64, return_sequences=True, input_shape=(x_train.shape[1:])))
model.add(SimpleRNN(units=64, return_sequences=True))
model.add(SimpleRNN(units=32))
model.add(Dropout(0.1))
model.add(Dense(output))

model.compile(optimizer=Adam(learning_rate=0.01), loss='mae', metrics=['acc'])
```

圖 5.3 RNN 模型程式碼

```
# create model
model = Sequential()
model.add(LSTM(64, return_sequences=True, input_shape=(x_train.shape[1:])))
model.add(LSTM(64, return_sequences=True))
model.add(LSTM(32))
model.add(Dropout(0.1))
model.add(Dense(output))

model.compile(optimizer=Adam(learning_rate=0.01), loss='mae', metrics=['acc'])
```

圖 5.4 LSTM 模型程式碼

```
callBack = [ModelCheckpoint(f'./model/BTC_LSTM(Input={time_stamp}, Output={output}).h5',
                            monitor='val_loss', verbose=0, save_best_only=True),
            ReduceLROnPlateau(monitor='val_loss', patience=3, factor=0.1, min_lr=1e-7)]
```

圖 5.5 LSTM 中的回呼函數設定

Experiment & Discussion 實驗討論

Comparison of RNN, LSTM

我們使用了相同資料(BTC)、相同架構但是分別為 RNN 跟 LSTM 的模型做為訓練，將 m 設定為 15，n 設定為 1，並且經過 50 論訓練，最終結果如表 6.1 所示。可以看到不管是在每回合平均的訓練時間，MAE, RMSE 上，RNN 的表現結果比起 LSTM 而言都非常的差。那也正如同我們之前所述的，RNN 那樣只能有一種記憶疊加方式。對很多需要“長期記憶”的任務來說，較為適合使用 LSTM。

並且觀察訓練中的 Loss 我們可以看到使用 RNN 的 Loss 大概落在 0.01-0.02 之間，而使用 LSTM 的 Loss 則是落在 0.002-0.008 之間。

最後，觀察預測結果跟實際收盤價的差異，如圖 6.3 和圖 6.4 所示。可以看到 RNN 的預測結果對於實際收盤價而言有肉眼可見的差異性，對比起 LSTM 的結果幾乎完全貼合在實際收盤價上，可以再次證明 LSTM 的表現結果是比較好的。

BTC(15/1)	each_epoch time(s)	MAE	RMSE
RNN	152.49	939.009	1154.830
LSTM	22.88	75.676	104.557

表 6.1 不同模型的訓練結果

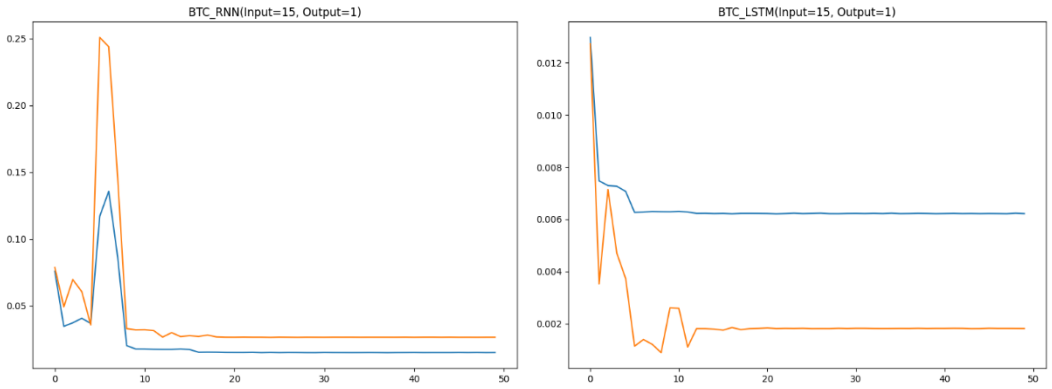

圖 6.1 使用 RNN 的訓練過程的 Loss(左)

圖 6.2 使用 LSTM 的訓練過程的 Loss(右)

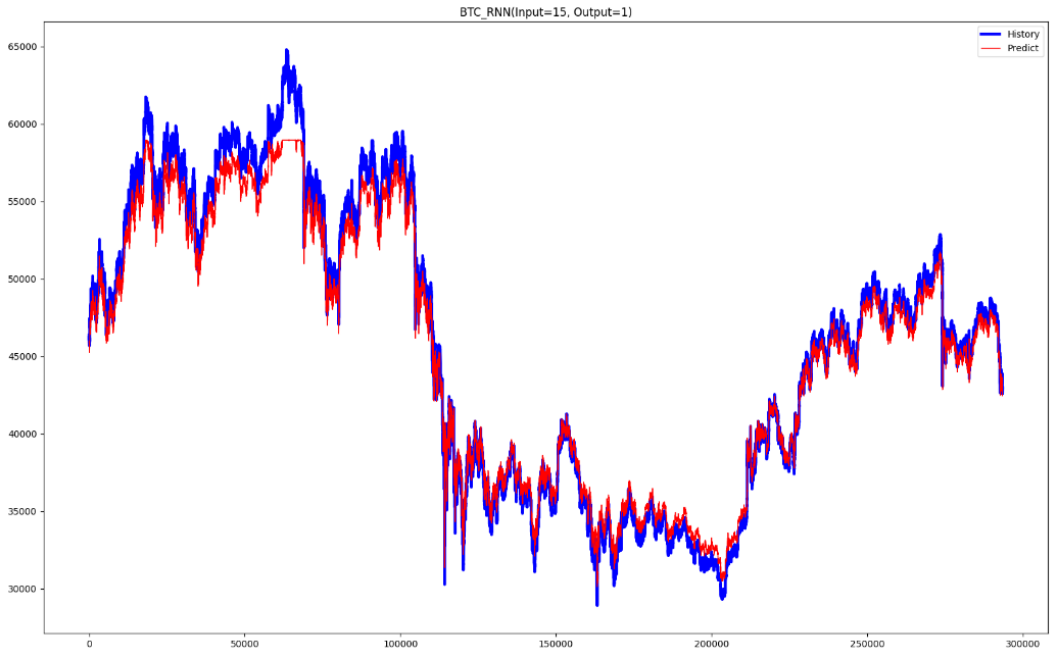

圖 6.3 RNN 預測結果與實際收盤價的比較圖

圖 6.4 RNN 預測結果與實際收盤價的比較圖

LSTM for a single currency

從表 6.2 和表 6.3 中我們可以探討當利用前 m 天的收盤價來預測後 n 天的價格，如果當 m, n 不同的情況下會有甚麼結果。此處的 MAE 和 RMSE 是用於討論預測的收盤價與實際上的收盤價的差異。

在大部分情況下固定 m 變動 n 時，由於向後預測的筆數變多，因此預測的偏差也會變大，進而有表現較差的 MAE 與 RMSE。而若是固定 n 變動 m 時，由於可以使用的資料筆數增加，模型的表現會稍好一點。

另外值得一提的是每一論的訓練時間與輸入資料維度並沒有呈現線性關係。以及由於 BTC 的歷史收盤價都遠大於 ETH 的歷史收盤價，因此 BTC 的 MAE, RMSE 皆都遠大於 ETH。

BTC(m/n)	each_epoch time(s)	MAE	RMSE
15/1	22.88	75.676	104.557
15/5	22.80	95.287	143.041
15/10	22.82	118.789	181.931
30/1	32.15	59.811	86.017
30/5	32.25	91.916	139.361
30/10	32.12	125.373	187.727
60/1	52.19	58.034	84.799
60/5	52.16	92.033	139.963
60/10	52.41	125.067	187.641

表 6.2 使用 BTC 數據做為 LSTM 模型輸入及預測

ETH(m/n)	each_epoch time(s)	MAE	RMSE
15/1	22.56	4.643	7.041
15/5	22.82	7.200.	11.411
15/10	22.91	8.910	14.469
30/1	32.30	4.118	6.541
30/5	32.24	9.062	13.230
30/10	32.32	9.095	14.637
60/1	52.27	6.342	8.894
60/5	52.22	7.202	11.465
60/10	52.27	10.169	15.654

表 6.3 使用 ETH 數據做為 LSTM 模型輸入及預測

Pre-trained model with LSTM

上述的作法，只討論單一幣種使用自身收盤價來訓練，但實際上，不同幣種間的相關性也很大，也許我們同時考慮不同幣種的趨勢，訓練出來的模型效能會更好。

為了實驗這個想法，我們將 BTC, ETH 個別做 MinMaxScaler，再將這些資料合併成更大的訓練集，用此訓練出來的預訓練模型，比較與使用自身收盤價訓練的模型(BNB、ADA)，看看有甚麼差異。結果如表 6.4 和表 6.5 所示。使用 BTC, ETH 的預訓練模型比起只使用自身收盤價的模型結果都好上許多。

BNB(m=15, n=1)	MAE	RMSE
Model: BTC+ETH	0.785	1.197
Model: BNB	0.913	1.368

表 6.4 預訓練模型與 BNB 模型的 MAE, RMSE 比較

ADA(m=15, n=1)	MAE	RMSE
Model: BTC+ETH	0.00347	0.00585
Model: ADA	0.00416	0.00646

表 6.5 預訓練模型與 ADA 模型的 MAE, RMSE 比較

Weight model with LSTM

先前提到的預訓練模型，將每種幣種的趨勢一視同仁，但事實上可能有某種貨幣的影響力大於其他幣種。為了考慮這個因素，我們希望將選定的幣種個別做 MinMaxScaler 後，再加權平均，產生新的自定義虛擬貨幣，並討論看看這樣訓練出來的模型效果如何。

但是由於 Kaggle 上的”train.csv”所提供的資料中，每個幣種的資料的時間並沒有對齊。即使我們想利用特定時間範圍的資料作為模型輸入，但也有中間資料筆數有缺漏的問題。因此我們另外下載了各幣種的歷史數據[4]，其中時間範圍由 2018/01/01 至 2022/01/22。並且我們將 2018/01/01 至 2021/12/31 作為訓練(共 1462 筆)，2022/01/01 至 2022/01/20 作為測試(共 20 筆)。

```
# 加權係數
weight = dict(BTC=6.78, ETH=5.89, BNB=4.31, ADA=4.40)
```

圖 6.5 在 Kaggle 上所提供的各幣種加權係數

如表 6.6 至表 6.9 所示，我們可以觀察到使用加權係數模型的表現並沒有比單一使用 BTC 作為訓練的模型來的好，甚至略遜一籌。那我們認為可能有以下兩個原因，一是比起原先資料集(以分鐘為單位)有上百萬條的訓練資料，新的資料集(以日為單位)只有一千多筆資料，因此沒有辦法使模型有足夠的訓練，進而使表現略遜於單一使用 BTC 作為訓練的模型；二是由於在加密貨幣本身是以 BTC 為主體的市場，因此 BTC 的走勢會帶動其他的幣種的走勢，所以加權係數的模型自然表現會略差一點。

BTC(m=10, n=1)	MAE	RMSE
Model: Weight	43003.666	43008.331
Model: BTC	42880.182	42883.013

表 6.6 加權模型與只使用 BTC 模型 BTC 的 MAE, RMSE 比較

ETH(m=10, n=1)	MAE	RMSE
Model: Weight	3315.563	3316.981
Model: BTC	3299.258	3300.431

表 6.7 加權模型與只使用 BTC 模型 ETH 的 MAE, RMSE 比較

BNB(m=10, n=1)	MAE	RMSE
Model: Weight	468.174	468.271
Model: BTC	466.593	466.718

表 6.8 加權模型與只使用 BTC 模型 BNB 的 MAE, RMSE 比較

ADA(m=10, n=1)	MAE	RMSE
Model: Weight	0.825	0.871
Model: BTC	0.823	0.866

表 6.9 加權模型與只使用 BTC 模型 ADA 的 MAE, RMSE 比較

Conclusion 結論

預測時間序列是一個非常有趣的領域。雖然我們在使用 LSTM 可以取得不錯的預測結果，但是由於加密貨幣市場跟一般金融市場的實際情況十分接近，並不是所有投資者都符合經濟學意義上的理性人假設。仍就會有投資人的不理性操作**(幫我還套在山頂的 EOS 哭哭)**，因此即使我們有不錯的預測結果仍舊應該視為輔助/參考用，並不應該完全相信模型的，自身仍舊需要有更多的思考能力，
Machine learning is just a tool, not everything.

Future Work 未來展望

我們在這次報告中有把 BTC, ETH 的收盤價訓練出來的模型做為預訓練模型，並且收到不錯的成效，這代表著：或許不同幣種間的漲跌是相關的，因此在未來有機會的話，我們希望去探討選擇那些幣種作為預訓練模型，例如選擇 BTC, ETH, BNB, ADA，看看怎樣的選擇方式可以讓模型最好。

並且希望可以取得更為詳細的資料，完成對於加權係數模型的探討，理解說為甚麼在本次報告中的加權係數模型表現並沒有比較好。並且從中得到新的啟發，進而設計出表現更加優秀的模型。

Reference 參考資料

[1] Bitcoin Price Prediction Using Recurrent Neural Networks and LSTM
(https://www.analyticsvidhya.com/blog/2021/05/bitcoin-price-prediction-using-recurrent-neural-networks-and-lstm/)

[2] 利用深度學習和機器學習預測股票市場（附程式碼）
(https://iter01.com/2251.html)

[3] pmdarima 1.8.4 Documention
(https://alkaline-ml.com/pmdarima/about.html)

[4] investing.com
(https://www.investing.com/)