r/wallstreetbets Trading Assistant Final Report

Chris Burson
Khaled Ibrahim
Bilal Mawji
Peter Rendina
Manuel Thomas

1 INTRODUCTION - MOTIVATION

Investing in the stock market is a critical practice required to maintain capital through inflation, though numerous individuals and organizations attempt to beat the market as a source of income. It requires experience, skill, and timeliness to navigate through the volume of data generated around the globe and around the clock to determine the direction an equity price will go. Large institutional investors have been incorporating analytical tools in their investment decisions, networks of people to focus on certain equity categories, and geography to be as close to exchange servers as possible. Retail investors of all experience levels are left at an extreme disadvantage without the computational edge of firms with buying power inside and outside of the global financial markets.

We created a prototype trading assistant designed for retail investors to present only equities at a favorable price in a timely manner from both price prediction and social media chatter. Thus providing the investor an cybernetic edge in equity markets.

2 PROBLEM DEFINITION

How to distill the volume of distributed data generated daily into actionable recommendations that are presented to a non-expert user in a timely, simple, and concise manner? To break down this big data problem: Gather equity indicator data efficiently, predict equity price response from indicator data, generate timely equity recommendations, Prevent overwhelming the user with too much information, Provide tools for a user to take action through interactive visualization, Feedback their actions.

3 SURVEY

Predicting future stock prices has been the focus of researchers and investment practitioners for the last few decades. The two main analytical approaches rely on either the Efficient Market Hypothesis (EMH) or the Adaptive Market Hypothesis (AMH). On the one hand, EMH's proponents subscribe to the view that stock prices follow a random walk and cannot be predicted and rely on fundamental analysis for trading decisions [3]. On the other, AMH's proponents believe stock prices are predictable and rely on technical analysis for trading decisions [6].

Studies have shown that the best performing technical indicators are those that included volume and price data [7]. Combining approaches such as textual analysis of company news with technical indicators yielded better results compared to a single approach [8]. Sentiment analysis of Twitter data showed a strong correlation between public sentiment in tweets and the direction of a stock price [5]. Public sentiment is the collective sentiment of the universe of individual investors. The risk tolerance or aversion levels of these investors are affected by their moods, which in turn is impacted by several external factors such as the news [4]. Using machine learning to analyse unstructured data, such as social media and news, coupled with structured stock price and volume data is an effective indicator of a stock price movement over a day [1]. However, to provide such capabilities for retail day traders requires processing and storing big data sets using a big data framework [2].

4 PROPOSED METHOD

Combining technical analysis trading suite's price, volume, and indicator charting with a light targeted queue of stocks predicted to rise in value from quantitative and qualitative social media data can assist novice investors in building a worthwhile portfolio while educating them in trading at a higher level. More experienced users can use the assistant to build portfolios bootstrapped with their own, their community's, and mathematical opinion

4.1 Intuition

The intuition behind our approach is to cut through the noise of distributed equity data by filtering the chatter of the noisiest controversial investment advice forum. Using the feed as both a gate way for stock selection and the volume data as a price prediction indicator with other traditional techniques taught in algorithmic trading courses, a queue can be streamed to a user to better focus on what is actually ready to make a profitable move up instead of holding sideways or even down trending securities.

The web application features three sections: A time series chart with relevant technical indicators, a sentiment analysis block with paper trading portfolio and a decision block where the action buttons are located.

In order to deliver this application, we broke it down to front and back end. The front end deals with the visual interactivity with the user, namely:

- Feed daily stocks to the user.
- Enable an environment which the user can make an educated agreement or disagreement with recommendations.
- Provide brokerage elements for portfolio management and progress tracking.

The back end deals with the analytical aspects:

- Streaming securities and social media.
- Data storage and processing.
- Modelling, prediction, and user results.

4.2 Description of Approaches

4.2.1 Revisiting the objective of the application. Our intention was to design an application that uses live data and presents the user with an ordered queue of equities, and broad social media analysis. Limitations mostly was driven by cost, Twitter did not grant us an academic API, exchange APIs at high resolution are expensive, and there was a major concern about running out of AWS credit. These concerns pulled the rug from under the initial plan, after studying several approaches, we opted to develop a simulated exercise that takes the user through a couple days of paper trading with similar data and insights. Our intuition was that such approach was still empowering to a non-expert user to embark on investing confidently in the stock market without a steep learning curve, and can later be applied to a live application as described above.

The simulated trading game takes place at a date range in the past. The user will be presented with a queue of stocks to take action on: buy, sell, or skip. Stocks in the SP500, mentioned in r/wallstreetbets submissions or comments for the day are eligible for trade if the stocks in the queue will contain stocks that have a predicted minimum percent increase threshold of 3% over a 5 day holding period. Prediction above the threshold are presented, prediction below the negative threshold and held by the user are also presented, and prices between are only accessible if the user chooses to sell from their portfolio. The user may move on to the next day until the end of the game for final results versus the market.

4.2.2 Data Collection. We chose to focus on stocks in the S&P500 to be able to simplify an already large data set considering social sentiment and compare results to the culmination of the index. We opted for using daily prices retrieved from Yahoo Finance, which has a python library named yfinance. We downloaded 10 years worth of daily prices to train and test our predictive model. This generated a data set with 1.3 million records.

The Reddit API developed and maintained by Jason Baumgartner: (PushShift Reddit API) allows for key-less date range filtering on subreddit submissions and comments, sorted by score, and currently limited at 100 instances. It would have been really helpful if the aggregation functionality was maintained as a much large count could have been gathered per day on ticker and company name mentions from the beginning January 31st, 2012 to the current date. The key-less access required clever timing between calls, processing, and IO. Ultimately the initial download took over 2 hours to run. Text returns were matched to contain the company name or a set of common ticker references, which

meant throwing out the ticker 'A'. An improvement to data aggregation would to be query each ticker each day, though processing time would, literally, be 500 times longer for the SP500.

4.2.3 Data Storage and Processing. Map Reduce technologies were initially considered for data storage and processing, but given the application level modularity required on a short timeline local machines were used with GitHub for local development. This latter setup was sufficient, despite slow, as we were initially using three technical indicators. The run time for learning the model using these three indicators on static data to produce a list of recommended stocks to buy was approximately a couple of minutes. As we added 12

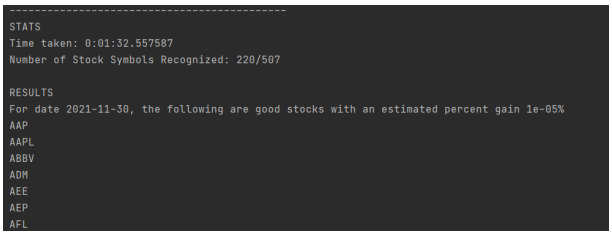

Figure 1: Initial run time for buy recommendations

more indicators and aimed to build a Random Forrest model for each stock separately using 2,500 rows and 15 features, our local storage and processing power became a challenge and we started to encounter run time problems due to size and computing power. This led us to opt for using AWS S3 buckets for high availability storage and EC2 cloud computing for faster processing. Improvements that were in development before the deadline included IO optimization and Big O considerations when training the entire system.

4.2.4 Modelling and Prediction. Early on, we decided to focus on technical analysis as opposed to fundamental analysis to predict the direction of a stock. The former relies mostly on trading data to inform the investment decision. For each stock, we had 10 years of price and volume data retrieved from Yahoo Finance. We also had 10 years of the number of mentions for each stock retrieved from Reddit. For technical indicators, we analysed the indicators that are widely used by practitioners and opted for the following:

- Price to Simple Moving Average (SMA), which measures the ratio of the price to the mean of prices over a specific number of days.
- Bollinger Bands, which measures the volatility in terms of price standard deviation from the mean over a period of time.
- Volume Adjusted Moving Average, which takes into consideration the trading volume as well as the SMA.
- Relative Strength Index, which measures the extent a stock is over- or undervalued.
- Momentum, which measure the rate of change in the stock price over a period of time.
- Moving Average Convergence Divergence, which measures the relationship between two moving averages of the price, one with a short time window (12 days) and longer time window (26 days).

We calculated these indicators over several time windows to capture both short (5, 10, 20 days) and long term (50, 200 days) trends. Along with the number of mentions on Reddit, we were able to create a data set to train the model.

We analysed several models, but found the use of Random Forrest to be relevant to our project. Random Forrest is an ensemble of decision trees that use bootstrap aggregation. We found that Random Forrest offers the following advantages:

- Accuracy of Random forest is generally very high
- Its efficiency is particularly notable in large data sets suitable for our application
- Provides an estimate of important variables in classification especially we are using a dozen feature list
- Forests generated can be saved and reused allowing several users to use the same platform without having to retrain the model
- Unlike other models it does not overfit with more features allowing us to add new features later on using the same code base

Our initial plan for choosing the dependent variable, i.e. label, was to classify the label as 1 if the majority of indicators increased by a threshold we specified and 0 if not. We then used these labels to train the model splitting the data 60/40 for training and testing taking into consideration we are dealing with time series data.

4.2.5 Visualization. We analysed several approaches, including tools taught during the course, we opted to use HighCharts . Highcharts is an SVG-based charting tool that allowed us to create stock charts suitable for the web and mobile apps, as well as over 40 built-in technical indicators.

In preparation for charting, we wrapped our back end that produces the predictions via an API that can be accessed by the web app which is hosted and run on AWS. We also developed ancillary utilities to allow us to upload finished code after debugging and testing from our local machines to AWS ensuring it runs smoothly on the web as it does locally. With the candlestick and volume data from the API, we were able to hide and show all technical indicators with check boxes, and generate any parameter the user wished to see with a submission button. Most of the charting is smooth for the associated data size, this could be lighter and not fully reload the chart if the indicators were regenerated or brought in as a full series from the API during a promise call, though this was another item brought to minimum viable product during time constraints.

4.2.6 User Interaction. The game, written in AJAX and JavaScript, starts on a given day and the user is presented with a queue of stocks. The initial buying power is $10,000. For each stock in the queue the user has three options: buy, sell or skip. The user is presented with buying opportunities and with sell opportunities if they hold the bearish stock. Each investment decision is recorded and visually displayed as an addition to the portfolio. The user moves to the next trading day either by completing the queue for the day or manually by finishing the trading session. The game ends after a number of days specified by us.

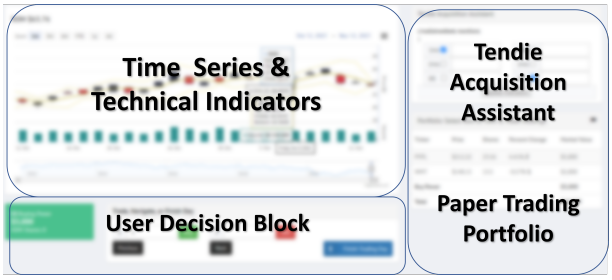

Figure 2: Initial Hi Fidelity Mock Up.

4.2.7 Displaying the results. We displayed the results as the return on the portfolio given the starting cash was set at the beginning. We also wanted to display the result of each decision. This required us to log the user interaction for every game run in order to be able to retrieve and calculate the performance for each user playing the game. We used market comparison in evaluation.

5 EXPERIMENTS & EVALUATION

Experiments aided in design and development as we followed an iterative prototyping method recommended by Dr. Polo versus a typical software engineering process. While we planned for automated confidence intervals, t-tests, and other ranking systems to compare results to different trader types, and even cluster users by their skill time did not allow for nearly enough data gathering from roommates, friends, and family. This project was true to the class in iterating up to a minimum viable data and visual analytics tool, so the development and data engineering experience itself warranted plenty of applicable tests:

- What are the best data APIs for social media, stock, news, and other distributed data
- How to deal with missing information and clean the data especially the constituents of the S&P500 changes periodically?
- Unlike the numerical nature of price data which can be assessed mathematically based on raw data, sentiment requires analytical tools to convert raw data into something we can train the model on. How would we be able to capture that as feature?
- What to use as features - independent variables - and label to develop a predictive model?
- What model to use that can be reused after training with several concurrent users without having to retrain for each trading session?
- How to deal with the capacity constrains arising from a data set with 1.3 million rows and 15 features?
- How to segregate between the analytical part of producing the predictions that requires significant processing from the visual interactivity expected from an optimized web application?
- How to keep an audit trail of user interactions in order to complete the user experience?
- Do we see the impact of human decision on the return results?

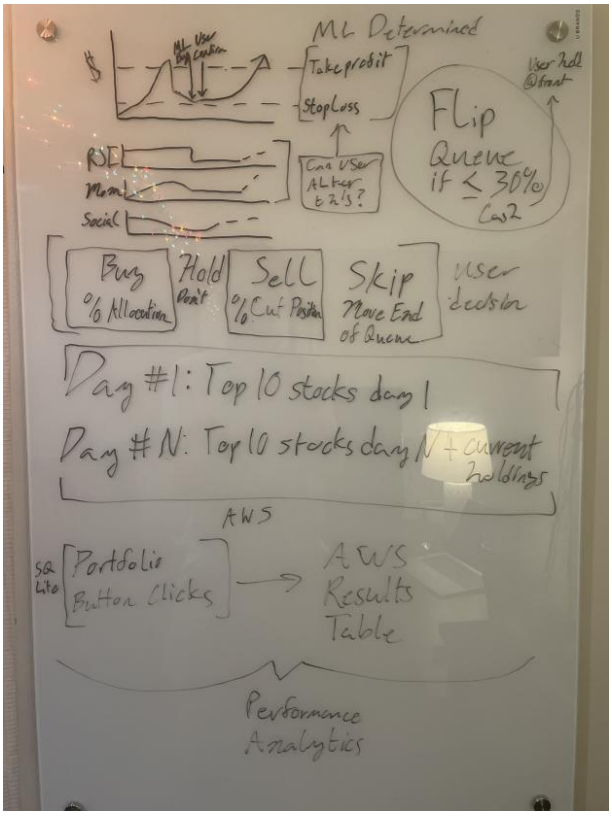

Figure 3: Our cross border brain storming session.

5.0.1 Dealing with a changing list of constituents. We were actually able to automate SP500 changes by downloading the Wikipedia HTML and converting the current and change log tables to CVS files. This was actually initially implemented on the Reddit API but never was integrated.

5.0.2 Dealing with the limitation of Twitter. Not having the Twitter API access was actually a silver lining as Reddit limited stocks to a nice, low, but variable subset of relevant stocks for the test week.

5.0.3 Capturing the sentiment from Reddit. NLP for sentiment is hard, luckily the score comes with each sub and comment ordered by it. We could also filter out the garbage with post flares related to, well, garbage. Identifying stocks took a bit more conditional and/or statements than would classify as elegant, but it worked well.

5.0.4 Deciding on what to use as a label for the decision trees. Initially when creating the labels for the decision tree, the output was generated from calculating the percent change among the indicators and returning 1 for buy and 0 for hold. If a majority of indicators thought it was a good day to buy based on percent change, then that record was marked as a 1.
This logic was later changed to simply use the output of the technical indicator itself and not calculate any changes on the indicators. Instead, the labeling would be done based on the percent change of the price from the current day to five days ahead. If there was a loss below the percentage threshold specified, then the label was marked as -1. If it was greater than the percent threshold, then it was marked as a 1. If it was between the upper and lower thresholds, then the label was 0. This allowed us to better categorize our results and return better predictions.

5.0.5 Dealing with run time when training the Random Forest. The initial run time of the random forest was low. We used 100 trees in our classifier and only 3 features to train upon. Adding more features caused the training time for 30 days of predictions to increase from 45 minutes to 2.5 hours. In order to gain a similar prediction with a decreased training time, we lowered the number of trees to 10. We were able to deploy models faster.

5.0.6 Developing predictions for buy and sell. In order to get predictions for buying and selling stocks, we needed to gather social metrics and financial data across a large time span from both Reddit and Yahoo Finance. Technical indicators were applied to the financial data and this was joined with the social metrics from reddit to create training data. Additionally, labels were created as mentioned above. A model was trained using the sklearn.ensemble.RandomForestClassifier with 10 trees and a random state of 42. Due to the data being time-series, we had to split directly on 60% of data to keep ordering of train/test correct instead of using the sklearn.model_selection.train_test_split function.

5.0.7 Automating the work flow to develop, debug, test and upload to AWS. Since we used git/GitHub as a VCS, we were easily able to load the application and data onto EC2. The steps for setting up a

continuous service for the WebApp included making minimal changes for host/port and running an Apache server via systemctl. Additionally, we needed to write a service to do the same for the WebAPI, and after doing so used Gunicorn for the server. This made the application accessible to all users.

5.0.8 Technical Indicator Overlay. The overlay of technical indicators and count of mentions on the subreddit helped users to perform better than compared to blindly investing. By viewing the trend lines and seeing how often the stock was mentioned was a reason some of the users liked the app. The development time on this was short so it is not as elegant as it should be, mentioned above.

5.0.9 Dealing with a long list of recommendations for a given trading day. While we received a long list of recommendations for a given trading day, we wanted to have the list catered to r/wallstreetbets. For this reason, we filtered the list of symbols. If a symbol was not seen on the subreddit for a given day, it was not included as a recommendation. This wasn't giving enough stocks, and was where we revisited comment additions into the mention volume. This involved restructuring how the code ran for furth optimization from the present to the past with continuous IO hits, luckily on an SSD.

5.0.10 Experiment Results. After having users day trade for two weeks we found that given the opportunities displayed and technical indicators presented, user were able to surpass SPY500 returns with an average of 1.4%. Nonetheless, there is a significant spread as with a standard deviation of 2.29%. This spread shows that there is still significant risk on the investment.

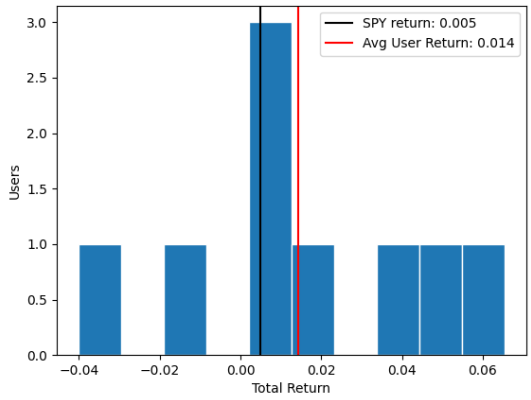

Figure 4: User returns after 2 weeks of trading.

6 CONCLUSIONS AND DISCUSSION

Despite challenges we faced during the development of the application, it was a great experience to go through the design choices and the accompanying trade offs that have to be made in order to develop an application geared for individual users. We are happy with what we managed to develop given the short time period and sparse global locations and availability of team members that have full time work and class commitments. Although, we had to alter our course few times as a result of experimentation, we got the app to work as close as possible to what we envisaged. The core innovation of the application is the queuing algorithm found in dating applications where potential partners are replaced by stocks with the highest predicted return. The second innovation is a stream of stocks suggested to users solely based on dynamic expected price changes rather than traditional similar stocks or historical trend offered by most online brokerages. The third innovation is the collection of user actions in concert with a predictive algorithm. We envisage continuing work on the project, namely:

- explore additional features to train the model on
- incorporate text analytics using several sources to analyse if sentiment is bullish or bearish
- develop a trading bot that uses the predictive model to automatically place trades on behalf of the user
- enable the user to go back in history to trade and assess his/her performance

7 DISTRIBUTION OF TEAM MEMBER EFFORT

All team members have contributed similar amount of effort.

REFERENCES

[1] Girija V Attigeri, Manohara Pai M M, Radhika M Pai, and Aparna Nayak. 2015. Stock market prediction: A big data approach. In *TENCON 2015 - 2015 IEEE Region 10 Conference*. 1–5. https://doi.org/10.1109/TENCON.2015.7373006

[2] Seungwoo Jeon, Bonghee Hong, and Victor Chang. 2018. Pattern graph tracking-based stock price prediction using big data. *Future Generation Computer Systems* 80 (2018), 171–187. https://doi.org/10.1016/j.future.2017.02.010

[3] Burton G Malkiel. 2003. The efficient market hypothesis and its critics. *Journal of economic perspectives* 17, 1 (2003), 59–82.

[4] Raúl Gómez Martínez, Miguel Prado Román, and Paola Plaza Casado. 2019. Big Data Algorithmic Trading Systems Based on Investors' Mood. *Journal of Behavioral Finance* 20, 2 (2019), 227–238. https://doi.org/10.1080/15427560.2018.1506786 arXiv:https://doi.org/10.1080/15427560.2018.1506786

[5] Venkata Sasank Pagolu, Kamala Challa, Ganapati Panda, and Babita Majhi. 2016. Sentiment analysis of Twitter data for predicting stock market movements. *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)* (2016), 1345–1350.

[6] Andrew Urquhart and Frank McGroarty. 2016. Are stock markets really efficient? Evidence of the adaptive market hypothesis. *International Review of Financial Analysis* 47 (2016), 39–49.

[7] Stéphane Meng-Feng Yen and Ying-Lin Hsu. 2010. Profitability of technical analysis in financial and commodity futures markets — A reality check. *Decision Support Systems* 50, 1 (2010), 128–139. https://doi.org/10.1016/j.dss.2010.07.008

[8] Yuzheng Zhai, Arthur Hsu, and Saman K. Halgamuge. 2007. Combining News and Technical Indicators in Daily Stock Price Trends Prediction. In *Advances in Neural Networks – ISNN 2007*, Derong Liu, Shumin Fei, Zengguang Hou, Huaguang Zhang, and Changyin Sun (Eds.). Springer Berlin Heidelberg, 1087–1096.