Using Data Analysis to Evaluate Companies on a Short and Long Term Scale

Tyler Kelly, Raidel Hernandez, and Ibrahim Atiya

I. Introduction

Stock market trends are no longer affected solely by the actions of majority shareholders. With the advent of social media, news and sentiment about a company spread quickly and affect daily stock evaluations. To monitor and accurately predict daily stock market changes, we will be examining social media data, classifying the data based on sentimental value. Our hypothesis is that social media trends will have an effect on the daily fluctuation of the stock market and can create a model for accurate daily predictions. While we predict that social media trends will play a big part in accurate forecasts of the stock market, we do not want to overlook the power board members can have on a company's evaluation. To do this, we will be analyzing video interviews of board members from public companies in order to monitor their emotional status. We believe that by being able to detect the target's emotional status, we will be able to make great estimates on that company's earnings, allowing potential users of this software to make a much more informative decision.

II. Literature Survey

As it happens, there is an excessive amount of research done in the field of social media analyzation, and stock market prediction methods. According to google scholar, "social media sentiment" brings up a staggering 700,000 results and "social media sentiment stock" brings up nearly 175,000 results. While there is no shortage of research on that matter, it does make it difficult to parse out the useful, and more novel documentation out there. Though the documentation is thorough, to the best of our knowledge, we have provided the first approach to incorporating multimedia data into the prediction model. While some work does incorporate multiple dimensions, they are primarily text-based only (twitter feed, restaurant reviews, product reviews) and do not explore multimedia based solutions.

Most of the literary work focus primarily on the way in which the research team analyzed the data. For instance some work used "Granger causality analysis and a Self-Organizing Fuzzy Neural Network" [Bollen] and others used Bayesian Networks and Autoregressive models [Zuo]. While this information was somewhat interesting, it did not have a lot of importance in our experimentation. We did not focus heavily on the techniques for analyzing this data. We did find some useful information in the literature however. [Nguyen] mentions in his paper the importance of topics. He analyzes these topics about a company, and uses this information to better predict changes in the stock market. Meaning, Nguyen will look at the sentiment value for a generic topic on Twitter, so instead of analyzing a specific company on Twitter, look at a more general topic. This method can provide some decent insight, and can get a more accurate reading in some cases on how a specific company will perform. For example, if the general sentiment about health, fitness are positive and sugar products is negative, then Coca-Cola and related brands might not fair well.

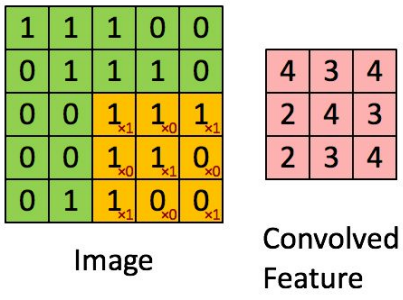
The biggest takeaway from much of the literary work was that emotional responses in the public do have an effect on stock market trends. The concept of Efficient Market Hypothesis was first studied in

1969 by [Malkiel] which resolved into the conclusion that the stock market fully reflects all available information. Meaning price fluctuations are purely based on new information or news. Due to the random nature of information, stocks follow a “random walk pattern and the best bet for the next price, is the current price ” [Nguyen]. However, as history tends to repeat itself, if we are able to predict new unreleased public information by reading the emotional status of CEO’s and monitoring general public sentiment, we just may be able to accurately predict stock market fluctuations.

III. Methodology

A good way to classify a dataset that has multiple features into a binary result is by using a decision tree. Based on the examination of each of these features, a decision is made on the class resulting from the data. This lends itself well to the problem that needs to be solved. For each day, a stock has multiple features pertaining to its price. This includes the opening and closing prices, the volume, and the day’s highs and lows. For each unique set of features pertaining to a stock’s price in a day, a path can be taken on the decision tree to produce a final classification.

A Convolutional Neural Network is great for image recognition, when it applies all the same concepts of an ANN. Additional convolution operations are applied to each image at every hidden convolutional layers. This is done to filter signals, and find patterns in the signals. Therefore. Every neuron in the artificial neural network implemented in this project is a called convolutional neuron. The most important part of convolutional operation on images is the filter size used. In the convolutional layers, suppose we have an image X, and a filter F, the model calculated the dot product of these two object (X * F) and adds a bias at each neuron. At each neuron each convolution operation will result in a single output which will be the input to the next hidden layer. The image to the left illustrates how this method is applied, It slides the filter down the image, until it finds the final output convolved feature (output of this neuron, input to the next one), this is know as strides.



After the model has passed the convolutional hidden layers, it will then step into the Pooling layer. The pooling layer is used to reduce the spatial size and complexity of the output from the previous layer, hence reducing amount of computation required since there are less parameters to worry about. A technique known as Max Pooling is applied, where it takes a filter of size F and multiplies it by a filter of the same size, and return the maximum operation over the filters. Therefore the pooling layer helps prevent overfitting, which is when the model predicts very well on the training data, but performs very poorly on real world images.

The final layer of a convolutional neural network is a fully connected layer, which receives input from all previous layers. This final layer is also fully connected to allow for back propagation through the network, allowing the network to adjust the weights neurons at previous layers. Therefore, allowing the network to conform to the images were initially used as input to the system. At the final layer a Softmax function is applied to the output of this interconnected final layer to produce a prediction between [0, 1] for any image the network tries to predict on.

IV. Implementation and Challenges

There are three main independent components to our application. These independent components were worked on separately, and then combined into a working application that uses all the data from the components in some meaningful way. These components were: twitter sentiment extraction and normalization, emotional recognition and normalization, and stock evaluations determined by positive or negative trends. All three components and the additional components that tie everything together, lie inside a docker image. Docker is an application that allows for portability amongst systems by providing containers that allow for an additional layer of abstraction and automation of operating-system-level virtualization. Because of the nature of our program, a large set of libraries are required, and forcing the user to install the myriad of libraries would be taxing, and can lead to conflicts on certain operating systems.

Twitter sentiment was originally meant to be the sole dimension in our model. It wasn't until further examination of the problem that we realized it wouldn't be completely reliable on its own. That being said, it was the first component in our model created, and with it came a few challenges.

The basic idea for the twitter sentiment component is quite simple. Using Twitter's official API, extract a series of tweets containing certain keywords (company name, company stock symbol) and analyze said tweet. The process of finding sentiment for a single tweet is quite straightforward, clean the tweet of extraneous information which includes stop words (words with little significance in determining sentiment), and other superfluous aspects of a modern sentence structure including irrelevant punctuation. Once the tweet has been cleaned, we can pass this off to a natural language processing library which will send back a sentiment value for the tweet. In particular, we used a python library called TextBlob, which is a general text based library that has a lot of useful components. We just use the NLP part of this library, which allows us to judge the sentiment for a specific tweet. The basics of sentiment analysis are this: TextBlob has a database of most words in the english dictionary, each one assigned a polarity and objectivity score. Polarity is based on how polar a certain word is in the range of [-1, 1] and objectivity is how objective or subjective a word is on the range [-1, 1]. For the purposes of this project, we exclusively look at the polarity, as we found it to be the best measure for determining the general sentiment about a company. TextBlob than utilizes a popular tool called PatternAnalyzer which takes into account the order and structure of the tweet, and uses that information to better inform the polarity score. This makes it so a tweet's overall sentiment is not just the sum of its parts, but instead takes into account the structure as well.

Once we receive the score of an individual tweet, we can then look at the next tweet in the series, and obtain its sentiment score. This process is continued for N number of tweets, where N's optimal value as yet to be determined. As it turns out, there a large number of tweets on the internet (6,000 new tweets created every second) so finding a large amount of tweets, while also taking processing time into consideration is a non-trivial task. We ended up sticking with a value 1000 for this N.

One of the larger problems associated with the twitter sentiment is a restriction put upon on accessers of the official Twitter API. Namely, this restriction is a max seven day limit history of tweets. This means that using the official API would make it impossible to go back in the past to access tweets, making data extraction for our model nearly impossible. Luckily for us, we were able to find a open source repository that can access tweets older than seven days. They are not using the official Twitter API, and instead access the data through a series of JSON requests to Twitter's site. This means that access times are noticeably more time consuming than using the official API, and our approach is also susceptible to not working in the future if Twitter changes their code base in a major way.

Emotion recognition is another component of the stock prediction model, for this process a convolutional neural network (CNN) powered by tensorflow is used to apply mathematical models on out network and produce an output. To train a CNN we had to obtain a number of images from each class we wished to classify on, we were able to obtain an open-source dataset of images from the Cohn Kanade, CK and CK+, database. Once we obtained the images, we used them to train the model and train on 5 of the 7 emotions provided by the database. We used the 5 emotions with the largest set of images, which are anger, disgust, neutral, happy, and surprised. We then used 80% of the images for training and the other 20% for validation. After training a model through the steps discussed in the methodology section, the training model has a validation accuracy of 73%. We predict the model to be overfitting due to the small dataset we are training on, even though the model is overfitting it still does a pretty good job at predicting on images outside of the training sample.

Any prediction model needs to be compared to real results to determine its accuracy. When analyzing a stock's price data, it's important to be able to extract trend information from a stock's daily price fluctuations. In order to do this, one needs to compare the daily opening and closing prices. However, the daily opening and closing prices aren't sufficient enough to make a determination.The highs and lows of each day must also be factored in to account for intraday trends. Only then will a determination on whether a day's price data indicates an upward or downward trend be accurate.

Price data for each stock is retrieved from Yahoo Finance using the Python "pandas" library. The data is retrieved for a timeframe indicated by a start and end date. The days' price information is read into a "pandas" data panel where individual features like the opening, closing, high, and low prices for each day can then be isolated. For a given time period, an assessment is made on whether the day's data is indicative of a positive trend (bullish), or a negative trend (bearish).

Analysis of price data begins by calculating the delta for the day. The delta is the change in price as a percentage of the stock's overall value. Then, a decision tree, shown in figure 1.a, is used to classify the daily data as bullish or bearish. If the delta is greater than a threshold, the program classifies the day as bullish or bearish according to the delta. The confidence for each classification is made based on the delta value after the retracement or rally values are considered. For a bearish classification, the rally is calculated as the amount a stock recovers after hitting its low for the day. For a bullish classification, the retracement is factored in as the change in a stock after hitting its daily high. These values are weighted against and combined with the delta to produce the confidence.

For data where the delta is low, classification is done differently. The trade day fluctuation is the main feature used to classify the day's data. A fluctuation that is greater than a certain threshold is used to

determine whether a reversal occurred. A bullish or bearish reversal occurs when a stock's price changes a great deal before it hits a high or low, respectively, only for it to return to a value close to its open price. The delta is weighed against and combined with the reversal change to make a classification for each of the patterns. When these patterns aren't present, the delta is used as the sole factor in classifying the data.

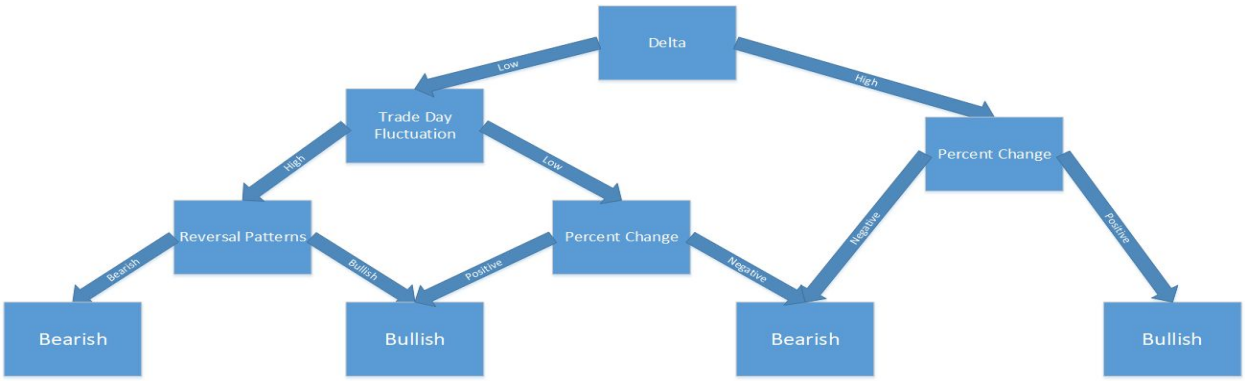


Figure 1.a. The Decision Tree used for analyzing a day's price data.

A design challenge with building the decision tree was how much to weigh each feature and how to determine the threshold at each juncture of the tree. The final thresholds were determined after testing different values against actual price data. Research into what constitutes a positive or negative trend to traders was also needed to help hone in on the appropriate values.

When doing the price analysis, the biggest problem encountered was fetching the price data for the stocks. There are a few APIs available, such as Google Finance, Quandl, and Yahoo Finance. However, many of these APIs were either deprecated or nonfunctional when the project was being developed. The pandas library proved to be a dependable method for getting the price data but this method doesn't work every time. Sometimes a response isn't returned and another query needs to be made to get the necessary results.

The emotion prediction model is called by the worker app, once it has downloaded a video and extracted the frames from the video. It will call the predict function which takes in 2 parameters, the stock name it wishes to predict on, as well as the option to extract faces from the video or not, by default it is of value 0 which means that faces will not be extracted from the images in the dataset provided. The CNN will predict on the raw images located at the company path supplied to the function parameter. To predict correctly, the model rebuilds the weights of each neuron and connection from a graph that was created during the training process. Our project is able to analyze many images from a stock symbol and produce a confidence value of the emotion being more negative or more positive, hence a value between [-1, 1]. To achieve this the application obtains the emotion of every image in the directory being classified and after obtaining a result for each image it places the image into a counter bucket, which keeps track of how many times each emotion was predicted for every frame of the video. The final output of the prediction model is a normalized value between [-1, 1], this is achieved by applying weights to each emotion and then adding or subtracting a bias for each object read in, the bias increases or decreases the value based on the overall predicted emotion of the images.

Finally, we needed to merge all three components together to streamline the entire process. To do this we have a single python script called worker.py that takes in a single parameter, a video file (which presumably contains contents of a CEO). The file is assumed to be in the form SYM_YYYY-MM-DD.mp4. We first parse through this video file, and split it into frames and place the frames into directory. We then import all other previously mentioned components into this script, which will call the necessary functions from each one. Specifically, we obtain the sentiment score for SYM from the period from YYYY-MM-DD to YYYY-MM-DD + 7. We then get the emotional score for the video that was submitted, and then attempt to classify the submission using a separate module. We then get the actual bullishness, and compare that value with the classification. Once we have all the values from the different classification algorithms implemented, we are able to use a K-nearest neighbors classifier to predict which class the new value belong to. To classify using KNN, we graph sentiment and emotion on the x and y coordinates of a 2-D graph, respectively. We then use a KNN with k=3 (because out dataset is small), which always gives us a prediction with one of the following probabilities (1, 0.6, 0.3, 0). Therefore, we use that value as the confidence that a stock will perform good or bad in the future.

V. Results

Our results were positive given our smaller dataset. After training on 18 videos, we then tested five videos and got the correct prediction for all cases. Unfortunately, due to the nature of data, most of the objects are primarily classified to be a “not buy”. This could be the nature in which we have defined our data. Meaning, since our data is primarily based on the videos (the date of the video is then used to analyze stock price changes and sentiment scores), there can be an argument made that the only time a CEO or CFO would go on air would be to clear the air about some issues in the company. Effectively trying to help out their stock’s chance at survival.

As you can see in Fig2 below, a good percentage of our data points were deemed to be “bad buys”. Based on this graphical representation of the data, we were able to determine a value for k that would have a meaningful impact. For a k larger than 3, we would never be able to get a “good buy” decision as no 4 “good buy” points are close enough to win a majority vote against the “bad buys”’.

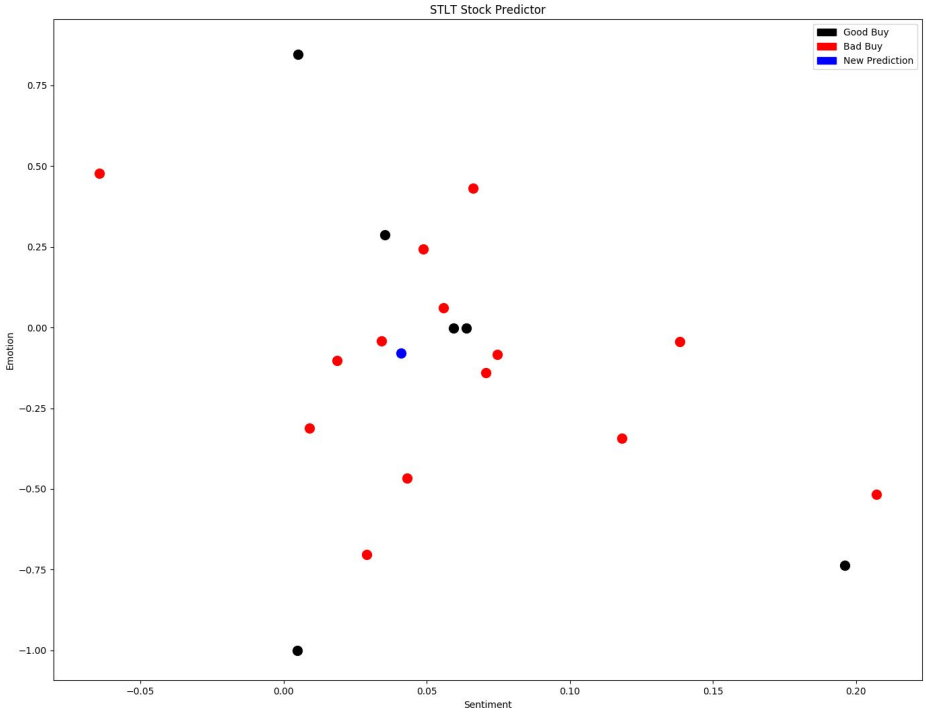


Fig2 : Using KNN with K=3, prediction is " Not Buy"

VI. Conclusion

Future work would revolve around a number of aspects that would improve the model. One of the key issues we have, is that the twitter sentiment score and the emotional score carry the same weight, independent from the bullish, or bearishness determined for that week. We would like to incorporate the following: Determine the significance of the scores by observing the bullishness, and comparing that to the two dimensions. Meaning, it is already clear that the two attributes don't have the same amount of weight through intuition alone, so applying a educated weight to each dimension would significantly increase the capabilities of our model. If we were to find, for example, that even the twitter sentiment was negative, and the emotional score was positive, but the stock was bullish over this period, then emotional score would have a higher weight. Perhaps each model performs better under varying temporal fields or bullishness. In this case, a different weight would be applied in each case.

One of the most challenging parts in a time restricted projected for us was the collection of the data. Twitter sentiment and stock evaluations were quite simple to retrieve as their data sources are well documented and easy to obtain (though we did encounter a few challenges here). However, obtaining the video data was a very time-consuming task as it involved looking up video interviews on youtube with members of the company, finding relevant information in the video (close-ups of said member), noting the date, and downloading the portion of the video that was relevant to us. From here we would need to split the video into its frames, and then analyze these frames and score the video based on this subset

frames. Being able to automate even a portion of this process would be hugely beneficial, and finding a database of videos with these CEO's and CFO's would be even more so.

Another direction we would like to go in is obtaining the context of the video interviews in an empirical way. As many of these interviews are hours long, they don't necessarily talk about the company the entire time. At which point it becomes clear that if they are talking about their struggling childhood and show an emotion of "sad", this shouldn't have any effect on our predictions of the company's value. However, due to time constraints, we were unable to obtain the context of many of the portions of video used by our model.

We are also hindered in our emotional data recognition. Right now, the model we have trained for this part is using a very small subset of faces. We are currently awaiting approval for a much larger database of faces, which would enable our emotional recognition model to be far better at predicting the correct emotion. This type of improvement would minimize errors, and provide us with a more accurate emotional score.

References

1. Bollen, Johan, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." *Journal of computational science* 2, no. 1 (2011): 1-8.
2. Nguyen, Thien Hai, Kiyoaki Shirai, and Julien Velcin. "Sentiment analysis on social media for stock movement prediction." *Expert Systems with Applications* 42, no. 24 (2015): 9603-9611.
3. Zuo, Yi, and Eisuke Kita. "Stock price forecast using Bayesian network." *Expert Systems with Applications* 39, no. 8 (2012): 6729-6737.
4. Malkiel, Burton G., and Eugene F. Fama. "Efficient capital markets: A review of theory and empirical work." *The journal of Finance* 25, no. 2 (1970): 383-417.
5. Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), Grenoble, France, 46-53.
6. Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101.