Trading

Choosing an API

There are a lot of exchanges for trading cryptocurrencies. Which all have pros and so leads to a comparison of APIs with respect to this project's use case. It is important that the API has a good and easy documentation and an extensive python library/wrapper.

The following table shows the full comparison of five cryptocurrency API's, which was made using this Article:

Platform	Binance	CoinBase	Kucoin	Coin
Pros	Directly sell and buy with the API, Connector to diff. Languages (python, Ruby), Commission when using BNB coin, 1200 free Requests per Minute, Binance has one of the biggest markets	API can be used as an Exchange or Wallet, Real-Time Notifications for Account Events	Many currencies(>200), Telegram Group for Support, 1800 Free Requests per MinuteCrypto Lending available	Goo lang 11 er
Cons	There have been outages in the past. When exceeding the transaction limits, the IP will be temporarily banned	Only etc, eth, btc cash, litecoin, Only 180 Requests per Minute	Performance issues (payments & trades hold back)	No h Pay histo 60 R Not
Historical Data	Yes	Yes	Yes	No
Fees	0.1%, but when using BNB can be lowered to 0.075%	1%	3-5%, when buying with FIAT 0.1% for crypto, 0.08% when buying with Kucoin Token	not f
Rating of Documentation	Very good	Good	Very good	Okay
Availability of Python Libraries/Wrappers	Very good and updated. https://github.com/sammchardy/python-binancehttps://github.com/binance/binance-connector-python	Last updated 8 years ago. https://github.com/resy/coinbase_python33	Very good and updated. https://github.com/sammchardy/python-kucoinhttps://github.com/Kucoin/kucoin-python-sdk	Not http coin

When it came to the decision which API to use, it was quite easy to filter out the ones that weren't suitable. CoinBase was the first to drop because it has only five coins and no currently active python libraries available.

Coinmarketcap was also not very intriguing because it has limited Endpoints and no historical Data. Paying 29$ a month for 1 month of historical data didn't seem worth it when others offer it for free. Their python wrapper was also not very detailed.

I personally use Kraken, so the ID Verification wouldn't even be a problem, but they do not offer a test/sandbox environment. Their way of working with request limits is also very different to the others and seemed to be a limiting factor. But the biggest problem with Kraken was that there is no recently updated python wrapper available.

Kucoin offers many currencies and the most free requests per minute, but reported some performance issues, which lead to the decision to Binance. At least at first.

Binance has one of the biggest markets with the highest liquidity and small fees. They have a very good documentation and an updated python wrapper with lots of tutorials. The only thing to keep in mind is the potential ban of my IP when the request limit is reached.

The Problem with Binance

The verification process was long and even required to send some Bitcoin to the Binance Wallet. Afterwards, it was possible to create the API-Keys, which are needed to connect the Client with Binance. Getting live price data from the Binance API was working perfectly, so everything seemed fine. Visualisations of Charts were made to have a better overview and make a small simulation to test the strategy (see below).

Then it was time to implement the trading strategy into the Binance sandbox and the problem appeared. The Sandbox Testnet didn't accept the API-Keys and even after a while of debugging the error, it wouldn't work, leaving us with no choice but to drop Binance for Kucoin. Setting up Kucoin was faster and their sandbox worked perfectly.

Kucoin API

After registration you need to go through a verification process. Typically, this is a call with a person that verifies your identity or uploading a photo of a personal ID-Card, but Kucoin only needs verification via Google Authenticator.

After that the API-Keys could be generated. These Keys are needed to connect and verify the Client with the API.

The *python-kucoin* wrapper helps with the Kucoin API, so there is no need to manually connect with the API via HTTP-Requests.

Acquiring data from the Kucoin API is fairly easy. Getting the latest price for BTCUSDT, the account balance, and making a market order are all done with a few lines of code. As can be seen below in the output, the current USDT Balance in the account is over 48k USDT. Kucoin offers every account a virtual amount of 250k USDT (~11 BTC) to trade with in their sandbox environment. They also offer Sub-Accounts, which is great to test a strategy for a specific amount of money, without risking to lose it all, when something goes wrong. This meant, papertrading could be performed and evaluated.

```
from kucoin.client import Client as kucoinClient

kClient = kucoinClient(kconf.KUCOIN_KEY, kconf.KUCOIN_SECRET,kconf.KUCOIN_PASS,sandbox=True)

tickers = kClient.get_ticker(symbol="BTC-USDT")
btc_price = tickers["bestAsk"]
print(f"Current BTC Price: {btc_price}")

accounts = kClient.get_accounts(account_type="trade")
usdt_balance = accounts[0]["balance"] #for Main Client
btc_balance = accounts[1]["balance"] #for Main client
btc_in_usdt = float(btc_balance) * float(btc_price)

print(f"USDT Balance: {usdt_balance} $")
print(f"BTC Balance: {btc_balance} ({btc_in_usdt} $)")

# Make a Market Order for 5% of the current USDT Balance
funds = re.match(r'\d+.\d{3}', str(usdt_balance*0.05)).group(0)
order = kClient.create_market_order('BTC-USDT', kClient.SIDE_BUY, funds = funds)
```

Output:

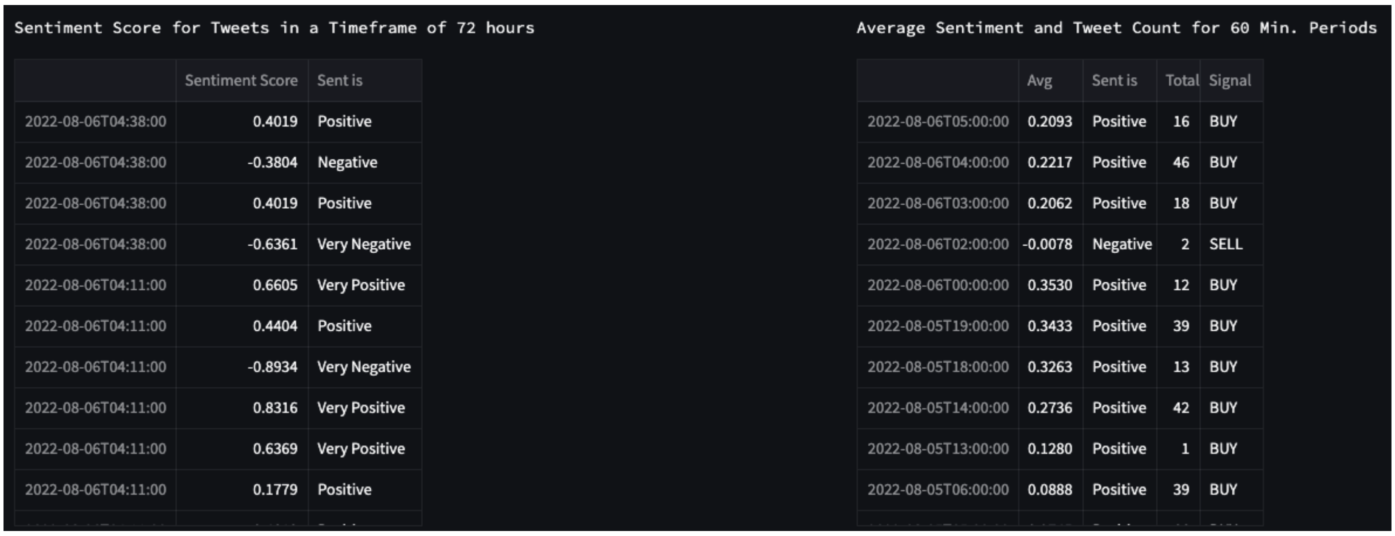
```
Current BTC Price: 22420.7

USDT Balance: 48942.07811348 $

BTC Balance: 0.00194057 (43.508937799 $)
```

Strategy

Building the strategy to make signals when a trade (buy or sell) should be executed, was a real challenge. There are hundreds of different strategies on different technical indicators, but now the buy or sell signal should be based on the sentiment of tweets on Twitter ([FR 30]).

At first, the idea was to calculate the average of sentiment for a particular time period and if this average reaches a certain threshold, a trade should be executed.

In the left table in Figure 14 you see the Timestamp, Sentiment Score and the corresponding meaning for *each tweet*.



Sentiment Score for Tweets in a Timeframe of 72 hours

	Sentiment Score	Sent is
2022-08-06T04:38:00	0.4019	Positive
2022-08-06T04:38:00	-0.3804	Negative
2022-08-06T04:38:00	0.4019	Positive
2022-08-06T04:38:00	-0.6361	Very Negative
2022-08-06T04:11:00	0.6605	Very Positive
2022-08-06T04:11:00	0.4404	Positive
2022-08-06T04:11:00	-0.8934	Very Negative
2022-08-06T04:11:00	0.8316	Very Positive
2022-08-06T04:11:00	0.6369	Very Positive
2022-08-06T04:11:00	0.1779	Positive

Average Sentiment and Tweet Count for 60 Min. Periods

	Avg	Sent is	Total	Signal
2022-08-06T05:00:00	0.2093	Positive	16	BUY
2022-08-06T04:00:00	0.2217	Positive	46	BUY
2022-08-06T03:00:00	0.2062	Positive	18	BUY
2022-08-06T02:00:00	-0.0078	Negative	2	SELL
2022-08-06T00:00:00	0.3530	Positive	12	BUY
2022-08-05T19:00:00	0.3433	Positive	39	BUY
2022-08-05T18:00:00	0.3263	Positive	13	BUY
2022-08-05T14:00:00	0.2736	Positive	42	BUY
2022-08-05T13:00:00	0.1280	Positive	1	BUY
2022-08-05T06:00:00	0.0888	Positive	39	BUY

Figure 14: Sentiment for a single tweet on the left, Average Sentiment and counted Tweets for a timeperiod of 1h

The right table shows the grouped and counted tweets for intervals of 60 min. Afterwards, the average sentiment is calculated and converted into a *Buy* or *Sell* Signal. It can be seen that there isn't a lot of tweets and some timestamps are missing. This was an error from SQL-Alchemy, which lead to the state of not committing the current Database-Session and thus leading to missing timestamps. This is fixed in the final version, but was kept here to correspond to the below chart.

The sentiment score for single tweets are all very different, but the average will naturally move towards neutral sentiment (0.0). This shift would have been even more if all the neutral tweets had not been removed beforehand.

It would have been easy to just define the buy signal when the average sentiment is positive and a sell signal when the average sentiment is negative. But this would mean a lot of buy signals for the above example and since the database is not quite big enough to see if this average is only positive at the moment or if the calculation just tends to be a little above neutral. The bigger the time intervals, the more tweets and the bigger the shift towards neutral. This is why the threshold is set to 0.2.

> An average sentiment above 0.2 means *Buy*, below 0.2 means *Sell*.

Papertrading

The image below shows the sentiment and bitcoin price from August 5th till August 6th. If the sentiment is above 0.2 (positive or very positive) it is marked as a buy signal in the chart (green triangle). Since this looked promising it was implemented to real-time Papertrading in the Kucoin Sandbox ([FR 40]).

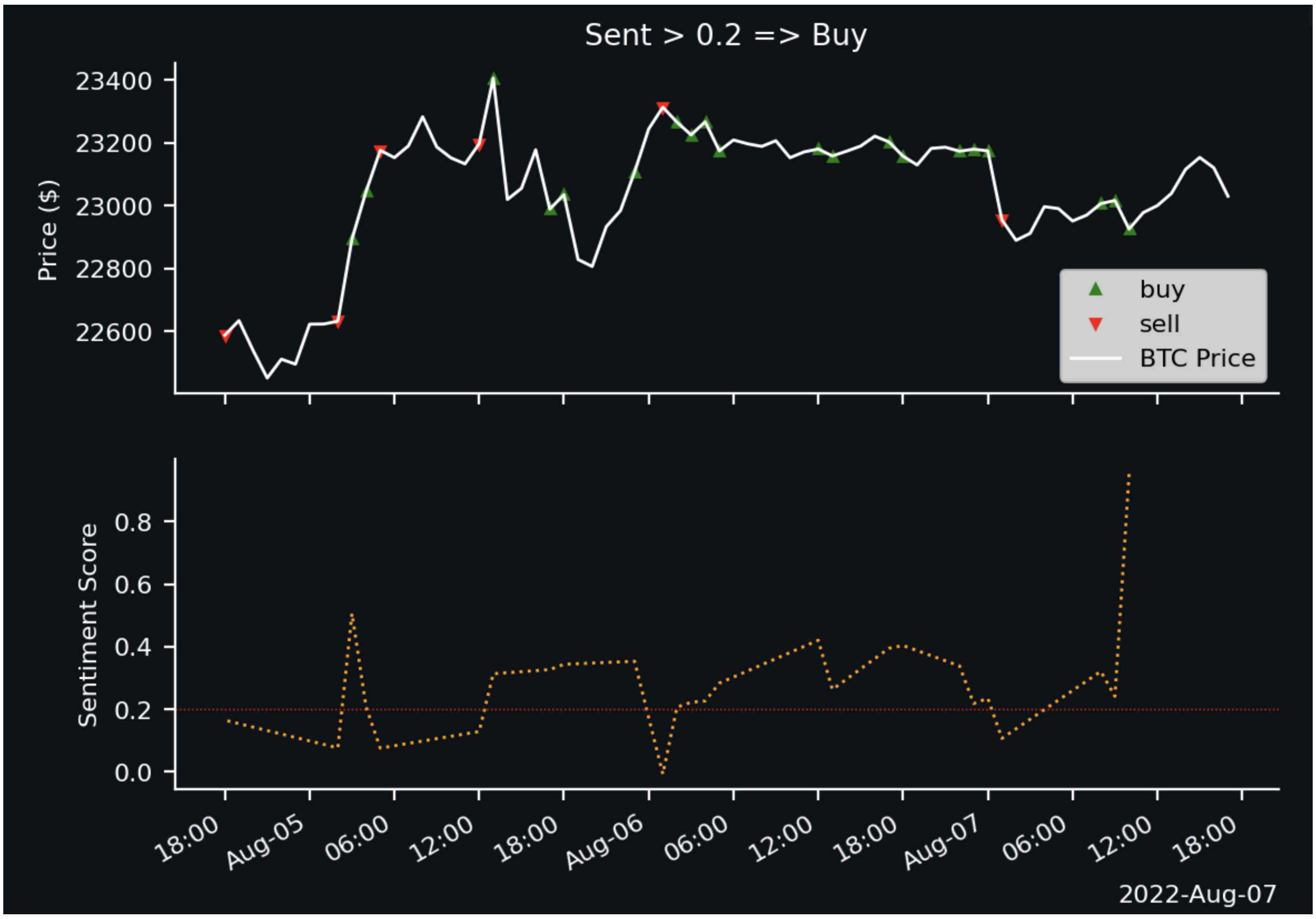


Figure 15: Charts for Bitcoin Price and Average Sentiment with corresponding signals

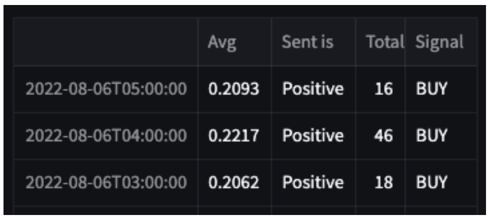
The trade-file starts with the essential function to collect all the single tweets from the database and stores them in a pandas DataFrame. This is needed to delete the duplicates with the following methods:

```
duplicates = list(
    df.index[
        df.duplicated(
            subset=["Tweet"], keep=False
        )
    ]
)

df.drop_duplicates(
    subset=["Tweet"], keep=False, inplace=True
    )
```

Afterwards, the neutral tweets are filtered and the rest are counted and the mean/avg is calculated for a resampled interval of 1h. This means all the timestamps from 1h (p.e. from 16:00 - 17:00) are grouped and a function is applied to all of them. This function could be to summarize, count or calculate the mean/avg.

```
df = df[df["Sentiment Score"] != 0.0]

count_tweets = df.resample(f"1H").count()

mean_df = df.resample(f"1H").mean().sort_index(ascending=False)
```

These two series get concatenated together to have one table:



	Avg	Sent is	Total	Signal
2022-08-06T05:00:00	0.2093	Positive	16	BUY
2022-08-06T04:00:00	0.2217	Positive	46	BUY
2022-08-06T03:00:00	0.2062	Positive	18	BUY

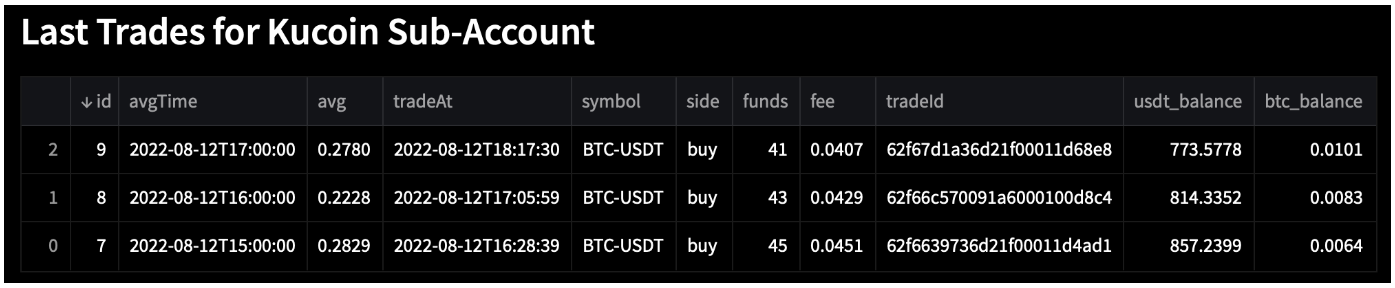
Figure 16: Last three time periods with average, total tweets and signal

And these last three timestamps are important for the trading. The first row is showing the actual timestamp. So at the time this picture was taken it was somewhere between 05:00 and 06:00. Since the tweets are collected live and every few seconds this row will be updated frequently and the avg and total tweets will change.

This is why the second row will be selected as the trading decision. Firstly, it will be checked if a trade was already made for this timestamp. Then, if the signal indicates buy, a market order will be created with 5% of the available USDT Balance (after a check, if there are any funds available).

A sell order will sell a quarter of the current bitcoin holdings.

At last, some metrics from the trade are uploaded into a separate table on Heroku for a better overview of the trades and later calculation of current PNL (Profit and Loss). An excerpt from this table is shown below in figure 17. It contains the sentiment average and the time period, as well as information about the trade itself. The last balances are actually the balances after the trade was made. So, it is possible, to look back at all the balances without gaps.



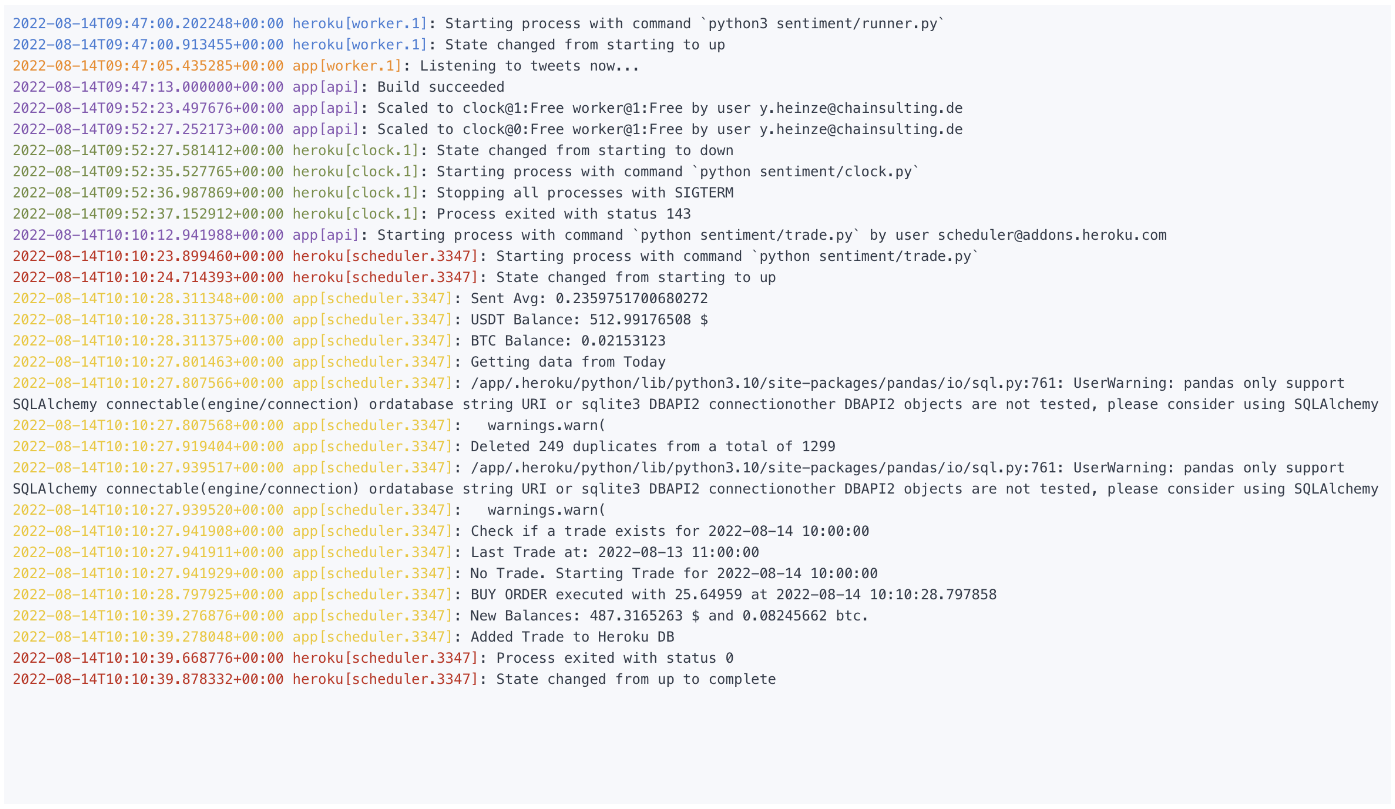
Last Trades for Kucoin Sub-Account

	↓id	avgTime	avg	tradeAt	symbol	side	funds	fee	tradeId	usdt_balance	btc_balance
2	9	2022-08-12T17:00:00	0.2780	2022-08-12T18:17:30	BTC-USDT	buy	41	0.0407	62f67d1a36d21f00011d68e8	773.5778	0.0101
1	8	2022-08-12T16:00:00	0.2228	2022-08-12T17:05:59	BTC-USDT	buy	43	0.0429	62f66c570091a6000100d8c4	814.3352	0.0083
0	7	2022-08-12T15:00:00	0.2829	2022-08-12T16:28:39	BTC-USDT	buy	45	0.0451	62f6639736d21f00011d4ad1	857.2399	0.0064

Figure 17: Last Trades with Kucoin Sub-Account

Looking at the trades

Figure 18 shows the Heroku Logs. At first, the runner is started, that listens to the tweets and adds them to the database. When you see the line `Listening to tweets now...`, you know, everything is working properly.



```
2022-08-14T09:47:00.202248+00:00 heroku[worker.1]: Starting process with command `python3 sentiment/runner.py`
2022-08-14T09:47:00.913455+00:00 heroku[worker.1]: State changed from starting to up
2022-08-14T09:47:05.435285+00:00 app[worker.1]: Listening to tweets now...
2022-08-14T09:47:13.000000+00:00 app[api]: Build succeeded
2022-08-14T09:52:23.497676+00:00 app[api]: Scaled to clock@1:Free worker@1:Free by user y.heinze@chainsulting.de
2022-08-14T09:52:27.252173+00:00 app[api]: Scaled to clock@0:Free worker@1:Free by user y.heinze@chainsulting.de
2022-08-14T09:52:27.581412+00:00 heroku[clock.1]: State changed from starting to down
2022-08-14T09:52:35.527765+00:00 heroku[clock.1]: Starting process with command `python sentiment/clock.py`
2022-08-14T09:52:36.987869+00:00 heroku[clock.1]: Stopping all processes with SIGTERM
2022-08-14T09:52:37.152912+00:00 heroku[clock.1]: Process exited with status 143
2022-08-14T10:10:12.941988+00:00 app[api]: Starting process with command `python sentiment/trade.py` by user scheduler@addons.heroku.com
2022-08-14T10:10:23.899460+00:00 heroku[scheduler.3347]: Starting process with command `python sentiment/trade.py`
2022-08-14T10:10:24.714393+00:00 heroku[scheduler.3347]: State changed from starting to up
2022-08-14T10:10:28.311348+00:00 app[scheduler.3347]: Sent Avg: 0.2359751700680272
2022-08-14T10:10:28.311375+00:00 app[scheduler.3347]: USDT Balance: 512.99176508 $
2022-08-14T10:10:28.311375+00:00 app[scheduler.3347]: BTC Balance: 0.02153123
2022-08-14T10:10:27.801463+00:00 app[scheduler.3347]: Getting data from Today
2022-08-14T10:10:27.807566+00:00 app[scheduler.3347]: /app/.heroku/python/lib/python3.10/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support
SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
2022-08-14T10:10:27.807568+00:00 app[scheduler.3347]:   warnings.warn(
2022-08-14T10:10:27.919404+00:00 app[scheduler.3347]: Deleted 249 duplicates from a total of 1299
2022-08-14T10:10:27.939517+00:00 app[scheduler.3347]: /app/.heroku/python/lib/python3.10/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support
SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
2022-08-14T10:10:27.939520+00:00 app[scheduler.3347]:   warnings.warn(
2022-08-14T10:10:27.941908+00:00 app[scheduler.3347]: Check if a trade exists for 2022-08-14 10:00:00
2022-08-14T10:10:27.941911+00:00 app[scheduler.3347]: Last Trade at: 2022-08-13 11:00:00
2022-08-14T10:10:27.941929+00:00 app[scheduler.3347]: No Trade. Starting Trade for 2022-08-14 10:00:00
2022-08-14T10:10:28.797925+00:00 app[scheduler.3347]: BUY ORDER executed with 25.64959 at 2022-08-14 10:10:28.797858
2022-08-14T10:10:39.276876+00:00 app[scheduler.3347]: New Balances: 487.3165263 $ and 0.08245662 btc.
2022-08-14T10:10:39.278048+00:00 app[scheduler.3347]: Added Trade to Heroku DB
2022-08-14T10:10:39.668776+00:00 heroku[scheduler.3347]: Process exited with status 0
2022-08-14T10:10:39.878332+00:00 heroku[scheduler.3347]: State changed from up to complete
```

Figure 18: Logs from Heroku on scheduled trades

As explained in the Backend-Section, the scheduler is executing the trading script every hour.

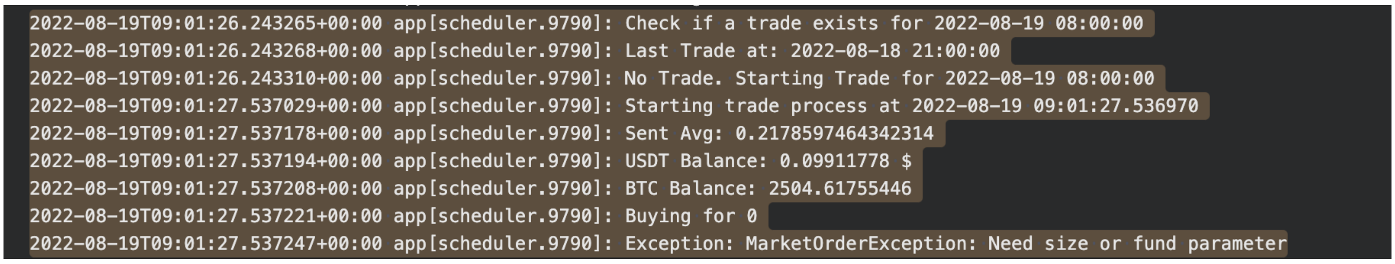
The sentiment average is shown and the Balances before and after the trade. It also says if a trade already exists and then either starts a new trade or does nothing.

The Heroku logs were an essential part of the development-phase, since they allowed for a good way of debugging. Usually, no `print`-statements are left in the final code, when it's entering the production phase. However, Heroku seems to only print the `print`-statements rather than from the logging-library, which is otherwise used. Therefore, a few print statements are left in the final code.

Challenges

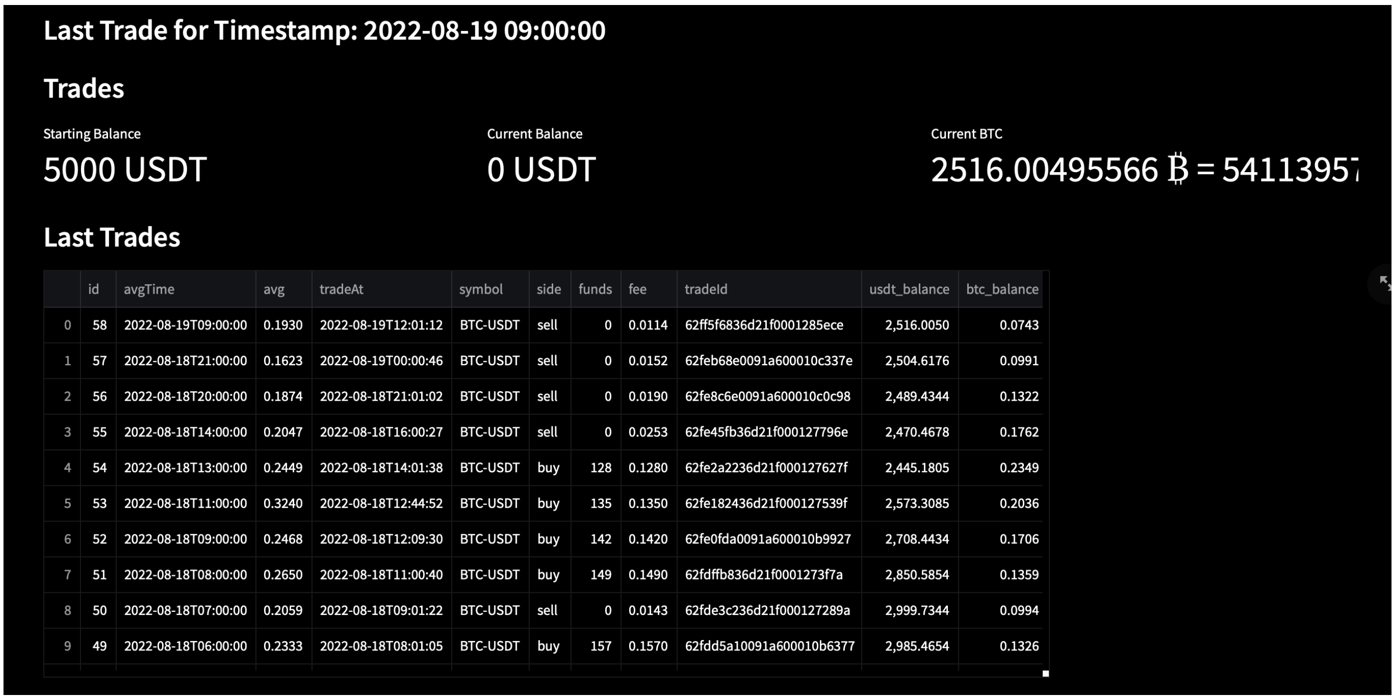
Getting false Account Balance

The System was trading fine for a couple of days, but then, all of a sudden, it did not execute any trades for a couple timestamps. As can be seen in figure 19 below, the USDT and BTC Balance are switched, and the system tried to buy for an amount of *0* USDT, which did not work. Interestingly, in figure 20, you can see that this was not a consistent state since there are some trades at random timestamps. The problem lay with Kucoin. When acquiring the current account balances, USDT normally came before BTC. However, this order seems to be switched at random. This unforeseen coincidence needed to be checked by the system and be acted upon.



```
2022-08-19T09:01:26.243265+00:00 app[scheduler.9790]: Check if a trade exists for 2022-08-19 08:00:00
2022-08-19T09:01:26.243268+00:00 app[scheduler.9790]: Last Trade at: 2022-08-18 21:00:00
2022-08-19T09:01:26.243310+00:00 app[scheduler.9790]: No Trade. Starting Trade for 2022-08-19 08:00:00
2022-08-19T09:01:27.537029+00:00 app[scheduler.9790]: Starting trade process at 2022-08-19 09:01:27.536970
2022-08-19T09:01:27.537178+00:00 app[scheduler.9790]: Sent Avg: 0.2178597464342314
2022-08-19T09:01:27.537194+00:00 app[scheduler.9790]: USDT Balance: 0.09911778 $
2022-08-19T09:01:27.537208+00:00 app[scheduler.9790]: BTC Balance: 2504.61755446
2022-08-19T09:01:27.537221+00:00 app[scheduler.9790]: Buying for 0
2022-08-19T09:01:27.537247+00:00 app[scheduler.9790]: Exception: MarketOrderException: Need size or fund parameter
```

Figure 19: Logs from Heroku: No Trade was executed because the size or fund parameter was false



Last Trade for Timestamp: 2022-08-19 09:00:00

Trades

Starting Balance
5000 USDT

Current Balance
0 USDT

Current BTC
2516.00495566 ₿ = 54113957

Last Trades

	id	avgTime	avg	tradeAt	symbol	side	funds	fee	tradeId	usdt_balance	btc_balance
0	58	2022-08-19T09:00:00	0.1930	2022-08-19T12:01:12	BTC-USDT	sell	0	0.0114	62ff5f6836d21f0001285ece	2,516.0050	0.0743
1	57	2022-08-18T21:00:00	0.1623	2022-08-19T00:00:46	BTC-USDT	sell	0	0.0152	62feb68e0091a600010c337e	2,504.6176	0.0991
2	56	2022-08-18T20:00:00	0.1874	2022-08-18T21:01:02	BTC-USDT	sell	0	0.0190	62fe8c6e0091a600010c0c98	2,489.4344	0.1322
3	55	2022-08-18T14:00:00	0.2047	2022-08-18T16:00:27	BTC-USDT	sell	0	0.0253	62fe45fb36d21f000127796e	2,470.4678	0.1762
4	54	2022-08-18T13:00:00	0.2449	2022-08-18T14:01:38	BTC-USDT	buy	128	0.1280	62fe2a2236d21f000127627f	2,445.1805	0.2349
5	53	2022-08-18T11:00:00	0.3240	2022-08-18T12:44:52	BTC-USDT	buy	135	0.1350	62fe182436d21f000127539f	2,573.3085	0.2036
6	52	2022-08-18T09:00:00	0.2468	2022-08-18T12:09:30	BTC-USDT	buy	142	0.1420	62fe0fda0091a600010b9927	2,708.4434	0.1706
7	51	2022-08-18T08:00:00	0.2650	2022-08-18T11:00:40	BTC-USDT	buy	149	0.1490	62fdffb836d21f0001273f7a	2,850.5854	0.1359
8	50	2022-08-18T07:00:00	0.2059	2022-08-18T09:01:22	BTC-USDT	sell	0	0.0143	62fde3c236d21f000127289a	2,999.7344	0.0994
9	49	2022-08-18T06:00:00	0.2333	2022-08-18T08:01:05	BTC-USDT	buy	157	0.1570	62fdd5a10091a600010b6377	2,985.4654	0.1326

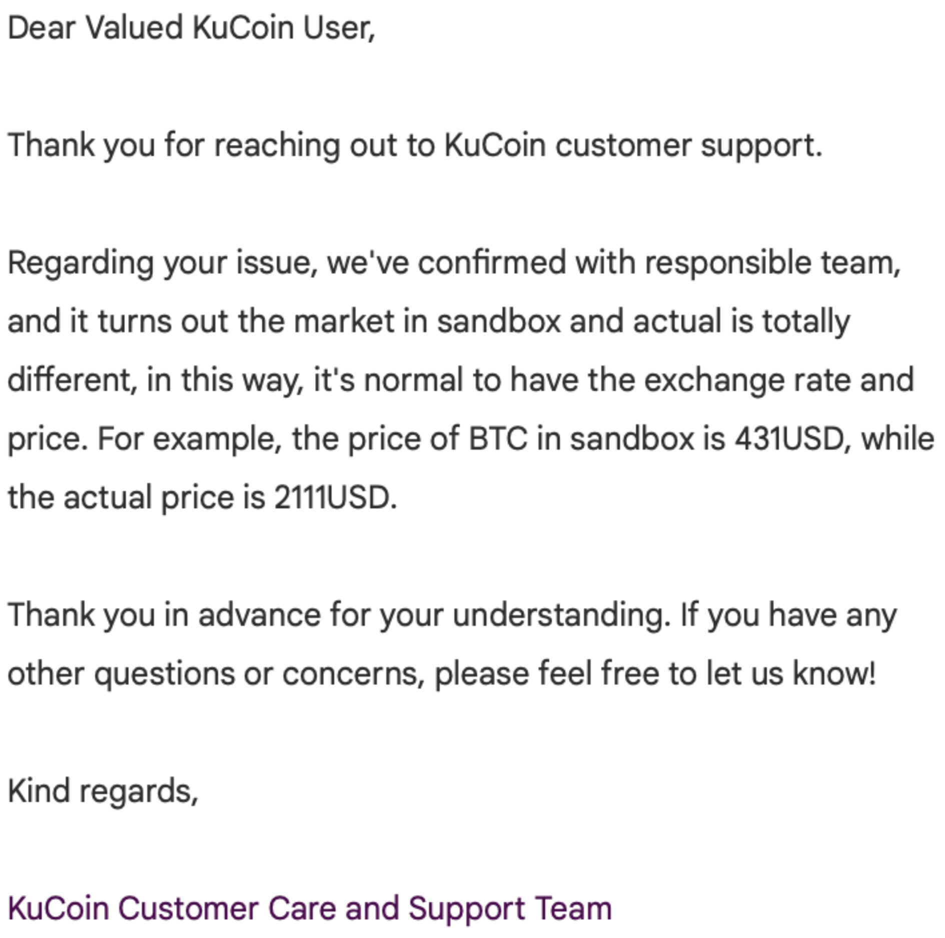
Figure 20: Wrong Account Balance on Kucoin

Different Sandbox Prices

The biggest challenge came with the sandbox. All the trading worked perfectly, but it was weird to see that the first trade of 250 $ equalled a BTC amount of 0.06, which would be more than 1300 $. This didn't make any sense.

Since these different prices were not prone to bitcoin alone, but all coins had different prices in the sandbox, the Kucoin Support was contacted.

The answer (figure 21), confirmed that the prices in the sandbox are just different, and another solution was needed.



Dear Valued KuCoin User,

Thank you for reaching out to KuCoin customer support.

Regarding your issue, we've confirmed with responsible team, and it turns out the market in sandbox and actual is totally different, in this way, it's normal to have the exchange rate and price. For example, the price of BTC in sandbox is 431USD, while the actual price is 2111USD.

Thank you in advance for your understanding. If you have any other questions or concerns, please feel free to let us know!

Kind regards,

KuCoin Customer Care and Support Team

Figure 21: Email from Kucoin regarding the different Bictoin Prices

The solution was to get the correct price from somewhere else and in this case: Binance.

To convert the current BTC Holdings a real-time price was needed.

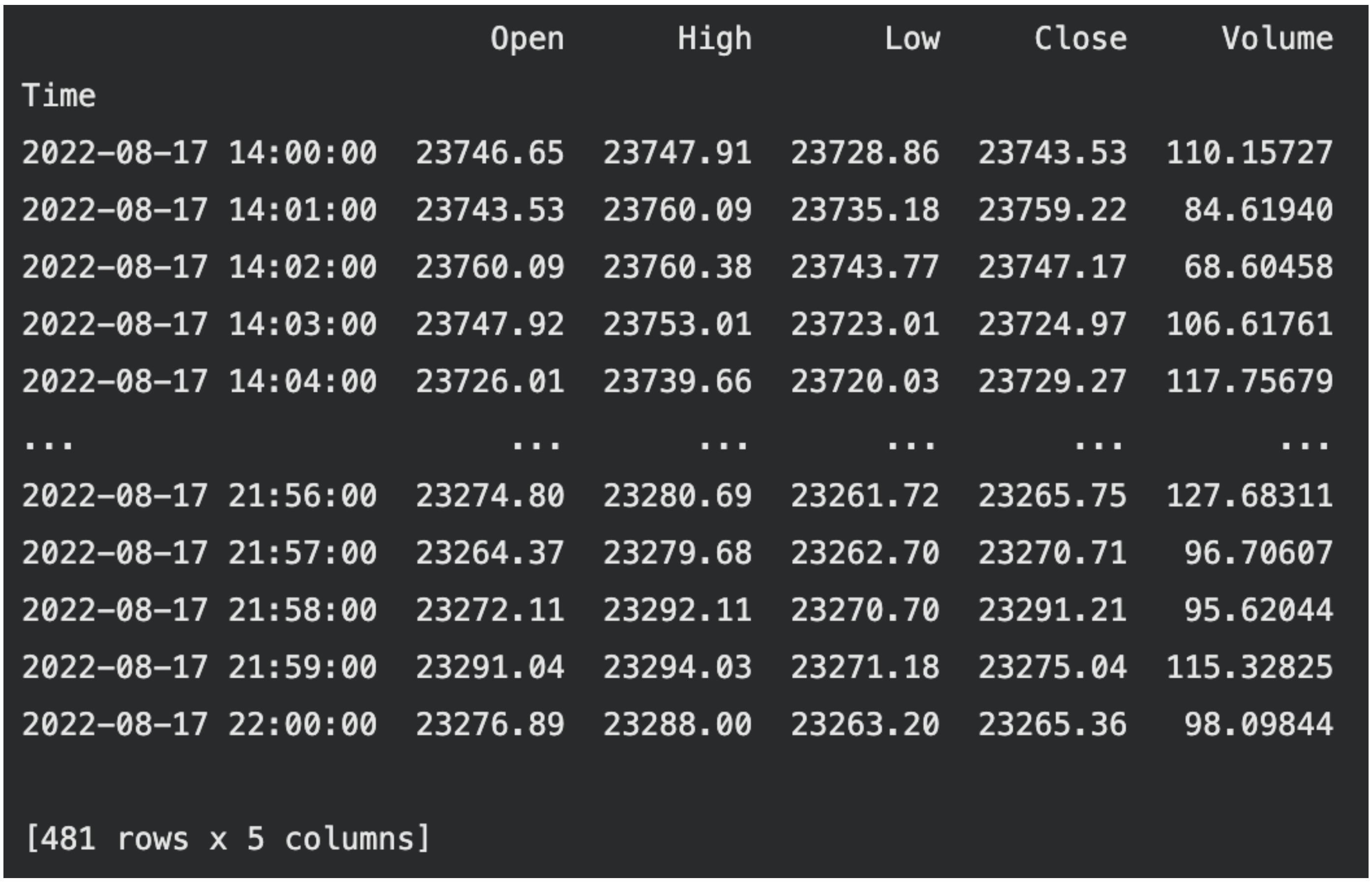
This is done with a few lines of code and a typical HTML-Request:

```
url = "https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT"
data = requests.get(url)
data = data.json()
```

To get all the historical data from Binance, the python-binance wrapper was used:

```
frame = pd.DataFrame(binance_client.get_historical_klines(symbol,1,lookback))
frame = frame.iloc[:,:6]
frame.columns= ["Time","Open","High","Low","Close","Volume"]
frame = frame.set_index("Time")
frame.index = pd.to_datetime(frame.index,unit="ms")
frame.index = frame.index + timedelta(hours=2) #utc to local
frame
```

The result looked like this:



```
                         Open      High       Low     Close     Volume
Time
2022-08-17 14:00:00  23746.65  23747.91  23728.86  23743.53  110.15727
2022-08-17 14:01:00  23743.53  23760.09  23735.18  23759.22   84.61940
2022-08-17 14:02:00  23760.09  23760.38  23743.77  23747.17   68.60458
2022-08-17 14:03:00  23747.92  23753.01  23723.01  23724.97  106.61761
2022-08-17 14:04:00  23726.01  23739.66  23720.03  23729.27  117.75679
...                       ...       ...       ...       ...        ...
2022-08-17 21:56:00  23274.80  23280.69  23261.72  23265.75  127.68311
2022-08-17 21:57:00  23264.37  23279.68  23262.70  23270.71   96.70607
2022-08-17 21:58:00  23272.11  23292.11  23270.70  23291.21   95.62044
2022-08-17 21:59:00  23291.04  23294.03  23271.18  23275.04  115.32825
2022-08-17 22:00:00  23276.89  23288.00  23263.20  23265.36   98.09844

[481 rows x 5 columns]
```

Figure 22: Prices (ohlc) and Volume for Bitcoin from Binance

The method `get_historical_klines()` accepts arguments to look for intervals in a past time period. For example, in figure 22 we are looking for the past day ("1d") with intervals of one minute ("1m").

The timestamp and the closing price were then used for further calculations and visualisations.