HOW TO BECOME RICH 101 trading crypto

not financial advice

by : Christopher Perez, Christy Dain, Jamison Gardner, Jackie Bai

Hmmm... how could I use machine learning to create my own crypto trading bot??

-If only there was a guide

Motivation and summary

-Machine learning is a powerful tool that could be applied to crypto trading in determining market trends and taking out emotional decision making. However, which machine learning models is the best to predict market conditions? Our mission is to analyze different machine learning models' performances to determine which model could make you a crypto trading genius tomorrow.

Using ARIMA to predict the future...

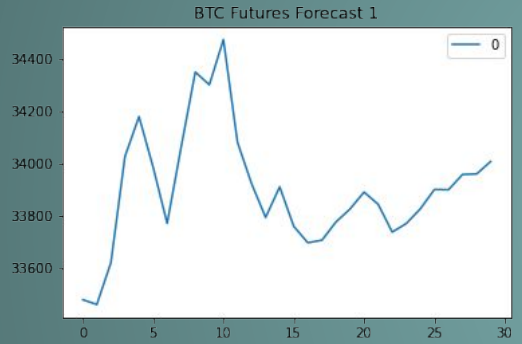
Using *ARIMA(AutoRegressive Integrated Moving Average)* to analyze and predict the future prices of several different cryptocurrencies with data from the past 6 1/2 months, from January 1st of this year.

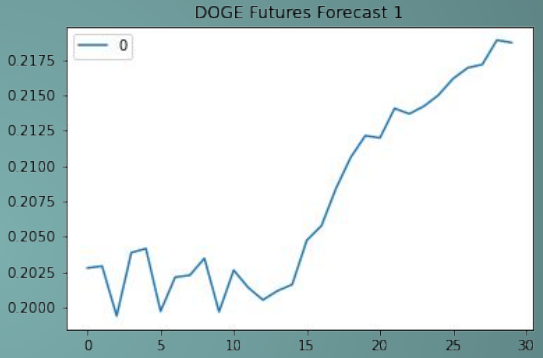
...from the past!

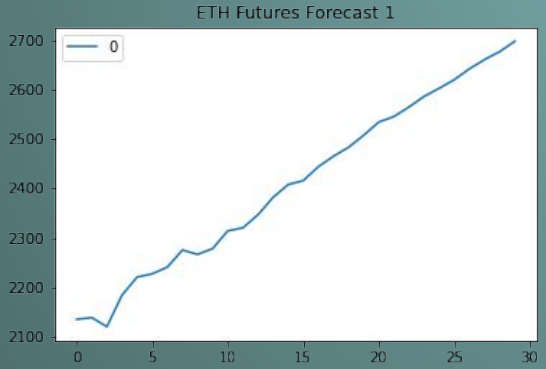
Data Source :

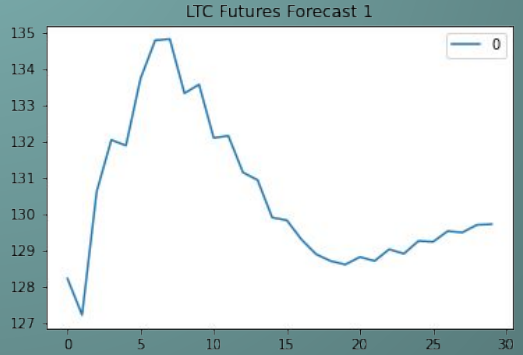
Toolkits used:

Trustworthy information?
ARIMA Results









LSTM(Long Short Term Memory networks)

- **A type of recurrent neural network**

-Pros

- Take advantage of thinking like our human brain does
- Does Not learn from zero each time as it remembers a set amount of data in its training
- Can be used for wide applications
- It finds patterns in data that may be complex

-Cons

- Require lots of data to get well trained model
- Have to balance out re,bering irrelevant data and not knowing enough of what has happened in the past
- Its difficult to prepare the data for the model

LSTM sounds interesting but how does that make me rich ????

- Hear me out.. what if we applied LSTM to the prices of CRYPTO to forecast the future PRICES OF CRYPTOCURRENCY?!?!
- That's exactly what our team did in our search for finding the best machine learning model to trade crypto !

I’m not going to keep you waiting... so i'll show you the results first and then explain after

ETH - The total profit/loss of the trading model is $3743722.57, with the total return on investment being 3743.72%

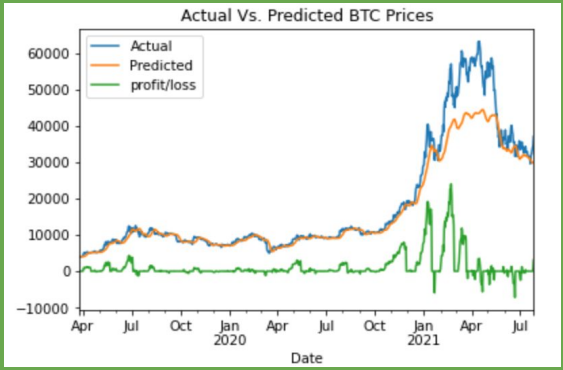
BTC - The total profit/loss of the trading model is $106421589.11, with the total return on investment being 106421.59%

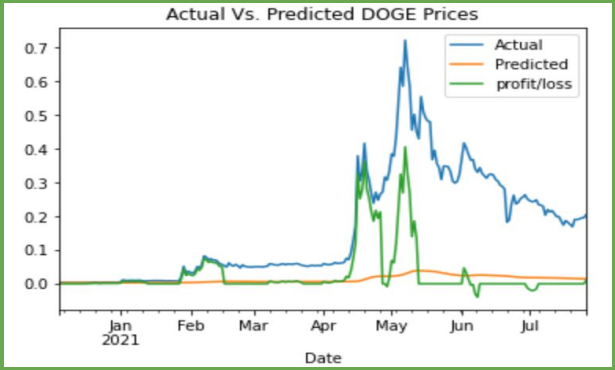
DOGE - The total profit/loss of the trading model is $667.2, with the total return on investment being 0.67%

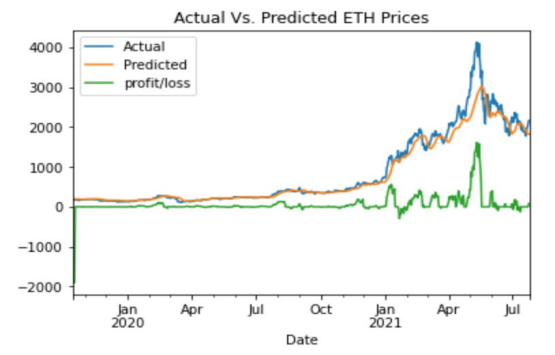
LTC - The total profit/loss of the trading model is $319289.0, with the total return on investment being 319.29%

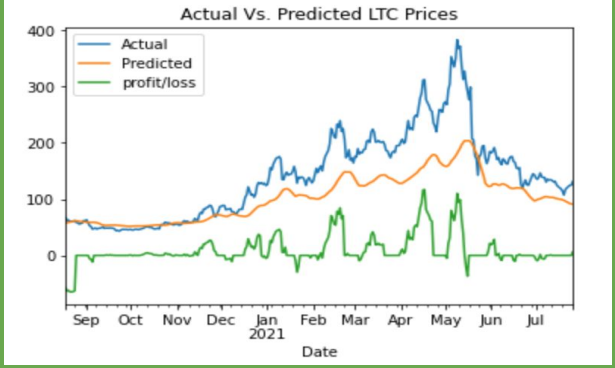
BCH- The total profit/loss of the trading model is $1279881.08, with the total return on investment being 1279.88%

Before I go into data preparation, let me show you the price predictions









Now Data preparation

```
[8]: # Define the window size
window_size = 9

# Set the index of the feature and target columns
feature_column = 2
target_column = 2

# Create the features (X) and target (y) data using the window_data() function.
X, y = window_data(eth_df, window_size, feature_column, target_column)

# Print a few sample values from X and y
print (f"X sample values:\n{X[:3]} \n")
print (f"y sample values:\n{y[:3]}")
```

```
# Manually splitting the data
split = int(0.7 * len(X))

X_train = X[: split]
X_test = X[split:]

y_train = y[: split]
y_test = y[split:]
```

```
[11]: # Reshape the features data
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Print some sample data after reshaping the datasets
print (f"X_train sample values:\n{X_train[:3]} \n")
print (f"X_test sample values:\n{X_test[:3]}")
```

```
[8]: #assign atributes to names
#future period desides how many days ahead data will be
FUTURE_PERIOD_PREDICT = 3
```

```
[9]: #set data 3 days back to compensate for lag
def add_future_column(eth_df):
    main_df= eth_df.copy()
    main_df['future'] = eth_df["Closing Price (USD)"].shift(-FUTURE_PERIOD_PREDICT)

    main_df.dropna(inplace=True)

    main_df.head()

    return main_df
```

Trading evaluation

[23]:
```
#create buy and sell signals
#calculate profit and loss

#create 'profit/loss' column to track trade metrics
eth_eval['profit/loss'] = np.nan

#create column to hold buy and cell signals
eth_eval['signals'] = np.nan

#create buy and sell list containers
buy = []
sell = []

#create column that has next days price
eth_eval['next_day'] = eth_eval["Predicted"].shift(-1)
```

```
#create loop that buys if next day price is higer and sells if next day price goes down
for index, row in eth_eval.iterrows():

    if row["Predicted"] > row["next_day"]:
        eth_eval.loc[index, "signals"] = "buy"
        buy.append(row["Actual"])
        eth_eval.loc[index, "profit/loss"] = 0
    elif row["Predicted"] < row["next_day"]:
        eth_eval.loc[index, "signals"] = "sell"
        sell.append(row["Actual"])
        eth_eval.loc[index, "profit/loss"] = sell[-1] - buy[-1]
    else:
        eth_eval.loc[index, "signals"] = "hold"
```

```
# calculate total profit/loss and percent return for 1000 total coins

# total amount of initial capital
initial_capital = 100000

# set total amount of coins
coin_order = 100

# calculate total profit/loss
total_profit_loss = round(eth_eval["profit/loss"].sum() * coin_order, 2)

# calculate return on investemnt
roi = round((total_profit_loss / initial_capital) * 100, 2)

# display profit/loss and roi
print(
    f"The total profit/loss of the trading model is ${total_profit_loss}, "
    f"with the total return on investment being {roi}%"
)
```

Building the LSTM model with 3 layers

```
# Define the LSTM RNN model.
model = Sequential()

# Initial model setup
number_units = 9
dropout_fraction = 0.2

# Layer 1
model.add(LSTM(
    units=number_units,
    return_sequences=True,
    input_shape=(X_train.shape[1], 1))
    )
model.add(Dropout(dropout_fraction))

# Layer 2
model.add(LSTM(units=number_units, return_sequences=True))
model.add(Dropout(dropout_fraction))

# Layer 3
model.add(LSTM(units=number_units))
model.add(Dropout(dropout_fraction))

# Output layer
model.add(Dense(1))
```

[16]:
```
# Train the model
model.fit(X_train, y_train, epochs=10, shuffle=False, batch_size=90, verbose=1)
```

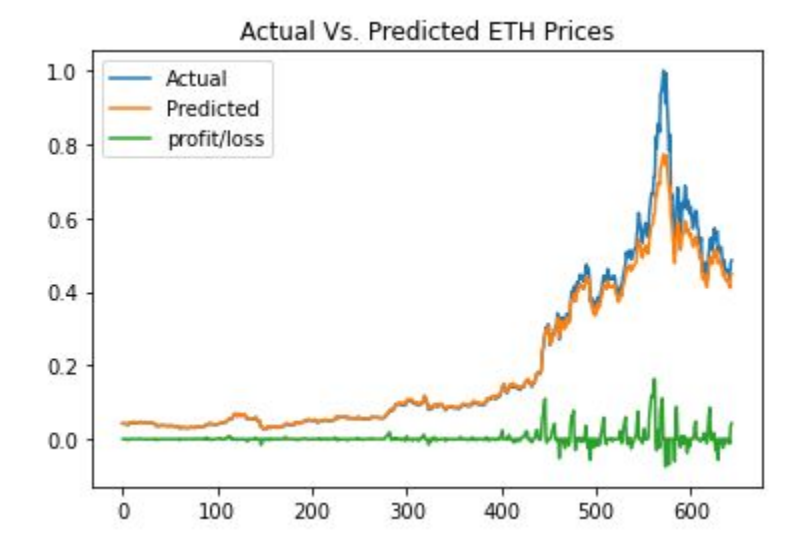
```
Epoch 1/10
17/17 [==============================] - 5s 8ms/step - loss: 0.0034
Epoch 2/10
17/17 [==============================
```

```
# Define the window size
window_size = 9

# Set the index of the feature and target columns
feature_column = 2
target_column = 2

# Create the features (X) and target (y) data using the window_data() function.
X, y = window_data(eth_df, window_size, feature_column, target_column)

# Print a few sample values from X and y
print (f"X sample values:\n{X[:3]} \n")
print (f"y sample values:\n{y[:3]}")
```

[22]:
```
#seprate actual and pred
Pred= eth_eval['Actual']
act= eth_eval['Predicted']

#model eval
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
import math
print(mean_squared_error(act, Pred))
print(math.sqrt(mean_squared_error(act, Pred)))
print(mean_absolute_error(act, Pred))
```

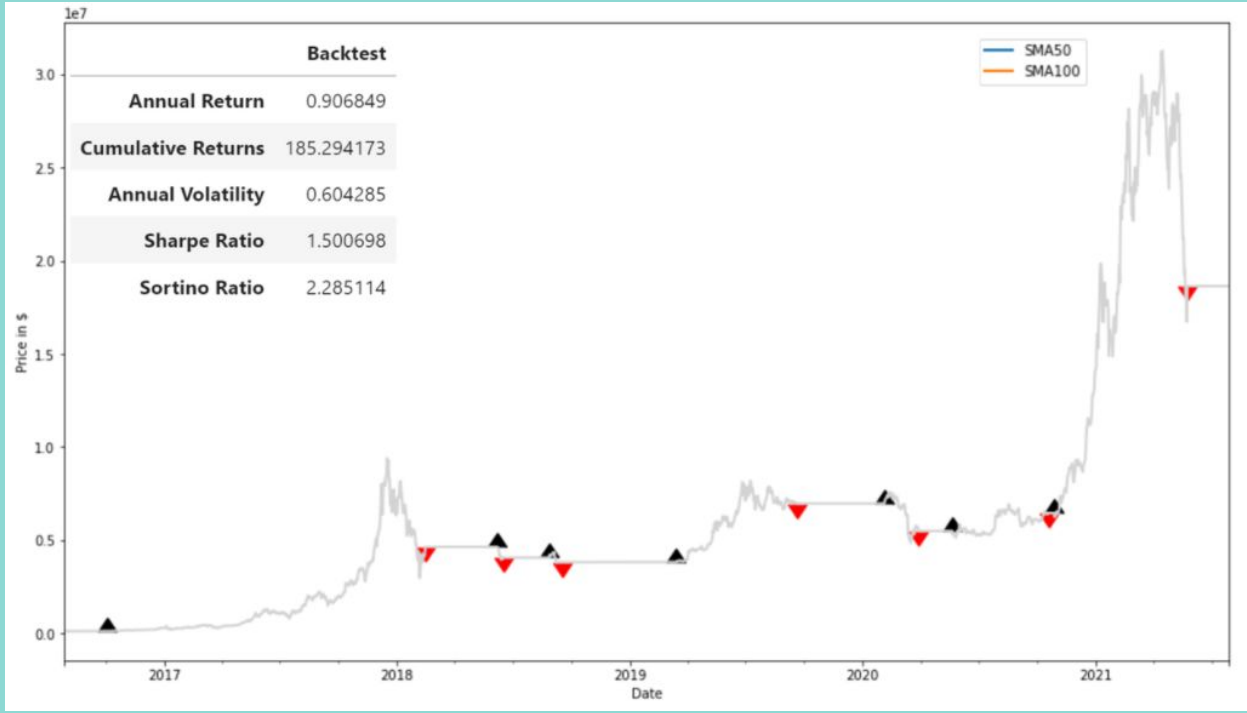
```
46647.24108071358
215.97972377219483
102.59572350824543
```

Conclusion of LSTM



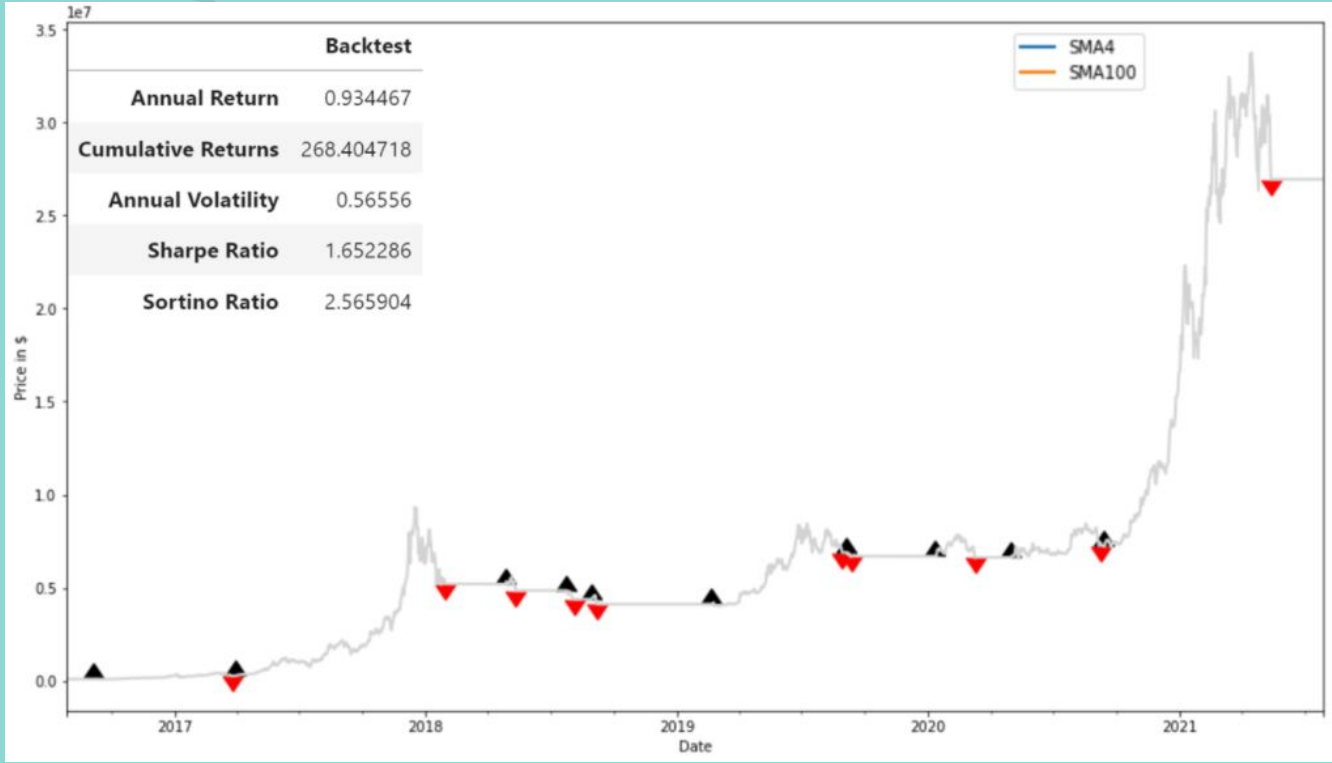
- The results of lstm models show promise
- They are far from perfect
- In theory could prove to be a great trading strategy

Algotrading for cryptocurrency long position



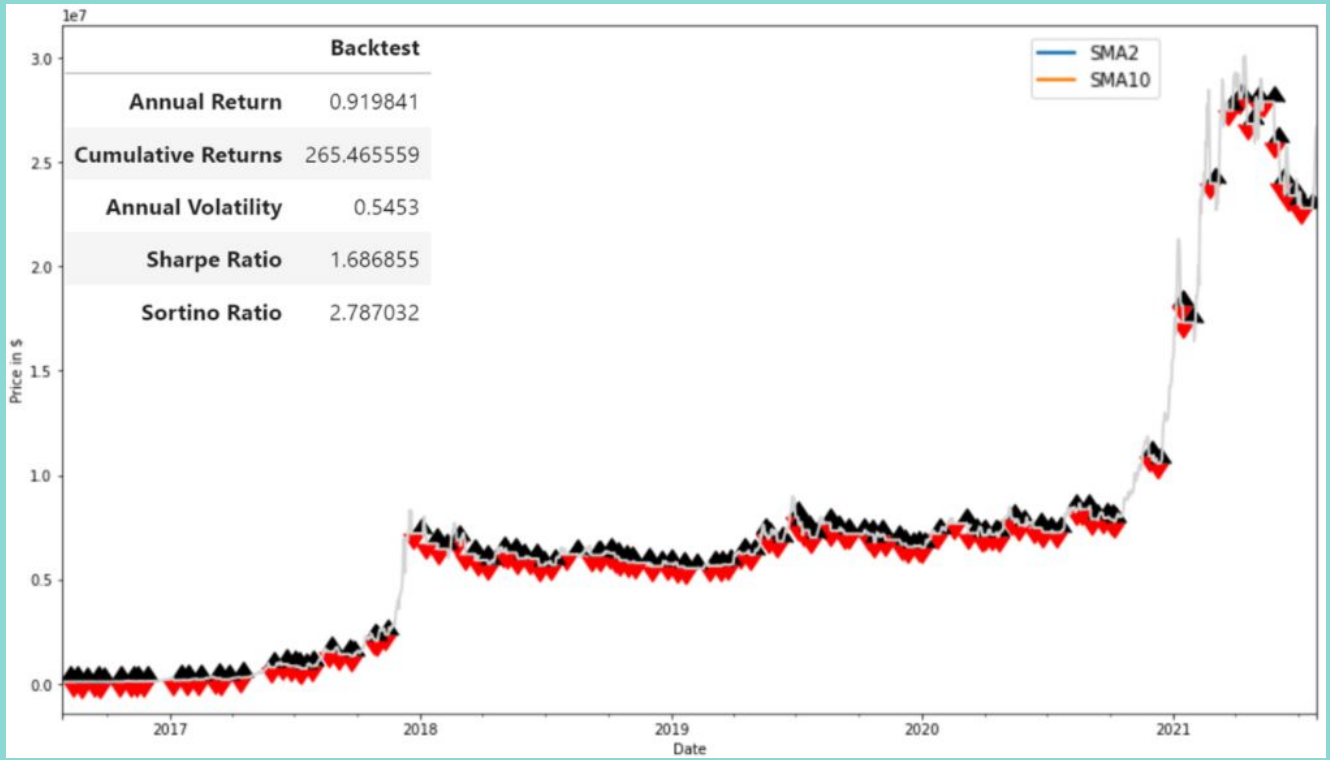
With the 50-day 100-day moving average algorithm, BTC gives us a cumulative return of 185 times the initial investment from 2016 up to current. The sharpe ratio is 1.5, and the sortino ratio is 2.3 which are acceptable.

4-day 100-day SMA crossover strategy



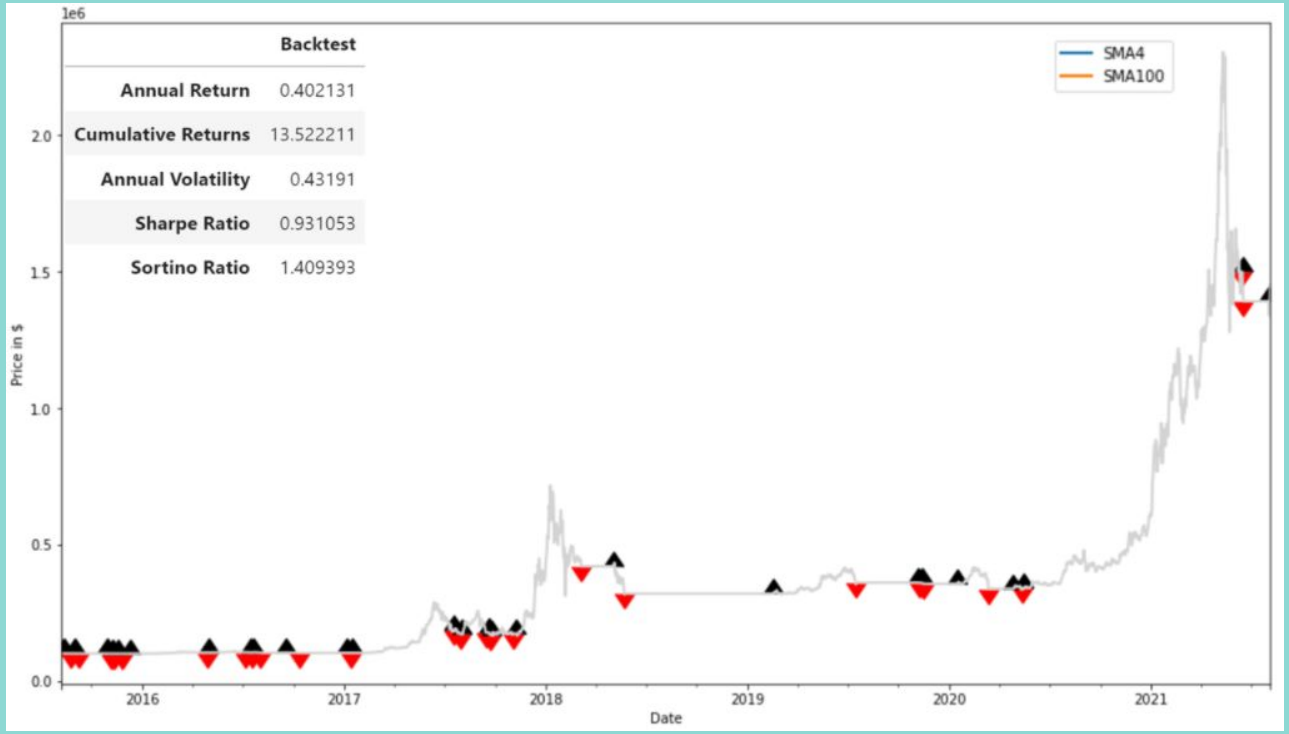
when we shorten the short-term days to just 4 days, we can sell at a better price, and the cumulative return is significant higher. Here we have a cumulative return of 268 times our initial investment for the the 4-day 100-day moving average crossover algorithm. The sharpe ratio and the sortino ratio are also better.

2-day 10-day SMA crossover strategy



The trading frequency is high, but the cumulative return and the metrics are almost the same as the 4-day 100-day moving average algorithm. (a cumulative return of 265 times the initial investment)

4-day 100-day SMA crossover strategy for ETH

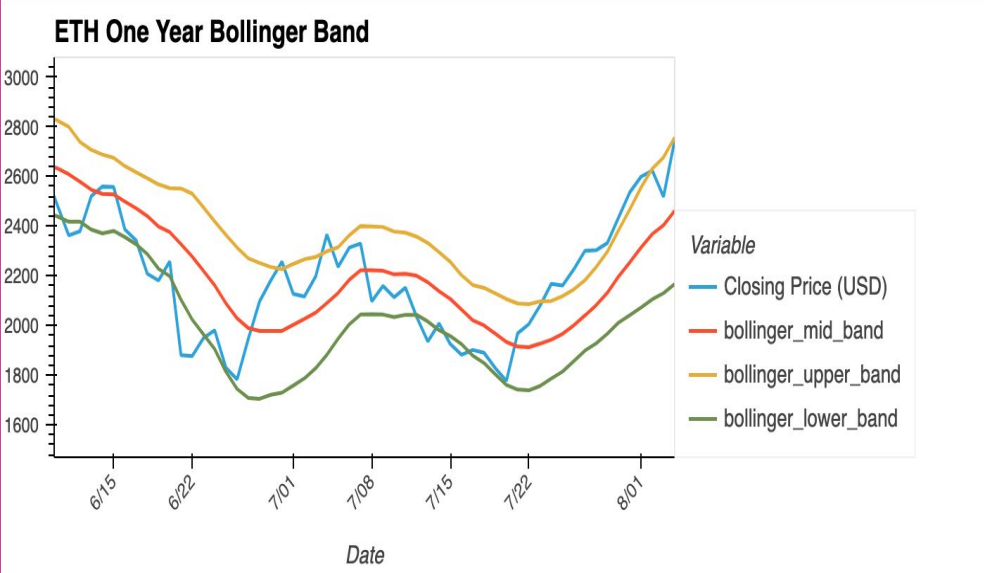


Backtest results of the 4-day 100-day moving average algorithm on ETH from 2015 up to current, it turns out the reutrns and the metrics are not as good as BTC.

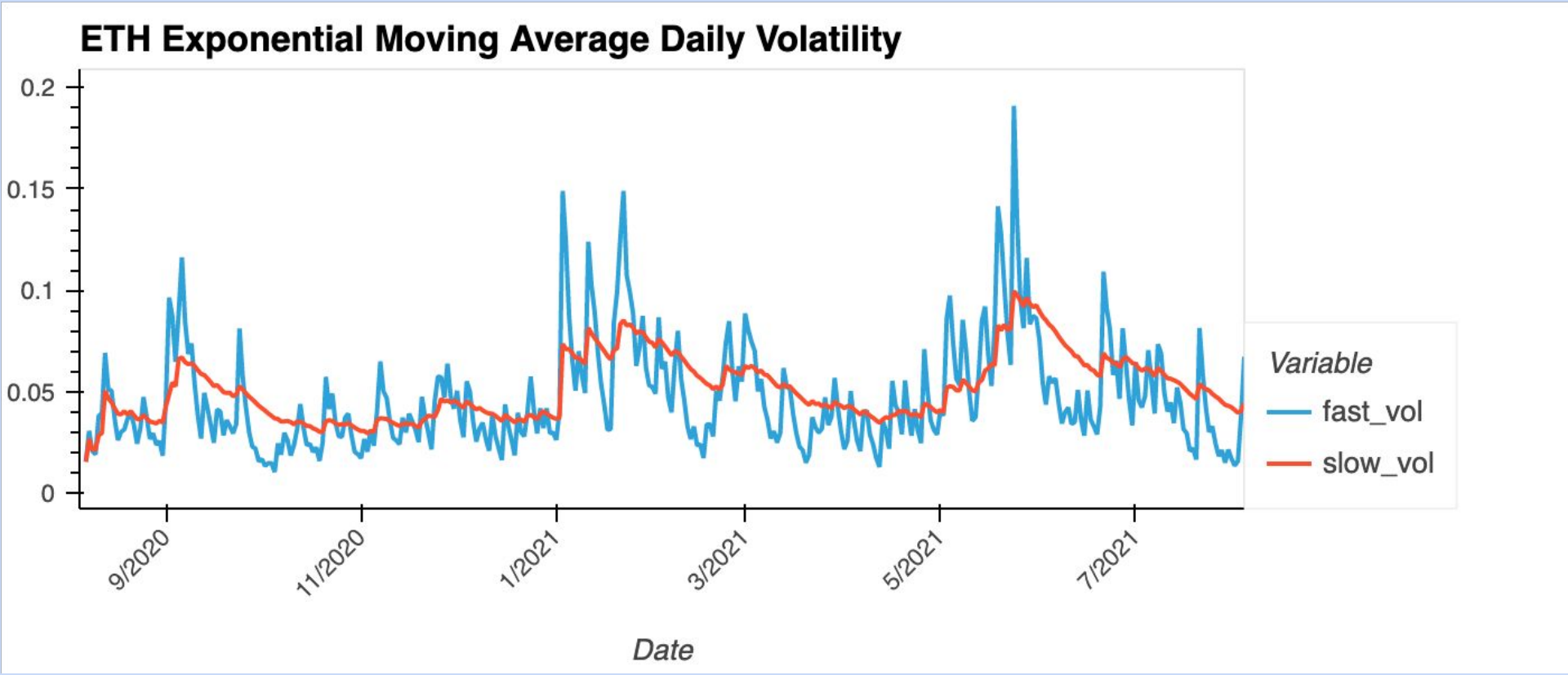
Exponential Moving Average
Daily Volatility
Bollinger Band

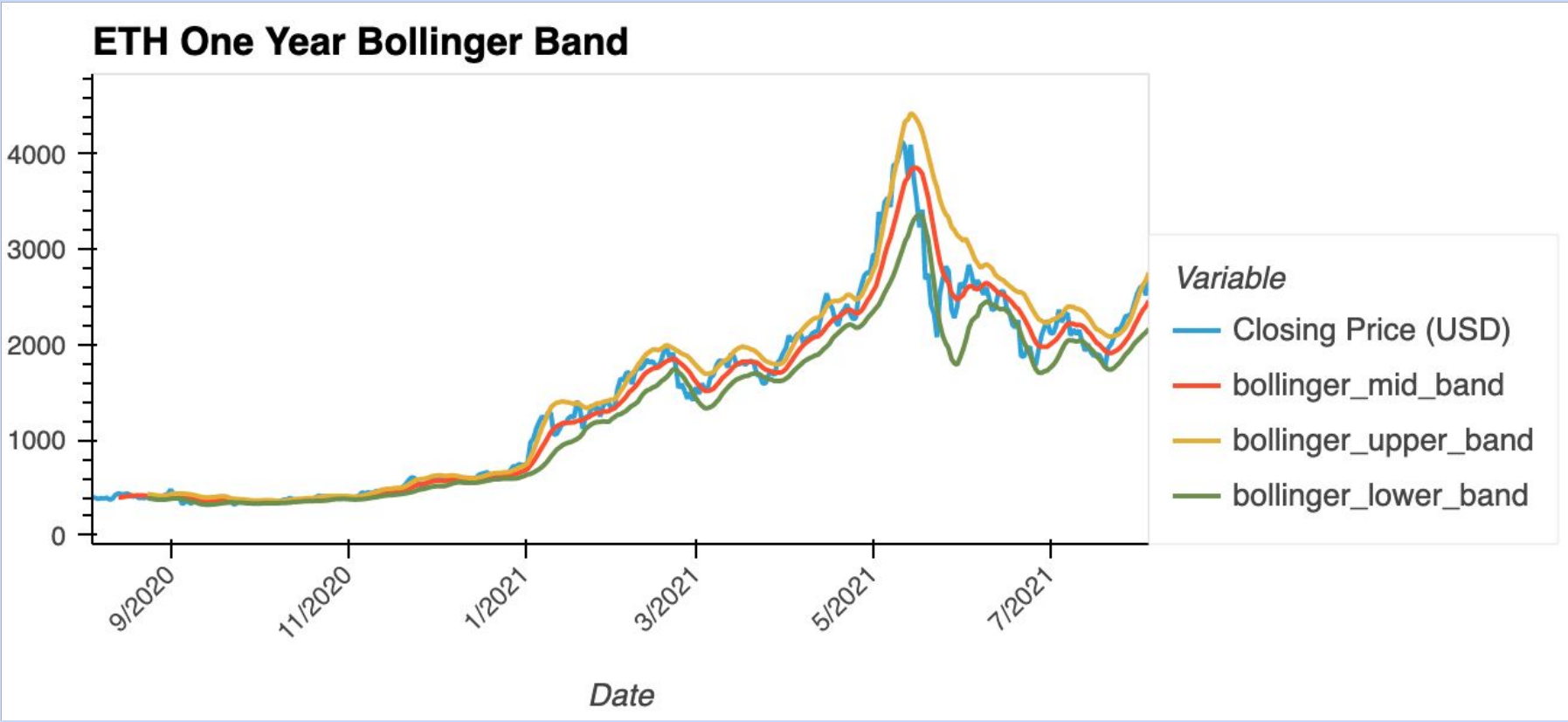
For both Bitcoin and Ethereum at a window of 10, the market prices were closely following the 20 day moving average at 2% standard deviation.

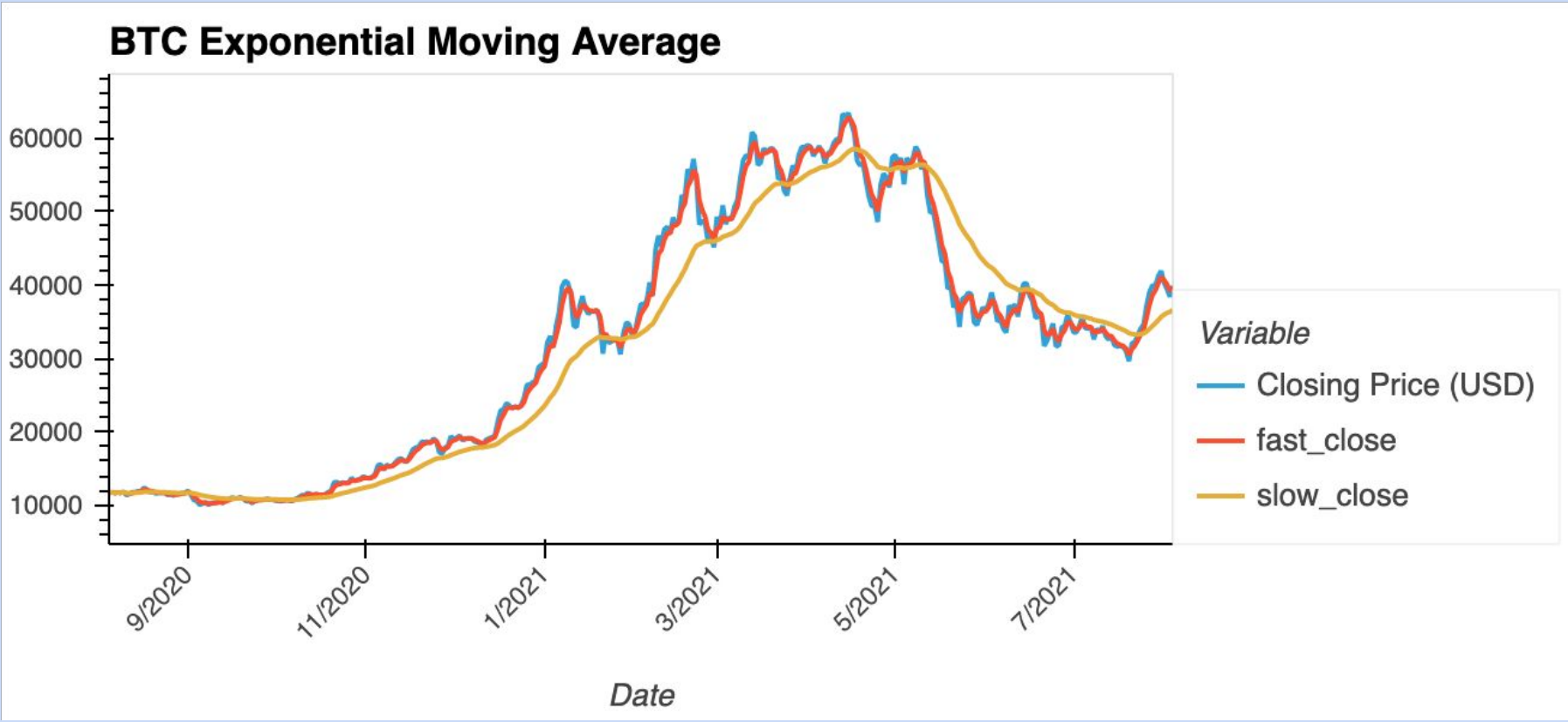
In mid May, Ethereum was over bought. So one would expect a reversion of the price to move back to the lower end of the Bollinger Band. In the end of June the market was oversold and the Bollinger Band was expanded which led to the reversion of the previous trend.

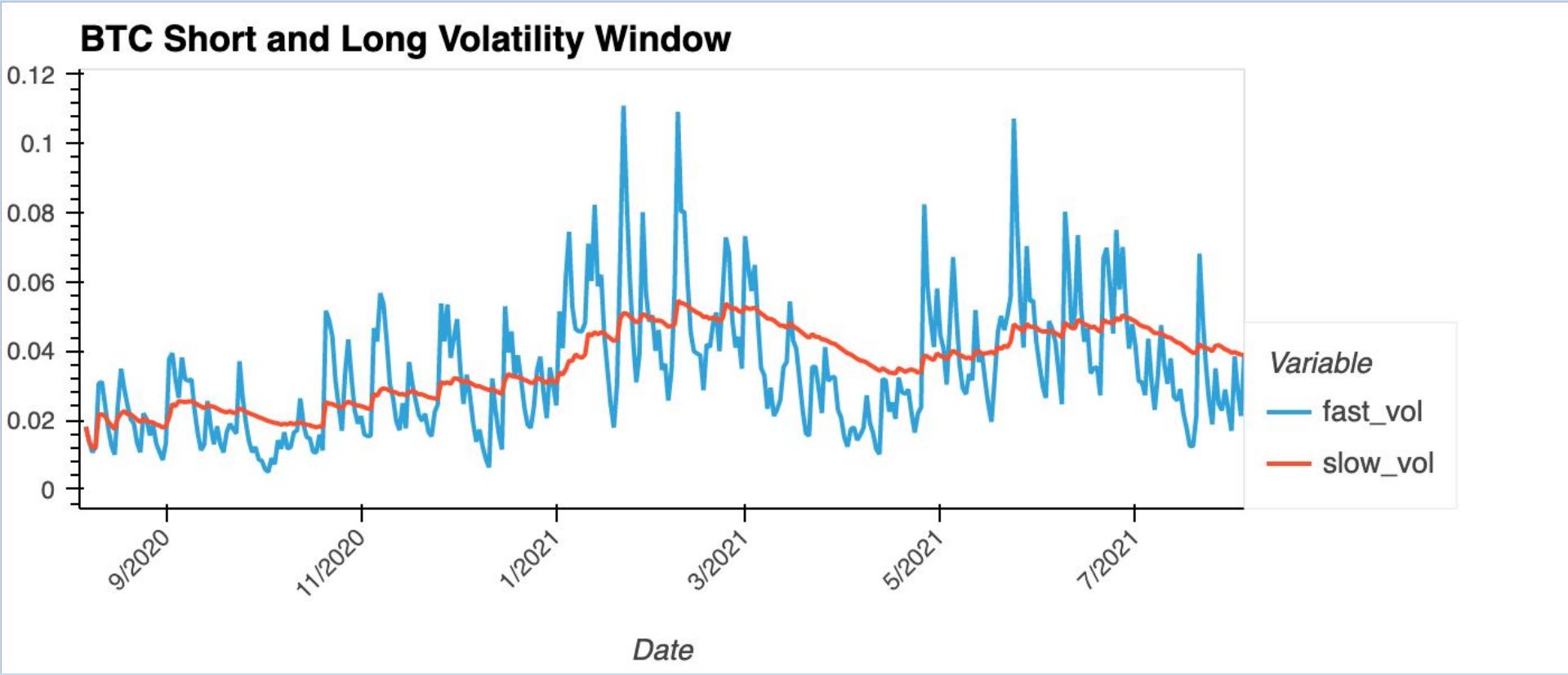


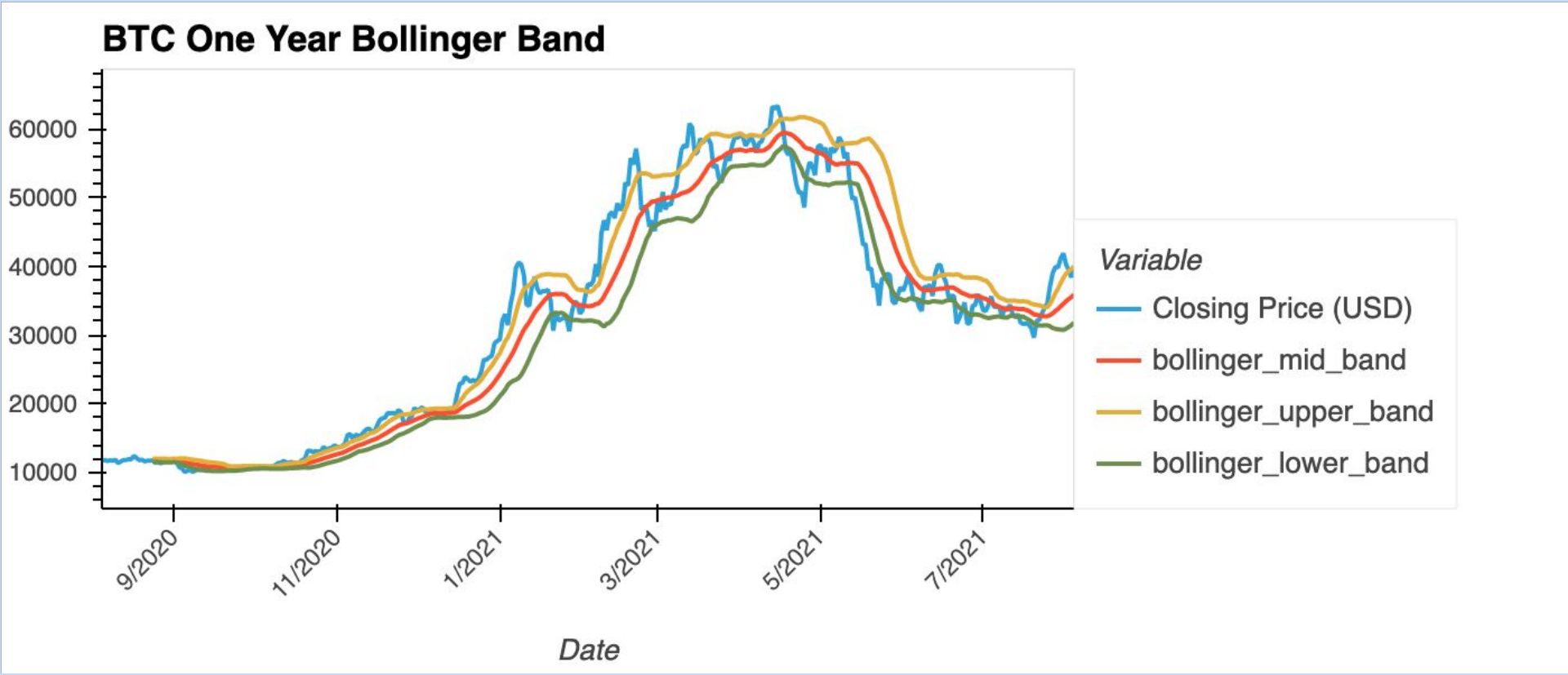
Currently, the price is on the upper side of the 20 day moving average. There is still potential, however, the margin of gain could be lowered.


ETH Exponential Moving Average Daily Volatility
0.2
0.15
0.1
0.05
0
9/2020
11/2020
1/2021
3/2021
5/2021
7/2021
Date
Variable
fast_vol
slow_vol


ETH One Year Bollinger Band
4000
3000
2000
1000
0
9/2020
11/2020
1/2021
3/2021
5/2021
7/2021
Date
Variable
Closing Price (USD)
bollinger_mid_band
bollinger_upper_band
bollinger_lower_band


BTC Exponential Moving Average
60000
50000
40000
30000
20000
10000
9/2020
11/2020
1/2021
3/2021
5/2021
7/2021
Date
Variable
Closing Price (USD)
fast_close
slow_close


BTC Short and Long Volatility Window
0.12
0.1
0.08
0.06
0.04
0.02
0
9/2020
11/2020
1/2021
3/2021
5/2021
7/2021
Date
Variable
fast_vol
slow_vol


BTC One Year Bollinger Band
60000
50000
40000
30000
20000
10000
9/2020
11/2020
1/2021
3/2021
5/2021
7/2021
Date
Variable
Closing Price (USD)
bollinger_mid_band
bollinger_upper_band
bollinger_lower_band

Questions ?