Project Based Learning Report

Create cryptocurrency analysis visualization using python pandas.

Submitted in the partial fulfillment of the requirements

For the Project based learning in (**essentials of data science**)

in

Electronics & Communication Engineering

By

Mohit Kumar Aman 2014111026
Akanchha 2014111024

Under the guidance of Course In-charge

Prof. dnyandeo lavhkare

Department of Electronics & Communication Engineering

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 4110043

Academic Year: 2021-22

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the Project Based Learning report entitled, **"** Create a cryptocurrency analysis visualization using python pandas

is work

2014111026 Mohit Kumar Aman
2014111024 Akanchha

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **create a cryptocurrency analysis visualization using python pandas** of Bachelor of Technology Semester IV, in Branch name.

Date:

Dnyandeo Lavhkare **Dr. Tanuja S. Dhope**

Course In-charge **PBL Co-Ordinator**

Dr. Arundhati A.Shinde

Professor & Head

ELECTRONICS & COMMUNICATION ENGINEERING

INDEX

Index: -

Introduction to crypto analysis

Cryptocurrencies are becoming mainstream. we have scraped together the code to download daily Bitcoin prices and apply a simple trading strategy to it.

Note that there already exists tools for performing this kind of analysis, eg. tradeview, but this way enables more in-depth analysis.

Requirements

- Python 3
- [Jupyter Notebook]
- [Pandas Data Analysis Library]
- [Bokeh interactive visualization library]
- [stock Statistics/Indicators Calculation Helper]

Getting cryptocurrency data

We have used daily Bitcoin data in USD on Bitstamp exchange.

The cryptocompare api returns following columns:

- **open**, the price at which the period opened,
- **high**, the highest price reached during the period,
- **low**, the lowest price reached during the period,
- **close**, the price at which the period closed,
- **volumefrom**, the volume in the base currency that things are traded into,
- **volumeto**, the volume in the currency that is being traded.

We have downloaded the data and store it to a file.

Trading strategy

A trading strategy is a set of objective rules defining the conditions that must be met for a trade entry and exit to occur.

We have applied Moving Average Convergence Divergence (MACD) trading strategy, which is a popular indicator used in technical analysis. MACD calculates two moving averages of varying lengths to identify trend direction and duration. Then, it takes the difference in values between those two moving averages (MACD line) and an exponential moving average (signal line) of those moving averages.

As we can see in the example below:

- exit trade (sell) when MACD line crosses below the MACD signal line,
- enter trade (buy) when MACD line crosses above the MACD signal line.

Calculate the trading strategy

We have used stockstats package to calculate MACD.

stockstats adds 5 columns to dataset:

- **close_12_ema** is fast 12 days exponential moving average,
- **close_26_ema** is slow 26 days exponential moving average,
- **macd** is MACD line,
- **macds** is signal line,
- **macdh** is MACD histogram.

Visualizing trading strategy

We have used bokeh interactive charts to plot the data.

The line graph shows daily closing prices with candlesticks. A candlestick displays the high, low, opening and closing prices for a specific period.

Below the line graph we plot the MACD strategy with MACD line (blue), signal line (orange) and histogram (purple).

Buy and Hold

In this part, we have analyzed which coin (Bitcoin, Ethereum or Litecoin) was the most profitable in last two months if we would invest using buy and hold strategy. We have gone through the analysis of these 3 cryptocurrencies and try to give an objective answer.

Getting the data

Firstly, we have downloaded hourly data for BTC, ETH and LTC from Coinbase exchange. This time we work with hourly time interval as it has higher granularity. Cryptocompare API limits response to 2000 samples, which is 2.7 months of data for each coin.

Extract closing prices

We have analyzed closing prices, which are prices at which the hourly period closed. We merge BTC, ETH and LTC closing prices to a Dataframe to make analysis easier.

Analysis

Basic statistics

In 2.7 months, all three cryptocurrencies fluctuated a lot as you can observe in the table below.

For each coin, we count the number of events and calculate mean, standard deviation, minimum, quartiles and maximum closing price. Quartiles divide the data into four equal groups, each group comprising a quarter of the data.

Few interesting facts

- The difference between the highest and the lowest BTC price was more than $15000 in 2.7 months.
- The LTC surged from $48.61 to $378.66 at a certain point, which is an increase of 678.98%.

We visualize the data in the table above with a box plot. A box plot shows the quartiles of the dataset with points that are determined to be outliers using a method of the inter-quartile range (IQR). `IQR = Q3 - Q1`. In other words, the IQR is the first quartile (25%) subtracted from the third quartile (75%).

On the box plot below, we see that LTC closing hourly price was most of the time between $50 and $100 in the last 2.7 months. All values over $150 are outliers (using IQR) in our sample.

Histogram of LTC closing price

Let's estimate the frequency distribution of LTC closing prices.

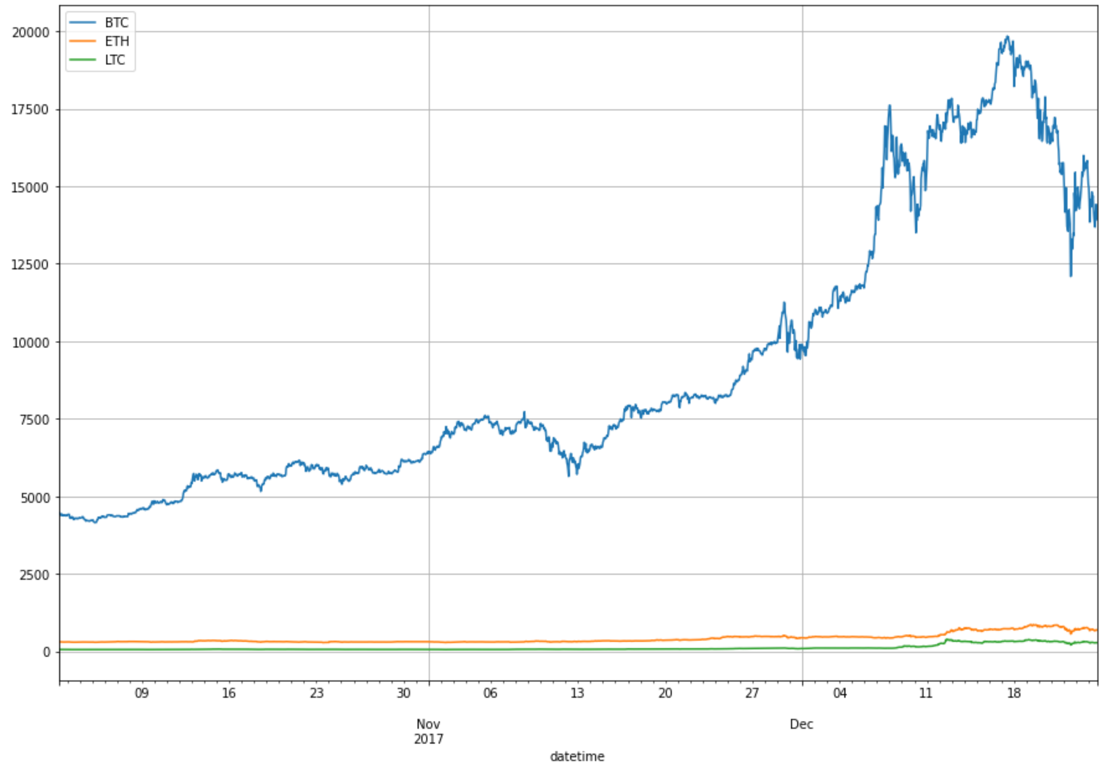
Observations

- it shows the number of hours LTC had a certain value. For example, we can observe that LTC closing price was not over $100 for many hours.

- it has right-skewed distribution because a natural limit prevents outcomes on one side.
- blue dotted line (median) shows that half of the closing prices were under 63.50$.

Visualize absolute closing prices

The chart below shows absolute closing prices. It is not of much use as BTC closing prices are much higher then prices of ETH and LTC.



Visualize relative changes of closing prices

We are interested in a relative change of the price rather than in absolute price, so we use three different scales.

We see that closing prices move in tandem. When one coin closing price increases so do the other.

Measure correlation of closing prices

We calculate [Pearson correlation](https://en.wikipedia.org/wiki/Pearson_correlation_coefficient) between closing prices of BTC, ETH and LTC. Pearson correlation is a measure of the linear correlation between two variables X and Y. It has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation.

Sifr Data daily updates Pearson correlations for many cryptocurrencies.

Observations

- Corelation matrix is symmetric so we only show the lower half.
- BTC, ETH and LTC were highly correlated in past 2 months. This means, when BTC closing price increased, ETH and LTC followed.

ETH and LTC were even more correlated with 0.9565 Pearson correlation coefficient.

Buy and hold strategy

Buy and hold is a passive investment strategy in which an investor buys a cryptocurrency and holds it for a long period of time, regardless of fluctuations in the market.

Let's analyze returns using buy and hold strategy for past 2.7 months. We calculate the return percentage, where t represents a certain time period and $price_0$ is initial closing price:

$$return_{t,0} = \frac{price_t}{price_0}$$

Visualize returns

We show that LTC was the most profitable for time period between October 2, 2017 and December 24, 2017.

Results

The cryptocurrencies we analyzed fluctuated a lot but all of gained in given 2.7 months period.

- What is the percentage increase?
- How many coins could we bought for $1000?

How much money would we make?

Cryptocurrency Analysis with Python - Log Returns

we analyzed raw price changes of cryptocurrencies. The problem with that approach is that prices of different cryptocurrencies are not normalized and we cannot use comparable metrics.

Load the data

Why Log Returns?

Benefit of using returns, versus prices, is normalization: measuring all variables in a comparable metric, thus enabling evaluation of analytic relationships amongst two or more variables despite originating from price series of unequal values (for details, see Why Log Returns).

Let's define return as:

$$r_i = \frac{p_i - p_j}{p_j},$$

where r_i is return at time i, p_i is the price at time i and $j = i - 1$.

Calculate Log Returns

Author of Why Log Returns outlines several benefits of using log returns instead of returns so we transform **returns** equation to **log returns** equation:

$$r_i = \frac{p_i - p_j}{p_j}$$
$$r_i = \frac{p_i}{p_j} - \frac{p_j}{p_j}$$
$$1 + r_i = \frac{p_i}{p_j}$$
$$\log(1 + r_i) = \log\left(\frac{p_i}{p_j}\right)$$
$$\log(1 + r_i) = \log(p_i) - \log(p_j)$$

Now, we apply the log returns equation to closing prices of cryptocurrencies:

Visualize Log Returns

We plot normalized changes of closing prices for last 50 hours. Log differences can be interpreted as the percentage change.

Are LTC prices distributed log-normally?

If we assume that prices are distributed log normally, then $\log(1 + r_i)$ is conveniently normally distributed (for details, see Why Log Returns)

On the chart below, we plot the distribution of LTC hourly closing prices. We also estimate parameters for log-normal distribution and plot estimated log-normal distribution with a red line.

Are LTC log returns normally distributed?

On the chart below, we plot the distribution of LTC log returns. We also estimate parameters for normal distribution and plot estimated normal distribution with a red line.

Pearson Correlation with log returns

We calculated Pearson Correlation from log returns. The correlation matrix below has similar values as the one at Sifr Data. There are differences because:

- we don't calculate volume-weighted average daily prices
- different time period (hourly and daily),
- different data source (Coinbase and Poloniex).

Observations

- BTC and ETH have moderate positive relationship,
- LTC and ETH have strong positive relationship.

Conclusion

We showed how to calculate log returns from raw prices with a practical example. This way we normalized prices, which simplifies further analysis. We also showed how to estimate parameters for normal and log-normal distributions.

Code

```
from_symbol = 'BTC'
to_symbol = 'USD'
exchange = 'Bitstamp'
datetime_interval = 'day'

import requests
from datetime import datetime
import pandas as pd
def get_filename(from_symbol, to_symbol, exchange, datetime_interval, d
ownload_date):

    return '%s_%s_%s_%s_%s.csv' % (from_symbol, to_symbol, exchange, da
tetime_interval, download_date)


def download_data(from_symbol, to_symbol, exchange, datetime_interval):
    supported_intervals = {'minute', 'hour', 'day'}
    assert datetime_interval in supported_intervals,           'datetim
e_interval should be one of %s' % supported_intervals
```

```
    print('Downloading %s trading data for %s %s from %s' %
          (datetime_interval, from_symbol, to_symbol, exchange))
    base_url = 'https://min-api.cryptocompare.com/data/histo'
    url = '%s%s' % (base_url, datetime_interval)

    params = {'fsym': from_symbol, 'tsym': to_symbol,
              'limit': 2000, 'aggregate': 1,
              'e': exchange}
    request = requests.get(url, params=params)
    data = request.json()
    return data


def convert_to_dataframe(data):
    df = pd.io.json.json_normalize(data, ['Data'])
    df['datetime'] = pd.to_datetime(df.time, unit='s')
    df = df[['datetime', 'low', 'high', 'open',
             'close', 'volumefrom', 'volumeto']]
    return df


def filter_empty_datapoints(df):
    indices = df[df.sum(axis=1) == 0].index
    print('Filtering %d empty datapoints' % indices.shape[0])
    df = df.drop(indices)
    return df


data = download_data(from_symbol, to_symbol, exchange, datetime_interva
l)
df = convert_to_dataframe(data)
df = filter_empty_datapoints(df)

current_datetime = datetime.now().date().isoformat()
filename = get_filename(from_symbol, to_symbol, exchange, datetime_inte
rval, current_datetime)
print('Saving data to %s' % filename)
df.to_csv(filename, index=False)



def read_dataset(filename):
    print('Reading data from %s' % filename)
    df = pd.read_csv(filename)
    df.datetime = pd.to_datetime(df.datetime) # change type from object
 to datetime
    df = df.set_index('datetime')
    df = df.sort_index() # sort by datetime
    print(df.shape)
    return df
```

```
df = read_dataset(filename)


from stockstats import StockDataFrame
df = StockDataFrame.retype(df)
df['macd'] = df.get('macd') # calculate MACD


df.head()


from math import pi

from bokeh.plotting import figure, show, output_notebook, output_file
output_notebook()

datetime_from = '2016-01-01 00:00'
datetime_to = '2017-12-10 00:00'


def get_candlestick_width(datetime_interval):
    if datetime_interval == 'minute':
        return 30 * 60 * 1000  # half minute in ms
    elif datetime_interval == 'hour':
        return 0.5 * 60 * 60 * 1000  # half hour in ms
    elif datetime_interval == 'day':
        return 12 * 60 * 60 * 1000  # half day in ms


df_limit = df[datetime_from: datetime_to].copy()
inc = df_limit.close > df_limit.open
dec = df_limit.open > df_limit.close

title = '%s datapoints from %s to %s for %s and %s from %s with MACD st
rategy' % (
    datetime_interval, datetime_from, datetime_to, from_symbol, to_symb
ol, exchange)
p = figure(x_axis_type="datetime",  plot_width=1000, title=title)

p.line(df_limit.index, df_limit.close, color='black')

# plot macd strategy
p.line(df_limit.index, 0, color='black')
p.line(df_limit.index, df_limit.macd, color='blue')
p.line(df_limit.index, df_limit.macds, color='orange')
p.vbar(x=df_limit.index, bottom=[
       0 for _ in df_limit.index], top=df_limit.macdh, width=4, color="
purple")
```

```
# plot candlesticks
candlestick_width = get_candlestick_width(datetime_interval)
p.segment(df_limit.index, df_limit.high,
          df_limit.index, df_limit.low, color="black")
p.vbar(df_limit.index[inc], candlestick_width, df_limit.open[inc],
       df_limit.close[inc], fill_color="#D5E1DD", line_color="black")
p.vbar(df_limit.index[dec], candlestick_width, df_limit.open[dec],
       df_limit.close[dec], fill_color="#F2583E", line_color="black")

output_file("visualizing_trading_strategy.html", title="visualizing tra
ding strategy")

show(p)
```

```
  import pandas as pd

def get_filename(from_symbol, to_symbol, exchange, datetime_interval, d
ownload_date):
    return '%s_%s_%s_%s_%s.csv' % (from_symbol, to_symbol, exchange, da
tetime_interval, download_date)

def read_dataset(filename):
    print('Reading data from %s' % filename)
    df = pd.read_csv(filename)
    df.datetime = pd.to_datetime(df.datetime) # change type from object
 to datetime
    df = df.set_index('datetime')
    df = df.sort_index() # sort by datetime
    print(df.shape)
    return df
df_btc = read_dataset(get_filename('BTC', 'USD', 'Coinbase', 'hour', '2
017-12-24'))
df_eth = read_dataset(get_filename('ETH', 'USD', 'Coinbase', 'hour', '2
017-12-24'))
df_ltc = read_dataset(get_filename('LTC', 'USD', 'Coinbase', 'hour', '2
017-12-24')

df_btc.head()
df = pd.DataFrame({'BTC': df_btc.close,
                   'ETH': df_eth.close,
                   'LTC': df_ltc.close})
```

```
df.head()
df.describe()

import seaborn as sns

ax = sns.boxplot(data=df['LTC'], orient="h")

df['LTC'].hist(bins=30, figsize=(15,10)).axvline(df['LTC'].median(), co
lor='b', linestyle='dashed', linewidth=2)

df.plot(grid=True, figsize=(15, 10))


import matplotlib.pyplot as plt
import numpy as np

fig, ax1 = plt.subplots(figsize=(20, 10))
ax2 = ax1.twinx()
rspine = ax2.spines['right']
rspine.set_position(('axes', 1.15))
ax2.set_frame_on(True)
ax2.patch.set_visible(False)
fig.subplots_adjust(right=0.7)

df['BTC'].plot(ax=ax1, style='b-')
df['ETH'].plot(ax=ax1, style='r-', secondary_y=True)
df['LTC'].plot(ax=ax2, style='g-')

# legend
ax2.legend([ax1.get_lines()[0],
            ax1.right_ax.get_lines()[0],
            ax2.get_lines()[0]],
           ['BTC', 'ETH', 'LTC'])
import seaborn as sns
import matplotlib.pyplot as plt

# Compute the correlation matrix
corr = df.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10, 10))

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, annot=True, fmt = '.4f', mask=mask, center=0, square=
True, linewidths=.5)
```

```python
df_return = df.apply(lambda x: x / x[0])
df_return.head()
df_return.plot(grid=True, figsize=(15, 10)).axhline(y = 1, color = "bla
ck", lw = 2)
df_perc = df_return.tail(1) * 100
ax = sns.barplot(data=df_perc)
df_perc
budget = 1000 # USD
df_coins = budget/df.head(1)

ax = sns.barplot(data=df_coins)
df_coins
df_profit = df_return.tail(1) * budget

ax = sns.barplot(data=df_profit)
df_profit
import pandas as pd

df_btc = pd.read_csv('BTC_USD_Coinbase_hour_2017-12-
24.csv', index_col='datetime')
df_eth = pd.read_csv('ETH_USD_Coinbase_hour_2017-12-
24.csv', index_col='datetime')
df_ltc = pd.read_csv('LTC_USD_Coinbase_hour_2017-12-
24.csv', index_col='datetime')


df = pd.DataFrame({'BTC': df_btc.close,
                   'ETH': df_eth.close,
                   'LTC': df_ltc.close})
df.index = df.index.map(pd.to_datetime)
df = df.sort_index()
df.head()
import numpy as np

# shift moves dates back by 1
df_change = df.apply(lambda x: np.log(x) - np.log(x.shift(1)))
df_change.head()
df_change[:50].plot(figsize=(15, 10)).axhline(color='black', linewidth=
2)

from scipy.stats import lognorm
import matplotlib.pyplot as plt
from scipy import stats

fig, ax = plt.subplots(figsize=(10, 6))

values = df['LTC']

shape, loc, scale = stats.lognorm.fit(values)
```

```
x = np.linspace(values.min(), values.max(), len(values))
pdf = stats.lognorm.pdf(x, shape, loc=loc, scale=scale)
label = 'mean=%.4f, std=%.4f, shape=%.4f' % (loc, scale, shape)

ax.hist(values, bins=30, density=True)
ax.plot(x, pdf, 'r-', lw=2, label=label)
ax.legend(loc='best')
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

values = df_change['LTC'][1:]  # skip first NA value
x = np.linspace(values.min(), values.max(), len(values))

loc, scale = stats.norm.fit(values)
param_density = stats.norm.pdf(x, loc=loc, scale=scale)
label = 'mean=%.4f, std=%.4f' % (loc, scale)

fig, ax = plt.subplots(figsize=(10, 6))
ax.hist(values, bins=30, density=True)
ax.plot(x, param_density, 'r-', label=label)
ax.legend(loc='best')
import seaborn as sns
import matplotlib.pyplot as plt

# Compute the correlation matrix
corr = df_change.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10, 10))

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, annot=True, fmt = '.4f', mask=mask, center=0, square=
True, linewidths=.5)
```

output

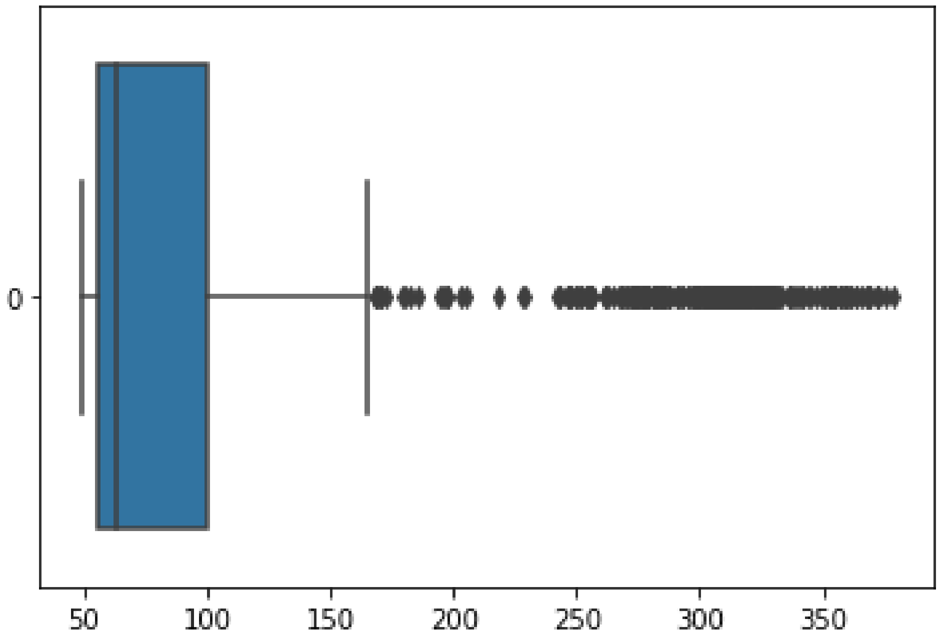
	low	high	open	close	volumefrom	volumeto	macd	macds	macdh
datetime									
2016-11-29	721.00	733.29	730.99	730.99	5193.25	3788691.62	0.000000	0.000000	0.000000
2016-11-30	727.00	744.49	730.72	742.06	6208.19	4577304.37	0.248365	0.137981	0.110385
2016-12-01	740.18	754.98	742.06	751.60	6097.42	4569793.09	0.612302	0.332375	0.279927
2016-12-02	750.77	778.07	751.55	769.99	8062.00	6181050.26	1.426785	0.703110	0.723675
2016-12-03	752.41	770.99	770.91	762.79	2763.36	2107293.54	1.552821	0.955880	0.596941

```
Reading data from BTC_USD_Coinbase_hour_2017-12-24.csv
(2001, 6)
Reading data from ETH_USD_Coinbase_hour_2017-12-24.csv
(2001, 6)
Reading data from LTC_USD_Coinbase_hour_2017-12-24.csv
(2001, 6)
```

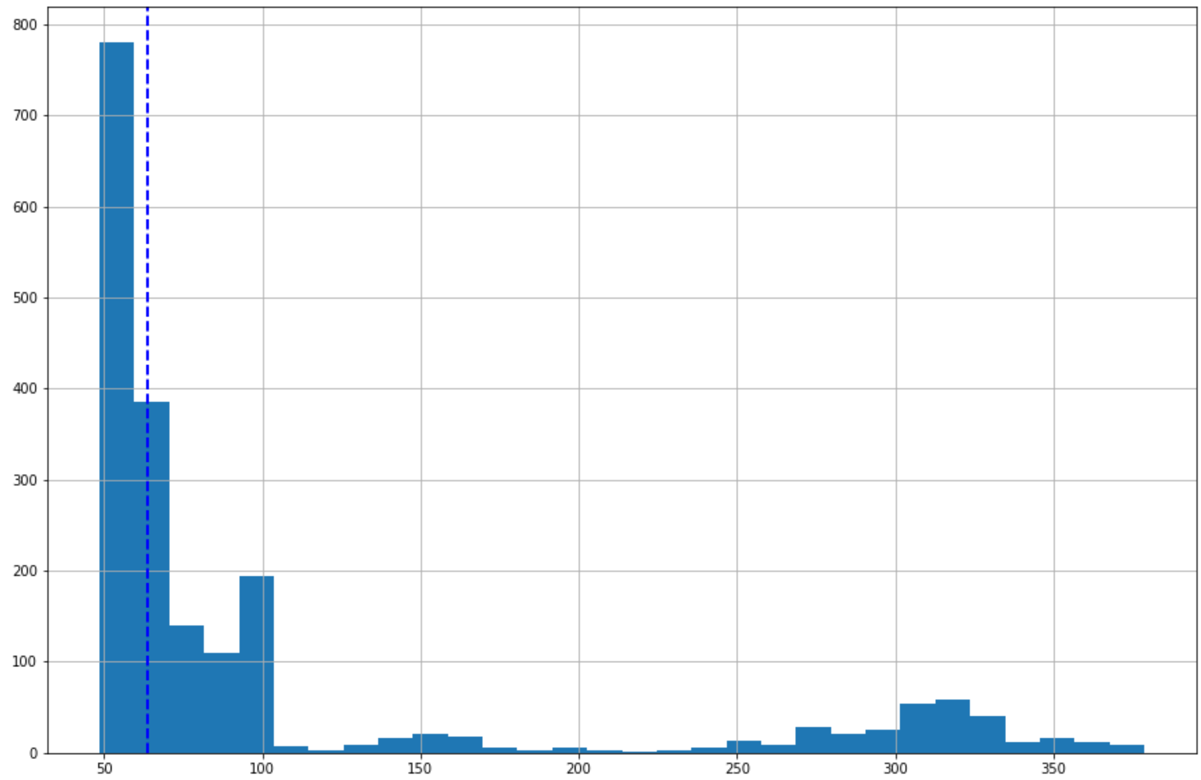
lowhighopenclosevolumefromvolumetodatetime2017-10-02 08:00:004435.004448.984435.014448.8585.51379813.672017-10-02 09:00:004448.844470.004448.854464.49165.17736269.532017-10-02 10:00:004450.274469.004464.494461.63194.95870013.622017-10-02 11:00:004399.004461.634461.634399.51326.711445572.022017-10-02 12:00:004378.224417.914399.514383.00549.292412712.73

	BTC	ETH	LTC
datetime			
2017-10-02 08:00:00	4448.85	301.37	54.72
2017-10-02 09:00:00	4464.49	301.84	54.79
2017-10-02 10:00:00	4461.63	301.95	54.63
2017-10-02 11:00:00	4399.51	300.02	54.01
2017-10-02 12:00:00	4383.00	297.51	53.71

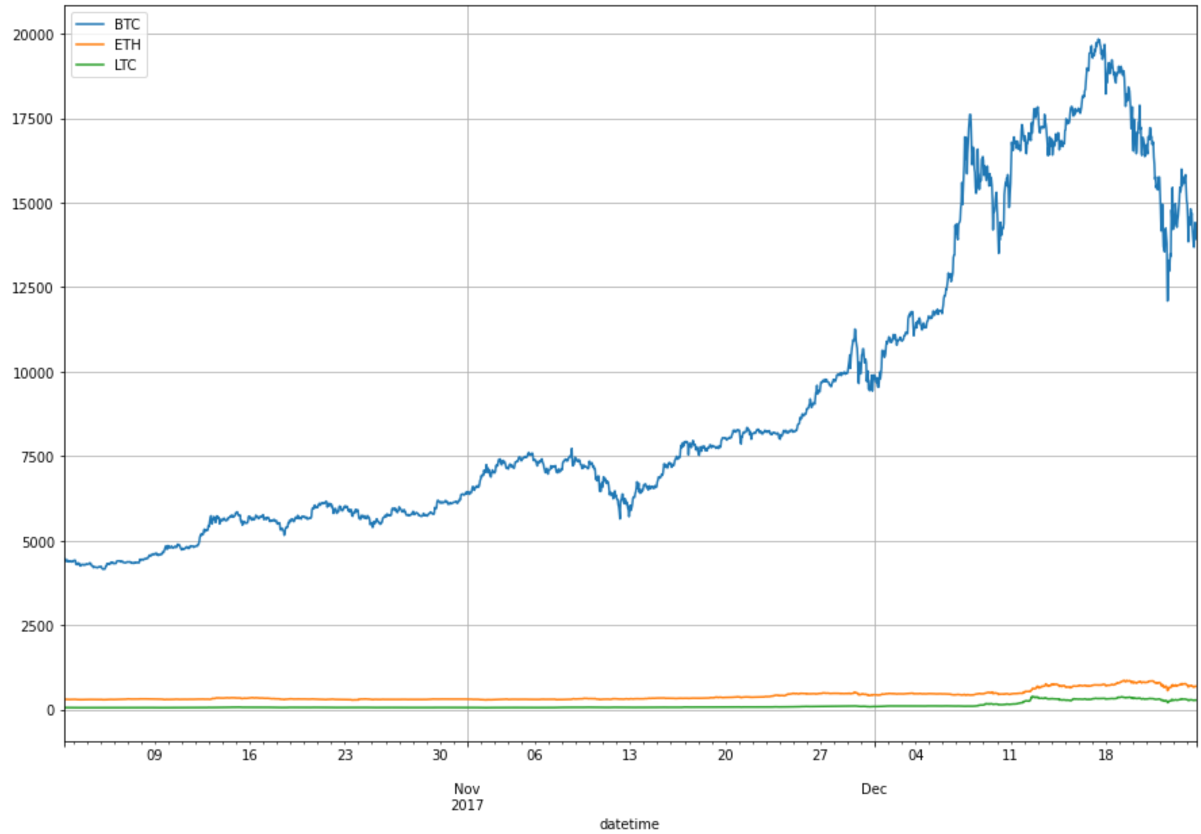
BTCETHLTCcount2001.0000002001.0000002001.000000mean9060.256122407.263793106.790100std4404.269591149.48041689.142241min4150.020000277.81000048.61000025%5751.020000301.51000055.58000050%7319.950000330.80000063.55000075%11305.000000464.390000100.050000max19847.110000858.900000378.660000



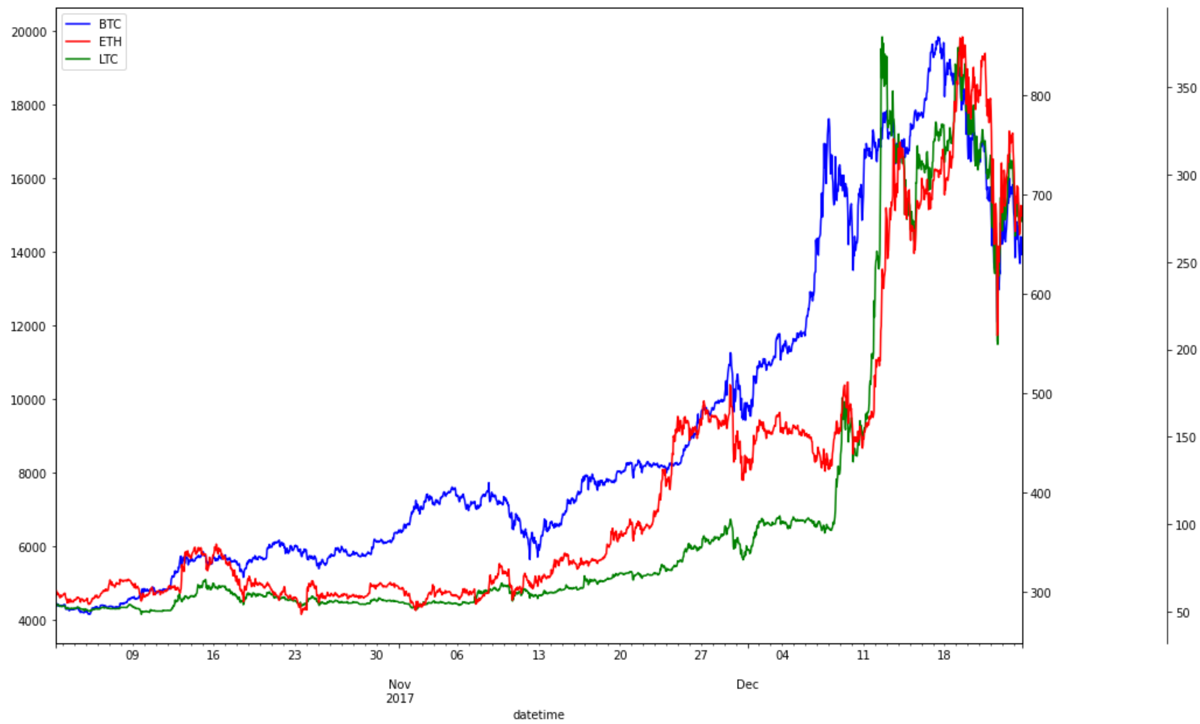
<matplotlib.lines.Line2D at 0x7f74c0aaed10>



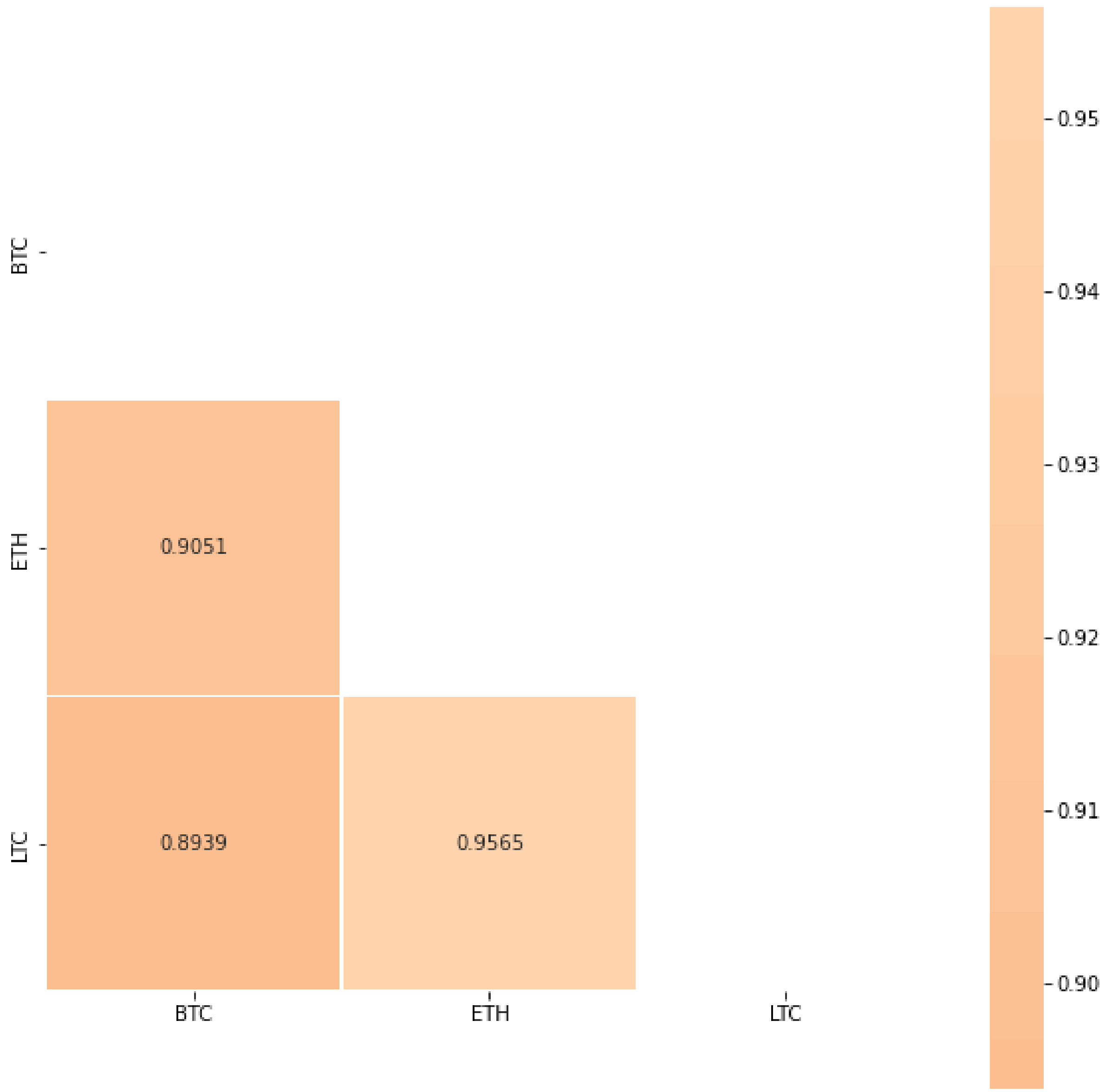
<matplotlib.axes._subplots.AxesSubplot at 0x7f74bead4190>



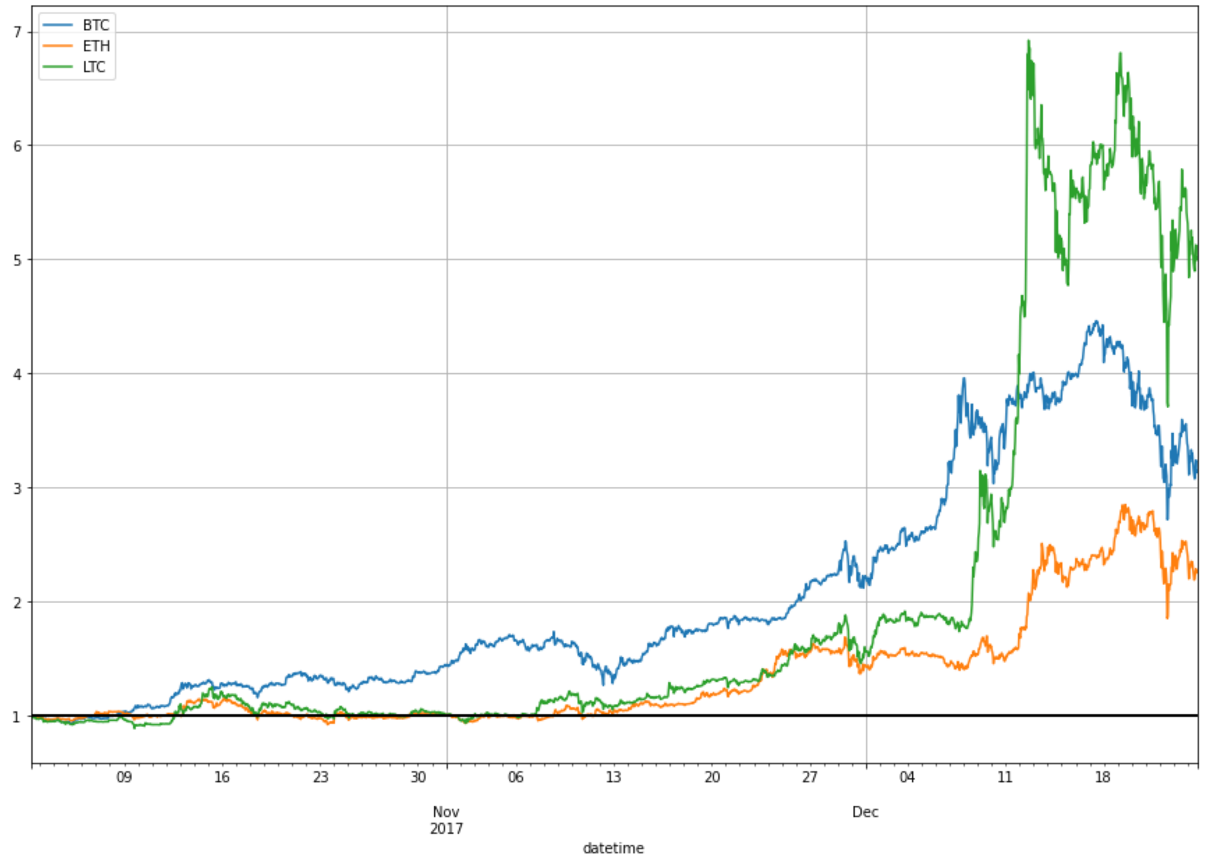
<matplotlib.legend.Legend at 0x7f74baf9c4d0>



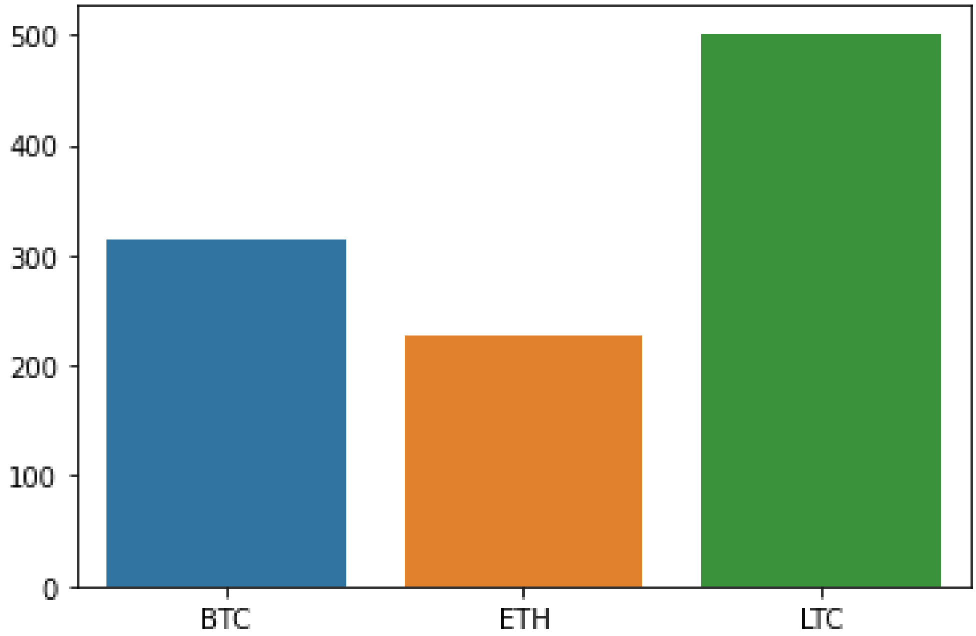
<matplotlib.axes._subplots.AxesSubplot at 0x7f74bae17b90>



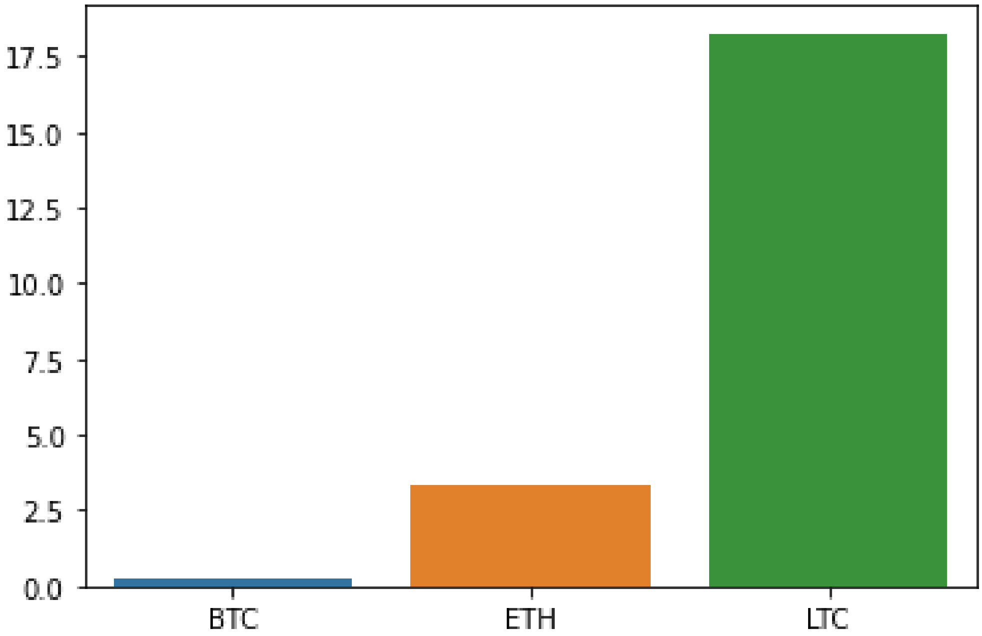

```
      BTC    ETH    LTC
datetime
2017-10-02 08:00:00    1.000000    1.000000    1.000000
2017-10-02 09:00:00    1.003516    1.001560    1.001279
2017-10-02 10:00:00    1.002873    1.001925    0.998355
2017-10-02 11:00:00    0.988909    0.995520    0.987025
2017-10-02 12:00:00    0.985198    0.987192    0.981542

<matplotlib.lines.Line2D at 0x7f74bb5f7090>
```


BTC
ETH
LTC
7
6
5
4
3
2
1
09
16
23
30
06
13
20
27
04
11
18
Nov
2017
Dec
datetime

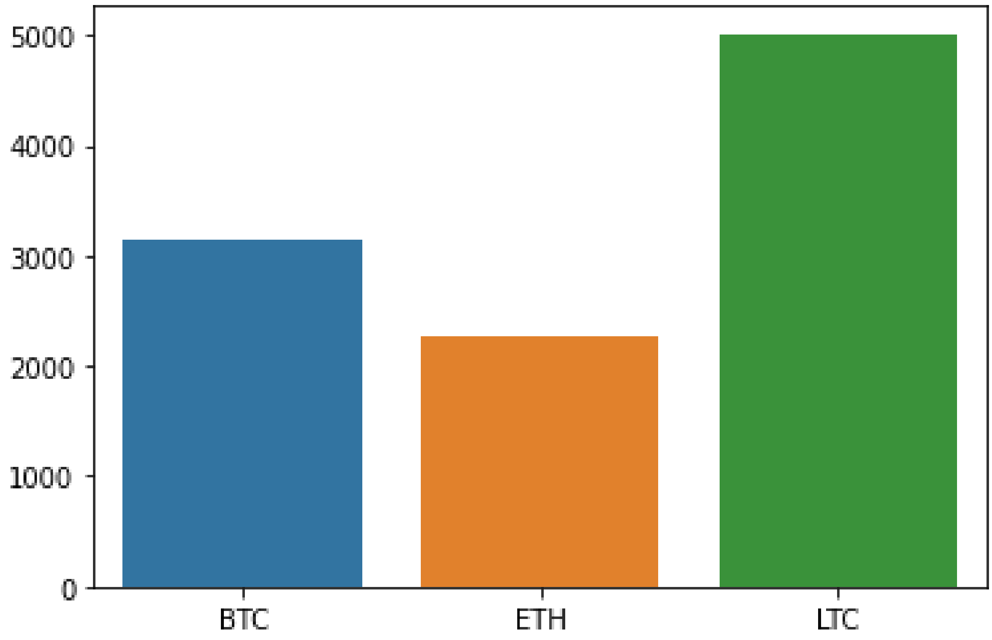
	BTC	ETH	LTC
datetime			
2017-12-24 16:00:00	314.688065	226.900488	501.407164



	BTC	ETH	LTC
datetime			
2017-10-02 08:00:00	0.224777	3.31818	18.274854



	BTC	ETH	LTC
datetime			
2017-12-24 16:00:00	3146.880655	2269.004878	5014.071637



	BTC	*ETH*	*LTC*
datetime			
2017-10-02 08:00:00	*4448.85*	*301.37*	*54.72*
2017-10-02 09:00:00	*4464.49*	*301.84*	*54.79*
2017-10-02 10:00:00	*4461.63*	*301.95*	*54.63*
2017-10-02 11:00:00	*4399.51*	*300.02*	*54.01*
2017-10-02 12:00:00	*4383.00*	*297.51*	*53.71*

	BTC	**ETH**	**LTC**
count	**2001.000000**	**2001.000000**	**2001.000000**
mean	**9060.256122**	**407.263793**	**106.790100**
std	**4404.269591**	**149.480416**	**89.142241**
min	**4150.020000**	**277.810000**	**48.610000**
25%	**5751.020000**	**301.510000**	**55.580000**
50%	**7319.950000**	**330.800000**	**63.550000**
75%	**11305.000000**	**464.390000**	**100.050000**
max	**19847.110000**	**858.900000**	**378.660000**

	BTC	**ETH**	**LTC**
datetime			
2017-10-02 08:00:00	**NaN**	**NaN**	**NaN**
2017-10-02 09:00:00	**0.003509**	**0.001558**	**0.001278**
2017-10-02 10:00:00	**-0.000641**	**0.000364**	**-0.002925**

2017-10-02 11:00:00	-0.014021	-0.006412	-0.011414
2017-10-02 12:00:00	-0.003760	-0.008401	-0.005570

```
<matplotlib.lines.Line2D at 0x7f74ba8309d0>
```

```
Reading data from BTC_USD_Bitstamp_day_2022-05-22.csv
(2001, 6)
```

```
df.head()
```

datetime	low	high	open	close	volumefrom	volumeto	macd	macds	macd
2016-11-29	721.00	733.29	730.99	730.99	5193.25	3788691.62	0.000000	0.000000	0.00000
2016-11-30	727.00	744.49	730.72	742.06	6208.19	4577304.37	0.248365	0.137981	0.11038
2016-12-01	740.18	754.98	742.06	751.60	6097.42	4569793.09	0.612302	0.332375	0.27992
2016-12-02	750.77	778.07	751.55	769.99	8062.00	6181050.26	1.426785	0.703110	0.72367
2016-12-03	752.41	770.99	770.91	762.79	2763.36	2107293.54	1.552821	0.955880	0.59694

final_crypto_analysis.ipynb

```python
                   'close', 'volumefrom', 'volumeto']]
    return df


def filter_empty_datapoints(df):
    indices = df[df.sum(axis=1) == 0].index
    print('Filtering %d empty datapoints' % indices.shape[0])
    df = df.drop(indices)
    return df


data = download_data(from_symbol, to_symbol, exchange, datetime_interval)
df = convert_to_dataframe(data)
df = filter_empty_datapoints(df)

current_datetime = datetime.now().date().isoformat()
filename = get_filename(from_symbol, to_symbol, exchange, datetime_interval, current_datetime)
print('Saving data to %s' % filename)
df.to_csv(filename, index=False)
```

```
Downloading day trading data for BTC USD from Bitstamp
Filtering 0 empty datapoints
Saving data to BTC_USD_Bitstamp_day_2022-05-22.csv
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:29: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.j
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:37: FutureWarning: Dropping of nuisance columns in DataFrame reductions (wit
```

final_crypto_analysis.ipynb

Read the data

We read the data from a file so we don't need to download it again.

```python
def read_dataset(filename):
    print('Reading data from %s' % filename)
    df = pd.read_csv(filename)
    df.datetime = pd.to_datetime(df.datetime) # change type from object to datetime
    df = df.set_index('datetime')
    df = df.sort_index() # sort by datetime
    print(df.shape)
    return df

df = read_dataset(filename)
```

```
Reading data from BTC_USD_Bitstamp_day_2022-05-22.csv
(2001, 6)
```

Trading strategy

- **macd** is MACD line,
- **macds** is signal line,
- **macdh** is MACD histogram.

```
[6] df.head()
```

datetime	low	high	open	close	volumefrom	volumeto	macd	macds	macdh
2016-11-29	721.00	733.29	730.99	730.99	5193.25	3788691.62	0.000000	0.000000	0.000000
2016-11-30	727.00	744.49	730.72	742.06	6208.19	4577304.37	0.248365	0.137981	0.110385
2016-12-01	740.18	754.98	742.06	751.60	6097.42	4569793.09	0.612302	0.332375	0.279927
2016-12-02	750.77	778.07	751.55	769.99	8062.00	6181050.26	1.426785	0.703110	0.723675
2016-12-03	752.41	770.99	770.91	762.79	2763.36	2107293.54	1.552821	0.955880	0.596941

Visualizing trading strategy

We use bokeh interactive charts to plot the data.

The line graph shows daily closing prices with candlesticks (zoom in). A candlestick displays the high, low, opening and closing prices for a

```
df_btc = read_dataset(get_filename('BTC', 'USD', 'Coinbase', 'hour', '2017-12-24'))
df_eth = read_dataset(get_filename('ETH', 'USD', 'Coinbase', 'hour', '2017-12-24'))
df_ltc = read_dataset(get_filename('LTC', 'USD', 'Coinbase', 'hour', '2017-12-24'))
```

```
Reading data from BTC_USD_Coinbase_hour_2017-12-24.csv
(2001, 6)
Reading data from ETH_USD_Coinbase_hour_2017-12-24.csv
(2001, 6)
Reading data from LTC_USD_Coinbase_hour_2017-12-24.csv
(2001, 6)
```

```
[10] df_btc.head()
```

datetime	low	high	open	close	volumefrom	volumeto
2017-10-02 08:00:00	4435.00	4448.98	4435.01	4448.85	85.51	379813.67
2017-10-02 09:00:00	4448.84	4470.00	4448.85	4464.49	165.17	736269.53
2017-10-02 10:00:00	4450.27	4469.00	4464.49	4461.63	194.95	870013.62
2017-10-02 11:00:00	4399.00	4461.63	4461.63	4399.51	326.71	1445572.02
2017-10-02 12:00:00	4378.22	4417.91	4399.51	4383.00	549.29	2412712.73

We are going to analyze closing prices, which are prices at which the hourly period closed. We merge BTC, ETH and LTC closing prices to a Dataframe to make analysis easier.

```
[11] df = pd.DataFrame({'BTC': df_btc.close,
                        'ETH': df_eth.close,
                        'LTC': df_ltc.close})
```

```
[12] df.head()
```

datetime	BTC	ETH	LTC
2017-10-02 08:00:00	4448.85	301.37	54.72
2017-10-02 09:00:00	4464.49	301.84	54.79
2017-10-02 10:00:00	4461.63	301.95	54.63
2017-10-02 11:00:00	4399.51	300.02	54.01
2017-10-02 12:00:00	4383.00	297.51	53.71

Analysis

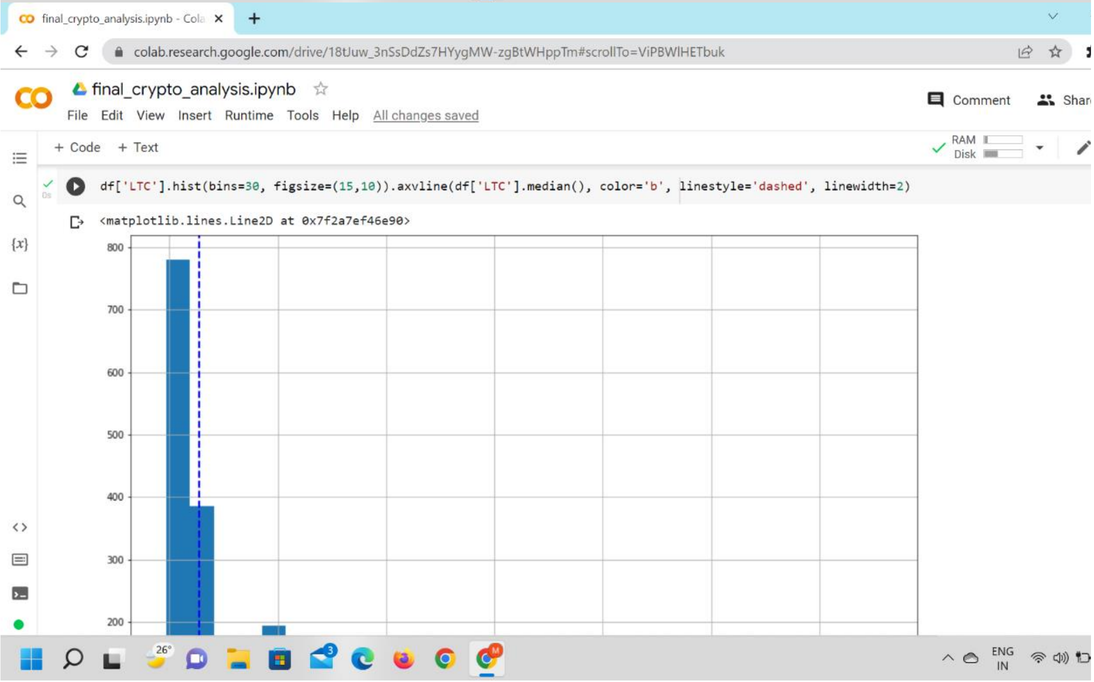
Few interesting facts

- The difference between the highest and the lowest BTC price was more than $15000 in 2.7 months.
- The LTC surged from $48.61 to $378.66 at a certain point, which is an increase of 678.98%.

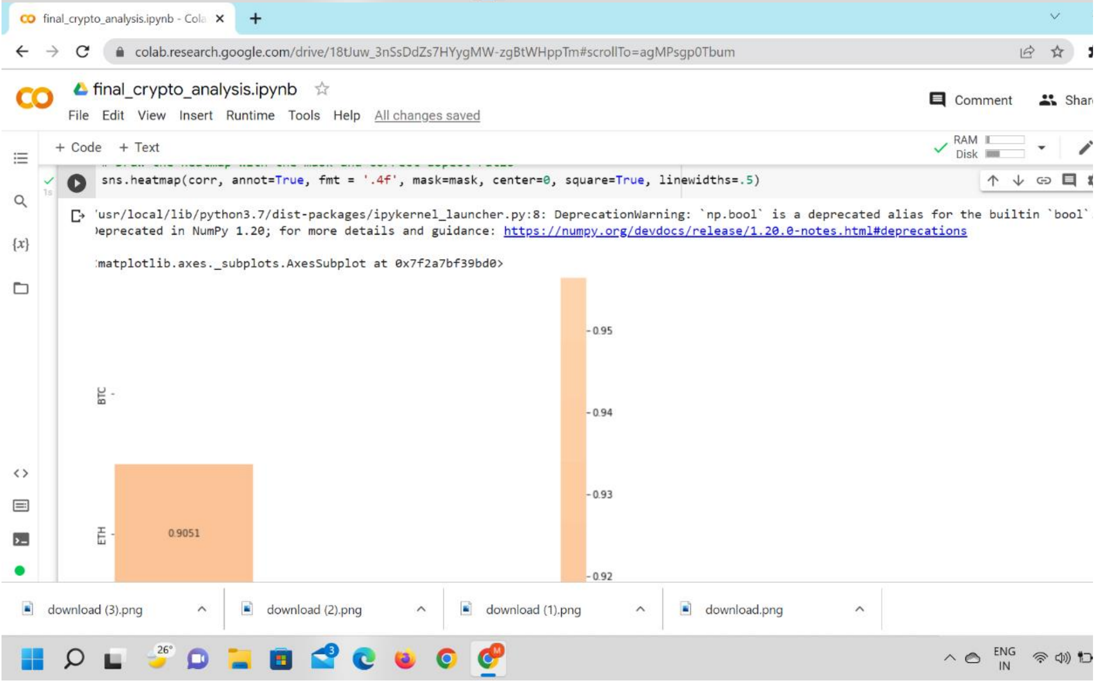
```
[13] df.describe()
```

	BTC	ETH	LTC
count	2001.000000	2001.000000	2001.000000
mean	9060.256122	407.263793	106.790100
std	4404.269591	149.480416	89.142241
min	4150.020000	277.810000	48.610000
25%	5751.020000	301.510000	55.580000
50%	7319.950000	330.800000	63.550000
75%	11305.000000	464.390000	100.050000
max	19847.110000	858.900000	378.660000

Lets dive deeper into LTC

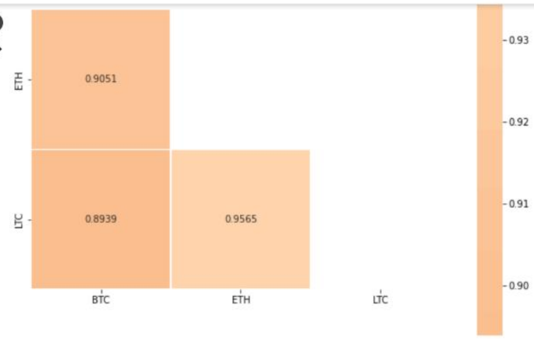

final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=gFJehc7dTbud
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
On the box plot below, we see that LTC closing hourly price was most of the time between $50 and $100 in the last 2.7 months. All values over $150 are outliers (using IQR) in our sample.
import seaborn as sns
ax = sns.boxplot(data=df['LTC'], orient="h")
0
50 100 150 200 250 300 350
Histogram of LTC closing price
ENG
IN

final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=ViPBWIHETbuk
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
df['LTC'].hist(bins=30, figsize=(15,10)).axvline(df['LTC'].median(), color='b', linestyle='dashed', linewidth=2)
<matplotlib.lines.Line2D at 0x7f2a7ef46e90>
800
700
600
500
400
300
200
ENG
IN


final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=IKzvqoLETbul
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
df.plot(grid=True, figsize=(15, 10))
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a7c14b310>
BTC
ETH
LTC
20000
17500
15000
12500
10000
7500
download.png
ENG
IN


final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=agMPsgp0Tbum
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
sns.heatmap(corr, annot=True, fmt = '.4f', mask=mask, center=0, square=True, linewidths=.5)
'usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`
eprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
matplotlib.axes._subplots.AxesSubplot at 0x7f2a7bf39bd0>
BTC
ETH
0.9051
0.95
0.94
0.93
0.92
download (3).png
download (2).png
download (1).png
download.png
ENG
IN

final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=agMPsgp0Tbum

final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text



Buy and hold strategy

download (3).png download (2).png download (1).png download.png

final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=agMPsgp0Tbum

final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
df_return = df.apply(lambda x: x / x[0])
df_return.head()
```

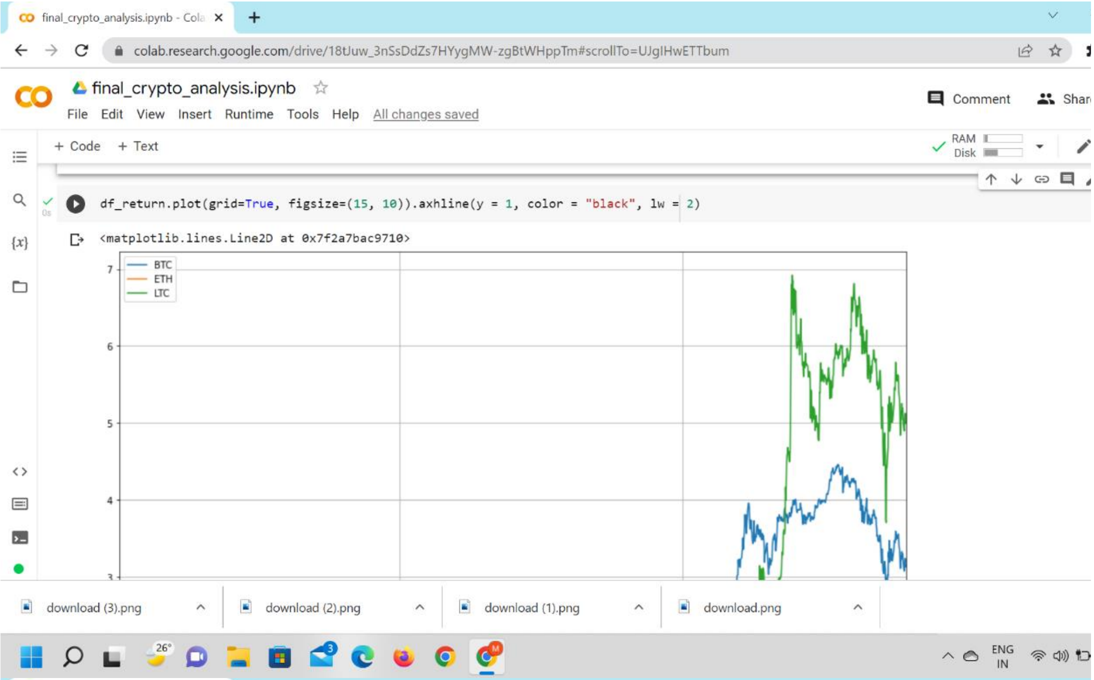
	BTC	ETH	LTC
datetime			
2017-10-02 08:00:00	1.000000	1.000000	1.000000
2017-10-02 09:00:00	1.003516	1.001560	1.001279
2017-10-02 10:00:00	1.002873	1.001925	0.998355
2017-10-02 11:00:00	0.988909	0.995520	0.987025
2017-10-02 12:00:00	0.985198	0.987192	0.981542

Visualize returns

We show that LTC was the most profitable for time period between October 2, 2017 and December 24, 2017.

[ ] 1 cell hidden

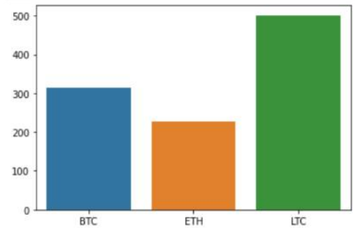
download (3).png download (2).png download (1).png download.png


final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tJuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=UJglHwETTbum
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
df_return.plot(grid=True, figsize=(15, 10)).axhline(y = 1, color = "black", lw = 2)
<matplotlib.lines.Line2D at 0x7f2a7bac9710>
BTC
ETH
LTC
download (3).png
download (2).png
download (1).png
download.png
ENG
IN


final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tJuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=UJglHwETTbum
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
Nov
2017
Dec
datetime
Conclusion
The cryptocurrencies we analyzed fluctuated a lot but all of gained in given 2.7 months period
download (3).png
download (2).png
download (1).png
download.png
ENG
IN

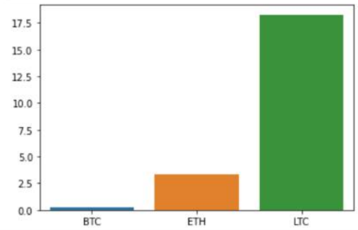
```
df_perc = df_return.tail(1) * 100
ax = sns.barplot(data=df_perc)
df_perc
```

	BTC	ETH	LTC
datetime			
2017-12-24 16:00:00	314.688065	226.900488	501.407164



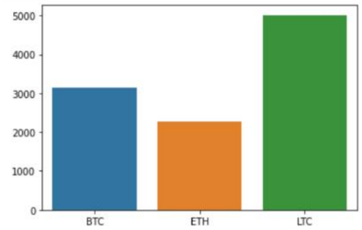

```
df_coins = budget/df.head(1)

ax = sns.barplot(data=df_coins)
df_coins
```

	BTC	ETH	LTC
datetime			
2017-10-02 08:00:00	0.224777	3.31818	18.274854



```
df_profit = df_return.tail(1) * budget

ax = sns.barplot(data=df_profit)
df_profit
```

datetime	BTC	ETH	LTC
2017-12-24 16:00:00	3146.880655	2269.004878	5014.071637




```
[25] df = df.sort_index()
```

```
df.head()
```

datetime	BTC	ETH	LTC
2017-10-02 08:00:00	4448.85	301.37	54.72
2017-10-02 09:00:00	4464.49	301.84	54.79
2017-10-02 10:00:00	4461.63	301.95	54.63
2017-10-02 11:00:00	4399.51	300.02	54.01
2017-10-02 12:00:00	4383.00	297.51	53.71

```
[27] df.describe()
```

	BTC	ETH	LTC
count	2001.000000	2001.000000	2001.000000

final_crypto_analysis.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
df.describe()
```

	BTC	ETH	LTC
count	2001.000000	2001.000000	2001.000000
mean	9060.256122	407.263793	106.790100
std	4404.269591	149.480416	89.142241
min	4150.020000	277.810000	48.610000
25%	5751.020000	301.510000	55.580000
50%	7319.950000	330.800000	63.550000
75%	11305.000000	464.390000	100.050000
max	19847.110000	858.900000	378.660000

Why Log Returns?

download (3).png download (2).png download (1).png download.png

final_crypto_analysis.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
[28] # shift moves dates back by 1
df_change = df.apply(lambda x: np.log(x) - np.log(x.shift(1)))
```

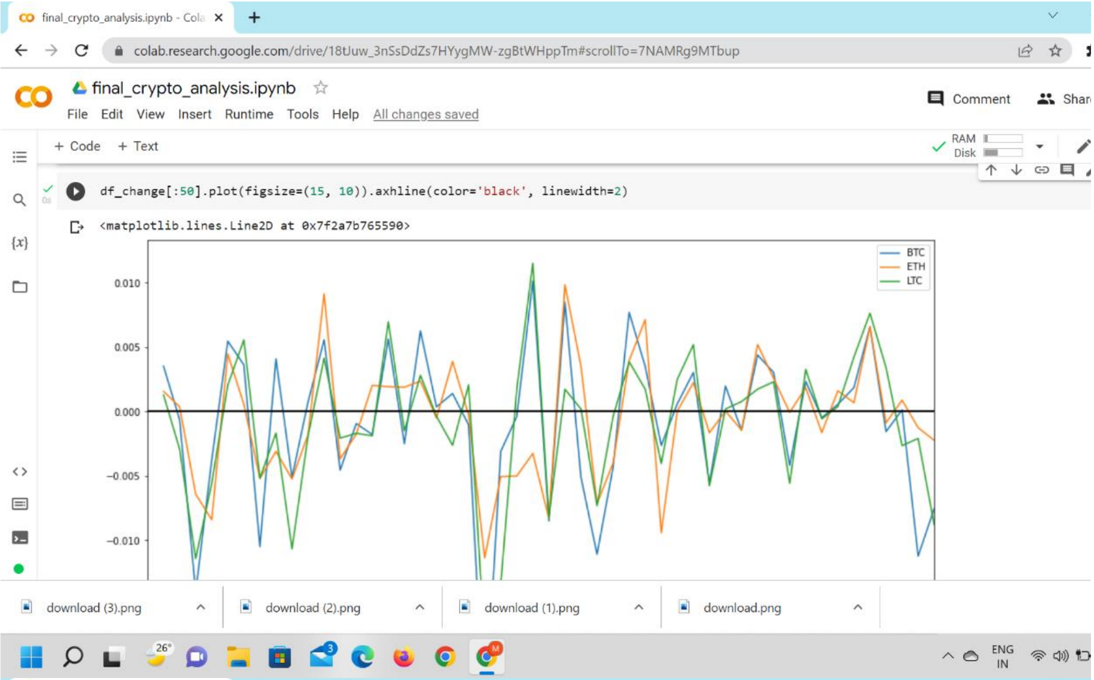
```
df_change.head()
```

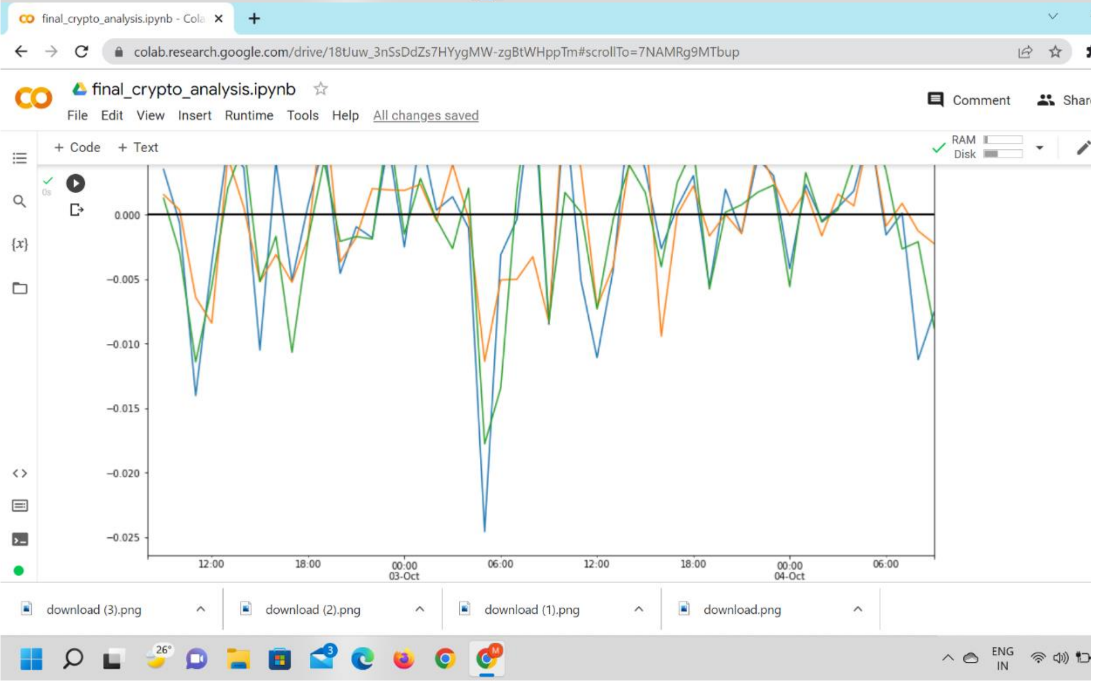
datetime	BTC	ETH	LTC
2017-10-02 08:00:00	NaN	NaN	NaN
2017-10-02 09:00:00	0.003509	0.001558	0.001278
2017-10-02 10:00:00	-0.000641	0.000364	-0.002925
2017-10-02 11:00:00	-0.014021	-0.006412	-0.011414
2017-10-02 12:00:00	-0.003760	-0.008401	-0.005570

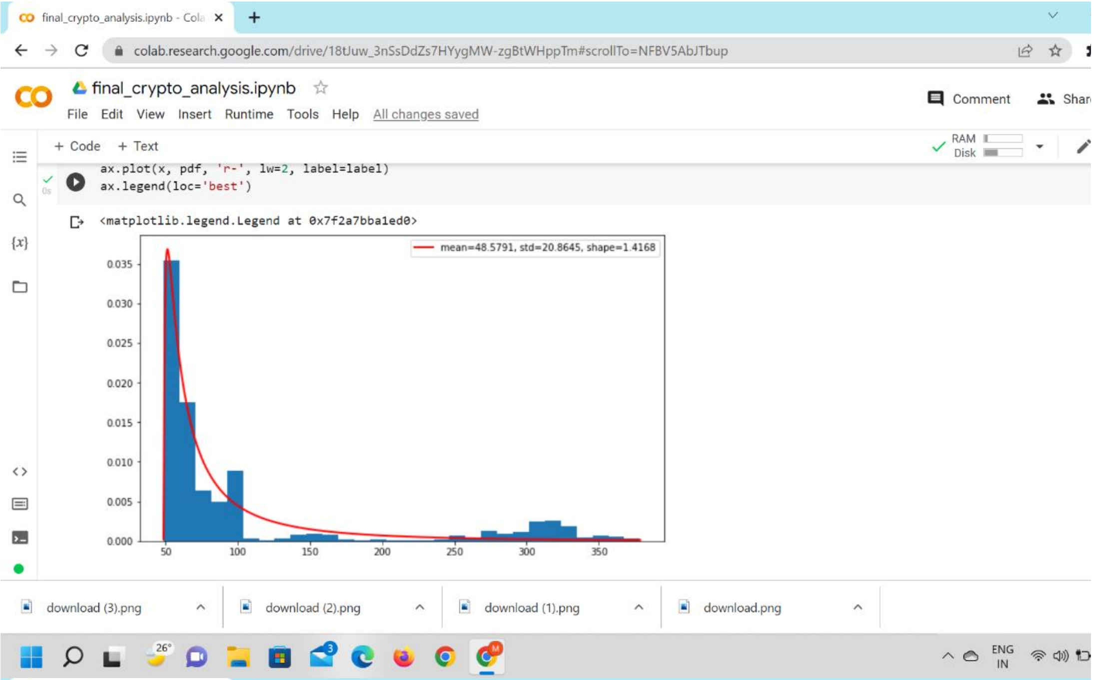
Visualize Log Returns

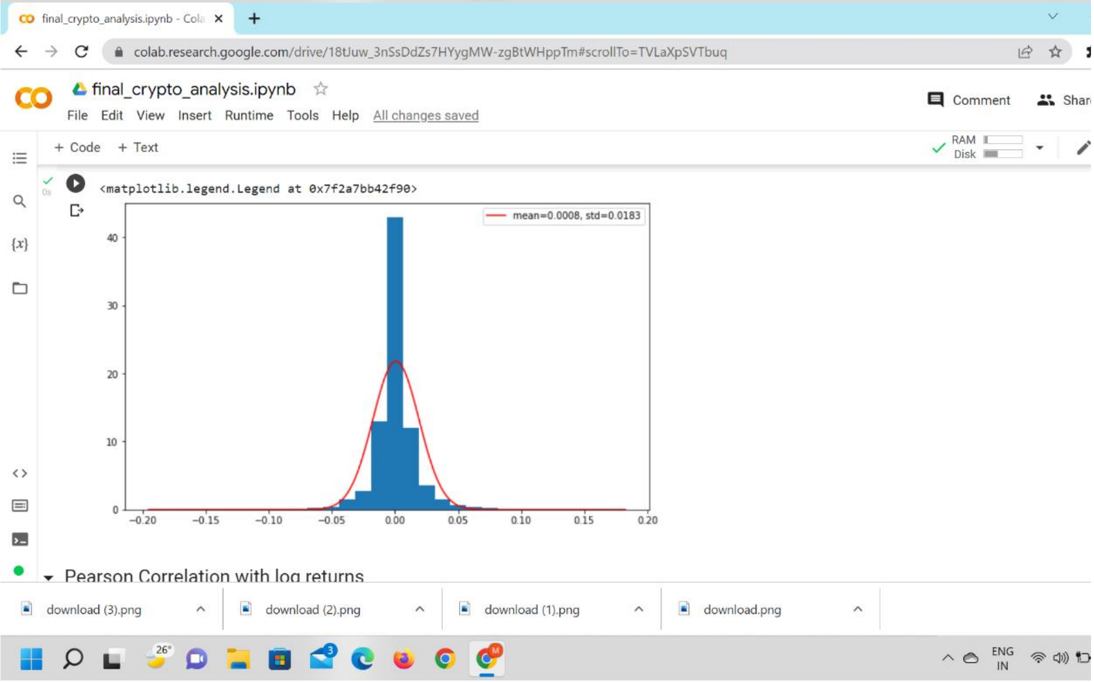
We plot normalized changes of closing prices for last 50 hours. Log differences can be interpreted as the percentage change.

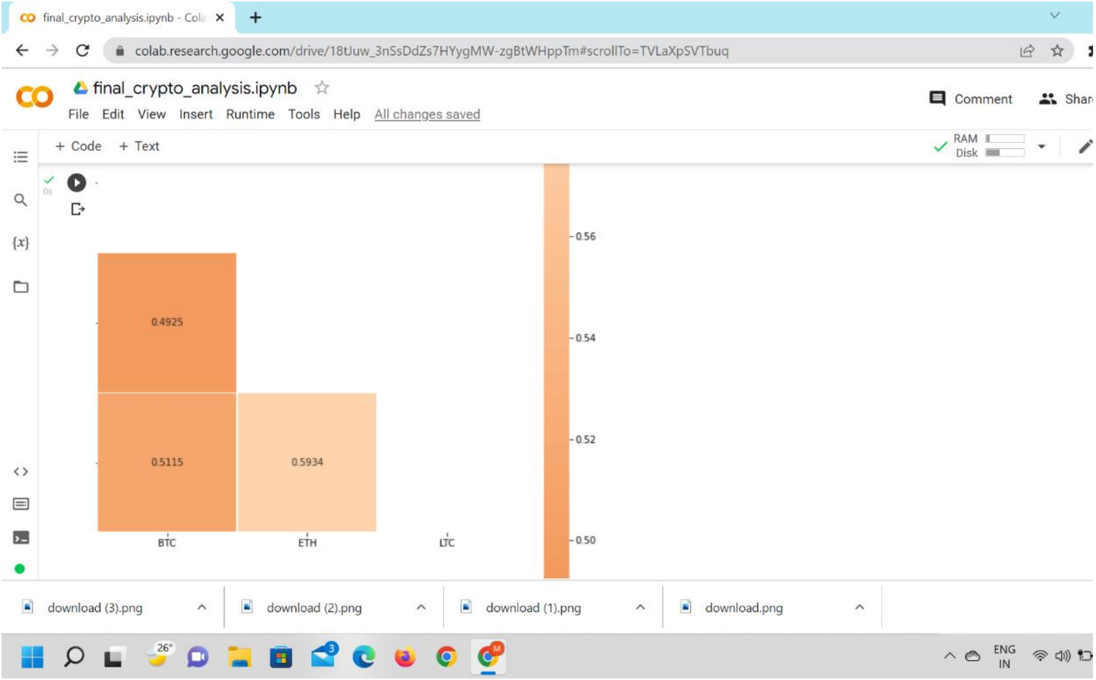
download (3).png download (2).png download (1).png download.png


final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=7NAMRg9MTbup
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Shar
+ Code + Text
RAM
Disk
df_change[:50].plot(figsize=(15, 10)).axhline(color='black', linewidth=2)
<matplotlib.lines.Line2D at 0x7f2a7b765590>
BTC
ETH
LTC
0.010
0.005
0.000
−0.005
−0.010
download (3).png
download (2).png
download (1).png
download.png
ENG
IN

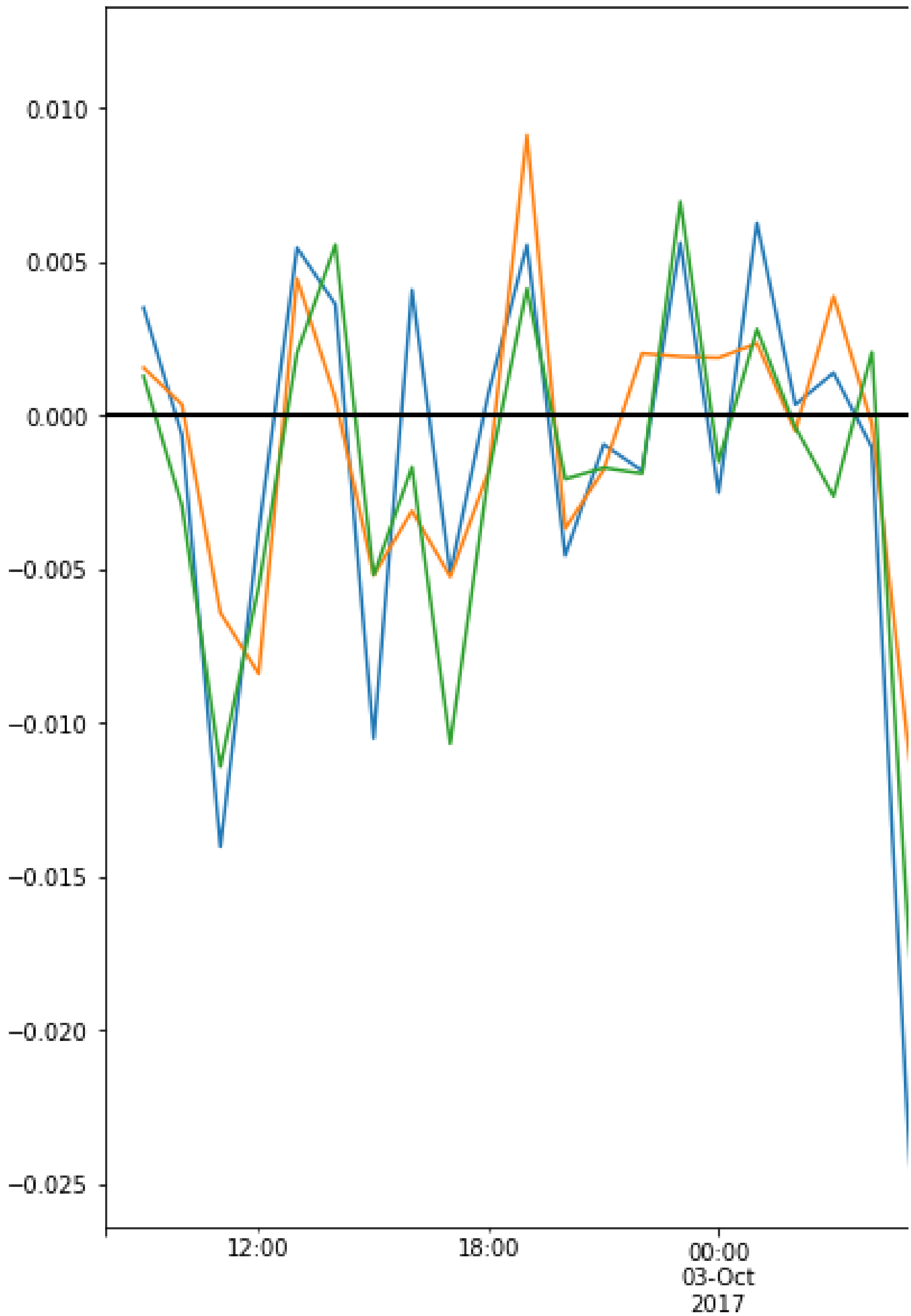

final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=7NAMRg9MTbup
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Shar
+ Code + Text
RAM
Disk
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
12:00
18:00
00:00
03-Oct
06:00
12:00
18:00
00:00
04-Oct
06:00
download (3).png
download (2).png
download (1).png
download.png
ENG
IN


final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=NFBV5AbJTbup
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
ax.plot(x, pdf, 'r-', lw=2, label=label)
ax.legend(loc='best')
<matplotlib.legend.Legend at 0x7f2a7bba1ed0>
mean=48.5791, std=20.8645, shape=1.4168
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
50
100
150
200
250
300
350
download (3).png
download (2).png
download (1).png
download.png
ENG
IN

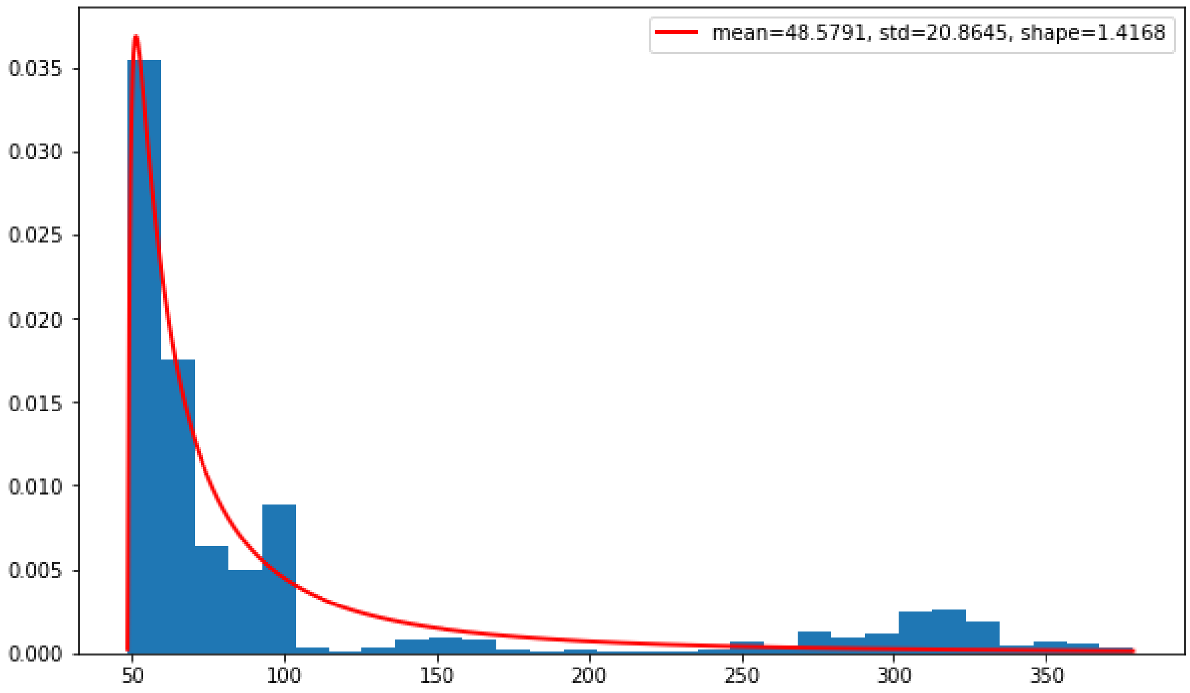

final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=TVLaXpSVTbuq
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
<matplotlib.legend.Legend at 0x7f2a7bb42f90>
mean=0.0008, std=0.0183
40
30
20
10
0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
Pearson Correlation with log returns
download (3).png
download (2).png
download (1).png
download.png
ENG
IN



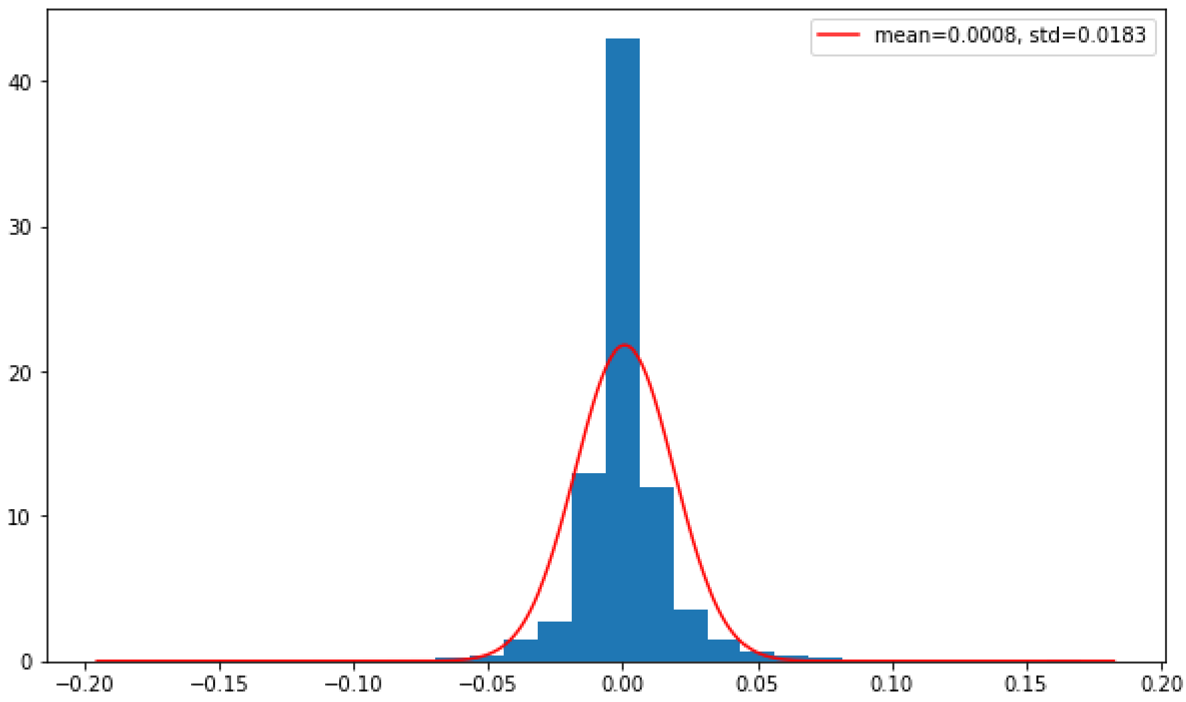
final_crypto_analysis.ipynb - Cola
colab.research.google.com/drive/18tUuw_3nSsDdZs7HYygMW-zgBtWHppTm#scrollTo=TVLaXpSVTbuq
final_crypto_analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
0.4925
0.5115
0.5934
BTC
ETH
LTC
0.56
0.54
0.52
0.50
download (3).png
download (2).png
download (1).png
download.png
ENG
IN



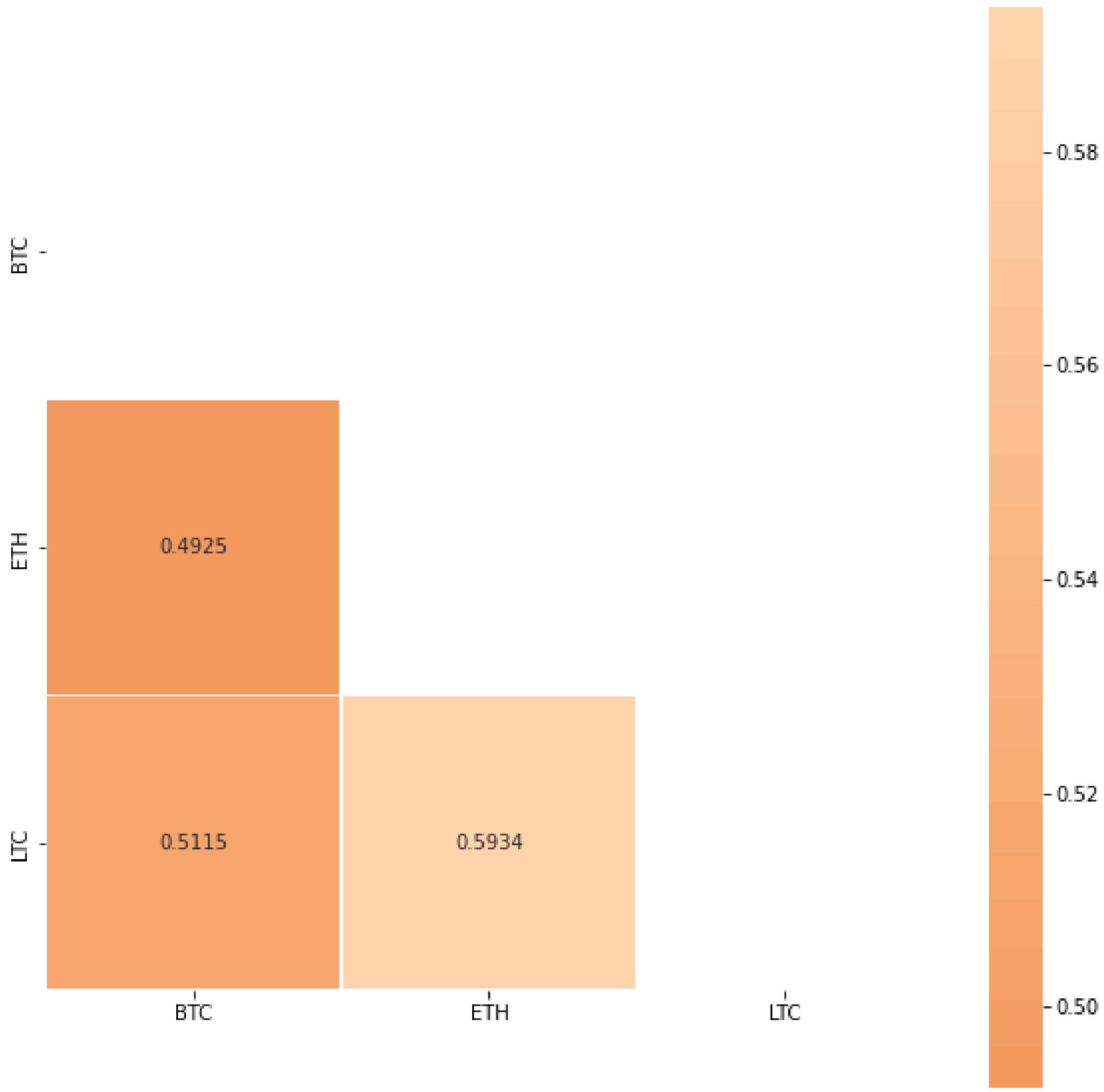

```
<matplotlib.legend.Legend at 0x7f74ba69fe90>
```




```
<matplotlib.legend.Legend at 0x7f74ba63b750>
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74ba4fe7d0>
```



BTC
ETH
LTC
0.4925
0.5115
0.5934
BTC
ETH
LTC
0.58
0.56
0.54
0.52
0.50