

**Hong Kong University of Science and Technology**  
**Computer Science & Engineering Department**

**MSBD 5013: CryptoPrediction**

<b>Name</b>	<b>Student ID</b>
LI Ga Lun	20751825
CHEUNG Him Wing Shirley	20660026
USMANI Muhammad Adam	20729858

## Abstract

Two-and-a-half-year training dataset of cryptocurrency exchange rates with USD (i.e., “BTCUSDT”, “ETHUSDT”, “LTCUSDT” and “XRPUSDT”) from May 2018 to Sep 2020 are used in this project. The forecast model can predict 10 minutes later “target”. It was trained by using time series analysis over the period of the dataset. Although the “target” value was calculated by only the “close” value, in our model, we considered all the features in the dataset to construct the forecast model.

## Introduction

From ancient stone currency to cryptocurrencies, from Yapese to distributed ledger system, from statistical techniques to predictive modelling and machine learning.

In this project, we were not only limited to use statistical prediction methodologies but also used stock market trend indicators to predict the 10 minutes later “target”. And we applied different time series modelling methods such as Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP) and Vector Autoregressions (VAR) for finding out the best method according to the prediction accuracy.

## Dataset

There are 2 files which contain two sets of data, one is for training and the other one is for testing.

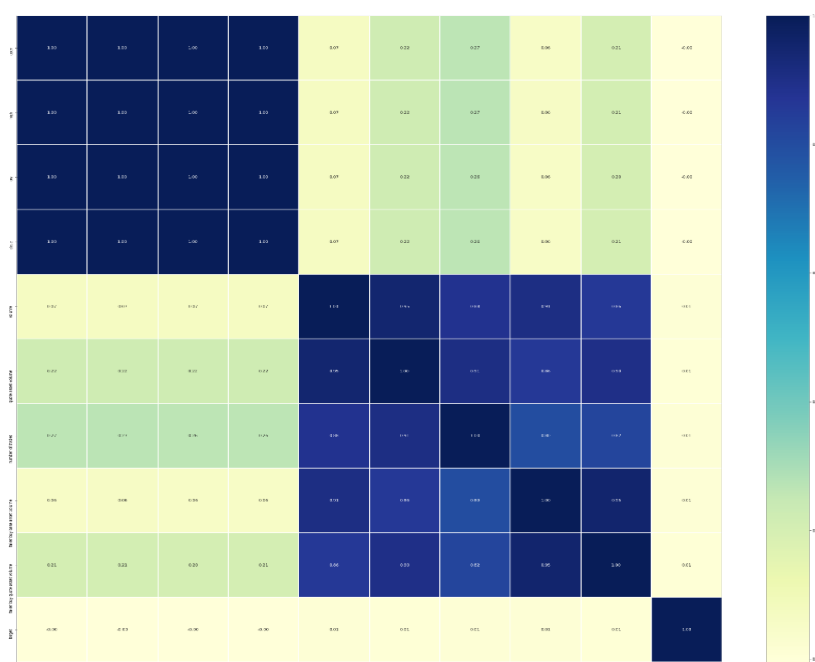
(1) Training Set - training.csv with 13 columns

(2) Testing Set - testing.csv with 12 columns.

id	time	name	Open	High	Low	Close	Volume	Quote asset volume	Number of trades	Taker buy base asset volume	Taker buy quote asset volume	target
id	time	name	Open	High	Low	Close	Volume	Quote asset volume	Number of trades	Taker buy base asset volume	Taker buy quote asset volume	

## Data pre-processing

At the beginning, we changed the “time” column into a datetime index. And then we grouped the data by the “name”. Then we did data cleansing (i.e., kept the last record if the datetime indexes are duplicated and add record if the datetime index is missing.) Finally, we removed “id” and “name” before training since they are not correlated with the “target”. Here is the correlation matrix of “BTCUSDT”. Other correlation matrix can be found in the Jupyter Notebook.



After the data cleansing, we tried to use different kinds of data preparation methods for the model to train up.

**Method 1:** Add five features, they are

1. 'Average buy price', 2. 'Taker sell base asset volume', 3. 'Taker sell quote asset volume', 4. 'Average sell price', and 5. 'TBBAV-TSBAV' - 'Taker sell base asset volume' - 'Taker buy base asset volume'.

Reason to add these five features is the buying and selling information may indicate the movement of the cryptocurrency.

**Method 2:** Add eight features, they are

1. '(Close - Open)', 2. '(High - Open)', 3. '(Low - Open)', 4. '(Close - High)', 5. '(Close - Low)', 6. '(High - Low)', 7. 'V over NOT', - 'Volume' / 'Number of trades' and 8. 'QAV over NOT' - 'Quote asset volume' / 'Number of trades'.

Reason to add these eight features is the massaged market information may increase the stationarity of the cryptocurrency.

**Method 3:** Add three stock indicators, they are

1. 10-min Simple Moving Average (SMA), 2. Relative Strength Index 14 (RSI), and 3. Moving Average Convergence Divergence 12 (MACD).

Reason to add these three stock indicators is intelligence of technical analysis may help in the prediction.

**Method 4:** Add five features and remove eight originally features in the dataset, they are

Add	Remove
1. Sum of 10-min return deviations (SumRet), SumRet = Mean Deviation * N - N = Number of calculation points 2. Return position (RetPos), RetPos = $\ln(1 + \text{'Close'}) - \ln(1 + \text{'Close' 10-min before})$ 3. Imbalanced Volume (ImbVol), ImbVol = 'Taker buy base asset volume' * 2 - 'Volume' 4. Average Volume per Trade (VolTrades), and VolTrades = 'Volume' / 'Number of trades' 5. Signal (Signal). Signal = 'Close' - 'Open'	1. Open, 2. High, 3. Low, 4. Close, 5. Quote asset volume, 6. Number of trades, 7. Taker buy base asset volume, and 8. Taker buy quote asset volume.

Reason to add these five features and remove eight features is market information in a period instead of a single point may be more informative.

**Method 5:** Approximation of Method 4

Use approximation for calculating similar features of method 4 to off load the CPU and RAM in features generation.

## Methodology

In this project, we attempted three different strategies. The first two models that we used were deep learning models whilst the last model was a statistical model.

## Univariate Long Short-Term Memory (LSTM) for Stock Prediction

The first strategy involved using a univariate LSTM model for stock prediction. We used this model because deep learning algorithms have the ability to identify existing patterns in data and are able to exploit them by using a soft learning process. The LSTM architecture is capable of finding short-term and also long-term dependencies in data which can lead to obtaining good predictions. This model works by learning from a series of past observations to predict the next value in the sequence. A drawback to using this model is that it only considers a single feature when making predictions namely the 'target'.

## Multilayer Perceptron (MLP) for Time Series

In the second strategy, we used the Multilayer Perceptron model for time series forecasting. The MLP model is one of the basic architectures of neural networks in that it is made up of an input layer which is a vector of features, hidden layers consisting of N neurons and an output layer which is the output of the network, and which is different dependent on the task.

We considered this model to be a viable choice for time series forecasting for several reasons. Firstly, it has a non-linear activation meaning that the network is able to model complex, non-linear relationships between the features and the target. Secondly, the model has the ability to accept multivariate inputs which is an improvement from the model that we used in the first strategy. However, a drawback to this model is that the input is being received and not being treated as sequenced data. This can be problematic in that the model will not be able to see the data with the sequence pattern that it has.

## Vector Autoregressions (VAR)

Our final strategy involved using the VAR model. This is short for Vector Autoregressions. This model is a multivariate time series model where each variable is a linear function of the past lag values of itself and past lag values of all the other variables.

### How does this model work?

The VAR model can be simply explained using a visual example where we have two variables y1 and y2 and want to forecast the values of these two variables at time t, from the given data for the past n values. The lag value has been set to 1.

Variable y1	Variable y2
$y1_{t-n}$	$y2_{t-n}$
...	...
$y1_{t-2}$	$y2_{t-2}$
$y1_{t-1}$	$y2_{t-1}$
$y1_t$	$y2_t$

We use the following mathematical expressions for calculating  $y1(t)$  and  $y2(t)$ , in these expressions we use the past values of both y1 and y2.

$$y1(t) = a_1 + w_{11} * y1(t-1) + w_{12} * y2(t-1) + e_1(t-1)$$

$$y2(t) = a_2 + w_{21} * y1(t-1) + w_{22} * y2(t-1) + e_2(t-1)$$

$a_1$  and  $a_2$  are constant terms,  $w_{11}$ ,  $w_{12}$ ,  $w_{21}$ ,  $w_{22}$  are the coefficients and  $e_1$  and  $e_2$  are the error terms.

We can rewrite the equations in the following form below:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} y_1(t-1) \\ y_2(t-1) \end{bmatrix} + \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix}$$

We can generalize this example for the VAR(P) process, where we have variables  $y_1, y_2, y_k$ .

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} + \begin{bmatrix} w_{11} & \dots & w_{1k} \\ w_{21} & \dots & w_{2k} \\ \vdots & \ddots & \vdots \\ w_{k1} & \dots & w_{kk} \end{bmatrix} \begin{bmatrix} y_1(t-1) \\ y_2(t-1) \\ \vdots \\ y_k(t-1) \end{bmatrix} + \dots + \begin{bmatrix} w'_{11} & \dots & w'_{1k} \\ w'_{21} & \dots & w'_{2k} \\ \vdots & \ddots & \vdots \\ w'_{p1} & \dots & w'_{pk} \end{bmatrix} \begin{bmatrix} y_1(t-p) \\ y_2(t-p) \\ \vdots \\ y_k(t-p) \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{bmatrix}$$

$K \times 1 \quad K \times 1 \quad K \times K \quad K \times 1 \quad K \times K \quad K \times 1$

This can be rewritten as follows:

$$\begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} a_1 \end{bmatrix} + \begin{bmatrix} w_1 \end{bmatrix} \begin{bmatrix} y_1(t-1) \end{bmatrix} + \dots + \begin{bmatrix} w_p \end{bmatrix} \begin{bmatrix} y_1(t-p) \end{bmatrix} + \begin{bmatrix} e \end{bmatrix}$$

This is the same as:

$$y(t) = a + w_1 * y(t-1) + \dots + w_p * y(t-p) + \varepsilon_t$$

The final error term represents multivariate vector white noise. For multivariate time series,  $e(t)$  should be a continuous random variable that satisfies the following conditions:

$$E(\varepsilon_t) = 0$$

Expected value for the error vector is 0.

$$E(\varepsilon_{t1}, \varepsilon_{t2}) = \sigma_{12}$$

Expected value of  $e(t)$  and  $e(t')$  is the standard deviation of the series.

We considered this to be an effective model because VAR is capable of understanding and using the relationship between several variables for describing the dynamic behaviour of the data. Additionally, implementing VAR is straight forward compared to other more complicated models.

### Attempted Strategies

#### Univariate LSTM model:

In the first attempted strategy, we used the Univariate LSTM model. Preparing the data involved scaling the data so that feature values would be between 0 and 1. This is simply a method used to improve the training process of a neural network. For this model, we reduced the number of features down to 1 (target). We let the timesteps be 60 (which means we were predicting the value of the future price by using past 60 mins of the price as an input). We split the data so that we had 80% of the data for training and 20% for testing. The Univariate LSTM model itself was set up with two LSTM

layers with 100 neurons and 1 Dense layer, with 1 neuron. We compiled the model using the mean squared error (MSE) loss function and the ADAM optimizer.

### MLP for Time Series:

In the second attempted strategy, we used the MLP model. The first stage of preparing the data involved performing several transformations including making the time series data stationary by implementing a lag of 1. This removed the increasing trend in the data. We transformed the time series into a supervised learning problem by organizing the data into input and output patterns where the observation at the previous time step was used as an input to forecast the observation at the current timestep. We split the data so that we had 80% of the data for training and 20% for testing. The MLP model itself was set up as a sequential model with 2 dense layers and a rectified linear activation function. We used a batch size of 2 and ran the model over 30 epochs. The model was fit using the ADAM optimization algorithm and the mean squared error loss function.

### Final Strategy

#### Multivariate Time Series (MTS) - Vector Autoregressions (VAR)

A Multivariate time series has more than one time-dependent variables. Each variable depends not only on its past values but also has some dependencies on other variables. These dependencies are used for forecasting future values.

Vector Auto Regression is one of the most commonly used method for multivariate time series forecasting.

In a VAR model, each variable is a linear function of the past values of itself and the past values of all the other variables.

VAR assumes that the passed time series are stationary. Non-stationary or trending data can often be transformed to be stationary by first-differencing or some other method. For direct analysis of non-stationary time series, a standard stable VAR(p) model is not appropriate. As a result, the data need to pass the Johansen test before pass to VAR.

Here is the Johansen test result of "BTCUSDT".

Johansen Test Results	
Eigen Values [4.62504703e-01 2.83937070e-01 2.58444182e-01 2.53963631e-01 2.04372785e-01 9.68409568e-02 8.84423791e-02 6.86226530e-02 6.61781520e-02 6.41020140e-08]	
Summary of Regression Results	
=====	
Model: VAR	
Method: OLS	
Date: Sun, 02, May, 2021	
Time: 01:13:28	
-----	
No. of Equations: 10.0000	BIC: 58.5955
Nobs: 736584.	HQIC: 58.5841
Log likelihood: -3.20251e+07	FPE: 2.75964e+25
AIC: 58.5797	Det(Omega_mle): 2.75586e+25
-----	

Stationary = Eigen Values \* ['Open', 'High', 'Low', 'Close', 'Volume', 'Quote asset volume', 'Number of trades', 'Taker buy base asset volume', 'Taker buy quote asset volume', 'target']

Since choose a lag order is not easy, we employed the Akaike Information Criterion for order selection.

Akaike Information Criterion (AIC):  $AIC(p) = \ln |\tilde{\Sigma}| + \frac{2K^2p}{T}$

where

- K is the number of variables in the system,
- T is the sample size, and
- $\tilde{\Sigma}$  is an estimate of the covariance matrix  $\Sigma$ .

We set the maximum number of lags to 10.

```
model_fit = model.fit(maxlags=10, ic='aic')
```

## Results and Evaluation

### Univariate LSTM model and MLP

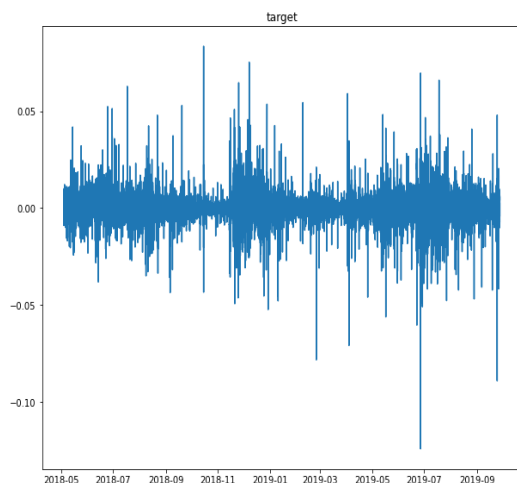
The results of both attempted strategies are not good at all. The reason is the model has its own limitation. For the Univariate LSTM model, it can only intake a series of data. For the MLP, it cannot consider the sequence of data intake.

### Multivariate Time Series (MTS) - Vector Autoregressions (VAR)

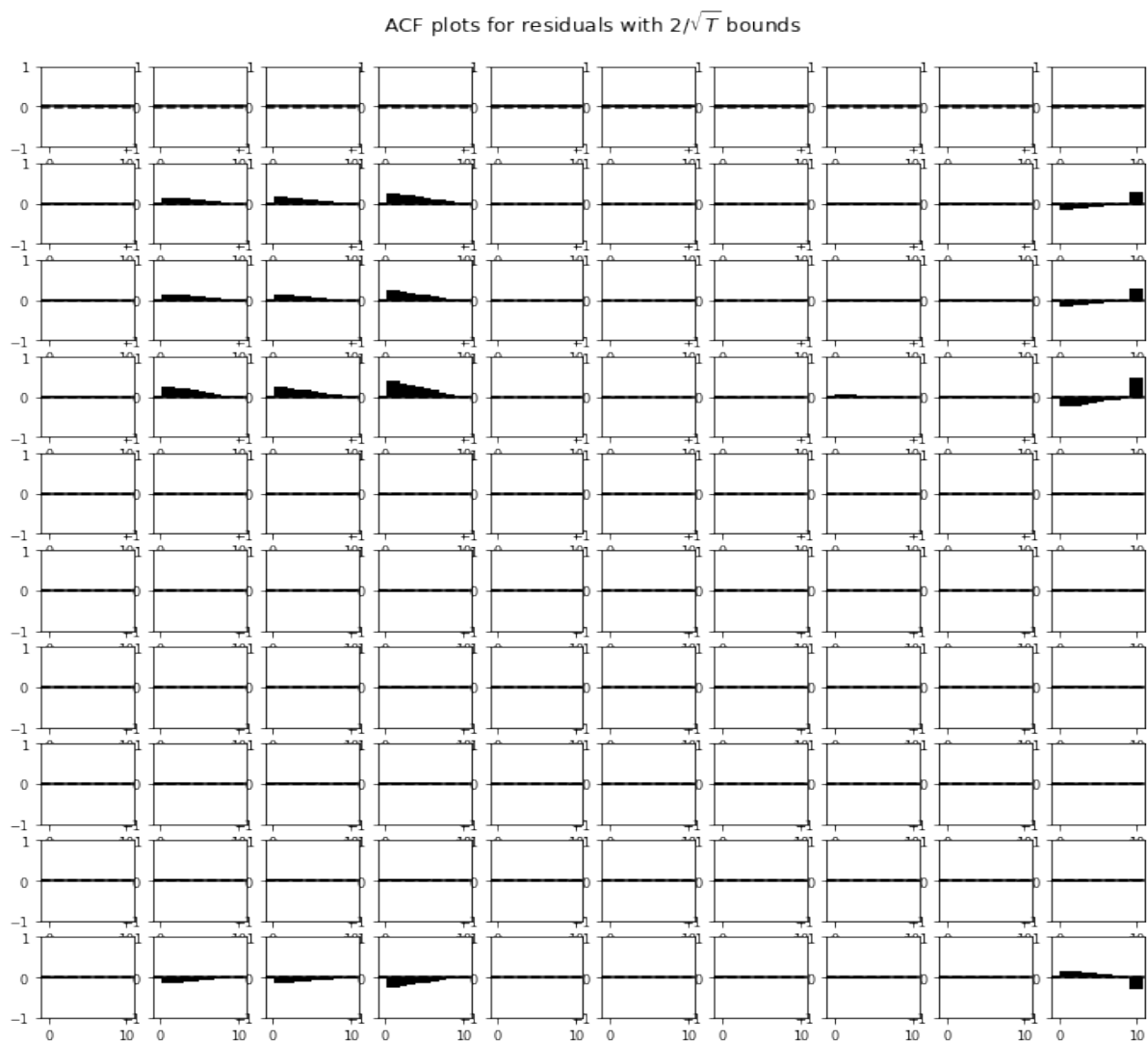
We fitted in different features to the model for a best trained model. Finally, we found out that the original features are the best for training. We use 60% of training records for training and 40% of training records for validation. Here are our results.

Cryptocurrency	RMSE	MAE	Execution Time (mins)
BTCUSDT	0.00332963	0.00169735	18.54
ETHUSDT	0.00395632	0.00225053	18.30
LTCUSDT	0.00412307	0.00237859	18.52
XRPUSDT	0.00068505	0.00039631	18.49

Input time series of 'target' of "BTCUSDT".






Time series autocorrelation function of “BTCUSDT”.



## Summary

At the very beginning (early Mar), we directly use the dataset to train the models (Univariate LSTM, MLP and VAR). We found the result of VAR was surprisingly good.

		<a href="#">Sign In</a>	<a href="#">Register</a>
	<a href="#">Overview</a>	<a href="#">Data</a>	<a href="#">Code</a>
	<a href="#">Discussion</a>	<a href="#">Leaderboard</a>	
#	Team Name	Score 	
1	CryptoPrediction	0.00422	
2	EggSuperman	0.00423	
	Benchmark	0.00431	



And then we try to improve the model by doing data cleansing and varies of data preparation but not success. At the other hands, we studied the Temporal Attention Model but we have not enough time to implement it. It is one of our future works.

### **How to execute our model**

We execute our code in Google Colaboratory. Here is the link.

<https://colab.research.google.com/drive/1BEizpNlvXJTxFa5y4xPB-3rYbHiW4fqo#scrollTo=2Tcg968ECOjm>

Our code and models are placed in Google drive as well. Here is the link.

<https://drive.google.com/drive/folders/1YBdZMM8eSJehQJleXthnfx6NR8UZCyGp?usp=sharing>

If you are not using Google drive, please comment out the code block for connecting Google drive and change the folder variable before executing. Folder variable should point to the folder you store the model files and the testing dataset.

After that click run all and enter '3' for performing testing action only.

## **Bibliography**

### **VAR**

[1] A Multivariate Time Series Guide to Forecasting and Modeling (with Python codes)

<https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/>

[2] Vector Autoregressions tsa.vector\_ar

[https://www.statsmodels.org/dev/vector\\_ar.html](https://www.statsmodels.org/dev/vector_ar.html)

[3] Lesson 18: Building a Vector Autoregressive Model

<http://www.phdeconomics.sssup.it/documents/Lesson18.pdf>

### **Johansen Test**

[4] A Guide to Conducting Cointegration Tests

<https://www.aptech.com/blog/a-guide-to-conducting-cointegration-tests/>

[5] Implement Johansen Test for Cointegration in Python

<https://blog.quantinsti.com/johansen-test-cointegration-building-stationary-portfolio/>

[6] Johansen Test for Cointegrating Time Series Analysis in R

<https://www.quantstart.com/articles/Johansen-Test-for-Cointegrating-Time-Series-Analysis-in-R/>