**Analysis of several different methods for trading Tesla stocks with a focus on natural language processing techniques applied to twitter data.**

**Research report**

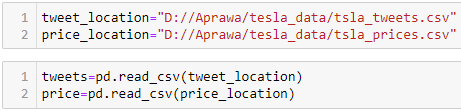
Author:

**Eilder Jorge García**

# Introduction

# Initial operations and data preparation

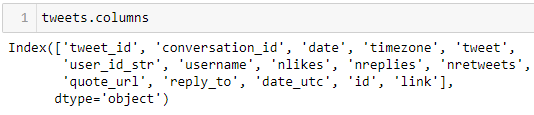
The first step is obtaining the data, since in this case it was already available in .csv format and it’s relatively small I can just load it in a dataframe.



Next is cleaning up the data. In the case of price there isn’t much to do except just check if there are missing values or any other weird problem with the data. After checking the data I conclude it’s clean and I only need to convert the dates to a date time object.



Cleaning up the tweets is a two-fold operation. The first part is cleaning up columns that aren’t of interest:



Date is redundant with date\_utc, so I’ll keep date\_utc only. Timezone is also redundant as date\_utc already accounts for it.

tweet\_id, link, and id are also redundant as we are interested in the content of the tweets and not in replying or operating with those tweets in any fashion.

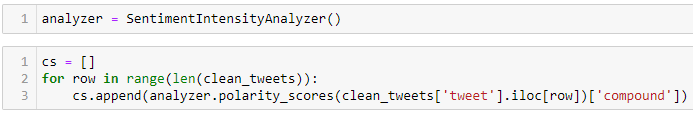
Conversation\_id, reply\_to and quote\_url could be interesting if we are going to be doing extended social analysis network but the cost of doing such an analysis and the relation to the tweets is not practical in real time operations.

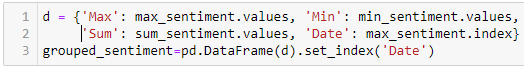
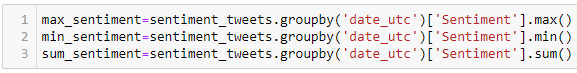
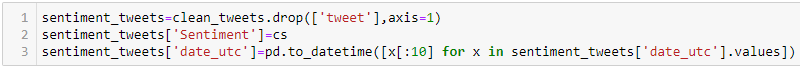
Finally user\_id\_str and username could be useful if we knew the ids of influential figures beforehand as we could assign a bigger weight to their tweets, but in this case we’ll assume everyone is equally popular and judge by number of likes, retweets, and the sentiment of their tweet.

This leaves us with tweet text, the number of likes, the number of replies, the number of retweets and the date in UTC.

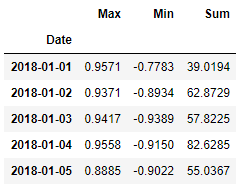
We then clean the tweets that are missing values and proceed to the 2nd part.

At this part we are interested in extracting a score of sentiment for each sentiment and how they affect TSLA stock. In a first iteration the idea is to use the VADER library in the NLTK package to extract the compound sentiment score for each tweet. We then select the maximum and minimum compound score per day, as well as the sum over the day and prepare a table with the scores for each day, this table is what’s going to guide our trading signals.





At this point we got a table with our overall sentiment each day, as well as our most positive and most negative sentiment.



A quick look shows that while the most positive and most negative sentiments are similar, the overall total is strongly positive.

We combine this sentiment data with our price data, removing tweets that lie in non-trading days for this first iteration.

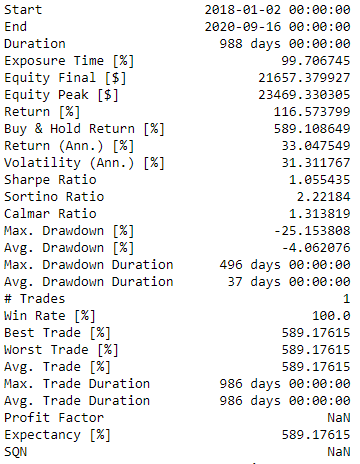
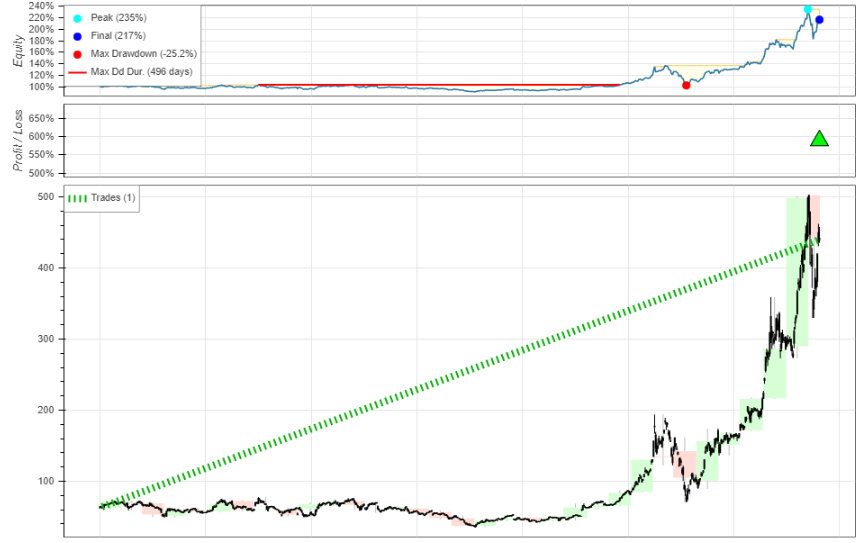


We set the date as index because we need it to be set as an index for the backtesting library.

At this point all our data is together and we can test some money management and trade control operations.

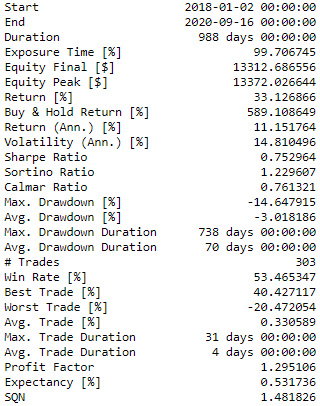
# First iteration: Using sentiment without modification

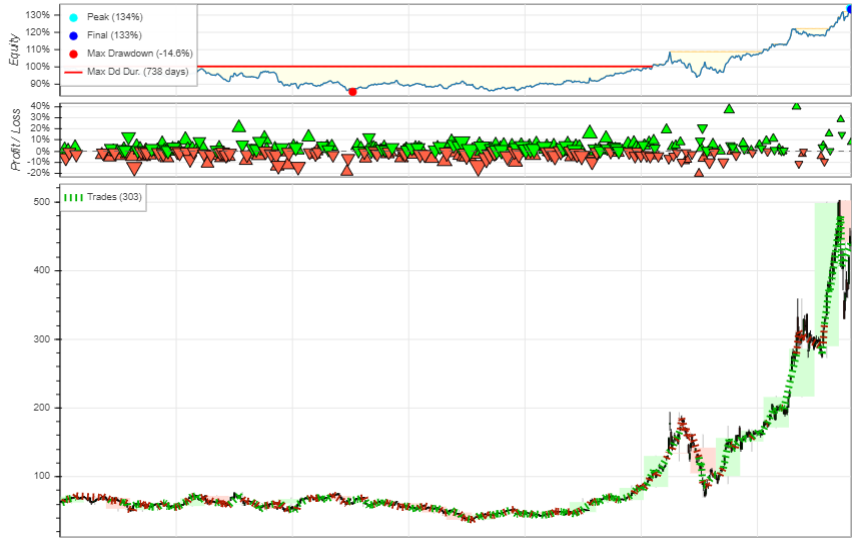
We start with a very simple idea, if our overall sentiment is positive we buy, otherwise we sell, and this in turn gives us the following result:

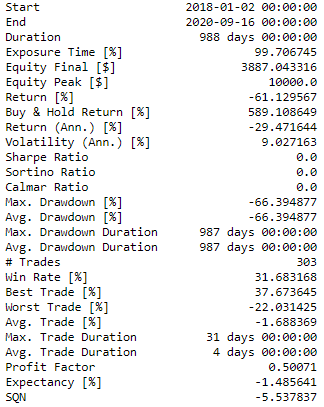
This means that the overall tweet sentiment was positive during the entire life of the stock, indicating to buy & hold.

While this could be an exception of TSLA and for other stocks we could have more variety of sentiment, we can also try comparing the maximum and minimum sentiment only, and this in turn gives significantly more trades:





The overall win rate is good too, but there is a catch: “commission”. Once we add a 2% commission for trading the stock (not to mention possible slippage) our profit plummets:



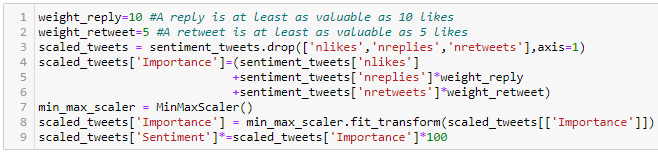
# Quick points we can explore and improve upon:

1. We can experiment with weights using the number of likes, retweets and replies.
2. We can experiment with the sentiment analyzer lexicon, adding more words and tuning weights for existing words, or trying another sentiment analyzer.
3. We can test on other price data that isn’t extremely bullish to check if the overall tweet sentiment is positive, and if we can confirm that, on average, tweet data is positive, apply a bigger multiplier to negative tweets to offset the positive ones and create some equilibrium.
4. We can experiment with better money management techniques, like trailing stops or incremental positions.
5. We can experiment combining our sentiment with technical indicators.
6. We can experiment combining our sentiment with other ML models such as RNN or random forest ensembles.
7. We can experiment adding the weekend sentiment onto the Monday candle.
8. We can experiment with weights using the username, but this is not scalable to other markets or even other news sources.
9. We can add market specific data, such as the beta of the stock, their fundamental data (such as earnings, etc…), TICK data, compare it with other similar stocks in the sector or aggregate their data, etc…

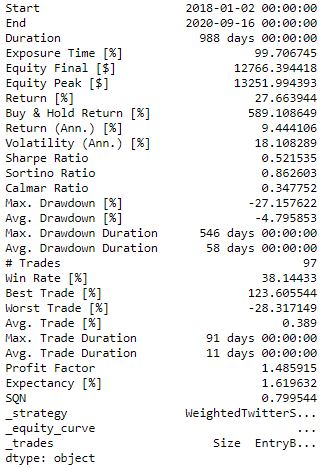
# Second iteration: Using weighted sentiment based on likes/retweets/replies

On a second iteration we can experiment with multiplying the compound score based on the number of likes, retweets and replies, as we assume more popular tweets have more impact on the underlying price.

Based on the small sample available I estimated that a retweet is roughly 5 times more important than a like and that a reply is roughly 10 times more important in determining the intensity of the tweet. Using these weights and multiplying this with our compound sentiment score:



Using a simple “buy if sentiment is positive and sell if sentiment is negative” strategy:



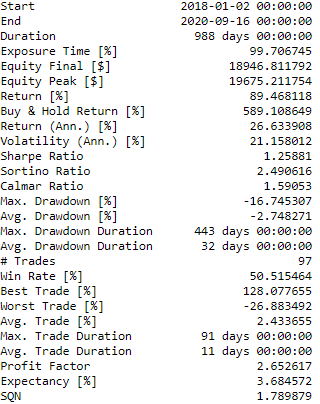
Considering that we’re paying 2% commission and that we manage to end up with positive equity I think it’s on overall pretty decent.



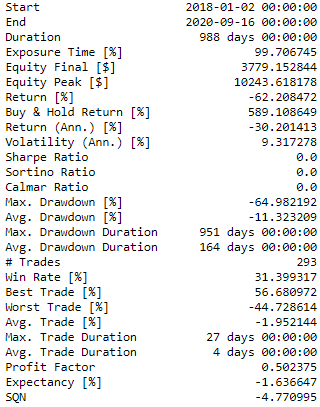
We notice that before the stock went parabolic there was indecision in the trades, mostly costing us money due to commissions, but afterwards the overall sentiment was very positive and the algorithm fully traded it, with a small short trade during a bearish moment:



Removing commission gives us a win rate slightly above 50% but a much higher final equity:



Now, how does this change affect the maximum over minimum sentiment strategy that we had before?

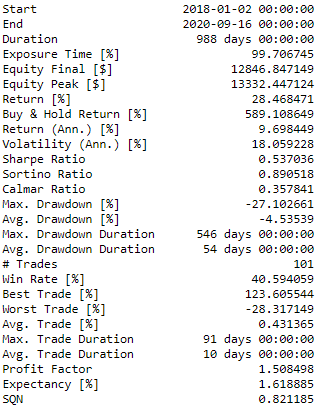


It got worse, and we can see some very drastic short trades while the stock is going up:



At this point it’s very safe to conclude that using the maximum and minimum of the sentiment in a day isn’t nearly as helpful as obtaining the overall sentiment, so no more research will be done using those two.

A grid search for the optimal weights returned 19 for replies and 4 for retweets, as those values are probably overfitting to the data, I’m going to pick 20 and 5 for the following iterations. A quick test using these weights improves very slightly the sum of sentiment model.

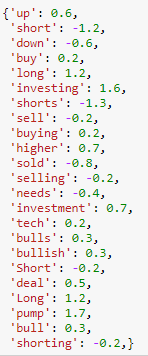


# Third iteration: Lexicon expansion

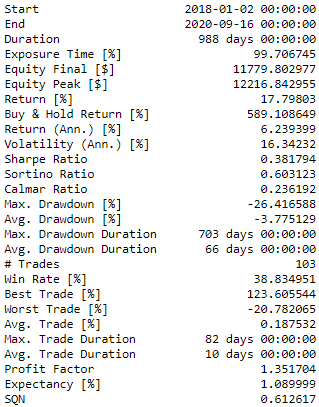
Research on publicly available sentiment dictionaries to replace VADER’s original one doesn’t provide any upgrade. VADER has 7502 words in its dictionary, while most sentiment lexicons have at most 500.

A first step would be to search for common words in our sentences that aren’t in VADER’s lexicon and add them in manually.

There are 825 common words that aren’t part of VADER, many of them are objective neutral words and thus won’t influence sentiment in any way. A small extension to the lexicon will be done with the following words:



The extension of the lexicon wasn’t exactly great as we can observe:



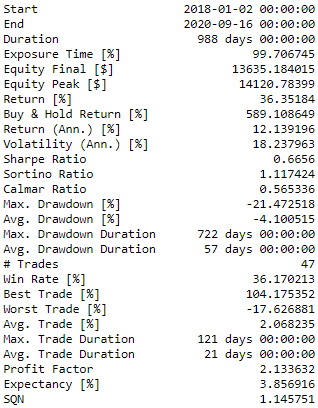
Manually fine tuning the weights or modifying other words could probably improve the results but this proves that just adding missing words won’t influence the final results much when you’re reading hundreds of thousands of words. Possible alternatives would include generating our own sentiment lexicon (a non-trivial matter) or do a massive grid search of optimal weights (an extremely expensive computational task that is likely to overfit).

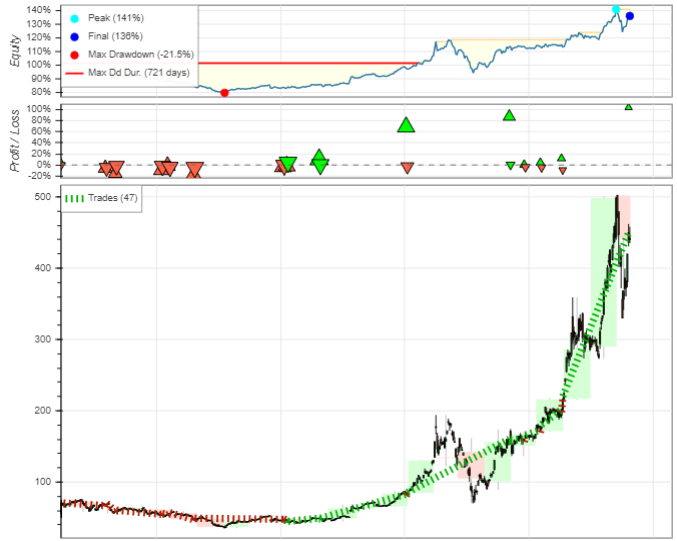
# Fourth Iteration: Using other sentiment analyzers.

There are two approaches: the first one is using sentiment lexicon that work on a variety of text like VADER, the other approach is to build a supervised learning model, but this would need to first label the tweets as positive or negative. As we don’t have the label for each tweet the only way is to use a generic sentiment analyzer.

## SentiWordnet

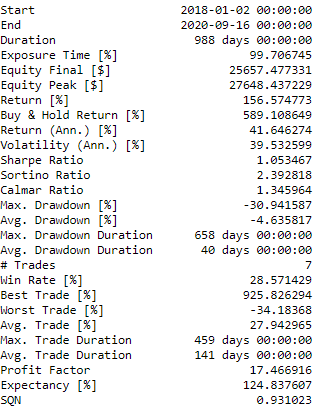
SentiWordnet is extremely slow and in messy twitter data it gives a lot of neutrals, as a result we end up with significantly less trades, but slightly better performance:





## Textblob

Textblob is also widely used for NLP and sentiment analysis. It’s simple to use and gives decent results. We can see here that it makes 7 trades, with the big ones making a huge impact on the profit.

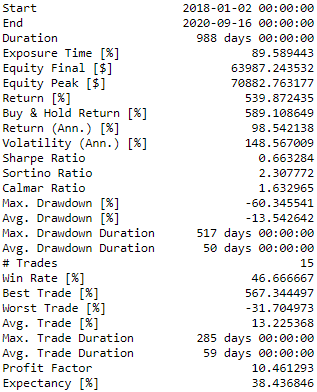
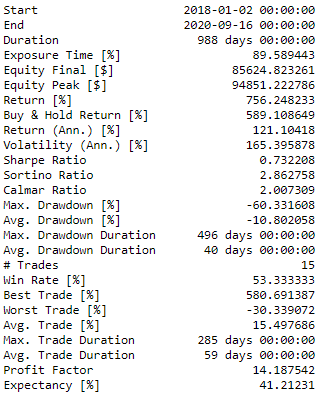




The biggest demerit of textblob is the huge amount of neutrals. As such we end up with very little trades. To try another approach I changed the conditions for opening and closing trades based on the intensity of the sentiment, and not only on if being positive or not. I also decided to close trades when there wasn’t a prevalent sentiment and we were mostly flat.

This doubles the number of trades although we also end up paying more in commission. In a commission free world where we use all our equity we even beat the buy & hold approach.

2% commission per trade No commission per trade.

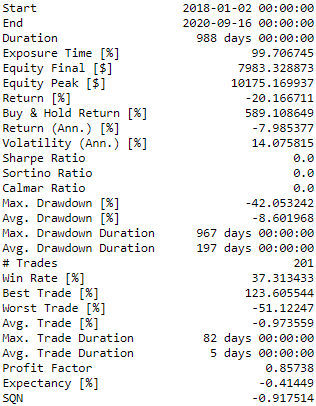


We can observe that while the general idea is good, it could use some fine tuning, specially on closing and entering, this is the same issue that VADER and Sentinet had.

As such the core improvement will surely be based on polishing those entry and exit conditions.

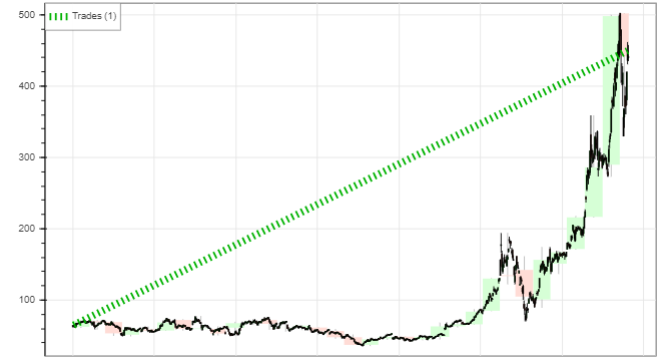
## Afinn

Afinn is the final library for sentiment analysis I’m going to try in this project. It’s considerably slow (on par with sentiwordnet) and doesn’t really perform too well on messy social media data like tweets. We can confirm that using our standard test bed it doesn’t really perform too well, although it can improve a bit tuning the entry conditions to only do trades when there is a substantial amount of sentiment, but still below Textblob.

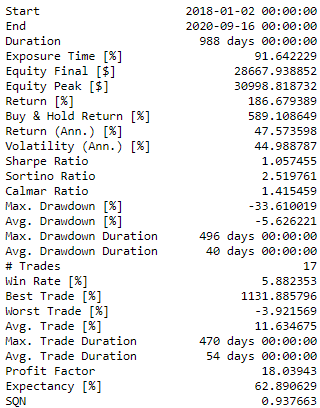
# Ensemble system, combining multiple sentiment analyzers

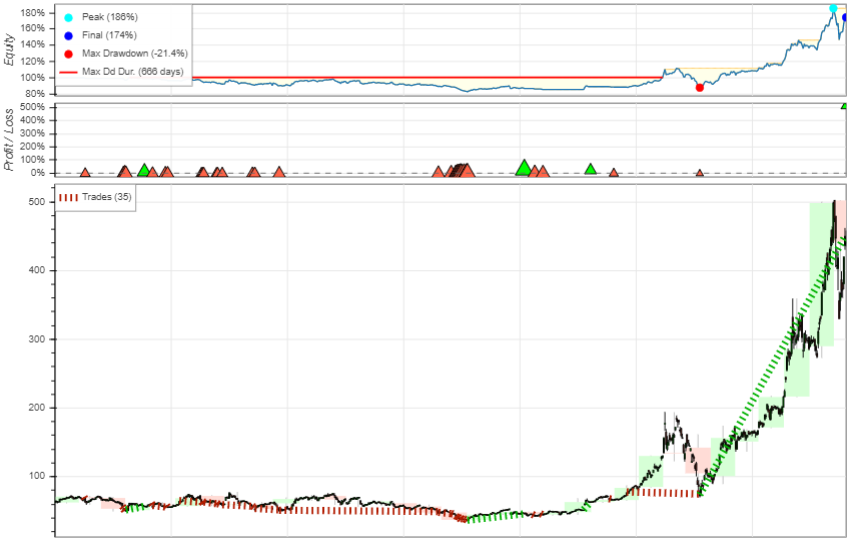
A final idea would be to combine the analysis of the different analyzers and scale them, so none of them is more important than the others. Doing this process gives us a very simple buy & hold strategy again:



This is a very interesting conclusion, and it would certainly be interesting to apply the same system to a company that has gone bankrupt instead.

However, while this could just be the case of TSLA tweets being extremely positive, we could adjust a bit our entries as we have done before, and add some stop loss for good measure as well. By adding a stop loss at 2% price variation and buying or shorting only on a significant sentiment change we can achieve a strategy with very low win rate (thanks to the stop loss) but very high P/L.





# Conclusion

All four sentiment analysis tools give a pretty good idea of general direction, even Afinn. It’s safe to conclude that the information obtained from sentiment analysis can be a powerful tool for deciding our trades, but it needs to be fine-tuned in order to maximize profits with better entries and exits.

Vader is solid overall and works fast.

SentiwordNet is very good and gives a balanced amount of sentiment information but it’s very slow.

Textblob gave us the best results, but the very small amount of trades could lead to overfitting or even coincidence.

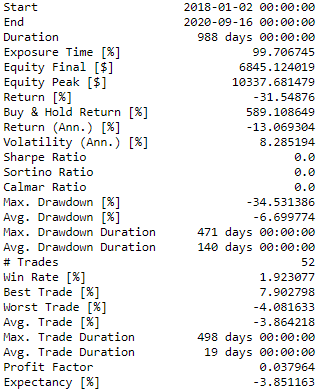
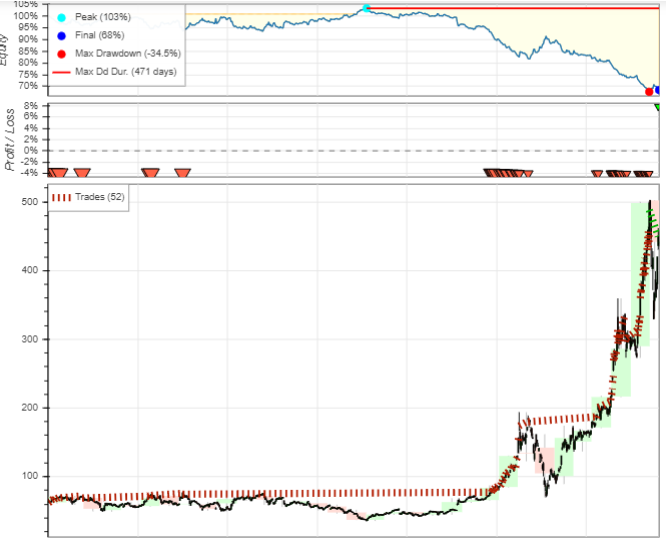
Afinn was disappointing but modifying the conditions for opening the trades themselves it does give decent results.

Combining them all into an ensemble is a very powerful idea, which combined with some basic money management concepts provided impressive returns, even with a 2% commission per trade.

There is more that can be done in the field of sentiment analysis, like using Deep neural network systems or supervised learning systems, but lexicon approaches gave good results overall.

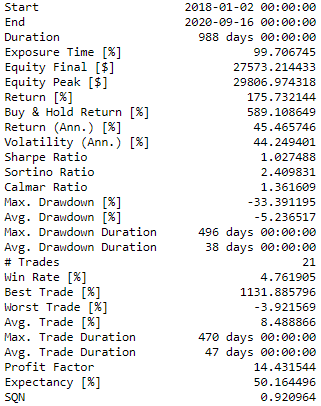
# Fifth iteration: Adding negative bias to tweets to try and compensate the fact that tweets are often positive.

Adding a small bias results in almost no change, while adding a significant bias just gives horrible results overall. It’s not worth trying to account for social media bias directly by adding weights manually to the sentiment.

# Sixth iteration. Amplifying overall sentiment based on activity

Just like on the 2nd iteration which gave good results overall, it’s of interest to amplify the overall sentiment at a given candle based on tweet activity: more tweets = more activity which should imply a bigger impact on price in either direction. Amplifying the sentiment based on the amount of tweets didn’t really affect our trades. This is due to the fact that if the overall sentiment is positive, having more tweets just means more positive sentiment, which in this case it just means that we can tune our entries based on smaller or bigger thresholds.

# Seventh iteration: Establishing more complex money management techniques.

Due to the small sample of data this is very hard to really test, but I’m going to adjust the ensemble system and try with varying stop losses and trailing stops, as well as establishing take profit values.

Multiple tests were made with stop losses values, trailing stops, take profits, incremental orders and varying the price delta or establishing windows of aggressive SL, amongst others.

The results are as following:

Too tight of a stop loss will create many losses, it’s better to have a wide margin of stop loss and use a smaller amount of money instead.

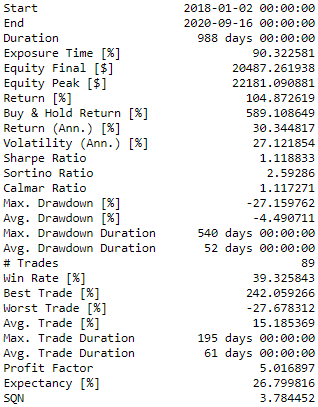
Margin can, and often is lethal and should only be considered for diversification.

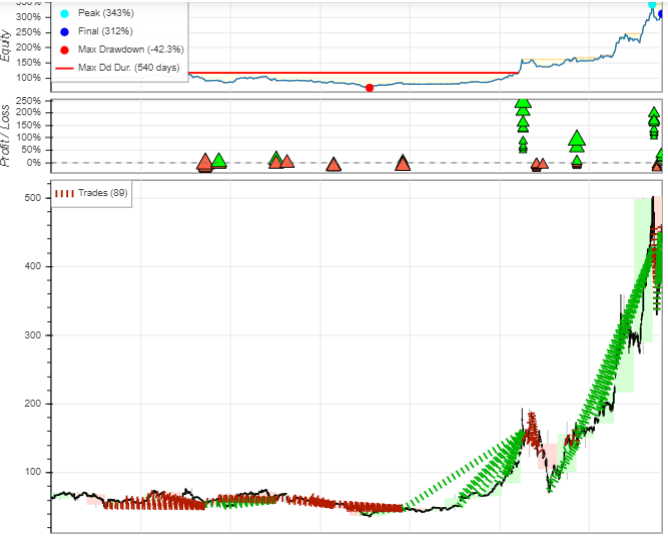
In general stop losses can reduce your maximum P/L in backtesting, but that’s mostly due to survivorship bias of data. Trailing stop losses can also help protect your profit once you’re above break-even.

Take profits are in general detrimental, you can backtest and select optimal value for TP that magically hit some daily highs, but in general it just limits the profit of your trades and forces you to pay more commission. It can see more value during intraday trading, however.

Modifying the sentiment entry trigger doesn’t have much of an impact, that’s because the overall sentiment is incredibly positive during the entire time series. Trying to close trades when there is little sentiment fails, and trying to nitpick shorts based on low (but still positive) sentiment fails as well and it shouldn’t really be done in the first place.

Incremental orders are interesting, using a fixed lot amount instead of variable equity and up to a fixed amount of risk it gives decent results at mitigating the drawdown.





## Conclusion

The traditional money management methods had a big impact on the models at first, during the previous iterations, but focusing completely on take profit, stop losses, trailing stops, etc… barely managed to diminish the drawdown and in general didn’t help much in the trades, as it was only longs, even when it would have been better to not trade at all.

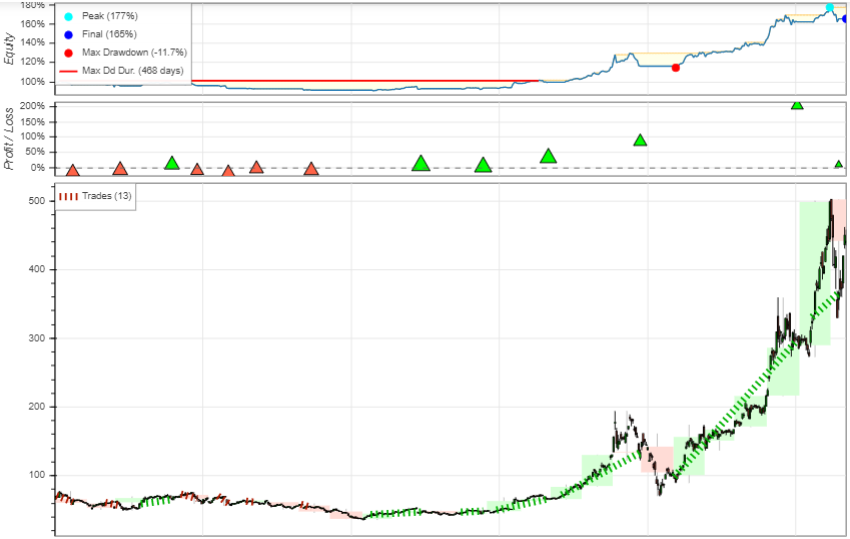
# Eight Iteration: Sentiment + Technical indicator trading system

Technical indicators and trading methods based on them have been around for decades, they’re simple to test and simple to execute, and can give an unbiased representation of information contained on the price.

As such, it’s a logical idea to extend our information of overall direction and sentiment with the underlying price structure and try to search for better entry points, and perhaps, acceptable exit points.

Using the traditional moving averages method of a fast (10 candle) over a slow (20 candle) weighted moving average for firing the trades and using the Bollinger bands for stop loss instead of a fixed stop gives us less profit due to less big trades but much less drawdown and risk:





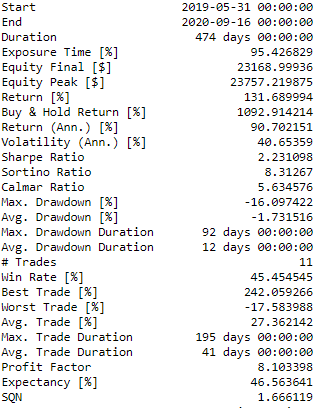
While there are many technical analysis methods we could combine with our sentiment to try and open trades only at optimal moments this process is very slow and prone to mistakes or overfitting. It’s an iterative process that can take a considerable time to reach a decent result.

# Ninth iteration: Using a classifier for automatic TA analysis

Using classifiers for these kind of operations is best done with plenty of data (up in the hundreds of thousands), but we can do a quick test with a very solid overall algorithm called random forest and set a big number of trees for automatic feature selection and classification. The approach is done using the next candle data and the next week’s candle data,

The first step is separating the data into train and test, the random forest has no memory or time lag so we just feed it the whole first 200 rows of data with the TA indicators and the price data. We then use it later with our sentiment and define the following rule:

“If our sentiment is positive and we predict a next day positive candle and a next week positive candle” then we open a long position, and the opposite for a short one. We add stop loss and a trailing stop loss and the resulting test is shown below:





The results are very amazing with a sharpe ratio of 2.23, although we could surely do better if we could train the classifier on much more data first and extend our test. We barely have a year of half of test data. Other classifiers could also help mitigate the errors we have on the price falls, so we could build an ensemble of those.

# Tenth iteration: Using a RNN with LSTM for price prediction.