RNN's Application in Cryptocurrency Price Prediction

Group 4 Members:

Liu Zekai	(A0215671H)
Chen Jing	(A0215606L)
Zhang Lin	(A0215610W)
Tang Jiayun	(A0215654E)
Wang Wenyi	(A0215642L)

Background

Cryptocurrency

Background

Cryptocurrencies

Cryptocurrencies are systems that allow for secure payments online which are denominated in terms of virtual "tokens," which are represented by ledger entries internal to the system.

What does "Crypto" means

"Crypto" refers to the various encryption algorithms and cryptographic techniques that safeguard these entries, such as elliptical curve encryption, public-private key pairs, and hashing functions.

Advantages

Make it easier to transfer funds directly between two parties with minimal processing fees

Disadvantages

Well-suited for a host of illegal activities, such as money laundering and tax evasion

Data

BINANCE

Margin Trading for KAVA & SUPER Enabled on Binance 04-01

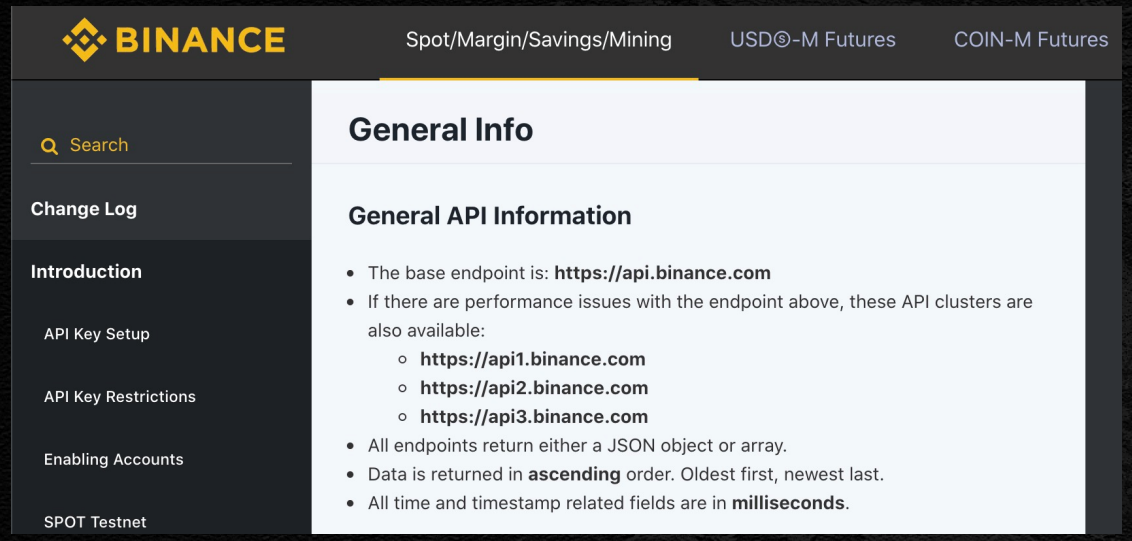
Name	Last Price	24h Change
BNB BNB	$316.80	+6.46%
BTC Bitcoin	$59,008.00	+1.59%
ETH Ethereum	$1,946.84	+7.36%
CHZ Chiliz	$0.498350	-0.95%
DOT Polkadot	$38.07	+11.22%

Data Resource

BNB, BTC, ETH 2018.3.22 0:00:00-2021.3.22 0:00:00 from Binance API

Data

- Binance's API




```python
import requests
import time
import pandas as pd
pd.set_option("expand_frame_repr",False)

names = locals()

BASE_URL = "http://api.binance.com"
currency = ["BNB","BTC","ETH"]

limit = 1000 # maximum enquiry per request
period = 29 # means 29 x 1000 = 29000 hours ; in 3 years we have around 365x3x24=26280 hours
time_now = 1616371200 # 2021/3/22 00:00:00 UTC (2021/3/22 08:00:00 Singapore time)

for j in currency:

    #initialize
    df_result = pd.DataFrame()
    start_time = int(time_now//60*60*1000 -limit*60*60*1000*period)
    end_time = int(start_time+limit*60*60*1000)

    for i in range(period):

        url = BASE_URL+"/api/v3/klines"+"?symbol="+j+"USDT&interval=1h&limit="+str(limit)+"&startTime="+str(start_ti
        resp = requests.get(url)
        data = resp.json()
        df = pd.DataFrame(data,columns={"open_time":0,"open":1,"high":2,"low":3,"close":4,"volume":5,
                                        "close_time":6,"quote_volume":7,"trades":8,
                                         "taker_base_volume":9,"taker_quote_volume":10,"ignore":11})
        df=df[["open_time","open","high","low","close","volume"]]
        df["open_time"] = pd.to_datetime(df["open_time"],unit="ms")

        # store the enquiry result
        df_result = pd.concat([df_result,df])

        # reset start time and end time
        start_time = end_time
        end_time = int(start_time+limit*60*60*1000)

    #store the csv file and dataframe
    # choose the data from 2018-03-22 00:00:00 to 2021-03-22 00:00:00
    names["df_"+j] = df_result[df_result["open_time"]>="2018-03-22 00:00:00"].reset_index(drop=True)
    names["df_"+j].to_csv("result_"+j+".csv")
```

Data

- Pretreatment

```
data = bit_data['close']
close_train=data.iloc[:len(data)-260]
close_test=data.iloc[len(close_train):]
close_train=np.array(close_train)
close_train=close_train.reshape(close_train.shape[0],1)
close_train.shape
```

```
(25981, 1)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
close_scaled=scaler.fit_transform(close_train)
close_scaled.shape
```

```
(25981, 1)
```

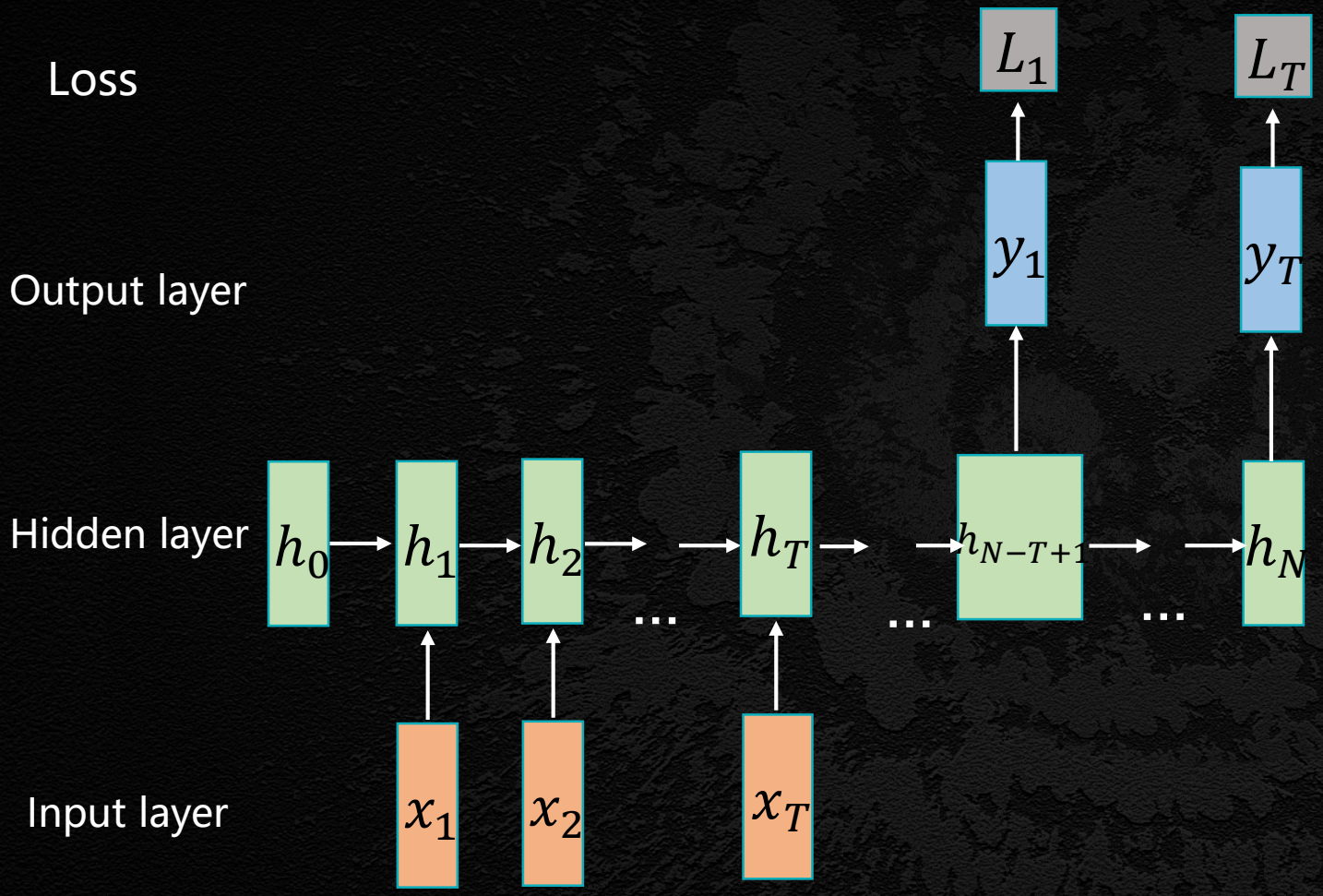
```
timestep=24
x_train=[]
y_train=[]
for i in range(timestep,close_scaled.shape[0]):
    x_train.append(close_scaled[i-timestep:i,0])
    y_train.append(close_scaled[i,0])
x_train,y_train=np.array(x_train),np.array(y_train)
x_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1)
print("x_train shape= ",x_train.shape)
print("y_train shape= ",y_train.shape)
```

```
x_train shape=  (25957, 24, 1)
y_train shape=  (25957,)
```

RNN Model

Cryptocurrency

RNN Introduction

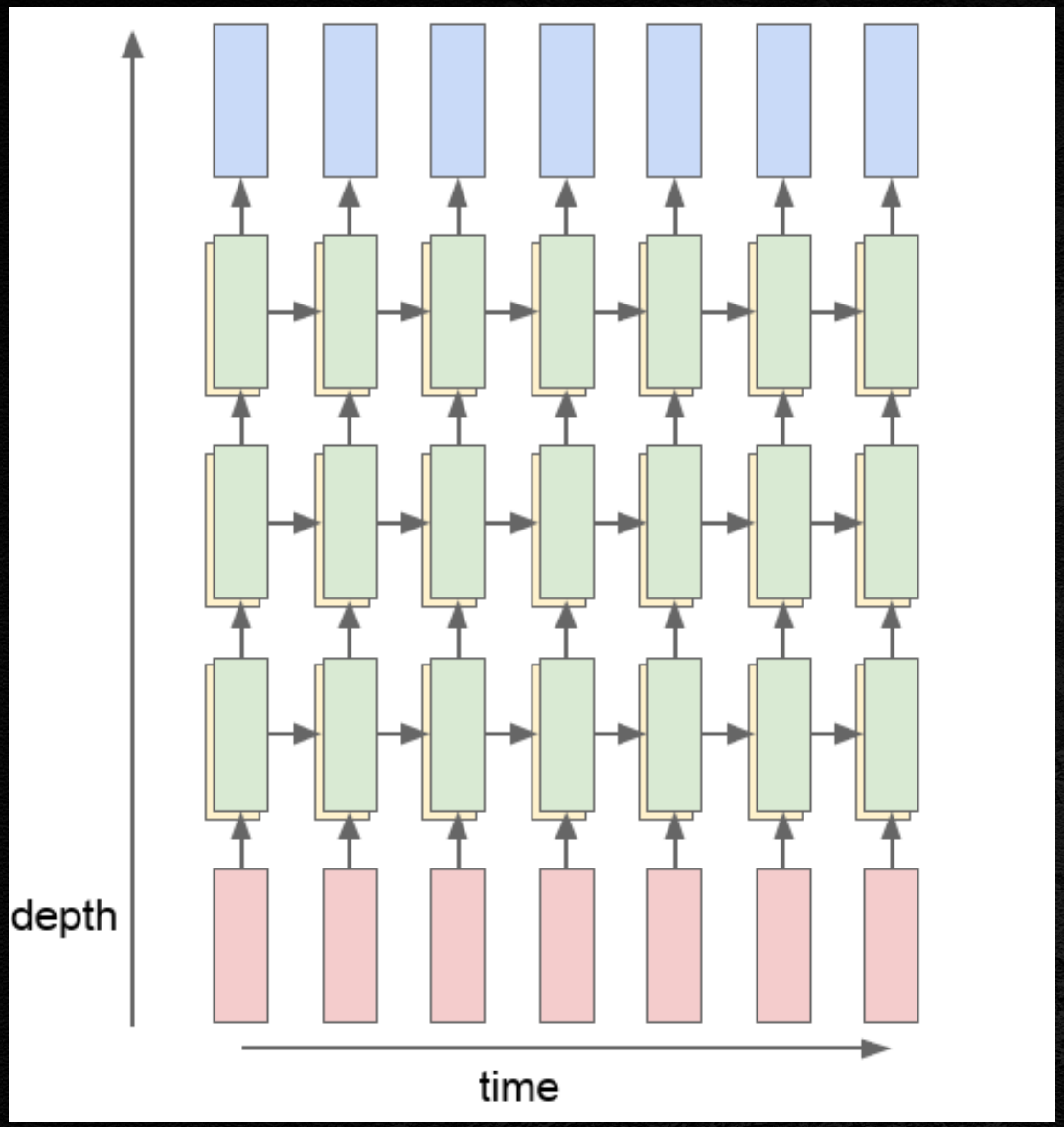


$$h_t = \sigma_h(W_h\, x_t + U_h\, h_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y\, h_{N-T+t} + b_y)$$

- x_t: input vector
- h_t: hidden layer vector
- y_t: output vector
- W, U and b: parameter matrices and vector
- σ_h and σ_y: Activation functions

Multilayer RNNs

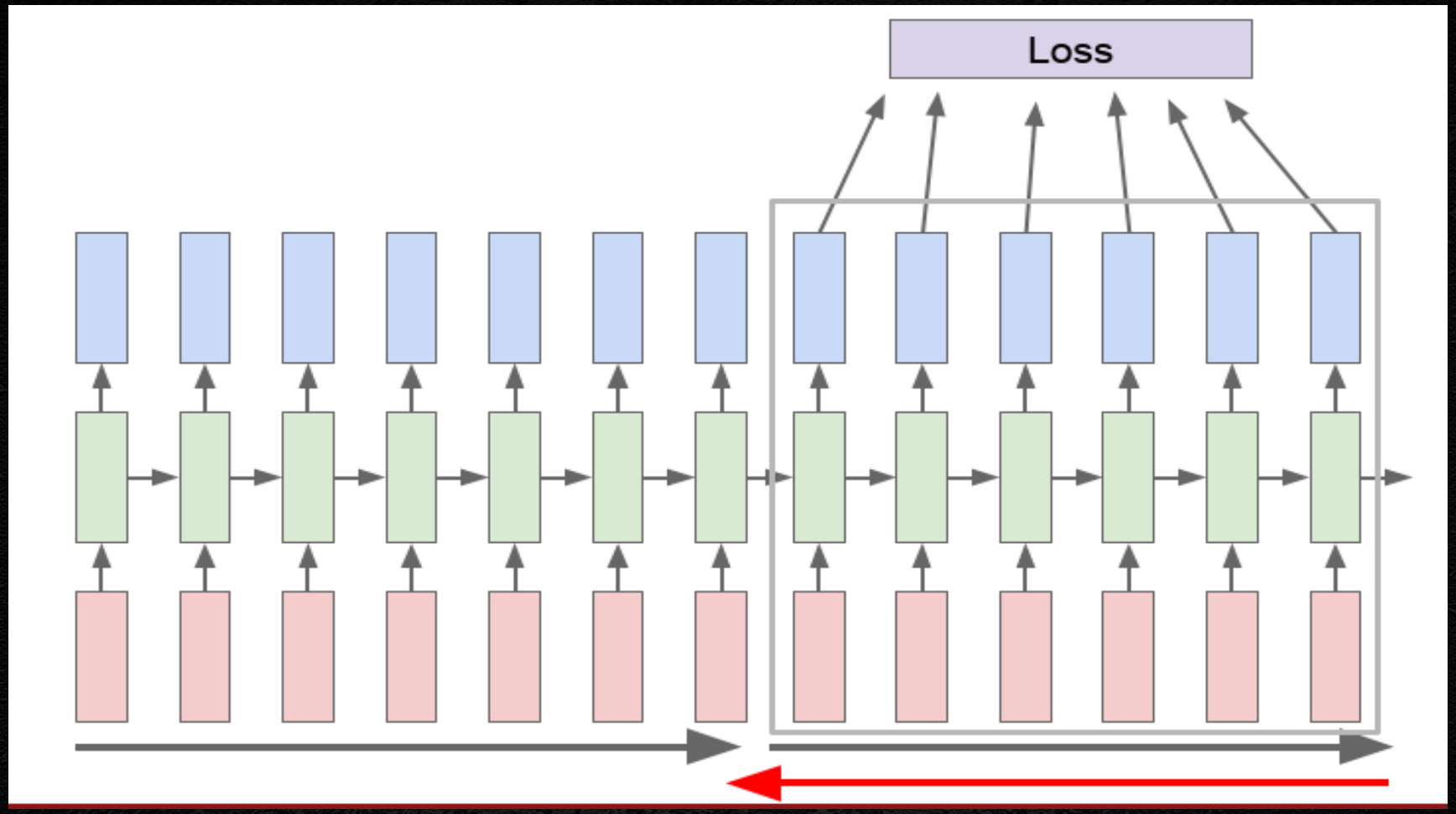



```
regressor=Sequential()

#the first layer
regressor.add(SimpleRNN(128,activation="relu",input_sha
pe=(x_train.shape[1],1),return_sequences=True))
regressor.add(Dropout(0.25))
#not work in 25% situation  (avoid overfit)

#the second layer
regressor.add(SimpleRNN(256,activation="relu",return_se
quences=True))
regressor.add(Dropout(0.25))
…
```

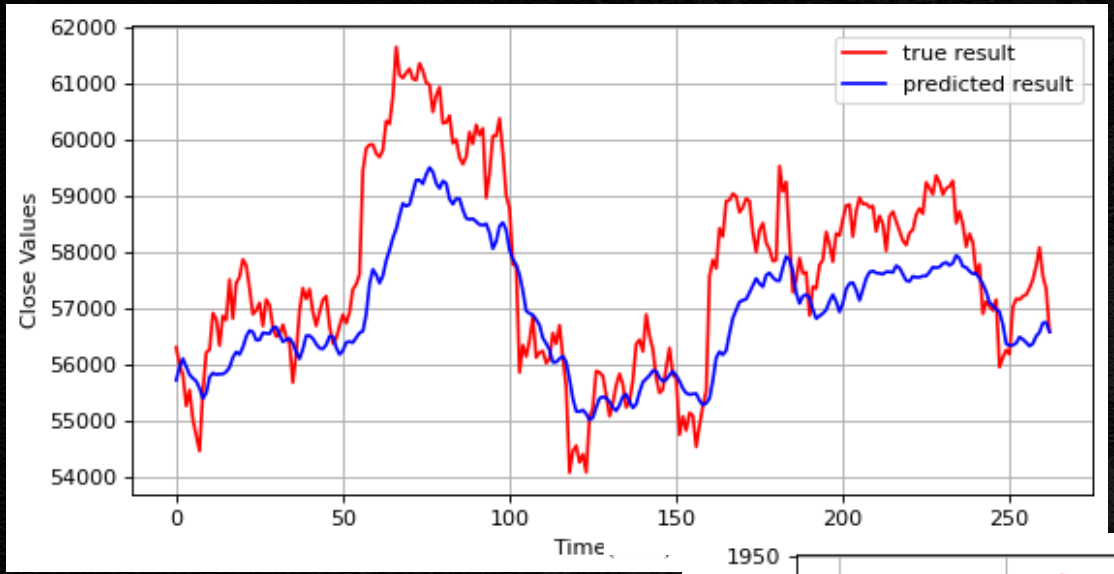
Truncated Backpropagation through Time



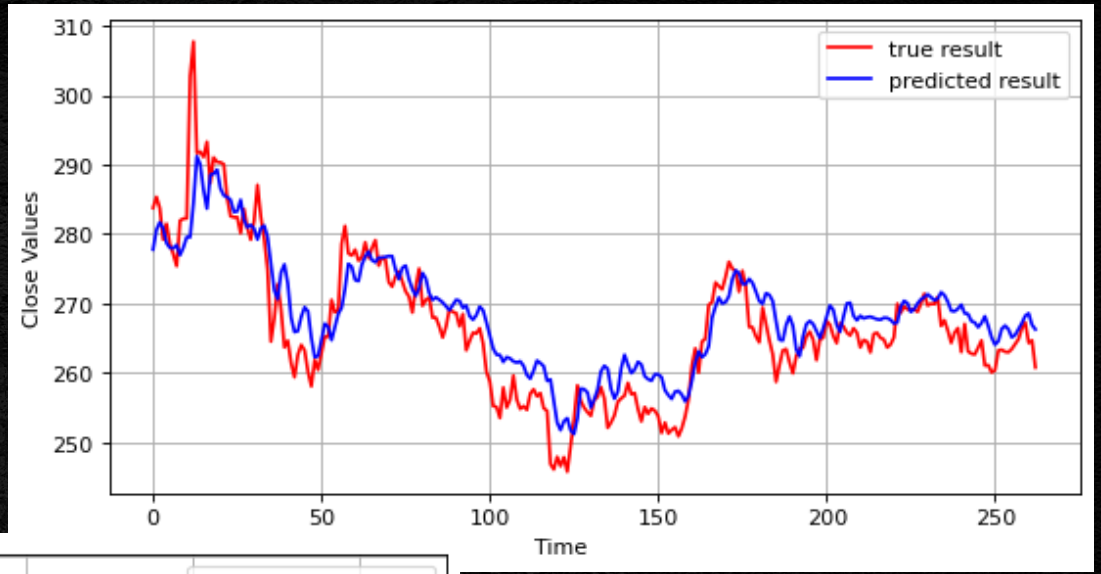
regressor.fit(x_train,y_train,epochs=100,batch_size=64)

Results

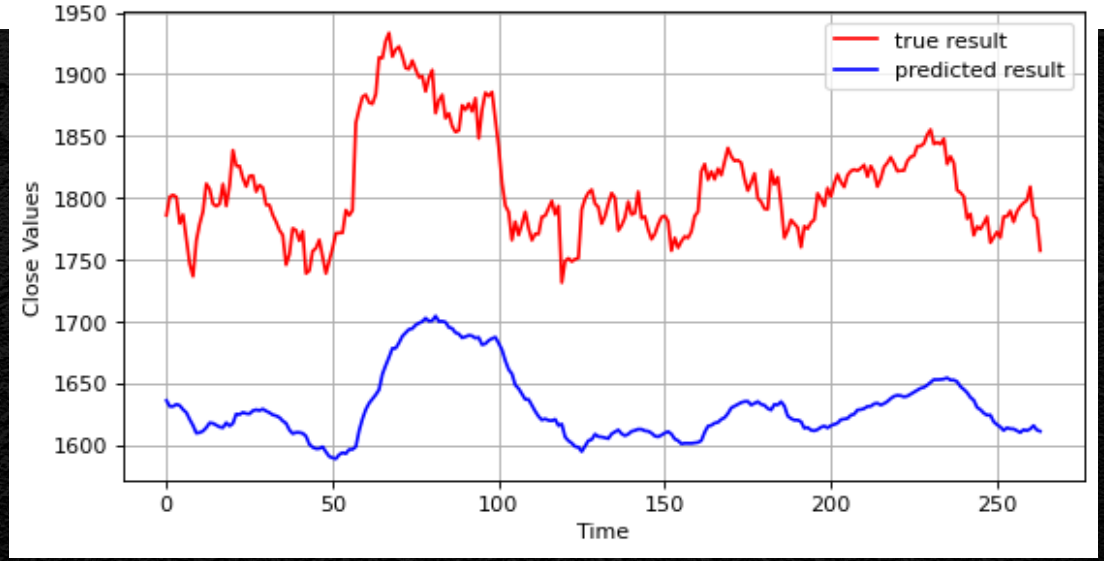
BTC correlation: 0.89



BNB correlation: 0.92



ETH
correlation: 0.76



Gradient Vanish

$$h_t = \sigma_h(W_h\ x_t + U_h\ h_{t-1} + b_h)$$

$$0 < U_h < 1$$

if it is larger than 1 then the model will face with gradient explosion

The model is unable to propagate useful gradient information from the output end of the model back to the layers near the input end of the model.

LSTM Model

Cryptocurrency

LSTM Structure

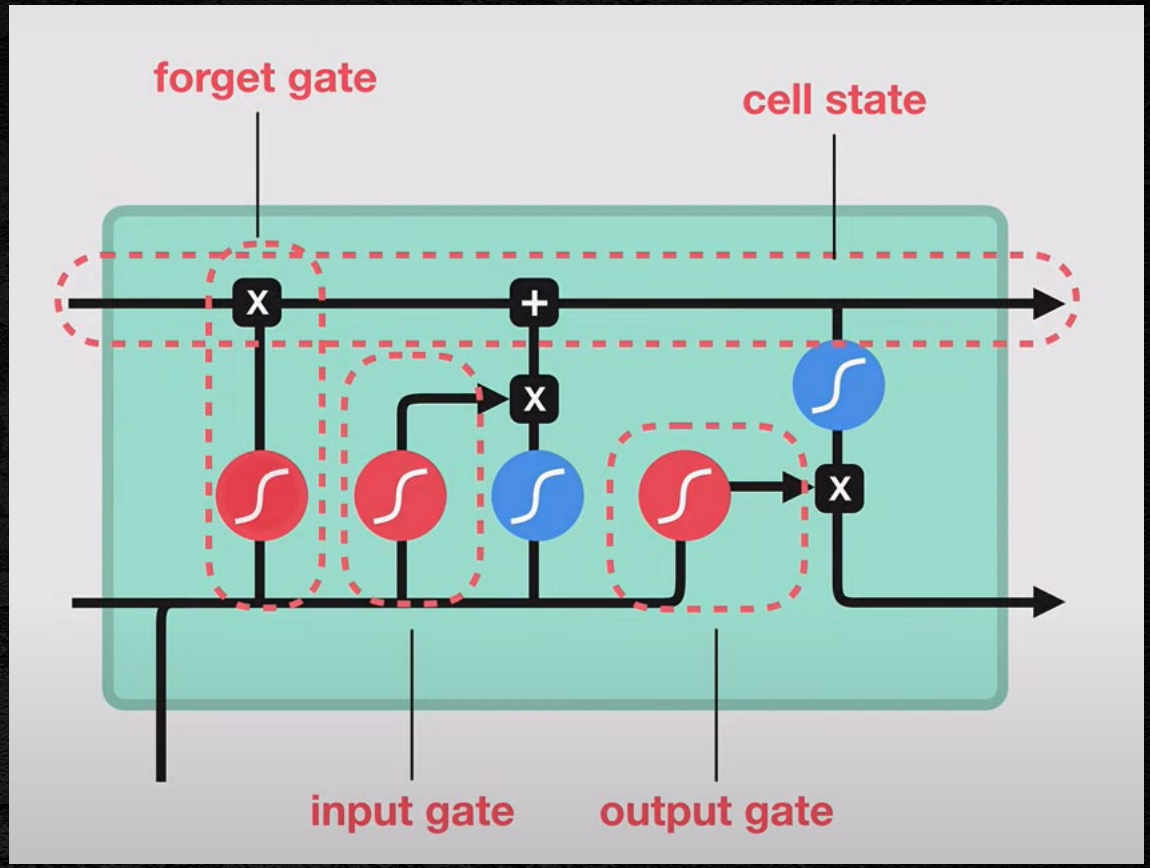
Cell States

Gates

Input Gate

Forget Gate

Output Gate



Model

LSTM:
The number of nodes: 200
Activation: Relu Function

Dense:
Fully Connected layer

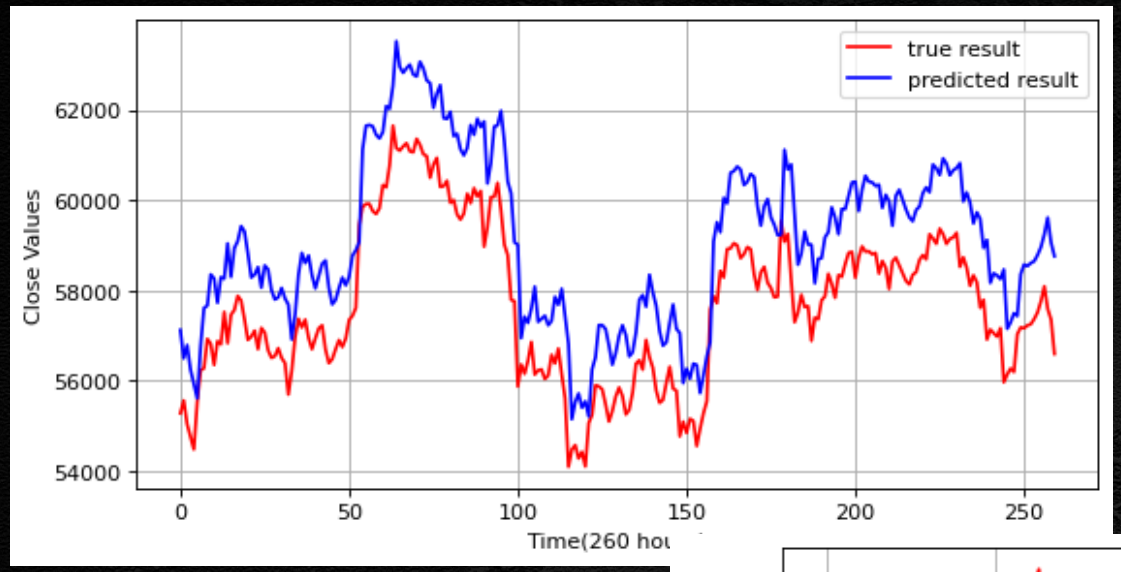
Compile:
Optimizer: Adam
Loss function: MSE

```
1  from sklearn.metrics import mean_absolute_error
2  from tensorflow.keras.layers import LSTM,GRU
3  from tensorflow.keras import Sequential
4  from tensorflow.keras.layers import Dense, SimpleRNN, Dropout,Flatten,Activation
5  from tensorflow.keras import activations
6
7  model=Sequential()
8  model.add(LSTM(200,input_shape=(None,1),activation="relu"))
9  model.add(Dense(1))
10 model.compile(loss="mean_squared_error",optimizer="adam")
11 model.summary()
12 model.fit(x_train,y_train,epochs=30,batch_size=32,validation_split=0.1,shuffle=True)
```

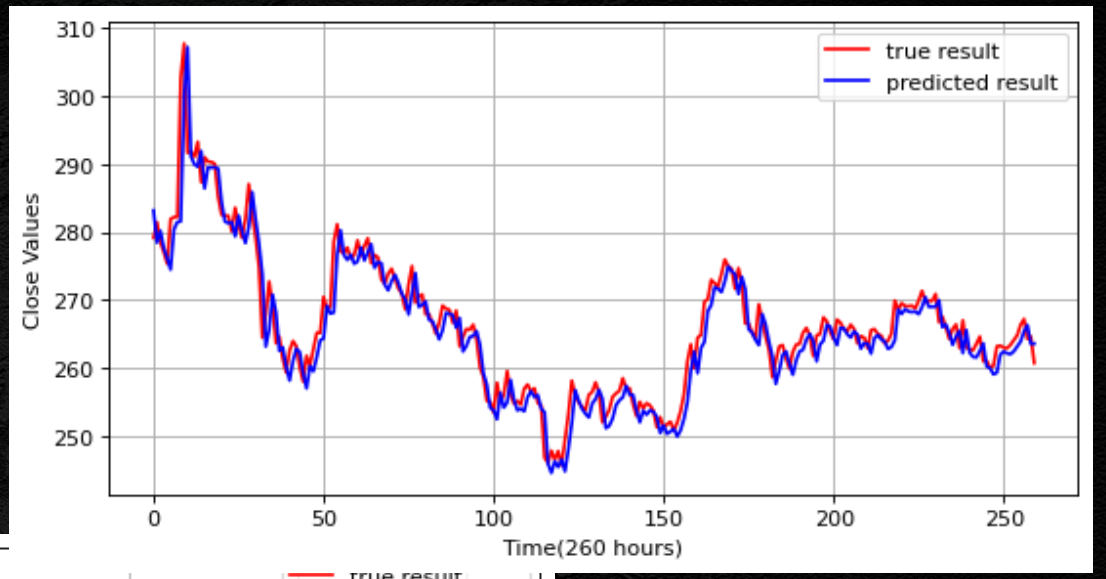
```
WARNING:tensorflow:Layer lstm will not use cuDNN kernel since it doesn't meet the cuDNN kernel
allback when running on GPU
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 200)               161600
_________________________________________________________________
dense (Dense)                (None, 1)                 201
=================================================================
Total params: 161,801
Trainable params: 161,801
Non-trainable params: 0
_________________________________________________________________
Epoch 1/30
```

Results

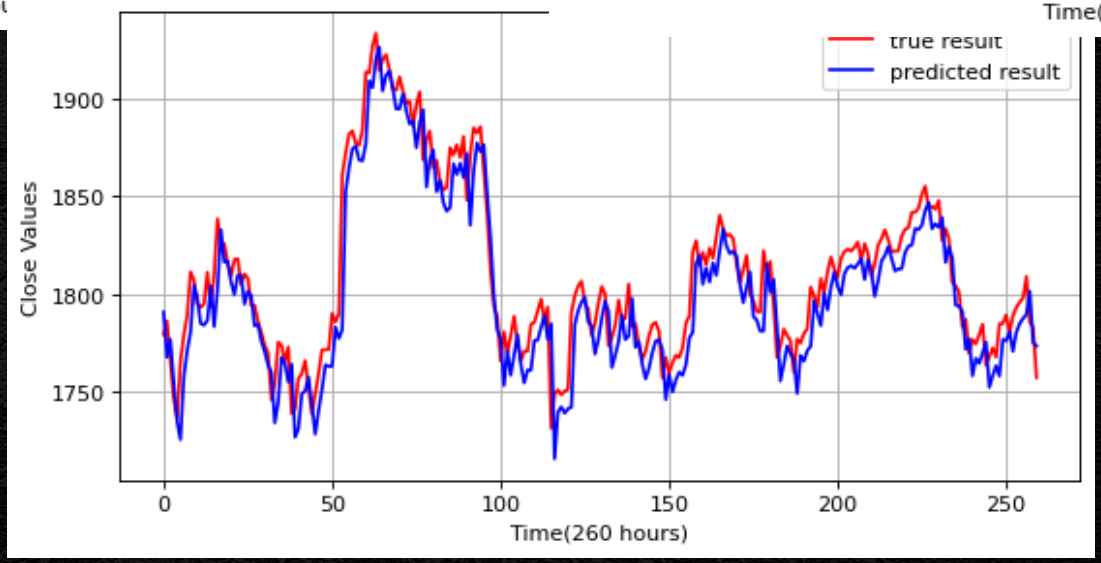
BTC correlation: 0.9636



BNB correlation: 0.9561



ETH
correlation: 0.9511



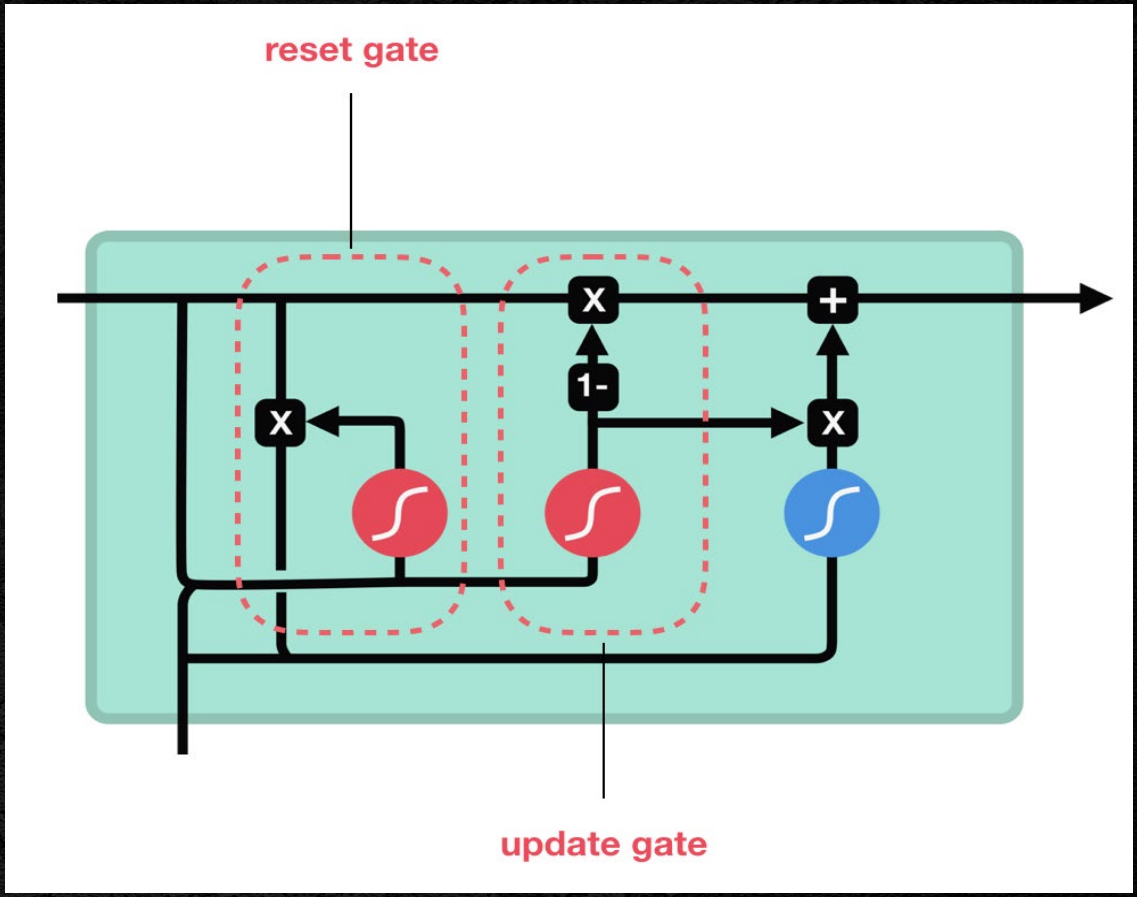
GRU Model

Cryptocurrency

GRU Structure

Reset Gate

Update Gate



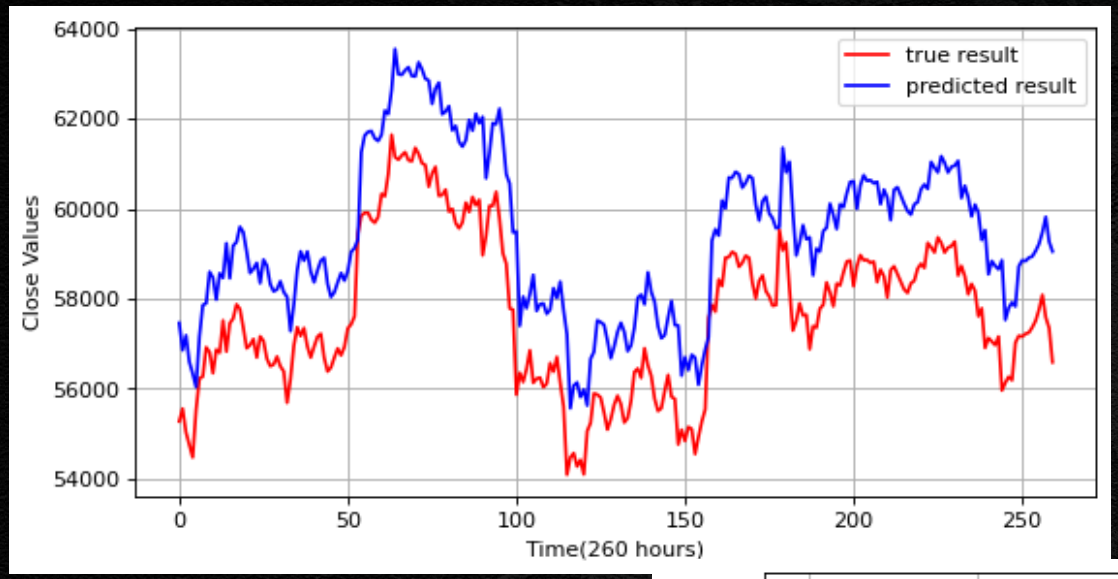
Model

```
model=Sequential()
model.add(GRU(200,input_shape=(None,1),activation="relu"))
model.add(Dense(1))
model.compile(loss="mean_squared_error",optimizer="adam")
model.summary()
model.fit(x_train,y_train,epochs=30,batch_size=32,validation_split=0.1,shuffle=True)
```

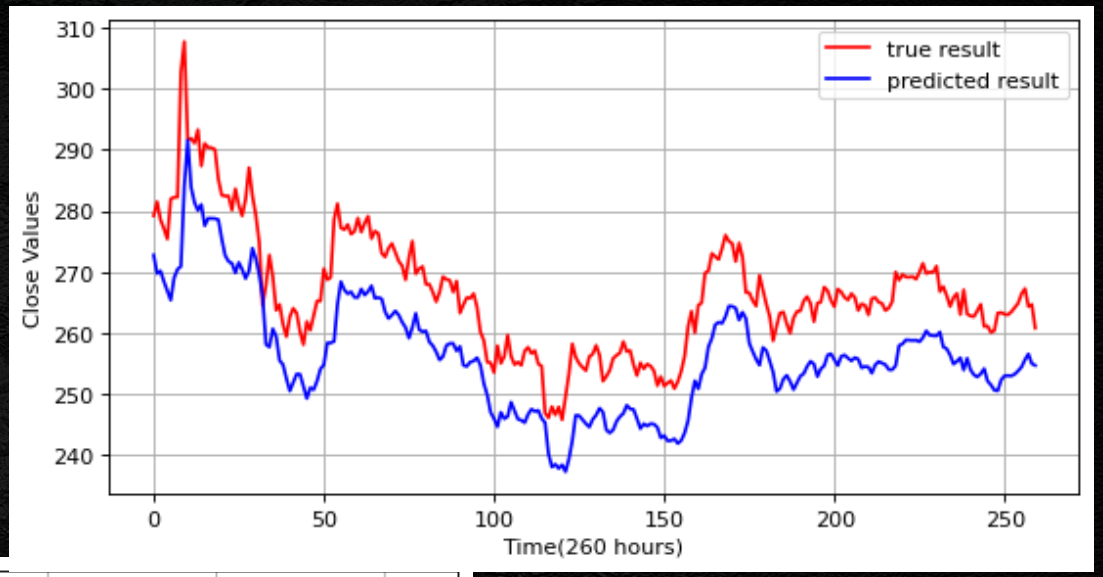
```
Epoch 26/30
731/731 [==============================] - 32s 44ms/step - loss: 3.0256e-06 - val_loss: 7.6953e-05
Epoch 27/30
731/731 [==============================] - 32s 44ms/step - loss: 2.8748e-06 - val_loss: 1.7457e-04
Epoch 28/30
731/731 [==============================] - 33s 45ms/step - loss: 3.1653e-06 - val_loss: 1.3639e-04
Epoch 29/30
731/731 [==============================] - 33s 45ms/step - loss: 3.0601e-06 - val_loss: 8.2174e-05
Epoch 30/30
731/731 [==============================] - 33s 45ms/step - loss: 2.6104e-06 - val_loss: 8.3836e-05
<tensorflow.python.keras.callbacks.History at 0x7f1f99d4e210>
```

Results

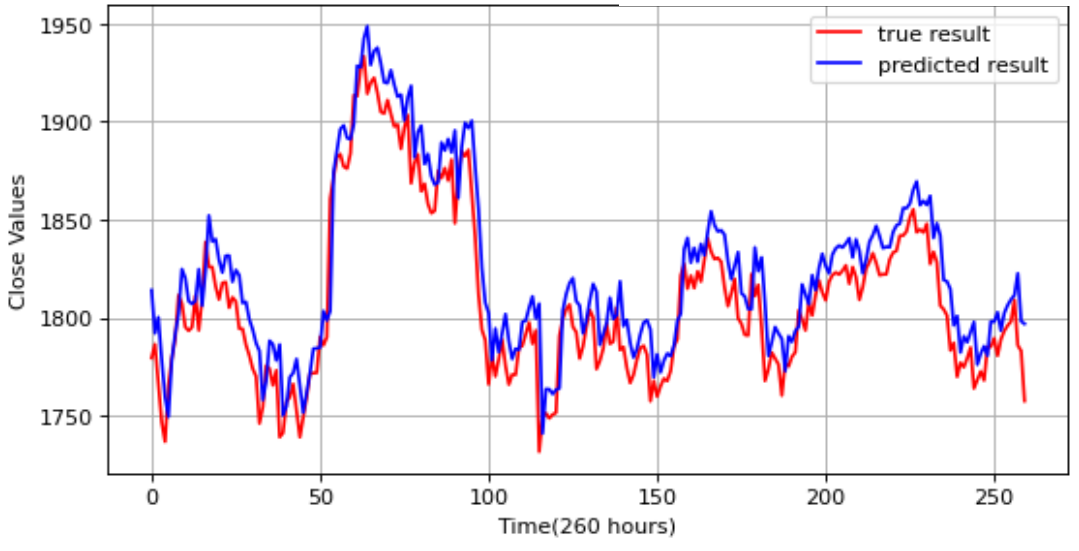
BTC correlation: 0.9633



BNB correlation: 0.9539



ETH
correlation: 0.9512



Conclusion

Cryptocurrency

Conclusion

RNN

- Possibility of processing input of any length
- Model size not increasing with size of input
- Computation takes into account historical information
- Computation is slow
- Hard to access information from a long time ago
- Cannot consider any future input for the current state

LSTM

- An advanced version of RNN to model long-range dependencies more precisely.
- LSTM is more accurate when using datasets with longer sequences compared with GRU.
- LSTM has an advantage in learning some latent features of the sequence that are not directly tied to the elements of the sequence

GRU

- A special variant of the RNN
- GRU is slightly less complex but is approximately as good as an LSTM performance-wise.
- GRU is simpler and thus easier to modify, for example adding new gates in case of additional input to the network. It's just less code in general.

Conclusion

Get more and/or better data: If past prices alone are sufficient to decently forecast future prices, we need to include other features that provide comparable predictive power. That way, the LSTM model wouldn't be so reliant on past prices, potentially unlocking more complex behaviors.

Change Loss Function: MAE doesn't really encourage risk taking. For example, under mean squared error (MSE), the LSTM model would be forced to place more importance on detecting spikes/troughs. More bespoke trading focused loss functions could also move the model towards less conservative behaviors.

Avoid overfit: It is not always good to increase the model accuracy because we may end up having an overfit model.


Thank you