Title

Bitcoin Price Analysis

Authors

Rohit Gampa(rgampa@iu.edu), Mitali Tavildar(mtavilda@iu.edu)

Abstract

The idea behind bitcoin price analysis is to ask very interesting questions regarding the price of bitcoin as well as to look at the analysis from an investor perspective to see if there are opportunities that can be observed from the historical bitcoin price data.

Keywords

Bitcoin, Binance, time-series, analysis, Cryptocurrency, trading, investment, backtest, prophet, python, tensorflow, LSTM, dropout, google trends, pytrends.

Introduction

We want to be able to analyze price data to gain understanding on how the price changes over a period as well as understanding the effect of google searches on the price via google trends data.

For this we had looked at traditional stocks as well as cryptocurrencies, to choose which will be better for analysis.

We have chosen to analyze bitcoin price due to the following reasons:

- Traditional Stocks trade only for 250 days in a year, for 8 hours a day Vs Cryptocurrencies like bitcoin trade 24 hours a day, 7 days a week.
- Good Quality historical data for the traditional markets is hard to get and is expensive as well, Eg: Yahoo finance has data for stocks daily candle data, but it has been known to have issues sometimes in Data Quality.
- Also getting data for traditional stocks for a hourly candles seems to be a very costly, as we need to pay a monthly subscription to get this kind of data.
- After filtering out the traditional stock market, we looked at various cryptocurrencies and choose bitcoin due to its highest volume of trades and it being a popular cryptocurrency.
- We choose bitcoin as it is known to almost everyone , and it makes it easier to get google trends data.

1.Methods

1.a Getting the hourly and Daily Bitcoin Data

Using Binance API 'https://api.binance.com/api/v3/klines' we created a custom code ‘download_candle_stick_data.py’to download the bitcoin data, as like all companies binance does not allow to download a large amount of data, as it can cause server load issues. To circumvent this we are not using any tokens generated by binance, as we do not need any

proprietary information from the account and created a loop for all the historical data we want, since we want the data from the beginning of the exchanges, we loop though this api with small chunks of time to get the data and we append this data to the main dataframe till we get the latest information, then the dataframe is stored as a csv.

This file can then be used to load the data into python for analysis. We used the above code to download the historical data for the training pair "BTC-USDT", as this has the highest volume of trades.

1.b. Exploratory data analysis

To EDA, we considered the records that described the highest price, lowest price, open time, close time, open price, and close price for each day.

The preprocessing steps:

- Converting the columns consisting of dates from Unix timestamp to date time format.
- Creating customized columns based on data: -
- To analyze the close price and open price for each day.
- To analyze the open price and the highest price for each day.

Using the customized columns, we use the below mentioned formula to calculate gain each day:

Gain(at each day) = Close price(at each day) – Open price(at each day)
If Gain > 0, Gain_Achieved = TRUE else,
Gain_Achieved = FALSE

Using the customized columns, the correlation between Gain and difference between (open and highest prices) was inferred to have a value of 0.60.
This provided an insight that the Gain and increase in highest rate for the day greater than open rate in positively correlated, increase in one could possibly increase the probability of increase the other.

1.c. Predicting gain using classic algorithms:

Feature columns: open, low, high of the day
Label column: Gain
i. Logistic regression
Using the logistic regression as a baseline classification algorithm, we fitted our data to the model to predict if there is a probability of achieving gain at the end of the day.
ii. KN neighbors classifier
looking at the open,high and low prices we create a classifier predict the gain for today(daily data)/hour(hourly data)

2.c. Trend Analysis using Prophet

Prophet is an open-source algorithm for generating time-series models that uses a few old ideas with some new twists. It is particularly good at modeling time series that have multiple seasonalities and doesn't face some of the above drawbacks of other algorithms.[1]

For trend analysis and prediction, records that described the highest price, lowest price, open time, close time, open price and close price for every hour were utilized.
The Fig3. Shows the general trend followed by our data.
Following the steps mentioned in the Prophet documentation[2], we fit our data into the Prophet model. The resultant trend prediction is displayed in Fig5, the Prophet was able to catch the upward trend and predict the same.
Prophet.plot_components method was used to see the trend, yearly seasonality, and weekly seasonality of the time series. Fig4 displays the output from Prophet.

1.d Getting google trends Data.

For getting google trends data, I have use the pytrends package. I have downloaded two types of Google trends data using this method.

1) Hourly breakup of google search data
2) Daily breakup of google search data.

In pytrends methods, we need to input the duration for which we need the data, for what type of google search , as well as the geographic information. Since we need the google trends data for the time we have historical data, I have downloaded data from 2017 August to present, setting the search to only google searches, not including image searches, and setting the geographic to "", which stands for global search volumes.
The code below should be able to explain how the data is downloaded.

```
google_trends_list = ['bitcoin chart',
 'bitcoin dollar',
 'bitcoin dolar',
 'bitcoin news',
 'bitcoin mining',
 'btc',
 'bitcoin usd',
 'bitcoin',
 'bitcoin to usd',
 'bitcoin price usd',
 'bitcoin stock',
 'bitcoin wallet',
 'bitcoin value',
 'cryptocurrency',
 'coinbase',
 'miner',
 'price bitcoin',
 'sell bitcoin',
 'sell btc',
 'what is bitcoin',
 'buy bitcoin']
# download daily trends data
from pytrends import dailydata
for topic in google_trends_list:
  print(topic)
  dailydata.get_daily_data(topic, 2017, 8, 2021, 12, geo = '').to_csv("google trends data/"+topic+'.csv')
```

```
#to download the google trends hourly data
google_trends_list = ['buy bitcoin','sell bitcoin']
from pytrends.request import TrendReq

pytrends = TrendReq(hl='en-US', tz=360)

pytrends.get_historical_interest(google_trends_list, year_start=2017,month_start=8,
                                 day_start=1,hour_start=0, year_end=2017,month_end=12, day_end=31, hour_end=23,cat=0, geo='', gprop='', sleep=0).to_csv('hourly_trends_data_2017.csv')
pytrends.get_historical_interest(google_trends_list, year_start=2018,month_start=1,
                                 day_start=1,hour_start=0, year_end=2018,month_end=12,day_end=31, hour_end=23,cat=0, geo='', gprop='', sleep=0).to_csv('hourly_trends_data_2018.csv')
pytrends.get_historical_interest(google_trends_list, year_start=2019,month_start=1,
                                 day_start=1,hour_start=0, year_end=2019,month_end=12, day_end=31, hour_end=23,cat=0, geo='', gprop='', sleep=0).to_csv('hourly_trends_data_2019.csv')
pytrends.get_historical_interest(google_trends_list, year_start=2020,month_start=1,
                                 day_start=1,hour_start=0, year_end=2020,month_end=12, day_end=31, hour_end=23,cat=0, geo='', gprop='', sleep=0).to_csv('hourly_trends_data_2020.csv')

pytrends.get_historical_interest(google_trends_list, year_start=2021,month_start=1,
                                 day_start=1,hour_start=0, year_end=2021,month_end=12, day_end=31, hour_end=23,cat=0, geo='', gprop='', sleep=0).to_csv('hourly_trends_data_2021.csv')
```

For downloading the daily google trends data, I cut the total time we needed the data into years and downloaded them individually, as the google api's were giving timeout errors.

In google trends data, to understand the sentiment of the search we look at keywords that contain that sentiment. I have selected 2 keywords "Buy Bitcoin" and "Sell Bitcoin" to be the keywords as they have the most searches, and when comparing the other search terms they have very low volumes compared to the above mentioned 2, since there are not may searches the data is not representative of the search effect , thus not included for analysis.

1.e Preprocessing the Data

We have combined the Data from Google trends and Bitcoin Price using the Date Column in google trends data and open_time column in bitcoin price data.

Before merging the data for daily bitcoin price, we need to contact the google trends data for all the years into one big dataframe.

As mentioned in EDA before, the open_time column in binance comes in epoch/unix time, using pandas we have converted that to string for visualization and merging of data.

Before we divide the dataset into training and testing dataset, Since we know that the open and close are really the same features, and also we are using the model to predict whether there is gain in that day, we will convert the close column to act as the isgain column which is true if close price is greater than open price and False for the other case.

Since we are working with time series data, we have divided the first 70% of the data as training data and the rest as testing data.

Using MinMaxScaler() we scale down the data to 0,1, since the LSTM model works better with scaling. We fit the MinMaxScaler() using the training data, as we do not want to introduce any future values during the training time.

Since we need to feed a history of the window time to get the output, we have chosen the window to be 60, and converted the timeseries data into the format required for the LSTM input layer for the training and testing datasets.

1f. Deep learning model using LSTM

We are using 3 LSTM layers with a final dense layer for the output. This model is the best of all the architectures tested out(4 layers of LSTM and 1 layer of LSTM).

We are classifying whether the price of close will be higher or lower than the open price, using binary classification.

The loss function taken is BinaryCrossentropy from the tensorflow and adam was chosen to be the optimizer.

For activation we are using the default for LSTM which is tanh.

Here is the model architecture, printed using the model.summary in the tensorflow package.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_9 (LSTM)               (None, 60, 60)            17040

 dropout_9 (Dropout)         (None, 60, 60)            0

 lstm_10 (LSTM)              (None, 60, 120)           86880

 dropout_10 (Dropout)        (None, 60, 120)           0

 lstm_11 (LSTM)              (None, 60)                43440

 dropout_11 (Dropout)        (None, 60)                0

 dense_3 (Dense)             (None, 1)                 61

=================================================================
Total params: 147,421
Trainable params: 147,421
Non-trainable params: 0
```

The input to the model contains history of the window for columns open, high,low, volume, quote_asset_volume, number_of_trades, taker_buy_base_asset_volume, taker_buy_quote_asset_volume, ignore,buy bitcoin, sell bitcoin columns. The last two columns are the google trends seach volumes.

I have created replica of the model designed above, where the only difference is that into one model, I am including the google trends columns and in other I am removing them, to benchmark the effect of google trends data on the prediction.

Results:

Results for EDA by Comparison using visualization:

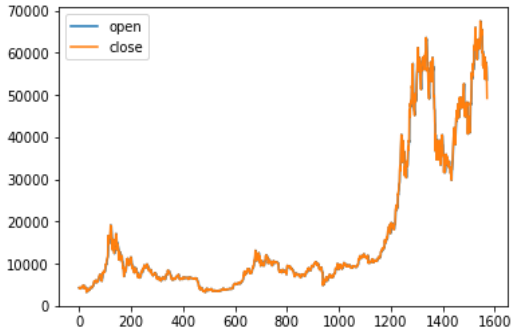

Fig1. Prices at open time vs prices at close time at each day

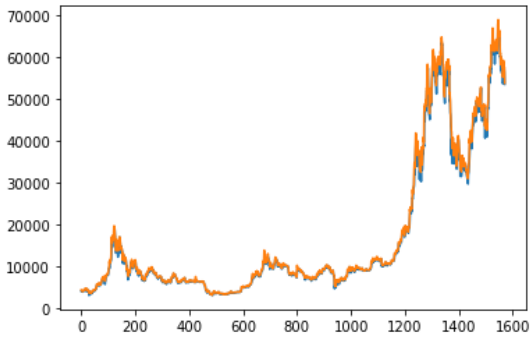

Fig2. Prices at open time vs highest price each day

We observe the overlap in the graphs for both the open price and close graph each day Fig1 with a lag of 1. The prices are continuous, which means the closing rate from previous record is same as the opening rate for the next record, this helps in feature selection. As compared to traditional stock market the open price of the stock next day changes from the previous day's close.
There is visible difference in the open price and highest price of the day Fig2. We see that the highest price (orange line graph) has risen above the price. This shows the volatility of the price of bitcoin.

We observed that if a person invested at the start of the day and sell by the end of the day, he would have made 53% profitable trades.

Rather than using the above strategy of investing all day, if we created a model, which helps the person investing know if the following day will be profitable of not, for this we created a logistic regression model, and a KNN model as specified previously in methods.

Here are the results:

1) Accuracy of 87% achieved from logistic regression
2) Accuracy of 71% achieved from KNN method.

From the above results we conclude that Logistic regression method performed the best.

Results for the Trend Analysis using Prophet.

Using the outputs, analysis on general trend was done.

- The Fig4 clearly displays a drop in trend during Fridays, which suggests approximately the best time to buy.
- Similarly, the trend suggests the best time to sell is Monday.
- By looking at the trend for the day, we see that 0100 is the best time to buy bitcoin and 2000 is the best time of the day to seel bitcoin.

Looking into the trend of bitcoin, we see that the trend is increasing which means currently we are in a bitcoin bull market.

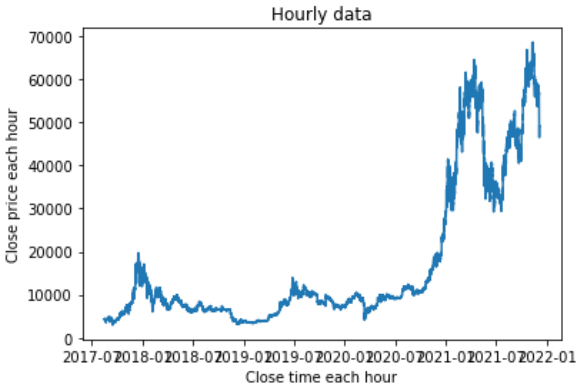

Fig3. Prices at close time at each hour of the day

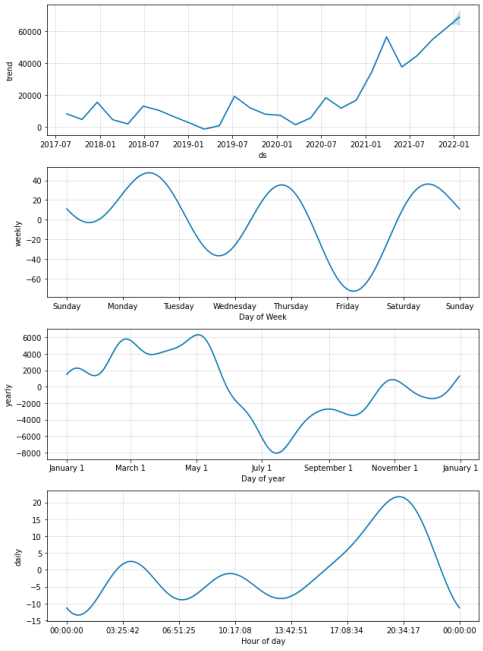

Fig4. Trend analysis and prediction for varied time components

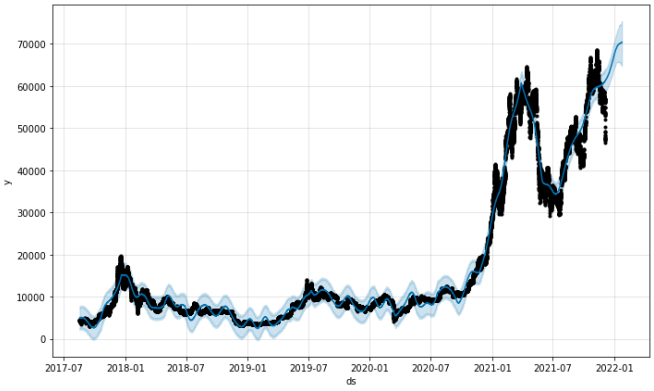

Fig5. Trend prediction by Prophet

Results for Effect of google trends data on LSTM model.

I have created models for both hourly and daily data, the training and testing accuracies of the models are close to each other, but the model with google trends data converged faster.

```
Epoch 1/100
758/758 [==============================] - 81s 100ms/step - loss: 0.6932 - accuracy: 0.4874
Epoch 2/100
758/758 [==============================] - 78s 103ms/step - loss: 0.6918 - accuracy: 0.4873
Epoch 3/100
758/758 [==============================] - 81s 107ms/step - loss: 0.6883 - accuracy: 0.4896
Epoch 4/100
758/758 [==============================] - 86s 114ms/step - loss: 0.6875 - accuracy: 0.4901
Epoch 5/100
758/758 [==============================] - 91s 120ms/step - loss: 0.6870 - accuracy: 0.4896
Epoch 6/100
758/758 [==============================] - 88s 116ms/step - loss: 0.6867 - accuracy: 0.4921
Epoch 7/100
758/758 [==============================] - 89s 118ms/step - loss: 0.6873 - accuracy: 0.4903
Epoch 8/100
758/758 [==============================] - 93s 123ms/step - loss: 0.6865 - accuracy: 0.4914
Epoch 9/100
758/758 [==============================] - 90s 118ms/step - loss: 0.6867 - accuracy: 0.4936
Epoch 10/100
758/758 [==============================] - 88s 117ms/step - loss: 0.6865 - accuracy: 0.4923
Epoch 11/100
758/758 [==============================] - 89s 118ms/step - loss: 0.6867 - accuracy: 0.4919
```

Google trends data as input to model

```
Epoch 1/100
758/758 [==============================] - 91s 113ms/step - loss: 0.6931 - accuracy: 0.4875
Epoch 2/100
758/758 [==============================] - 85s 113ms/step - loss: 0.6918 - accuracy: 0.4874
Epoch 3/100
758/758 [==============================] - 86s 113ms/step - loss: 0.6881 - accuracy: 0.4895
Epoch 4/100
758/758 [==============================] - 82s 108ms/step - loss: 0.6876 - accuracy: 0.4911
Epoch 5/100
758/758 [==============================] - 81s 107ms/step - loss: 0.6871 - accuracy: 0.4901
Epoch 6/100
758/758 [==============================] - 81s 106ms/step - loss: 0.6869 - accuracy: 0.4927
Epoch 7/100
758/758 [==============================] - 80s 106ms/step - loss: 0.6868 - accuracy: 0.4918
Epoch 8/100
758/758 [==============================] - 78s 103ms/step - loss: 0.6866 - accuracy: 0.4905
Epoch 9/100
758/758 [==============================] - 78s 102ms/step - loss: 0.6867 - accuracy: 0.4915
Epoch 10/100
758/758 [==============================] - 81s 107ms/step - loss: 0.6866 - accuracy: 0.4913
Epoch 11/100
758/758 [==============================] - 84s 110ms/step - loss: 0.6866 - accuracy: 0.4925
```

Model without google trends data as input

From this, we can see that the deep learning model even without google trends data is able to create a component for network effect, but when the data is directly given, we can converge the model faster.

```
13/13 - 2s - loss: 0.6974 - accuracy: 0.4647 - 2s/epoch - 123ms/step
13/13 - 1s - loss: 0.7143 - accuracy: 0.4647 - 1s/epoch - 100ms/step
```

The first line has the testing accuracy for the model without the google trends data, and the one below has the model with input which includes google trends data. We can see that the testing and prediction time is faster for the second model.

Discussion

Using the methods above, we investigated signal's seasonality to be able to get insights on when to buy and sell in the week.

The effect of google trends data was not as apparent as expected by us. We initially assumed the with the addition of google trends data, we would get better accuracy, but the results do not support that assumption.

Due to bitcoin being a volatile asset class, the model needs mode data for improving the accuracies. These are the due directions that can be looked at:

- Getting additional data from a data aggregator like coingecko or coinmarketcap.
- Adding the bitcoin fear and greed index as an input.
- Using the technical indicators from talib to do feature engineering.

References:

https://towardsdatascience.com/time-series-analysis-with-facebook-prophet-how-it-works-and-how-to-use-it-f15ecf2c0e3a

https://facebook.github.io/prophet/

https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f

https://github.com/GeneralMills/pytrends

https://binance-docs.github.io/apidocs/spot/en/#change-log

https://www.kaggle.com/odins0n/exploring-time-series-plots-beginners-guide

https://alternative.me/crypto/fear-and-greed-index/

https://github.com/mrjbq7/ta-lib