Machine Learning for Automated Cryptocurrency Trading

Mathee Prasertkijaphan[1*] Ekarat Rattagan[2]

Abstract – This study explores the implementation of five models: RNN, LSTM, GRU, LightGBM and Moving average to forecast the cryptocurrency price movement and uses the result of models to generate buying/selling signal. Moreover, we use the buy/sell signal to build an automated trading system. The result shows that RNN model has the best performance with the lowest coefficient of variation (CV) 6.397%. RNN model can generate the highest profit both single coin portfolio (SOL) and multiple coins portfolio (SOL and BTC). The highest profit is 69 USDT or 7% of 1,000 USDT for the single coin portfolio (SOL). The highest profit is 162 USDT or 16% of 1,000 USDT for the multiple coins portfolio (SOL and BTC).

Keywords: Cryptocurrency, Bitcoin, RNN, LSTM, GRU, LightGBM and Moving average

[*] *Corresponding author. E-mail: mathee.pra@stu.nida.ac.th*

[1] *Graduate Students at Business Analytics and Data Science Graduate School of Applied Statistics National Institute of Development* Administration

[2] *Assistant Professor Dr. at Business Analytics and Data Science Graduate School of Applied Statistics National Institute of Development Administration*

1. Introduction

Data analysis techniques are presently widely used in a variety of industries, including health, marketing, and investment, in order to use the findings for planning to cope with varied situations that have high volatility in the worldwide market. Using historical data, personal experience, or statistical research, we may make predictions about the future. If the prognosis is correct, we can better prepare for taking risks. Considering how to examine data in the investing in cash, government bonds, corporate bonds, mutual funds, stocks, derivatives, or cryptocurrencies is difficult to analyze because each assets has its own volatility. It may lead to a more challenging analytical procedure if the asset is very volatile. Notably the kinds of assets used in cryptocurrencies. It is a sophisticated and unstable asset because the market is always open (24x7), an asset's price may change to reflect market conditions. Therefore, making decisions is necessary. Uncertainty emotion of human such as fear or greed will impact with the trading. According to Morgan Stanley research from 2012, 84 percent of all stock traded on the U.S. Stock Market were carried out by computer algorithm, while just 16 percent were executed by actual investors (Financial Times, 2012). One of the most obvious changes in the financial sector over the past ten years has been the rise of algorithmic trading, or computerized trading systems. An automated trading system generates specific trading choices using sophisticated quantitative models, submits orders automatically, and handles those orders after submission. For example, market declines are a great stress test for an algorithmic trading strategy's usability and competitiveness. Due to the devastating effects of the worldwide coronavirus epidemic in 2020, financial markets were in disarray and sustained significant losses. March 2020 had a 38 percent decline in U.S. markets, which recovered in April 2020 and the following months. Numerous marketplaces and exchanges throughout the world showed this pattern. During the Covid 19 epidemic, algorithmic trading tactics used by hedge funds resulted in significant losses. Many trading systems were taught to adhere to established trade signals, which were unable to adjust to the current market situation. As a result, human-run hedge funds defeated Quant funds (Bloomberg, 2020). For instance, discretionary funds outperformed quantitative models and systematic investment in equity markets during the Covid 19 market shock (Factor Research, 2020). However, their trading algorithms' capacity to continuously process, learn, and react to changing market situations, artificial intelligence funds greatly outperformed discretionary funds. AI funds produced returns that were nearly three times as high as those of other hedge funds. The better returns created by AI and machine learning funds are depicted in (Figure 1 and 2) below, which are supported by both (Friedman, 2019) and (Eurekahedge, 2018). One of the main benefits of automated trading is that since everything is systematized, it removes the emotional component from human traders. Additionally, this technology can lower trading expenses and increase market liquidity.

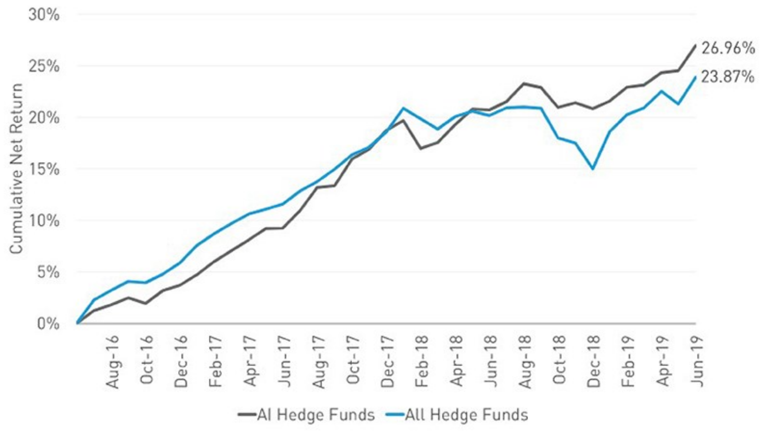


Fig 1: Cumulative 3-Year Returns: AI Hedge Funds vs All Hedge Funds (Friedman, 2019)

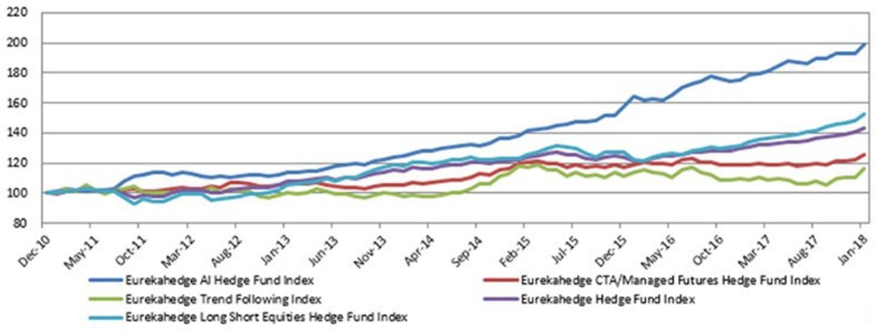


Fig 2: Long-Term Analysis - AI vs Quants vs Traditional Hedge Fund Indices (Eurekahedge, 2018)

This study's firstly focus on predicting changes in cryptocurrency prices utilizing Deep Learning algorithms, Traditional Machine Learning techniques, and Traditional Indicators. In order to improve prediction accuracy, we investigate different deep learning methods, including Simple Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU), as well as Light Gradient Boosting Machine and moving average. We also consider several approaches to constructing the dataset that is used to train algorithms. In order to fully utilize the best prediction models, we then adjust a trading strategy. In addition, we get out-of-sample confirmation of the system's performance by backtesting our suggested automated trading strategy using a market simulator.

For the contribution, we would like to provide the best performance models for predicting the cryptocurrency price movement. Then, investors can use it like a tool to help them make decision for executing. Moreover, we provided more appropriate automated trading strategy to generate the profit from the model under uncertainty market.

2. Related Work

Hamayel, et al. (2021) explored the use of three different recurrent neural network (RNN) algorithms—Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and bidirectional LSTM (bi-LSTM) models—to estimate the movement of cryptocurrency prices. They employed these algorithms to forecast values for three different kinds of cryptocurrencies: Bitcoin (BTC), Litecoin (LTC), and Ethereum (ETH). In relation to the mean absolute percentage error, the models had strong predictive power (MAPE). Therefore, GRU outperformed LSTM and bi-LSTM models in terms of performance for all types of cryptocurrencies. GRU offered the most precise forecast for LTC, with MAPE percentages for BTC, ETH, and LTC of 0.2454 percent, 0.8267 percent, and 0.2116 percent, respectively. The MAPE percentages for BTC, ETH, and LTC were 5.990 percent, 6.85 percent, and 2.332 percent, respectively, for the bi-LSTM algorithm, which had the lowest prediction outcome when compared to the other two algorithms.

Micha´nków et al. (2019) explored deep learning possibilities in time series forecasting by applying buy and sell signal that are generated by LSTM to algorithmic investment strategies, tested on various frequencies: daily, 1h, and 15 min data for BTC and S&P500 index. They constructed 6 portfolios as below

- Portfolio A: BTC combined frequencies (1 day, 1 hour, and 15 min)
- Portfolio B: S&P500 combined frequencies (1 day, 1 hour, and 15 min)
- Portfolio C: Combined assets, RB = 3M, weights 10/90
- Portfolio D: Combined assets, RB = 3M, weights 20/80
- Portfolio E: Combined assets, RB = 6M, weights 10/90
- Portfolio F: Combined assets, RB = 6M, weights 20/80

Their model consisted of three LSTM layers with 512/256/128 neurons respectively and one single neuron dense layer on the output. Each of LSTM layers was using tanh activation function, which allowed to retain negative values. L2 kernel regularization (0.0005) and dropout (0.02) were also applied to each of these layers. The first two layers return sequences with the same shape as the input sequence. The last LSTM layer returned only the last output. They did not use any recurrent dropout because of GPU acceleration during the training process. They used Adam optimizer with learning rate 0.0015. At the result, they found that the combination of weights equal to {S&P500 = W20%, BTC = 80%} was always better than {S&P500 = W10%, BTC = W90%}. Moreover, they found that the length of rebalancing period equal to RB6m was always better than RB3m.

Atsalakis et al. (2019) explored the use of neuro-fuzzy methods to anticipate the movement of the Bitcoin price. Their research suggested a method of computational intelligence that forecasted the direction of change in the daily price of Bitcoin using a hybrid Neuro-Fuzzy controller called PATSOS. The suggested strategy outperforms two previous computational intelligence models, the first of which was created using a less complex neuro-fuzzy approach, and the second using artificial neural networks. Additionally, the investment returns generated by a trading simulation based on the signals of the suggested model are 71.21 percent greater than those generated by a simple buy-and-hold strategy. The PATSOS system performed well even while using other coins.

Wu et al. (2018) studied about how to forecast Bitcoin price movement utilizing applied LSTM models. They created a novel forecasting framework using the LSTM model to anticipate the daily price of bitcoin using two different LSTM models (conventional LSTM model and LSTM with AR (2) model). The suggested models' performance was assessed using daily bitcoin price data from January 1, 2018 to July 28, 2018. As a consequence, they discovered that LSTM with AR (2) model outperformed conventional LSTM model. The goal of this study was to provide a novel forecasting framework for bitcoin price prediction that can overcome and solve the difficulty of input variable selection in LSTM without tight data assumption assumptions.

Ning Lu (2016) conducted research on a model for predicting stock prices in the US market. To forecast a stock's future price changed, he utilized its prior performance. His approach involved analyzing the stock's past performance across N-day (N=4) windows to identify patterns that may be used to forecast the price in the future given a fresh N-day window. His research used both an individual and a sector perspective. The first method predicted stock price movement for each unique approach using just the Ridge Logistic Regression model. The final technique combined two ensemble approaches, Majority Vote and Random Subset, with four models: Lasso Logistic Regression, Decision Tree, Naive Bayes, and Support Vector Machine. He discovered that analyzing a company's historical stock price alone was insufficient to forecast its future results. In order to anticipate the target company's next-day return, the whole sector in which it operated was examined. This was done using previous pricing data for all of the firms in the sector. His trading strategies surpassed risk-free interest rates and some even outperformed the S&P 500 over the out-of-sample testing period, achieving promising Sharpe ratios with almost 20% returns after transaction costs.

3. Background

Recurrent Neural Network (RNN)

A recurrent neural network is a feed-forward neural network with an internal memory. RNN is recurrent in nature since it executes the same function for each data input. The outcome of the current input is dependent on the previous calculation. After the output is generated, it is replicated and returned to the recurrent network. It evaluates the current input and the output that it has learnt from the prior input while making a decision. RNNs, unlike feed-forward neural networks, may process input sequences using their internal state (memory). As a result, they may be used for tasks like unsegmented, linked handwriting recognition or speech recognition. In other neural networks, the inputs are completely independent of one another. However, with RNN, all of the inputs are connected to one another.

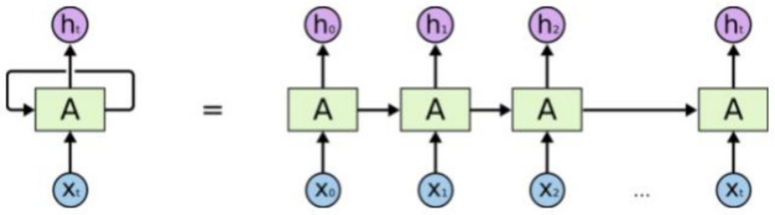


Fig 3: The architecture of RNN (Colah blog, 2015)

The equations behind RNN operations are as follows.

$$s_t = f(st{-}1 * W + x_t * U + b_h) \quad (1)$$
$$y_t = s_t * V + b_y \quad (2)$$

Where s_t is the vector from the output of hidden layer function f. It takes the combination between output vector of previous state s_{t-1} multiplied by internal weight W and the current state input x_t multiplied by the weight V and the bias of hidden layer s_t then multiplied by the weight V and combined with bias by. The final output of RNN is y_t. s_t will be passed to the next state at t + 1 while the weight U, W and V are shared

Long Short-Term Memory (LSTM)

LSTM networks are a form of recurrent neural network (RNN) that can maintain track of long-term relationships in data, allowing them to partially address the vanishing gradient problem (Goodfellow et al., 2016). They're commonly employed to simulate sequential data like text, audio, and time series. LSTM units are made up of memory cells, each with three different types of gates (input gate, output gate and forget gate). These gates utilize tanh and sigmoid functions to control the flow of information through the cell, determining how much and which information should be kept in the long term, sent on to the next stage, or deleted. The input vector for the LSTM network (x_t) in our study was a series of previous observations from SOL and BTC data, and the output vector (h_t) was a single value predicted for the following period. Figure 4 illustrates that there are four neuron network layers interacting in each node. Output of one node is used as an input for the subsequent node.

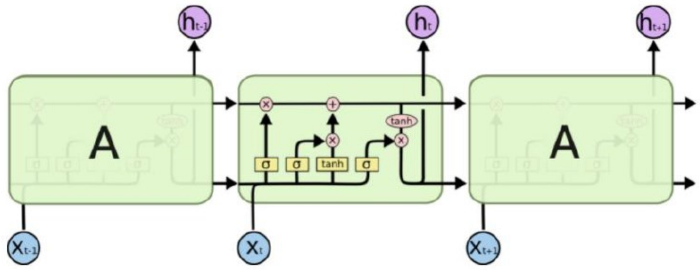


Fig 4: The architecture of LSTM (Colah blog, 2015)

The equations behind LSTM operations are as follows.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \quad (3)$$
$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \quad (4)$$
$$\acute{C}t = tanh(WC * [ht{-}1, xt] + bC) \quad (5)$$
$$Ct = ft * Ct{-}1 + it * \acute{C}t \quad (6)$$
$$Ot = \sigma(Wo * [ht{-}1, xt] + bo) \quad (7)$$
$$ht = Ot * \tanh(Ct) \quad (8)$$

where f_t, i_t, $\acute{C}_t$, C_t, O_t, and h_t represent forget gate layer, input gate layer, a vector of new candidate value created by tanh layer, a new cell state, output gate, and output information respectively. σ indicates sigmoid function.

Gated recurrent units (GRU)

A gated recurrent unit (GRU) was proposed by (Cho et al., 2014) to make each recurrent unit to adaptively capture dependencies of different time scales. Similarly, to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cell. The update gate and the reset gate decide how much past information is passed on and how much is discarded respectively. The cell state and hidden state are also combined. The GRU is less complex than LSTM model as it has less parameters. GRU is preferred when the dataset is small while LSTM is elected for larger dataset.

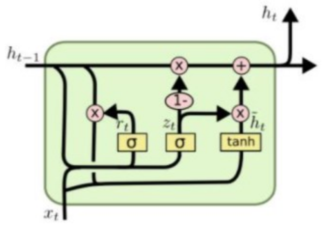


Fig 5: The architecture of GRU (Colah blog, 2015)

The equations behind GRU operations are as follows.

$$r_t = \sigma(W_r h_{t-1} + U_r x_t) \tag{9}$$
$$\hat{h}_t = tanh(\mathrm{W}\,(r_t * h_{t-1}) + \mathrm{U}x_t) \tag{10}$$
$$z_t = \sigma(W_z h_{t-1} + U_z x_t) \tag{11}$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t) \tag{12}$$

where h_t, h_{t-1}, r_t, and z_t represent the output of the current and previous states, the reset gates, the update gates, respectively. σ is the sigmoid function while W_r and U_r indicate the weight matrices.

Light Gradient Boosting Machine (LightGBM)

LightGBM splits the tree leaf-wise as opposed to other boosting algorithms that grow tree level-wise. It chooses the leaf with maximum delta loss to grow. Since the leaf is fixed, the leaf-wise algorithm has lower loss compared to the level-wise algorithm. Leaf-wise tree growth might increase the complexity of the model and may lead to overfitting in small datasets. (Ke et. Al., 2017). Below is a diagrammatic representation of Leaf-Wise Tree Growth.

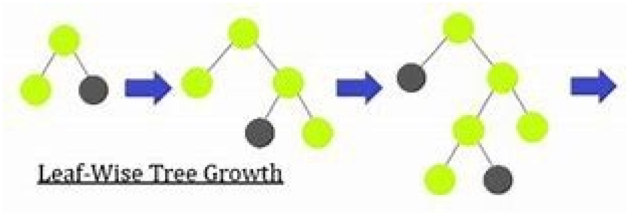


Fig 6: Leaf-wise growth (GeeksforGeeks.org, 2021)

Simple Moving Average (SMA)

SMA is a simple technical indicator calculated by adding the most recent data points in a set and dividing the total by the number of time periods. The SMA indicator is used by traders to create signals on whether to join or quit a market. Because it is based on prior price data for a certain period, a SMA is backward-looking. It may be calculated for several sorts of pricing, such as high, low, open, and close. Analysts and investors in financial markets use the SMA indicator to establish buy and sell recommendations for stocks. The SMA assists in identifying support and resistance prices in order to receive signals on where to start or quit a trade. Traders must first compute the SMA by adding prices over a certain time and dividing the total by the entire number of periods. The data is then displayed on a graph.
The formula for Simple Moving Average is written as follows:

$$\mathrm{SMA} = (\mathrm{R}_1 + \mathrm{R}_2 + \ldots\ldots\ldots\ \mathrm{R}_n) / \mathrm{n} \tag{13}$$

where R is the return in period n and n is the number of periods.

Evaluate Model

Mean Square Error (MSE), Root mean square error (RMSE), Mean Absolute Error (MAE) and Coefficient of Variation (CV) measures defined by equations (14), (15), (16) and (17) were utilized to evaluate the performance of the models in this study.

MSE measures the average squared difference between actual values and its predicted values. The lower the MSE, the better a model fits a dataset.

$$\text{MSE} = \frac{\sum(\hat{y}_i - y_i)^2}{n} \tag{14}$$

RMSE measures the square root of the average squared difference between actual values and its predicted values. The lower the RMSE, the better a model fits a dataset.

$$\text{RMSE} = \sqrt{\frac{\sum(\hat{y}_i - y_i)^2}{n}} \tag{15}$$

MAE measures the difference between actual values and its predicted values. The expected value for best performance model is zero.

$$\text{MAE} = \frac{\sum|\hat{y}_i - y_i|}{n} \tag{16}$$

where Σ is a symbol that means "sum", $\hat{y}_i$ is the predicted value for the ith observation, y_i is the observed value for the ith observation and n is the sample size

CV is the standard deviation to mean ratio. The degree of dispersion around the mean is inversely proportional to the coefficient of variation. In most cases, it is stated as a percentage. Without units, it enables comparison of distributions of values with dissimilar measuring scales. The CV links the estimated value to the estimated standard deviation. The estimate is more accurate the lower its value is for the coefficient of variation.

$$\text{CV} = \frac{\sigma}{\mu} \tag{17}$$

where σ is standard deviation and μ is mean

4. Methodology

The suggested process design is depicted in Figure 7. After getting the data set from the Binance API, the first stage is to undertake data exploration, which includes checking the completeness of the data, exploring the type of data, and investigating if the data matches the Normal distribution. Following that, features are chosen and data is pre-processed. Following that, experiments on various models are carried out. We will make the prediction and use the outcome to do the data inverse transform. Following the data inverse transform, we will convert the return to the closing price before doing prediction and assessment. The next phase is to backtest a trading strategy using five models: RNN, LSTM, GRU, LightGBM, and MA.

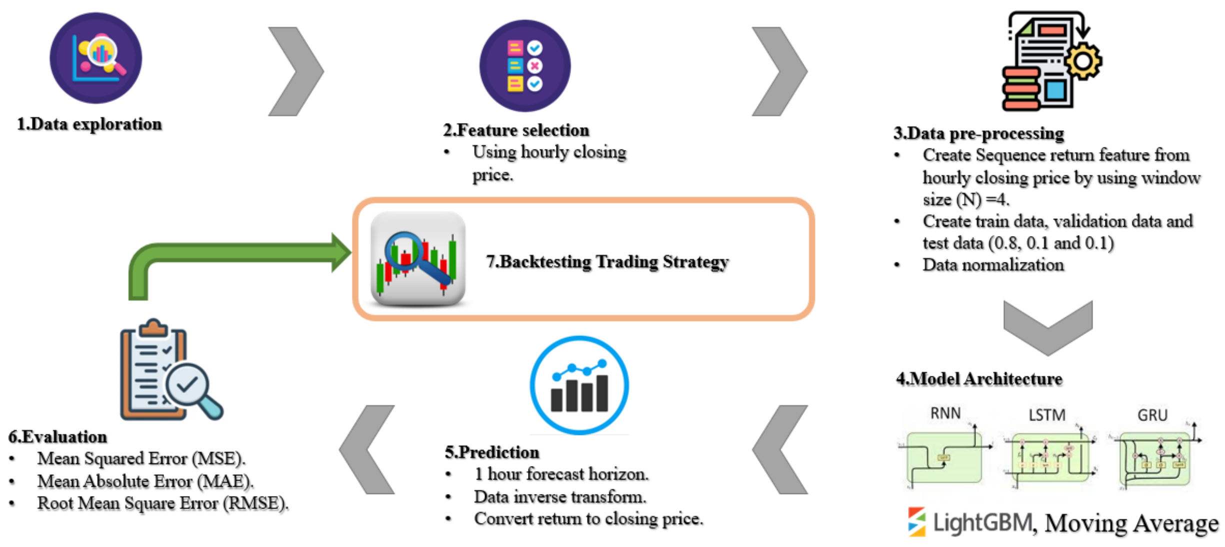

1.Data exploration
2.Feature selection
• Using hourly closing price.
3.Data pre-processing
• Create Sequence return feature from hourly closing price by using window size (N) =4.
• Create train data, validation data and test data (0.8, 0.1 and 0.1)
• Data normalization
4.Model Architecture
RNN
LSTM
GRU
LightGBM, Moving Average
5.Prediction
• 1 hour forecast horizon.
• Data inverse transform.
• Convert return to closing price.
6.Evaluation
• Mean Squared Error (MSE).
• Mean Absolute Error (MAE).
• Root Mean Square Error (RMSE).
7.Backtesting Trading Strategy

Fig 7: Process design

4.1 Dataset

1. SOL-USDT

Data are gathered from Binance API (from binance.client import Client). The period of the collected hourly data is from December 1, 2020 to April 30, 2022. The details of this cryptocurrency were collected as followed:

- Symbol
- Date
- Closing price
- Volume
- Number of trades

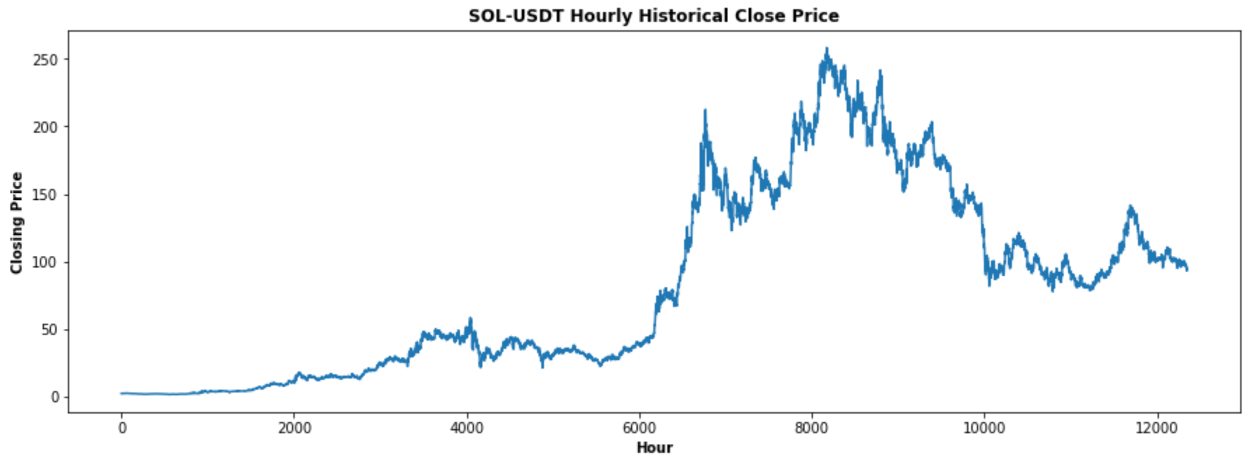


Fig 8: SOL-USDT Hourly Historical Close Price

Items	Close Price	Volume	Number of trades	Return
count	12,351	12,351	12,351	12,351
mean	81.76	182,837.80	20,989.21	0.000458
std	69.74	222,052.90	31,649.80	0.02
min	1.18	0	0	-0.19
25%	24.21	68,304.95	5,167.50	-0.008
50%	54.46	116,192.40	11,676.00	0
75%	139.17	210,888.60	24,559.50	0.00785
Max	258.44	4,974,114.00	585,919.00	0.19
Var	4,863.33	49,307,480,000.00	1,001,710,000.00	0.0003
skew	0.61	5.74	5.54	0.55
kurt	-0.84	65.05	51.70	10.44

Table 1: The descriptive statistics of SOL-USDT

2. BTC-USDT

Data are gathered from Binance API (from binance.client import Client). The period of the collected hourly data is from December 1, 2020 to April 30, 2022. The details of this cryptocurrency were collected as followed:

- Symbol
- Date
- Closing price
- Volume
- Number of trades

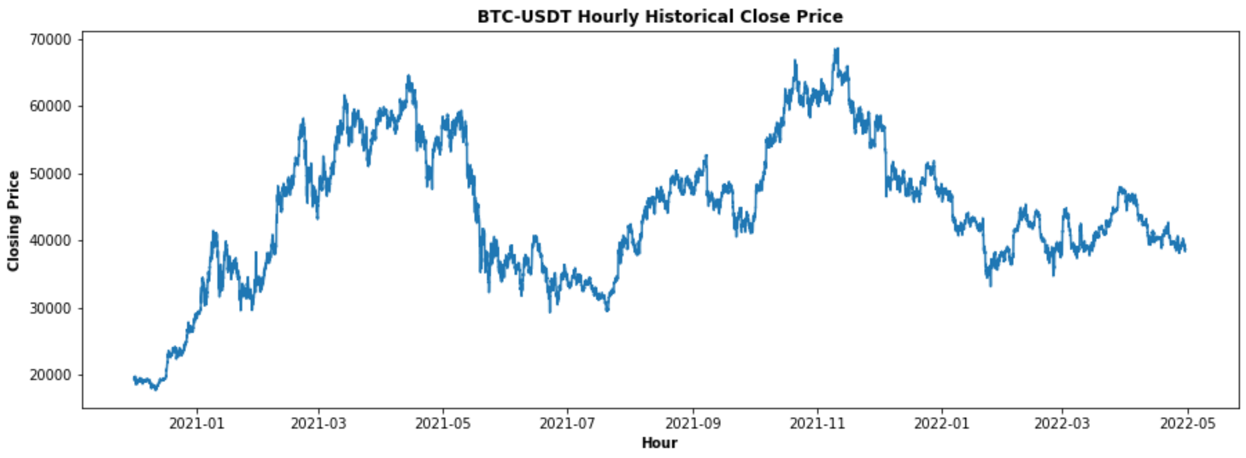


Fig 9: SOL-USDT Hourly Historical Close Price

Items	Close Price	Volume	Number of trades	Return
count	12,351	12,351	12,351	12,351
mean	44,423.46	2,697.04	68,849.19	0.000093
std	10,499.25	2,293.15	44,741.990000	0.008587
min	17,647.71	0	0	-0.09
25%	37,646.29	1,341.03	39,558.50	-0.004
50%	43,841.57	2,085.94	57,780.00	0.00011
75%	51,323.53	3,267.04	84,828.50	0.00397
Max	68,633.69	44,239.81	799,206.00	0.12
Var	110,234,200.00	5,258,556.00	2,001,846,000.00	0.0001
skew	-0.16	4.17	3.34	-0.04
kurt	-0.25	34.82	25.75	11.59

Table 2: The descriptive statistics of BTC-USDT

4.2 Feature selection

We selected daily returns over prices for constructing both BTC dataset and SOL dataset. Because most price series are geometric random walks, the anticipated value of error in a regression must be zero. However, the returns are frequently distributed randomly around a mean of zero (Chan et. Al., 2013).

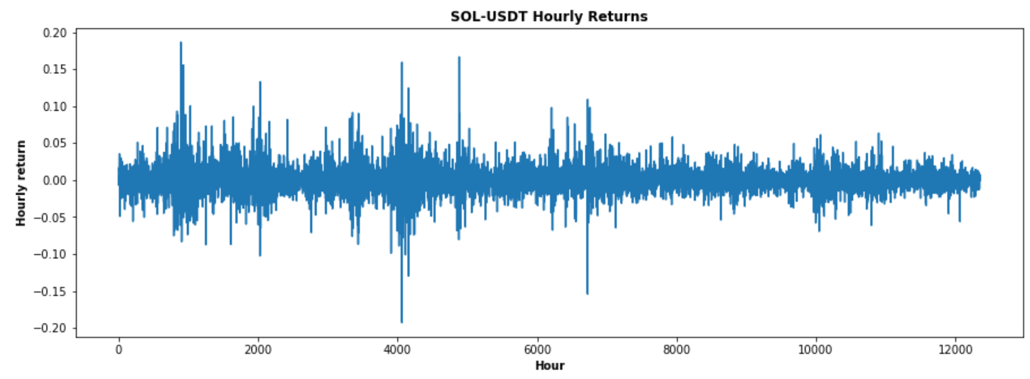


Fig 10: SOL-USDT Hourly Return

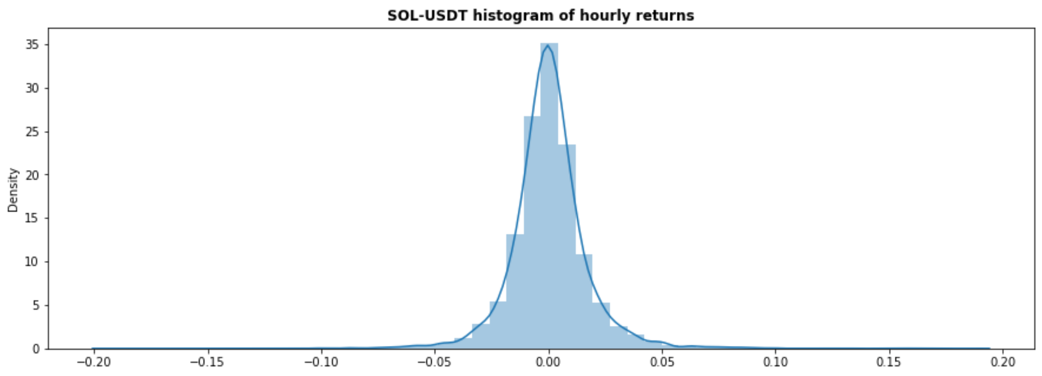


Fig 11: SOL-USDT Histogram of Hourly Return

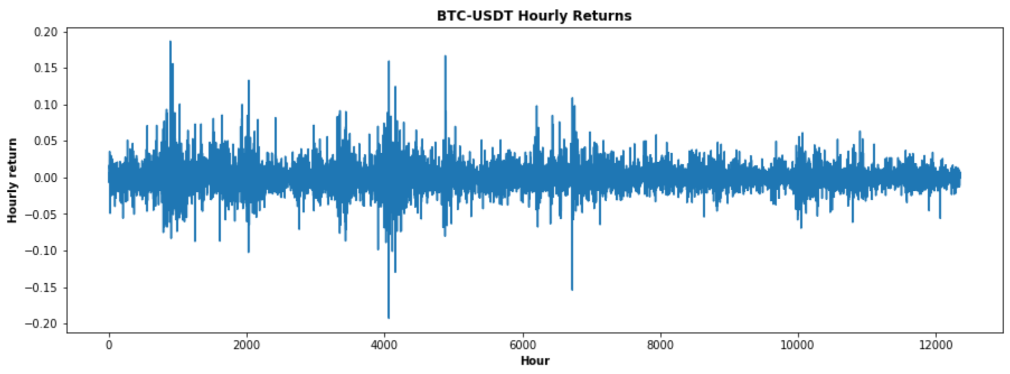


Fig 12: BTC-USDT Hourly Return

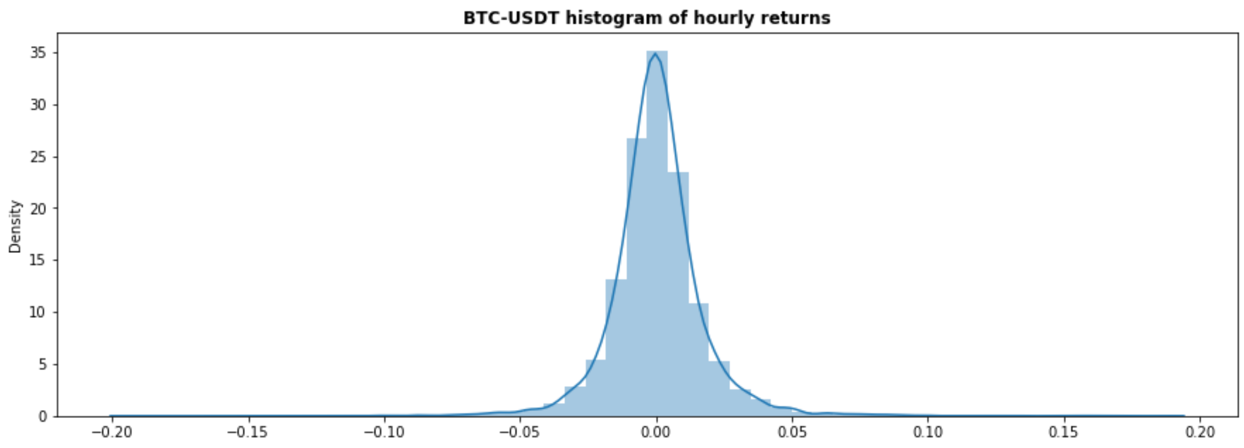


Fig 13: BTC-USDT Histogram of Hourly Return

4.3 Problem statement

Model

$$Y_4 = [R_1, R_2, R_3] \quad (18)$$

Input

- R_1 is the return of SOL from day 1 to day 2.
- R_2 is the return of SOL from day 2 to day 3.
- R_3 is the return of SOL from day 3 to day 4.
- For example, [0.43412612, 0.43116426, 0.43277213] → [0.43006414]
- The return is the percentage of different price between buying price and selling price.

Output

- Y_4 is the return of SOL from day 4 to day 5.

Objective

- We used the returns of SOL itself from day 1 to day N to predict the return at day N+1. So, N is the window size that represents how far we want the model to look back (For this study, we use N = 4).

4.4 Data Preprocessing

Input preparation

We created sequence return feature from hourly closing price by using window size (N) = 4.

Train Test Validation Split

The data were split into training set, validation set and test set with 80%, 10% and 10%, respectively and random state = 0. Total transactions are 12,346 records. These transactions were split into training set, validation set and test set with 9,876, 1,236 and 1,234%, respectively. We can find more detail as below Figure 14.

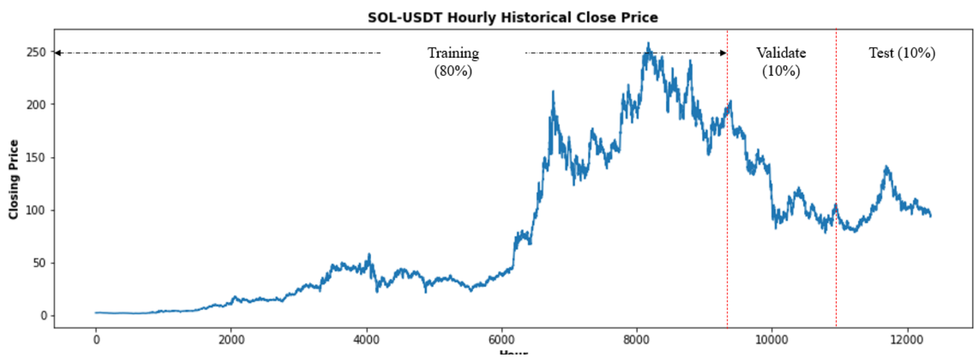


Fig 14: Train Test Validation Split

Data normalization

The features were transformed by min max scaler range 0 to 1 because of the little change in return. The transformation was given by the following formular.

$$X_{scaled} = X_{std} * (\max - \min) + min \quad (19)$$

$$X_{std} = \frac{(X - min)}{(max - min)} \quad (20)$$

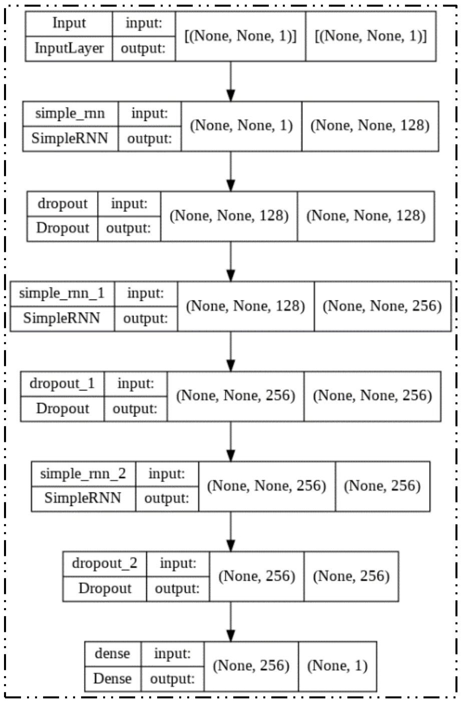
where min, max is range 0 to 1

4.5 Architecture

There are five models that must be implemented and improved for optimal performance. Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated recurrent units are deep learning models (GRU). Light Gradient Boosting Machine (LightGBM) and Simple Moving Average (SMA) are the others. Because we would like to determine the impact of each deep learning model if the model is only altered, we employ the same architecture for all deep learning models.

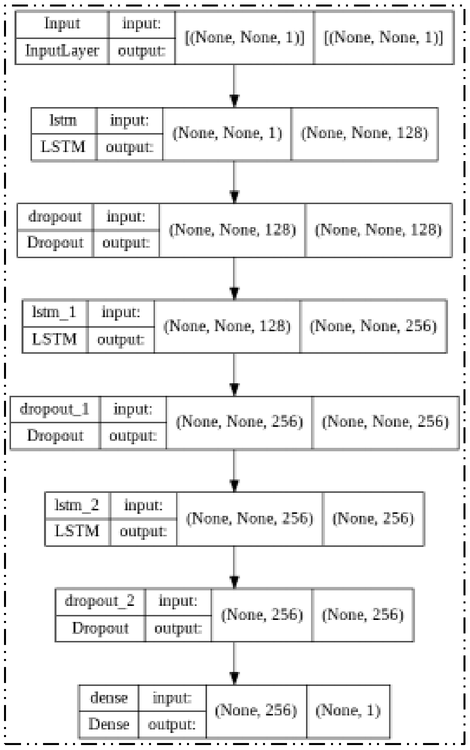
4.5.1) Deep learning models (RNN, LSTM and GRU)

Their models consist of three layers with 128/256/256 neurons respectively and one single neuron dense layer on the output with Linear activation function. Each of LSTM layers is using ReLU activation function. We set dropout (0.25) are also applied to each of these layers. The first two layers return sequen ces with the same shape as the input sequence. The last LSTM layer returns only the last output. Moreover, we use Stochastic gradient descent (SGD) for optimization algorithms with learning rate = 0.0001, decay=1e-5, momentum=0.9 and nesterov=True. We use Mean Squared Error (MSE) and mean absolute error (MAE) as loss function. The summary is depicted as below


Input | InputLayer | input: | output: | [(None, None, 1)] | [(None, None, 1)]
simple_rnn | SimpleRNN | input: | output: | (None, None, 1) | (None, None, 128)
dropout | Dropout | input: | output: | (None, None, 128) | (None, None, 128)
simple_rnn_1 | SimpleRNN | input: | output: | (None, None, 128) | (None, None, 256)
dropout_1 | Dropout | input: | output: | (None, None, 256) | (None, None, 256)
simple_rnn_2 | SimpleRNN | input: | output: | (None, None, 256) | (None, 256)
dropout_2 | Dropout | input: | output: | (None, 256) | (None, 256)
dense | Dense | input: | output: | (None, 256) | (None, 1)

Hyperparameter	Selected Value
No. hidden layer	3
No. neurons	128/256/256
Activation function	relu/linear
Dropout rate	0.25
Optimizer	SGD
Learning Rate	0.0001
Decay	0.00001
Momentum	0.9
Nesterov	TRUE
SOL train/test	9,876/1,236
Batch size	64
Epochs	30

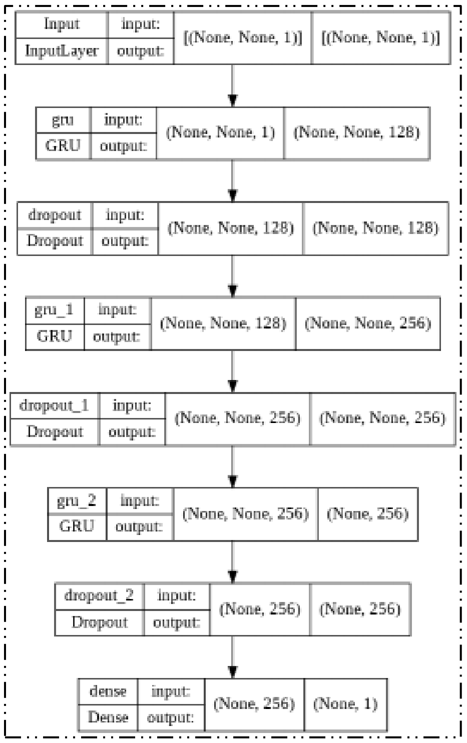
Fig 15: RNN's Architecture



Input | input: | [(None, None, 1)] | [(None, None, 1)]
InputLayer | output:
lstm | input: | (None, None, 1) | (None, None, 128)
LSTM | output:
dropout | input: | (None, None, 128) | (None, None, 128)
Dropout | output:
lstm_1 | input: | (None, None, 128) | (None, None, 256)
LSTM | output:
dropout_1 | input: | (None, None, 256) | (None, None, 256)
Dropout | output:
lstm_2 | input: | (None, None, 256) | (None, 256)
LSTM | output:
dropout_2 | input: | (None, 256) | (None, 256)
Dropout | output:
dense | input: | (None, 256) | (None, 1)
Dense | output:

Hyperparameter	Selected Value
No. hidden layer	3
No. neurons	128/256/256
Activation function	relu/linear
Dropout rate	0.25
Optimizer	SGD
Learning Rate	0.0001
Decay	0.00001
Momentum	0.9
Nesterov	TRUE
SOL train/test	9,876/1,236
Batch size	64
Epochs	30

Fig 16: LSTM's Architecture



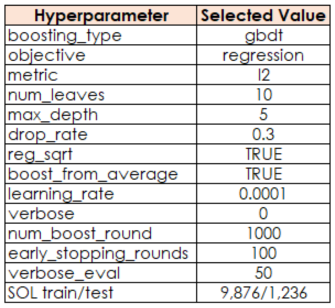
Input | input: | [(None, None, 1)] | [(None, None, 1)]
InputLayer | output:
gru | input: | (None, None, 1) | (None, None, 128)
GRU | output:
dropout | input: | (None, None, 128) | (None, None, 128)
Dropout | output:
gru_1 | input: | (None, None, 128) | (None, None, 256)
GRU | output:
dropout_1 | input: | (None, None, 256) | (None, None, 256)
Dropout | output:
gru_2 | input: | (None, None, 256) | (None, 256)
GRU | output:
dropout_2 | input: | (None, 256) | (None, 256)
Dropout | output:
dense | input: | (None, 256) | (None, 1)
Dense | output:

Hyperparameter	Selected Value
No. hidden layer	3
No. neurons	128/256/256
Activation function	relu/linear
Dropout rate	0.25
Optimizer	SGD
Learning Rate	0.0001
Decay	0.00001
Momentum	0.9
Nesterov	TRUE
SOL train/test	9,876/1,236
Batch size	64
Epochs	30

Fig 17: GRU's Architecture

4.5.2) Light Gradient Boosting Machine (LightGBM)

LightGBM is a quick and high-performance model due of employing Gradient Boosting. It is learning techniques to create high precision models. It learns from the accumulated mistake caused by the predictions of the previously formed model. We specify parameters like {'boosting type': 'gbdt', 'objective': 'regression', 'metric':'l2', 'num leaves':10, 'max depth':5, 'drop rate': 0.3, 'reg sqrt': True, 'boost from average': True, 'learning rate': 0.0001, 'verbose': 0, } and set the train parameter to num boost round=1000, early stopping rounds=100, verbose eval=50.. The summary is depicted as below

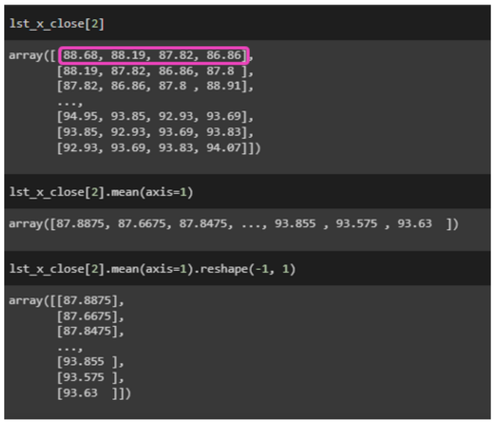


Hyperparameter	Selected Value
boosting_type	gbdt
objective	regression
metric	l2
num_leaves	10
max_depth	5
drop_rate	0.3
reg_sqrt	TRUE
boost_from_average	TRUE
learning_rate	0.0001
verbose	0
num_boost_round	1000
early_stopping_rounds	100
verbose_eval	50
SOL train/test	9,876/1,236

Fig 18: LightGBM's Architecture

4.5.3) Simple Moving Average (SMA)

It is simply the average price over the specified period (N=4).



```
lst_x_close[2]

array([[88.68, 88.19, 87.82, 86.86],
       [88.19, 87.82, 86.86, 87.8 ],
       [87.82, 86.86, 87.8 , 88.91],
       ...,
       [94.95, 93.85, 92.93, 93.69],
       [93.85, 92.93, 93.69, 93.83],
       [92.93, 93.69, 93.83, 94.07]])

lst_x_close[2].mean(axis=1)

array([87.8875, 87.6675, 87.8475, ..., 93.855 , 93.575 , 93.63  ])

lst_x_close[2].mean(axis=1).reshape(-1, 1)

array([[87.8875],
       [87.6675],
       [87.8475],
       ...,
       [93.855 ],
       [93.575 ],
       [93.63  ]])
```

Fig 19: Simple Moving Average (SMA)

5. Empirical result

A validation dataset is created with 10% of the training data in order to assess the performance of the model on that validation dataset. To protect overfitting by introducing some noise, neurons have drop out in their structure. For data used for validation, the drop out does not remove random neurons. As shown in Figures (20, 21 and 22), training loss and validation loss were shown by epoch to evaluate the overfitting problem.

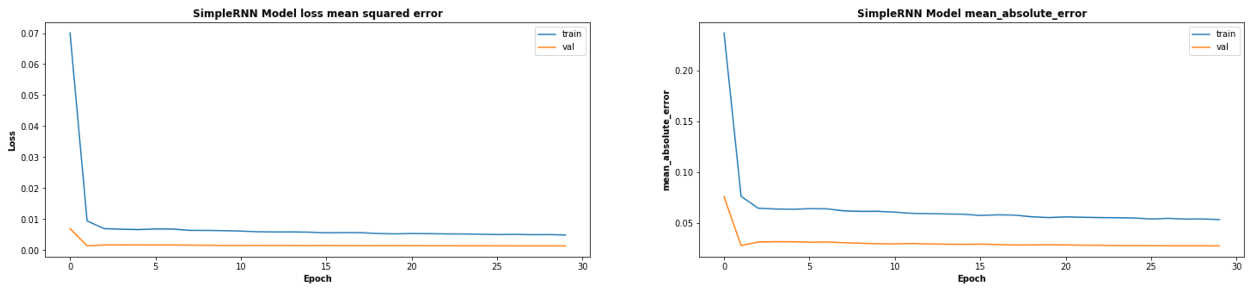


Fig 20: Training loss vs. validation loss (RNN)

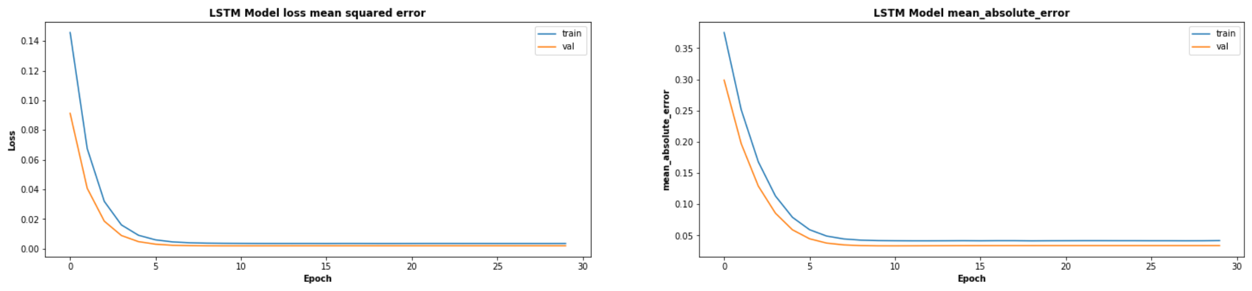


Fig 21: Training loss vs. validation loss (LSTM)

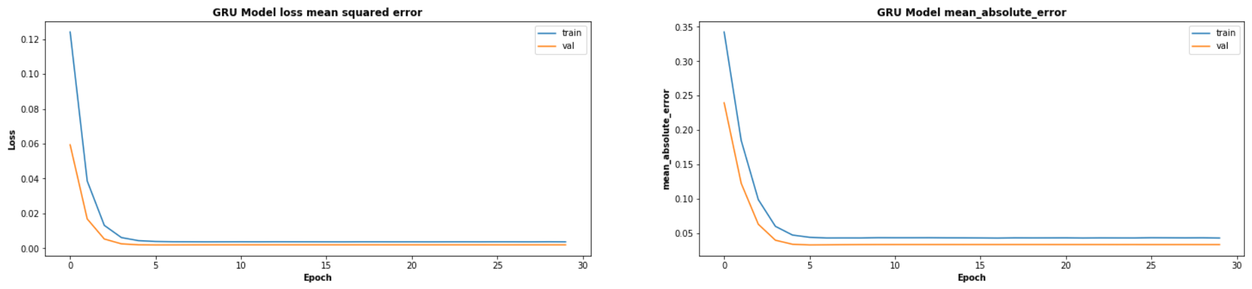


Fig 22: Training loss vs. validation loss (GRU)

5.1 Evaluation

After the experiments, as shown in Figures 23, the MA model has performed best with the lowest MSE of 1.88, RMSE of 1.37, and MAE of 0.99. Second best model is RNN with MSE, RMSE, MAE of 2.75, 1.66, and 1.21 respectively. However, if we look at Coefficient of Variation (CV), we found that SimpleRNN has the best performance model with the lowest CV of 6.397% as shown in Figures 24

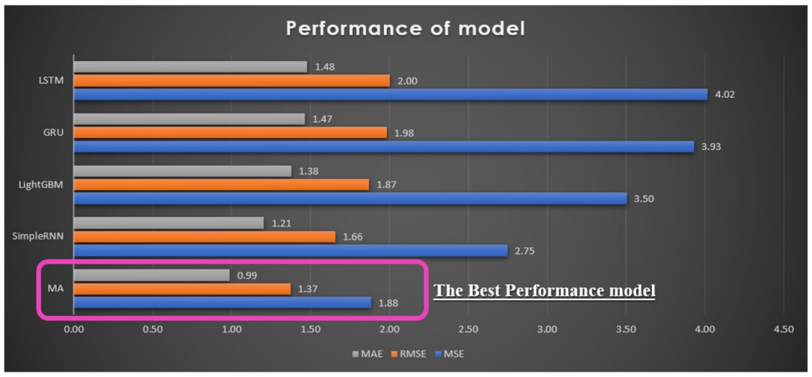


Fig 23: Performance of model

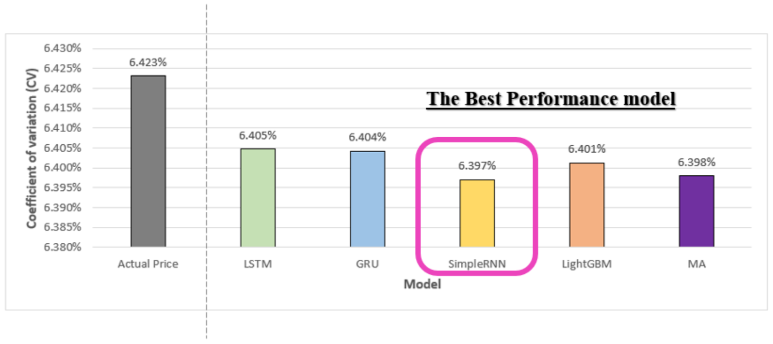


Fig 24: Coefficient of variation (CV)

5.2 Diagnostic analysis

After analyzing evaluation result, MA is the best performance model. The model's predicted SOL price and actual SOL price during test period is illustrated in Figure 25. The line plot shows that the model able to satisfactorily predict SOL price. However, when zooming in to investigate closer, Figure 26 indicates that during high volatility period the gap between actual and predicted values widen.

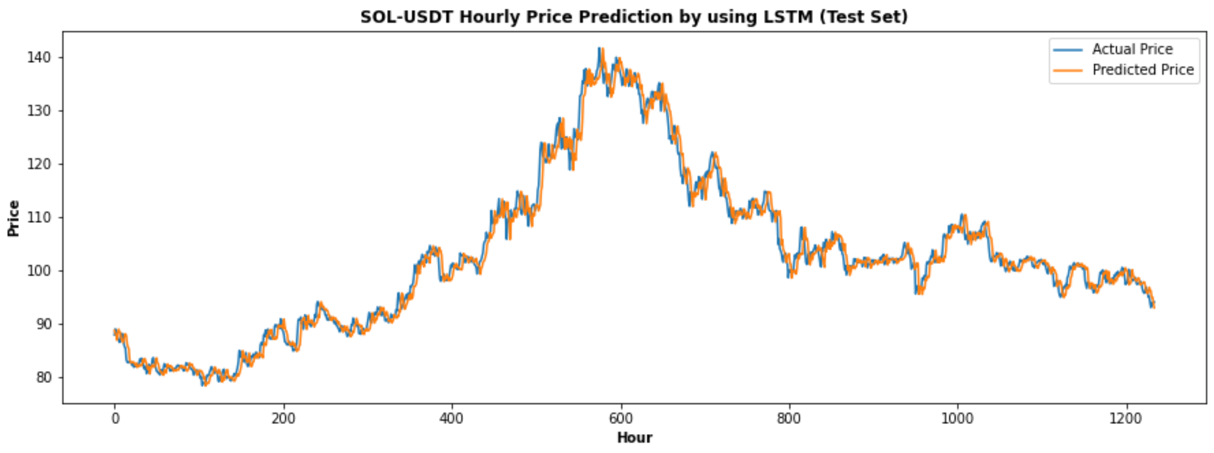


Fig 25: SOL-USDT Hourly Price Prediction by using LSTM

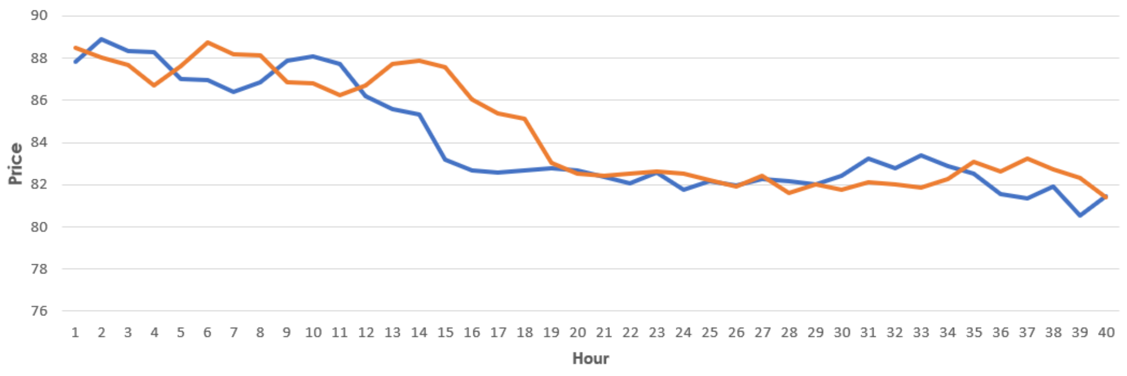


Fig 26: SOL-USDT Hourly Price Prediction by using LSTM (Zoom in)

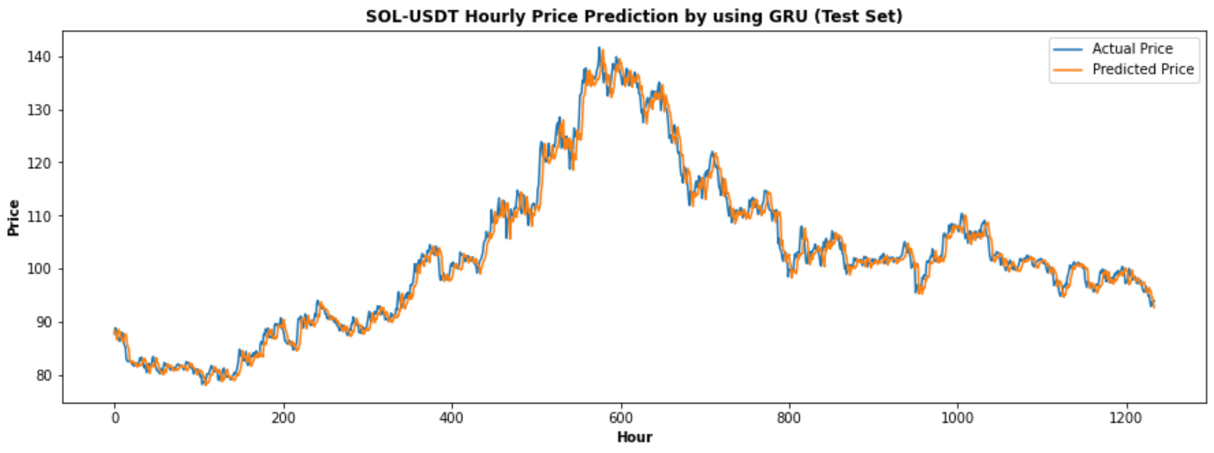


Fig 27: SOL-USDT Hourly Price Prediction by using GRU

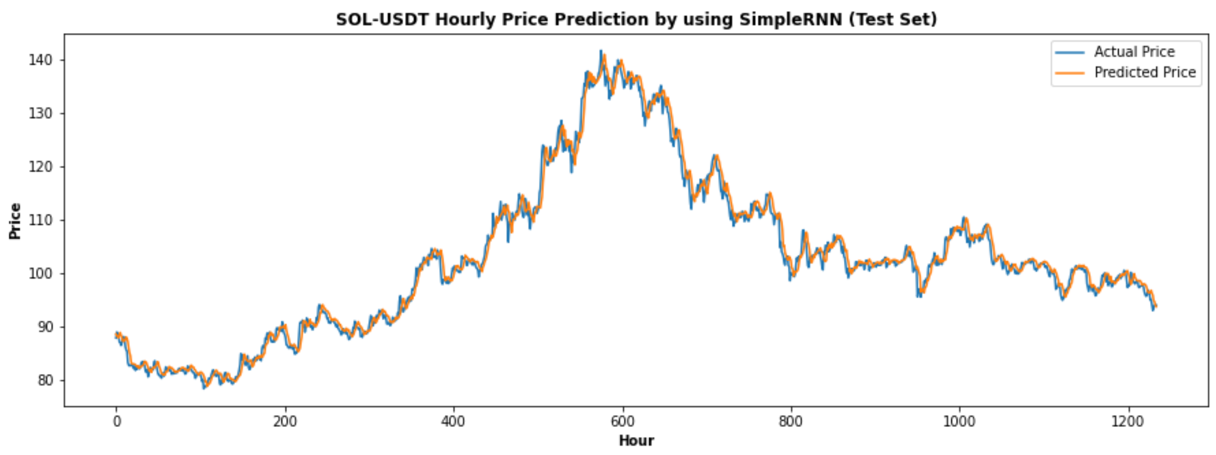


Fig 28: SOL-USDT Hourly Price Prediction by using SimpleRNN

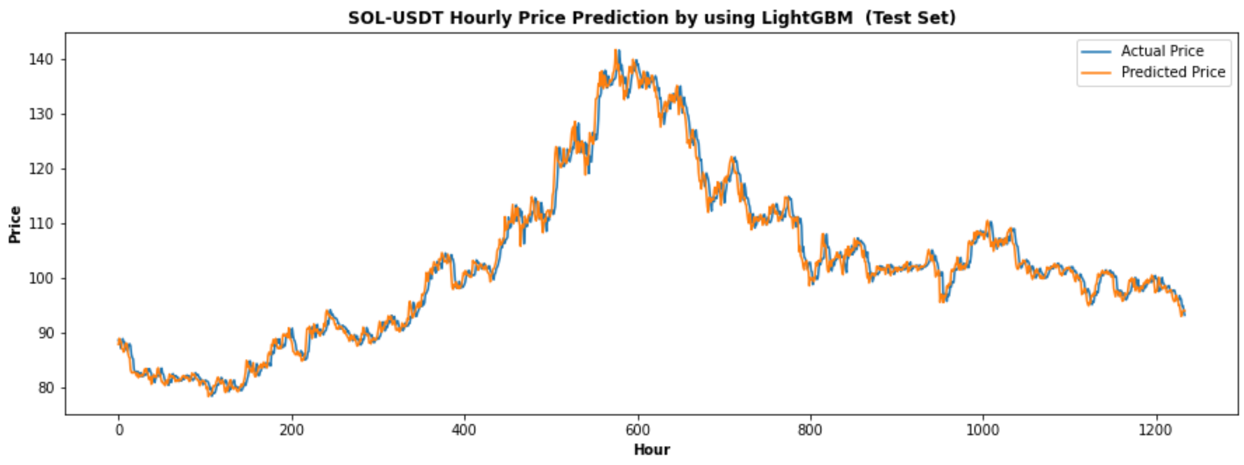


Fig 29: SOL-USDT Hourly Price Prediction by using LightGBM

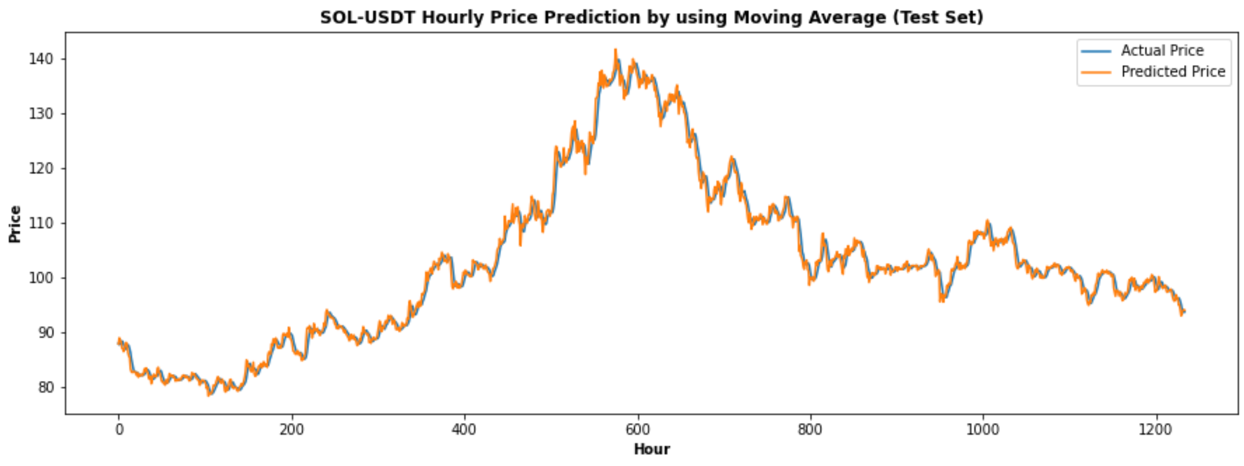
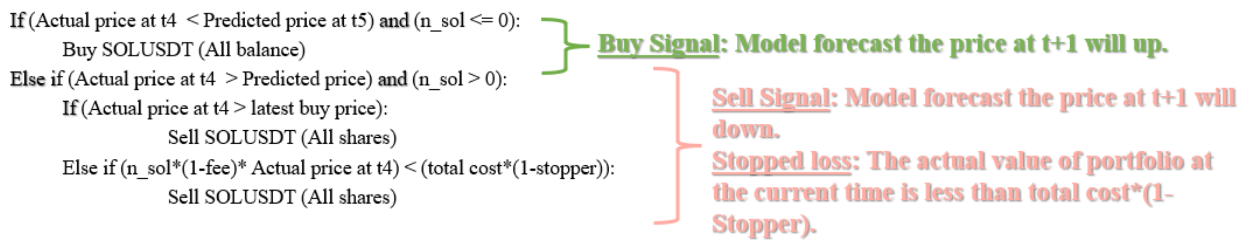


Fig 30: SOL-USDT Hourly Price Prediction by using Moving Average

5.3 Backtesting Trading Strategy

The building an automated trading system is to specify a trading strategy that use our model predictions as input and outputs actual buy/sell orders. We construct two portfolios for backtesting and set the budget 1,000 USDT. The first of portfolios is only trading for single coin that is SOL-USDT. The last of portfolios is trading for multiple coins that are SOL-USDT and BTC-USDT. We use same trading strategy for two portfolios to generate buy/sell signal. If the model forecast the price at t+1 will up and our portfolio don't have SOL coin or BTC coin, we will buy SOL-USDT or BTC-USDT into the portfolio by using all budgets. If the model forecast the price at t+1 will down and our portfolio have SOL coin or BTC coin, we will sell all SOL-USDT or BTC-USDT from the portfolio. If the actual value of portfolio at the current period is less than the stop loss, we will sell all SOL-USDT or BTC-USDT from the portfolio. The stop loss is calculated from total cost*(1-Stopper). Stopper is set at 15%. For the portfolio of multiple coins, we have set some assumptions to perform Backtesting. Total testing period is 1,236 hours. The first part (hour: 1-617) will use the models to trade only SOL. We will force the model to sell all SOL coins, if the model holds them in our portfolio at hour 618. The last part (hour: 619-1,236) will use the models to trade only BTC. The summary of trading strategy is depicted as below.



```
If (Actual price at t4 < Predicted price at t5) and (n_sol <= 0):
    Buy SOLUSDT (All balance)
Else if (Actual price at t4 > Predicted price) and (n_sol > 0):
    If (Actual price at t4 > latest buy price):
        Sell SOLUSDT (All shares)
    Else if (n_sol*(1-fee)* Actual price at t4) < (total cost*(1-stopper)):
        Sell SOLUSDT (All shares)
```

Buy Signal: Model forecast the price at t+1 will up.

Sell Signal: Model forecast the price at t+1 will down.

Stopped loss: The actual value of portfolio at the current time is less than total cost*(1-Stopper).

Fig 31: Detail illustration of our customized trading strategy

For the single coin of portfolio, RNN is the only model that can generate a profit based on our customized trading strategy during a volatile market. For the multiple coins of portfolio, RNN is the model that can generate the highest profit based on our customized trading strategy during a volatile market. We summary the result of two portfolios and the detail of executing transactions as below.

Items	RNN	LSTM	GRU	LightGBM	MA
Initial budget	**1,000**	1,000	1,000	1,000	1,000
Maker/Taker	**0.1%/0.1%**	0.1%/0.1%	0.1%/0.1%	0.1%/0.1%	0.1%/0.1%
Period (hourly)	**9-Apr-22 20:00 – 30-Apr-22 07:00**	9-Apr-22 20:00 – 30-Apr-22 07:00	9-Apr-22 20:00 – 30-Apr-22 07:00	9-Apr-22 20:00 – 30-Apr-22 07:00	9-Apr-22 20:00 – 30-Apr-22 07:00
Total hours	**1,236**	1,236	1,236	1,236	1,236
Total Cost at the end	**1,252**	1,040	1,079	1,120	1,040
Total Portfolio value at the end	**1,069**	906	940	986	925
Unrealized Profit	69	-94	-60	-14	-75
% Profit	7%	-9%	-6%	-1%	-7%

Table 3: Model result for single coin portfolio

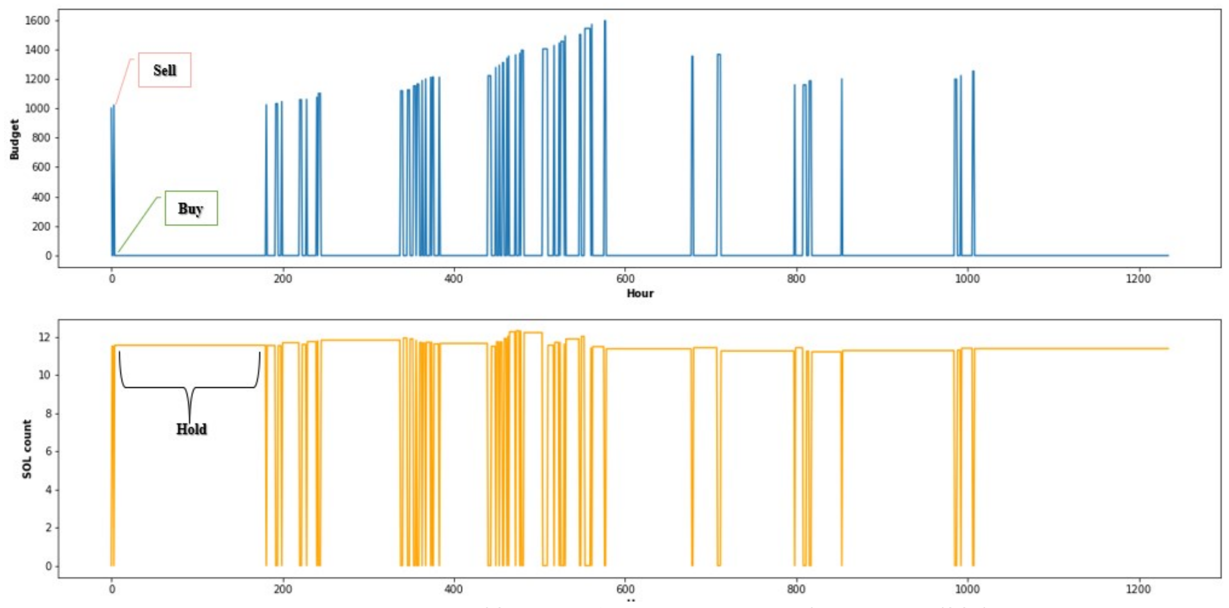


Fig 32: The detail of SimpleRNN 's executing transactions for single coins (SOL)

Items	RNN	LSTM	GRU	LightGBM	MA
Initial budget	**1,000**	1,000	1,000	1,000	1,000
Maker/Taker	**0.1%/0.1%**	0.1%/0.1%	0.1%/0.1%	0.1%/0.1%	0.1%/0.1%
Period (hourly)	**9-Apr-22 20:00 – 30-Apr-22 07:00**	9-Apr-22 20:00 – 30-Apr-22 07:00	9-Apr-22 20:00 – 30-Apr-22 07:00	9-Apr-22 20:00 – 30-Apr-22 07:00	9-Apr-22 20:00 – 30-Apr-22 07:00
Total hours	**1,236**	1,236	1,236	1,236	1,236
Total Cost at the end	**1,280**	1,177	1,187	1,239	1,148
Total Portfolio value at the end	**1,162**	1,078	1,088	1,135	1,043
Unrealized Profit	**162**	78	88	135	43
% Profit	**16%**	8%	9%	14%	4%

Table 4: Model result for multiple coins portfolio

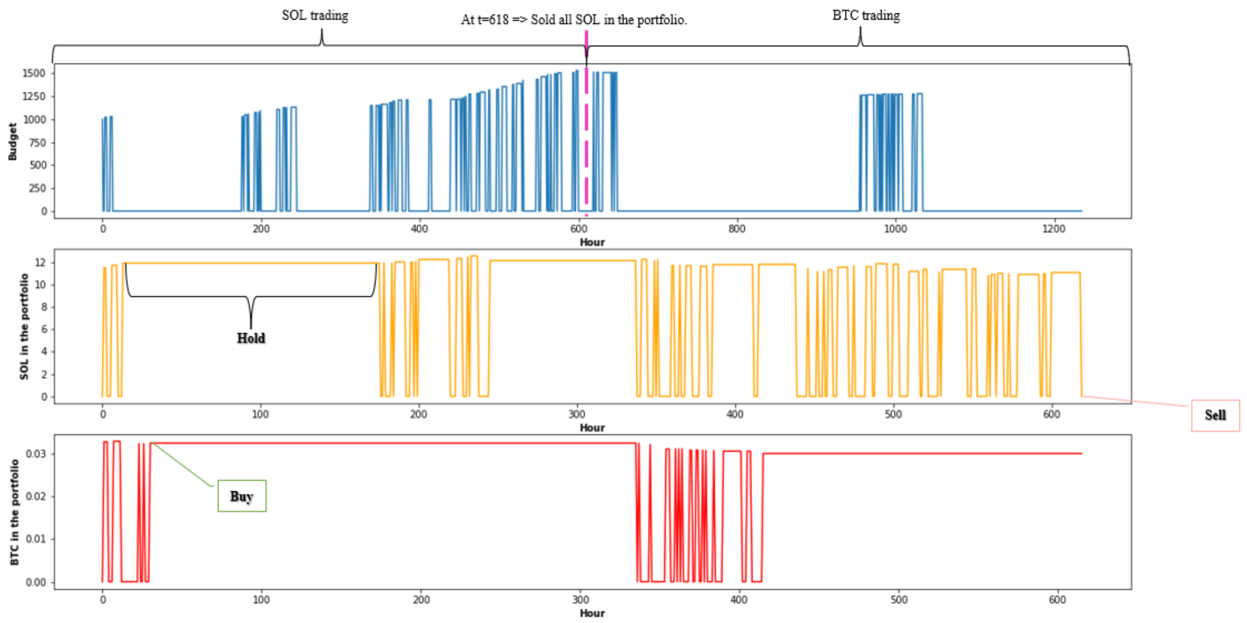


Fig 33: The detail of SimpleRNN 's executing transactions for multiple coins (SOL and BTC)

6. Conclusion

The main objective of this study is to identify the performance of models that can efficiently forecast cryptocurrency price movement among deep learning, traditional machine learning and traditional indicator and to test on which models among deep learning, traditional machine learning and traditional indicator can generate the highest profit under same algorithm investment strategy. From the empirical result, we find that RNN has the best performance when compared with the result of other models. We find that RNN can generate the maximum profit when compared with other models under the same algorithm investment strategy for single coin and multiple coins. If we use RNN for trading multiple coins, it can provide more profitable than trading single coin. So, Investor can use this algorithm investment strategy as a supporting tool for automated trading to help them to make more profit under uncertain market.

For further improvement, we'd like to suggest to add other features such as trading volume, number of trades and so on as independent variables for improving the model prediction. Try to change the architecture of deep learning model for improving the model prediction. Try to use other technical analysis for seeking the signal (buying signal, selling signal and holding signal) such as Directional Movement System (DMS) to make more the profit.

7. Acknowledgement

I would like to express my appreciation to Dr. Ekarat Rattagan for his valuable and constructive suggestions during the planning and development of this research work, all professor at national institute of development administration (NIDA) for academic knowledge and various valuable recommendations, my family for all support which contributes this research work.

References

1. Atsalakis, S.G.; Atsalaki, G.I.; Pasiouras, F. and Zopounidis, C., " Bitcoin price forecasting with neuro-fuzzy techniques", European Journal of Operational Research, 2019, vol. 276, pp. 770-780.
2. Chan, Ernest P., " Algorithmic Trading: Winning Strategies and Their Rationale (Wiley Trading Series)", John Wiley & Sons, 2013. Print.
3. Cho, K.; Merriënboer, B. V.; Gulcehre, C.; Bahdanau, D.; Schwenk, H. and Bengio, Y., " Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724–1734.
4. Colah blog, “Understanding LSTM Networks”, 2015. [online] Available at: < https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
5. Eurekahedge, “Eurekahedge Research: Artificial Intelligence: The New Frontier for Hedge Funds”, 2018. [online] Available at: < https://www.eurekahedge.com/Research/News/1724/Artificial-Intelligence-Hedge-Fund-Index-Strategy-Profile-2-February-2018>.
6. Factor Research, “Hedge Fund Battle: Discretionary vs Systematic Investing”, 2020. [online] Available at: <https://www.factorresearch.com/research-hedge-fund-battle-discretionary-vs-systematic-investing>.
7. Financial Times, “Real’ investors eclipsed by fast trading”, 2012. [online] Available at: <https://www.ft.com/cms/s/0/da5d033c8e1c11e1bf8f00144feab49a.html>.
8. Friedman B. Prequin Blog, “The Rise of the Machines: AI Funds Are Outperforming the Hedge Fund Benchmark”, 2019. [online] Available at: < https://www.preqin.com/insights/research/blogs/the-rise-of-the-machines-ai-funds-are-outperforming-the-hedge-fund-benchmark>.
9. Hamayel, M.J. and Owda, A.Y., "A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms", AI, 2021, vol. 2, pp. 477-496.
10. GeeksforGeeks.org, “LightGBM (Light Gradient Boosting Machine)”, 2021. [online] Available at: <https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/>.
11. Goodfellow I.; Bengio Y.; Courville A.; " Deep learning", MIT Press, 2016.
12. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q. and Liu, T. -Y., " LightGBM: A Highly Efficient Gradient Boosting Decision Tree", Advances in Neural Information Processing Systems, 2017, vol. 30, pp. 3146–3154.
13. Micha´nków, J.; Sakowski, P. and ´Slepaczuk, R., " LSTM in Algorithmic Investment Strategies on BTC and S&P500 Index" Sensors, 2022, vol. 22, pp. 917.
14. Ning, L., " A Machine Learning Approach to Automated Trading", Boston College Computer Science Senior Thesis, 2016.
15. Wu, C. -H.; Lu, C. -C.; Ma, Y. -F. and Lu, R. -S., " A New Forecasting Framework for Bitcoin Price with LSTM", 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 2018, pp. 168-175.