Stock Movement Prediction with LSTM

Anapat Pararaman 6180280

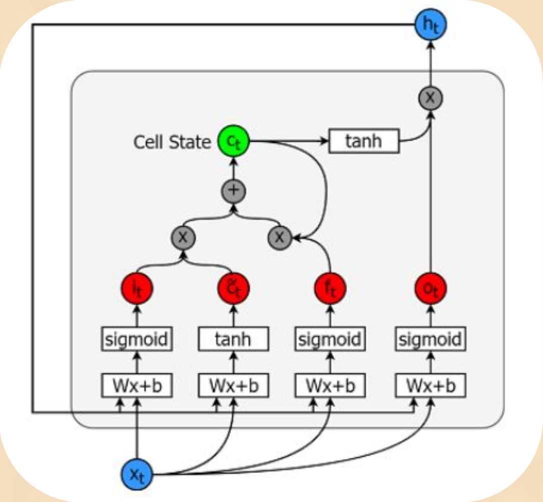
PROJECT GOALS AND MOTIVATION

To have a second pair of eyes.

1

What and Why LSTM?

What and Why LSTM



- **Cell state (c_t)** - This represents the internal memory of the cell which stores both short term memory and long-term memories
- **Hidden state (h_t)** - This is output state information calculated w.r.t. current input, previous hidden state and current cell input which you eventually use to predict the future stock market prices. Additionally, the hidden state can decide to only retrieve the short or long-term or both types of memory stored in the cell state to make the next prediction.
- **Input gate (i_t)** - Decides how much information from current input flows to the cell state
- **Forget gate (f_t)** - Decides how much information from the current input and the previous cell state flows into the current cell state
- **Output gate (o_t)** - Decides how much information from the current cell state flows into the hidden state, so that if needed LSTM can only pick the long-term memories or short-term memories and long-term memories

2

Codes

2.1.1: Importing Data

LSTM.py

```python
import yfinance as yf

def getData(symbol, data):
    ticker = yf.Ticker(symbol)
    histData = ticker.history(period='5y')
    return histData[data][:-360]

price = getData("PTT.BK", "Close")
volume = getData("PTT.BK", "Volume")
```

```
Date
2017-04-04    31.940359
2017-04-05    31.777393
2017-04-07    31.940359
2017-04-10    32.184795
2017-04-11    32.103321
                ...
2020-10-01    30.750196
2020-10-02    30.051325
2020-10-05    30.051325
2020-10-06    30.750196
2020-10-07    30.750196
Name: Close, Length: 857, dtype: float64
```

2.1.2: Applying MinMax Scaler

LSTM.py

```python
import numpy as np
from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(price).reshape(-1,1))

scalerPrice = MinMaxScaler(feature_range= (0,1))
scalerVolume = MinMaxScaler(feature_range= (0,1))

df1 = scalerPrice.fit_transform(np.array(price).reshape(-1,1))
df2 = scalerVolume.fit_transform(np.array(volume).reshape(-1,1))
```

2.1.3: Splitting Datasets

LSTM.py

```
def splitData(df):
    trainingSize = int(len(df)*0.65)
    testSize = len(df)-trainingSize
    trainData, testData = df[0:trainingSize,:], df[trainingSize:len(df),:1]
    return trainData, testData

df1TrainData, df1TestData = splitData(df1)
df2TrainData, df2TestData = splitData(df2)
```

2.1.4: Data Pre-Processing

Training Dataset = [104, 129, 112, 102, 131, 141, 122, 152]

Timestep = 3

	X_Train			Y_Train
Price	104	129	112	102
	129	112	102	131

2.1.4: Data Pre-Processing

LSTM.py

```python
def create_dataset(dataset, time_step):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)

df1XTrain, df1YTrain = create_dataset(df1TrainData, 100)
df1XTest, df1YTest = create_dataset(df1TestData, 100)

df2XTrain, df2YTrain = create_dataset(df2TrainData, 100)
df2XTest, df2YTest = create_dataset(df2TestData, 100)
```

2.2: Initializing Model

LSTM.py

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM

priceModel=Sequential()
priceModel.add(LSTM(50,return_sequences=True,input_shape=(100,1)))
priceModel.add(LSTM(50,return_sequences=True))
priceModel.add(LSTM(50))
priceModel.add(Dense(1))
priceModel.compile(loss='mean_squared_error',optimizer='adam')

volumeModel=Sequential()
volumeModel.add(LSTM(50,return_sequences=True,input_shape=(100,1)))
volumeModel.add(LSTM(50,return_sequences=True))
volumeModel.add(LSTM(50))
volumeModel.add(Dense(1))
volumeModel.compile(loss='mean_squared_error',optimizer='adam')
```

2.2: Initializing Model

```
priceModel.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 lstm_1 (LSTM)               (None, 100, 50)           20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

```
volumeModel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 100, 50)           10400

 lstm_4 (LSTM)               (None, 100, 50)           20200

 lstm_5 (LSTM)               (None, 50)                20200

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

2.2: Initializing Model

LSTM.py

```
priceModel.fit(df1XTrain,df1YTrain,validation_data=(df1XTest,df1YTest),epochs=100,batch_size=64,verbose=1)
volumeModel.fit(df2XTrain,df2YTrain,validation_data=(df2XTest,df2YTest),epochs=100,batch_size=64,verbose=1)
```

2.3.1: Predict Output

LSTM.py

```python
import tensorflow as tf

df1TrainPredict=priceModel.predict(df1XTrain)
df1TestPredict=priceModel.predict(df1XTest)

df2TrainPredict=volumeModel.predict(df2XTrain)
df2TestPredict=volumeModel.predict(df2XTest)

##Transformback to original form
df1TrainPredict=scalerPrice.inverse_transform(df1TrainPredict)
df1TestPredict=scalerPrice.inverse_transform(df1TestPredict)

df2TrainPredict=scalerVolume.inverse_transform(df2TrainPredict)
df2TestPredict=scalerVolume.inverse_transform(df2TestPredict)
```

2.3.2: RMSE Performance

```
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(df1YTrain,df1TrainPredict))
```

40.82547518029299

```
math.sqrt(mean_squared_error(df1YTest,df1TestPredict))
```

35.28085723288469

```
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(df2YTrain,df2TrainPredict))
```

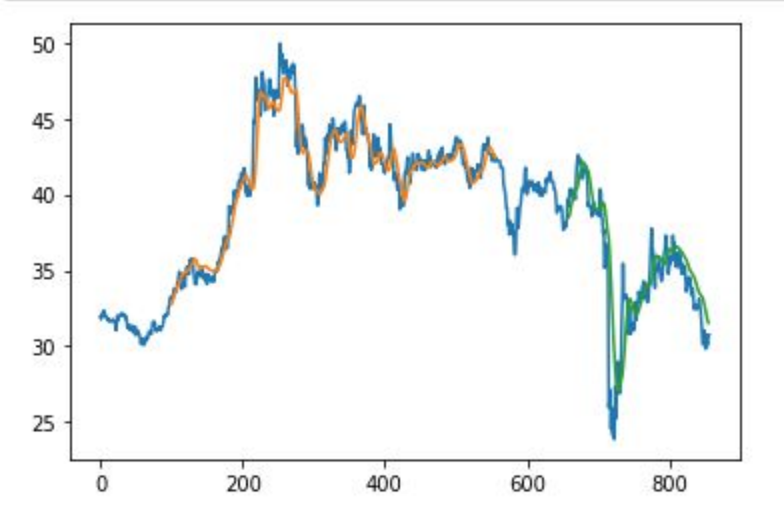
77521299.5104038

```
math.sqrt(mean_squared_error(df2YTest,df2TestPredict))
```

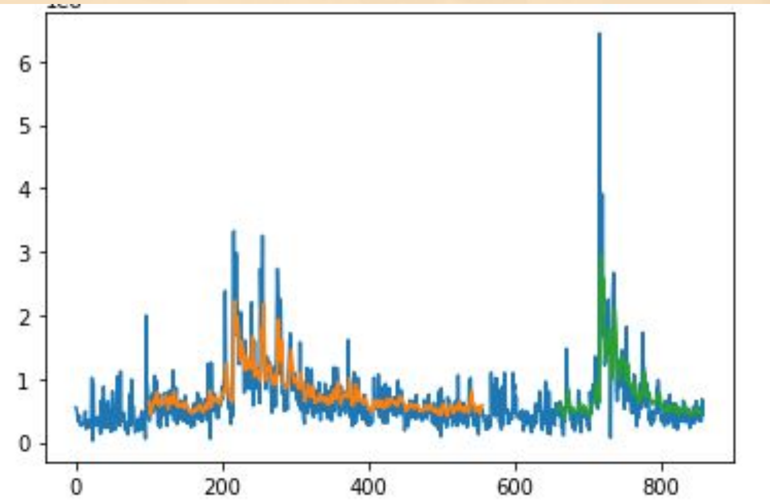
84757047.79172568

2.3.3: Plotting

PRICE



VOLUME



2.4.1: Predicting Future (30 Days)

LSTM.py

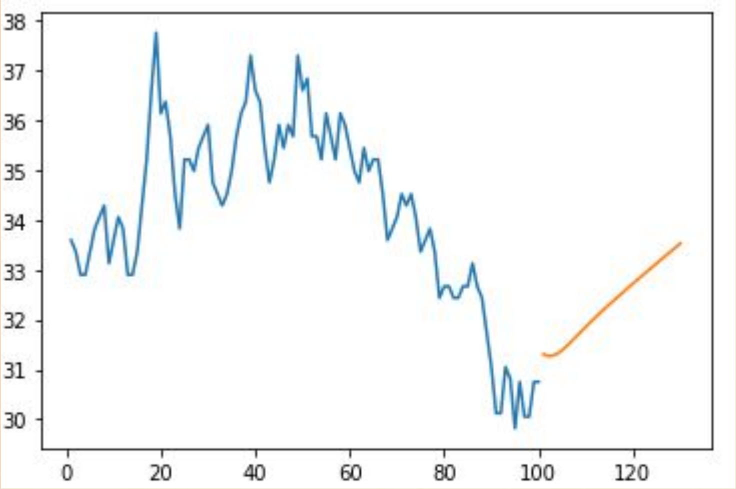
```
def tempInput(testData):
    xInput=testData[200:].reshape(1,-1)
    tempInput=list(xInput)
    tempInput=tempInput[0].tolist()
    return tempInput, xInput

priceTempInput, priceInput = tempInput(df1TestData)
volumeTempInput, volumeInput = tempInput(df2TestData)
```

LSTM.py

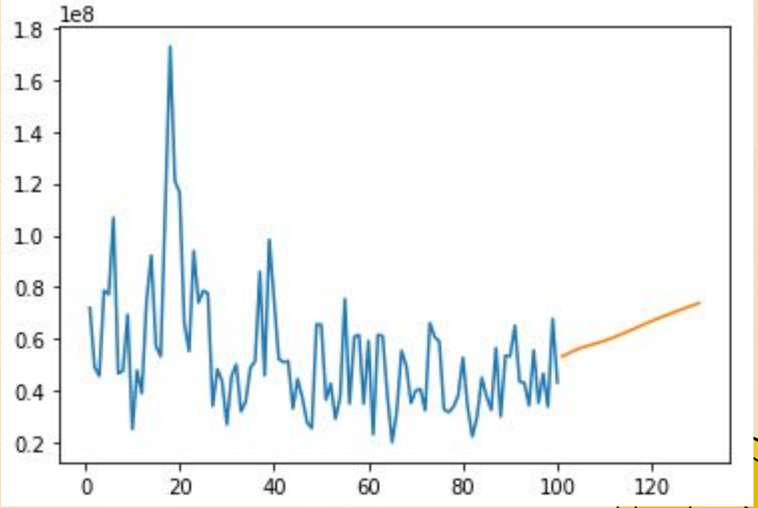
```
from numpy import array

priceOutput=[]
n_steps=100
i=0
while(i<30):

    if(len(priceTempInput)>100):
        priceInput=np.array(priceTempInput[1:])
        priceInput=priceInput.reshape(1,-1)
        priceInput = priceInput.reshape((1, n_steps, 1))
        yhat = priceModel.predict(priceInput, verbose=0)
        priceTempInput.extend(yhat[0].tolist())
        priceTempInput=priceTempInput[1:]
        priceOutput.extend(yhat.tolist())
        i=i+1
    else:
        priceInput = priceInput.reshape((1, n_steps,1))
        yhat = priceModel.predict(priceInput, verbose=0)
        print(yhat[0])
        priceTempInput.extend(yhat[0].tolist())
        print(len(priceTempInput))
        priceOutput.extend(yhat.tolist())
        i=i+1


print(priceOutput)
```

2.4.2: Plotting

PRICE



VOLUME

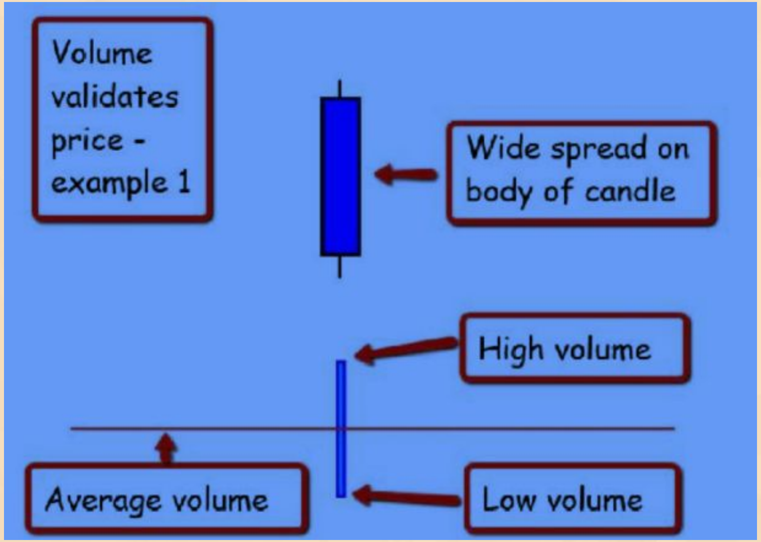


3

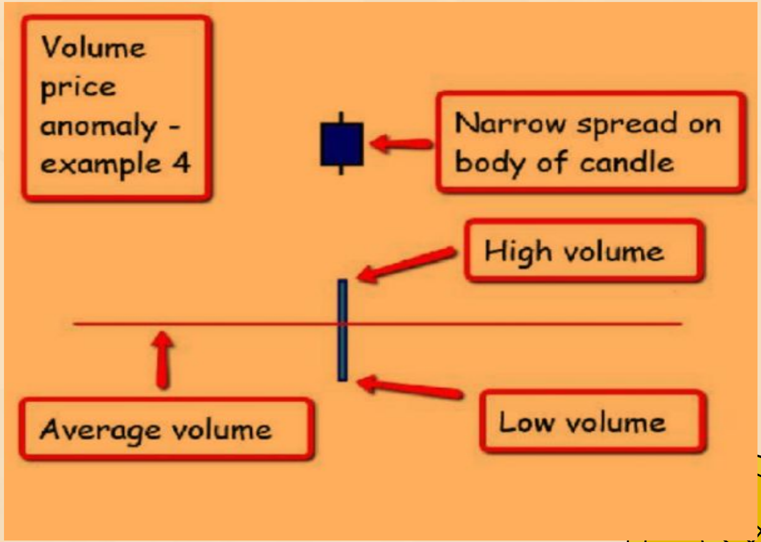
Analyzing Results

3.1: Why Both?

VALIDATION

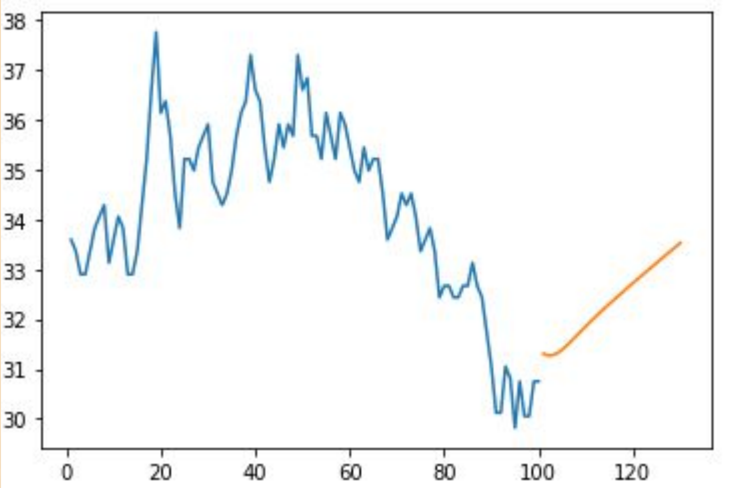


ANOMALY

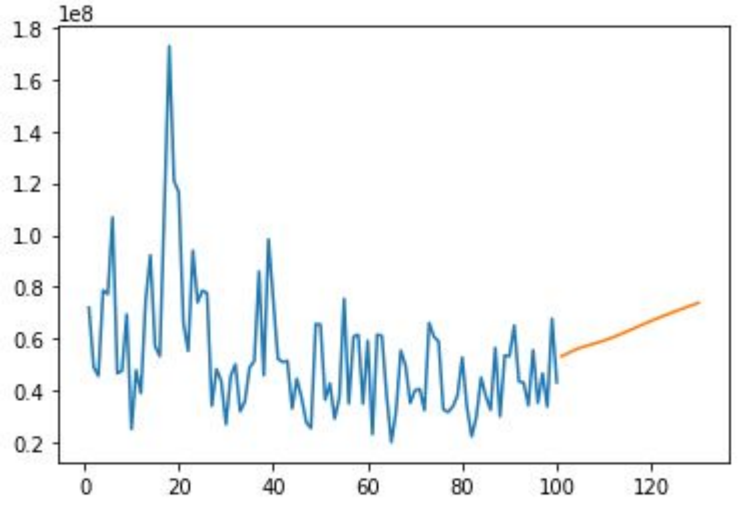


3.2: Analyzing

PRICE



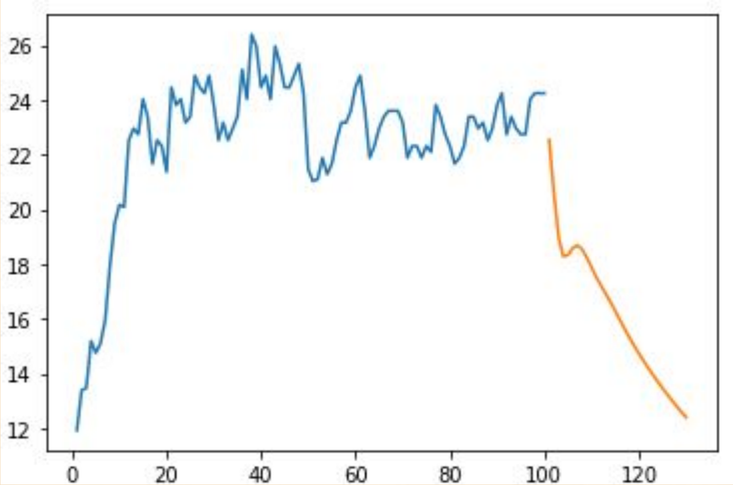
VOLUME



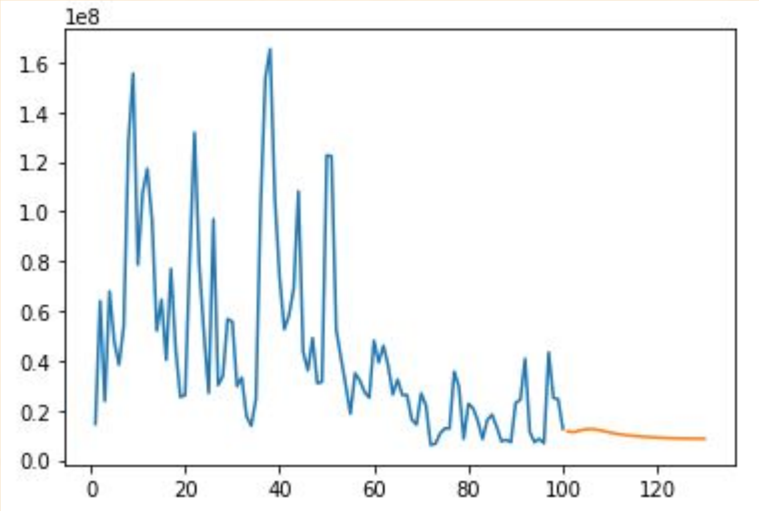

6.50 (19.91%) , 26
23 bars, 35d
Vol 1.472B
power of TradingView
e for 30 days. We'll help you trade
24 Sep '20
07 Oct '20
11 Nov '20
Sep
Nov
Dec
2021
Feb
42.50
39.00
32.50
05:38:06

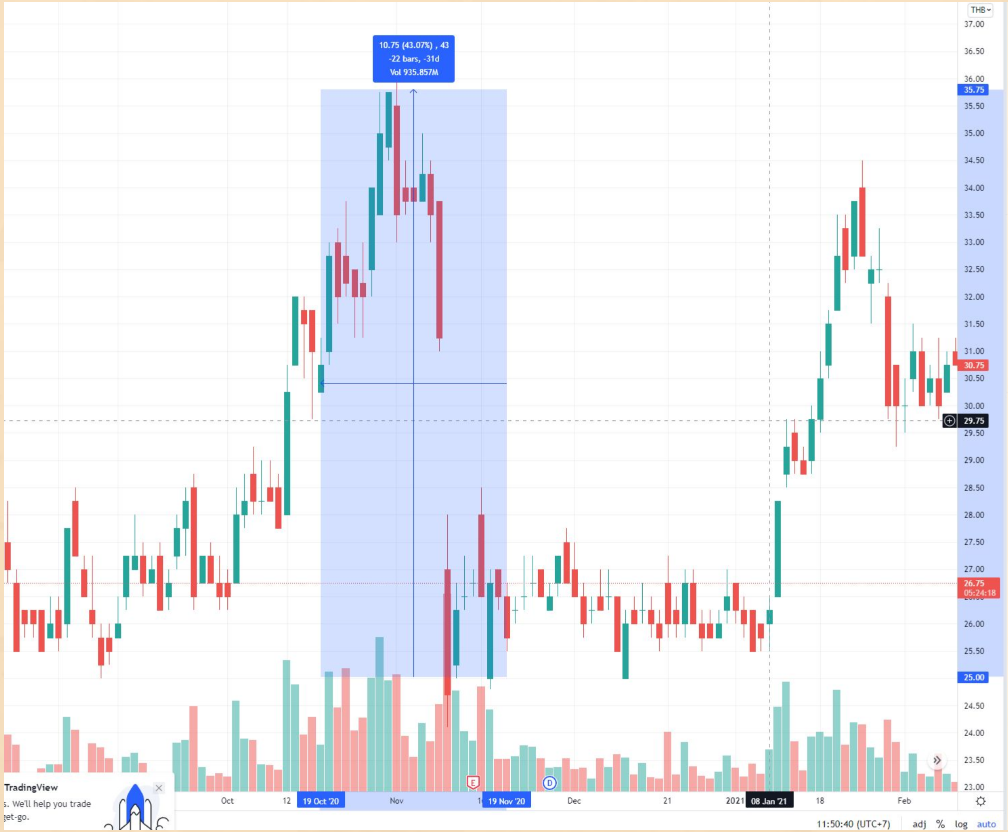
3.2: Analyzing

PRICE



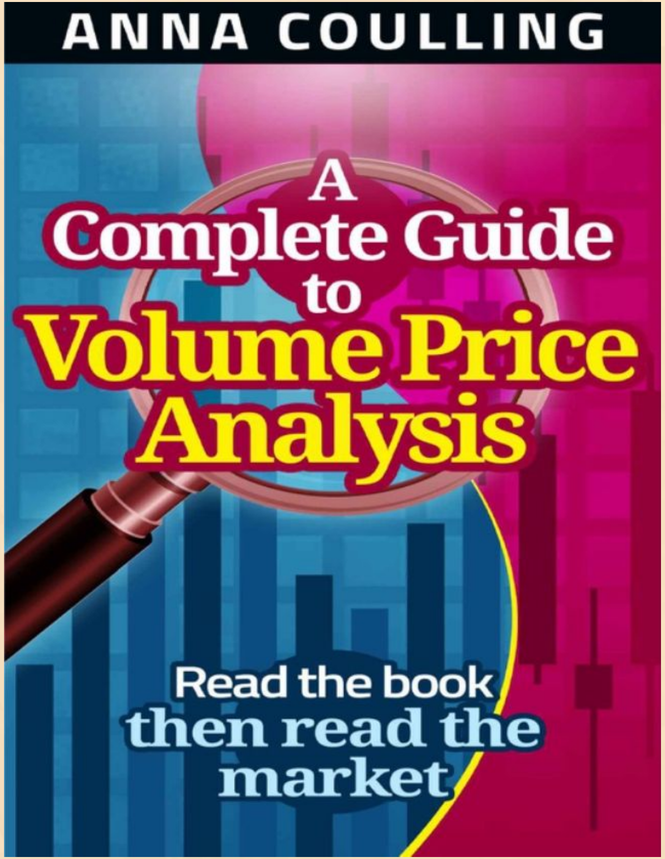
VOLUME

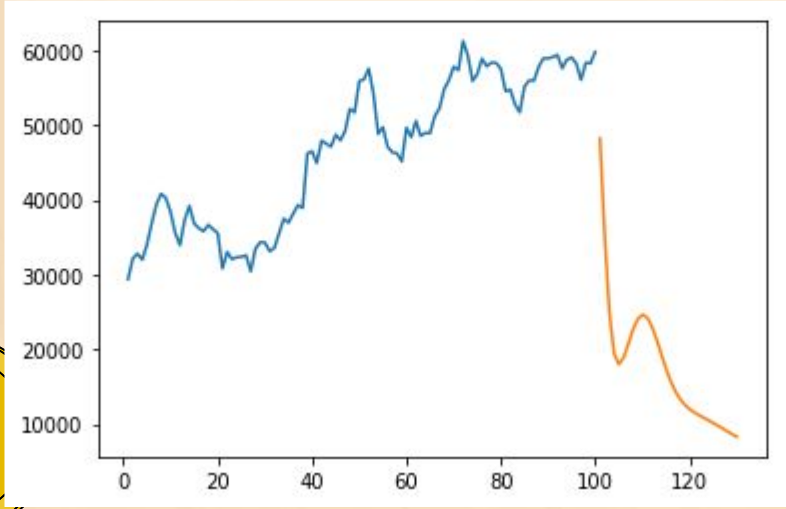



THB
10.75 (43.07%) , 43
-22 bars, -31d
Vol 935.857M
37.00
36.50
36.00
35.75
35.50
35.00
34.50
34.00
33.50
33.00
32.50
32.00
31.50
31.00
30.75
30.50
30.00
29.75
29.50
29.00
28.50
28.00
27.50
27.00
26.75
05:24:18
26.00
25.50
25.00
24.50
24.00
23.50
23.00
Oct
12
19 Oct '20
Nov
19 Nov '20
Dec
21
2021
08 Jan '21
18
Feb
TradingView
11:50:40 (UTC+7)
adj
%
log
auto

3

Final Remark


ANNA COULLING
A Complete Guide to Volume Price Analysis
Read the book then read the market


60000
50000
40000
30000
20000
10000
0
20
40
60
80
100
120


−4998.92 (−8.58%) , −499892
30 bars, 30d
Vol 146.127K