Final

January 6, 2023

1 Final Project: Crypto Currency from 2013 to 2021

Member of our team:

Fullname

Student ID

Đoàn Ánh Dương

20127474

1.1 Import Library

Necessary file imported here.

Note: Some external library out of `min_ds-env` need to be installed: - plotly

[1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go

from datetime import datetime, timedelta
from dateutil import relativedelta

import os
import glob
```

Environment using within this final project.

[2]:
```
import sys
sys.executable
```

[2]: '/home/duong/anaconda3/bin/python'

Set options here.

[3]:
```python
pd.set_option('display.max_colwidth', None)
# pd.set_option('display.float_format', '{:.2f}'.format)
```

1.2 I. Idea and Expectation

1.2.1 Idea

For recent years, we have been heard at least once about crypto, particularly Bitcoin and its application. Although digital currency brought many controversial opinions up to now, we cannot realize its success and popularity.

So I decided to deep in crypto market and fortunately, I found the historical price data of crypto currency on Kaggle. Although the data does not contain all information about a specific coin, the historical price can bring us to take an overview of digital currency evolution.

Throughout this project, I hope that we can have a perspective objectively and a basic knowledge about digital currency. ### Expectation In this project, we will find out the crypto currency market and its history, with expectation that we can find a method to buy a coin and take profit.

1.3 II. Datasets and Collection

1.3.1 Source and Public Data

I found the datasets used in this project was public on Kaggle: Crypto Currrency price history.

1.3.2 License

The data is taken from coinmarketcap and it is free to use the data.

More about the license I found on kaggle: - CC0: Public Domain (No copyright)

1.3.3 About Data

The dataset has one csv file for each currency. **Price history is available on a daily basis from *April 28, 2013***. This dataset has the historical price information of some of the top crypto currencies by market capitalization, including Bitcoin, Etherium, Binance Coin,…

1.3.4 Description

All information of columns contain in the file named `description.txt`.

1.3.5 Author

Datasets were collected by Sudalai Rajkumar(registered account name SRK) on Kaggle.

1.3.6 How did the author collect data?

As the author's answer, there is not mentioned clearly how the way he collected data, maybe collected by scraping historical data web. For instance: bitcoin.

1.3.7 Is dataset subjective or objective?

Besides the historical price of digital currency, there are many factors in a coin we need to take consideration on. For example: Holders Statistic over time, legality, team and organization investors, who created the coin...

So that the dataset is subjective.

1.4 III. Preprocessing Data

1.4.1 1. Change files.

First, we cannot access each files to analyze, it must be combined into one.

How many files does we have?

[4]:
```
path = 'archive/'
os.chdir(path)
ext = 'csv'
all_files = [f for f in glob.glob(f'*.{ext}')]

print(f'Containing number of files: {len(all_files)}.')
print(all_files)

os.chdir('../')
```

```
Containing number of files: 23.
['coin_XRP.csv', 'coin_Ethereum.csv', 'coin_Aave.csv', 'coin_USDCoin.csv',
'coin_Solana.csv', 'coin_EOS.csv', 'coin_WrappedBitcoin.csv',
'coin_Cardano.csv', 'coin_NEM.csv', 'coin_Tron.csv', 'coin_CryptocomCoin.csv',
'coin_BinanceCoin.csv', 'coin_Iota.csv', 'coin_ChainLink.csv',
'coin_Monero.csv', 'coin_Dogecoin.csv', 'coin_Tether.csv', 'coin_Bitcoin.csv',
'coin_Cosmos.csv', 'coin_Litecoin.csv', 'coin_Stellar.csv', 'coin_Polkadot.csv',
'coin_Uniswap.csv']
```

Are there duplicated in each files? Lets check duplicated rows.

[5]:
```
for f in all_files:
    temp = pd.read_csv(path+f)
    print(f'duplicated in {f}:', temp.index.duplicated().sum())
```

```
duplicated in coin_XRP.csv: 0
duplicated in coin_Ethereum.csv: 0
duplicated in coin_Aave.csv: 0
duplicated in coin_USDCoin.csv: 0
duplicated in coin_Solana.csv: 0
duplicated in coin_EOS.csv: 0
duplicated in coin_WrappedBitcoin.csv: 0
duplicated in coin_Cardano.csv: 0
```

```
duplicated in coin_NEM.csv: 0
duplicated in coin_Tron.csv: 0
duplicated in coin_CryptocomCoin.csv: 0
duplicated in coin_BinanceCoin.csv: 0
duplicated in coin_Iota.csv: 0
duplicated in coin_ChainLink.csv: 0
duplicated in coin_Monero.csv: 0
duplicated in coin_Dogecoin.csv: 0
duplicated in coin_Tether.csv: 0
duplicated in coin_Bitcoin.csv: 0
duplicated in coin_Cosmos.csv: 0
duplicated in coin_Litecoin.csv: 0
duplicated in coin_Stellar.csv: 0
duplicated in coin_Polkadot.csv: 0
duplicated in coin_Uniswap.csv: 0
```

So there is no duplicated rows. We carry out combining files.

[6]:
```
combined_csv = pd.concat([pd.read_csv(path + fname) for fname in all_files])
print('duplicated in combine files: ', combined_csv.index.duplicated().sum())
```

```
duplicated in combine files:  34091
```

Why the combined files take duplicated? Because column 'SNo' is the index starting with 1 for each files. So we must reset the index in combined files.

[7]:
```
combined_csv.drop(columns='SNo',inplace = True)
combined_csv.to_csv('crypto_finance.csv', index=False)

combined_csv = pd.read_csv('crypto_finance.csv')
combined_csv.index.rename('SNo', inplace = True)
combined_csv.to_csv('crypto_finance.csv')
```

1.4.2 2. Pre-processing

Read new file. Let's see some values.

[8]:
```
cryp_df = pd.read_csv('crypto_finance.csv', index_col='SNo')

cryp_df.sample(10)
```

[8]:
```
                Name Symbol                 Date        High         Low  \
SNo
22767       Dogecoin   DOGE  2015-12-15 23:59:59    0.000153    0.000145
2549             XRP    XRP  2020-07-28 23:59:59    0.233693    0.217894
11085            NEM    XEM  2016-10-28 23:59:59    0.003937    0.003696
23259       Dogecoin   DOGE  2017-04-20 23:59:59    0.000483    0.000438
18607      Chainlink   LINK  2019-03-31 23:59:59    0.535640    0.488088
4345        Ethereum    ETH  2019-07-30 23:59:59  213.614075  206.867613
```

```
11237             NEM    XEM  2017-03-29 23:59:59    0.014892    0.013265
398               XRP    XRP  2014-09-07 23:59:59    0.004813    0.004704
14559  Crypto.com Coin   CRO  2019-12-19 23:59:59    0.033643    0.032308
8059              EOS    EOS  2020-12-30 23:59:59    2.670463    2.551347

             Open       Close        Volume     Marketcap
SNo
22767    0.000145    0.000150  2.357910e+05  1.533710e+07
2549     0.223889    0.230277  1.754696e+09  1.032762e+10
11085    0.003937    0.003743  6.190920e+04  3.368925e+07
23259    0.000448    0.000443  1.828250e+06  4.830447e+07
18607    0.494278    0.507613  9.259723e+06  1.776646e+08
4345   211.339197  210.522604  5.489919e+09  2.255101e+10
11237    0.013594    0.014892  6.480060e+05  1.340280e+08
398      0.004796    0.004704  8.785810e+04  1.363579e+08
14559    0.033246    0.033331  5.710471e+06  4.144266e+08
8059     2.642033    2.609923  2.556476e+09  2.450865e+09
```

More information about dataset.

[9]:
```python
cryp_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37082 entries, 0 to 37081
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Name       37082 non-null  object
 1   Symbol     37082 non-null  object
 2   Date       37082 non-null  object
 3   High       37082 non-null  float64
 4   Low        37082 non-null  float64
 5   Open       37082 non-null  float64
 6   Close      37082 non-null  float64
 7   Volume     37082 non-null  float64
 8   Marketcap  37082 non-null  float64
dtypes: float64(6), object(3)
memory usage: 2.8+ MB
```

How many rows and columns in the file?

[10]:
```python
cryp_df.shape
```

[10]:
```
(37082, 9)
```

There are 37082 rows and 9 columns.

What is the meaning of each row (sample)? The dataset shows that each samples contain the information of a specific coin at each time.

Does the data have duplicated samples?

[11]:
```
cryp_df.index.duplicated().sum()
```

[11]:
```
0
```

There is no duplicated samples.

What does each column mean?

[12]:
```
f = open('description.txt', 'r')
print(f.read())
f.close()
```

```
1) SNo:        Serial Number.
2) Name:       Name of coin.
3) Symbol:     Symbol of coin.
4) Date:       Date of observation.
5) Open:       Opening price on the given day.
6) High:       Highest price on the given day.
7) Low:        Lowest price on the given day.
8) Close:      Closing price on the given day.
9) Volume:     Volume of transactions on the given day (in USD).
10) Market Cap: Market capitalization in USD.
```

Check the data types?

[13]:
```
cryp_df.dtypes
```

[13]:
```
Name          object
Symbol        object
Date          object
High         float64
Low          float64
Open         float64
Close        float64
Volume       float64
Marketcap    float64
dtype: object
```

Overall, all the column types is correct except `Date`. We need to change `Date` column to appropriate type (`datetime`).

[14]:
```
cryp_df['Date'] = pd.to_datetime(cryp_df['Date'], format="%Y-%m-%d %X")
```

With the numeric columns, how are values distributed? Is there any abnormal value?

[15]:
```
nume_cols_df = cryp_df.describe(include=['number', 'datetime'],
    datetime_is_numeric=True)
nume_cols_df.loc['missing_ratio'] = cryp_df.agg(lambda x: x.isna().mean()*100)
```

```
nume_cols_df
```

[15]:

```
                                     Date          High           Low  \
count                               37082  37082.000000  37082.000000
mean        2018-08-16 07:12:30.221616896   1016.058015    952.987707
min                   2013-04-29 23:59:59      0.000089      0.000079
25%                   2017-03-05 23:59:59      0.075664      0.069536
50%                   2019-01-09 23:59:59      1.008733      0.999850
75%                   2020-05-13 23:59:59     31.916399     28.996246
max                   2021-07-06 23:59:59  64863.098908  62208.964366
std                                   NaN   5249.503670   4907.932082
missing_ratio                         0.0      0.000000      0.000000

                      Open         Close        Volume     Marketcap
count         37082.000000  37082.000000  3.708200e+04  3.708200e+04
mean            985.323755    987.120511  3.022542e+09  1.542943e+10
min               0.000086      0.000086  0.000000e+00  0.000000e+00
25%               0.072456      0.072648  4.937190e+06  2.395955e+08
50%               1.001157      1.001138  8.512805e+07  1.405335e+09
75%              30.459673     30.512205  9.388489e+08  5.159305e+09
max           63523.754869  63503.457930  3.509679e+11  1.186364e+12
std            5088.101367   5093.703878  1.190963e+10  7.059128e+10
missing_ratio     0.000000      0.000000  0.000000e+00  0.000000e+00
```

There is no missing value in DataFrame. But it exists abnormal values in `min` row of `Volume` and `Marketcap` col.

So, lets take a look at these columns.

[16]:

```
check_df = cryp_df[(cryp_df['Marketcap'] == 0) | (cryp_df['Volume'] == 0)]
check_df
```

[16]:

```
           Name Symbol                Date      High       Low      Open  \
SNo
0           XRP    XRP 2013-08-05 23:59:59  0.005980  0.005613  0.005875
1           XRP    XRP 2013-08-06 23:59:59  0.005661  0.004629  0.005637
2           XRP    XRP 2013-08-07 23:59:59  0.004682  0.004333  0.004669
3           XRP    XRP 2013-08-08 23:59:59  0.004424  0.004175  0.004397
4           XRP    XRP 2013-08-09 23:59:59  0.004367  0.004253  0.004257
…           …    …                  …       …         …         …
36477  Polkadot    DOT 2020-08-28 23:59:59  6.333746  5.540963  5.639486
36478  Polkadot    DOT 2020-08-29 23:59:59  6.562906  6.042309  6.175925
36479  Polkadot    DOT 2020-08-30 23:59:59  6.219506  5.749978  6.153440
36480  Polkadot    DOT 2020-08-31 23:59:59  6.459377  5.772966  5.905918
36481  Polkadot    DOT 2020-09-01 23:59:59  6.838898  6.172324  6.298810

          Close        Volume      Marketcap
```

```
SNo
0      0.005613  0.000000e+00  4.387916e+07
1      0.004680  0.000000e+00  3.659101e+07
2      0.004417  0.000000e+00  3.453412e+07
3      0.004254  0.000000e+00  3.325863e+07
4      0.004291  0.000000e+00  3.354750e+07
…         …           …             …
36477  6.159955  7.271622e+08  0.000000e+00
36478  6.159143  5.272900e+08  0.000000e+00
36479  5.869881  4.853351e+08  0.000000e+00
36480  6.300020  5.126048e+08  0.000000e+00
36481  6.288767  6.015274e+08  0.000000e+00

[971 rows x 9 columns]
```

We got some problems here: Abnormal values on `Volume` and `Marketcap` col. We can calculate the ratio of abnormal values to decide whether to drop or not.

[17]:
```
abn_df = check_df.groupby(['Symbol']).size()

all_df = cryp_df.groupby(['Symbol']).size()

(abn_df/all_df *100).dropna()
```

[17]:
```
Symbol
AAVE     0.363636
ATOM     5.443787
BTC      8.090939
CRO      0.213904
DOGE     0.398551
DOT      3.750000
EOS      0.068213
LTC      8.090939
SOL     11.504425
TRX      1.005747
USDC     0.798403
WBTC    22.072072
XRP      4.977532
dtype: float64
```

As the results, the number of invalid values is quite small, we can delete samples containing invalid values. But once we dropped it, we will face the intermittence of time data. So I decided not to drop.

With the categorical columns, how are the values distributed? Is there any abnormal value?

[18]:
```python
cate_cols = cryp_df.describe(include=[object]).columns

cate_cols_df = cryp_df[cate_cols].agg([
    lambda x: x.isna().mean() *100,
    pd.Series.nunique,
    pd.Series.unique])

cate_cols_df.rename(index ={
    '<lambda>': 'missing_ratio',
    'nunique':'num_dif_val',
    'unique':'diff_vals'
})
```

[18]:
```
                                                             Name  \
missing_ratio
0.0
num_dif_val
23
diff_vals      [XRP, Ethereum, Aave, USD Coin, Solana, EOS, Wrapped Bitcoin,
Cardano, NEM, TRON, Crypto.com Coin, Binance Coin, IOTA, Chainlink, Monero,
Dogecoin, Tether, Bitcoin, Cosmos, Litecoin, Stellar, Polkadot, Uniswap]

                                                     Symbol
missing_ratio
0.0
num_dif_val
23
diff_vals      [XRP, ETH, AAVE, USDC, SOL, EOS, WBTC, ADA, XEM, TRX, CRO, BNB,
MIOTA, LINK, XMR, DOGE, USDT, BTC, ATOM, LTC, XLM, DOT, UNI]
```

There is no missing value in categorical columns. Everything seems to be fine.

1.5 IV. Data Exploratory and Analysis.

1.5.1 Data Exploratory

which coin is stable/unstable? Before investing any markets, investors must know the basis to save their money. One of the most crucial knowledge is stable/unstable coins: - A stablecoin is a digital currency that is pegged to a "stable" reserve asset like the U.S dollar. - A unstable coin is a digital currency that has high price volatility, high inflation and weak economy.

We can find the answer by calculating the volatility.

Middle-term Investor (Month)

1. Preprocess

Now we are carrying out calculating the volatility of each coin month by month.

$$Monthly\_volatility = \frac{Values(Close\_end\_of\_month) - Values(Close\_month\_before)}{Values(Close\_month\_before)} * 100 \quad (1)$$

[19]:
```
last_dom_prep = cryp_df.reset_index('SNo').drop(columns=['SNo','Symbol'])
last_dom_prep['Month'] = last_dom_prep.Date.dt.month
last_dom_prep['Year'] = last_dom_prep.Date.dt.year

last_dom = last_dom_prep.groupby(['Name','Year','Month'])['Date'].max().
 ↪reset_index()

#make condition
Name = (cryp_df['Name'].isin(last_dom['Name']))
date = (cryp_df['Date'].isin(last_dom['Date']))

#gat all cols of last date each month and each Name.
last_dom_df = last_dom_prep[Name & date]
last_dom_df=last_dom_df.drop(columns=['Month','Year'])

#find min date and max date of each coin => delete min date and duplicate max␣
 ↪date.
min_max = last_dom_df[['Name','Date']].groupby('Name').agg([min,max])
min_max = min_max.reset_index(col_level=1).droplevel(level=0,axis=1).
 ↪set_index(['Name','min'])

substr = last_dom_df.set_index(['Name','Date']).drop(index=min_max.index)

min_max = min_max.reset_index().set_index(['Name','max'])
besubstr = last_dom_df.set_index(['Name','Date']).drop(index=min_max.index).
 ↪reset_index()

#increase to end of next month
besubstr = besubstr.assign(
    Date = besubstr.Date + pd.Timedelta(days=1)
    )

besubstr = besubstr.assign(
    Date = besubstr.Date.apply(
        lambda x:x+ relativedelta.relativedelta(months=1)
        )
    )

besubstr = besubstr.assign(
    Date = besubstr.Date - pd.Timedelta(days=1)
    )
besubstr = besubstr.set_index(['Name','Date'])
```

```python
#calculating the volatility
res = ((substr - besubstr)*100/besubstr).reset_index()
res = pd.pivot(res, columns = 'Name',index='Date',values='Close')

res
```

[19]:
```
Name                      Aave  Binance Coin    Bitcoin    Cardano  Chainlink  \
Date
2013-05-31 23:59:59        NaN           NaN  -7.194245        NaN        NaN
2013-06-30 23:59:59        NaN           NaN -25.105428        NaN        NaN
2013-07-31 23:59:59        NaN           NaN   9.808100        NaN        NaN
2013-08-31 23:59:59        NaN           NaN  27.580366        NaN        NaN
2013-09-30 23:59:59        NaN           NaN  -1.736244        NaN        NaN
…                          …           …          …          …          …
2021-04-30 23:59:59  17.962651    106.605778  -1.983501  13.340260  29.904701
2021-05-31 23:59:59 -13.861344    -43.223781 -35.354564  28.860927 -15.840958
2021-06-30 23:59:59 -34.639432    -14.402834  -6.139409 -20.616898 -39.239339
2021-07-06 23:59:59        NaN           NaN        NaN        NaN        NaN
2021-07-31 23:59:59        NaN           NaN        NaN        NaN        NaN

Name                    Cosmos  Crypto.com Coin    Dogecoin        EOS  \
Date
2013-05-31 23:59:59        NaN              NaN         NaN        NaN
2013-06-30 23:59:59        NaN              NaN         NaN        NaN
2013-07-31 23:59:59        NaN              NaN         NaN        NaN
2013-08-31 23:59:59        NaN              NaN         NaN        NaN
2013-09-30 23:59:59        NaN              NaN         NaN        NaN
…                          …                …          …          …
2021-04-30 23:59:59  18.930354        -8.210962  529.105892  33.862844
2021-05-31 23:59:59 -38.427119       -36.641910   -3.481047   3.160672
2021-06-30 23:59:59 -14.224586        -8.173372  -21.974458 -37.777659
2021-07-06 23:59:59        NaN              NaN         NaN        NaN
2021-07-31 23:59:59        NaN              NaN         NaN        NaN

Name                  Ethereum  …        NEM   Polkadot      Solana  \
Date                            …
2013-05-31 23:59:59        NaN  …        NaN        NaN         NaN
2013-06-30 23:59:59        NaN  …        NaN        NaN         NaN
2013-07-31 23:59:59        NaN  …        NaN        NaN         NaN
2013-08-31 23:59:59        NaN  …        NaN        NaN         NaN
2013-09-30 23:59:59        NaN  …        NaN        NaN         NaN
…                          …  …        …          …          …
2021-04-30 23:59:59  44.561196  …  -7.944032  -1.313496  120.680681
2021-05-31 23:59:59  -2.100881  … -44.645530 -35.877113  -23.604979
2021-06-30 23:59:59 -16.221234  … -31.126707 -29.997673    8.350011
2021-07-06 23:59:59        NaN  …        NaN        NaN         NaN
```

```
2021-07-31 23:59:59        NaN  …        NaN        NaN         NaN

Name                    Stellar       TRON    Tether  USD Coin    Uniswap  \
Date
2013-05-31 23:59:59        NaN        NaN       NaN       NaN        NaN
2013-06-30 23:59:59        NaN        NaN       NaN       NaN        NaN
2013-07-31 23:59:59        NaN        NaN       NaN       NaN        NaN
2013-08-31 23:59:59        NaN        NaN       NaN       NaN        NaN
2013-09-30 23:59:59        NaN        NaN       NaN       NaN        NaN
…                           …          …         …         …          …
2021-04-30 23:59:59  28.870459  43.241110 -0.001135  0.021877  45.418800
2021-05-31 23:59:59 -23.817326 -41.827906  0.052419  0.044035 -30.291554
2021-06-30 23:59:59 -29.853028 -11.406481 -0.031692 -0.017570 -32.045981
2021-07-06 23:59:59        NaN        NaN       NaN       NaN        NaN
2021-07-31 23:59:59        NaN        NaN       NaN       NaN        NaN

Name                  Wrapped Bitcoin         XRP
Date
2013-05-31 23:59:59               NaN         NaN
2013-06-30 23:59:59               NaN         NaN
2013-07-31 23:59:59               NaN         NaN
2013-08-31 23:59:59               NaN         NaN
2013-09-30 23:59:59               NaN   94.118035
…                                 …          …
2021-04-30 23:59:59         -2.008534  177.358124
2021-05-31 23:59:59        -35.398383  -34.246312
2021-06-30 23:59:59         -6.154983  -32.506669
2021-07-06 23:59:59               NaN         NaN
2021-07-31 23:59:59               NaN         NaN

[102 rows x 23 columns]
```

2. Visualization

Visualizing the volatility of each coin to observe clearly. From now on, we can assume a coin is stable if its volatility is not beyond to $[-15, 15]$ and the data is enough for our decisions.

[20]:
```
# fig, axs = plt.subplots(int(res.columns.size/2 + 1),2, figsize=(15,60))
# fig.tight_layout()
# for i in range(res.columns.size):
#     axs[int(i/2),i%2].plot(res[res.columns.values[i]])
#     axs[int(i/2),i%2].set_title(res.columns.values[i])

fig = make_subplots(rows = int(res.columns.size/2 + 1), cols = 2,
                    vertical_spacing=0.02,
                    subplot_titles=tuple(res.columns.values))

for i in range(res.columns.size):
```

```
    fig.add_trace(
        go.Scatter(x = res.index,
                    y =res[res.columns.values[i]],
                    mode='lines',
                    name = res.columns.values[i]
                ),
        row = int(i/2)+1, col = i%2 + 1
        )

fig.update_layout(height=4000,
                showlegend=False,
                title_text="Volatility of each coin")
fig.show()
```

Long-term Investor (Year) We could take a look at the vilatility year by year if we are a long-term investor.

All processes like above. We appoximates the yearly volatility base on monthly volatility.

Standard Deviation of monthly volatility.

$$STD\_Monthly\_volatility = \sqrt{\frac{\sum(vola\_month_i - \mu)^2}{N = Num\_of\_month}} \tag{2}$$

Anually volatility.

$$Annual\_volatility = STD\_Monthly\_volatility\sqrt{Num\_of\_month} \tag{3}$$

1. Pre-processing

[21]:
```
res1 = ((substr-besubstr)*100/besubstr).reset_index()

#groupby year
res1['Year'] = res1.Date.dt.year
res1 = res1[['Name','Year','Close']].groupby(['Name','Year']).agg([np.std,np.
 ↪size])\

 ↪reset_index(col_level=1)
res1 = res1.droplevel(level=0,axis=1)

#Monthly volatility = standard deviation of month-base volatility.
#Annual volatility = Monthly Volatility * sqrt(num time period)
res1['AnnualVola'] = res1['std']*np.sqrt(res1['size'])
res1 = pd.pivot(res1, columns = 'Name',index='Year',values='AnnualVola')
res1
```

[21]:

```
Name        Aave  Binance Coin     Bitcoin     Cardano   Chainlink  \
Year
2013         NaN           NaN  457.109181         NaN         NaN
2014         NaN           NaN   66.556166         NaN         NaN
2015         NaN           NaN   66.726704         NaN         NaN
2016         NaN           NaN   49.309806         NaN         NaN
2017         NaN   1955.111950   95.957177  206.464997  290.086523
2018         NaN     73.193491   70.876683  158.615169  147.557427
2019         NaN    119.253488   77.494807   85.825074  267.470088
2020  142.704238     72.601705   79.796929  141.519815  156.576276
2021  287.504510    430.664039   74.921018  318.187039  135.334441

Name      Cosmos  Crypto.com Coin    Dogecoin         EOS    Ethereum  …  \
Year                                                                   …
2013         NaN              NaN         NaN         NaN         NaN  …
2014         NaN              NaN  362.003329         NaN         NaN  …
2015         NaN              NaN   86.039260         NaN   59.311735  …
2016         NaN              NaN   95.502047         NaN  236.411589  …
2017         NaN              NaN  410.123269  320.308490  262.211431  …
2018         NaN              NaN  151.124506  222.328876  124.953128  …
2019   88.550594       272.493451   56.080850  113.331961   88.975236  …
2020  127.325853       109.703283   76.319909   97.297391  103.716362  …
2021  150.046174       157.141160  901.397567   78.032515   98.454019  …

Name         NEM    Polkadot      Solana     Stellar         TRON     Tether  \
Year
2013         NaN         NaN         NaN         NaN          NaN        NaN
2014         NaN         NaN         NaN  162.539517          NaN        NaN
2015   96.102720         NaN         NaN   58.689299          NaN   0.000000
2016  600.502428         NaN         NaN   93.278866          NaN   0.006333
2017  459.309983         NaN         NaN  688.910700  1892.737961  13.311229
2018  122.565094         NaN         NaN  158.527085   203.551788   3.668578
2019  103.686407         NaN         NaN   69.377906    85.089444   2.223591
2020  192.908441   89.340022  242.260489  187.488532    85.112881   1.125439
2021  215.813788  164.222431  266.540549  173.644518   141.507427   0.321865

Name  USD Coin     Uniswap  Wrapped Bitcoin         XRP
Year
2013       NaN         NaN              NaN  539.563746
2014       NaN         NaN              NaN  199.281864
2015       NaN         NaN              NaN  100.429210
2016       NaN         NaN              NaN   65.282208
2017       NaN         NaN              NaN  869.089489
2018  0.795658         NaN              NaN  132.638788
2019  2.669780         NaN        76.247803   62.627851
2020  3.647628   98.530297        81.547819  204.431611
2021  0.118152  285.846007        75.285197  252.159734
```

```
[9 rows x 23 columns]
```

2. Visualization

[22]:
```
# fig, axs = plt.subplots(int(res1.columns.size/2 + 1),2, figsize=(15,60))
# fig.tight_layout()
# for i in range(res1.columns.size):
#     axs[int(i/2),i%2].plot(res1[res1.columns.values[i]])
#     axs[int(i/2),i%2].set_title(res1.columns.values[i])

fig = make_subplots(rows = int(res1.columns.size/2 + 1), cols = 2,
                   vertical_spacing=0.02,
                    subplot_titles=tuple(res1.columns.values))

for i in range(res1.columns.size):
    fig.add_trace(
        go.Scatter(x = res1.index,
                    y =res1[res1.columns.values[i]],
                    mode='lines',
                    name = res1.columns.values[i]
                ),
        row = int(i/2)+1, col = i%2 + 1
        )

fig.update_layout(height=4000,
                showlegend=False,
                title_text="Volatility of each coin")
fig.show()
```

Conclusion Before deciding to invest a digital currency, investors must save their capital. How to save their own capital while almost digital currencies have a high volatility? ⇒ They must find and reserve in a stable coin.

According to the result, we can consider which one is a stablecoin. Here we have the coins take least volatility whether investors is middle-term or long-term: - Tether. - USD Coin.

Crypto currency proportion of 2021 To see crypto currency evolution, we could use `MarketCap` col.

[23]:
```
true_mcap = cryp_df[(cryp_df['Marketcap'] > 0) & (cryp_df['Date'].dt.year ==␣
 ↪2021)]

mean_mcap = true_mcap[['Name','Symbol','Marketcap']].groupby(['Name','Symbol']).
 ↪mean()
pd.set_option('display.float_format', '{:.2f}'.format)
mean_mcap
```

[23]:

```
                              Marketcap
Name            Symbol
Aave            AAVE       4342005247.88
Binance Coin    BNB       45863429127.74
Bitcoin         BTC      850308418885.51
Cardano         ADA       36073727647.25
Chainlink       LINK      11780371063.65
Cosmos          ATOM       3439341205.25
Crypto.com Coin CRO        3452219867.56
Dogecoin        DOGE      23253147261.34
EOS             EOS        4767997966.08
Ethereum        ETH      239910890392.16
IOTA            MIOTA      3403257862.25
Litecoin        LTC       13150059074.47
Monero          XMR        4459267395.45
NEM             XEM        2850426717.84
Polkadot        DOT       25611960118.95
Solana          SOL        6176877049.63
Stellar         XLM        9270609681.84
TRON            TRX        5192913205.79
Tether          USDT      44067450019.42
USD Coin        USDC      12663720269.43
Uniswap         UNI       12035294126.12
Wrapped Bitcoin WBTC       6764035608.35
XRP             XRP       33799135106.02
```

Visualyzing…

[24]:
```
mean_mcap = mean_mcap.reset_index()

fig = px.pie(mean_mcap, values= 'Marketcap', names='Name',
            title="Proportion of currency in 2021")
fig.update_traces(textposition = 'inside',textinfo='percent+label')
```

Result

- Bitcoin takes the highest market capitalization (60.6%), and the second is Etherium(17.1%).

As a result, it is said that Bitcoin is the main part of crypto currency, and other coins are assumed to be alternative coins (called altcoins).

Bitcoin History By Year History price chart.

[25]:
```
btc_df = cryp_df[cryp_df['Symbol'] == 'BTC'].drop(columns='Marketcap')

fig = make_subplots(cols=1, rows=2,
                shared_xaxes=True,
                vertical_spacing=0.05,
                subplot_titles=('Bitcoin history price','Volume'))
```

```
fig.add_trace(go.Candlestick(x=btc_df['Date'],
                    open=btc_df['Open'],
                    close = btc_df['Close'],
                    high=btc_df['High'],
                    low = btc_df['Low'],
                    ),
            row=1,col=1
)

btc_df = btc_df[btc_df['Volume'] > 0]

fig.add_trace(
    go.Scatter(x=btc_df['Date'], y=btc_df['Volume']),
    row=2, col =1
)

fig.update_layout(height=1000,
                xaxis_rangeslider_visible=False,
                showlegend=False,
                title='Bitcoin analysis')

fig.show()
```

1.5.2 Propose meaningful question & answer.

After preprocessing and exploring, we had an overview about the dataset. Lets turn back to our purpose. There are 2 main questions we must answer: - How does the price fluctuations of currencies correlate with each other? - Which coins do we need to take consideration?

1. How does the price fluctuations of currencies correlate with each other? With this question and the results of data exploratory, we could know how the digital currency work. The answer of this question tells us we should invest alternative coins or bitcoin.

We choose top_7 coins and stable coins except Bitcoin representing alternative coins.

[26]:
```
stb_coin=['USDT','USDC']
mean_mcap = mean_mcap.sort_values('Marketcap',ascending=False)
stb_coin = mean_mcap[mean_mcap['Symbol'].isin(stb_coin)]
alt_and_BTC_coin = mean_mcap[:8]

alt_and_BTC_coins = pd.concat([stb_coin,alt_and_BTC_coin],axis=0)
alt_and_BTC_coins = alt_and_BTC_coins.drop_duplicates()

corr_df = cryp_df[cryp_df['Symbol'].isin(alt_and_BTC_coins['Symbol'])]
corr_df = pd.pivot(corr_df,values='Close', index='Date',columns='Symbol')

corr_df
```

[26]:
```
Symbol                ADA    BNB      BTC  DOGE   DOT     ETH  USDC  USDT  XRP
Date
2013-04-29 23:59:59  NaN    NaN   144.54   NaN   NaN     NaN   NaN   NaN  NaN
2013-04-30 23:59:59  NaN    NaN   139.00   NaN   NaN     NaN   NaN   NaN  NaN
2013-05-01 23:59:59  NaN    NaN   116.99   NaN   NaN     NaN   NaN   NaN  NaN
2013-05-02 23:59:59  NaN    NaN   105.21   NaN   NaN     NaN   NaN   NaN  NaN
2013-05-03 23:59:59  NaN    NaN    97.75   NaN   NaN     NaN   NaN   NaN  NaN
…                     …     …        …     …     …       …     …     …    …
2021-07-02 23:59:59 1.39 287.42 33897.05  0.25 15.34 2150.04  1.00  1.00 0.66
2021-07-03 23:59:59 1.41 298.24 34668.55  0.25 15.55 2226.11  1.00  1.00 0.67
2021-07-04 23:59:59 1.46 307.73 35287.78  0.25 16.01 2321.72  1.00  1.00 0.69
2021-07-05 23:59:59 1.40 302.38 33746.00  0.23 15.24 2198.58  1.00  1.00 0.65
2021-07-06 23:59:59 1.42 320.93 34235.19  0.23 16.14 2324.68  1.00  1.00 0.67

[2991 rows x 9 columns]
```

Calculating correlation efficient.

[27]:
```
corr_df.corr()
```

[27]:
```
Symbol   ADA   BNB   BTC  DOGE   DOT   ETH  USDC  USDT   XRP
Symbol
ADA     1.00  0.89  0.86  0.83  0.85  0.96 -0.24 -0.07  0.65
BNB     0.89  1.00  0.86  0.89  0.85  0.90 -0.22 -0.08  0.48
BTC     0.86  0.86  1.00  0.65  0.95  0.90 -0.32  0.04  0.62
DOGE    0.83  0.89  0.65  1.00  0.57  0.84 -0.17  0.00  0.49
DOT     0.85  0.85  0.95  0.57  1.00  0.84  0.02 -0.08  0.71
ETH     0.96  0.90  0.90  0.84  0.84  1.00 -0.27  0.03  0.70
USDC   -0.24 -0.22 -0.32 -0.17  0.02 -0.27  1.00  0.23 -0.11
USDT   -0.07 -0.08  0.04  0.00 -0.08  0.03  0.23  1.00  0.07
XRP     0.65  0.48  0.62  0.49  0.71  0.70 -0.11  0.07  1.00
```

Result & Conclusion Result

- Among Bitcoin and stable coins, we has negative correlation (weak correlation with USDT). And with others coins, it is shown that almost cases have negative correlations (weak correlation with USDT).
- Among Bitcoin and alternative coins, we has strong positive correlations.

Conclusion

- As the results showed us, We could choose alternative coins to invest by following Bitcoin. This conclusion implices that we could invest alternative coins to optimize profits and following Bitcoin if its profit is greater.
- Secondly, we can understand a little bit about other investors' sentiment: if the scenario is bullish market, some investors exchanged from stable coins (USDC) to coins having higher volatility. ⇒ stable coins is down and other is up (negative correlation).

Analyze some results

[28]:
```python
eth_df = cryp_df[cryp_df['Symbol']=='ETH']
tether_df = cryp_df[cryp_df['Symbol']=='USDT']
```

[29]:
```python
fig = make_subplots(cols=1, rows=3,
                shared_xaxes=True,
                vertical_spacing=0.05,
                subplot_titles=('Bitcoin historical price',
                                'Ethereum historical price',
                                'Tether historical price'
                                ))
fig.add_trace(
    go.Candlestick(x=btc_df['Date'],
                    open=btc_df['Open'],
                    close = btc_df['Close'],
                    high=btc_df['High'],
                    low = btc_df['Low'],
                    ),
    row=1,col=1
)

fig.add_trace(
    go.Candlestick(x=eth_df['Date'],
                    open=eth_df['Open'],
                    close = eth_df ['Close'],
                    high=eth_df['High'],
                    low = eth_df['Low'],
                    ),
    row=2, col=1
)

fig.add_trace(
    go.Candlestick(x=tether_df['Date'],
                    open=tether_df['Open'],
                    close = tether_df['Close'],
                    high=tether_df['High'],
                    low = tether_df['Low'],
                    ),
    row=3, col=1
)

for i in range(3):
    fig.update_xaxes(row=i+1, col=1, rangeslider_visible=False)

fig.update_layout(height=1500,
                showlegend=False,
                title='Crypto analysis')
```

```python
fig.show()
```

But, how to choose other coins for optimizing profits. We continue to the next question.

2. Which coins do we need to take considerations? We can answer this question by being greedy in profits... which coins has the best profits.

Do `High`,`Low`,`Open`,`Close` cols have values been less than or equal to 0?

[30]:
```python
(cryp_df[['High','Low','Open','Close']] <= 0).sum()
```

[30]:
```
High     0
Low      0
Open     0
Close    0
dtype: int64
```

About profits Get max profits by dividing max of high to min of low for each coins.

$$Profits\_of\_coins = \frac{Max(High)}{Min(Low)} \tag{4}$$

[31]:
```python
calc_coins = cryp_df[['Name','Symbol','High','Low']].groupby(['Name','Symbol']).
↪agg([np.min,np.max])

#Find max of High and min of Low
calc_coins = calc_coins.assign(
    High_max = calc_coins['High']['amax'],
    Low_min = calc_coins['High']['amin']
)
calc_coins = calc_coins.drop(columns=['High','Low']).droplevel(level=1,axis=1)

#Calculating profits
profits = pd.DataFrame(calc_coins['High_max'] / calc_coins['Low_min'])
profits = profits.rename({0:'Proportion'},axis=1)
profits = profits.reset_index().drop(columns='Symbol')

#
profits = profits.sort_values('Proportion',ascending=False)
profits
```

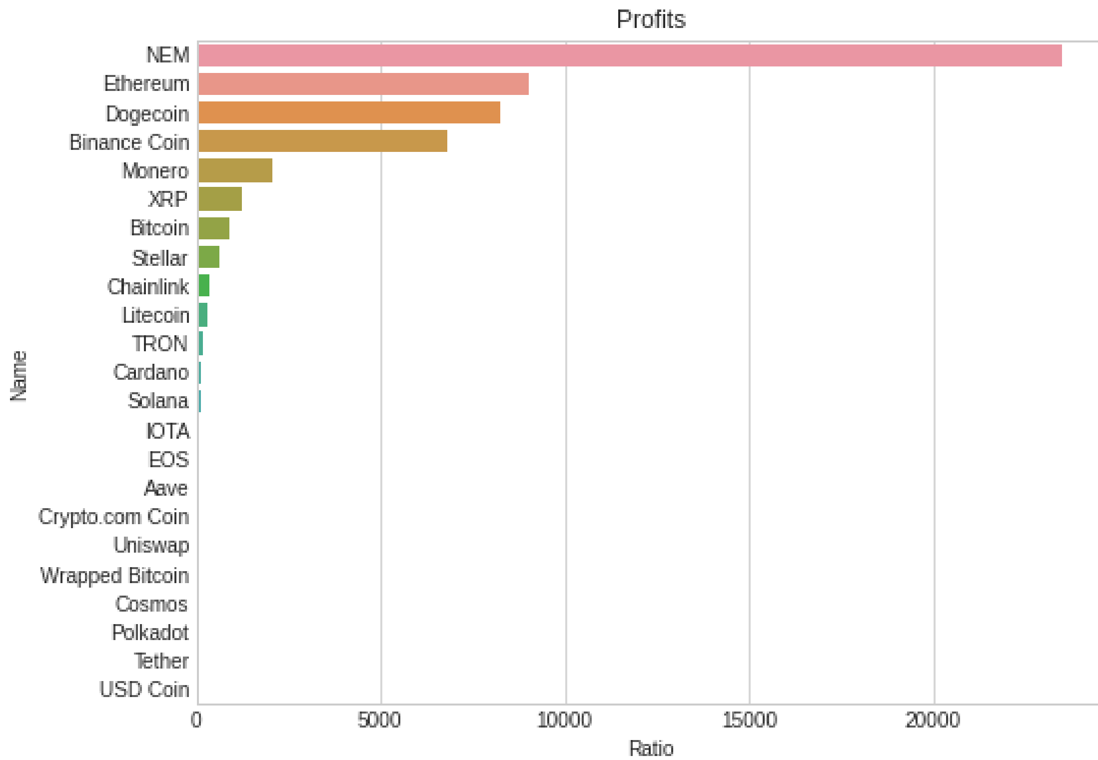
[31]:

	Name	Proportion
13	NEM	23506.73
9	Ethereum	9032.01
7	Dogecoin	8269.84
1	Binance Coin	6826.65
12	Monero	2056.03

```
22               XRP     1246.49
2            Bitcoin      869.93
16           Stellar      621.70
4          Chainlink      335.85
11          Litecoin      307.08
17              TRON      164.07
3            Cardano      116.95
15            Solana      104.16
10              IOTA       45.62
8                EOS       44.13
0               Aave       23.45
6    Crypto.com Coin       21.74
20           Uniswap       20.82
21   Wrapped Bitcoin       18.70
5             Cosmos       17.72
14          Polkadot       16.08
18            Tether        2.00
19          USD Coin        1.11
```

Visualyzing…

[32]:
```
plt.figure(figsize=(8,6))
plt.style.use('seaborn-whitegrid')
sns.barplot(x = profits['Proportion'], y = profits['Name'])
plt.title('Profits')
plt.xlabel('Ratio');
```



Conclusion

We can take consideration on NEM, Ethereum, Dogecoin, Binance Coin, Monero,... before actually investing.

About risks Calculating risk based on the last day of Dataset collected:

$$Risks = \frac{min(Low\_of\_last\_day) - max(High)}{max(High)} * 100 \tag{5}$$

[33]:
```
max_date = cryp_df[['Name','Symbol','Date']].groupby(['Name','Symbol']).max()\
                            .reset_index().set_index(['Name','Symbol','Date'])

#get last days
max_date_df = cryp_df.set_index(['Name','Symbol','Date'])
max_date_df = max_date_df[max_date_df.index.isin(max_date.index)]\
                        .reset_index().set_index(['Name','Symbol']).
 ↪drop(columns='Date')

#Take risks
risk = pd.DataFrame(
        (max_date_df['Low'] - calc_coins['High_max'])*100/calc_coins['High_max']
```

```
    ).rename({0:'Risks'},axis=1).reset_index()

#sorting values
risk = risk.sort_values('Risks')
risk
```

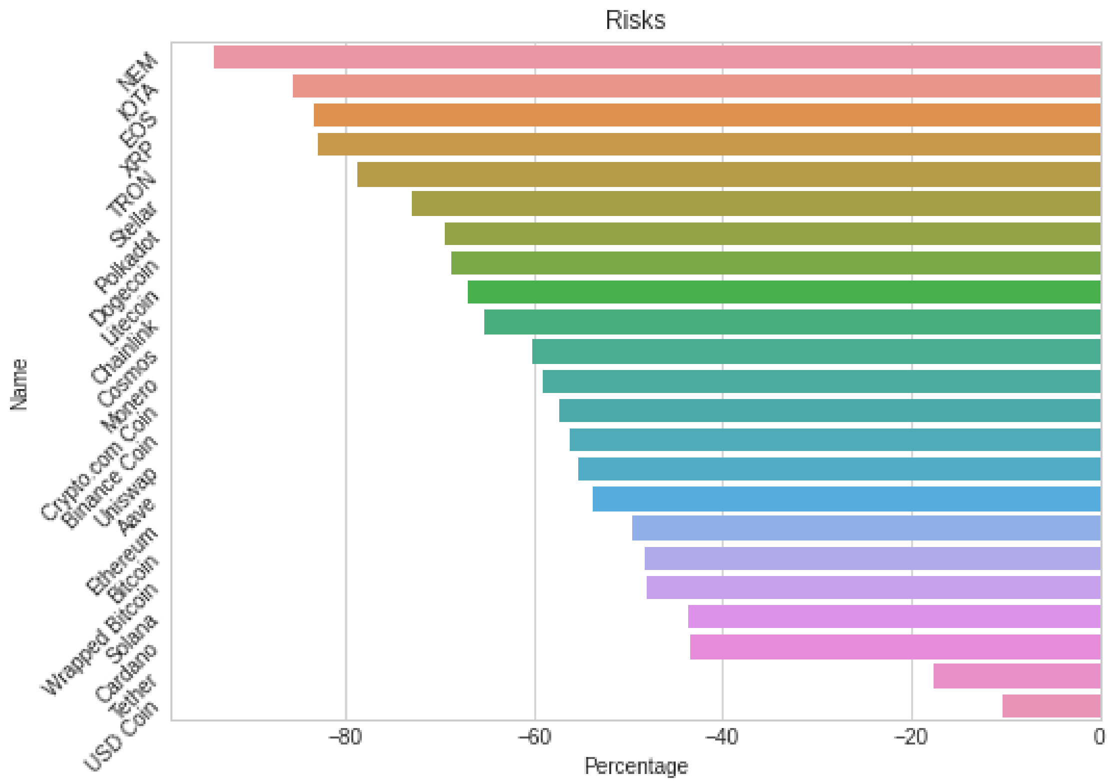
[33]:

```
                 Name Symbol  Risks
13                NEM    XEM -93.84
10               IOTA  MIOTA -85.58
8                 EOS    EOS -83.41
22                XRP    XRP -83.01
17               TRON    TRX -78.69
16            Stellar    XLM -72.97
14           Polkadot    DOT -69.39
7            Dogecoin   DOGE -68.84
11           Litecoin    LTC -67.09
4           Chainlink   LINK -65.29
5              Cosmos   ATOM -60.10
12             Monero    XMR -58.99
6     Crypto.com Coin    CRO -57.29
1        Binance Coin    BNB -56.26
20            Uniswap    UNI -55.36
0                Aave   AAVE -53.81
9            Ethereum    ETH -49.62
2             Bitcoin    BTC -48.20
21    Wrapped Bitcoin   WBTC -47.99
15             Solana    SOL -43.52
3             Cardano    ADA -43.40
18             Tether   USDT -17.53
19           USD Coin   USDC -10.24
```

Visualyzing…

[34]:

```
plt.figure(figsize=(8,6))
plt.style.use('seaborn-whitegrid')
sns.barplot(x = risk['Risks'], y = risk['Name'])
plt.title('Risks')
plt.yticks(rotation=45)
plt.xlabel('Percentage');
```



Conclusion

We can take consideration on NEM, IOTA,EOS, XRP... before actually investing.

Question Conclusion We can easily observe that some coins having very good profits(like NEM can bring to investors >20000 times based on their money in), but besides it, it could take the investors' money by substact 93.84 based on their money in.

So investors can pour out their money on the coins that take less risks, E.X: Solana or Bitcoin,...

Note:The answer to this question is for reference only

1.6 Project Reference

- Documents of subject Code for Data Science - HCMUS.
- Searching functions on Plotly - Plotly Documentation.
- Searching functions on Pandas - Pandas Documentation.
- Searching some financial formula - Wall street.