```
# Library installation
!pip install Historic-Crypto


# the crypto functions using
from Historic_Crypto import HistoricalData
from Historic_Crypto import Cryptocurrencies

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.decomposition import PCA
import numpy as np
import datetime


# Connecting to CoinBase Pro API to get the entire list of Crypto
Cryptocurrencies().find_crypto_pairs()


end_date = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M")
eth=HistoricalData('ETH-USD',60*60*24,'2021-01-01-00-00',end_date).retrieve_data()
#data extraction


eth


#Visualizing the data
eth.plot.line()



sns.pairplot(eth);
```

All of the price indicators are strongly corelated and the volume is not really corelated.

```
eth['price'] = eth[['low','high','open','close']].mean(axis=1)
eth


eth['price'].plot(logy=True);
```

The log space plot should be linear in an ideal case but becasue of several reasons, the value of Eth has been showing a downward plot. Which is not good

```
for item in Cryptocurrencies().find_crypto_pairs()['id'].tolist():
  if 'USD' in item: print(item)
#Listing only assets computed againt the USD


coins2eval = ['BTC-USD','ETH-USD', 'XLM-USD','ALGO-USD','DOGE-USD']
coinpricesD = {}
#Implementing a dictionary to store the data for each ticker and conerting that dictionary into a dataframe
end_date = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M")
for ticker in coins2eval:
  #importing the historical data
  tmp = HistoricalData(ticker,60*60*24,'2021-01-01-00-00',end_date).retrieve_data()
  #Taking the average price
  Avg = tmp[['low','high','open','close']].mean(axis=1)
  #Store that price in the dictionary
  coinpricesD[ticker] = Avg


coinpricesD


coinprices = pd.DataFrame(coinpricesD)
coinprices


coinprices.plot(); #Plotting the data as they are


#Scaling the all the columns from 0 to 1
coinpricesScaled = (coinprices-coinprices.min())/(coinprices.max()-coinprices.min())
```

```
coinpricesScaled.plot(title = 'Scaled to Max', fontsize=20, figsize=(10,6));


coinpricesScaled = (coinprices-coinprices.min())/(coinprices.loc['2021-11-11'].values[:]-coinprices.min()) #Scaled to a value of 1 at 11th of
coinpricesScaled.plot(title = 'Scaled to 1 at 11 Nov 2021', fontsize=20, figsize=(10,6));


coinprices.corr()


#PCA
pca = PCA()
pca.fit(coinprices.dropna())
#PCA for Scaled Values
pcaS = PCA()
pcaS.fit(coinpricesScaled.dropna())
#PCA for mean
coinpricesScaledCentered = coinpricesScaled.sub(coinpricesScaled.mean(axis=1),axis=0)
pcaM = PCA()
pcaM.fit(coinpricesScaledCentered.dropna())
plt.plot(100*pca.explained_variance_ratio_,'o-',label='Raw data')
plt.plot(100*pcaS.explained_variance_ratio_,'o-',label='Scaled data')
plt.plot(100*pcaM.explained_variance_ratio_,'o-',label='Centered data')
plt.xticks(range(pca.n_components_))
plt.xlabel('Components')
plt.ylabel('Percent Variance Explained')
plt.legend()
plt.title('Scree Plot')
plt.show()


#simulation 1

dailyInvest = 10

# which coin to simulate
whichCoin = 'ETH-USD'

#initialize our investment amounts
dolInvest = 0
coinInvest = 0

#loop through days
for dayi in range(coinprices.shape[0]):

  #buy some coin
  coin = dailyInvest/coinprices[whichCoin][dayi]
  #add to the totals
  dolInvest += dailyInvest
  coinInvest += coin

#compute the final value in euros of our investment
dolsAtEnd = coinInvest*coinprices[whichCoin][-1]

print(f'Total dollars invested: ${dolInvest:,.2f}')
print(f'Total {whichCoin[:-4]} purchased: {coinInvest:.7f}')
print(f'End result: ${dolsAtEnd:,.2f}')


coinprices[whichCoin][100]


#simulation 2

dailyInvestUp = 7
dailyInvestDn = 15
# which coin to simulate
whichCoin = 'ETH-USD'

#initialize our investment amounts
dolInvest = 0
coinInvest = 0

#loop through days
for dayi in range(1,coinprices.shape[0]):

  #buy some coin
  if(coinprices[whichCoin][dayi] > coinprices[whichCoin][dayi-1] ):
    coin = dailyInvestUp/coinprices[whichCoin][dayi]
  else:
```

```
  else:
    coin = dailyInvestDn/coinprices[whichCoin][dayi]
    dolInvest += dailyInvestDn

  #add to the totals
  coinInvest += coin

#compute the final value in euros of our investment
dolsAtEnd = coinInvest*coinprices[whichCoin][-1]

print(f'Total dollars invested: ${dolInvest:,.2f}')
print(f'Total {whichCoin[:-4]} purchased: {coinInvest:.7f}')
print(f'End result: ${dolsAtEnd:,.2f}')


#simulation 3

dailyInvestUp = 7
dailyInvestDn = 15
# which coin to simulate
whichCoin = 'ETH-USD'

#initialize our investment amounts
dolInvest = 0
coinInvest = 0

pctchng = [0]*coinprices.shape[0]

#loop through days
for dayi in range(1,coinprices.shape[0]):

  #compute the percent change from the previous day
  pctchng[dayi] = 100*(coinprices[whichCoin][dayi] - coinprices[whichCoin][dayi-1]) / coinprices[whichCoin][dayi-1]

  #buy some coin
  if pctchng[dayi]>0:
    toinvest = dailyInvest
  else:
    toinvest = dailyInvest * -pctchng[dayi]

  #add to the totals
  coin = toinvest / coinprices[whichCoin][dayi]
  dolInvest += toinvest
  coinInvest += coin

#compute the final value in euros of our investment
dolsAtEnd = coinInvest*coinprices[whichCoin][-1]

print(f'Total dollars invested: ${dolInvest:,.2f}')
print(f'Total {whichCoin[:-4]} purchased: {coinInvest:.7f}')
print(f'End result: ${dolsAtEnd:,.2f}')


plt.hist(pctchng,bins=40);
```