Machine Learning for Automated Cryptocurrency Trading

Advisee: Mathee Prasertkijaphan 6310422053
Advisor: Assistant Professor Dr. Ekarat Rattagan

Agenda

- Introduction
- Literature Review
- Theoretical Frameworks
- Data
- Methodology
- Empirical result
- Conclusion

Introduction

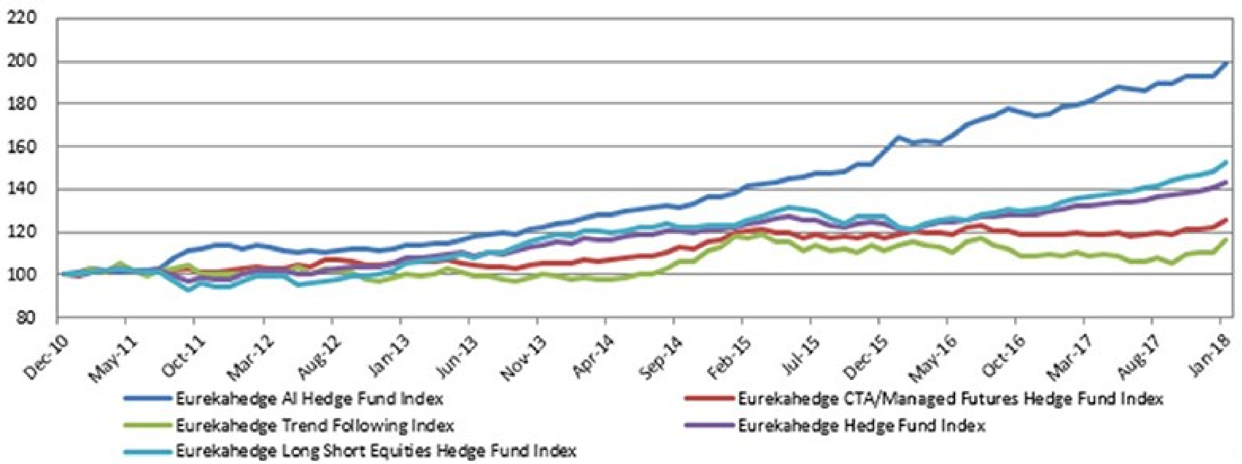
As of 2012, a report from Morgan Stanley showed that 84% of all stock trades in the U.S. Stock Market were done by computer algorithm, while only 16% were by human investors.

AI / Machine Learning Hedge Fund Returns During Key Market Risk Events

Date	Event	AI Hedge Fund Index	CTA/ Managed Futures	Trend Following Index	Hedge Fund Index
Mar-21	**Covid19 Drawdown**	**3.27%**	**1.82%**	**4.10%**	**-2.23%**
Nov-16	Trump Win	-0.94%	-0.18%	0.38%	0.31%
Jun-16	Brexit	1 29%	2.32%	4.18%	0.32%
Feb-16	Oil Price Dip/China growth concerns	-0.86%	1.71%	1.92%	0.00%
Jan-16	Oil Price Dip/China growth concerns	4.33%	1.33%	2.42%	-1.74%
Aug-15	China Equity Crash	0.72%	-1.72%	-2.54%	-1.92%
Jul-15	China Equity Crash	0.43%	0.97%	-2.25%	-0.06%
Jun-15	Greek referendum	1.84%	-2.00%	-3.28%	-1.14%
Jan-15	Swiss Franc De-pegging	1.30%	3.23%	3.87%	0.78%
Sep-14	Oil Price Dip	-0.57%	198.00%	333.00%	-0.22%
Jun-13	Taper Tantrum	1.56%	-0.93%	-1.68%	-1.31%
May-13	Taper Tantrum	1.55%	-1.41%	-1.07%	0.45%

Source: Adapted from (Eurakahedge 2017) to include Covid19 risk assessment (https://europepmc.org/article/PPR/PPR306564).

Long-Term Analysis - AI vs Quants vs Traditional Hedge Fund Indices



Source: Eurekahedge Database (Eurekahedge, 2018). (https://europepmc.org/article/PPR/PPR306564)

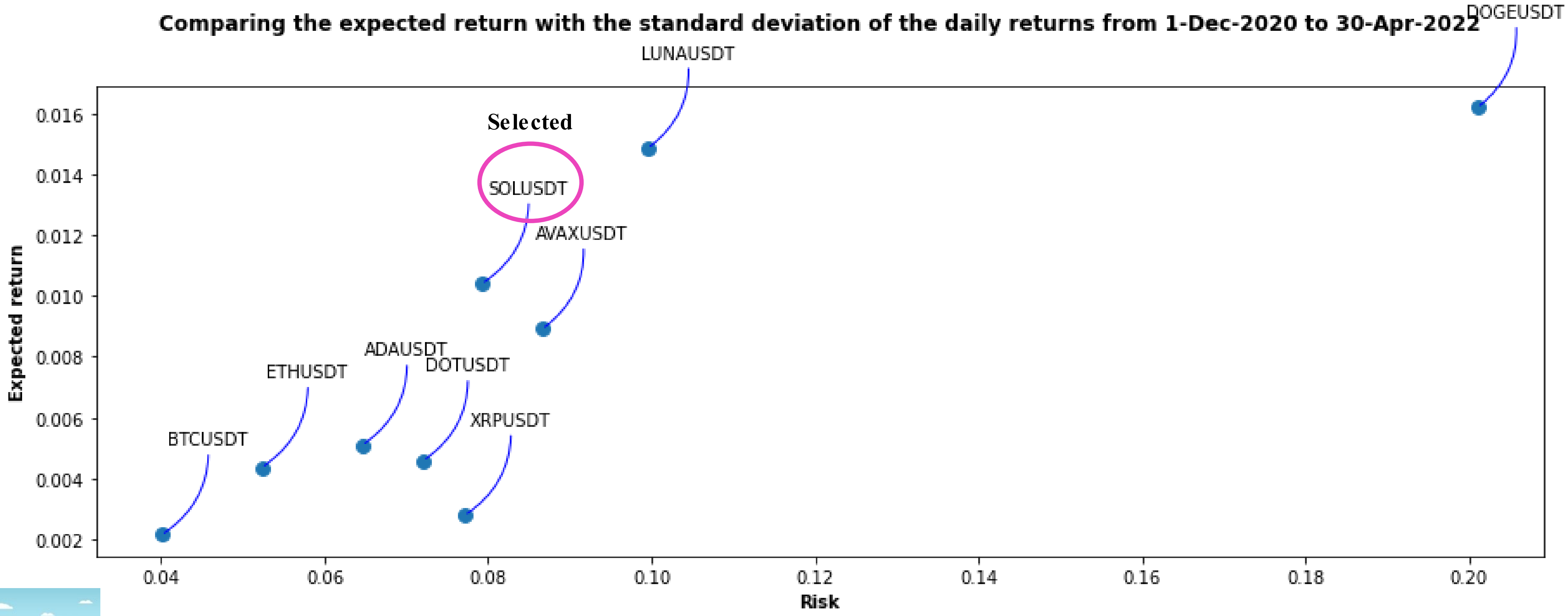
Introduction

Items	Cryptocurrency
Definition	Cryptocurrency is the digital currency or exchange medium which can be used to purchase goods and services. U.S. dollars can be swapped to purchase cryptocurrency in the same way they can be swapped to purchase arcade tokens or casino chips. Cryptocurrency transactions are maintained and safeguarded by encryption in a public ledger utilizing blockchain technology.
Exchange	Decentralized crypto exchange (DEX) and Centralized crypto exchange (CEX)
CEX	Binance, FTX, Coinbase Exchange, Kraken, KuCoin, Gate.io and so on.
DEX	Uniswap (V3), dYdX, PancakeSwap, Astroport, Apoloox DEX, Curve Finance and so on
Trading Time	24/7

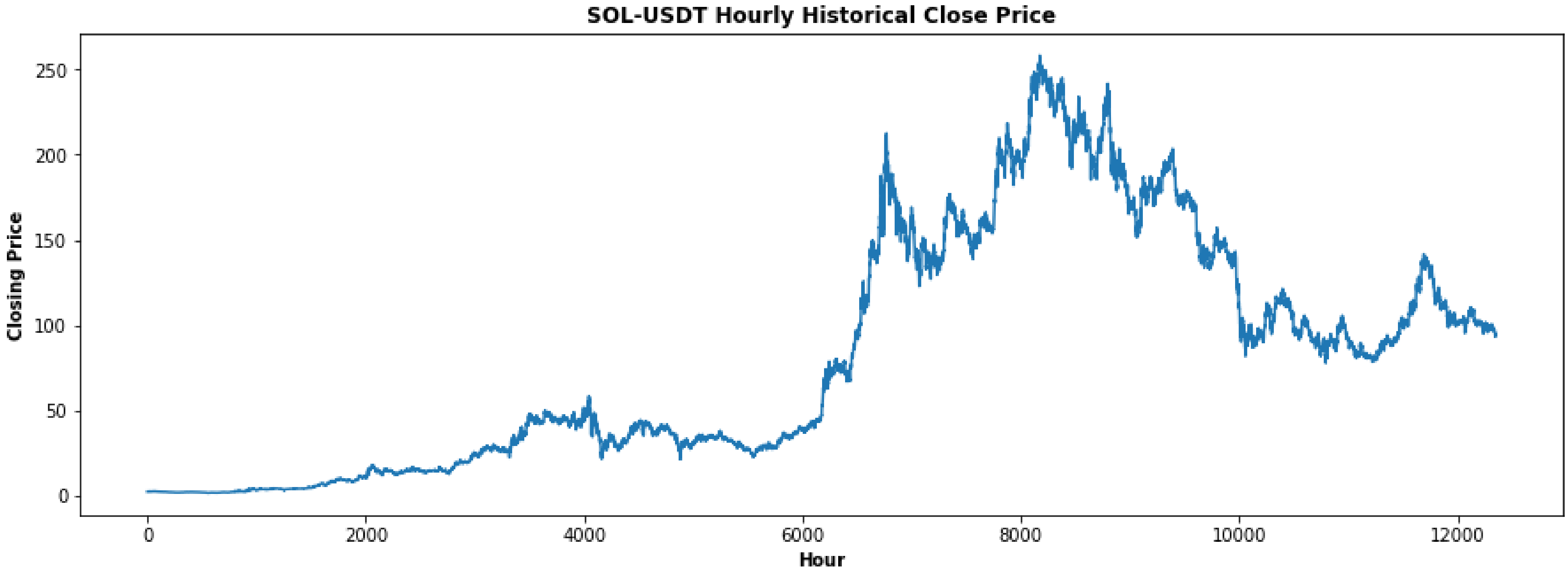
Noted: CEX and DEX are sorted by the size of market share (coinmarketcap)

Introduction

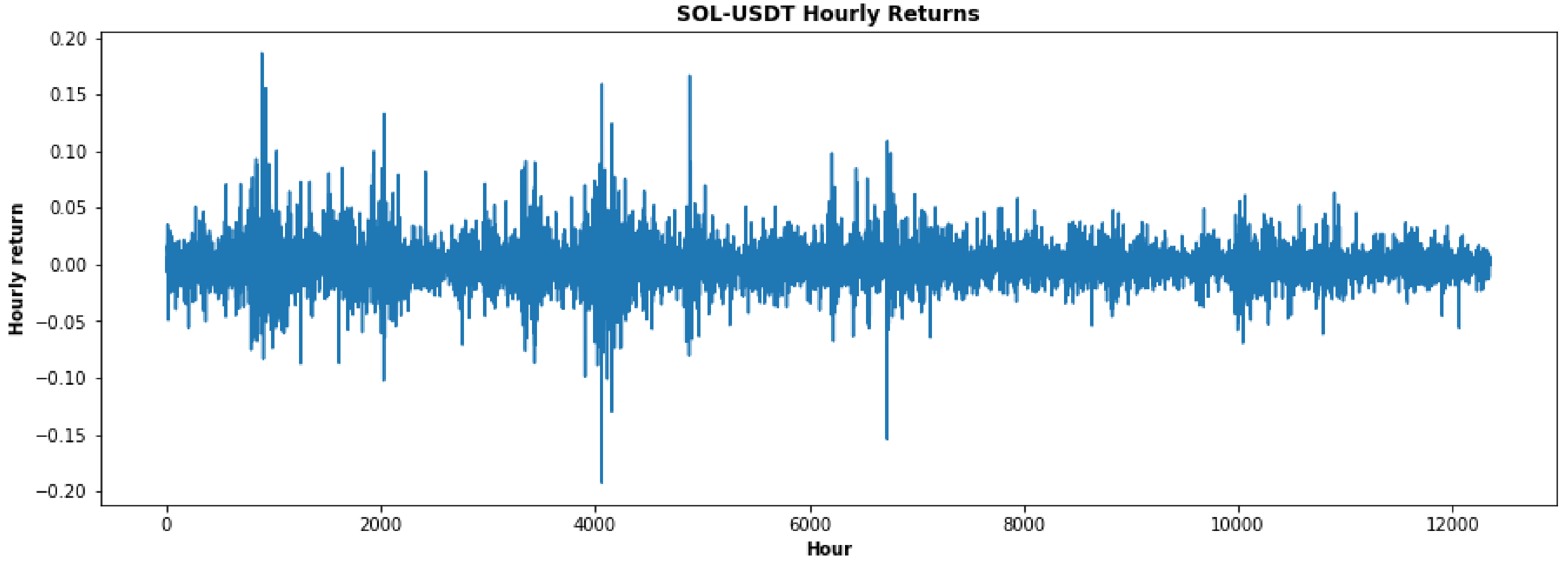
Top 10 of cryptocurrency by Market Capitalization from CoinMarketCap

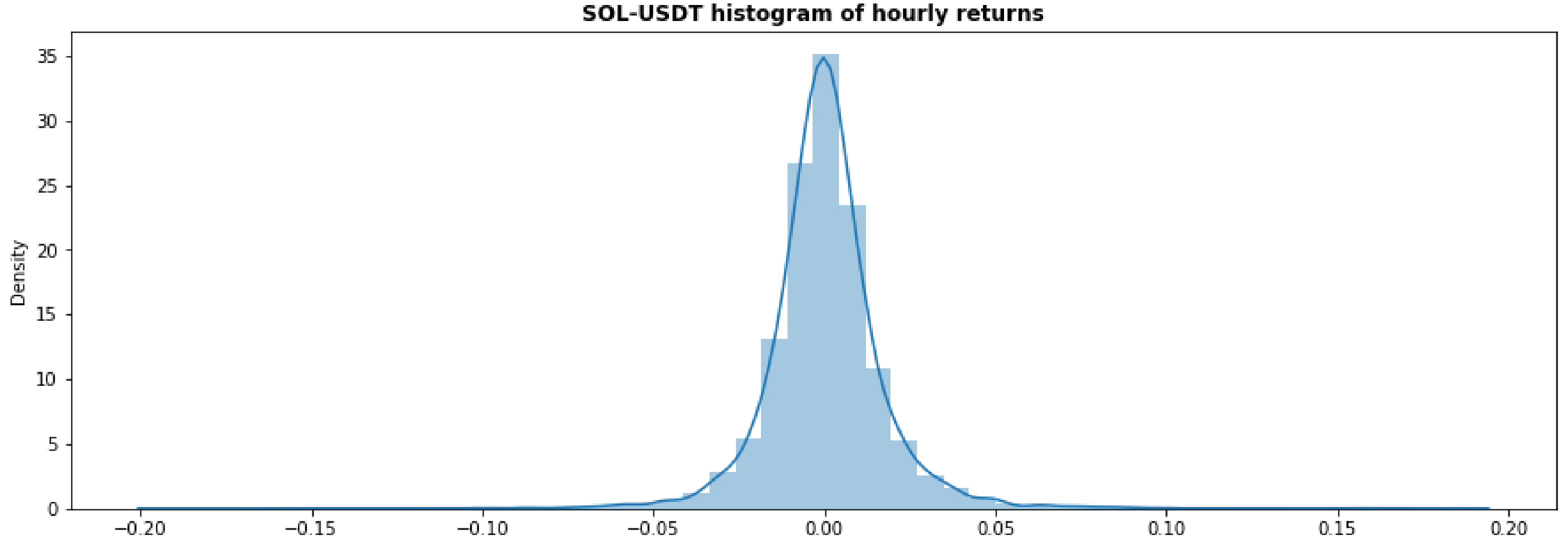


Introduction

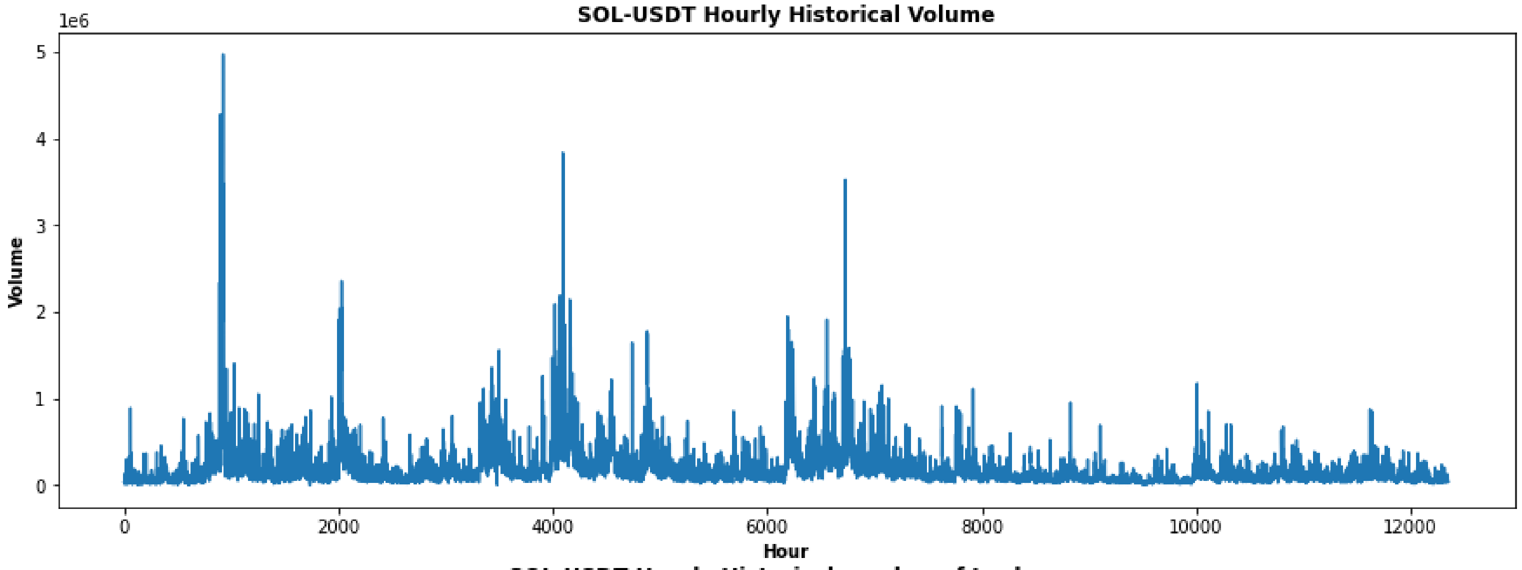


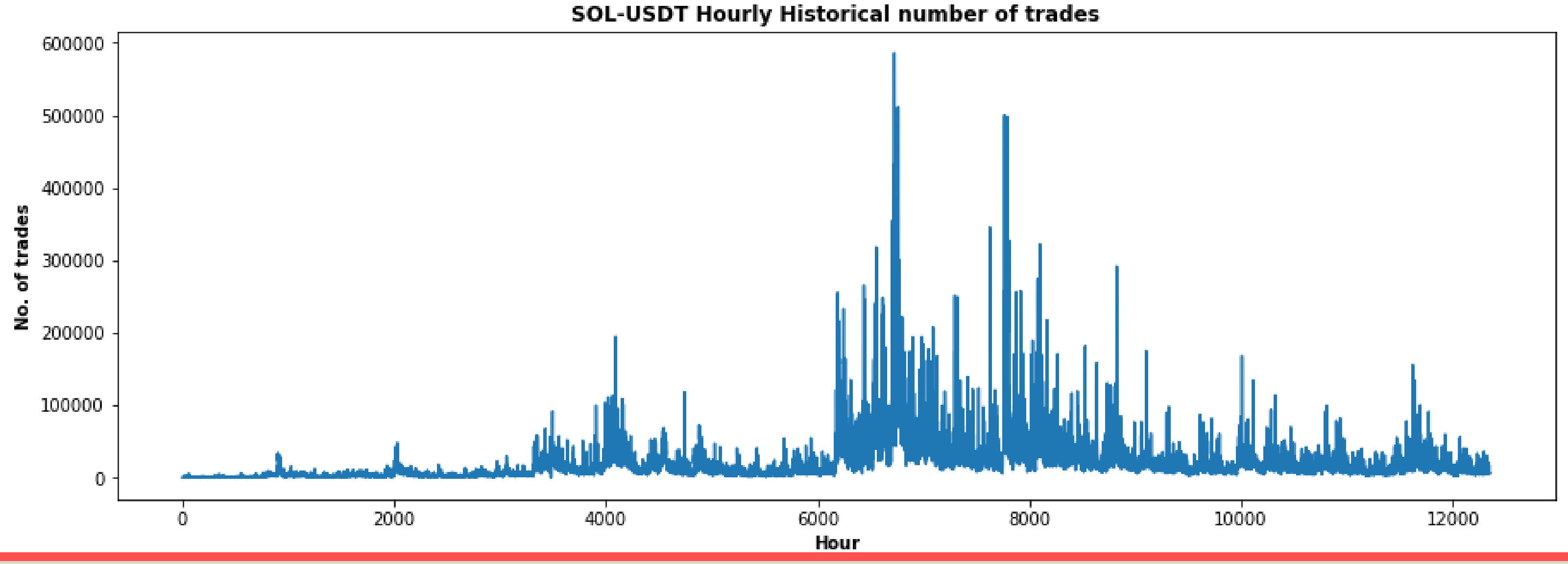
Introduction





Introduction





Motivation, Objective, Research Question and Contribution

Motivations

- To study the performance of which models among deep learning, traditional machine learning and traditional indicator for forecasting are the best models.
- To study which models among deep learning, traditional machine learning and traditional indicator can generate the highest profit .

Objectives

- Identify the performance of models that can efficiently forecast cryptocurrency price movement among deep learning, traditional machine learning and traditional indicator
- Test on which models among deep learning, traditional machine learning and traditional indicator can generate the highest profit under same algorithm.

Motivation, Objective, Research Question and Contribution

Research Questions

- Can we use the model to forecast cryptocurrency price movement?
- Which models can provide the highest profit from automated trading?

Contributions

- Providing the best performance models for predicting the cryptocurrency price movement.
- Providing more appropriate automated trading strategy to generate the highest profit from the model.

Agenda

- Introduction
- Literature Review
- Theoretical Frameworks
- Data
- Methodology
- Empirical result
- Conclusion

Literature Review: Summary

<table>
<tr><th>Paper</th><th>Methodology</th><th>Models</th><th>Result</th></tr>
<tr>
<td>Ning Lu (2016)</td>
<td>

Individual Approach (N=4)

<table>
<tr><th>X</th><th>Y</th></tr>
<tr><td>R_{A1} R_{A2} R_{A3} … R_{AN}</td><td>Y_{AN+1}</td></tr>
<tr><td>R_{A2} R_{A3} R_{A4} … R_{AN+1}</td><td>Y_{AN+2}</td></tr>
</table>

Sector Approach (N=4)

<table>
<tr><th>X</th><th>Y</th></tr>
<tr><td>R_{A1} R_{A2} R_{A3} R_{A4} R_{B1} R_{B2} R_{B3} R_{B4} … R_{Z1} R_{Z2} R_{Z3} R_{Z4}</td><td>Y_{A5}</td></tr>
<tr><td>R_{A2} R_{A3} R_{A4} R_{A5} R_{B2} R_{B3} R_{B4} R_{B5} … R_{Z2} R_{Z3} R_{Z4} R_{Z5}</td><td>Y_{A6}</td></tr>
</table>
</td>
<td>

Individual Approach
Ridge logistic regression
Sector Approach

Lasso logistic regression
Decision tree
Naïve bayes
Support vector machine
2 ensemble methods: Majority Vote and Random Subset

</td>
<td>

His trading systems achieved promising Sharpe ratios with nearly 20 percent return, after transaction cost, during the out-of-sample testing period.
His trading systems was able to beat risk-free interest rate and some even outperformed the S&P 500.

</td>
</tr>
<tr>
<td>J. Michankow, P.Sakowski and R. Slepaczuk (2022)</td>
<td>

Portfolio A: BTC combined frequencies (1 d, 1 hour, and 15 min)
Portfolio B: S&P500 combined frequencies (1 d, 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

</td>
<td>

LSTM

<table>
<tr><th>Hyperparameter</th><th>Selected Value</th></tr>
<tr><td>No. hidden layers</td><td>3</td></tr>
<tr><td>No. neurons</td><td>512/256/128</td></tr>
<tr><td>Activation function</td><td>tanh</td></tr>
<tr><td>Dropout rate</td><td>0.02</td></tr>
<tr><td>l2 regularizer</td><td>0.0005</td></tr>
<tr><td>Optimizer</td><td>Adam</td></tr>
<tr><td>Learning rate</td><td>0.00015</td></tr>
<tr><td>BTC train/test</td><td>1371/90</td></tr>
<tr><td>S&P train/test</td><td>948/65</td></tr>
<tr><td>Batch size</td><td>80</td></tr>
<tr><td>Sequence length</td><td>14/20</td></tr>
</table>
</td>
<td>

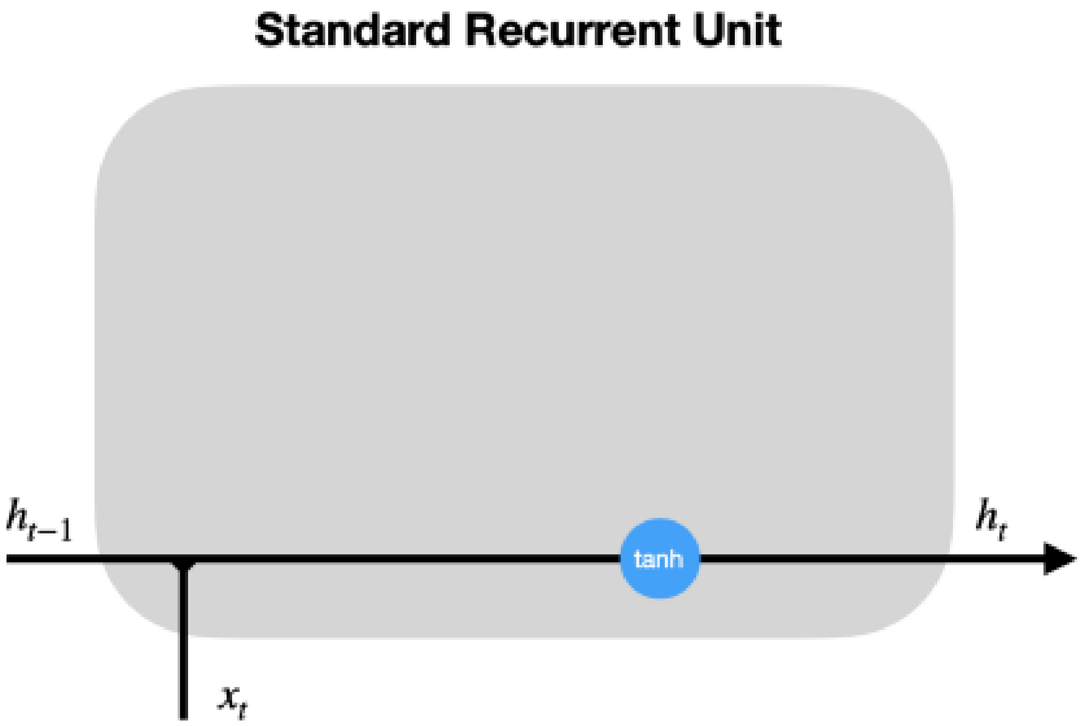
The combination of weights equal to {S&P500 = W20%, BTC = 80%} was always better than {S&P500 = W10%, BTC = W90%}.
The length of rebalancing period equal to RB6m was always better than RB3m

</td>
</tr>
</table>

Agenda

- Introduction
- Literature Review
- Theoretical Frameworks
- Data
- Methodology
- Empirical result
- Conclusion

SimpleRNN



h_{t-1} - hidden state at previous timestep t-1 (memory)

x_t - input vector at current timestep t

h_t - hidden state at current timestep t

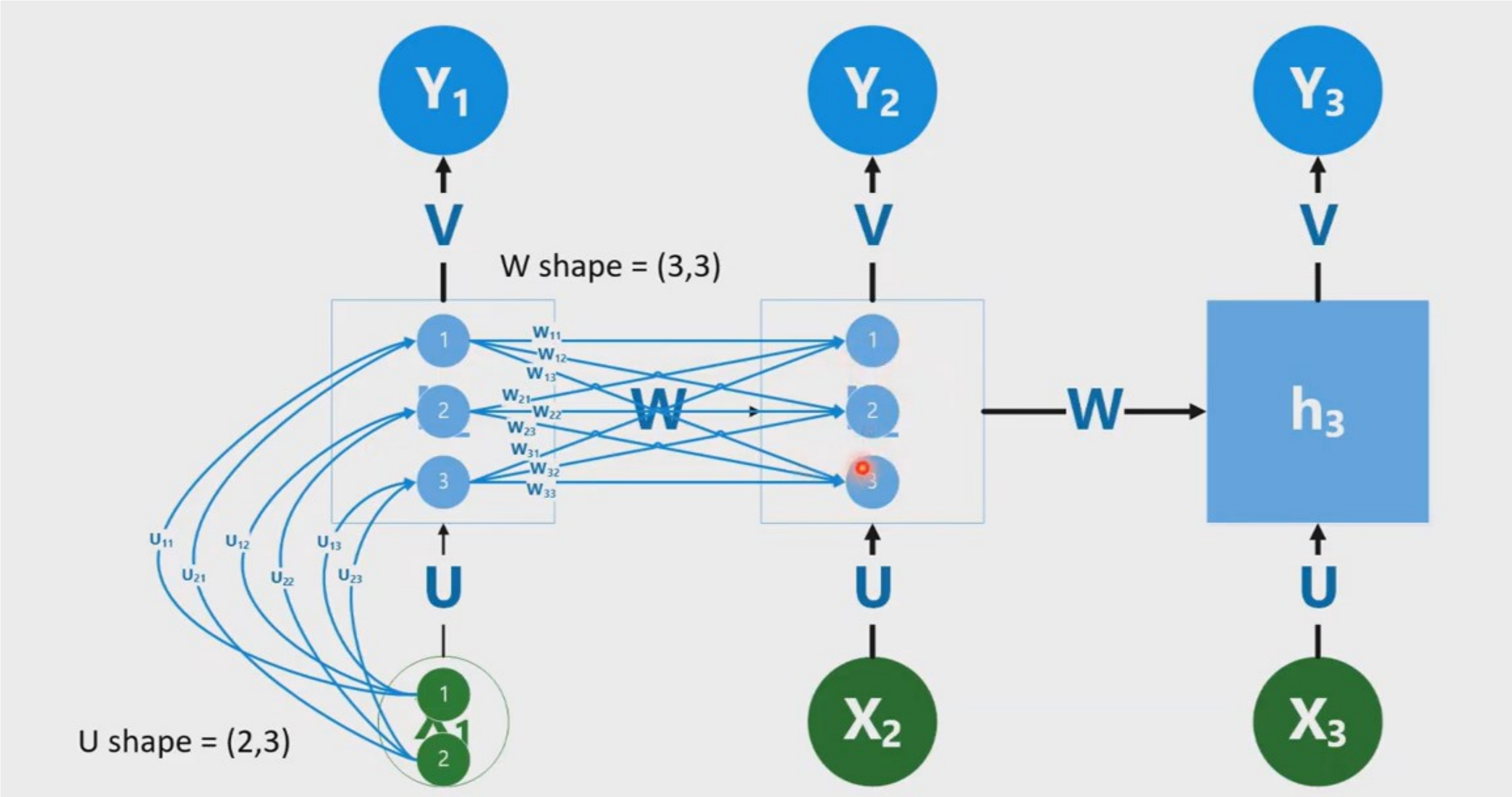
tanh - tanh activation function

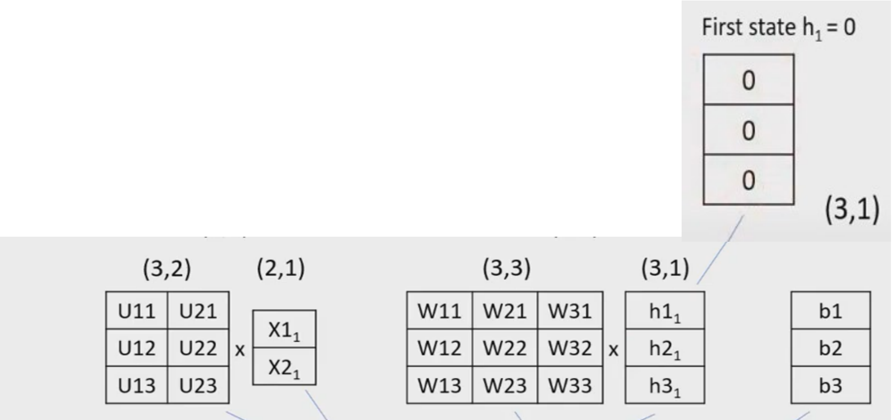
- concatenation of vectors

$$h_t = \sigma(w_{xh}^T x_t + w_{hh}^T h_{t-1} + b_h)$$
$$\hat{y}_t = \sigma(w_i^T h_t + b_i)$$

SimpleRNN

- Number of **hidden node** = 3
- Number of **feature input** =2
- Number of sequence (**time step**) =3





$$h_t = f_h(\quad UX_t \quad + \quad Wh_{t-1} \quad + b_h \quad)$$

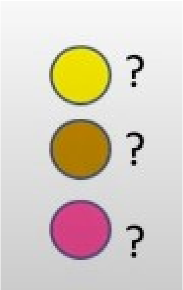
```
                   3                     3         2
model.add(SimpleRNN(n_hidden, input_shape=(n_sequence, n_features), return_sequences=True))
```

SimpleRNN

RNN Problem

- Given, A like to eat ice-cream. We want to predict which flavor of ice cream A will order today. From historical record, A’s flavor selection based on previous 2 days.

T-2	T-1	T
Strawberry	Vanilla	Chocolate
Chocolate	Vanilla	Strawberry
Vanilla	Vanilla	Vanilla



Vanilla

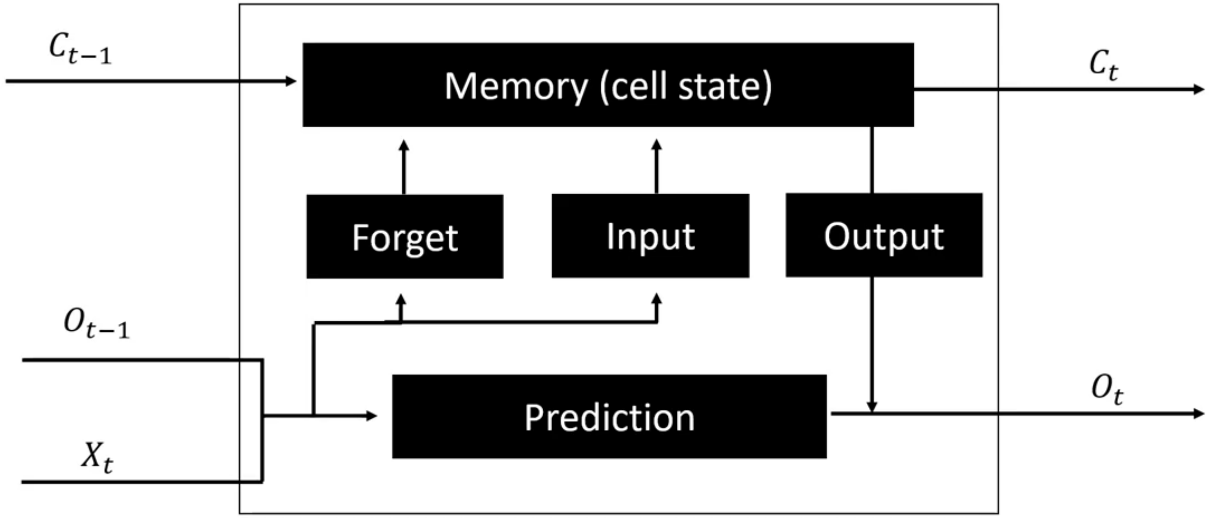
Chocolate

Strawberry

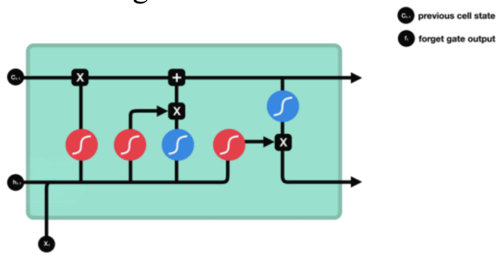
LSTM

LSTM is RNN with memory which can decide: Cell state

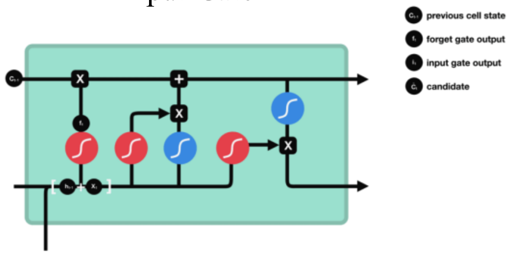
- What to forget: **Forgetting gate**
- What new information to add: **Input gate**
- When to let memory impact prediction: **Output gate**

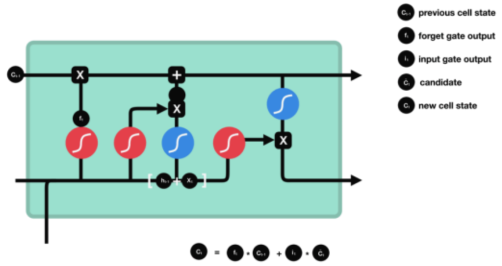


Forget Gate

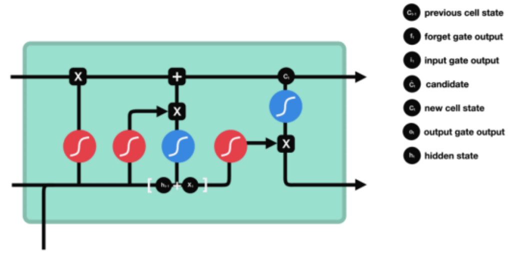


Input Gate



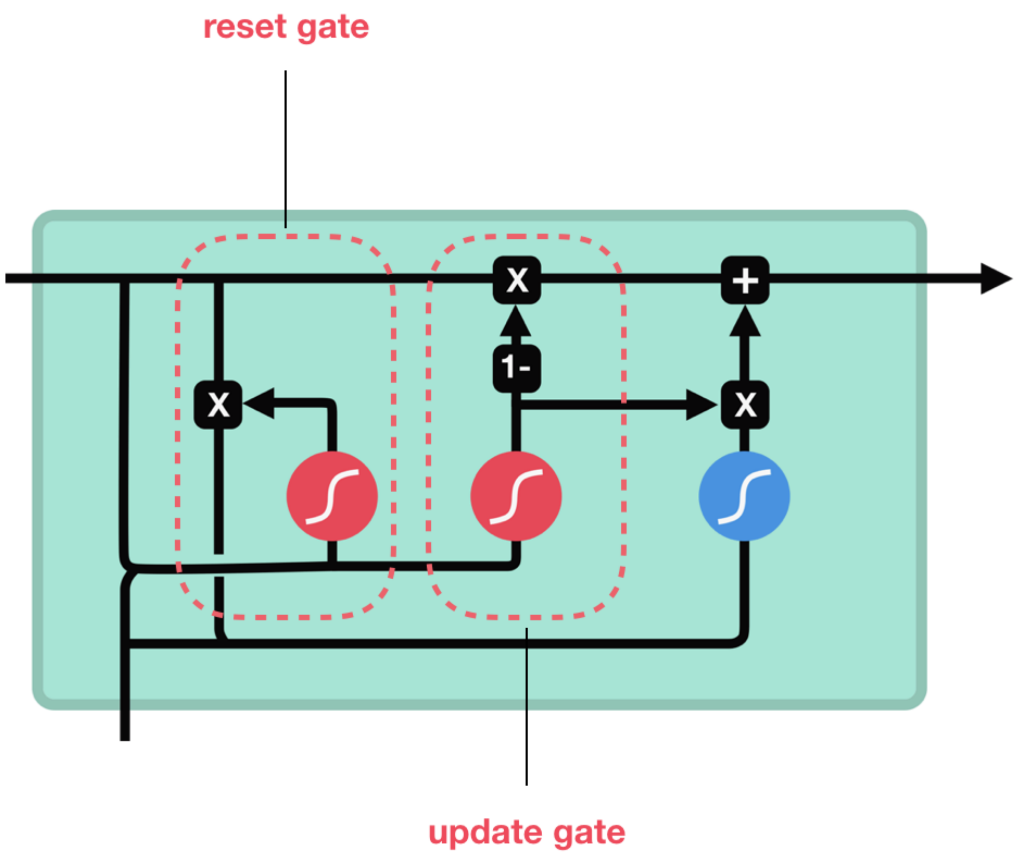


Calculating cell state



Output Gate

GRU

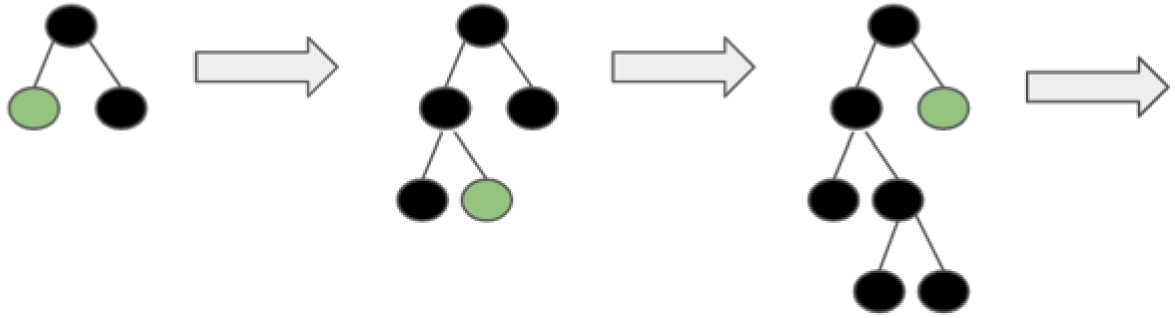


Reset Gate: It is another gate is used to decide how much past information to forget.

Update Gate: It acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.

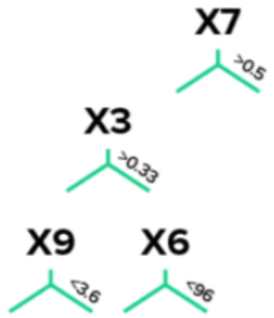
LightGBM

LightGBM leaf-wise (grows tree vertically):



- Leaf-wise (best-first) tree growth.
- Grow the leaf that minimizes the loss, allowing a growth of an imbalanced tree.
- Overfitting can happen when data is small because it doesn't grow level-wise
- Need to control the tree depth

Tree growth example:



Agenda

- Introduction
- Literature Review
- Theoretical Frameworks
- Data
- Methodology
- Empirical result
- Conclusion

Introduction

1. **SOL-USDT**

   ➢ They were obtained from Binance API. The period of the collected hourly data is from December 1, 2020 to April 30, 2022. The details of this cryptocurrency was collected as followed:

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

Data

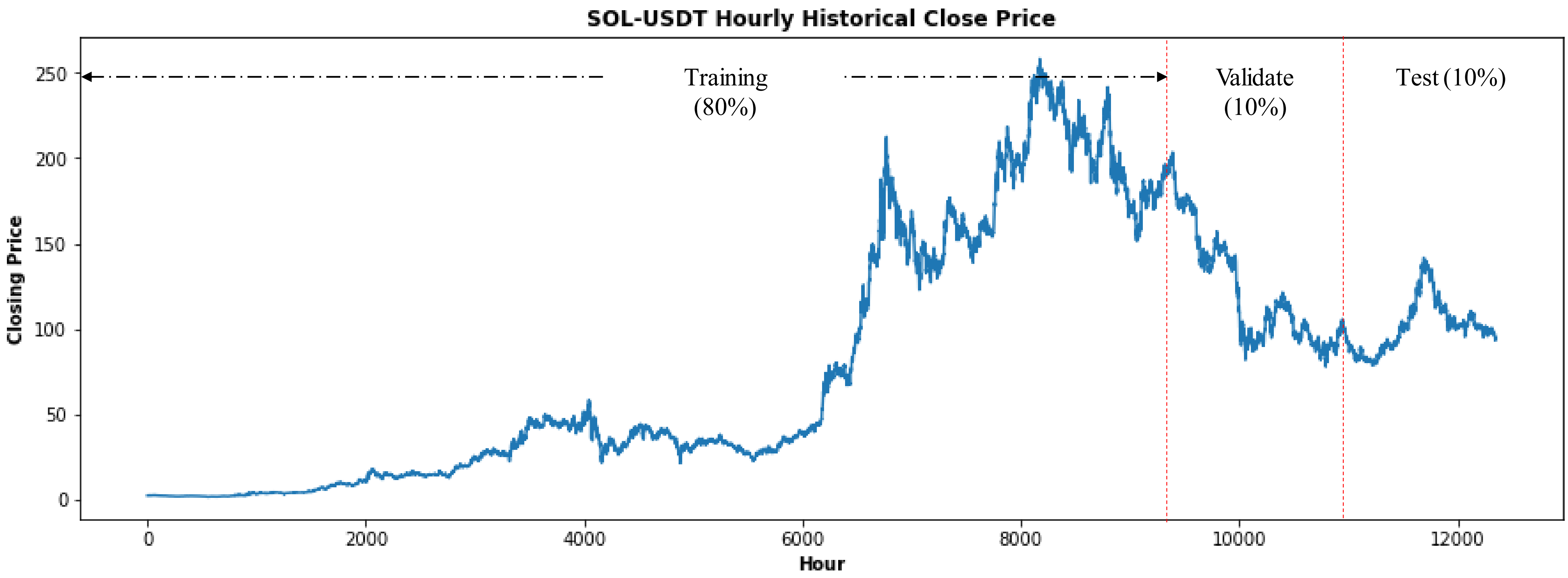
	symbol	date	close_price	volume	Number_of_trades	return
0	SOLUSDT	2020-12-01 00:00:00	1.9768	37873.34	543	0.000000
1	SOLUSDT	2020-12-01 01:00:00	1.9715	38888.52	512	-0.002681
2	SOLUSDT	2020-12-01 02:00:00	1.9680	43257.60	539	-0.001775
3	SOLUSDT	2020-12-01 03:00:00	1.9699	45466.77	677	0.000965
4	SOLUSDT	2020-12-01 04:00:00	1.9667	14399.31	427	-0.001624

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15025 entries, 0 to 15024
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   symbol            15025 non-null  object
 1   date              15025 non-null  datetime64[ns]
 2   close_price       15025 non-null  float64
 3   volume            15025 non-null  float64
 4   Number_of_trades  15025 non-null  int64
 5   return            15025 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(1)
memory usage: 821.7+ KB
```

	symbol	date	close_price	volume	Number_of_trades	return
12346	SOLUSDT	2022-04-30 03:00:00	93.69	72027.87	8820	0.008178
12347	SOLUSDT	2022-04-30 04:00:00	93.83	49799.06	7552	0.001494
12348	SOLUSDT	2022-04-30 05:00:00	94.07	37230.90	5694	0.002558
12349	SOLUSDT	2022-04-30 06:00:00	93.96	53316.48	5572	-0.001169
12350	SOLUSDT	2022-04-30 07:00:00	94.52	42360.98	6543	0.005960

	close_price	volume	Number_of_trades	return
count	12,351.00000	12,351.00000	12,351.00000	12,351.00000
mean	81.75690	182,837.80000	20,989.21000	0.00046
std	69.73758	222,052.90000	31,649.80000	0.01707
min	1.18320	0.00000	0.00000	(0.19284)
25%	24.21255	68,304.95000	5,167.50000	(0.00777)
50%	54.45900	116,192.40000	11,676.00000	0.00000
75%	139.17000	210,888.60000	24,559.50000	0.00785
max	258.44000	4,974,114.00000	585,919.00000	0.18646
var	4,863.32934	49,307,480,000.00000	1,001,710,000.00000	0.00029
skew	0.61299	5.74427	5.54499	0.54876
kurt	(0.84430)	65.05483	51.69817	10.43626

Data



SOL-USDT Hourly Historical Close Price
Training
(80%)
Validate
(10%)
Test (10%)
Closing Price
Hour
250
200
150
100
50
0
0
2000
4000
6000
8000
10000
12000

Agenda

- Introduction
- Literature Review
- Theoretical Frameworks
- Data
- Methodology
- Empirical result
- Conclusion

Problem statement (with notation and definition)

➢ **Model:**

$$Y_4 = [R_1, R_2, R_3]$$

- 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 use 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).

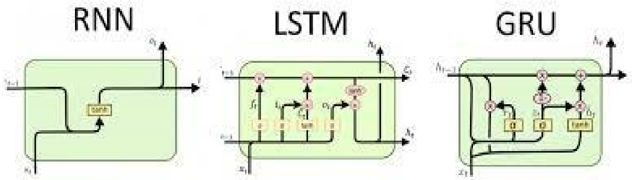
Process design

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



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

Data pre-processing

➤ **Create Sequence return feature from hourly closing price by using window size (N) =4:**

X	Y
$R_0 R_1 R_2 \ldots R_N$	R_{N+1}

- R_0 is the return of SOL-USDT from day 0 to day 1.

- R_{N+1} is the predicted return of SOL-USDT from day N to day N+1.

```
#Create Sequence feature
window_size=4
x_close,x_return,y_close,y_return = window_return(df_trans["close_price"].values, window_size)
print(f"x_close.shape : {x_close.shape}")
print(f"x_return.shape : {x_return.shape}")
print(f"y_close.shape : {y_close.shape}")
print(f"y_return.shape : {y_return.shape}")
```

```
x_close.shape : (12346, 4)
x_return.shape : (12346, 3)
y_close.shape : (12346, 1)
y_return.shape : (12346, 1)
```

```
# Split close price and return
def window_return(val, window):
  lst_x = []
  lst_y = []
  lst_x_return = []
  lst_y_return = []
  for i in range(0,len(val) - window -1,1):
    first_price = val[i]
    lst_x.append(val[i:i+window])
    lst_y.append(val[i+window])
    lst_x_return.append((val[i+1:i+window]-first_price)/first_price)
    lst_y_return.append((val[i+window]-first_price)/first_price)
  array_x_return = np.array(lst_x_return).reshape(-1,window-1)
  array_y_return = np.array(lst_y_return).reshape(-1,1)
  array_x = np.array(lst_x).reshape(-1,window)
  array_y = np.array(lst_y).reshape(-1,1)
  return array_x,array_x_return,array_y,array_y_return
```

Data pre-processing

➢ **Train/Test/Validation_split:**

```
# Split data tuples x-(train, val, test), y-(train, val, test)
lst_x_close ,lst_y_close = split(x_close, y_close, 0.8, 0.1)
lst_x_return ,lst_y_return = split(x_return, y_return, 0.8, 0.1)
```

```
print(f"Shape Price:{lst_x_close[0].shape} ,Val:{lst_x_close[1].shape}, Test:{lst_x_close[2].shape}")
print(f"Shape Return:{lst_x_return[0].shape} ,Val:{lst_x_return[1].shape}, Test:{lst_x_return[2].shape}")
```

```
Shape Price:(9876, 4) ,Val:(1236, 4), Test:(1234, 4)
Shape Return:(9876, 3) ,Val:(1236, 3), Test:(1234, 3)
```

```
print(f"Shape Price:{lst_y_close[0].shape} ,Val:{lst_y_close[1].shape}, Test:{lst_y_close[2].shape}")
print(f"Shape Return:{lst_y_return[0].shape} ,Val:{lst_y_return[1].shape}, Test:{lst_y_return[2].shape}")
```

```
Shape Price:(9876, 1) ,Val:(1236, 1), Test:(1234, 1)
Shape Return:(9876, 1) ,Val:(1236, 1), Test:(1234, 1)
```

```
def split(x_array, y_array, ratio_train=0.8, ratio_test=0.1):
  n_data = x_array.shape[0]
  n_train = int(n_data * ratio_train)
  n_test = int(n_data *ratio_test)
  n_val = n_data -n_train -n_test

  x_train, y_train = x_array[:n_train], y_array[:n_train]
  x_val, y_val      = x_array[n_train:n_train+n_val], y_array[n_train:n_train+n_val]
  x_test, y_test   = x_array[n_train+n_val:], y_array[n_train+n_val:]
  return [x_train, x_val, x_test], [y_train, y_val, y_test]
```

Data pre-processing

➢ **Data normalization:**

```
#Normalize Data form train data set
#Fit Normalize tools
min_max_scaler = MinMaxScaler().fit(lst_x_return[0].reshape(-1,1))
lst_x_norm   ,lst_y_norm= norm_val(min_max_scaler, lst_x_return, lst_y_return)
print(f"Shape x:{lst_x_norm[0].shape} ,Val:{lst_x_norm[1].shape}, Test:{lst_x_norm[2].shape}")
print(f"Shape y:{lst_y_norm[0].shape} ,Val:{lst_y_norm[1].shape}, Test:{lst_y_norm[2].shape}")
```

```
Shape x:(9876, 3) ,Val:(1236, 3), Test:(1234, 3)
Shape y:(9876, 1) ,Val:(1236, 1), Test:(1234, 1)
```

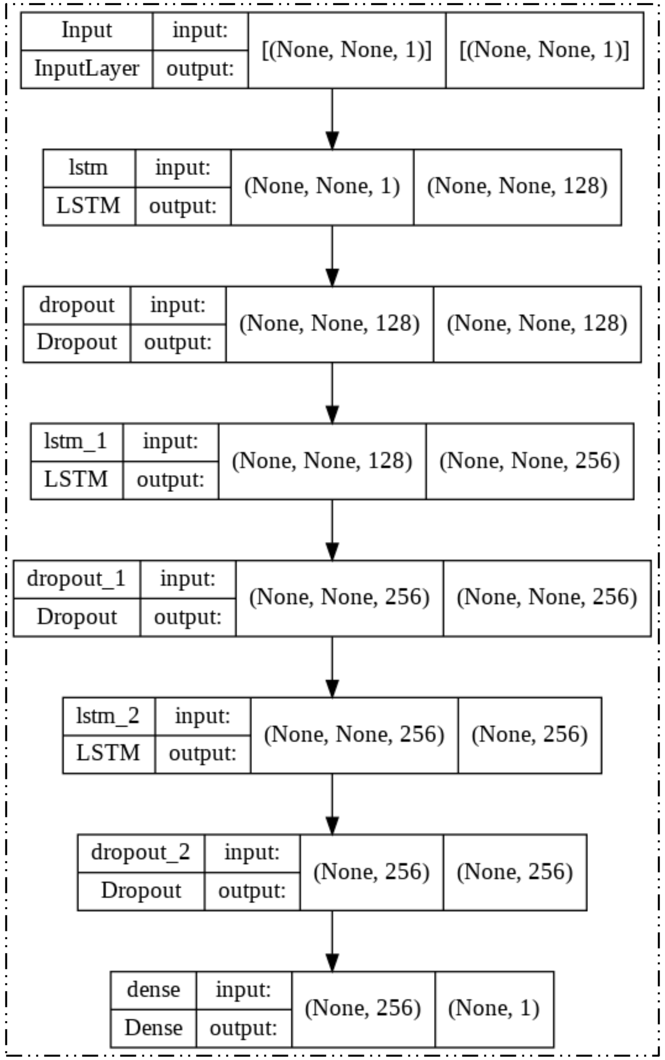
```
#Create data train - validate - test ( Size , Time , Feature)
#Create x array
x_train_norm = lst_x_norm[0]
x_val_norm   = lst_x_norm[1]
x_test_norm  = lst_x_norm[2]

#Create y array
y_train_norm = lst_y_norm[0]
y_val_norm   = lst_y_norm[1]
y_test_norm  = lst_y_norm[2]
y_test = lst_y_close[2]

print(f"\n x_train_norm.shape {x_train_norm.shape}{y_train_norm.shape} \n{x_train_norm[0]} -->y= {y_train_norm[0]}")
print(f"\n x_val_norm.shape {x_val_norm.shape}{y_val_norm.shape} \n{x_val_norm[0]} -->y={y_val_norm[0]}")
print(f"\n x_test_norm.shape {x_test_norm.shape}{y_test_norm.shape}\n{x_test_norm[0]} -->y={y_test_norm[0]}")
```

```
 x_train_norm.shape (9876, 3)(9876, 1)
[0.43412612 0.43116426 0.43277213] -->y= [0.43006414]

 x_val_norm.shape (1236, 3)(1236, 1)
[0.44042191 0.42718119 0.4174487 ] -->y=[0.40805571]

 x_test_norm.shape (1234, 3)(1234, 1)
[0.42936787 0.4223882  0.40427878] -->y=[0.42201092]
```

```
def norm_val(norm_fn, lst_x, lst_y):
  return_x = [i for i in lst_x ]
  return_y = [i for i in lst_y ]
  for i in range(len(lst_x)):
    return_x[i] = norm_fn.transform(lst_x[i].reshape(-1,1)).reshape(lst_x[i].shape)
    return_y[i] = norm_fn.transform(lst_y[i].reshape(-1,1)).reshape(lst_y[i].shape)
  return return_x, return_y
```

Model Architecture

- **LSTM:**

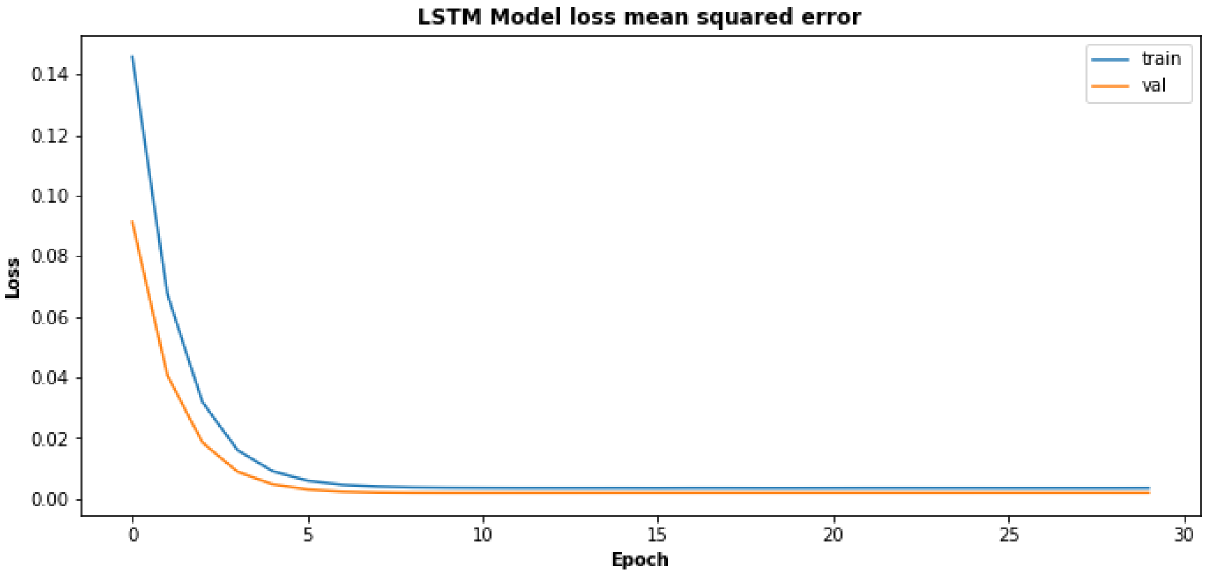


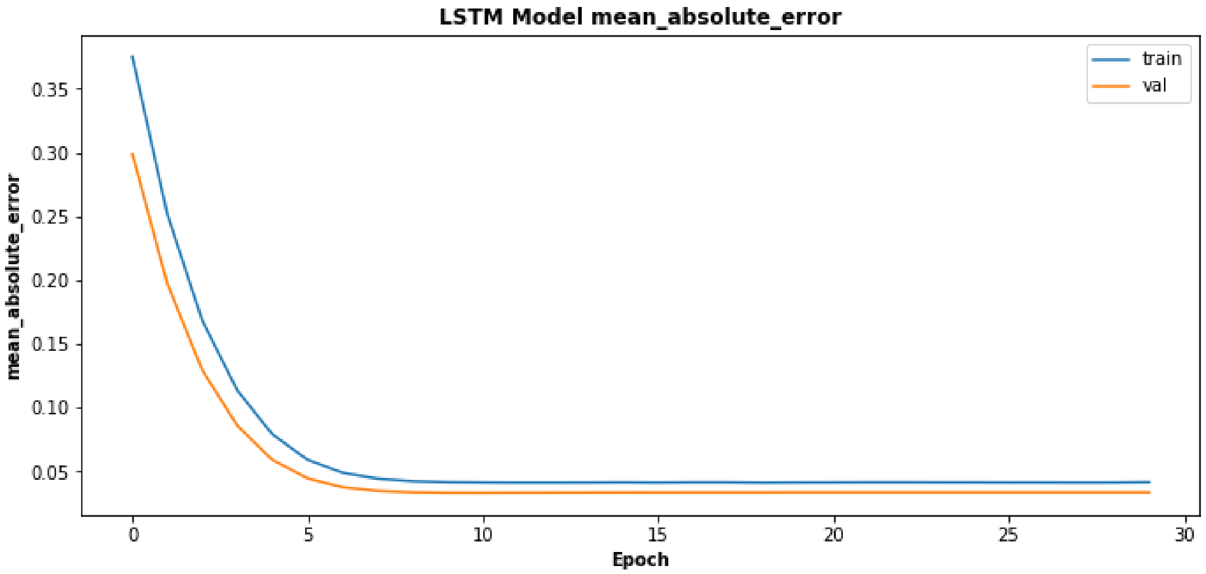
Hyperparameter	Selected Value
No. hidden layer	3
No. neurons	128/256/256
Activation function	relu/linear
Dropout rate	0.25
Optimizer	SGD
Learning Rate	0.0001
Decay	0.00001
Momentum	0.9
Nesterov	TRUE
SOL train/test	9,876/1,236
Batch size	64
Epochs	30

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, None, 128)         66560

 dropout (Dropout)           (None, None, 128)         0

 lstm_1 (LSTM)               (None, None, 256)         394240

 dropout_1 (Dropout)         (None, None, 256)         0

 lstm_2 (LSTM)               (None, 256)               525312

 dropout_2 (Dropout)         (None, 256)               0

 dense (Dense)               (None, 1)                 257

=================================================================
Total params: 986,369
Trainable params: 986,369
Non-trainable params: 0
_________________________________________________________________
```

Model Architecture

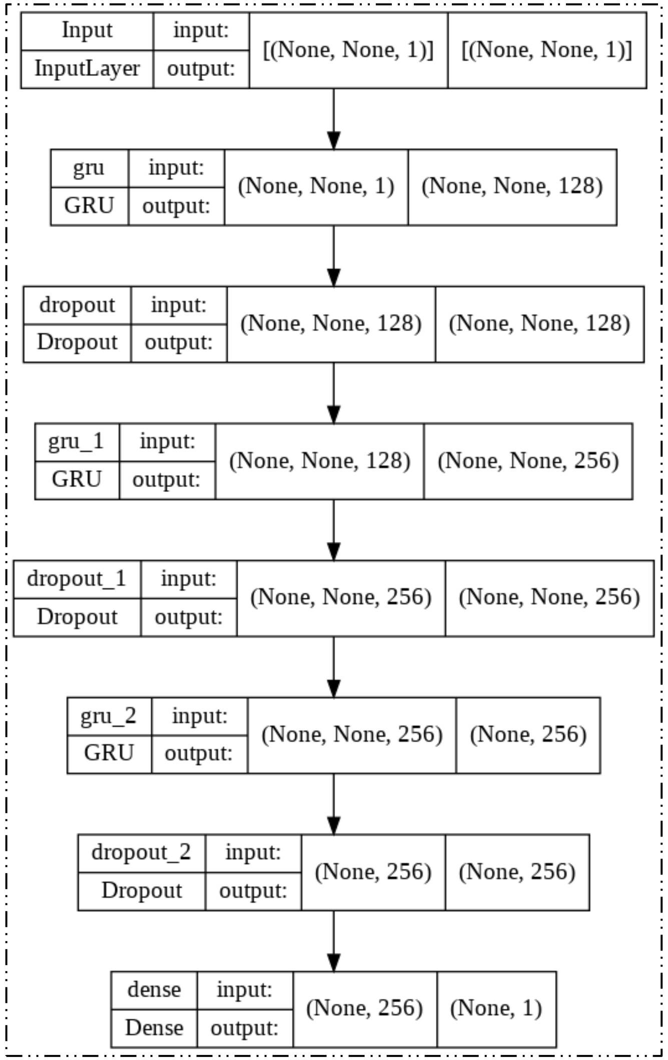
➢ **LSTM:**





Model Architecture

- **GRU**:

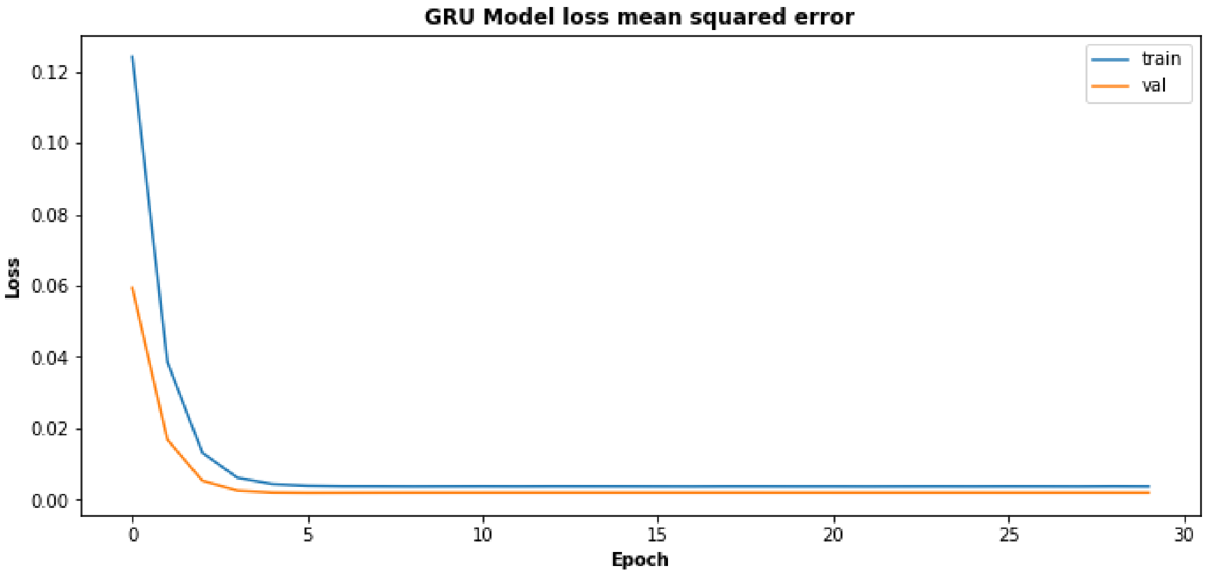


Hyperparameter	Selected Value
No. hidden layer	3
No. neurons	128/256/256
Activation function	relu/linear
Dropout rate	0.25
Optimizer	SGD
Learning Rate	0.0001
Decay	0.00001
Momentum	0.9
Nesterov	TRUE
SOL train/test	9,876/1,236
Batch size	64
Epochs	30

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, None, 128)         50304

 dropout (Dropout)           (None, None, 128)         0

 gru_1 (GRU)                 (None, None, 256)         296448

 dropout_1 (Dropout)         (None, None, 256)         0

 gru_2 (GRU)                 (None, 256)               394752

 dropout_2 (Dropout)         (None, 256)               0

 dense (Dense)               (None, 1)                 257

=================================================================
Total params: 741,761
Trainable params: 741,761
Non-trainable params: 0
_________________________________________________________________
```

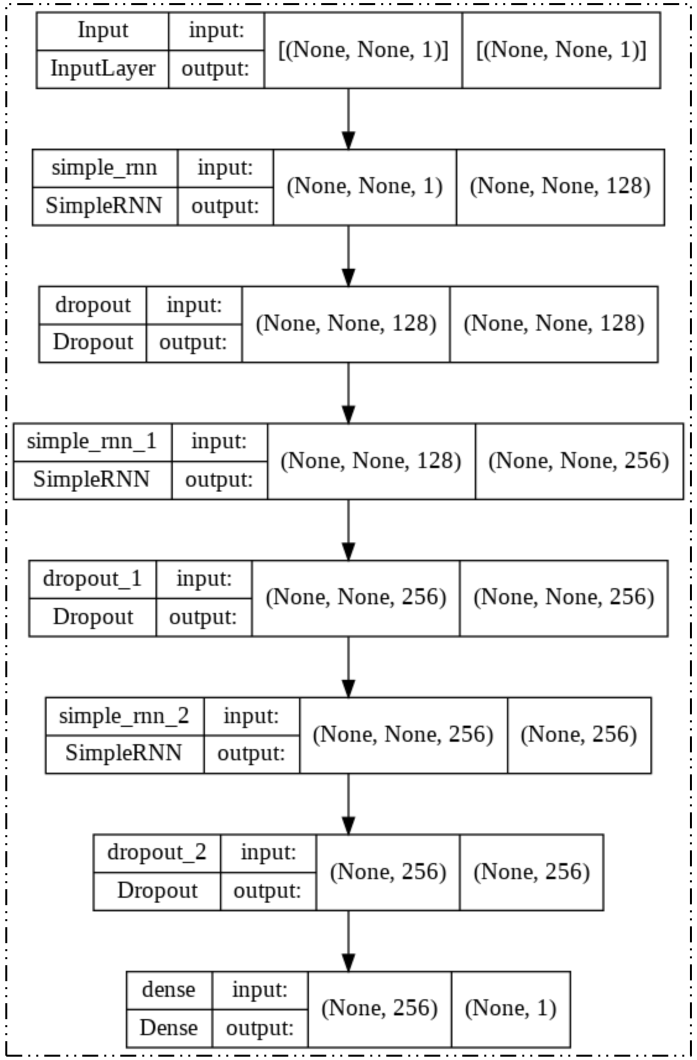
Model Architecture

➢ **GRU:**



Model Architecture

➢ **<u>SimpleRNN</u>:**

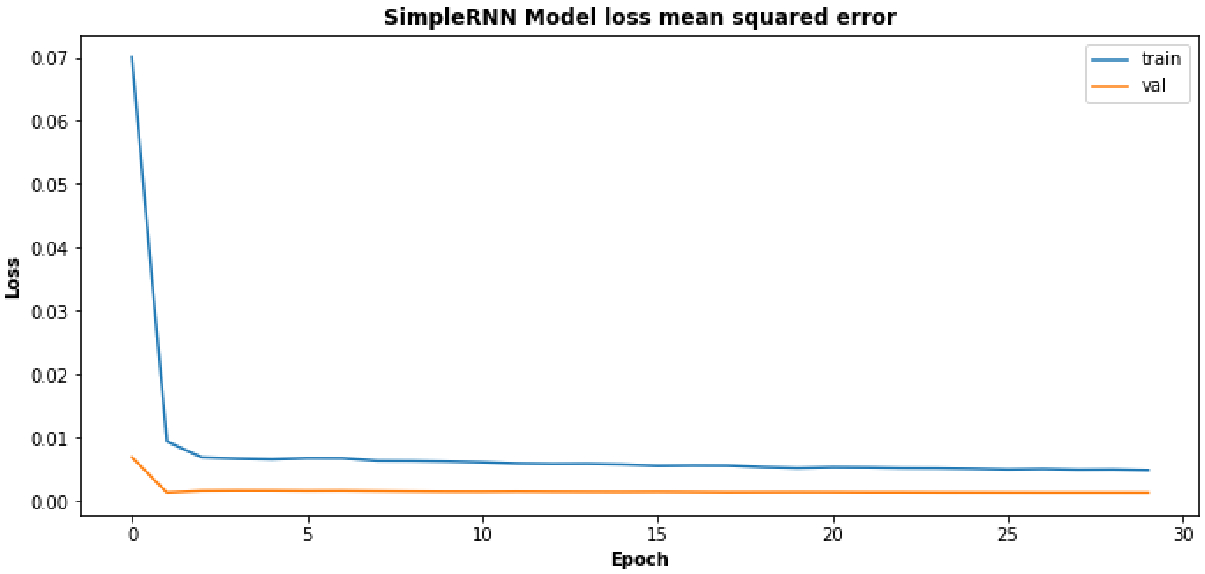


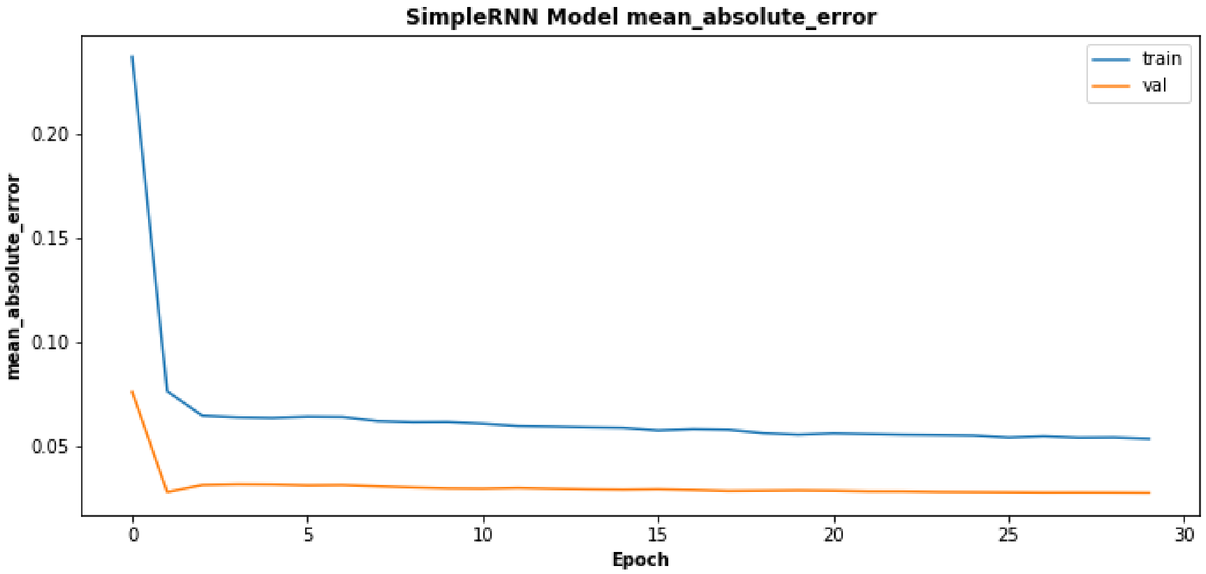
Hyperparameter	Selected Value
No. hidden layer	3
No. neurons	128/256/256
Activation function	relu/linear
Dropout rate	0.25
Optimizer	SGD
Learning Rate	0.0001
Decay	0.00001
Momentum	0.9
Nesterov	TRUE
SOL train/test	9,876/1,236
Batch size	64
Epochs	30

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, None, 128)         16640

 dropout (Dropout)           (None, None, 128)         0

 simple_rnn_1 (SimpleRNN)    (None, None, 256)         98560

 dropout_1 (Dropout)         (None, None, 256)         0

 simple_rnn_2 (SimpleRNN)    (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 dense (Dense)               (None, 1)                 257

=================================================================
Total params: 246,785
Trainable params: 246,785
Non-trainable params: 0
_________________________________________________________________
```

Model Architecture

➢ **<u>SimpleRNN</u> :**





Model Architecture

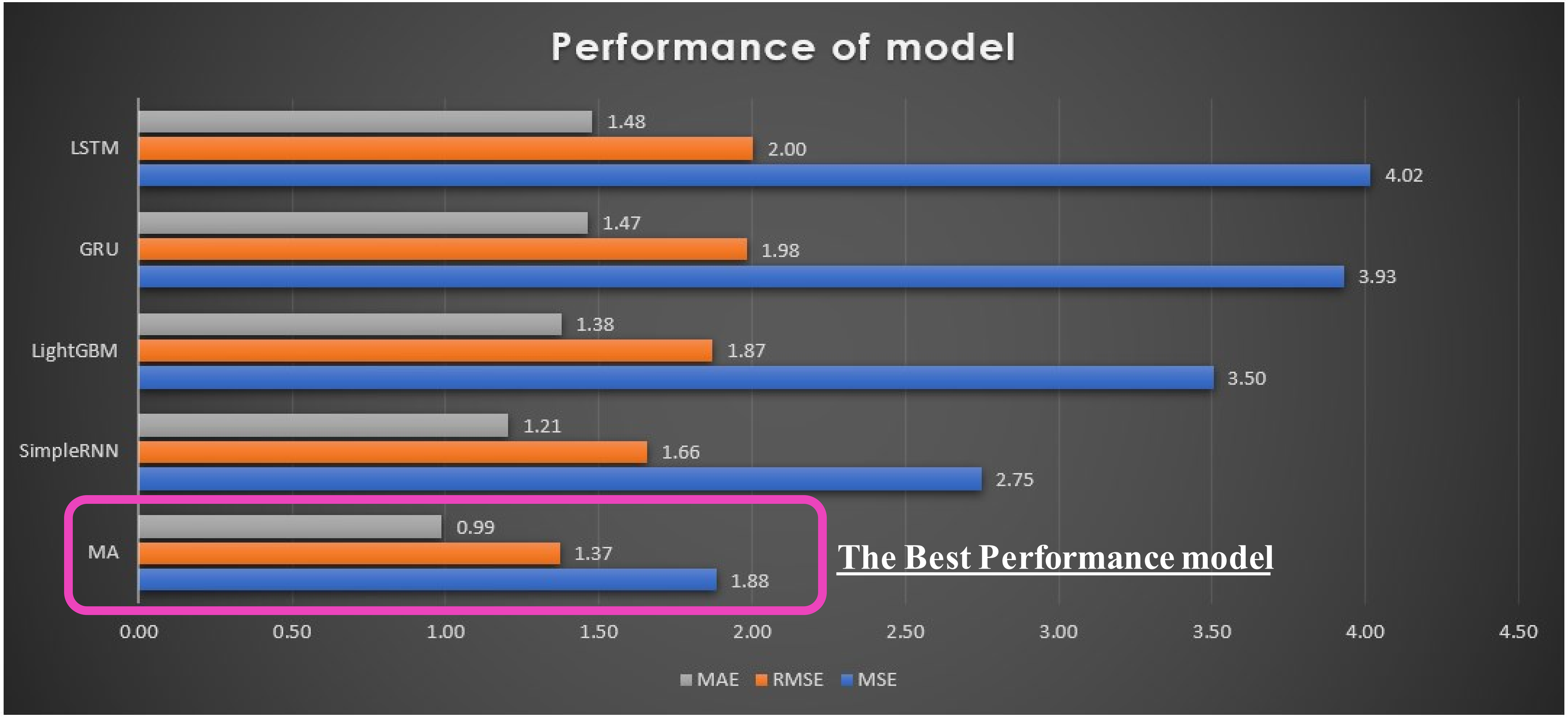
➢ **LightGBM**:

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

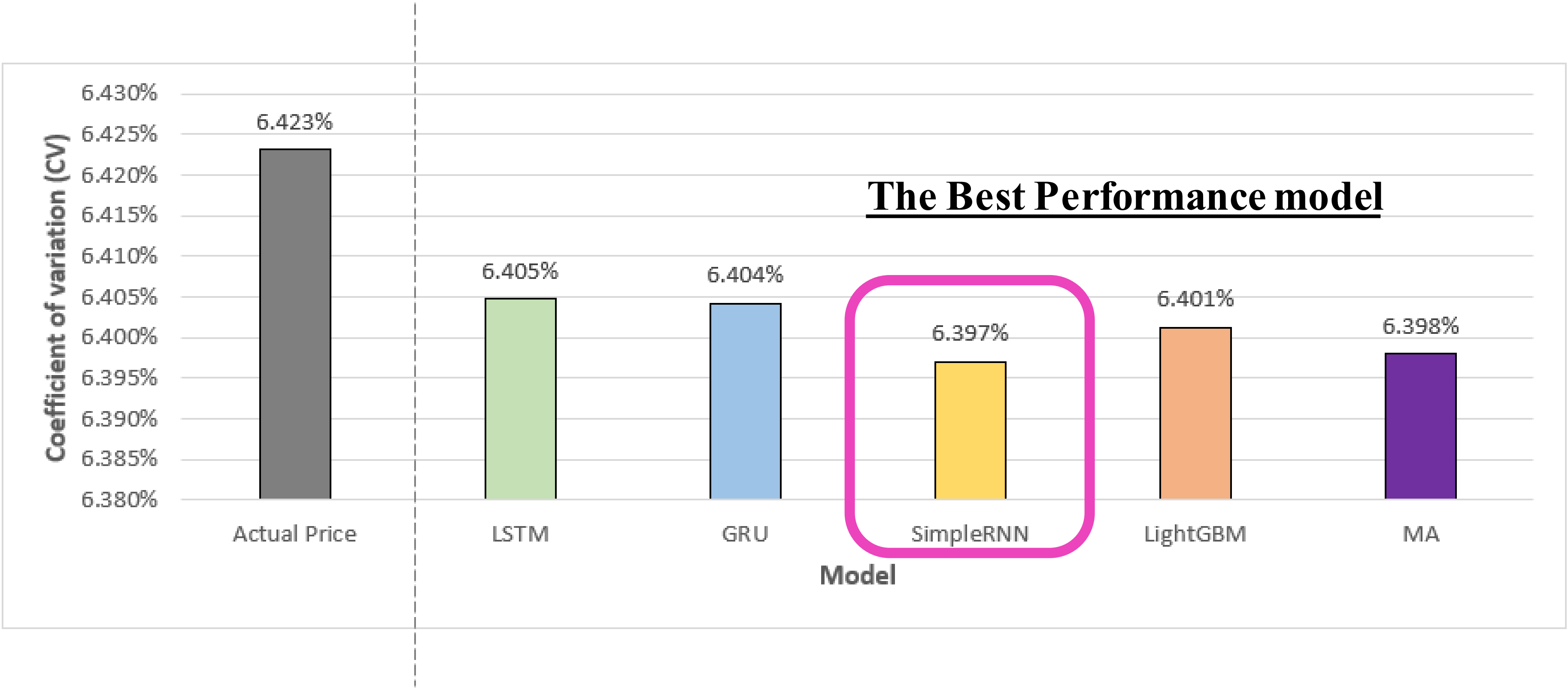
➢ **Moving Average**:

```
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  ]])
```

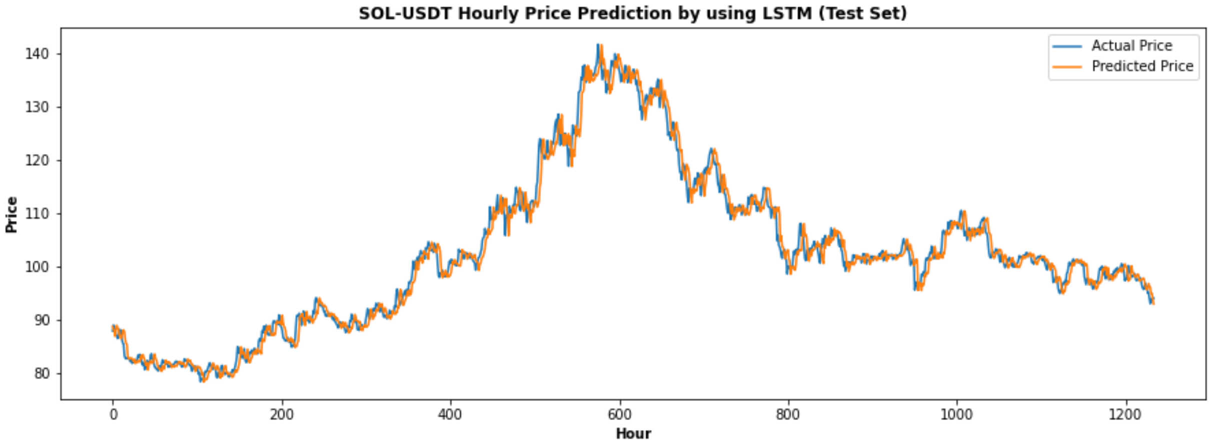
Prediction & Evaluation

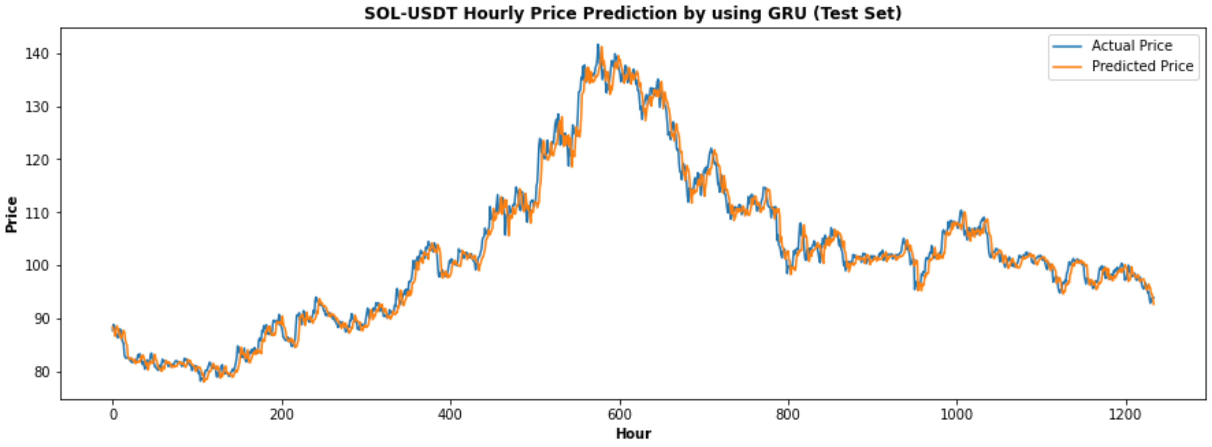


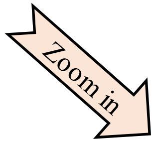
Prediction & Evaluation

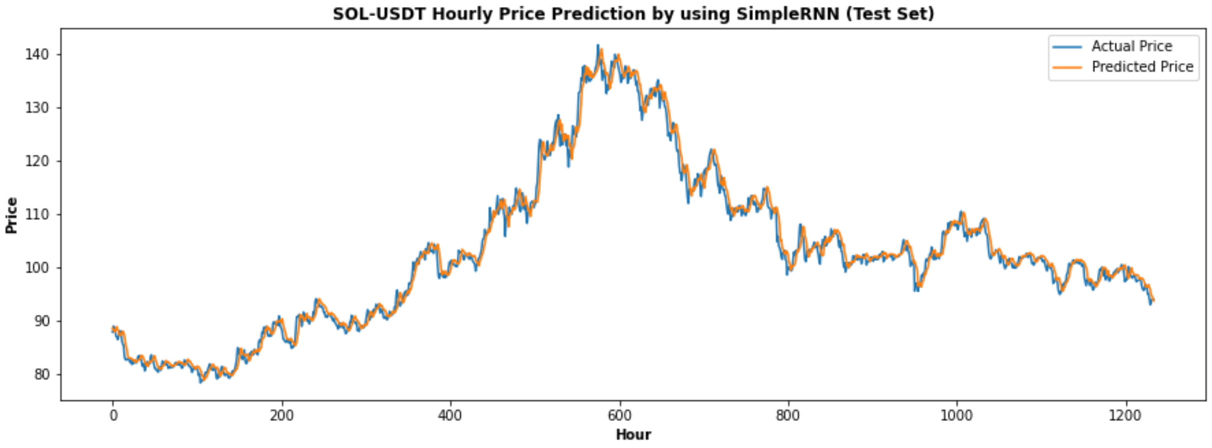


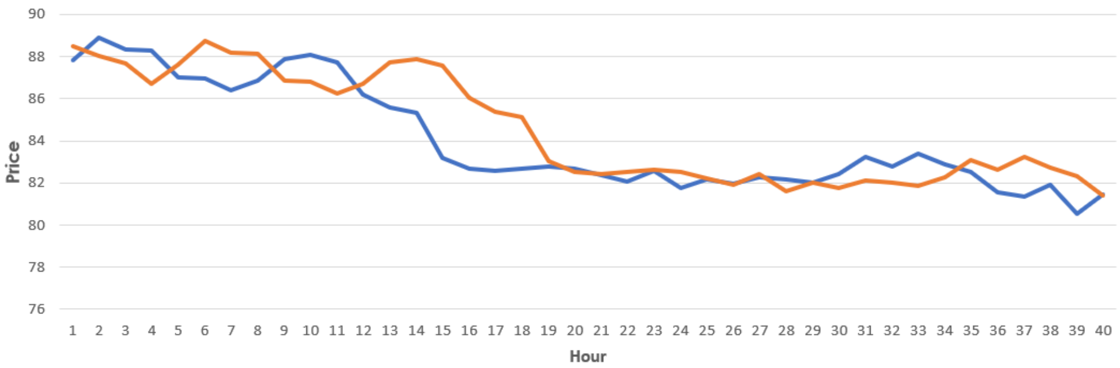
Prediction & Evaluation



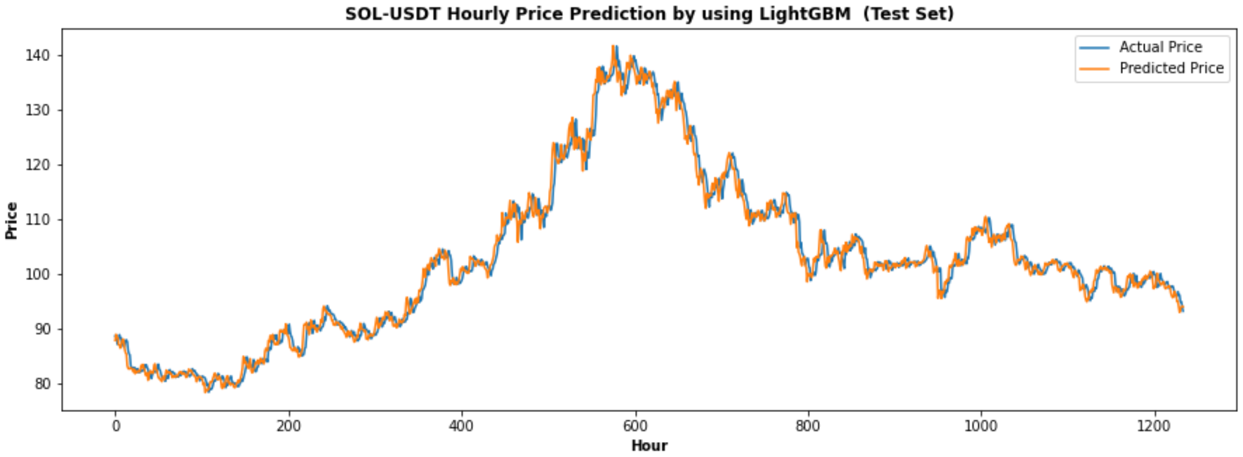


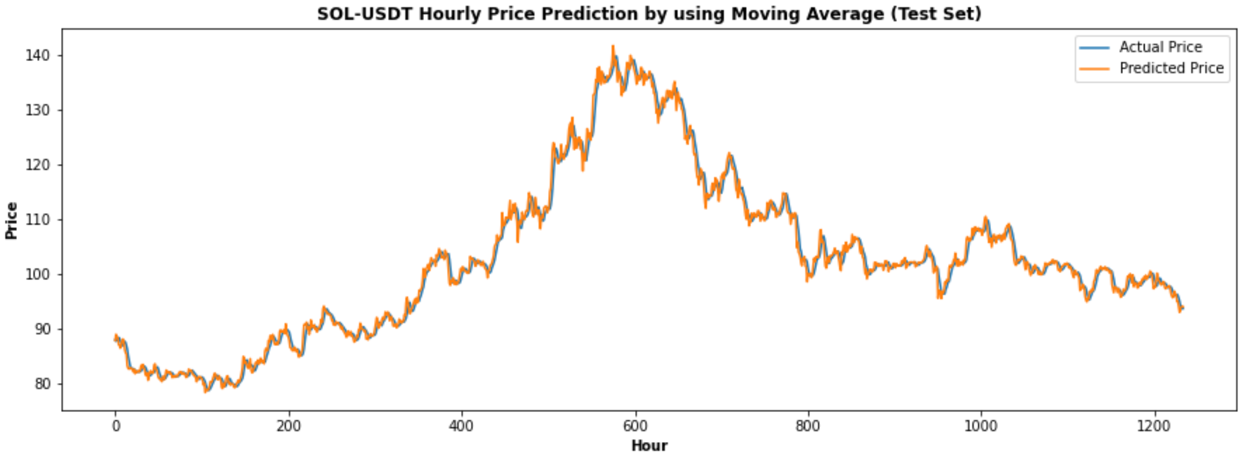






Prediction & Evaluation





Agenda

- Introduction
- Literature Review
- Theoretical Frameworks
- Data
- Methodology
- Empirical result
- Conclusion

Backtesting Trading Strategy: Single Coin

➢ **Trading Strategy 1: single coin**

```
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).

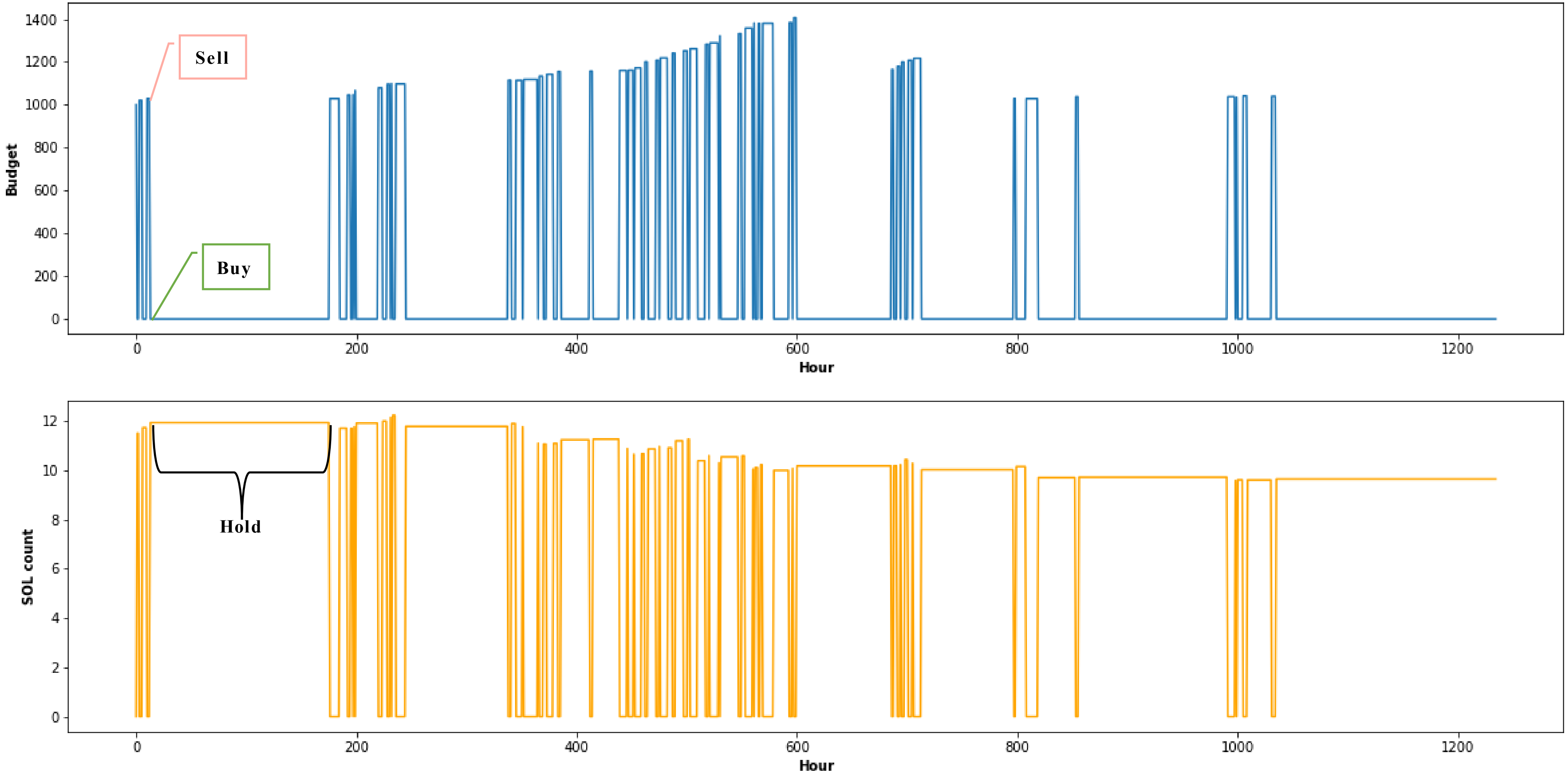
Noted:
-Stopper = 15%

Backtesting Trading Strategy: Single Coin

The highest unrealized profit

Items	LSTM	GRU	SimpleRNN	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 data)	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	1,040	1,079	1,252	1,120	1,040
Total Portfolio value at the end	906	940	1,069	986	925
Unrealized Profit	**(94)**	**(60)**	**69**	**(14)**	**(75)**
% Profit	**-9%**	**-6%**	**7%**	**-1%**	**-7%**

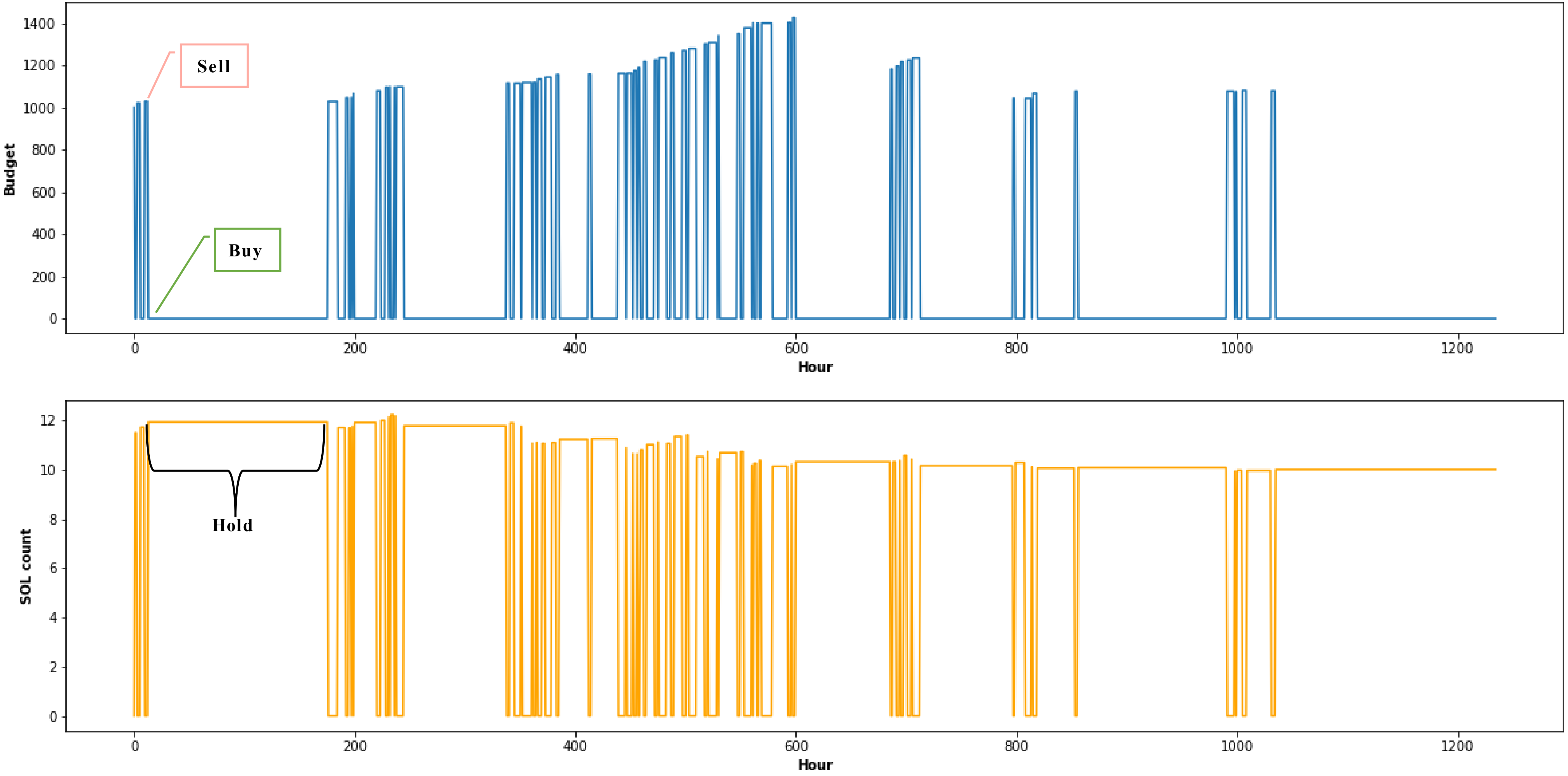
Backtesting Trading Strategy: Single Coin



LSTM

Items	LSTM
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,040
Total Portfolio value at the end	906
Unrealized Profit	(94)
% Profit	-9%

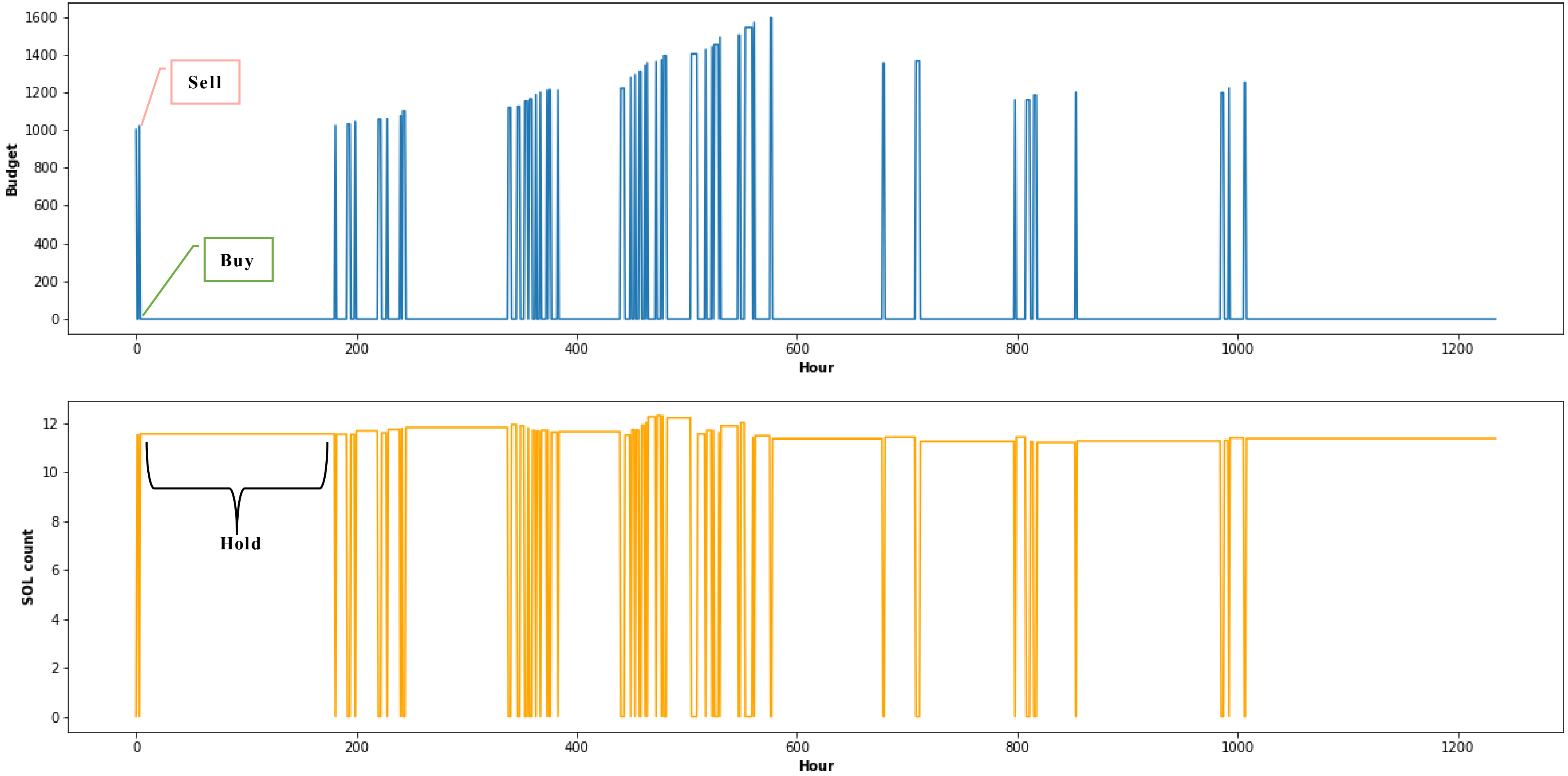
Backtesting Trading Strategy: Single Coin



GRU

Items	GRU
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,079
Total Portfolio value at the end	940
Unrealized Profit	(60)
% Profit	-6%

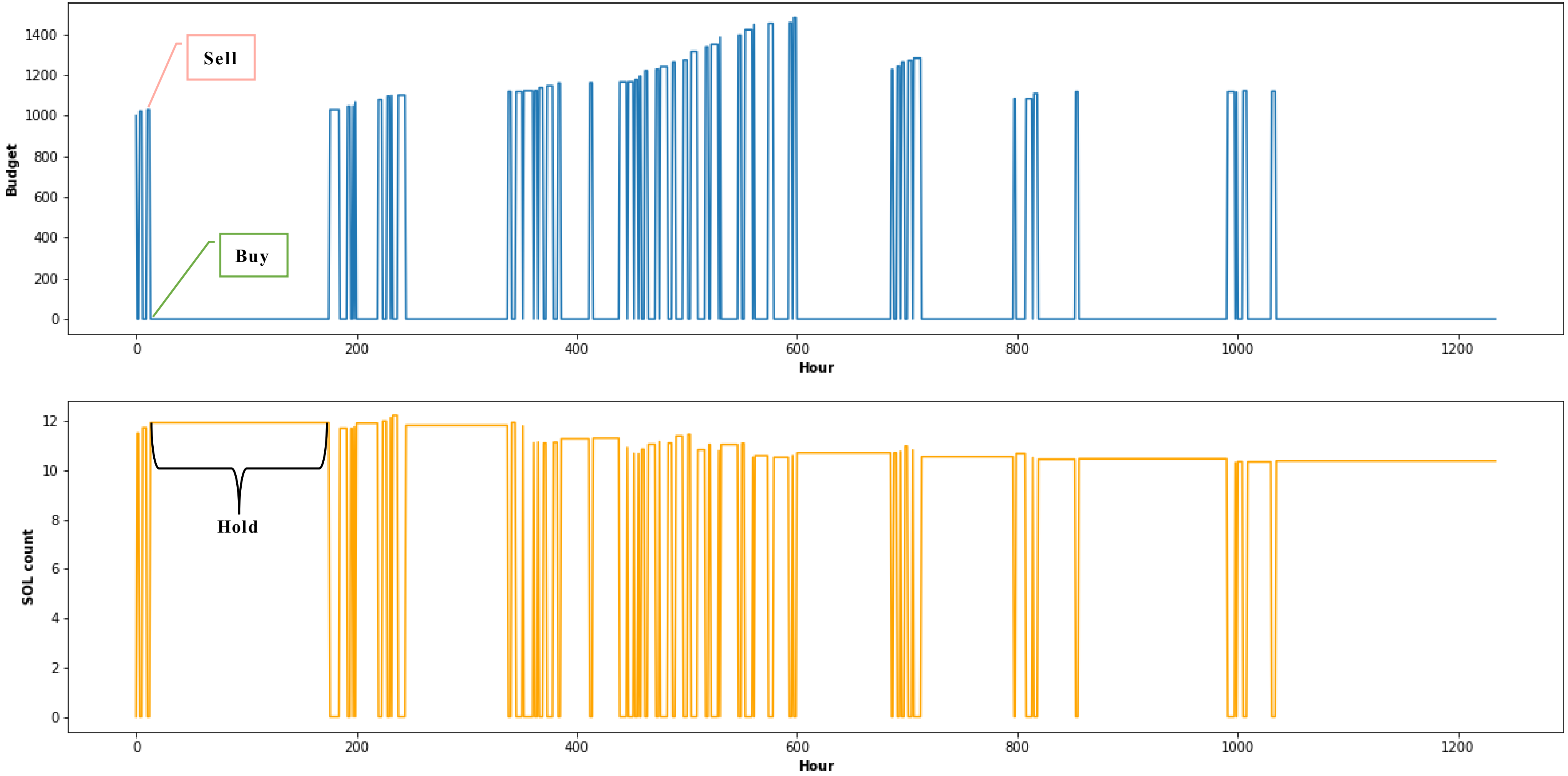
Backtesting Trading Strategy: Single Coin



SimpleRNN

Items	SimpleRNN
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,252
Total Portfolio value at the end	1,069
Unrealized Profit	69
% Profit	7%

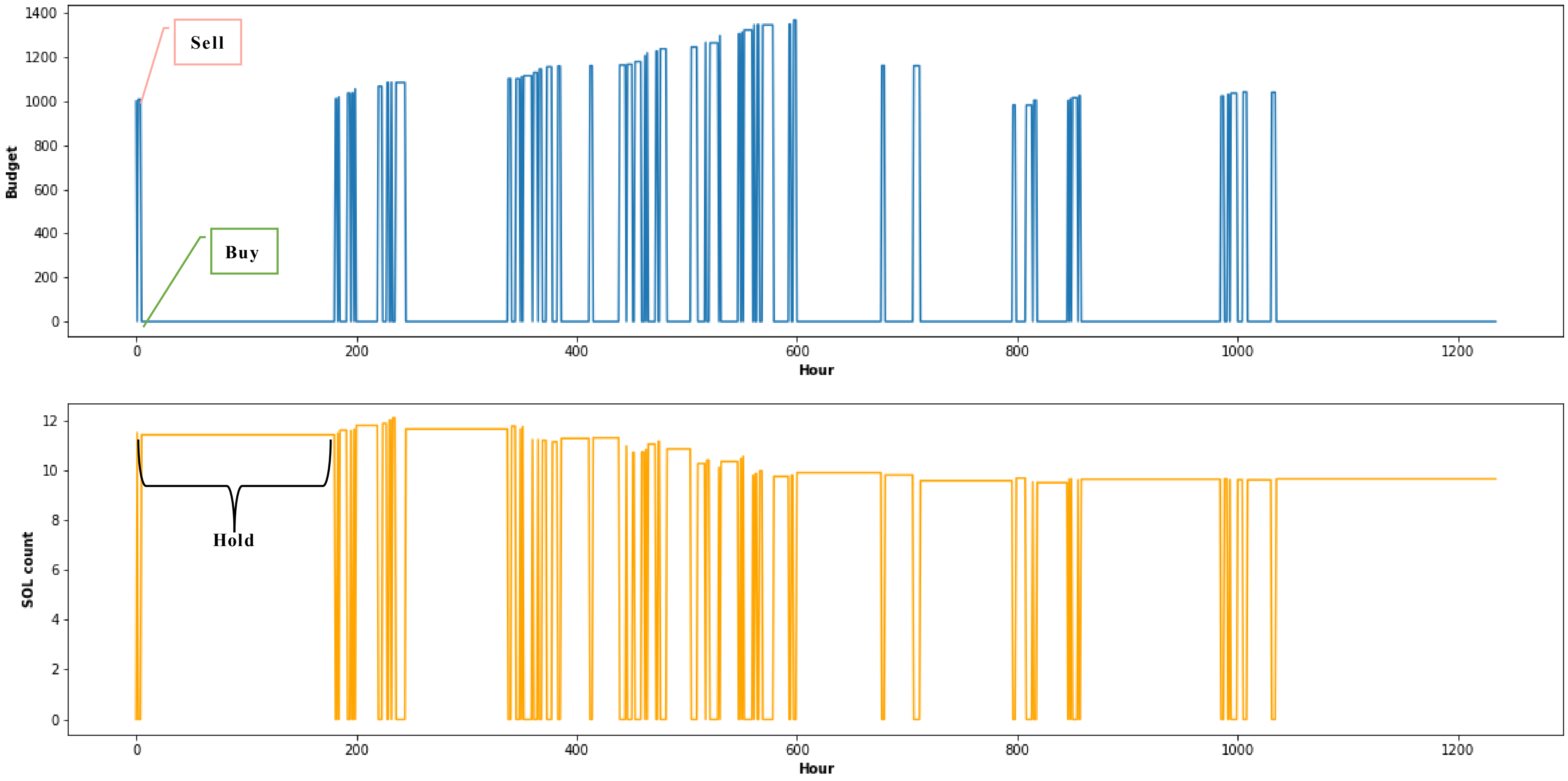
Backtesting Trading Strategy: Single Coin



LightGBM

Items	LightGBM
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,120
Total Portfolio value at the end	986
Unrealized Profit	(14)
% Profit	-1%

Backtesting Trading Strategy: Single Coin



MA

Items	MA
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,040
Total Portfolio value at the end	925
Unrealized Profit	(75)
% Profit	-7%

Backtesting Trading Strategy: Multiple Coins (SOL and BTC)

➢ **Trading Strategy 1: multiple coins (SOL and BTC)**

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).

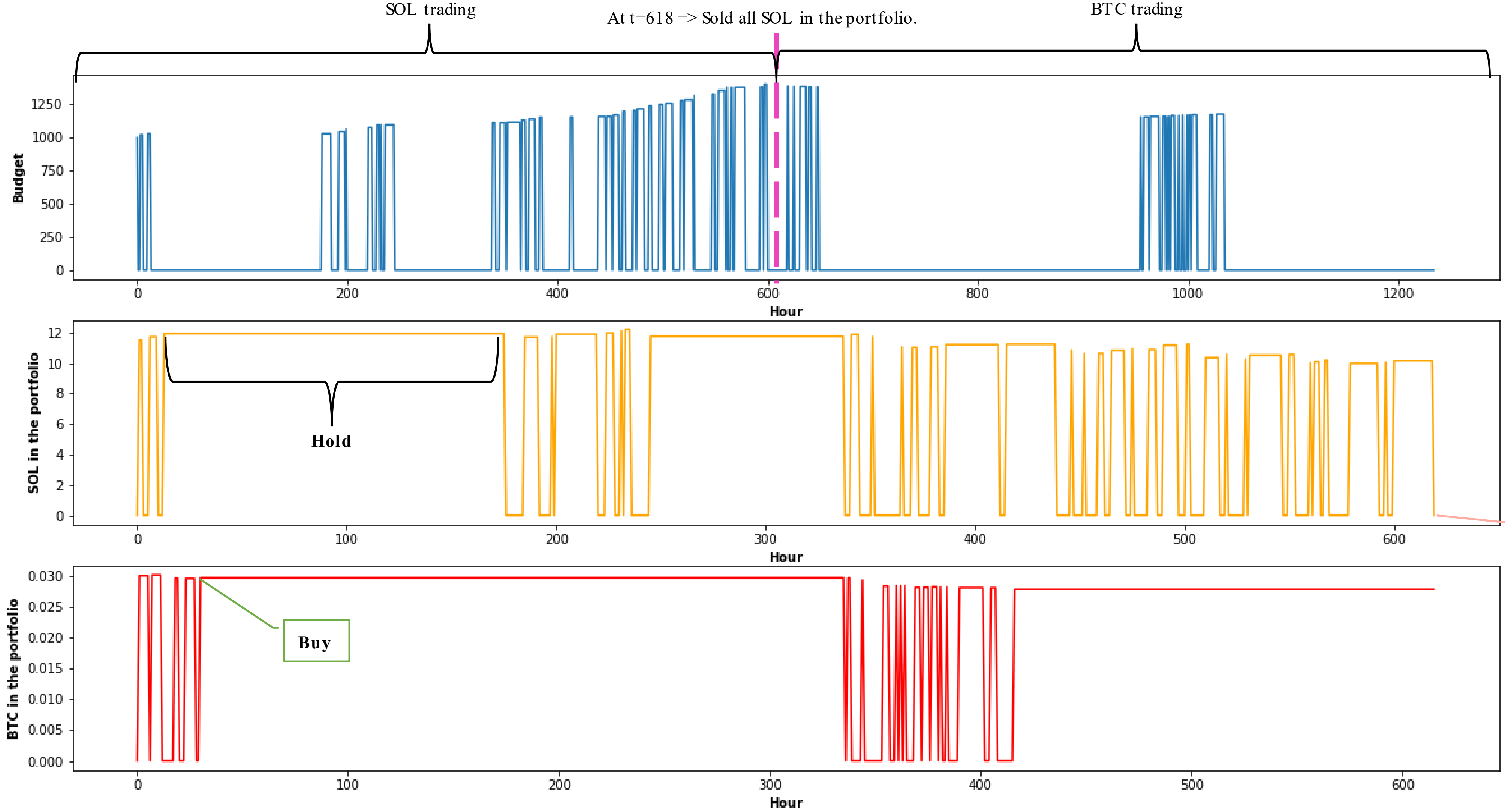
Noted:
-Total hours = 1,236 hours
-The first part (Hours: 1-617) will use ML models to trade only SOL.
-If the hour is 618, we will sell all SOL coins in our portfolio.
-The last part (hours: 619-1,236) will use ML models to trade only BTC.
-Stopper = 15%

Backtesting Trading Strategy: Multiple Coins (SOL and BTC)

The highest unrealized profit

Items	LSTM	GRU	SimpleRNN	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 data)	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	1,177	1,187	1,280	1,239	1,148
Total Portfolio value at the end	1,078	1,088	1,162	1,135	1,043
Unrealized Profit	**78**	**88**	**162**	**135**	**43**
% Profit	**8%**	**9%**	**16%**	**14%**	**4%**

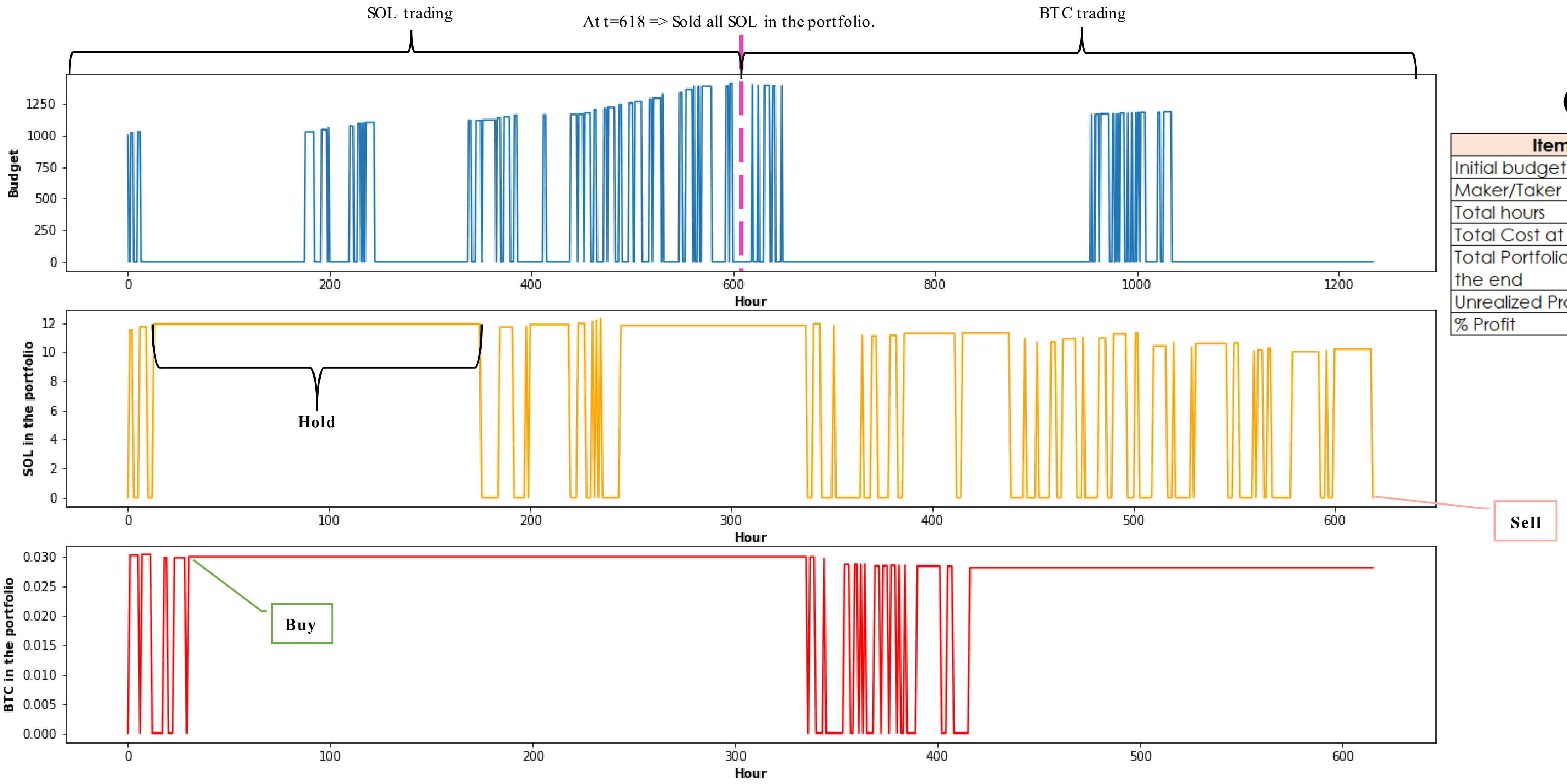
Backtesting Trading Strategy: Multiple Coins (SOL and BTC)



LSTM

Items	LSTM
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,177
Total Portfolio value at the end	1,078
Unrealized Profit	78
% Profit	8%

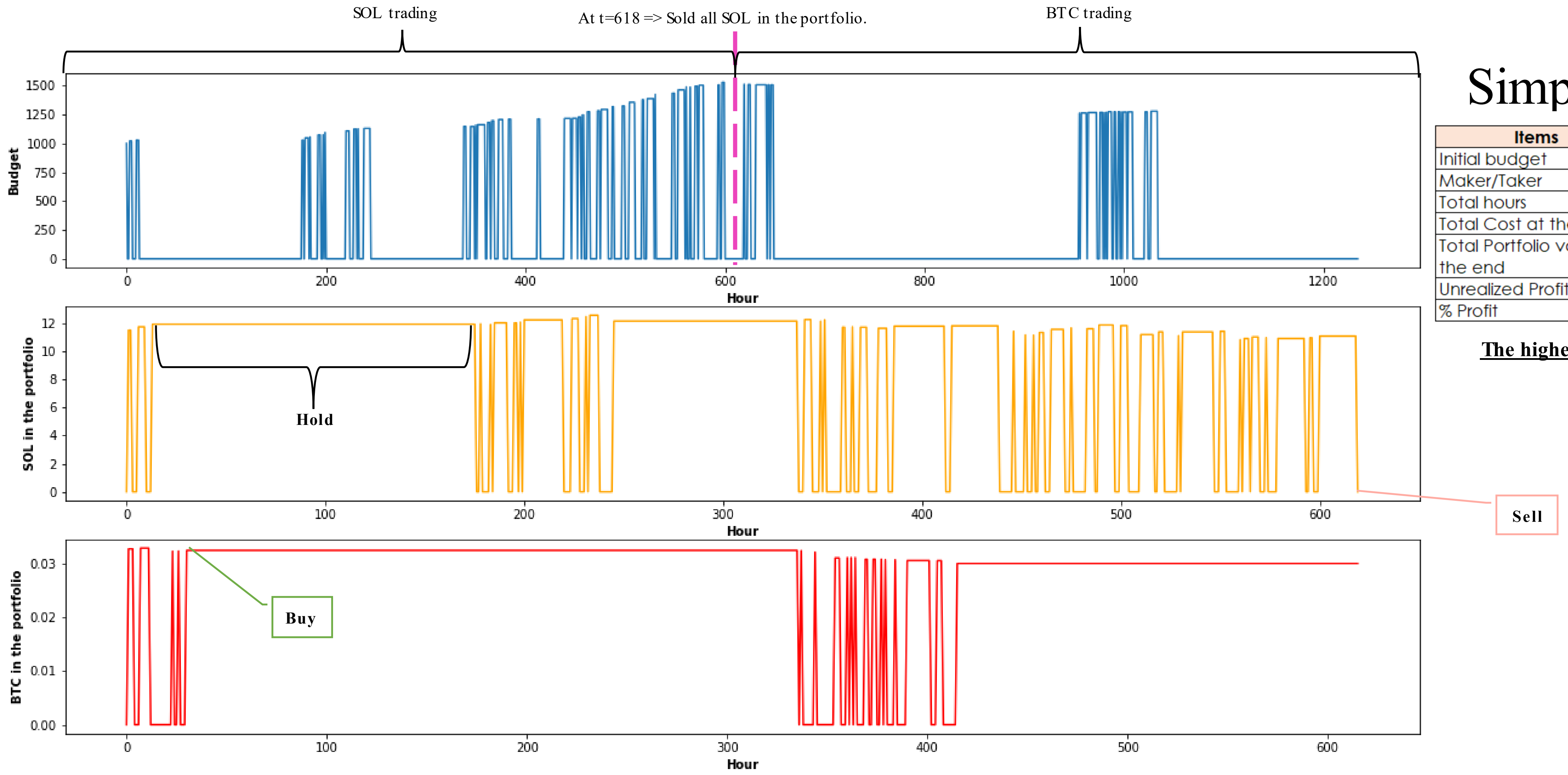
Backtesting Trading Strategy: Multiple Coins (SOL and BTC)



GRU

Items	GRU
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,187
Total Portfolio value at the end	1,088
Unrealized Profit	88
% Profit	9%

Backtesting Trading Strategy: Multiple Coins (SOL and BTC)

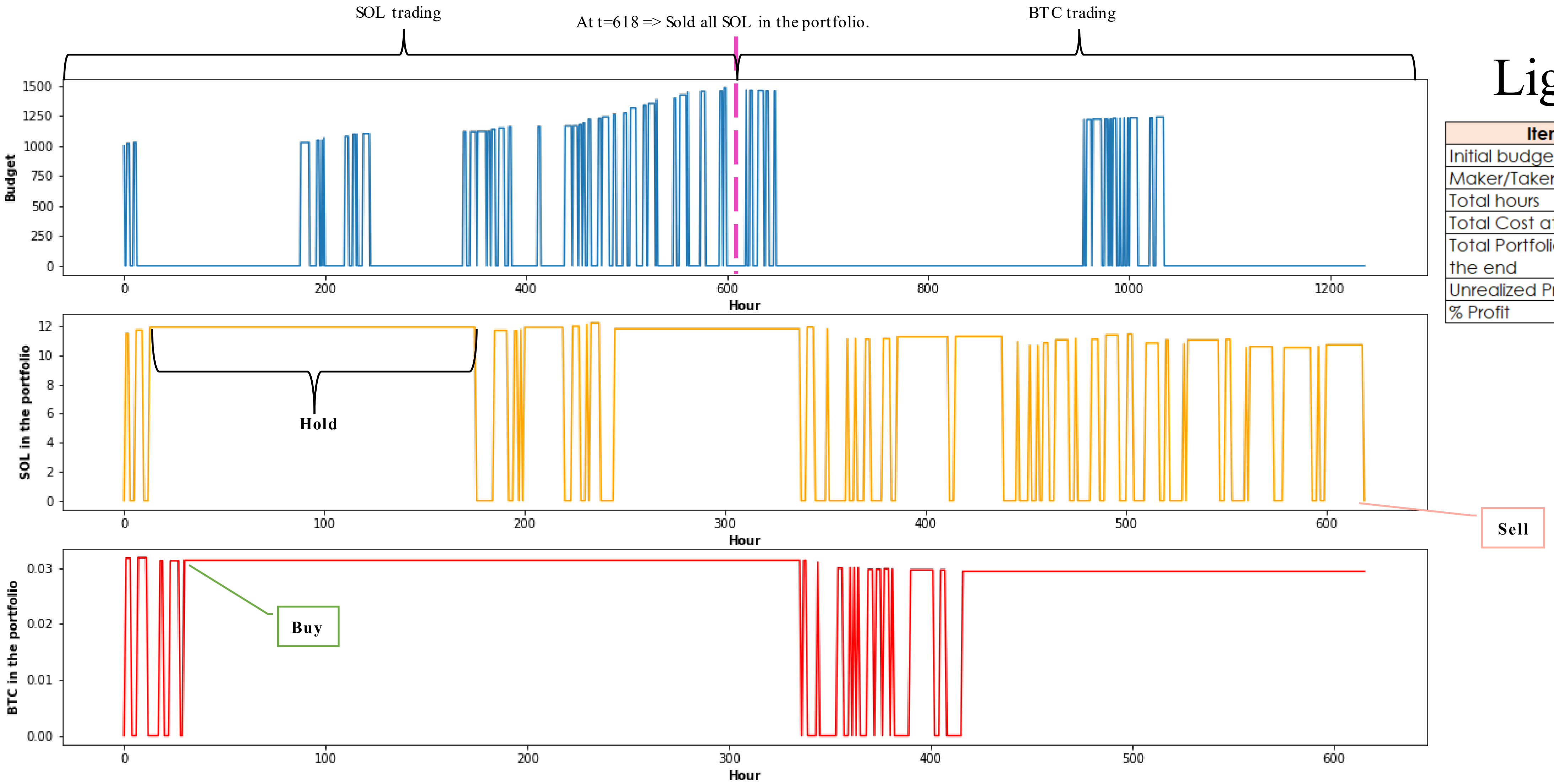


SimpleRNN

Items	SimpleRNN
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,280
Total Portfolio value at the end	1,162
Unrealized Profit	162
% Profit	16%

The highest unrealized profit

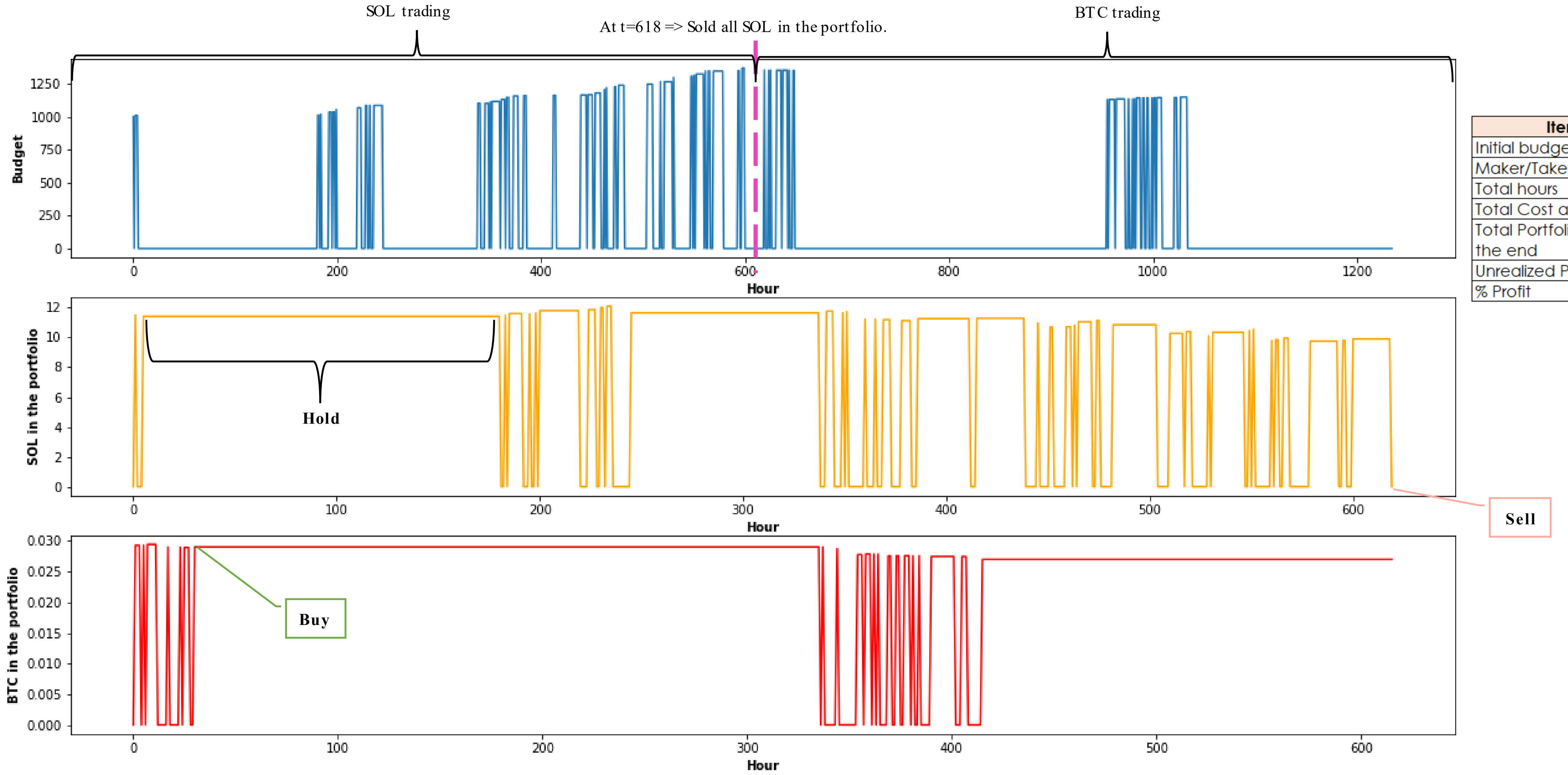
Backtesting Trading Strategy: Multiple Coins (SOL and BTC)



LightGBM

Items	LightGBM
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,239
Total Portfolio value at the end	1,135
Unrealized Profit	**135**
% Profit	**14%**

Backtesting Trading Strategy: Multiple Coins (SOL and BTC)



MA

Items	MA
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,148
Total Portfolio value at the end	1,043
Unrealized Profit	**43**
% Profit	**4%**

Agenda

- Introduction
- Literature Review
- Theoretical Frameworks
- Data
- Methodology
- Empirical result
- Conclusion

Conclusion

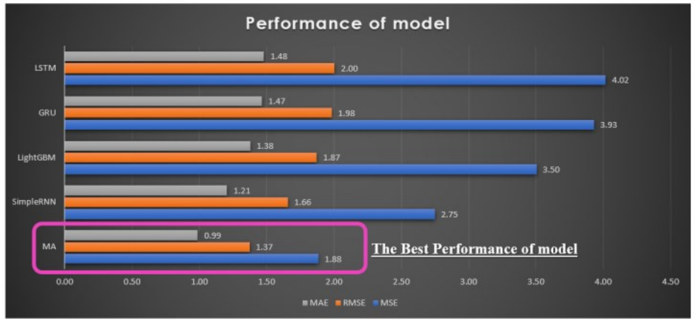
LightGBM

MA → **Best Performance**

LSTM

GRU

Simple RNN → **Maximum Profit**



Single Coin

Items	SimpleRNN
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,252
Total Portfolio value at the end	1,069
Unrealized Profit	69
% Profit	7%

Multiple Coins

Items	SimpleRNN
Initial budget	1,000
Maker/Taker	0.1%/0.1%
Total hours	1,236
Total Cost at the end	1,280
Total Portfolio value at the end	1,162
Unrealized Profit	162
% Profit	16%

Future work

- 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.