bitcoin

December 11, 2023

0.1 Data Acquisition

[2]:
```
import time
from collections import deque
from datetime import date, timedelta, datetime
from concurrent.futures import ThreadPoolExecutor

import requests
from loguru import logger

import pandas as pd
```

[3]:
```
api_v4_trades = "https://api.mercadobitcoin.net/api/v4/{symbol}/trades"
trades = deque()
```

[4]:
```
 # fetch most recent trades
response_trades = requests.get(url=api_v4_trades.format(symbol="BTC-BRL"))

if response_trades.status_code != 200:
    extra = {"error": response_trades}
    logger.bind(**extra).error("Error to fetch initial trade data")

trades.extend(response_trades.json())

initial_trade = trades[0]["tid"]
next_trades = initial_trade - 1000

logger.info(f"Initial trade: {initial_trade}")
```

```
2023-12-11 13:55:28.813 | INFO     |
__main__:<module>:13 - Initial trade:

15760543
```

[5]:
```
def fetch_trades(payload):
    response_trades = requests.get(url=api_v4_trades.format(symbol="BTC-BRL"),␣
 ↪params=payload)
    # logger.info(f"URL: {response_trades.url}")
```

```
    if response_trades.status_code != 200:
        logger.error(f"Error to fetch initial trade data: {response_trades.
↪text}")

    return response_trades.json()
```

[6]:
```
while next_trades > 0:
    with ThreadPoolExecutor() as executor:
        # time.sleep(1)
        future = executor.submit(fetch_trades, {"since": next_trades})
        future_result = future.result()
        trades.extendleft(reversed(future_result))
        next_trades -= 1000

        if future_result[-1]['date'] < 1696118349:
            logger.info("Done!")
            break
```

```
2023-12-11 13:56:58.933 | INFO     |
__main__:<module>:10 - Done!
```

[7]:
```
print(trades[0])
```

```
{'tid': 15502544, 'date': 1696021739, 'type': 'sell', 'price':
'135721.68121649', 'amount': '0.00007643'}
```

[8]:
```
columns = ["tid", "date", "type", "price", "amount"]

btc_trades_df = pd.DataFrame(trades, columns=[c for c in columns])
btc_trades_df.set_index('tid', inplace=True)

btc_trades_df['price'] = pd.to_numeric(btc_trades_df['price'])
btc_trades_df['amount'] = pd.to_numeric(btc_trades_df['amount'])
```

[9]:
```
btc_trades_df.drop_duplicates()
```

[9]:

tid	date	type	price	amount
15502544	1696021739	sell	135721.681216	7.643000e-05
15502545	1696021983	buy	135812.579450	2.945200e-04
15502546	1696022072	buy	135837.638253	1.457769e-02
15502547	1696022300	buy	135839.089990	3.625980e-03
15502548	1696022322	sell	135702.000000	2.332380e-03
…	…	…	…	…
15761538	1702313672	sell	206255.000000	3.280000e-01
15761539	1702313672	sell	206083.000000	3.667298e-01
15761540	1702313673	buy	206727.276741	4.000000e-08
15761541	1702313688	buy	206683.940000	7.257400e-04

```
15761542  1702313712   buy  206683.940000  1.488074e-02

[255701 rows x 4 columns]
```

[10]:
```
btc_trades_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 259000 entries, 15502544 to 15761542
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    259000 non-null  int64
 1   type    259000 non-null  object
 2   price   259000 non-null  float64
 3   amount  259000 non-null  float64
dtypes: float64(2), int64(1), object(1)
memory usage: 9.9+ MB
```

[11]:
```
btc_trades_df.head()
```

[11]:

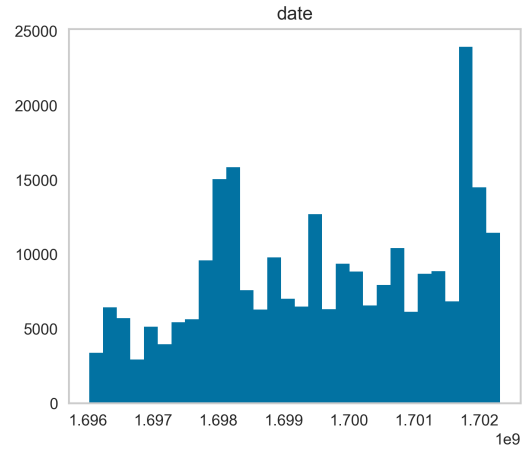
tid	date	type	price	amount
15502544	1696021739	sell	135721.681216	0.000076
15502545	1696021983	buy	135812.579450	0.000295
15502546	1696022072	buy	135837.638253	0.014578
15502547	1696022300	buy	135839.089990	0.003626
15502548	1696022322	sell	135702.000000	0.002332

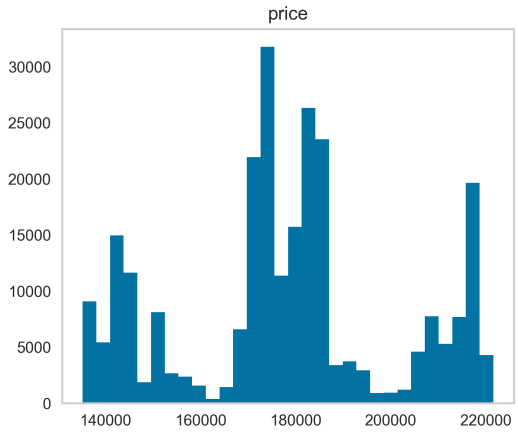
[12]:
```
btc_trades_df.to_csv("btc_trades_df.csv", sep='\t', index=False)
btc_trades_df.to_parquet("btc_trades_df.parquet", engine="fastparquet")
```

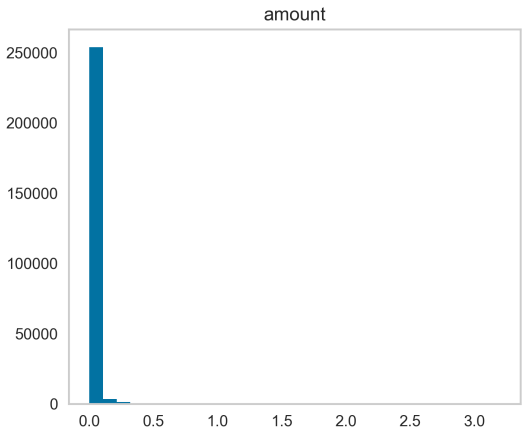
0.2 Exploratory Data Analysis

[13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
from pycaret.clustering import *
mpl.rcParams['figure.dpi'] = 300
```

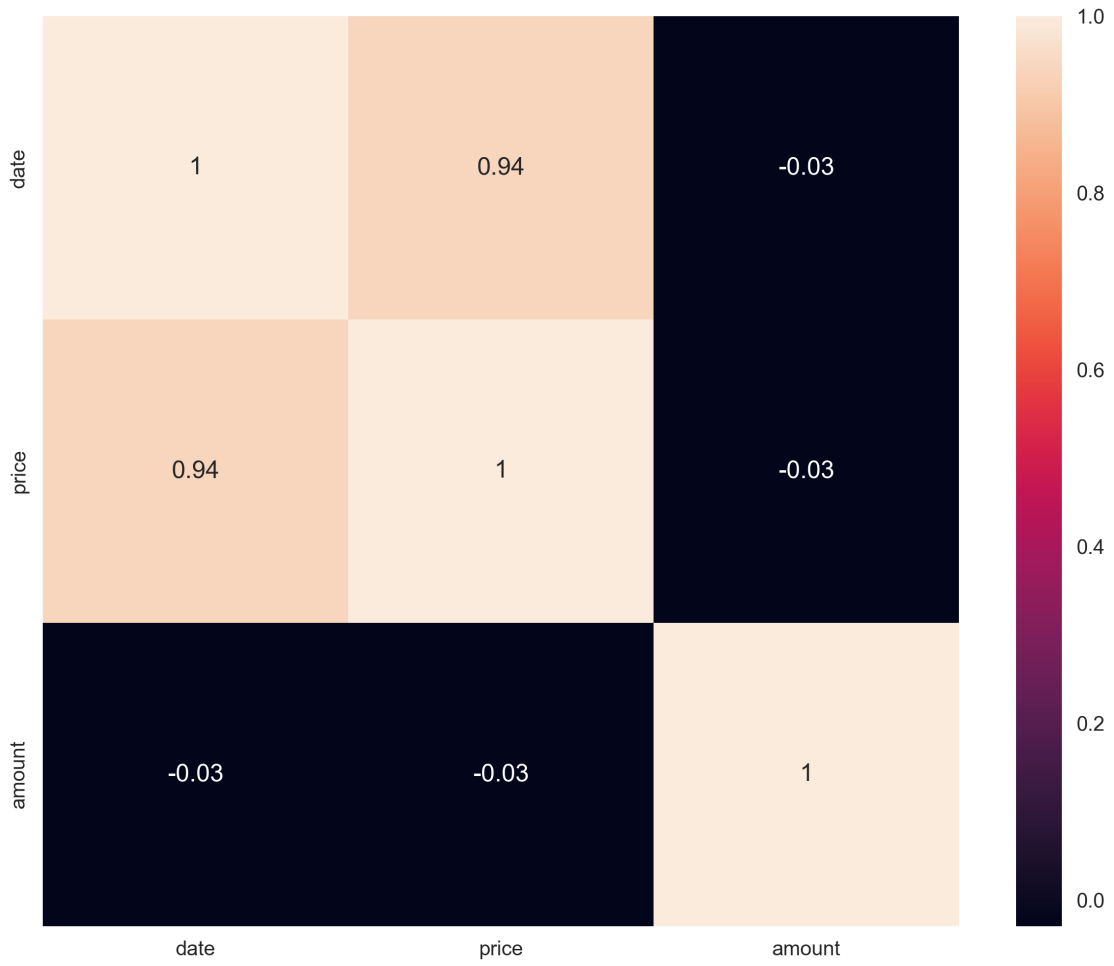
[14]:
```
btc_trades_df.hist(bins = 30, figsize = (12,10), grid = False)
plt.show()
```







[15]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(btc_trades_df.corr().round(decimals=2), annot=True)
plt.show()
```



[16]:
```
# plot_kws = {'scatter_kws': {'s': 2}, 'line_kws': {'color': 'red'}}
# sns.pairplot(btc_trades_df, kind='reg', vars=btc_trades_df['price'],␣
 ↪plot_kws=plot_kws)
# plt.show()
```

0.3 Model

[17]:
```
cluster = setup(btc_trades_df, session_id=7652, index=False, normalize=True,␣
 ↪ignore_features=["tid", "date"], use_gpu=True)
```

```
<pandas.io.formats.style.Styler at 0x29f04ac10>
```

[18]:
```
kmeans = create_model('kmeans')
```

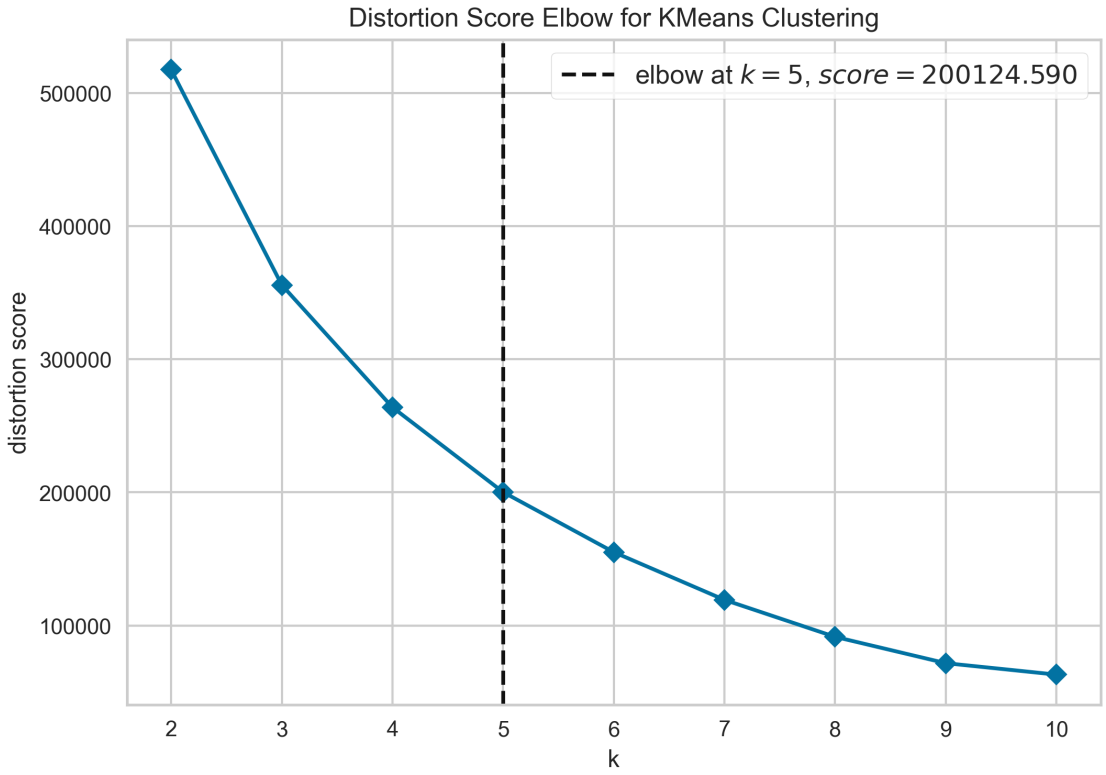
```
<IPython.core.display.HTML object>
```

```
<pandas.io.formats.style.Styler at 0x29fd48a50>
```

```
<IPython.core.display.HTML object>
```

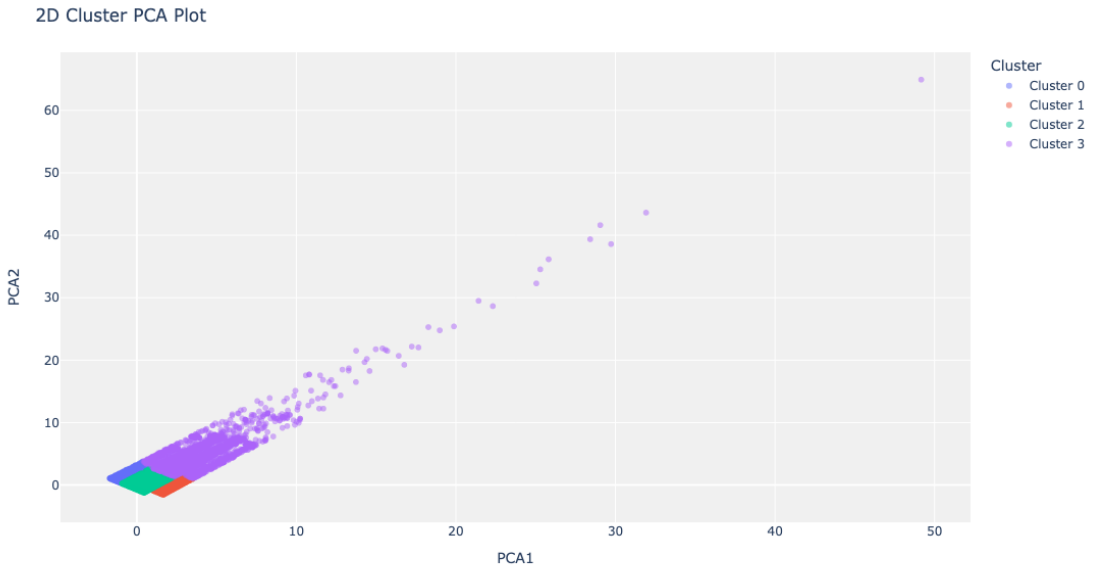
[24]:
```
plot_model(kmeans, 'elbow')
```

```
<IPython.core.display.HTML object>
```



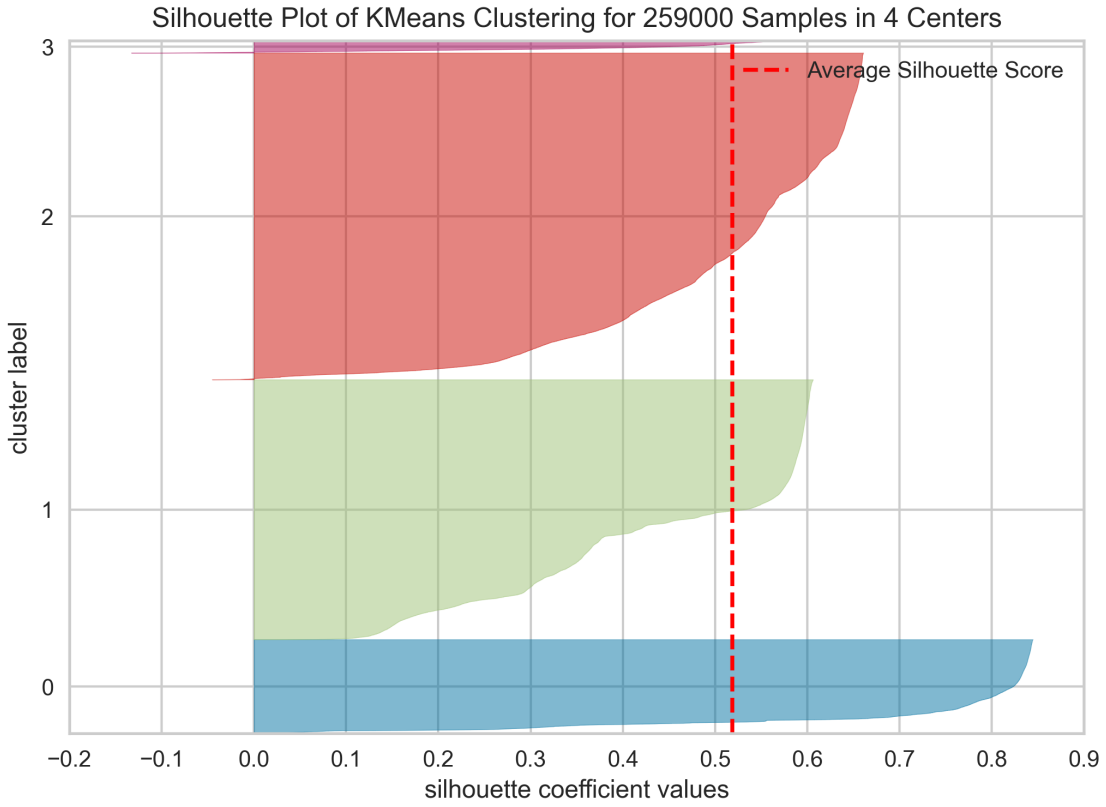
[20]:
```
plot_model(kmeans)
```

```
<IPython.core.display.HTML object>
```



[21]:
```
plot_model(kmeans, plot='silhouette')
```

```
<IPython.core.display.HTML object>
```



[22]:
```
plot_model(kmeans, plot='distribution')
```

```
<IPython.core.display.HTML object>
```