Comparing Natural Language Processing and trends based algorithmic trading strategies with a basic mean reversion strategy.

Note from the author:

These algorithms have not been tested with real money, nor backtested (because there is not such data for the first 2 algorithms) and do not take in count fees and liquidity issues. The first bot has been running on a free server (PythonAnywhere) during several days. This service provider has terminated several times the bot (because it took too many CPU) and the bot could run at full speed only 100 seconds each 24 hours (because of CPU management). The 2 other algorithms have been tested on Google Colab. All 3 programs can be executed at a specific time on a server on a daily basis, for instance on PythonAnywhere, for free (create 3 accounts). The tests have been execute on a really small amount of days, but it can be done once again on a larger scale quite easily, especially with a good server.

1. NLP based algorithmic trading strategy: trading cryptocurrencies with Twitter.

A) Idea

Let's imagine a class of 100 students. Let's imagine the professor asks the class if they like Tesla at 10:00. 60% will say yes. Then the professor asks again at 10:30. 70% say yes. That means that there is an increase in the sentiment about Tesla. Now he asks at 11:00 and only 20% say yes, that's a decrease.

This trading bot says: increase in sentiment → Buy, decrease → Sell.

NLP stands for Natural Language Processing. In Python there is some good libraries that provide NLP, like TextBlob or Vader. I will use TextBlob.

The idea is to select 1 crypto (I chose Bitcoin, but it can be almost anything, details in 1.C), to identify its underlying sentiment, by using sentiment analysis with NLP. Then to check once again the sentiments trends. If it increases, we buy X shares / coins. Once the position is open, we check again the sentiments movements. If it decreases, we close the position.

B) Code

```python
import tweepy
import time
from textblob import TextBlob
import yfinance as yf

########################################################################
ckey = ""
csecret = ""
atoken = ""
atsecret = ""
nb = 500 # Choose how many tweets you want the bot to analyze
keywords = ["BTC", "#BTC", "Bitcoin"] # Choose keywords related to the asset
you want to trade
########################################################################


auth = tweepy.OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, atsecret)
api2 = tweepy.API(auth, wait_on_rate_limit=True,
wait_on_rate_limit_notify=True)


def perc(a,b):
    temp = 100 * float(a) / float(b)
    return format(temp, '.2f')
```

```python
def get_current_price(symbol):
    ticker = yf.Ticker(symbol)
    todays_data = ticker.history(period='1d')
    return todays_data['Close'][0]

def get_twitter_BTC():
    ratios = 0

    for keyword in keywords:
        tweets = tweepy.Cursor(api2.search, q=keyword, lang="en").items(nb)

        pos = 0
        neg = 0

        for tweet in tweets:
            analysis = TextBlob(tweet.text)

            if 0 <= analysis.sentiment.polarity <= 1:
                pos += 1
            elif -1 <= analysis.sentiment.polarity < 0:
                neg += 1

        pos = perc(pos, nb)
        neg = perc(neg, nb)

        if float(neg) > 0:
            ratio = float(pos) / float(neg)
        else:
            ratio = float(pos)
        ratios += ratio

    return ratios


if __name__ == "__main__":


    for k in range(1000):
        score = get_twitter_BTC()
        min1 = score + (score * 30 / 100)
        time.sleep(60*5)
        new_score = get_twitter_BTC()

        if new_score > min1:
            btc_price = get_current_price("BTC-USD")
            buy = "\nBUY : " + str(btc_price)
            f = open("output.txt", "a")
            f.write(buy)
            f.close()


            time.sleep(60*5)
            new_new_score = get_twitter_BTC()
```

```
73            min2 = new_score - (new_score * 30 / 100)
74
75            if new_new_score < min2:
76                new_btc_price = get_current_price("BTC-USD")
77                sell_at = " SELL : " + str(new_btc_price)
78                trade_profit = new_btc_price - btc_price
79                perc_profit = trade_profit / btc_price * 100
                  perc_profit_round = round(perc_profit, 3)
80                sell_message = sell_at + " | " + " Profit = " +
81 str(perc_profit_round) + " %"
82                f = open("output.txt", "a")
83                f.write(sell_message)
84                f.close()
85                time.sleep(60*5)
86
87        else:
              time.sleep(60*5)
```

Comment:

L20-22: getting percentages

L24-27: getting current price of a ticker

L28-52: returns a number that corresponds to the current sentiment concerning all the keywords entered before.

L58-87: first check the current sentiment, sleeps, check once again. If new sentiment > previous one: buy. Sleep, check once again, if new sentiment < previous one: sell.

C) Pros and cons

What I like about that algorithm is that we can modulate it how we want. We can choose how "strict" the conditions must be to open or close a position. The "strictness" corresponds to how much the new sentiment analysis must be different from the previous one. For instance, we can say that a 5% increase is not an increase, and so we don't buy anything.

The main problem is clearly related to NLP. How NLP deals with noise, ironic or sarcastic ect. content? I have no idea, so I am quite pessimistic with that approach. Also, this approach must be proven by a correlation between prices and sentiment. Which brings me to the last part.

D) Areas for improvement

One could try to get a large dataset of sentiments upon an asset within a fixed time scope. Once it is done, one could compare it with the market price of that asset, and check for correlation. What would be really interesting is that if the sentiments trends **precede** price movements. If not, most of the profit and losses from this bot were random. I wasn't able to do this because there is query rate limit which doesn't allow me to get the data each second.

To make the bot fully automated, one must take in count the fact that when the sentiment only increases, positions are held open, so assets are stacked. So, one must find the right moment to close the remaining positions. One moment could be to close when the price hits the higher position held + 0.1 % for instance.

2. Trends based algorithmic trading strategy: trading stocks with Reddit

A) Idea

Here, we will get the 20 stocks that are the most spoken about in Reddit. Once it is done, we will buy 1000 shares of them, and then close all the positions once the balance hits about 0.1% more from the previous one. The first part of the code is from asad70 on github, who wanted to use Vader for sentiment analysis. I used it and saw a lack of correlation, which makes no sense to buy only those with good compound scores.

B) Code

The part from line 9 to line 41 is taken from Asad70's Github (https://github.com/asad70/wallstreetbets-sentiment-analysis/blob/master/Wallstreetbets.py), this part gives me the 20 most mentioned stocks from Reddit. I modified it and replaced his sentiment analysis part (which I consider useless) with a direct connection to Alpaca's API.

```
   import praw
1  import alpaca_trade_api as tradeapi
   import time
2  import pandas as pd

3  APCA_API_KEY_ID = ""
   APCA_API_SECRET_KEY = ""
4  APCA_API_BASE_URL = "https://paper-api.alpaca.markets"
   api = tradeapi.REST(APCA_API_KEY_ID, APCA_API_SECRET_KEY,
5  APCA_API_BASE_URL, api_version='v2')
   account = api.get_account()
6

7  reddit = praw.Reddit(client_id='6cq9yDMVtyRcLA',
                        client_secret='cpwIv0_xvj-3rXILTQeU0KjNwh-7ew',
8                       user_agent='ksjdhsaq',
                        check_for_async = False)

   us = {this list contains about 10 pages of us stocks}

   # includes common words and words used on wsb that are also stock
   names
   blacklist = {This list contains common words not to mess with stocks
   names}

   '''######################################################################
   #########'''
   # set the program parameters
   subs = ['wallstreetbets', 'stocks', 'investing', 'stockmarket']      #
   sub-reddit to search
   post_flairs = {'Daily Discussion', 'Weekend Discussion', 'Discussion'}
   # posts flairs to search || None flair is automatically considered
   goodAuth = {'AutoModerator'}    # authors whom comments are allowed
   more than once
   uniqueCmt = True                # allow one comment per author per
   symbol
   ignoreAuthP = {'example'}       # authors to ignore for posts
   ignoreAuthC = {'example'}       # authors to ignore for comment
   upvoteRatio = 0.70         # upvote ratio for post to be considered,
   0.70 = 70%
   ups = 20       # define # of upvotes, post is considered if upvotes
   exceed this #
   limit = 10      # define the limit, comments 'replace more' limit
   upvotes = 2     # define # of upvotes, comment is considered if
   upvotes exceed this #
   picks = 20     # define # of picks here, prints as "Top ## picks are:"
   '''######################################################################
   #########'''

   posts, count, c_analyzed, tickers, titles, a_comments = 0, 0, 0, {},
   [], {}
   cmt_auth = {}

   for sub in subs:
       subreddit = reddit.subreddit(sub)
       hot_python = subreddit.hot()     # sorting posts by hot
       # Extracting comments, symbols from subreddit
       for submission in hot_python:
           flair = submission.link_flair_text
           author = submission.author.name

           # checking: post upvote ratio # of upvotes, post flair, and
   author
```

```python
        if submission.upvote_ratio >= upvoteRatio and submission.ups > ups and (flair in post_flairs or flair is None) and author not in ignoreAuthP:
            submission.comment_sort = 'new'
            comments = submission.comments
            titles.append(submission.title)
            posts += 1
            submission.comments.replace_more(limit=limit)
            for comment in comments:
                # try except for deleted account?
                try: auth = comment.author.name
                except: pass
                c_analyzed += 1

                # checking: comment upvotes and author
                if comment.score > upvotes and auth not in ignoreAuthC:
                    split = comment.body.split(" ")
                    for word in split:
                        word = word.replace("$", "")
                        # upper = ticker, length of ticker <= 5, excluded words,
                        if word.isupper() and len(word) <= 5 and word not in blacklist and word in us:

                            # unique comments, try/except for key errors
                            if uniqueCmt and auth not in goodAuth:
                                try:
                                    if auth in cmt_auth[word]: break
                                except: pass

                            # counting tickers
                            if word in tickers:
                                tickers[word] += 1
                                a_comments[word].append(comment.body)
                                cmt_auth[word].append(auth)
                                count += 1
                            else:
                                tickers[word] = 1
                                cmt_auth[word] = [auth]
                                a_comments[word] = [comment.body]
                                count += 1

# sorts the dictionary
symbols = dict(sorted(tickers.items(), key=lambda item: item[1], reverse = True))
top_picks = list(symbols.keys())[0:picks]
```

```
times = []
top = []
for i in top_picks:
    times.append(symbols[i])
    top.append(i)

print("Stocks loaded successfully")

for t in top:
  try:
    api.submit_order(
      symbol= t,
      qty=1000,
      side='buy',
      time_in_force='gtc',
      type='market')
  except:
    print(t, "can't buy / or sell")
    continue

print("Positions opened successfully")

while(True):
  time.sleep(1)
  account = api.get_account()
  min = float(account.last_equity) + float(account.last_equity) *
0.1/100
  if (float(account.equity) > min):
    for t in top:
      try:
        api.submit_order(
          symbol= t,
          qty=1000,
          side='sell',
          time_in_force='gtc',
          type='market')
      except:
        print(t, "can't close position")
        continue
    break

print("Positions closed successfully")
account = api.get_account()
print("Profit : $",(float(account.equity) -
float(account.last_equity)))
```

Comment:

The initial algorithm from Asad70 outputs something like:

```
10 most mentioned picks:
PLTR: 54
TSLA: 33
AAPL: 31
TLRY: 23
GME: 23
AMD: 22
NOK: 19
RKT: 19
NET: 17
MSFT: 16

Sentiment analysis of top 5 picks:
     Bearish Neutral Bullish Total/Compound
PLTR   0.061   0.790   0.150          0.269
TSLA   0.123   0.752   0.125          0.001
AAPL   0.098   0.764   0.138          0.176
TLRY   0.102   0.824   0.074          0.057
GME    0.078   0.804   0.118          0.195
```

As I told, I chose not to use the sentiment analysis part, but only to pick the most mentioned US stocks.

L43-47: Buys all the top stocks (1000 shares, GTC "Good till canceled")

L48-53: Checks every second if the balance has increased by 0.1%. If yes : close the positions

L54: prints the profit

C) Pros and cons

Which is important with that strategy is to understand how correlated to the trend it is. Actually, sometimes Reddit **is** the trend. So, one must switch from buy-then-sell to sell-then-buy according to the current trend. That's why one must first check with both directions (buy-then-sell then sell-then-buy) to invest more money on the most profitable direction. Beyond all that, that strategy is simple, and seems to work quite well.

D) Areas for improvement

To nuance what I just said, sometimes one could do buy-then-sell and then sell-then-buy, and for both, lose money (getting for both sides a negative profit). It happened sometimes to me. For this I don't have ideas, but I think that one would need to create a program which tells the stocks that are making us losing money (whether we are in short or long positions). Once we get them, we just launch the script once again making sure the "bad" stocks are not traded the same way that others.

3. Mean reversion algorithmic strategy: Trading stocks with the Simple Moving Average

A) Idea

This strategy is widely spread and often opposed to the momentum strategy. Mean reversion means that every asset has an average price. If the current market price is lower than the current average, then it should increase. On the other hand, if it is higher than the average, then it should decrease. So, when it should increase, we buy, and when it should decrease, we sell. I made the following algorithm simple in order to compare with the first 2 bots.

B) Code

Note that I chose to trade 10 financial stocks during/post-Covid crisis.

```
import pytz
import datetime
import alpaca_trade_api as tradeapi
import investpy

APCA_API_KEY_ID = ""
APCA_API_SECRET_KEY = ""
APCA_API_BASE_URL = "https://paper-api.alpaca.markets"
api = tradeapi.REST(APCA_API_KEY_ID, APCA_API_SECRET_KEY,
APCA_API_BASE_URL, api_version='v2')
account = api.get_account()

def get_price(stock):
  full_stock_price = api.get_barset(stock, '1Min', limit=1)
  close_price = full_stock_price.df.iat[0,3]
  return close_price

mean_r_universe=["JPM","HSBC","C","WFC","GS","MS","PNC","USB","SCHW","
BCS"]
portfolio = api.list_positions()
open_positions = []

for position in portfolio:
  open_positions.append(position.symbol)

for i in mean_r_universe:
  utc = datetime.datetime.now(pytz.timezone("UTC"))
  try:
    data = investpy.moving_averages(name=i, country='United States',
product_type='stock', interval='daily')
```

```
1   except:
6     print("Error with :", i)
1     continue
7   sma = data.iloc[0, 1]
1   price = get_price(i)
8   if price < sma :
1     if i not in open_positions:
9       print("BUY", i, "at", price, utc)
2       try:
0         api.submit_order(
2           symbol= i,
1           qty=1000,
2           side='buy',
2           time_in_force='gtc',
2           type='market')
3       except:
2         print("Can't buy", i)
4         continue
2   else:
5     print("SELL", i, "at", price, utc)
2     try:
6       api.submit_order(
2         symbol= i,
7         qty=1000,
2         side='sell',
8         time_in_force='gtc',
2         type='market')
9     except:
3       print("Can't sell", i)
0       continue
```
Comment:

L7-9 : returns the current price of a stock

L11-13: stacking current opened positions (tickers)

L14-30: getting SMA value and market price value for each stock. If SMA > price and position not opened yet : buy. Else if SMA < price : sell.

Investpy's moving averages function looks like this :

```
  period  sma_value sma_signal  ema_value ema_signal
0      5      5.279        buy      5.274        buy
1     10      5.211        buy      5.223        buy
2     20      5.120        buy      5.097        buy
3     50      4.767        buy      4.867        buy
4    100      4.584        buy      4.529        buy
5    200      3.856        buy      4.225        buy
```

C) Pros and cons

This strategy is extremely simple. However, we must adapt our strategy between momentum and mean reversion. I have not done it, but this strategy can be backtested, because it relies on stock prices and movements only. I consider it more of a long-term strategy and would not advise to use it within a market relying on trends.

D) Areas for improvement

In my opinion, one could use all the SMAs (5 days, 10 days...) combined with the EMAs (Exponential Moving Averages) to take decisions. One could also combine all that with other technical indicators (a lot of libraries provide some, like TA-lib)

4. Comparing results

1st Algorithm)

First a decided to buy when there was 40% of sentiment increase:

```
BUY : 34527.50390625 SELL : 34934.4375 |  Profit = 1.179 %
BUY : 34705.78125
BUY : 35875.6328125 SELL : 35901.76953125 |  Profit = 0.073 %
BUY : 35814.38671875 SELL : 35834.828125 |  Profit = 0.057 %
BUY : 35404.38671875 SELL : 35323.16796875 |  Profit = -0.229 %
BUY : 35450.9140625
```

40% is quite strict, so I decided to lower it to 30%:

```
BUY : 36145.83984375 SELL : 36124.16796875 |  Profit = -0.06 %
BUY : 36213.40625 SELL : 36168.55859375 |  Profit = -0.124 %
BUY : 36256.453125
BUY : 36699.03515625 SELL : 36729.83984375 |  Profit = 0.084 %
BUY : 36773.92578125
BUY : 36675.89453125 SELL : 36688.19921875 |  Profit = 0.034 %
BUY : 36170.921875 SELL : 36191.08984375 |  Profit = 0.056 %
BUY : 36364.03515625 SELL : 36417.52734375 |  Profit = 0.147 %
BUY : 36865.4296875 SELL : 36680.76171875 |  Profit = -0.501 %
BUY : 36423.2421875 SELL : 36358.07421875 |  Profit = -0.179 %
BUY : 36115.24609375
BUY : 36187.515625
BUY : 36492.41796875
BUY : 36789.21875 SELL : 36842.26953125 |  Profit = 0.144 %
BUY : 36884.6484375 SELL : 37128.74609375 |  Profit = 0.662 %
BUY : 37318.54296875
BUY : 37270.625 SELL : 37278.3046875 |  Profit = 0.021 %
```

```
18 BUY : 38088.01953125 SELL : 38038.94921875 |  Profit = -0.129 %
19 BUY : 37287.16796875 SELL : 37261.2109375 |  Profit = -0.07 %
20 BUY : 38840.984375
21 BUY : 38673.09375 SELL : 38657.91015625 |  Profit = -0.039 %
22 BUY : 38840.0859375 SELL : 38811.48046875 |  Profit = -0.074 %
23 BUY : 38527.58984375 SELL : 38453.19921875 |  Profit = -0.193 %
24 BUY : 37744.546875
25 BUY : 36894.6015625 SELL : 36772.21875 |  Profit = -0.332 %
26 BUY : 35931.7734375 SELL : 35859.265625 |  Profit = -0.202 %
27 BUY : 36071.19921875
28 BUY : 36106.11328125 SELL : 35970.46484375 |  Profit = -0.376 %
29 BUY : 36142.52734375 SELL : 36132.40625 |  Profit = -0.028 %
30 BUY : 35119.19140625
31 BUY : 35625.203125 SELL : 35562.3359375 |  Profit = -0.176 %
32 BUY : 36173.3984375 SELL : 36164.9375 |  Profit = -0.023 %
33 BUY : 36123.421875
34 BUY : 36123.828125 SELL : 36191.2109375 |  Profit = 0.187 %
35 BUY : 35903.81640625 SELL : 35875.41796875 |  Profit = -0.079 %
36 BUY : 36118.94140625 SELL : 36143.51171875 |  Profit = 0.068 %
37 BUY : 35928.0546875 SELL : 35939.76171875 |  Profit = 0.033 %
38 BUY : 35620.83984375 SELL : 35634.2890625 |  Profit = 0.038 %
39 BUY : 35800.88671875 SELL : 36131.80078125 |  Profit = 0.924 %
```

As we can see, there is much more trades when conditions to buy are less strict.

Then I decided to make conditions to sell stricter (only when current sentiment is < previous one – 30%):

```
1  BUY : 36325.25
 2 BUY : 36284.109375
 3 BUY : 36572.5625
 4 BUY : 33764.7265625
 5 BUY : 32961.4296875
 6 BUY : 32732.67578125
 7 BUY : 32850.5859375
 8 BUY : 33498.90234375 SELL : 33593.84375 |  Profit = 0.283 %
 9 BUY : 36810.0625
10 BUY : 37874.4609375
11 BUY : 37934.12109375 SELL : 37778.3359375 |  Profit = -0.411 %
12 BUY : 36995.49609375
13 BUY : 36732.31640625
14 BUY : 36306.640625
15 BUY : 37199.54296875 SELL : 37070.07421875 |  Profit = -0.348 %
16 BUY : 37016.49609375
17 BUY : 37228.3359375 SELL : 37345.73046875 |  Profit = 0.315 %
18 BUY : 37391.3203125
```

As we can see, there is less position closing.

There were 63 opened positions and 36 closed, so about 57% closed positions. Among this 53%, 17 were winning trades, and 19 losing, so about 47% winning trades. So, it looks like randomness was high (which is bad). We can't say yet if the whole algorithm is profitable because we have to sell the 27 remaining Bitcoins.

2nd Algorithm)

Daily program outputs:

Day 1	Day 2	Day 3	Day 4	Day 5
Profit : $ 640.0799999999581	PE can't buy / or sell Profit : $ 2228.710000000079	Profit : $ 383.71999999997206	Algorithm interrupted (always losing money)	PE can't buy / or sell PE can't close position Profit : $ 600.0

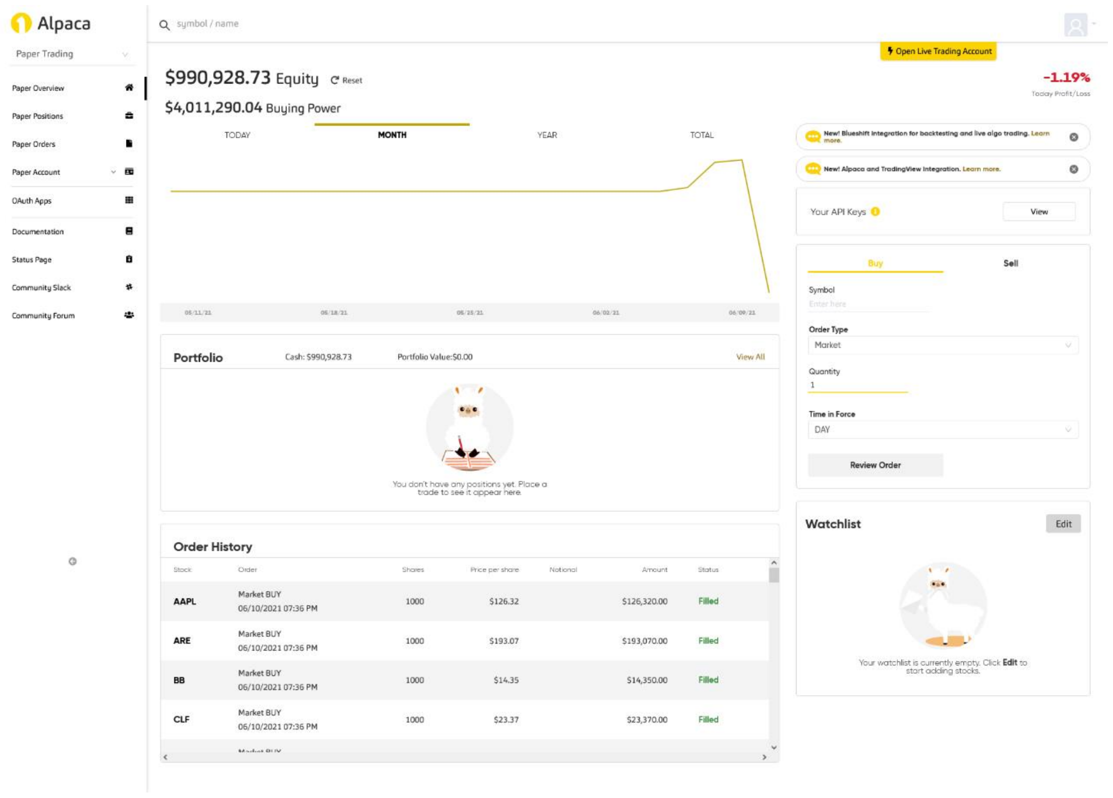
List of open / closed positions (screenshots):

Day 1: https://ibb.co/tz3pVFD, Day 2: https://ibb.co/VmSHJBQ

Day 3: https://ibb.co/6R3PZH4, Day 4: https://ibb.co/3F7mQr7

Day 5: https://ibb.co/Lnk4KxG

Day 4 was tough:



$990,928.73 Equity

$4,011,290.04 Buying Power

-1.19% Today Profit/Loss

Portfolio — Cash: $990,928.73 — Portfolio Value:$0.00

You don't have any positions yet. Place a trade to see it appear here.

Symbol: Enter here
Order Type: Market
Quantity: 1
Time in Force: DAY

Your watchlist is currently empty. Click **Edit** to start adding stocks.

Order History

Stock	Order	Shares	Price per share	Notional	Amount	Status
AAPL	Market BUY 06/10/2021 07:36 PM	1000	$126.32		$126,320.00	Filled
ARE	Market BUY 06/10/2021 07:36 PM	1000	$193.07		$193,070.00	Filled
BB	Market BUY 06/10/2021 07:36 PM	1000	$14.35		$14,350.00	Filled
CLF	Market BUY 06/10/2021 07:36 PM	1000	$23.37		$23,370.00	Filled

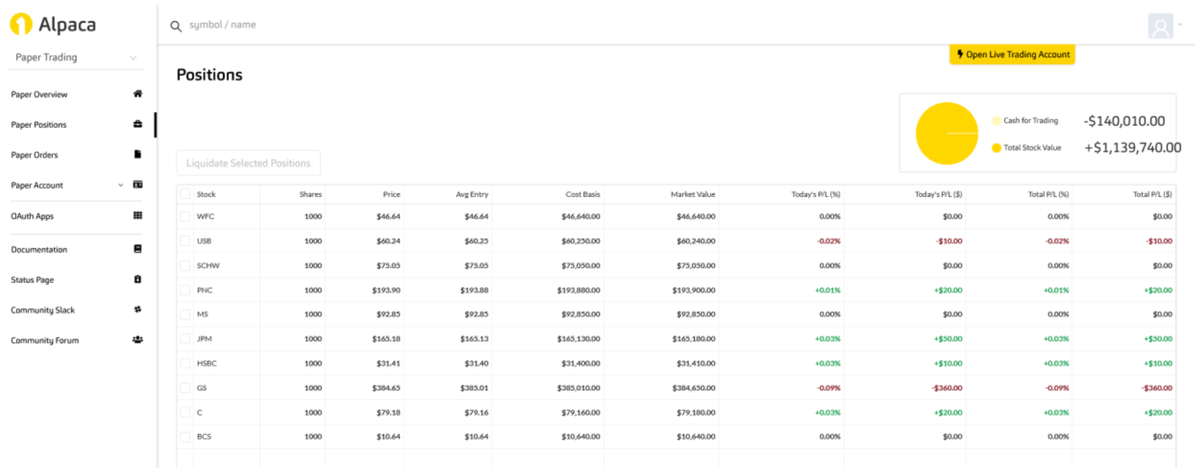
There is more profit on day 2 because I ran the bot several times on this day. I could make the bot even more profitable if the condition to close positions was not so strict (i.e. a higher take profit).

3rd Algorithm)

First (and only) output:

```
BUY JPM at 165.15 2021-06-08 19:49:37.499971+00:00
BUY HSBC at 31.405 2021-06-08 19:49:38.283278+00:00
BUY C at 79.15 2021-06-08 19:49:38.811455+00:00
BUY WFC at 46.635 2021-06-08 19:49:39.293030+00:00
BUY GS at 385.08 2021-06-08 19:49:39.934558+00:00
BUY MS at 92.875 2021-06-08 19:49:40.430601+00:00
BUY PNC at 193.94 2021-06-08 19:49:41.108389+00:00
BUY USB at 60.25 2021-06-08 19:49:41.676054+00:00
BUY SCHW at 75.06 2021-06-08 19:49:42.223928+00:00
BUY BCS at 10.635 2021-06-08 19:49:42.849609+00:00
```

Orders on Alpaca:



Positions

● Cash for Trading	-$140,010.00
● Total Stock Value	+$1,139,740.00

Stock	Shares	Price	Avg Entry	Cost Basis	Market Value	Today's P/L (%)	Today's P/L ($)	Total P/L (%)	Total P/L ($)
WFC	1000	$46.64	$46.64	$46,640.00	$46,640.00	0.00%	$0.00	0.00%	$0.00
USB	1000	$60.24	$60.25	$60,250.00	$60,240.00	-0.02%	-$10.00	-0.02%	-$10.00
SCHW	1000	$75.05	$75.05	$75,050.00	$75,050.00	0.00%	$0.00	0.00%	$0.00
PNC	1000	$193.90	$193.88	$193,880.00	$193,900.00	+0.01%	+$20.00	+0.01%	+$20.00
MS	1000	$92.85	$92.85	$92,850.00	$92,850.00	0.00%	$0.00	0.00%	$0.00
JPM	1000	$165.18	$165.13	$165,130.00	$165,180.00	+0.03%	+$50.00	+0.03%	+$50.00
HSBC	1000	$31.41	$31.40	$31,400.00	$31,410.00	+0.03%	+$10.00	+0.03%	+$10.00
GS	1000	$384.65	$385.01	$385,010.00	$384,650.00	-0.09%	-$360.00	-0.09%	-$360.00
C	1000	$79.18	$79.16	$79,160.00	$79,180.00	+0.03%	+$20.00	+0.03%	+$20.00
BCS	1000	$10.64	$10.64	$10,640.00	$10,640.00	0.00%	$0.00	0.00%	$0.00

PNL Dashboards:

Day 2: https://ibb.co/BP2LrHy, Day 3: https://ibb.co/KLkQ3Mf

Day 4: https://ibb.co/NsSbH9q, Global PNL: https://ibb.co/gRmx9mM

5. Conclusion

These results have been extracted from a very few days of experimentation, so it must not be taken as a proof of anything. On the overall, all three algorithms have made me lose money, even though, as I said, the first one is not finished yet, as it required some manual actions, and the second one could have been perfect if there was not the problem I encountered the 4th day. The mean reversion bot results were quite bad but seemed to get greater while time passed.