Market Modeling and Analysis with High Frequency Data

Eduardo Ferrari Magalhães

Advisor: Raul Ikeda Gomes da Silva

Insper, 2018/2019

INDEX

1 Introduction 1
2 Methodology 1
2.1 Data Structure 2
2.2 Preliminary Tests and Platform Selection 3
2.3 Model Architecture 4
2.3.1 General Pipeline 4
2.3.2 Filters 4
2.3.3 Trading Algorithm 5
2.3.4 Overview 7
2.4 End User Interface 7
2.5 Parameters Calibration 9
2.6 Backtesting 11
3 Results 11
3.1 Data correlation tests 11
3.1.1 Time-Price Fluctuations correlation tests 11
3.1.2 Volume-Trades and Volume-Price Fluctuation Correlations 12
3.1.3 Ticks-Price Fluctuation correlation 14
3.2 Moving Average Tests 15
3.3 Model variables analysis 17
3.4 Bagging 19
3.5 Grid Search 21
3.6 Backtesting 23
4 Conclusions 24
5 References 25

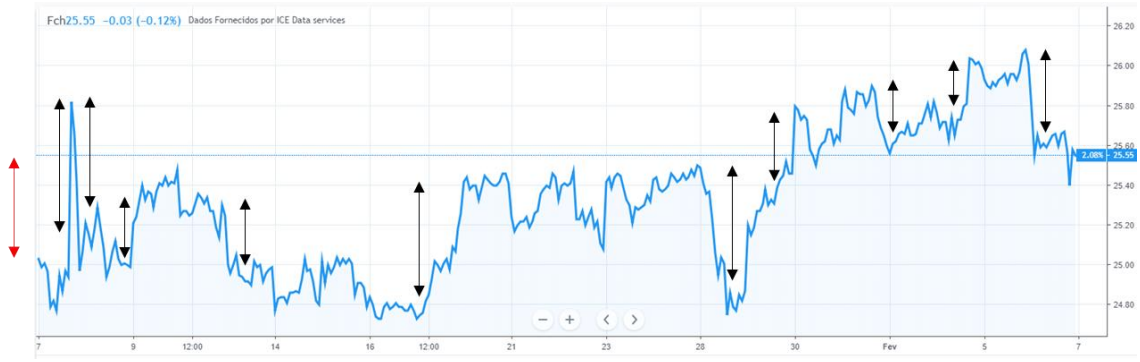
1 Introduction

The advances in high frequency market data registration and availability allowed the development of new methods of stock trading that take advantage on smaller and quicker market oscillations with really short-term investments. These methods are generally grouped as High Frequency Trading algorithms, and are the focus of the current report.

Usually most investment funds avoid intraday trading due to high trading taxes and tend to focus on mid or long-term investments that can yield a very reasonable return, but miss on smaller market oscillations that, with new technologies, may be used to increase return.

For instance, looking at a 1-month investment in the stock represented by the graph in Figure 1 it is possible to estimate about 2% profit with the stock price going from R$25.03 to R$25.55 (red arrow). It was a good profit averaging a return of 26.8% per year, still so many opportunities were wasted. Looking at all the black arrows it becomes obvious that with much shorter investments very similar results are achievable. In the same period the overall profit could be over 5 times greater with smaller, more precise investments.

Figure 1: PETR4 Stock value from 01/07/2019 to 02/07/2019



Another disadvantage of a longer-term investment is that the market high volatility makes the results of an investment really unstable overtime and can lead to an early withdraw of the investor. For instance, in the first few days of the month the one-month investment yields a huge profit that disappears the next day, then the profit raised again for couple days just to become a loss that could discourage the investor.

The difficult part is having a trade model good enough to detect market tendencies and react quickly so it can take advantages on more opportunities. Here, High Frequency Data as studied by Engle, R. (1996) can be analyzed and filtered to create a much better than usual investment strategy.

Despite not existing as many models in high frequency trading as in a normal trade rate (Aldritch, I. – 2013) this actually means there are many ways to develop this system and maybe create a somewhat reasonable trading model for the Bovespa in Brazil.

On that premises, this study is based on the process of high frequency data modeling, presenting: the search for market variables correlations, creation of a noise filter, a trading algorithm, a testing platform, and model backtesting.

2 Methodology

In this section all data analyzed and methods of both filtering, modeling and trading are explained. Overall, the steps taken to develop this project are summarized in the algorithm below.

Procedure: HFT Study

Initialize:
- Understand the Data Structure
- Choose a platform to work on (Excel, R, Python, MatLAB)
- Define a model score metric
- Create a simple model for architecture tests
- Develop the architecture for high flexibility
- Develop an Interface

Loop:
- Search for new methodologies
- Develop filters and trading algorithms
- Test the model

2.1 Data Structure

The first step into developing any market model is understanding the data structures that are related to the market, for that matter there are three main types of events that are recorded and available with less restrictions, those are "BID"s, "ASK"s and "TRADE"s, and there are three variables that can be associated with them, the time of transaction, the volume traded and the price offered.

Let´s consider for now X being a volume amount, Y being a price and T being a time, now defining the 3 events above can be done as a function of these 3 variables. ASKs are the attempt of selling a X amount of stocks for a Y price at a T time and Bids are the attempt of buying a X amount of stocks for a Y price at a T time, finally Trades mean that a X amount of stocks were sold/bought for a Y price at a T time.

These three main actions combined with these three variables constitutes most high frequency data datasets available in the market. As an example, the data being used in this project from Bloomberg[1] terminals can be summed in a list of this events organized by timestamps with 1 second resolution, together with the volume and stock price. A data sample is shown in Table 1.

Table 1: Data Sample

Time	Event	Value	Volume
20/07/2018 10:55:02	TRADE	19.17	600
20/07/2018 10:55:02	BID	19.16	42300
20/07/2018 10:55:02	ASK	19.17	300
20/07/2018 10:55:02	ASK	19.16	300
20/07/2018 10:55:02	TRADE	19.16	300
20/07/2018 10:55:02	BID	19.16	42000

In Table 1 is possible to see the evolution of events in this short period of time. In the first line there is a TRADE with BIDs and ASKs sent previous to this sample, setting the current stock price to 19.17. After that, a BID of 19.16 appears and subsequently two ASKs are sent, the first one with a higher value (19.17) than the current BID not matching a trade, and a second with the same value as the BID (19.16) filling a trade, registered in the fifth line. After the 300 volumes trade, the old 42,300 volume´s BID in line two now become a 42,000 volume´s BID in line six.

[1] https://www.bloomberg.com/

This data makes it possible to keep track of a determined stock value, best ASK and best BID prices and volume traded. Therefore, it can be used to feed a model to simulate trades as a normal investor would.

A very important factor that can be observed with BID and ASK values is the value spread. The value spread is the difference between the ASK and BID prices of a stock in a certain time. For example, in the third line of the sample data in table 1 the spread between the 19.16 BID to the 19.17 ASK is 0.01. This difference between ASK and BID prices can influence in a model profit, especially if it is a high frequency trading model. Mainly because if your model takes advantage of very small price differences and the spread between ASKs and BIDs is bigger than that price difference it may lose profit. Basically, if there is no one selling the stock at the last trading price, the investor is forced to go to the next best ASK if he wants to complete the trade, usually paying a higher price for the stock. The same rule can be applied when buying a stock.

2.2 Preliminary Tests and Platform Selection

After understanding the dataset making some quick analysis regarding the data distribution on time, volume and price of the stocks. In these analyses it was also possible to compare different platforms in processing speed, flexibility and interface capability.

Using Python[2] programing language and Pandas[3] library some simple models such as moving average in timestamps were tested and some first fit curves started to develop. The standard moving average, in stock price filtering, uses a fixed window to iterate over data and takes the mean of the window as the current stock price, by doing so the moving average filters noise from a sample but creates lag between the model and the real stock price. Therefore, despite moving average being very extensively used in market analysis, depending on the size of the window, it can filter to little noise from the data or lag too much behind the actual stock value.

Usually, for longer term investments, choosing a bigger window and having some lag in the filtering process, is not a big problem assuming time triggers are less sensitive, however it is a problem for HFT models, where couple of seconds can be the difference between a good and a bad trade. At the same time, using a smaller window can take noise as price fluctuations and send wrong signals for the trade model. The lag is illustrated in Figure 2 through the use of a moving average in daily stock price, the green line represents the actual price and the blue and red lines represent a 20 day and 100 day moving average respectively.

Figure 2: Moving average delay demonstration

[2] https://www.python.org/
[3] https://pandas.pydata.org/

This first tests also showed that Python would be a very efficient language to work with the data, despite not having an interface like excel, it is a really versatile language with many different libraries that would allow the whole backtesting and interface development be done on it.

2.3 Model Architecture

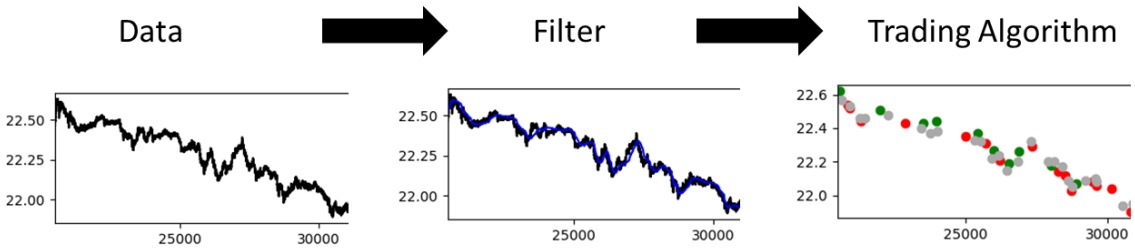
In this section details on the development of the architecture pipeline, filters, and trading algorithms are presented.

2.3.1 General Pipeline

It was decided that three different parts that work sequentially would be the best approach for the architecture pipeline as it offers the ability to test each part individually, making the whole structure more reliable and modular. Specifically, the modular characteristic is important so that testing different combinations of filters and trading algorithms is possible, not requiring any extra work when only part of the model needs to be changed.

The first part of the model is the data receiver and broadcaster; this part would be responsible for reading the data from the dataset and broadcasting it to the model. The second part is the filter; it would receive all the data from the dataset and create a filtered version of the data with less noise and smoother transitions. The final one is the trading algorithm; this last part uses the filtered data to decide whether it should or should not trade the stock. Figure 3 illustrates the whole pipeline.

Figure 3: Created model architecture



To validate the structure models were coded and some test were performed with a simple trading system that calculated profit based on 3 simplifications: no value spread in BID and ASK which meant that the stock price was always the stock value (last trade value), no taxes regarding number of trades or amount traded and unlimited stock availability at any given time of the day. Another simplification that was made was the exclusion of the initial auction since it works in completely different rules and could impair the filters and model capabilities.

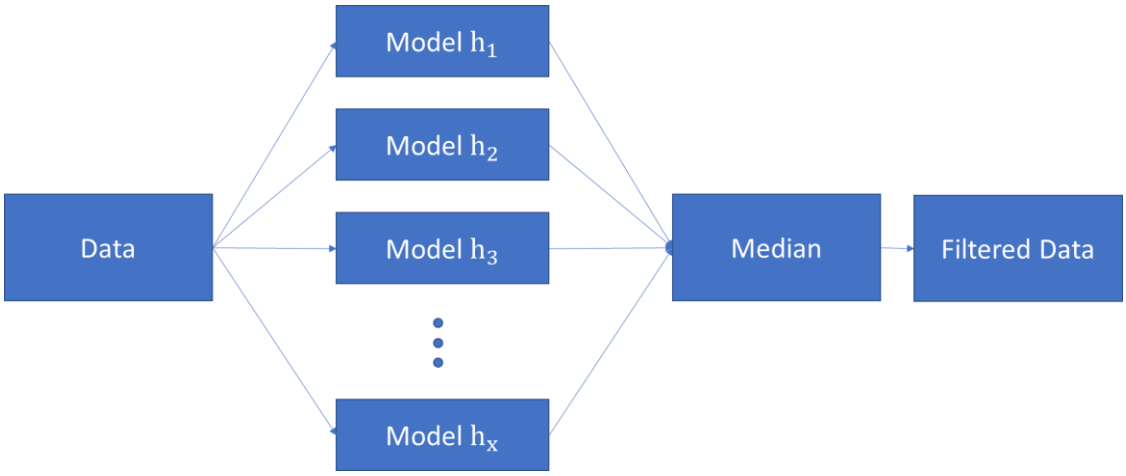
After the first results the simplifications started to be removed, starting with taxes and going to value spread. And the first models developed quickly got worst, and from profitable, went to completely failures. But the architecture was more robust and displayed more realistic results. After more tests, the architecture results were compared to simpler standalone models so it could be validated, and the implementation of an interface and backtesting system could be done.

2.3.2 Filters

The first 3 filters tested were moving averages in 1-minute window (makes the mean of the last 60 seconds of trades values every second), moving average in 150,000 volumes traded and moving average in the last 120 ticks. The steps were actually chosen based on the average number steps that the time filter needed to fill 1 minute of real time, in other word, it takes about 1 minute to have 150,000 volumes traded or 120 ticks to pass in average. Yet it quickly became clear that for the filters to work better a variable had to be assigned for the filter steps so it could be adjusted to match any trading algorithm it could work with. Additionally, it would allow testing the effect on different parameters in the model calibration.

The variable h was assigned to the filter step, and despite helping to tweak the filter, all models became really sensitive to changes in this variable. To work around this issue a Bagging[4] methodology was implemented as it can usually help different models to converge in a better solution. This technique involves running different filter s simultaneously and doing a pool with all the results to take a course of action. In this case it would be the same filter, but with a wide range of steps for the filter. As for the pool the filter used the median approach to find the best result. Figure 4 illustrates how the bagging method works.

Figure 4: Illustration on how Bagging worked on the system



The improvement in the models was clear for some combinations of parameters and despite the increase in computational time there was much less guessing and the results became much more consistent for all different basis filters. The downside of this method was the addition of a search range to the model that mostly replaced the need to tweak h for the need to tweak how big was the range of h's that ran concurrently.

After all the tests there were a total of 4 filters, all based in moving averages. The filters were: moving average in Ticks, moving average in Time, moving average in Volume and moving average in Ticks with Bagging

2.3.3 Trading Algorithm

The trading algorithm, would in practice be a sum of two parts, a signal, and a broker since the operation of detecting signals and trading can become fairly complex and allowing modularity between signals and brokers would allow more detailed tests. The signal would analyze the filtered data, detect a tendency and send an order of buying, selling, zeroing or holding for the broker, and the broker would attempt to fulfill the order according to a set of rules.

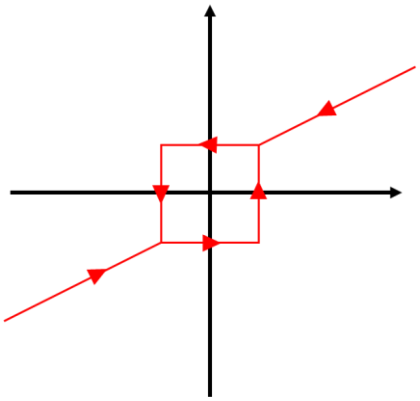
The first signal developed was an equation based on the numerical derivative of the filtered data. This algorithm calculates the numerical derivative of the stock price with difference of the last step of the filter and current step divided by the time that past between them (Equation 1), then it would see if the stock price is rising (derivative greater the 0), decreasing (derivative smaller than 0) or stable (derivative equal to 0). With the derivative, the trading algorithm assumed that the value tendency would remain constant and bought stocks when the price was rising, sold when it was decreasing and zeroed when it was stable.

$$\frac{dp}{dt} = \frac{p_t - p_{t-1}}{\Delta t} \qquad (1)$$

[4] Hands-On Machine Learning with Scikit-Learn and TensorFlow

However, since it's almost impossible for the derivative to be exactly 0, and the stable option is what triggers the model to zero the stocks, for this signal to work a dead-zone filter was required. A dead-zone filter is a gap between values received as shown in Figure 5, in this case derivatives, and it is used so model doesn't skip any signals or abruptly change signal with small price fluctuations. Equation 2 illustrates the calculation for the signal, the parameter α represents the length of the dead zone and regulates how sensitive the signal is to the price derivatives

Figure 5: Dead-Zone Representation



$$F(x,h) = \begin{Bmatrix} BID,\ if\ F > \alpha \\ Zero,\ if\ -\alpha < F < \alpha \\ ASK,\ if\ F < \alpha \end{Bmatrix} \quad (2)$$

The downside of the dead zone is that, just like with the step adjustment for the filter, the variable α had to be tweaked for the best results to be achieved and finding the best α manually was not viable. Therefore, a similar approach was implemented with the same method of bagging being used for the Signal as well.

After all the tests there were a total of 2 signals, both based on derivatives but one using bagging for the parameter α and the other using the user input for it. It is important to notice that other signals could have been implemented for the architecture such as Bollinger bands or Parabolic SAR.

As for the Broker the first developed was a really simple "fulfill the order regardless of the price" and was named standard as no algorithm is necessary for this kind of trade. Some stock markets actually have a kind of BID/ASK order that follow this exact rule so it was the simplest possible broker. The problem of this kind of trade is the spread and the high frequency attainable as the price being paid for the volume required is whatever is being asked for the stock no factors considered and the speed in which stocks can be bought when no price is taking into account is considerably faster than pondering whether the trade is at a good price or not.

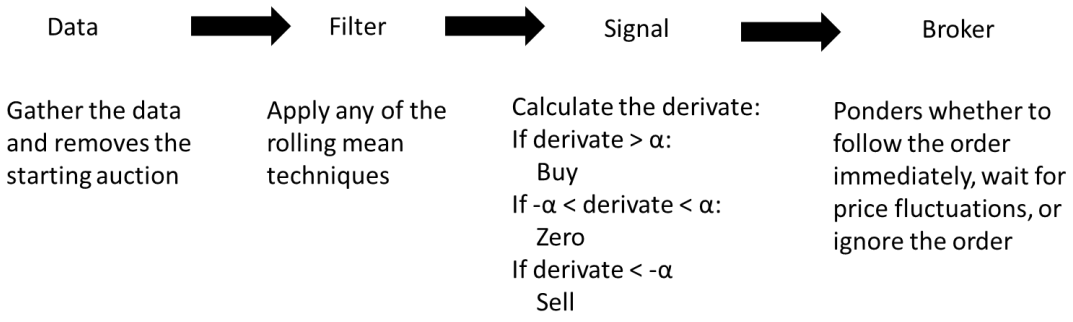
As a solution for this problem a new broker called "waiter" was developed, this new algorithm after receiving the signal did not immediately react and bought or sold stocks. Instead it waited to see if the BID or ASK price would fluctuate toward the exact price it wished to buy or sell the stock. If that did not happen it would wait a certain amount of ticks and then proceed to buy the stock if it was still close enough to the original price. Because of the market noise that causes really small prices fluctuations, with this technic, most of the time, the waiter broker actually managed to save the spread and get better trades overall.

After all the tests there were a total of 2 brokers, the standard (instant order execution) and the waiter

2.3.4 Overview

In general, the pipeline worked as presented in Figure 6. But besides of the pipeline development some other tests were performed with the data to understand why some models performed better or worst in certain situations, specifically correlation tests between model basis and the price of the stocks helped to differentiate the filters even further.

Figure 6: Model used in first tests



2.4 End User Interface

The interface was designed in Tkinter[5] Python library since it can illustrate well enough the desired outcome of this first interaction with user inputs and program outputs.

The software had to have for inputs:

- trade date or range – To allow for different dates to be simulated
- Graph refresh rate – To allow the user to see the day develop as an actual simulation
- Filter selection – Select time, volume, or ticks moving average with or without bagging
- Filter specifications – Select moving average step
- Trading algorithm selection – Select signal (Derivatives) and Trading Algorithm (Standard, Waiter)
- Trading algorithm specifications – Select the signal α and amount of stocks to buy
- Begin/Stop simulation button – Allow users to start and restart the simulation

And for outputs:

- Filter precision and lag – Graph to display how well the filter is following the data
- Trading algorithm trades – Graph to display the type, time and value of trades made
- Raw profit (not considering taxes) – Graph to display how the model profit is behaving
- Real profit – A absolute value with the taxes calculation result
- Current BID and ASK – The absolute value of best BID and best ASK values
- Number of trades – The absolute value of trades made in the day

With all that considered the interface starts with a language selection screen with two options Portuguese or English and have English as the default. The small language selection popup is displayed in Figure 7, and the interface in English and Portuguese is shown in the Figure 8 and 9 respectively.

[5] https://docs.python.org/3/library/tkinter.html

Figure 7: Language Selection Window

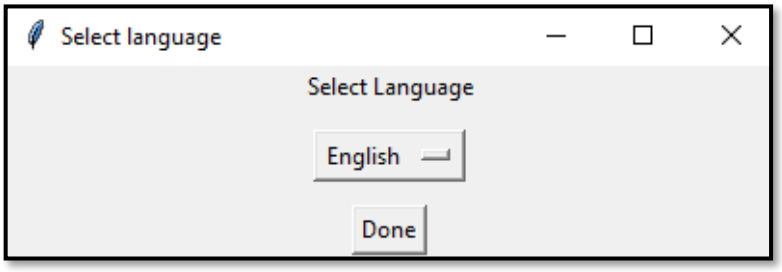


Figure 8: Interface developed for the trading simulation software English

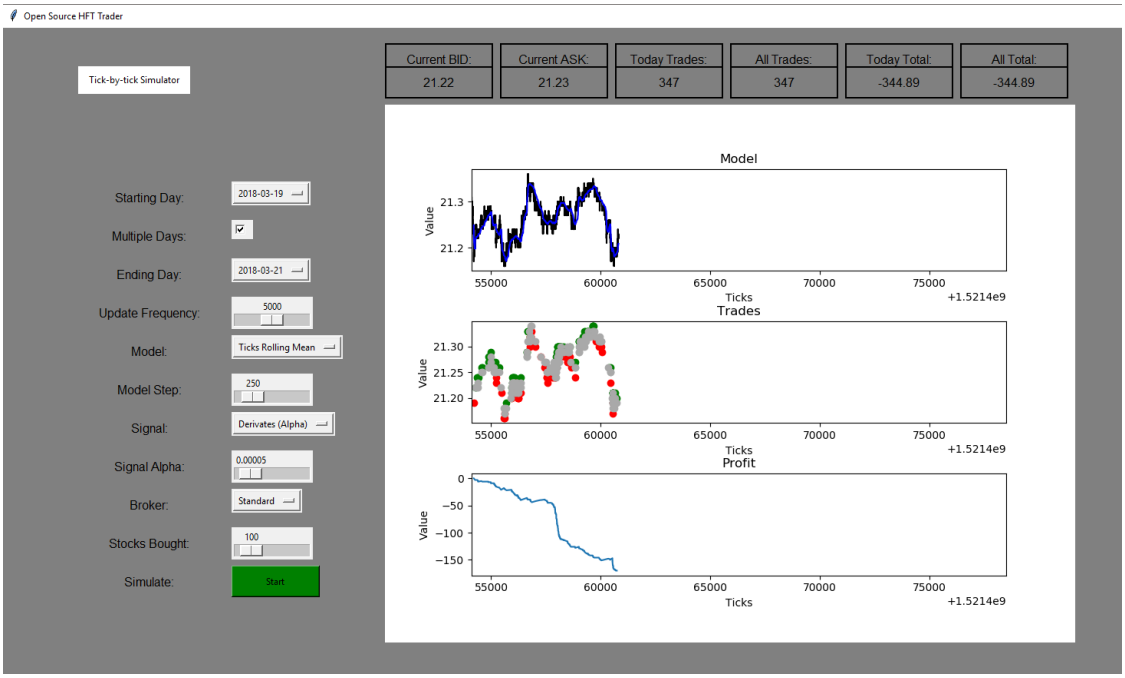
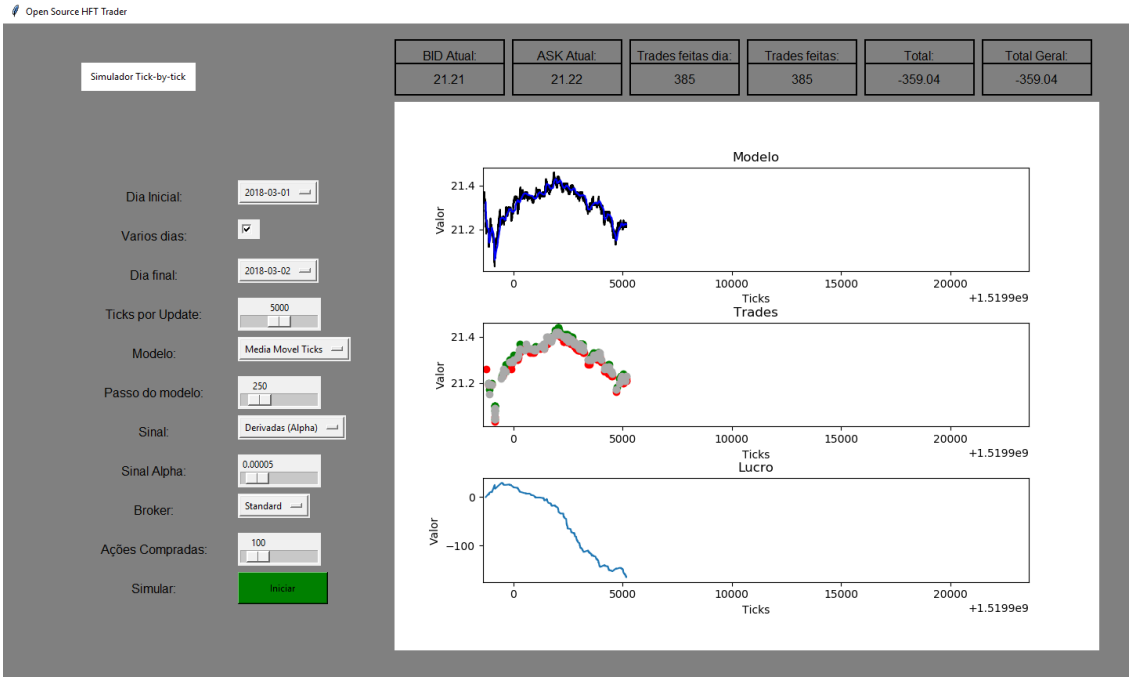
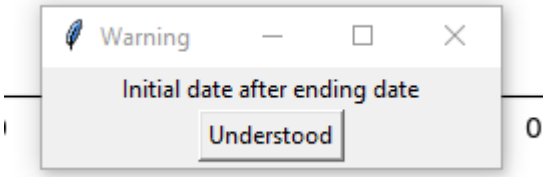


Figure 9: Interface developed for the trading simulation software Portuguese



The interface also has an error display box in case any option selected is invalid, like a starting date after the ending date or no trading date selected. Figure 10 illustrates one of these error´s popups.

Figure 10: Error PopUp example

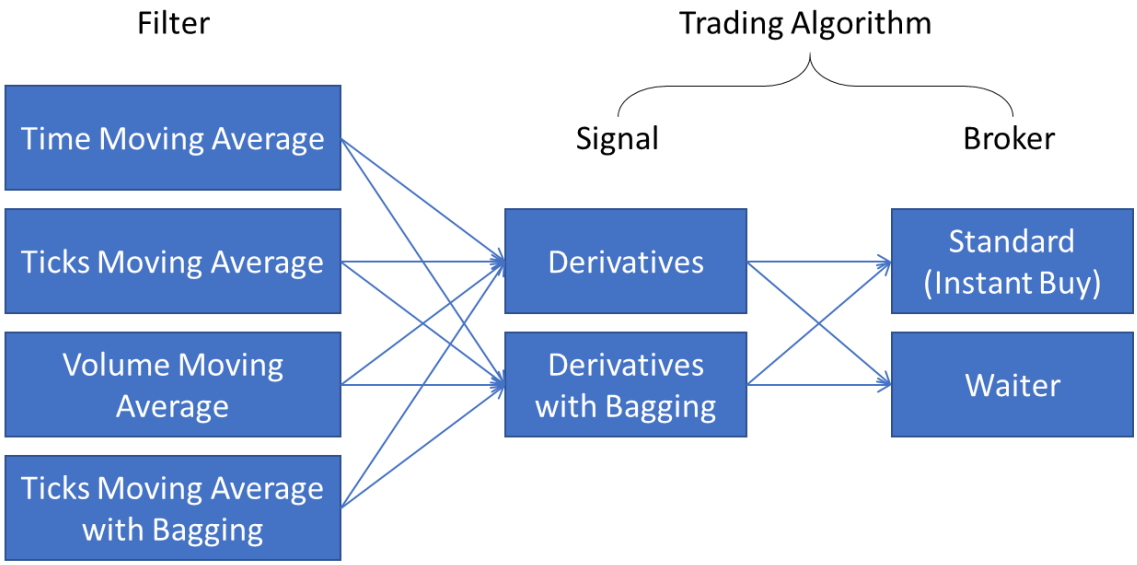


After all user errors were mapped to correct debug prompts and all functionality of the interface tested it was ready to be used, and more tests could be performed focusing on the models.

2.5 Parameters Calibration

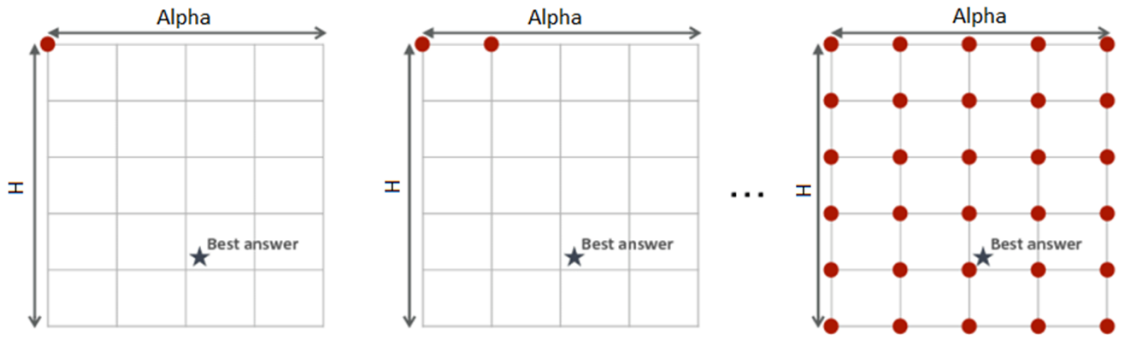
After more tests there were a total of 4 useful filters (Ticks, Time, Volume and Ticks Bagging), 2 different signals (Derivatives and Derivatives with Bagging) and 2 brokers (Standard and Waiter) which together could generate 16 different moving average-based models as seen in Figure 11. Considering there were also countless possibilities for the values of the parameters α and h using the interface to do tests became inefficient. A solution to the problem was going back to the source code and programing a grid search to try and find the best models and parameters.

Figure 11: Possible Model Combinations



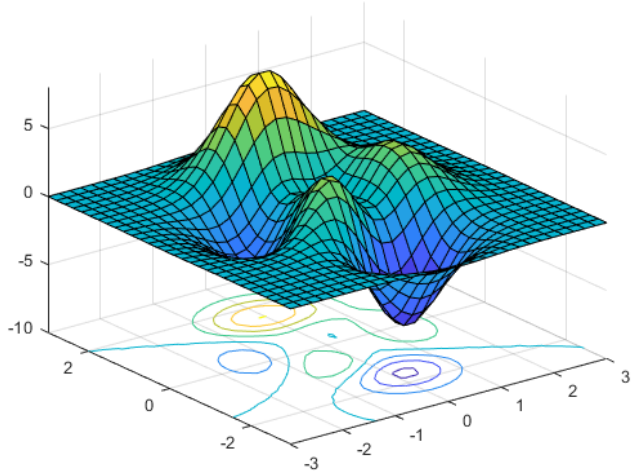
A grid search consists of testing a different set of variables in a user defined range in the attempt of finding the best combination of parameters and therefore getting the best model possible. Another interesting thing about the grid search is that since it tests the results for many combinations of variables it can create a surface (when there are only 2 different tested variables) that describe the performance of the model. The process of grid search is illustrated by Figure 12 and the surface that can be created to look at model performance in function of those variables is illustrated in Figure 13.

Figure 12: Illustration of a Grid Search process



Source: Adapted from "An Efficient Planner for Large-scale Predictive Analytic Queries"

Figure 13: Model quality surface in function of parameters



Source: https://www.mathworks.com/help/matlab/ref/surfc.html

Considering that many possibilities had to be tested, a first grid search was performed with the model's configurations listed in Table 2, and values for α's and h's according to Table 3.

Table 2: Models used in the Grid Search

Filter (Moving Average)	Signal	Broker
Ticks Bagging	Derivatives (α)	Waiter
Ticks	Derivatives (α)	Standard
Ticks	Derivatives (α)	Waiter
Volume	Derivatives (α)	Waiter
Time	Derivatives (α)	Waiter
Ticks	Derivatives Bagging	Waiter
Ticks Bagging	Derivatives Bagging	Waiter
Ticks Bagging	Derivatives (α)	Standard
Volume	Derivatives (α)	Standard
Time	Derivatives (α)	Standard

Table 3: Parameters used for grid Search

h´s \ α's	0.0001	0.0002	0.0003	0.0004
100	100/0.001	100/0.002	100/0.003	100/0.004
500	500/0.001	500/0.002	500/0.003	500/0.004
1000	1000/0.001	1000/0.002	1000/0.003	1000/0.004
1500	1500/0.001	1500/0.002	1500/0.003	1500/0.004
2000	2000/0.001	2000/0.002	2000/0.003	2000/0.004

The biggest advantage of doing the grid search outside the interface created was the implementation of a multi-process python library called joblib[6] that allowed the computer to do one grid search step per processor thread, and therefore reduce the computing time by 8 or 12 times depending on the processor used. In this report it was an i7 8750-H[7] with 12 cores running at 4.10Ghz, that allowed to run the whole search about 12 times faster.

Despite some further testing being able to reveal even better parameters with bigger and more detailed α's and h's ranges, the results were good enough to make a backtesting and verifying the created model's quality considering that even with multi-process enabled the whole search for 5 market days took an average of 6 hours.

2.6 Backtesting

The top 3 performing models in the grid search were selected for a backtesting of 6 months, again the developed architecture was used, but not the developed interface considering that again the multi-process library could be used to optimize the backtesting by simulating 12 days concurrently reducing the simulation time substantially. After the backtesting the best model could be selected, and considering a 6-month margin, it would be a more assertive judgment on the model's quality.

3 Results

In this section the study results are discussed, and some of the information used for methodology development is shown and clarified.

3.1 Data correlation tests

The first analysis performed was the correlation between the model basis (time, volume, ticks) and price fluctuations. For instance, if a big volume change could be related to a big price change, using a volume-based model would most likely have some sort of advantage over models with 0 base-price correlation.

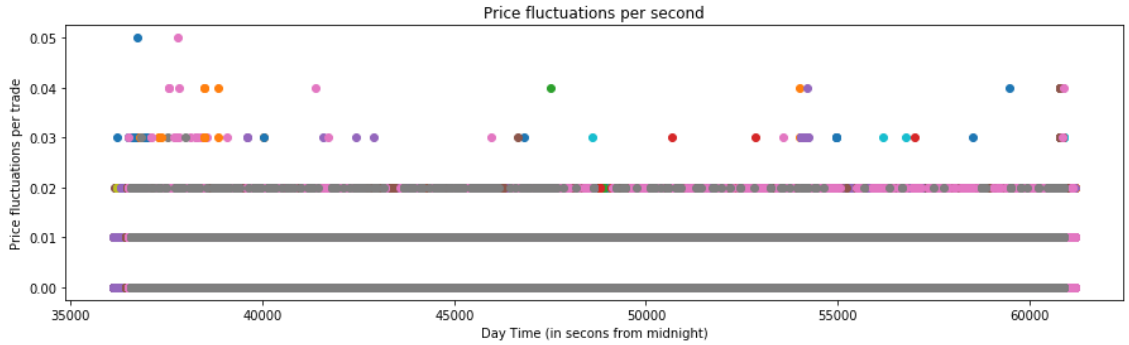
3.1.1 Time-Price Fluctuations correlation tests

Starting with time, checking if there is a specific time of day when price fluctuations were more significant, or, at least, if a specific time of day had more trades than others. The test was performed with all trading days of march 2018 for the PETR4 stock showed that, despite the usual impression that the beginning and end of days are busier in price fluctuations, in average, that's not true.

[6] https://joblib.readthedocs.io/en/latest/
[7] https://www.intel.com.br/content/www/br/pt/products/processors/core/i7-processors/i7-8750h.html

First a graph with all price changes per second was plotted to visually confirm the busier hours hypothesis, the Figure 14 showed a higher number of big price fluctuations in the beginning of the day, so a numerical check was performed.

Figure 14: Price fluctuation per second graph



For the numerical test, the day was divided in 4 equal periods of 1 hour and 45 minutes and both number of trades and price fluctuations were counted in every period.

Table 4: Price fluctuation and number of trades per day period

Day Period	Average number of trades (trades /day)	Average price fluctuation (R$/trade)
10:00 to 11:45	10710	0.00211
11:45 to 13:30	10914	0.00215
13:30 to 15:15	9657	0.00222
15:15 to 17:00	11240	0.00259

With the data in Table 4 it was possible to conclude that the average number of trades in every day period is pretty much the same, regarding price fluctuations, opposing to what the indicated, the price fluctuations and trades tend to happen more often in the end of the day, but only by 11.5% which is a relative small margin to consider time of day a relevant factor in the model.

3.1.2 Volume-Trades and Volume-Price Fluctuation Correlations

Regarding the volume-base the first test was measuring the average volume per trade made and what could be considered a big and a small trade regarding share volumes. The first result was that (for the 20 days tested) the average volume in trades is about 900 volumes which could be accounted as about R$20,000 per trade. This initial result was interesting but after checking the absolute amount of trades made it was possible to get to a very different conclusion.

Observing the graphs in Figure 15, Figure 16 and Table 5 is possible to see that almost 80% trades are below average volume traded which indicates that most trades are actually really low volume (over 50% trades are less than 300 volumes) and some absurdly high volume trades that happen quite rarely make the volume trade average to go much higher.

Table 5: Volume traded in quantiles of total trades

Average volume	Quantile
Bellow 100	32%
Bellow 300	54%
Bellow 900	77%
Bellow 2500	90%

Figure 15: Average trade volume in overall quantile

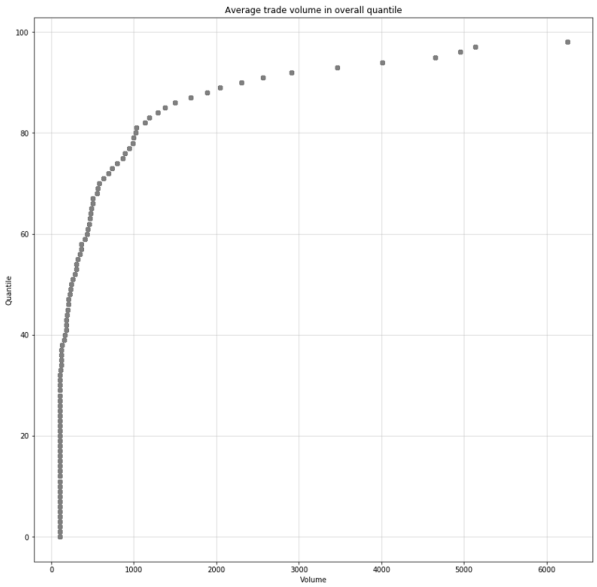
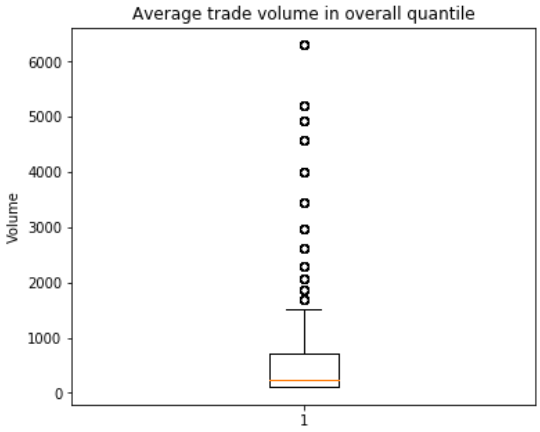


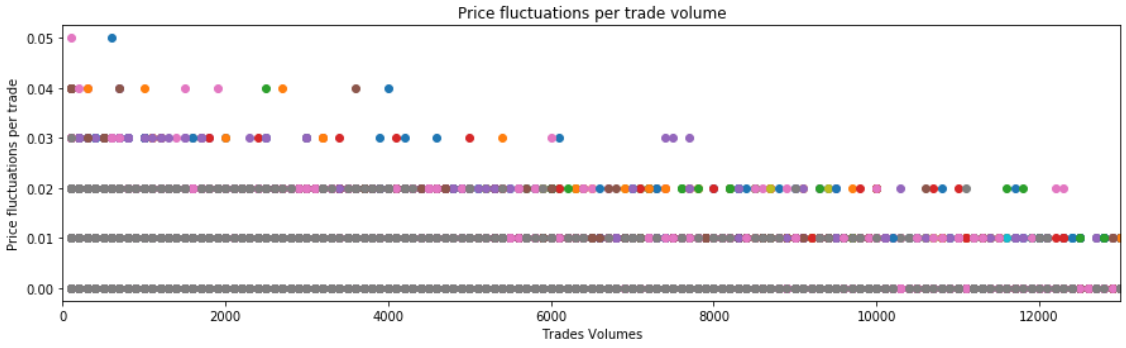
Figure 16: Box Plot of quantile of volumes



Is safe to assume that trades with over 5000 shares may be considered outliers since they area a fairly small part of the data being analyzed (less than 3%). This conclusion is not only important but it will also be used to justify the next graphs in this correlations search.

The next step was to try to see a price fluctuation-volume correlation, this time the results were surprising. With the graph in Figure 17 it became clear that the higher the volume in the trade the least the stock price seemed to vary while the BID or ASK created was still active.

Figure 17: Price fluctuation per trade volume



The same methods of dividing the tested value in groups was used, and considering that most trades had a stock volume of under 5000 the data was divided by thousands in 6 different groups. The results showed in Table 6 confirm what was observed in the graph, as the trades volume grows the price fluctuations seem to get smaller. In other words, there is an inversely proportional relation between volume traded and price fluctuations.

Table 6: Average price fluctuation per traded value

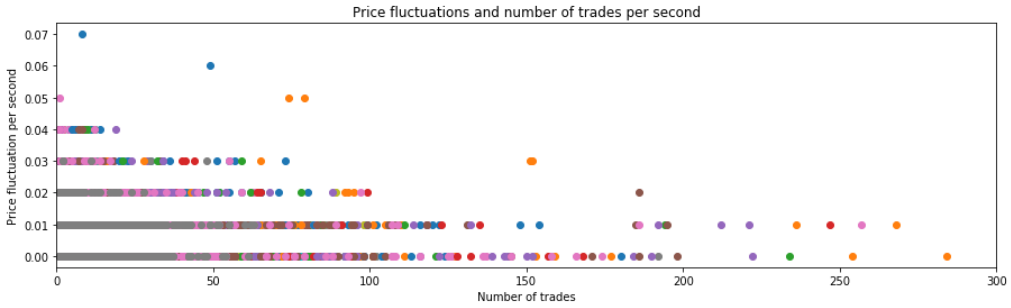
Trade Volume	Average traded value fluctuation (R$/trade)
Under 1000	0.00242
From 1000 to 2000	0.00193
From 2000 to 3000	0.00199
From 3000 to 4000	0.00188
From 4000 to 5000	0.00174
Over 5000	0.00113

3.1.3 Ticks-Price Fluctuation correlation

For the ticks-price fluctuations test a similar process to the volume analysis was used, but this time it was all summed according to time, in this analysis, to make the data more understandable the comparison is between number of trades and price change in a second instead of per trade, this way not only price fluctuations could be monitored but also the average number of trades per second.

The resulting graph shows a very similar result to the traded volume, in Figure 18 is possible to see that ticks have an inversely proportional correlation with price fluctuations as with a larger amount of trades per second the price seems to change less.

Figure 18: Price fluctuation per second Vs Number of trades per second



As for trade distribution during the day, the general result seems to show no real busier time. Again, to assure the visual results in Figure 19 the data was actually counted in previously defined day periods (1 hour and 45 minutes) resulting in Table 7. With this data it was then possible to confirm what was observed in the graph, there is no real correlation in the number of trades per second and any specific time of day.

Figure 19: Trades per second distribution over the day

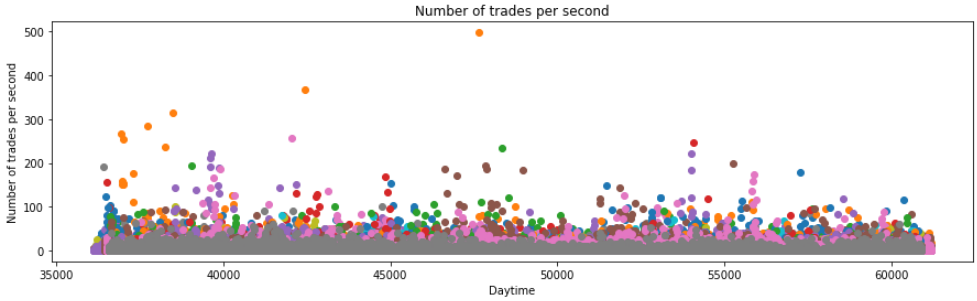


Table 7: Average number of trades per second per day period

Time of the day	Average number of trades per second
10:00 to 11:45	1.700
11:45 to 13:30	1.732
13:30 to 15:15	1.533
15:15 to 17:00	1.784

3.2 Moving Average Tests

Different moving averages were compared using the closest step possible. Since there was little difference between number of trades and volume traded during the day it was possible to establish a direct comparison between time and both volume and ticks. Figures 20 and 21 show how linear the relation between time and volume and time and ticks is, and how using a linear regression it was possible to estimate time, ticks and volume steps that would add up to the same relative value during the day.

Figure 20: Total volume traded per daytime in seconds

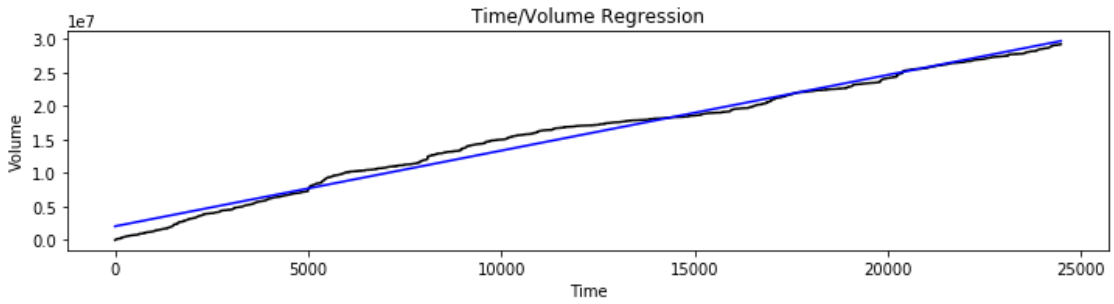
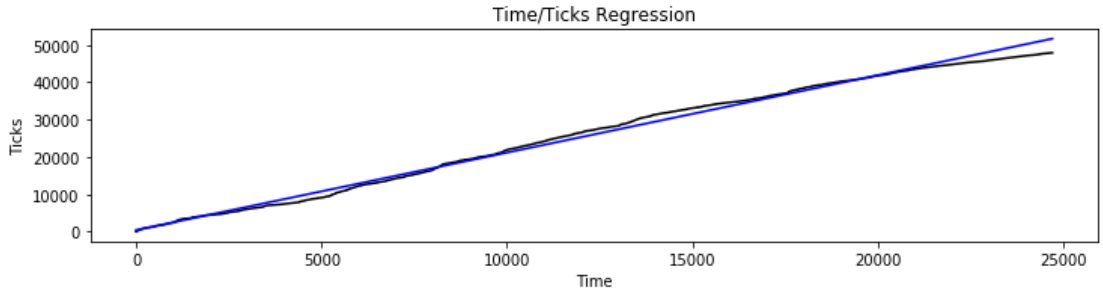


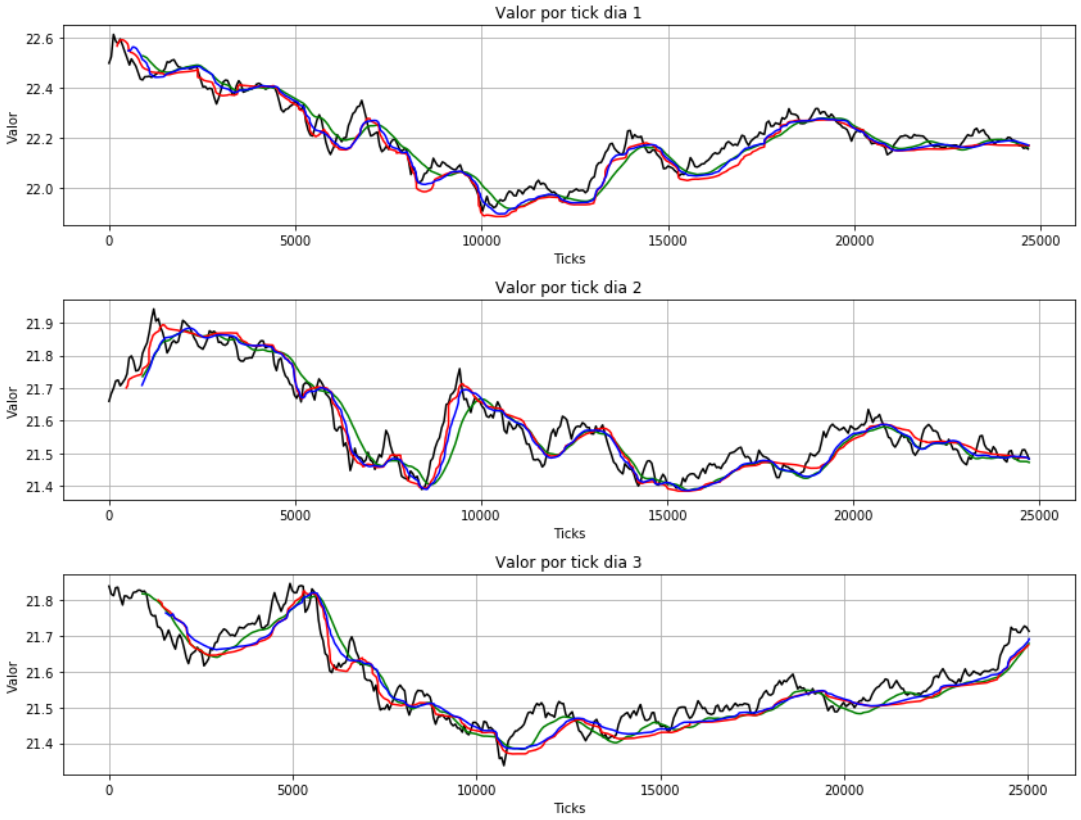
Figure 21: Total ticks passed per daytime in seconds



After the linear correlation was made the coefficients were an average of 1571 volumes traded per second with a root mean square error (RMSE) of 1,918,100 or 8.5% of the mean and 1.61 trades per second with a RMSE of 1136.6 or 5.6% of the mean. With this number it was possible to create 3 different moving averages with the same approximated step of 10 minutes. These ten minutes were equivalent to 600 seconds in the time moving average, 942,000 volumes in the volume moving average and 966 ticks in the tick moving average.

After this conversion process it was the possible to use these filters in the data and compare how they perform in a day with the same step. In Figures 22 and 23 the green line represents the time moving average; blue represents the ticks moving average and red the volume moving average.

Figure 22: Time, Volume and Tick moving average performance comparison



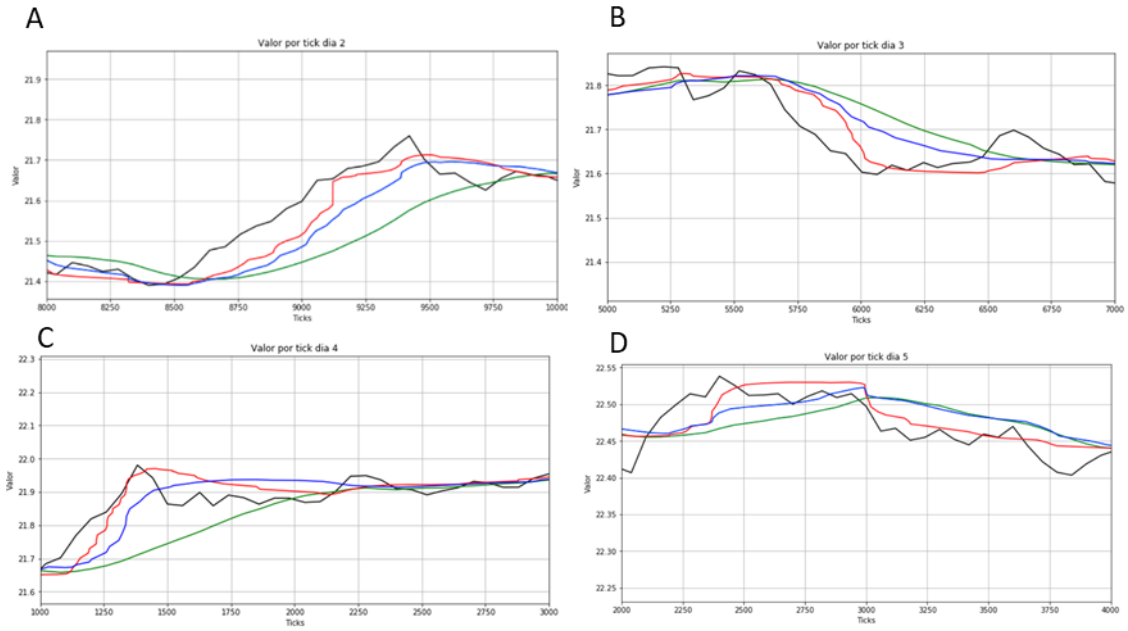
In Figure 22 is already possible to see that despite the moving averages having theoretically the same time step they respond very differently to the sudden changes in price in the market. The red line (volume moving average) seems to react the faster but also doesn't seem to filter as much noise from the sample. The blue line (tick moving average) seems to react reasonably well, being faster the green line (time moving average) while still removing most of the noise. Finally, the green line, seem to react the slowest taking a lot of time to adapt to sudden changes.

Besides, both the tick and volume moving averages would only react fast when there were enough trades happening, preventing strong buy or sell signals when the market did not have enough liquidity for higher volume trades. This is very important, especially on high frequency models, to avoid losing money to high value spread.

To have a better look at the graphs some zoomed images of sudden value changes are presented in Figure 23. In this figure is possible to confirm what was observed in the less zoomed images, in fact the volume moving average tend to adapt faster but actually seems to generate even more noise in the data as can be observed in Figures 23A and 23C. This is most likely because trades with to high volume tend to make the function quickly turn towards the price as can be seen especially in the mark of 9125 ticks in Figure 23A.

Regarding the ticks moving average, again, is possible to verify what was observed in Figure 22, as it follows the market price much faster than the time moving average it also seems to filter a lot more noise than the volume moving average. Finally, the time moving average show no surprises as it can filter the noise of the sample really well but lag behind the actual stock value too much.

Figure 23: Zoomed comparisons of moving average performance in steep price changes



With that conclusion it was then possible to proceed to testing all moving averages to see whether the change in variable is actually worth the risk of noise generation by using volume-based moving averages for instance.

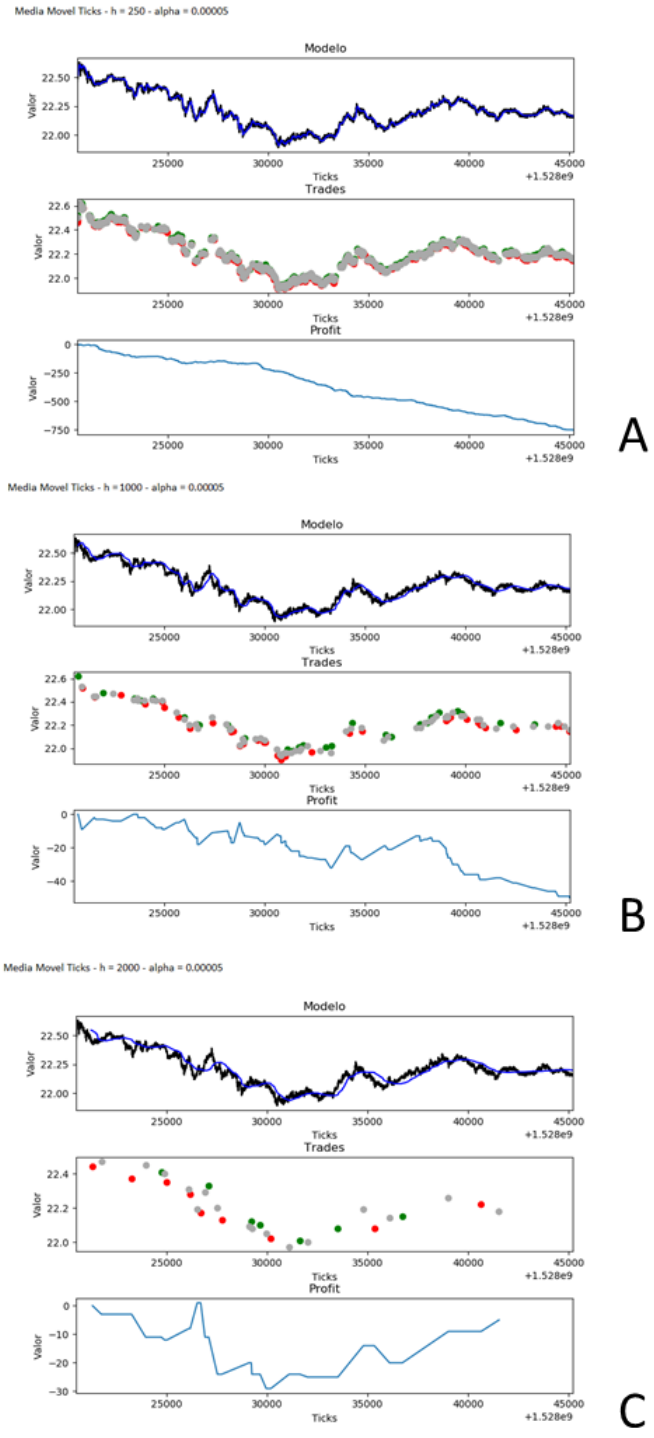
3.3 Model variables analysis

The difference in profitability of the model is completely related to the filter step h and the signal dead zone α. Yet, before showing the model results it is important to keep clear that for each filter the h means a different thing since the step is a measure of how long will the moving average act in the filter base. So, using a volume-based filter will have the h counting volumes traded, a time-based filter, seconds and a ticks-based filter, ticks.

The Figures 24A, 24B and 24C show results for three different steps of the tick-based filter with the derivatives-based trading algorithm performing in the same day. As the step grows, it is possible to observe the smoothing of the filter (dark blue line in each first graph) as the step goes up, at the same time it is also possible to observe a delay starting to appear between the filter and the real stock value (black line in the first graph).

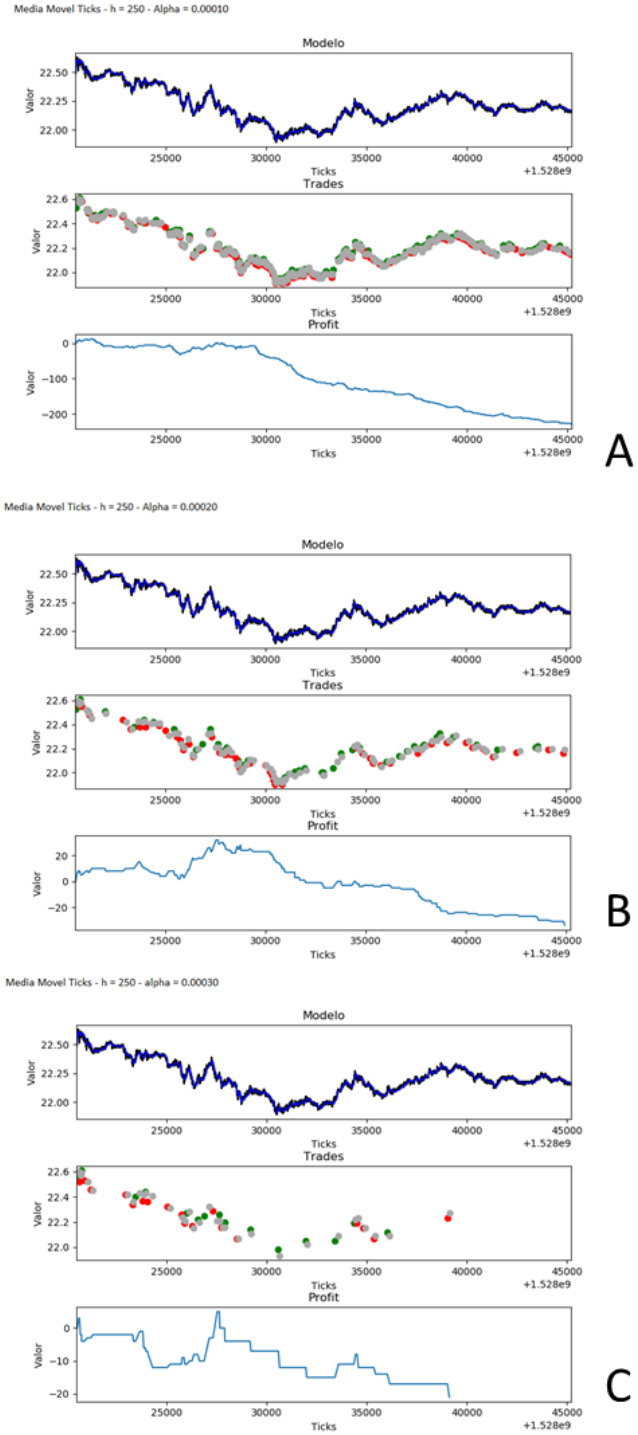
These two factors combine are both good and bad for the bot, the smoothing makes so the bot detects only the more abrupt changes in the market price and make less but more efficient trades, at the same time, if the filter makes the stock price too smooth, the bot will miss most of the important trades it could make. Besides, the increase in the step also increases delay, which makes the bot reacts much slower and therefore, buy in less optimal times as can be seen in Figure 24C.

Figure 24: Model h variable comparison A) h =250, B) h = 1000, C) h = 2000



For a second analysis the α variable was tested, the Figures 25A, 25B and 25C show results for three different α's for the tick-based filter (same as before) with the derivatives-based signal performing in the same day with constant step (250 ticks). This time the change doesn't influence the filter in any way but make so the bot only trade with a steeper derivative calculated by the model, this makes so the bot only trade on bigger price changes. By the images this effect is obvious as the bot in Figure 24B trades way less than the bot in the 25A one, and the bot in 25C doesn't even recognizes the last period of the day as a price changing period.

Figure 25: Model "α" variable comparison A) α =0.00010, B) α = 0. 00020, C) α = 0.00030



That being said, it becomes clear that there is an optimal value for both the step and the α, and that one would influence directly in the other. It was at this moment in the methodology section, that a bagging approach was decided upon.

3.4 Bagging

After the difficulty with the step and signal parameters trying bagging helped raise stability for the models, nevertheless it needed to have the range adjusted for how many, and how far would the clone filters and signals go.

Trying to go from half the selected value to two times the value with a total of 10 clones (11 models running together) seemed to work somewhat decent but further tests had to be done, and the improvements did not seem to pay off.

For the filter the main problem was that the inconsistency of steps would make it oscillate too much causing the signal to trigger too often making the model look undecisive losing allot of money to spread and taxes. As for the signal with bagging a similar problem happened, near the dead zones the winner of each election kept changing and the signal kept sending buy and sell orders too often, again, losing allot of money to spread and taxes.

Figures 26, 27, 28 and 29 show the difference between the normal, bagging filter, bagging signal, and both bagging, all models were tested with base h value of 1000 ticks per update and a base α of 0.0005.

Figure 26: Test with ticks moving average and derivatives signal

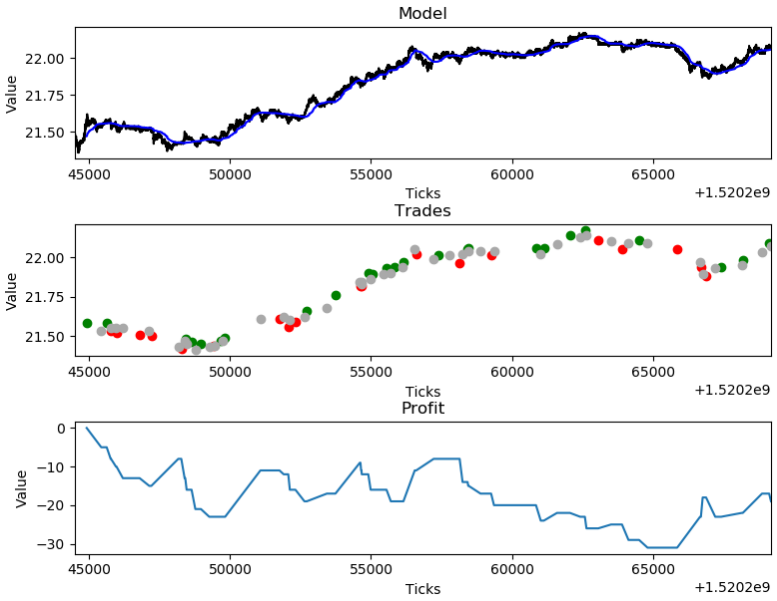


Figure 27: Test with ticks moving average with bagging and derivatives signal

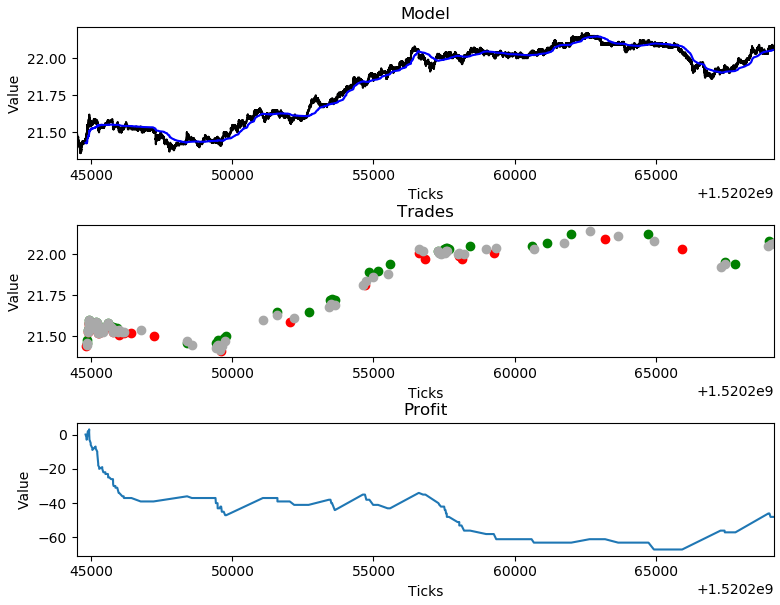


Figure 28: Test with ticks moving average and derivatives signal with bagging

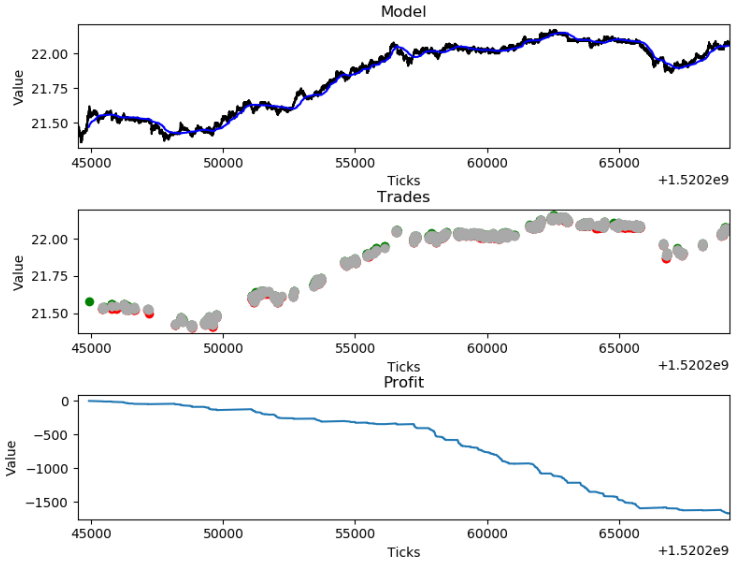
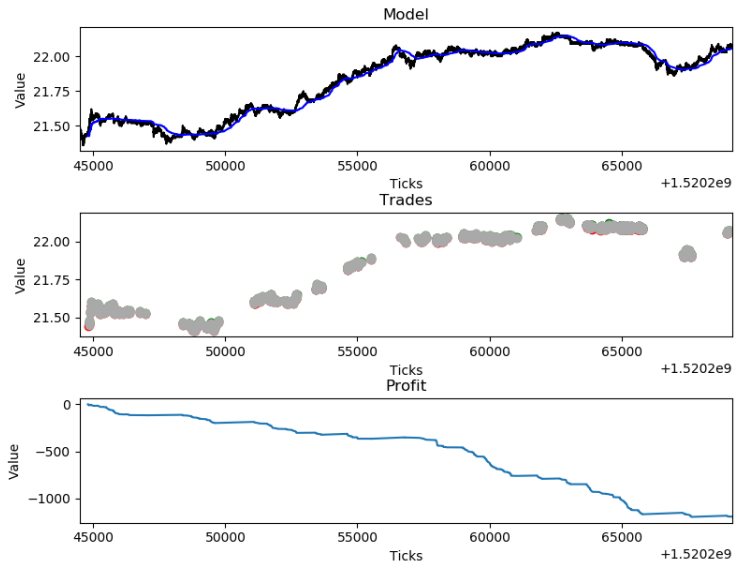


Figure 29: Test with ticks moving average with bagging and derivatives signal with bagging

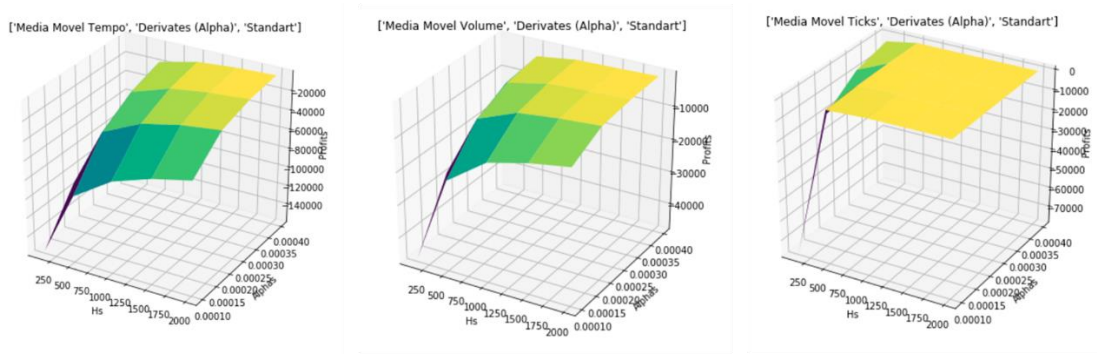


Due to the undesirable high frequency in trades the bagging approach was left aside, and the grid search approach was decided upon to find optimal parameters for the models. It is important to notice that a tweak of the parameters in the ranges could make them work better, but considering there was already many good options to test, it did not seem like the best approach.

3.5 Grid Search

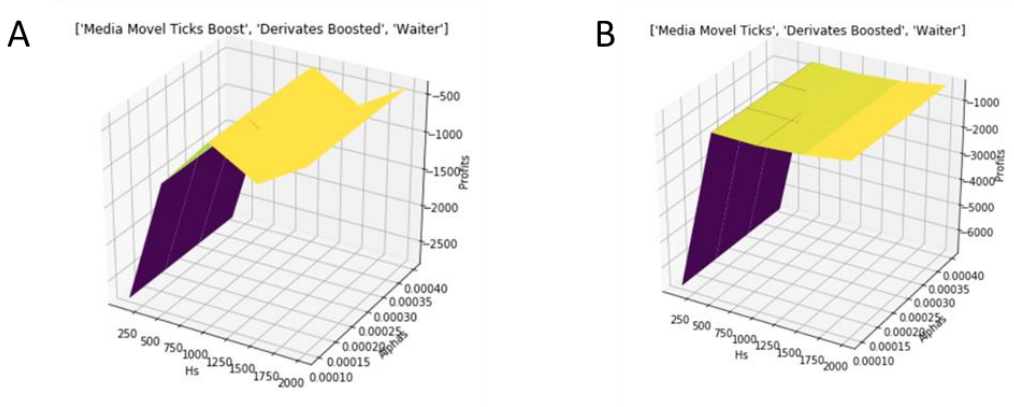
The results of the grid search were firstly interpreted as graphs, Figure 30 help to illustrate that for the 3 simplest models they tend to get increasingly better results with the raise of both α and h parameter. Unfortunately for those models the raise in both parameters actually make them lower and lower frequency to the point there were days with no trades. The graphs also showed that the time moving average needs a much lower frequency (higher α´s and h´s) to be stable while the ticks moving average seemed to be stable with smaller parameters which helped the model to preserve it higher frequency characteristic.

Figure 30: Grid search results for the 3 original moving averages with same Signal and Broker



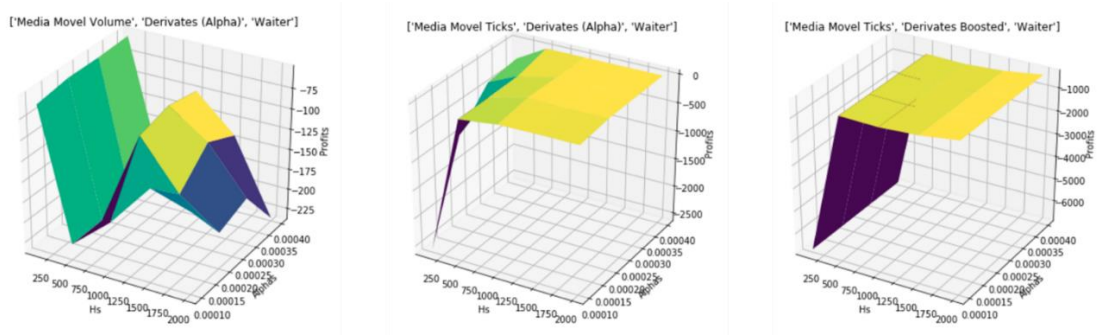
Another very interesting results was the bagging models graphs; they had a really high consistency over different α values which can explain why tweaking them was so difficult in first place. These results show clearly that while the bagging in the filter step made the model more stable (Figure 30 B) compared to the simple filter (Figure 30 A), the bagging range was way too big in the signal as it made so changes in the α parameter not change the model whatsoever.

Figure 31: Grid search results for 2 combinations of bagging models with the same filter



The final relevant graphic of the grid search is the comparison of three completely different models, the first one a volume-based moving average that looked very unstable over the variables tested, as observed before, it had great reaction times, but could not filter the noise well enough. The second, a standard tick moving average model, that had a much faster adaptation to market sudden changes and did filter the noise correctly. Finally, a model using the bagging technic that becomes much more stable over parameters change, but did not perform as well as some other models, probably because the wide range in the α variable.

Figure 32: Grid Search results for very different models for comparison



Besides the graphs, the most important result of the grid search was a table of results. Table 8 shows for every model tested the best h and α found, and how much the model earned considering both spread and taxes. This table not only shows that the best results were obtained with the Ticks Moving average it also shows how poorly the time moving average performed having the worst result for both the "standard broker" and the "waiter broker" without bagging models.

Table 8: Grid search compiled results

Filter (Moving Average)	Signal	Broker	α	h	Totals
Ticks Bagging	Derivatives (α)	Waiter	0.0003	500	16.73
Ticks	Derivatives (α)	Standard	0.0003	500	14.91
Ticks	Derivatives (α)	Waiter	0.0002	1000	0.0
Volume	Derivatives (α)	Waiter	0.0004	100	-51.64
Time	Derivatives (α)	Waiter	0.0004	1000	-97.35
Ticks	Derivatives Bagging	Waiter	0.0001	2000	-370.33
Ticks Bagging	Derivatives Bagging	Waiter	0.0001	2000	-382.57
Ticks Bagging	Derivatives (α)	Standard	0.0004	2000	-417.68
Volume	Derivatives (α)	Standard	0.0004	2000	-556.42
Time	Derivatives (α)	Standard	0.0004	2000	-2644.36

It was then possible to proceed to the final step of the project, the Backtesting to test whether the best models actually perform good, but before that, it is important to point out that some of the better results on the grid search were on the edge of the grid so most likely, increasing the search range would bring even better parameters for the model final test. The only disadvantage is that with the increase of range there is also an increase in computational time which was already very long (6 hours per every 5 days of simulation).

3.6 Backtesting

To validate the grid search results was necessary, so by using the entire dataset collected for the research (approximately 6 months) the tests were performed for the top three grid search models. The results weren't as good as expected; all tested models had profit if the taxes are disregarded but did not profit considering stock trading taxes.

In the backtesting the best model was actually the model that had the worst result between the grid search top three, by doing less more consistent trades it managed to trade over R$500,000 in R$2000 trades and only loose R$16.20 which is actually a quite impressive result. The second-best model was the top scoring in the grid search, it made 5 times more trades than the top scoring model and profited twice as much, but because of taxes again it had a considerable loss. The worst model of the three selected was actually really bad, the Broker not working against the spread really made a difference and the model was not even close to having profit, especially if taxes are accounted.

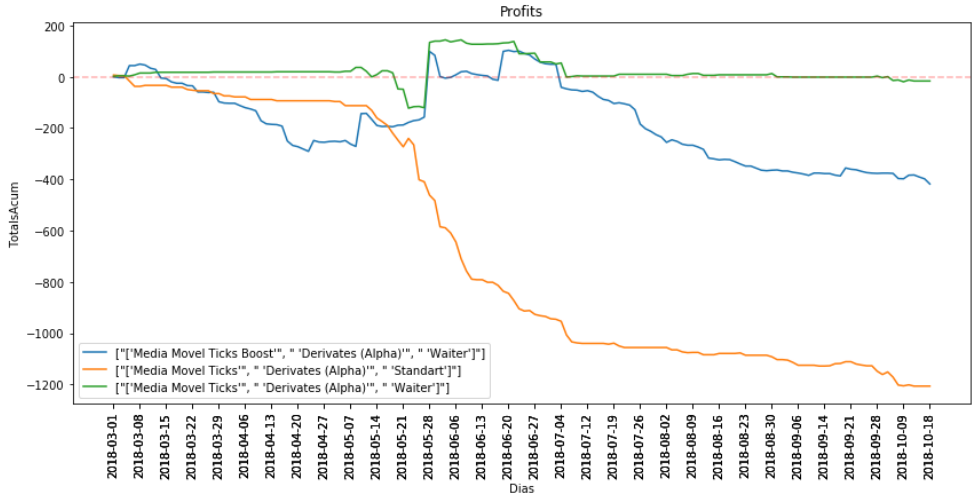
Table 9 is a summary of the results of the backtesting, and presents testing conditions, number of trades, profit without taxes and a total that includes taxes.

Table 9: Backtesting results for the best Grid Search models

Model	α	h	Trades	Profits	Totals
Moving Average Ticks Bagging, Derivatives (α), Waiter	0.0003	500	1250	212.00	-417.98
Moving Average Ticks, Derivatives (α), Standard	0.0003	500	944	-731.00	-1206.89
Moving Average Ticks, Derivatives (α), Waiter	0.0002	1000	268	119.00	-16.20

To check the daily performance of the models a graph of accumulated profit per day traded was created, Figure 33 presents this graph. In it, is possible to see that the best model mostly waited for good trades and had some quick raises, only to lose the accumulated money afterwards, the second best model (first in the grid search) had a much more erratic trading process, but still had a very good result in the month of July, but again lost all the profit afterward. Finally, the worst result can be seen as an actual bad result since it had almost every day with a considerable loss.

Figure 33: Accumulated model daily earnings for the Backtesting period



4 Conclusions

Despite the overall neutral/negative results of the models created most models could have profited in the stock market if some of the taxes were lower. Working with such a high frequency and so many trades the lower frequency taxes quickly start to add up reducing the model's overall profit. Even so, some of the developed models could be used to generate liquidity to stocks, since the high frequency models can trade huge amount of volumes with a fairly low cost, for instance, the second best model managed to loose only R$420 to do 1250 trades, buying and selling 125,000 volumes and moving an average of R$2,500,000 with only R$2000 trades.

Furthermore, the whole project resulted in a program that can be used to teach people about high frequency trading and maybe incentivize students or stock market aspirants to make their own models. The program for testing is available in a simpler version at: https://github.com/edufm

Other contribution of the project was finding correlations between volume and stock price, and ticks per second and stock price that can help to create unorthodox models that could perform well in the market. Overall, the whole study helps to break paradigms of how the market is usually looked at, proposing different variables to explore and a different than normal trading model.

5 References

1. TheStreet - Investment Funds - 20 Best Mutual Investment Funds - https://www.thestreet.com/topic/21421/top-rated-mutual-funds.html (Access in 02/15/2019)
2. US-News and World Reports – Mutual Funds Year Return - https://money.usnews.com/funds/mutual-funds/rankings/large-growth?sort=return1yr (Access in 02/15/2019)
3. Engle, R., "The Econometrics of Ultra High Frequency data", 1996, National Bureau of Economic Research.
4. Aldritch, I., "High Frequency Trading", 2013. 2a Edition, Wiley Trading Series.
5. Easley, D., and M. O'Hara, "Time and the Process of Security Price Adjustment", 1992, Journal of Finance, Volume 47, Pages 905–927.
6. Engle, R., and A. J. Patton, "Impacts of Trades in an Error-Correction Model of quote prices,", 2004, Journal of Financial Markets, Volume 7, Pages 1–25.
7. Schotman P., Frijns B. "Price Discovery in Tick Time", 2004, Journal of Empirical Finance Volume 16-5, December 2009, Pages 759-776
8. Géron A., "Hands-On Machine Learning with Scikit-Learn and TensorFlow" – 2017 – O'Reilly
9. Rogers, L., Zane, O. "Designing and estimating models of high-frequency data", 1998, University of Bath preprint
10. Engle, R., Sun Z., "Forecasting Volatility Using Tick by Tick Data", 2005, SSRN Electronic Journal
11. Palágyi Z., Mantegna R., "Empirical investigation of stock price dynamics in an emerging market", 1999, Physica A: Statistical Mechanics and its Applications Volume 269-1, Pages 132-139.
12. Amihud Y.,"Illiquidity and stock returns: cross-section and time-series effects", 2002, Journal of Financial Markets Volume 5, Pages 31-56
13. Lesmond D., "Liquidity of emerging markets", 2005, Journal of Financial Economics Volume 77, Pages 411-452
14. Liu W., "A liquidity-augmented capital asset pricing model", 2006, Article in Journal of Financial Economics Volume 86, Pages 631-671
15. Sparks, Evan & Talwalkar, Ameet & J. Franklin, Michael & Jordan, Michael & Kraska, Tim, "An Efficient Planner for Large-scale Predictive Analytic Queries", 2015