CMPUT466 Final Project Report

Cryptocurrency Prediction by Linear Regression, CNN and RNN

Name: Jiale Hu

ID: 1505183

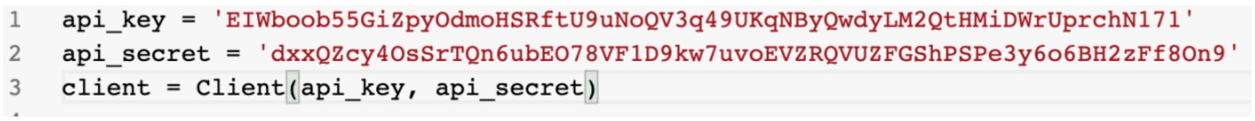
Date: Dec 20, 2020

Introduction

Cryptocurrency has been very popular in recent years. It is defined as a digital asset designed to work as a medium of exchange wherein individual coin ownership records are stored in a ledger existing in a form of computerized database using strong cryptography to secure transaction records, to control the creation of additional coins, and to verify the transfer of coin ownership. Bitcoin, one of the most popular cryptos, which was created more than ten years ago. The price of it fluctuated unexpectedly, where the low of recent price is around 4000 USD and the all-time-high is more than 24k in December 2020. There is a huge risks of trading Bitcoins, but at the same time, a reasonable opportunity of gaining a lot from it within a short time. Therefore, predicting the future price of the bitcoin is worthy some research. In this research project, three machine learning algorithms are applied, namely linear regression, RNN and CNN for comparison.

Formulation

Dataset were provided and fetched from Binance.com with their API. Specifically, a Binance account is needed to get a unique api_key and api_secret as following:



```
api_key = 'EIWboob55GiZpyOdmoHSRftU9uNoQV3q49UKqNByQwdyLM2QtHMiDWrUprchN171'
api_secret = 'dxxQZcy4OsSrTQn6ubEO78VF1D9kw7uvoEVZRQVUZFGShPSPe3y6o6BH2zFf8On9'
client = Client(api_key, api_secret)
```

Then use the credentials and follow the guidance on Binance's APi website in which specific data can be retrieved.

In this project, we are going to predict the price of BTCUSDT, where BTCUSDT means the price of BTC in USDT (A stable crypto in which the value of 1 USDT = 1 USD). The range of date where the dataset was fetched is from 2017-09-01 to 2020-12-01 with a time interval of 30 minutes.

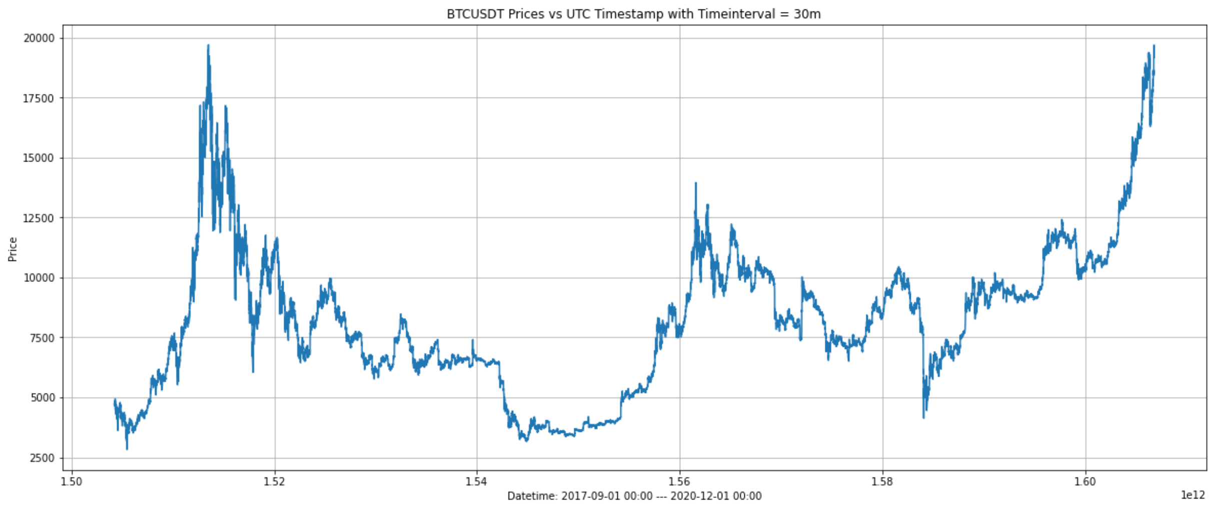


Figure 1:Recent Bitcoin Price in USD

The names of the data column (as known as Klines) are:

```
1.        # Open time
2.        # Open price
3.        # High price
4.        # Low price
5.        # Close price
6.        # Volume
7.        # Quote asset volume
8.        # Number of trades
9.        # Taker buy base asset volume
10.       # Taker buy quote asset volume
11.       # Ignore
12.       # Close time
```

Close price are chosen to be ground truth labels or outputs. For linear regression and CNN, ***#3 to #12*** *are chosen* to be input features as well as the percentage change of ***Close price*** (The percentage change of the open price compared to the last one)**.** Since RNN is dealing with time series data, a sequence of target itself (***Close price***) will be made as inputs, and the specific splitting methods will be introduced in the next section. The dataset is in the datetime order with a total number of 56744 samples, where the training data is from the first 65% portion of the dataset, followed by 20% validation dataset and the rest are for test dataset.

Here is the summary for data splitting:

Dataset: (56744, 13)	
Train: (36884, 12)	~65%
Valid: (11349, 12)	~20%
Test: (8511, 12)	~15%

Approaches

The baseline for all three maching learning algorithm is that the mean square error of predicted result cannot be larger than the mean square error of the mean value of targets and the ground truth label, and each predicted value cannot be negative since there is no way that the price of a Bitcoin is negative. If the approximation of a given data doesn't touch the baseline, then it is a reasonable approximation.

<u>RNN</u>:

The dataset for training a RNN is different from others. 6 consecutive **target**s are regarded as inputs to predict the 7th target. For example, if the target dataset is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], with an interval of 6, 5 datasets will be genertated:

1st: x: [1, 2, 3, 4, 5, 6], y : [7]
2nd: x: [2, 3, 4, 5, 6, 7], y : [8]
3rd: x: [3, 4, 5, 6, 7, 8], y : [9]

4th: x: [4, 5, 6, 7, 8, 9], y : [10]
5th: x: [5, 6, 7, 8, 9, 10], y : [11]

The Hyperparameter of RNN are listed below:
Num_features = 6
Dropout = 0.2
num_epochs = 100
batch_size = 128

Criterion:
Loss criterion: MSE
Model based on: Best validation loss

The tunning method used for RNN is to loop over a list of reasonable Dropouts, and then find the best Dropout with respect to the best test result (main square error).

RNN Model structure:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm (LSTM)                  (None, 6, 50)             10400     
_________________________________________________________________
dropout (Dropout)            (None, 6, 50)             0         
_________________________________________________________________
lstm_1 (LSTM)                (None, 6, 50)             20200     
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 50)             0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 6, 50)             20200     
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 50)             0         
_________________________________________________________________
lstm_3 (LSTM)                (None, 50)                20200     
_________________________________________________________________
dropout_3 (Dropout)          (None, 50)                0         
_________________________________________________________________
dense (Dense)                (None, 1)                 51        
=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

CNN:

The Hyperparameter of CNN are listed below:
num_epochs = 50
batch_size = 128

Criterion:
optimizer = adam
loss = MSE
Model based on: Best validation loss

The tunning method used for CNN is to find the best optimizer between adam and SGD with respect to the best test result (main square error).

CNN Model structure:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 11, 64)            192
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 5, 64)             0
_________________________________________________________________
flatten (Flatten)            (None, 320)               0
_________________________________________________________________
dense (Dense)                (None, 50)                16050
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 16,293
Trainable params: 16,293
Non-trainable params: 0
_________________________________________________________________
```

<u>Linear Regression</u>:

The Hyperparameter of linear regression are listed below:
alpha   = 0.001   # learning rate
batch_size   = 10   # batch size
MaxIter = 100   # Maximum iteration
decay_list = 0.01

Criterion:
loss = MSE
Model based on: Best validation loss

The tunning method used for linear regression is to loop over a list of reasonable learning rate and a list of weight decay and find the best ones with respect to the best test result (main square error).

Results

RNN:
Best Validation loss: 1.1061e-04
Test loss: 3.542795884614927e-05

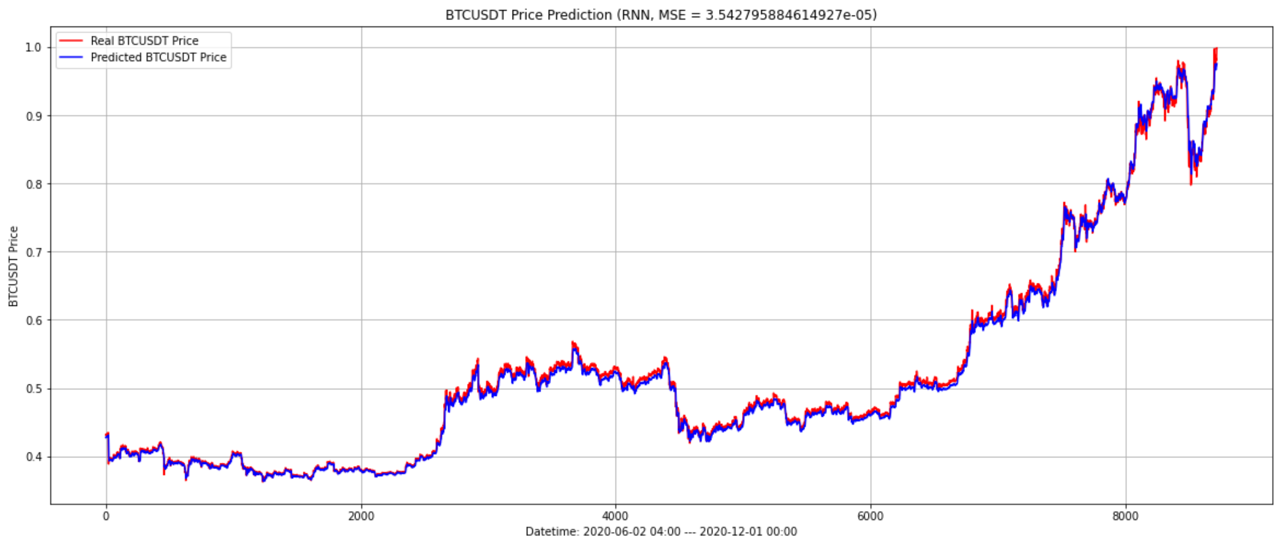


Figure 2: RNN Test Result

CNN:
Best Validation loss: 7.36982299e-05
Test loss: 4.032468515151091e-04

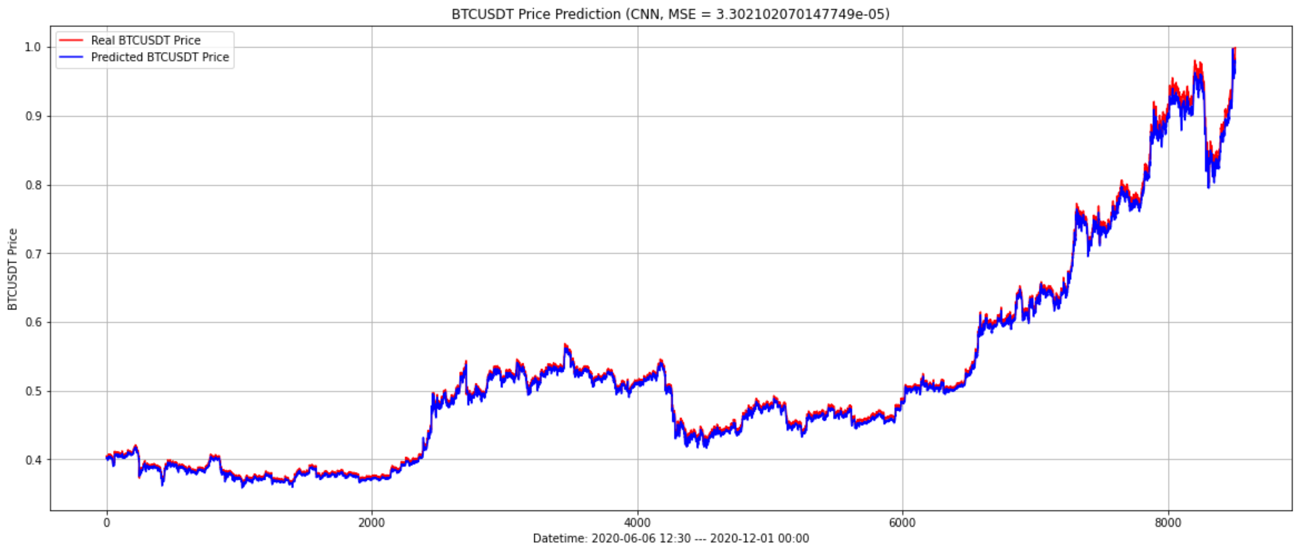


Figure 3: Linear Regression Test Result

Linear Regression:
Best Validation loss: 7.36415916e-05
Test loss: 4.0341044967361783e-04

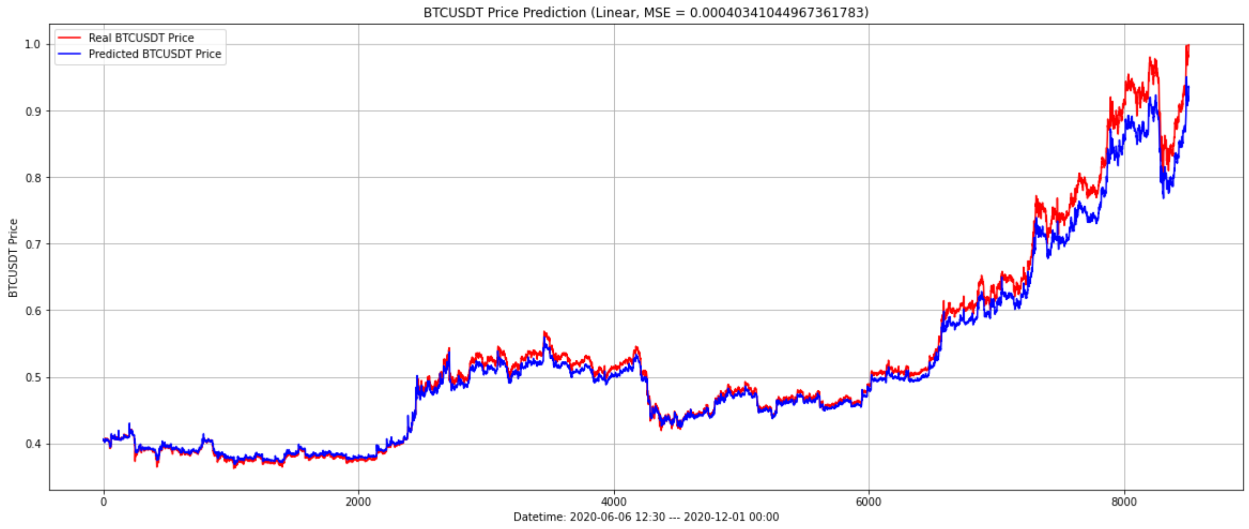


Figure 4: Linear Regression Test Result

Conclusion

The mean square error yield from RNN, CNN, linear regression is 3.542795884614927e-05, 4.032468515151091e-04 and 4.0341044967361783e-04, respectively. It can be seen from the result that CNN performed better in term of the Bitcoin price prediction. As it can be observed from three graphs that the predicted price trend line is very close to the real price trend line, and no negative predicted price is observed, therefore the result of each approach does not touch the baseline. Thus, we can conclude that that they are all good approximations.

Reference:
https://en.wikipedia.org/wiki/Cryptocurrency

https://binance-docs.github.io/apidocs/spot/en/#change-log

https://machinelearningmastery.com/how-to-get-started-with-deep-learning-for-time-series-forecasting-7-day-mini-course/