National Technical University of Athens
MSc - Data Science and Machine Learning

Programming Tools and Technologies for Data Science
Time-Travel Problem

February 2022

Contents

List of Figures

1 Introduction

In this report, we are exploring the problem of generating high profits by trading stocks from New York stock exchange. Trading is possible by travelling back in time and assuming there is a bank account with 1$ available on 1/1/1960. This amount is then used to buy and sell stocks given a large dataset consisting of historical data on stocks available on the New York stock exchange. The stock trading moves then span from 1/1/1960 until the last date in the data used.

1.1 Dataset

A dataset of 7195 stock files is provided. Out of those 32 are empty and are discarded from the analysis upon their discovery, hence the remaining dataset consists of 7163 files. Each file is a CSV file named "s.us.txt" where s is the stock's id. For instance for the AAPL stock of Apple Inc. the corresponding file is named "aapl.us.txt".

Below are presented the five first rows along with the header of "aapl.us.txt":

```
Date,Open,High,Low,Close,Volume,OpenInt
1984-09-07,0.42388,0.42902,0.41874,0.42388,23220030,0
1984-09-10,0.42388,0.42516,0.41366,0.42134,18022532,0
1984-09-11,0.42516,0.43668,0.42516,0.42902,42498199,0
1984-09-12,0.42902,0.43157,0.41618,0.41618,37125801,0
1984-09-13,0.43927,0.44052,0.43927,0.43927,57822062,0
...
```

As illustrated above, each CSV stock file contains seven attributes, out of which the first six are utilized (OpenInt attribute is dropped).

Attribute	Attribute's explanation
Date	Date for which stock metrics are provided
Open	Open price of stock at the start of a day
High	Highest price of stock within a day
Low	Lowest price of stock within a day
Close	Close price of stock at the end of a day
Volume	Volume of transactions (stocks sold and purchased) in a day

Each stock's data might span across a different daterange, for instance given that they are publicly offered in the New York stock exchange at different times.

1.2 Trading moves

There are certain moves that are allowed by the stock exchange. The two main moves allowed are the following:

- buy-low: buy at 'Low' price (lowest in a day)
- sell-high: sell at 'High' price (highest in a day)

Although, the 'Low' and 'High' prices of a stock are given for a certain date, it is unknown whether the lowest stock price 'Low' appeared before or after

the highest stock price 'High' was observed. Thus, those two moves (buy-low, sell-high) are considered to take place independently, i.e. one can only buy-low and sell-high a specific stock in a single day if there is at the same time sufficient balance (cash available) to perform the buy-low move as well as all the stocks being sold-high are already within the stocks' portfolio.

Moreover, there are also an additional 4 moves related with intra-day trading that are allowed.

- buy-open: buy at 'Open' price
- sell-open: sell at 'Open' price
- buy-close: buy at 'Close' price
- sell-close: sell at 'Close' price

1.3 Restrictions

There are certain restriction in the way trading takes place. First of all, moves within a day are executed in a particular order and there are three move teams:

$$\{buy{-}open, sell{-}open\} > \{buy{-}low, sell{-}high\} > \{buy{-}close, sell{-}close\}$$

The above order implies that first, transaction moves during market opening are executed, then during the day and finally at the market closing. Before each transaction move from a certain team can be executed, enough balance (cash) should be available for all purchase orders and enough stock count for all sales orders. The order of transactions within a transaction team is not taken into account.

There are also some additional restrictions:

- The total count of a stock s that is bought on day d should be at most 10% of stock's s volume on day d
- The total count of a stock s that is sold on day d should be at most 10% of stock's s volume on day d
- Assuming count $n >= 0$ of stock s available before the opening of day d, a maximum of $n + 1$ stocks from s can be bought during day d

1.4 Deliverable

By defining profit (in dollars \$) of a transaction moves sequence as the balance of the bank account once all moves have been executed (starting with 1\$ on 1/1/1960), we are looking to achieve a high profit for a sequence of N transaction moves, each one of which has the following format:

(d_i, m_i, s_i, x_i) for $0 <= i < N$, where:

d_i: day/date of move execution, m_i: move code (eg buy-low),
s_i: stock id (eg 'AAPL'), x_i: count of stocks for buying or selling.

In the next section, a valid sequence of $N <= 1,000$ moves that yields a high profit is generated programmatically as well as another one with more moves $N <= 1,000,000$

2 Data Analysis

In this section, we describe first an exploratory analysis on the stocks' data before setting up a trading mechanism.

2.1 Data parsing

Initially, all stock files within the dataset directory are parsed one by one in order to identify any data issues. 32 empty data files were identified and discarded. Then, the data of the parsed CSV files were aggregated and analyzed to identify prominent stocks.

2.2 Stock discovery

Firstly, we define three metrics to evaluate stocks and their potential to perform well and yield profits through their trading.

- AHLPPS - This is the average 'High' - 'Low' difference for a stock over all days it is available for trading
- MHLPPS - This is the average ('High' - 'Low') x 'Volume' for a stock over all days it is available for trading.
- LLHH - Highest 'High' - lowest 'Low' value for a stock in the daterange it is offered for trading

We consider MHLPPS the best metric among the above given that it considers volume also for a stock which is an important metric for profits assuming high number of stocks being traded.

Sorting the aggregated data for each stock in descending MHLPPS order and taking only stocks that are available for more than 1000 different days for trading, we get the following table:

	Stock	Days	AHLPPs	MHLPPs	LLHH	Init Price	Path	Start Date	End Date
2337	FB	1381	1.754384	73785974.963553	165.35	38.0	C:/Users/User/Downloads/DSML MsC/1st semester/...	2012-05-18 00:00:00	2017-11-10 00:00:00
12	AAPL	8364	0.441623	55229954.023734	175.37949	0.41874	C:/Users/User/Downloads/DSML MsC/1st semester/...	1984-09-07 00:00:00	2017-11-10 00:00:00
2849	GOOGL	3333	7.164068	52365077.965987	1015.64	47.98	C:/Users/User/Downloads/DSML MsC/1st semester/...	2004-08-19 00:00:00	2017-11-10 00:00:00
513	BRK-B	5415	1.111337	42501121.807071	170.88	23.1	C:/Users/User/Downloads/DSML MsC/1st semester/...	1996-05-09 00:00:00	2017-11-10 00:00:00
6487	TSLA	1858	5.210873	35856341.475408	381.58	17.0	C:/Users/User/Downloads/DSML MsC/1st semester/...	2010-06-28 00:00:00	2017-11-10 00:00:00
1700	DAL	9548	80.66043	34737877.259725	11765.0634	1119.46	C:/Users/User/Downloads/DSML MsC/1st semester/...	1960-01-02 00:00:00	2017-11-10 00:00:00
4327	MSFT	7983	0.413583	33357658.698252	86.1328	0.0672	C:/Users/User/Downloads/DSML MsC/1st semester/...	1986-03-13 00:00:00	2017-11-10 00:00:00
770	BIDU	3090	3.130992	33202482.440047	272.27	2.7	C:/Users/User/Downloads/DSML MsC/1st semester/...	2005-08-04 00:00:00	2017-11-10 00:00:00
350	AMZN	5153	4.413968	32701553.292797	1134.23	1.71	C:/Users/User/Downloads/DSML MsC/1st semester/...	1997-05-16 00:00:00	2017-11-10 00:00:00
1541	CSCO	6964	0.452222	32511759.568361	67.69114	0.06599	C:/Users/User/Downloads/DSML MsC/1st semester/...	1990-03-26 00:00:00	2017-11-10 00:00:00
6537	TWTR	1011	1.171975	32340410.737703	61.005	44.0	C:/Users/User/Downloads/DSML MsC/1st semester/...	2013-11-07 00:00:00	2017-11-10 00:00:00
618	BAC	7929	0.440964	30672800.689815	49.7503	3.6107	C:/Users/User/Downloads/DSML MsC/1st semester/...	1986-05-29 00:00:00	2017-11-10 00:00:00
2897	GS	4661	3.190439	30587074.518186	210.179	62.329	C:/Users/User/Downloads/DSML MsC/1st semester/...	1999-05-04 00:00:00	2017-11-10 00:00:00
3	AABA	5434	0.936624	29135074.716803	124.38	1.02	C:/Users/User/Downloads/DSML MsC/1st semester/...	1996-04-12 00:00:00	2017-11-10 00:00:00
647	BB	4715	1.119538	27204590.662653	146.99	1.9	C:/Users/User/Downloads/DSML MsC/1st semester/...	1999-02-04 00:00:00	2017-11-10 00:00:00
4532	NFLX	3201	1.203928	27114602.914924	203.1071	1.5214	C:/Users/User/Downloads/DSML MsC/1st semester/...	2005-02-25 00:00:00	2017-11-10 00:00:00
5355	QCOM	6527	0.816732	24369291.322962	84.10279	0.43127	C:/Users/User/Downloads/DSML MsC/1st semester/...	1991-12-16 00:00:00	2017-11-10 00:00:00
4313	MS	3201	0.936255	23355192.157904	62.4076	42.575	C:/Users/User/Downloads/DSML MsC/1st semester/...	2005-02-25 00:00:00	2017-11-10 00:00:00
3408	INTC	11556	0.301137	21922789.873856	60.73609	0.01592	C:/Users/User/Downloads/DSML MsC/1st semester/...	1972-01-07 00:00:00	2017-11-10 00:00:00
3597	JNPR	4624	1.493195	20240407.50718	228.4356	14.896	C:/Users/User/Downloads/DSML MsC/1st semester/...	1999-06-25 00:00:00	2017-11-10 00:00:00

Figure 1: Top 20 MHLPPs stocks with over 1000 days of availability

We observe that Apple Inc ('AAPL') which is listed as second from top on the table, has the following attributes:

- High MHLPPs value (potential high profits due to large price differences and high volume values)
- Large daterange of availability (over 8,000 days)
- First date of availability 1984, which along with the previous attribute makes the stock suitable for a long term trading strategy
- Initial price lower than 1$ making the stock suitable to get started investing on it

Further analyzing the stock, we draw the following diagrams:

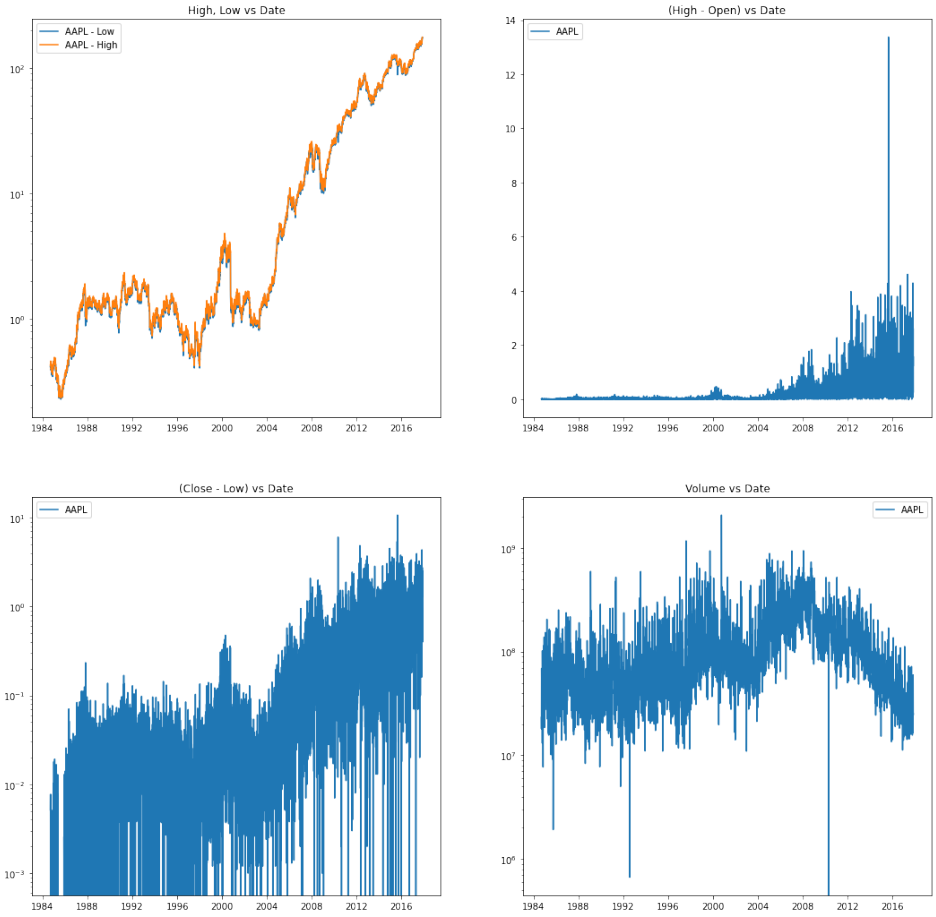

Figure 2: Apple (AAPL) stock diagrams

Here we can graphically see that AAPL is a stock that has a positive uptrend over time (logarithmic scale) as well as quite steady volume over a large daterange it is available on the stock exchange.

As a comparison, we have drawn a few comparative diagrams of AAPL vs MSFT (Microsoft) which is also present in the top 20 MHLPPs table and exhibits some of the properties of AAPL.

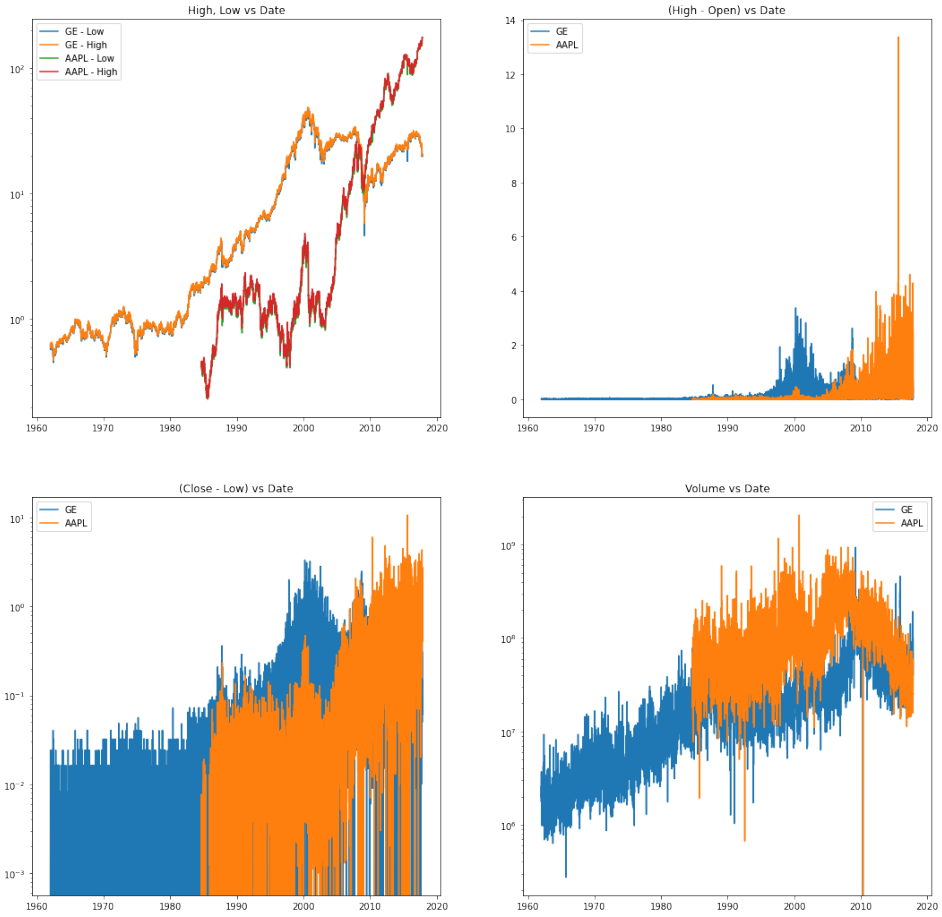

Figure 3: Apple (AAPL) vs Microsoft (MSFT) stock diagrams

Here one can see that MSFT is much more limited than AAPL in terms of the daterange it covers. Therefore, if we were to select a single stock to work with in order to generate a large number of profitable moves, AAPL would be more suitable given it is listed in the stock exchange earlier in time. However, a switch to MSFT would work well after some amount of time, since its volume surpasses the one of AAPL over time.

3 Trading

In this section, we proceed in implementing a trading strategy for generating a highly profitable sequence of valid moves.

3.1 One stock moving window strategy

For simplicity reasons, we devised a strategy for a single strategy trading. The way the algorithm works is the following.

Initially, it loads the stock CSV data of the selected stock in a pandas dataframe and given a number of moves N we have devised the algorithm to generate, it divides the dataframe in N parts, keeping the first one on the first iteration. On this working window's sub-dataframe, we seek out the lowest 'Low' price and if balance (amount in bank account) is available to purchase a number of stocks, then a buy-low move is executed purchasing as many stocks as allowed by the restrictions of the problem. Once the move is executed, a transactions/moves counter is increased and the new dataframe to be divided contains dates from the last transaction to its end date. Each subsequent dataframe partition is then made by taking the remaining rows of the dataframe and dividing this number by (N-T), where T is the number of transactions executed so far. This logic implements a sliding window effectively that takes into consideration that in general the AAPL stock is increasing over time. Hence, by buying at the lowest price of a window, we expect a high price in the next window, generating profits from the difference if bought at 'High'. Whenever there is no ability to buy additional stocks (eg volume restriction or no sufficient balance), then the highest 'High' price is found within the current window and stocks are being sold.

In order to advise the algorithm to favor more buying than selling, in order to avoid selling all accumulated stocks in a single day once, we introduce a parameter called inverse selling rate. This allows the algorithm to keep some stocks in the portfolio even after a selling event which allows it to purchase a larger amount of stocks a subsequent date, since only buying d+1 stocks on a next date is allowed when having d stocks the previous one. After experimentation with different values, we have selected 1.4 as a satisfying inverse selling rate for the algorithm. This means that the selling rate is $1/1.4 = 0.714$ or in other words, the max amount of stocks in the portfolio the algorithm can sell in a single sale event. Therefore, at least 30% of the portfolio's amount of stocks is preserved even after selling. A sale event generates extra balance that is then used for more buying events before another sale event is triggered. On top of that, for the first 50 transactions only, the inverse selling rate has been set to 1 so that it can more freely buy and sell at the beginning to generate an initial capital capable of working with the inverse selling rate custom value and higher volumes.

Naturally, towards the end of the N moves, there are several stocks left in the portfolio. A way to avoid being left with the stocks in the portfolio at the end of the daterange, is to set a number of keep out moves so that towards the end of the dates only sell moves are executed. This way the portfolio value which is the balance + current stock value (stocks value at 'Close' price of a date) ends up being equal to balance by the end of the N moves and the final moves counter might not be exactly N, but is surely less than or equal to N. For instance, for N=200 moves we might set a number of 20 keep out moves and the final moves count would be anywhere between 180 - 200 moves.

At the end, a diagram displaying the evolution of the portfolio and balance values over time is drawn and a txt file with the transactions is generated with the following format, where the first row is the number of transactions/moves and then the transactions follow.

382
1984-10-01 buy-low AAPL 1
1984-10-12 buy-low AAPL 1
1984-10-19 sell-high AAPL 2
1984-11-09 buy-low AAPL 1
1984-11-19 buy-low AAPL 2
1984-12-18 sell-high AAPL 3
...

3.1.1 AAPL $\leq$ 1,000 moves

The aforementioned algorithm is put to work with N=400 moves, keep out moves = 20, inverse selling rate = 1.4 for the 'AAPL' stock and it completes 382 moves in total. At the end a txt file is generated with the number of moves on top and all of the transaction moves below. This is provided along with the report as 'small.txt'

Additionally, a balance - portfolio (logarithmic scale) vs date diagram is created and the final balance or profit is calculated. This has been measured as c. \$1.793 billion. This profit is remarkably high for such a relatively low number of moves and proves empirically the validity of the initial hypothesis made.

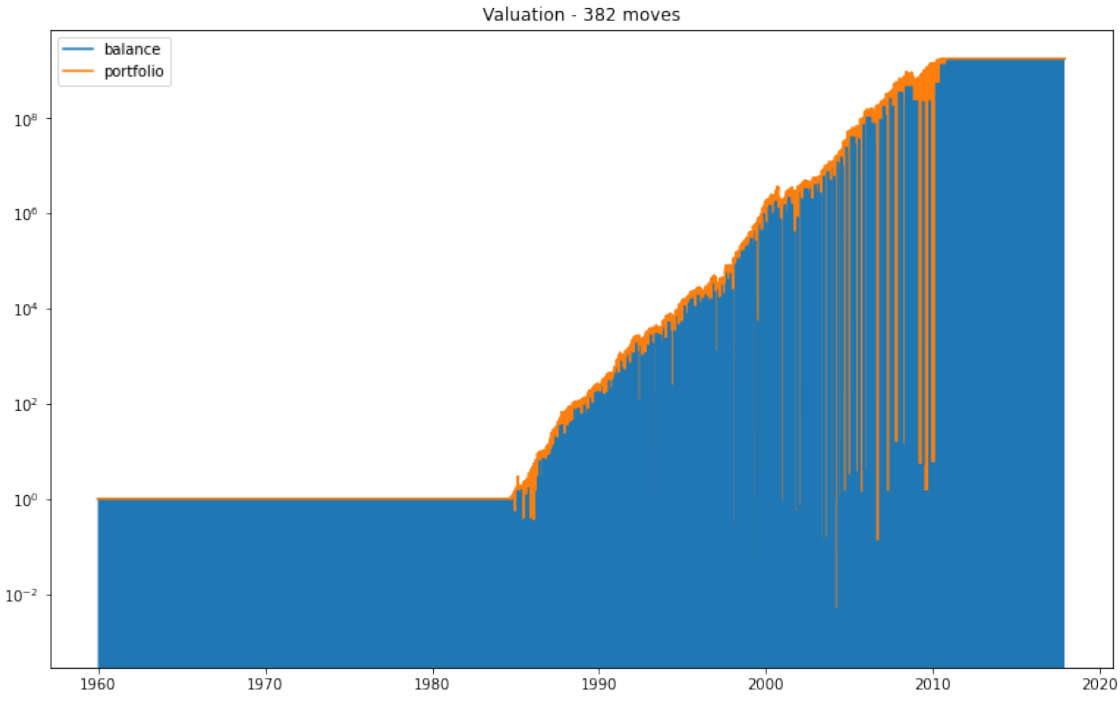

Figure 4: AAPL - simple stock strategy - 382 moves

We also performed the same experiment for Microsoft (MSFT) and for 381 moves, the profits yielded were significantly lower. More specifically, the profit

was c. $49.1 million. Below, is the relevant diagram for MSFT.

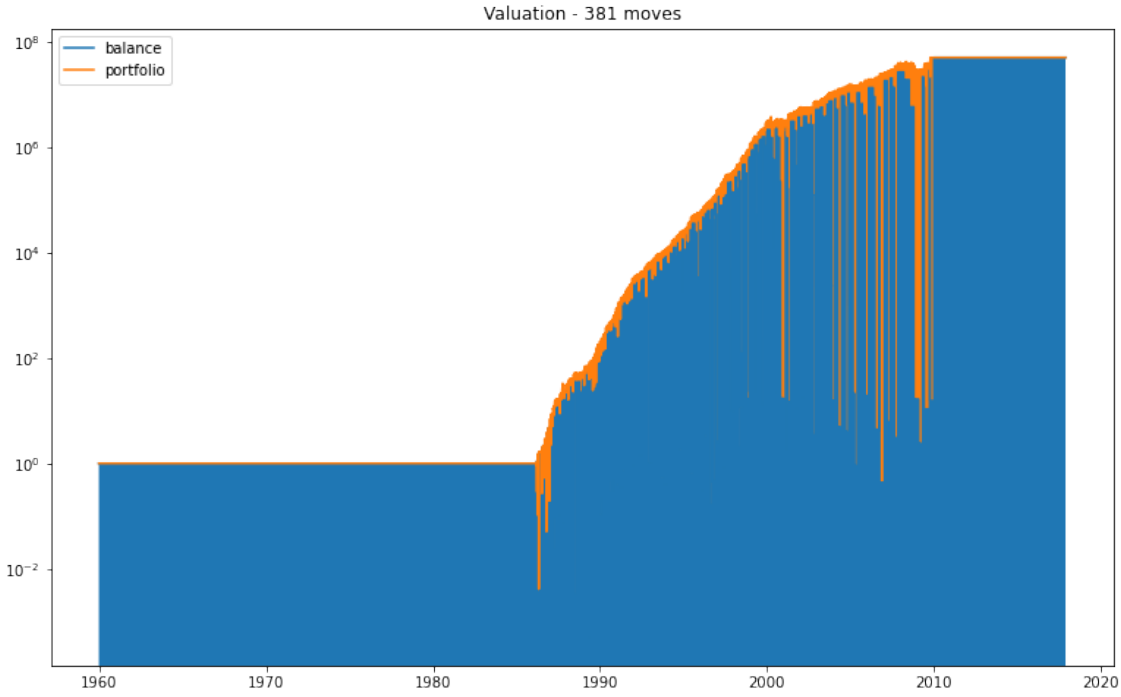

Figure 5: MSFT - simple stock strategy - 381 moves

3.1.2 AAPL $\leq$ 1,000,000 moves

Next, we examine the performance of the algorithm on more than 1,000 moves. In fact we select 7,000 moves for the single-stock sliding window strategy and 200 keep out moves for the final selling of stocks with an inverse selling rate of 1.4

For AAPL, the profits that we get at the end of the daterange and for 6990 moves are c. $ 108.63 billion. Below is the diagram of balance and profit vs date.

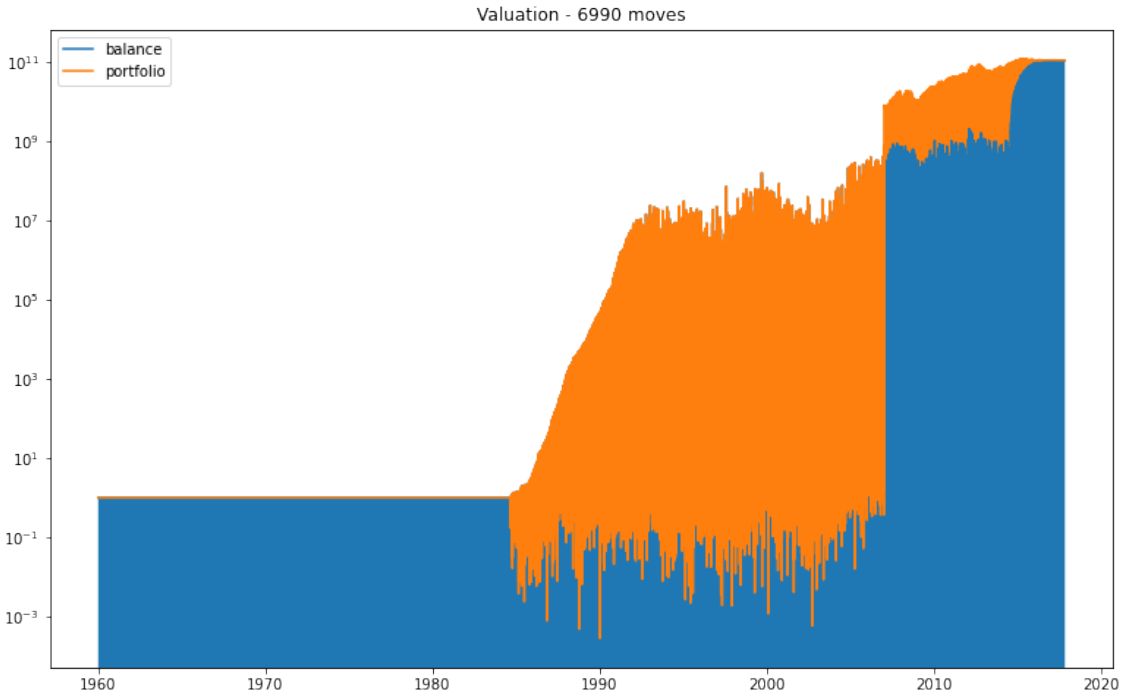

Figure 6: AAPL - simple stock strategy - 6990 moves

Similarly, we compare this to MSFT with the same settings having completed 6900 moves. The profit in this case is c. $ 17.65 billion.

Again AAPL performs better than MSFT with the devised strategy.

Below we present also the same diagram for MSFT for the 6900 moves.

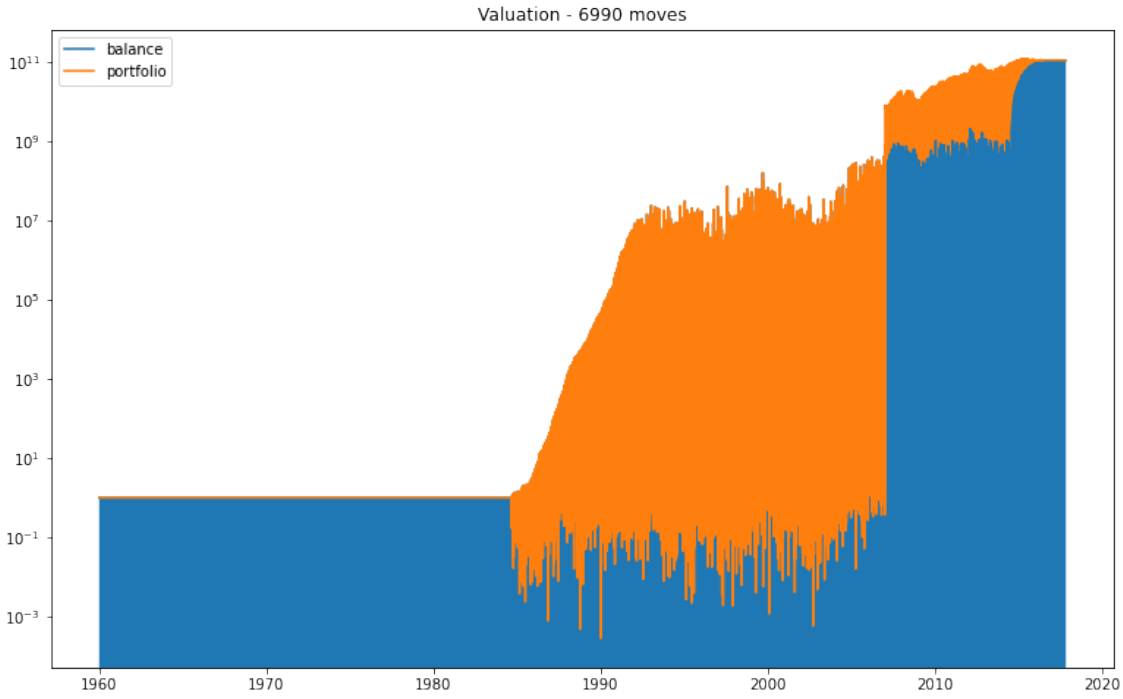

Figure 7: MSFT - simple stock strategy - 6900 moves

4 Conclusions

As a conclusion, the profits for AAPL are quite significant and therefore we are not going to implement a more complex model in this work. We also observe that AAPL is a highly profitable stock for trading over time.
The selection of the stock is essential for the algorithm to work. First of all, a stock with an initial price of less than $ 1 is required to get it started, which means that it is limited to a number of stocks only that satisfy this criterion. A few examples are AAPL, MSFT and GE stocks.
The inverse selling rate is a quite important parameter in the model given that it allows for the number of stocks held to overcome the restrictions and grow in a meaningful manner.
A stock with a high volume towards the end of the daterange is essential given that it allows for profit generation, if the stock is generally growing in value over time.
As a note also, the amount of profits generated are very large numbers that cannot be obeserved in reality, given that in live settings the high and low prices are not known.

5 Future work

It is interesting to note that a more complex model incorporating the ability to trade multiple stocks instead of a single one or the ability for intra-day trading would be a natural continuation of the work presented in this report. There are certain stocks that are quite suitable for generating large profits via intra-day trading based on the following metric for a date (High-Open)*Volume or (Close-Low)*Volume. An example is AMZN, which has a difference (High-Open) of around 10 for each date between around 2010 - 2017. Same applies to GS.

There are also some stocks that exhibit a remarkably odd behavior, which is a spike in price for a certain date. We regarded this data as corrupted, however, a targeted algorithm could explore such stocks and generate large revenues. For example, SMBK has a peak in early 2016, with its price going from around $15 to $199,999.99 in just a single day. Some other stocks with similar characteristics are BRK-A, BOFIL, TGEN and OFG_A. Below is some diagrams displaying this behavior of SMBK.

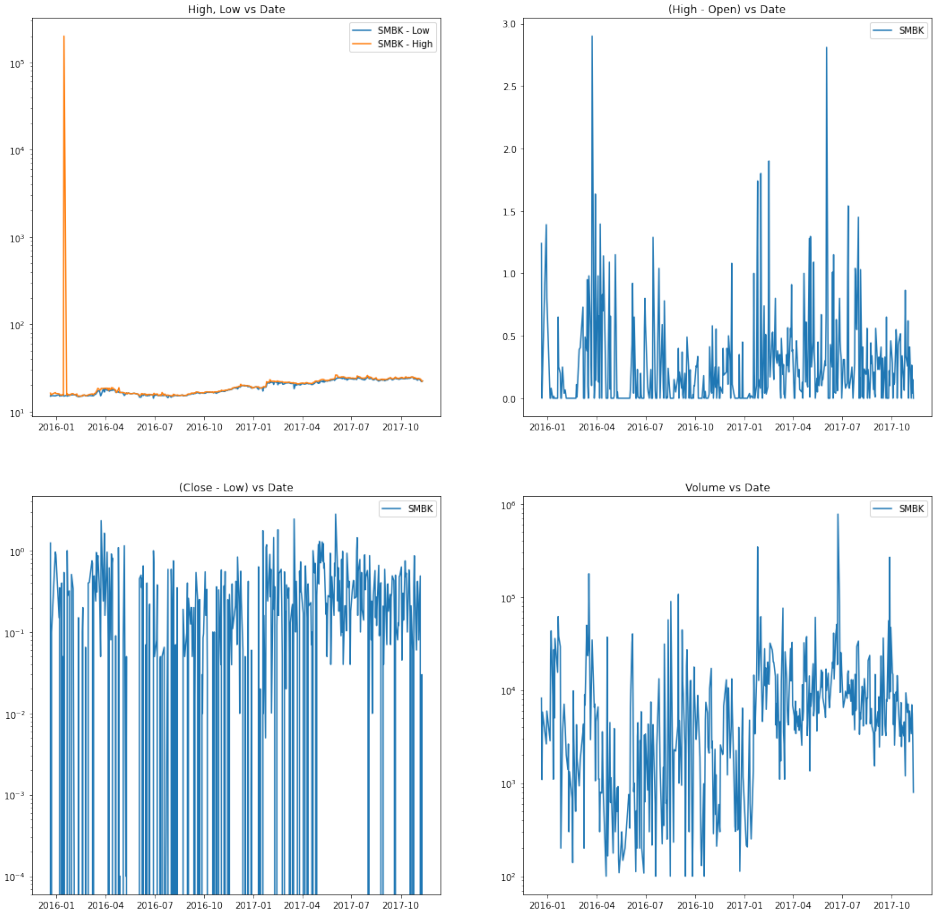

Figure 8: SMBK - analysis