ISyE 7406 Group 2

Spring 2022 April 26, 2022

Final Report for Semester Project

Predicting the Next Meme Stocks with Sentiment Analysis of Social Media Data

Task Distribution

Team Member	Task Description
Te-Kai (Kelvin) Chou	Data Collection, Data Modeling Strategy Analysis
David Cortes Gomez	Data Modeling Strategy/Analysis and Results (Change Point Detection) & Introduction
Colin I Diggs	Data Modeling Strategy/Analysis and Results (Wavelet Coherence Analysis) & Conclusion
Thien Dong	Introduction, Theoretical Framework and Research Objectives, Collecting tweets on meme stocks, Collecting daily stock prices, Panelizing data for analysis, and Developing the four-pronged framework
Zhelun Rong	Data Collection, Data Cleaning and Scoring Sentiment Analysis
Weiyu Xue	The first three criteria of the four-pronged framework, Random Forest, Logistic Regression, ANN
Chia-Te (Sam) Yi	Data Modeling/Analysis of Results, Model Evaluation

1. Introduction

Early 2021, a group of traders who knew each other from the WallStreetBets forum in Reddit concocted a get-rich-quick scheme: mass purchase to turn the tide for the then struggling videogame retailer's stock—GameStop. The initiative quickly caught fire, attracting the attention of almost 900,000 retail investors, many of whom did not have any investing knowledge whatsoever. The outcome, however, was remarkable. From $20 per share, GameStop's price jumped to nearly $500. Three months after the GameStop saga, the AMC's stock price movement exhibited a pattern that is not unlike to that of GameStop. Together, AMC and GameStop set the foundation for a new (and highly underexplored) phenomenon: the emergence of meme stocks. Defined as stocks whose trading volume increases not necessarily because of a company's performance, but because of social media attention which may result from a variety of factors unrelated to the company's performance, financial position, or other business fundamentals, meme stocks are subject to high volatility and thus extremely sensitive to speculative forces. Despite the popularity of meme stocks within the community of both professional, institutional investment funds and retail investors, the body of research on this subject has been surprisingly scant. Most research on this topic has been largely exploratory, i.e., to understand meme stock trading as a result of the herding behavior (Aloosh, Choi, & Ouzan, 2021) or the drawbacks of traditional information disclosure requirements (Yahya & Chiu, 2021). In other words, as meme stock is still a relatively recent phenomenon, a large body of research in this area is aimed at unveiling the dynamics and mechanism through which meme stocks transpire.

Our aim for this project is one step further than merely exploration. In particular, we attempt to predict the meme status of a stock, using its social media sentiment as the predictor. In other words, it is a classification problem in which each meme stock is categorized as 'being likely to go meme' or not. To this end, we assemble a complete dataset on all the stocks that were mentioned by Twitter and Reddit users in the year of 2021. Specifically, we take advantage of the Reddit and Twitter APIs to download all the social media posts containing a set of keywords comprised of terminologies developed by the meme stock trading community. We quantify textual data to sentiment score with the well-established NLTK library.

To solve this binary classification problem, we proceed with two modeling strategies. First, based on the Robinhood's trading ban that was enforced on several stocks in early 2021, we construct a complete list of ground truth labels (i.e., meme stocks whose effects were vast enough

to trigger an intervention from Robinhood) to implement three popular supervised machine learning (ML) methods: random forest, logistic regression, and artificial neural networks. For a trading platform that prides itself on the mission to provide everyone with a fair access to the financial market, we can reasonably assume that Robinhood does not take such trading restrictions lightly and thus can treat the banned stocks as truly meme stocks. More importantly, we develop a novel four-pronged framework to determine whether a stock is going meme: 1) not a blue-chip stock, 2) very popular in social media (measured by the frequency the stock was mentioned in Twitter and Reddit), 3) very large spread (to capture the high volatility of its price), and 4) there is a co-movement between the sentiment scores in social media posts featuring the stock and its price. We consider this framework our main contribution to the meme stock literature. Moreover, since our framework does not require a large sample size to run, we are able to compare the predictions with those yielded by conventional ML methods.

As an overview, our results suggest that valuable insights can be gained from using past social media data in predicting the meme status of a stock on a particular day, with all methods achieving at least the accuracy of at least 90%. Moreover, there seems to be a tradeoff between accuracy and recall ratios, suggesting that different models may be appreciated differently by risk-averse vs. risk-loving investors. A comparison of our own four-pronged framework and the three conventional supervised ML methods also shows that the predictions given by ANN are most aligned with those by our framework. In all cases, the number of meme stocks predicted by our framework is much smaller than that predicted by RF, LR, and ANN. In other words, our framework tends to err on the side of caution and thus produces very conservative results, which may be preferred by risk-averse investors. By design, the number of meme stocks is much smaller than that of non-meme ones. Therefore, rather than relying on traditional accuracy scores, we use the macro averaged $f1$-score to evaluate model performances on our highly unbalanced sample.

2. Problem Statement and Data Background

2.1. Theoretical Framework and Research Objectives

The body of research on meme stocks is still in its infancy: at the time of writing, searching for 'meme stock*' on Google Scholar, the largest repository for academic publications, returns only a little more than 360 results in English. On the one hand, this observation suggests that any research on this topic has the potential to make significant contributions to our understanding of this genre

of financial assets. However, on the other hand, it also signals particular academic challenges as we do not have much prior research to build on.

2.1.1. Definition and characteristics of meme stocks

In this research, we informally define a stock that goes meme when it gains a cult-like following on social media platforms that significantly influences the stock's price without any substantial changes in the fundamentals of the issuing company (Hayes, 2022). Around meme stock, online communities have also devised several market terminologies to refer to related subjects and actions. For example, "diamond hands" are 'strong' investors who will not sell even on dips; "to the moon" refers to investors' expectation of extremely higher-than-average returns, while "YOLO" ("you only live once") is used by meme stock traders to encourage each other to go all-in on one particular stock). This glossary has an essential role in our search directions, which we will describe in more detail in the data collection section.

In an attempt to capture meme stocks as a research subject, scholars from many fields have contrasted meme stocks against traditional securities to highlight the former's characteristics. For example, Costola, Iacopini, and Santagiustina (2021) coined the term 'mementum' to refer to a market manipulation strategy that allows "small investors to act as a single large trader, able to successfully manipulate prices," with the potential of destabilizing effects (p. 6). More importantly, they postulate three conditions to define whether a given stock is going through a meme period (aka a 'mementum'). First, the buying signal has to be initiated by social media. Second, there must be synchronicity in the timing between the stock's price/volume changes and its tweet series. Finally, such synchronicity must be persistent. Their study econometrically shows that meme stocks exhibit such patterns stronger than otherwise comparable stocks that are also subject to a high volume of activities on social media. Birgersson and Carlén (2021) apply a similar approach: they compare meme stocks with S&P 500 stocks and find that retail order imbalance (aka, the difference between the number of 'buy' and 'sell' orders) affects S&P 500 and meme stocks differently. In detail, while retail order imbalance increases returns on S&P 500 stocks, it leads to negative returns for meme stocks. On a related note, Vasileiou et al.. (2021) find that returns on meme stocks exhibit exemplary patterns of a speculative investment strategy. Specifically, by focusing on GameStop, the authors find that while information on trading volume and Google searches relating to a stock do provide helpful information that can explain the returns on meme stocks' investment, such returns are reaped by early investors only. In another study, Vasileiou

(2021) shows that returns on GameStop are typical of a short squeeze, which is "a phenomenon in financial markets where a sharp rise in the price of an asset forces traders who previously sold short to close out their positions" (Corporate Finance Institute, 2021).

2.1.2. How does it start for a stock to experience a 'mementum'?

A number of scholars are concerned with the triggering conditions that allow a stock to become a meme stock. For instance, Gianstefani et al. (2022) develop an alert system to detect suspicious users' activities on social networks that are likely to destabilize the financial market. Their "early warning system" builds on two principles. First, there must be an unusual activity revolving around the stock on a social network (measured in trading volume). Second, such activities have the potential to develop into a coordinated, collective movement to influence the market trends of a financial asset. They test their system with an event study on two meme stocks (GameStop and AMC) versus two non-meme stocks (Apple and Microsoft) and find significant differences.

In a similar vein, Zheng et al. (2022) analyze the subreddit "r/wallstreetbets" and find that the topics discussed on meme stocks become more centralized, while users' sentiment becomes both more positive and more divergent. In addition, the topological structure of the networks of investors trading meme stocks tends to become more efficient, which allows new information to be diffused even faster while using even less energy. Further, Choy et al. (2021) build upon the Eulerian fluid flow system of physics equations to measure investor impatience, showing that conservation of energy equations can be applied to detect abnormal market activity in advance. Similarly, Takagi (2021) computes the log-periodic power law confidence indicator to identify potential bubbles of meme stocks, using a sample of four notable meme stocks in 2021: GameStop, Koss, AMC, and Tesla.

But what triggers these retail investors to initiate such a scheme to salvage the stocks of financially struggling companies? In what they call the 'Meme stock paradox', Chiu and Yahya (2021) argue that such a desire to oust institutional investors with their own short positions comes from three factors. First, retail investors, be financially naïve as they are, may be able to discover new pieces of information that are unknown to sophisticated, institutional investors. For example, although it may seem irrational to invest in a brick-and-mortar chain of video games that will soon become obsolete, investors are optimistic that GameStop's management would be able to turn the tides after the pandemic. Indeed, the company timely recognized this newfound faith in its investors, repeatedly raising capital by issuing new stocks and debts and updating its e-commerce

capacities to stay relevant. Cahill et al. (2022) contrast institutional investors with retail investors and find that the two are different in the sources and types of company information they pay attention to. Second, the authors argue that retail investors who rallied up the prices of these meme stocks play the role of trend makers. Financial assets are no longer evaluated by their intrinsic values or potential future earnings. Instead, owning a GME or AMC is considered "cool" and "trendy." Investors who belong to this group do not care whether these stocks would yield them any profits or not, other than the symbolism associated with owning the stock. Another study that supports this hypothesis is by Aloosh et al. (2021), who find that meme stock traders exhibit herd behavior beyond the short squeeze period. Finally (and related to the second reason), the authors posit that meme stock investors are highly motivated by non-financial reasons: the pride in owning financial assets. The GameStop saga clearly was not the first short squeeze in history: in 1923, the owner of a grocery chain named Piggly Wiggly attempted to short squeeze Wall Street on his own. However, it was especially noteworthy: for the first time in history, the power of the Internet echoes the social pressure of millions of individual investors and empowers trading to virtually anyone who happens to own a smartphone. The Internet has elevated this big group of retail investors to a worthwhile counterpart of 'evil' institutional investors who bet on the failures of businesses. For example, AMC Theaters investors take pride in resurrecting a dying movie theater chain, calling themselves part of "#MyAMC" or the "#AMCArmy" on Twitter. In fact, 80% of AMC stocks were owned by individual investors. Anderson et al. (2022) reverberate this finding. In a recent study, they call it "expressive trading," which is an investment strategy that is not solely driven by profits but by the desire to send a message "as a form of social protest, political speech, or aesthetic expression" (p. 1233), even though they know that they might incur financial losses down the road. Armed with social media, diffused investors can coordinate their trading to cause a momentum whose effect on the targeted stock is considerable. On a related note, Ricci et al. (2021) call these investors "wireless investors" who leverage technologies, online forums, and gaming dynamics to exercise corporate governance and realize social and environmental causes (Haan, 2021; Stiebel, 2021). Additionally, Cao and Liu (2022) call this factor "resistance psychology," through which individual investors compete with institutional investors. Historically, institutional traders have informational advantages and publish short-selling announcements to suppress stock prices and make big profits out of their short positions while long-term investors incur losses.

2.1.3. The effect of social media sentiment on stock returns

A large body of research on meme stocks attempts to investigate the link between social media-related predictors (textual sentiment being one of them) and the market response captured in the stock's price. However, the empirical findings are equivocal. For example, a recent meta-analysis by Gric et al. (2021) on 30 studies investigating the relationship between sentiment and stock returns shows that once corrected for the publication bias, sentiment seems to affect stock returns negatively. In particular, one standard deviation increase in sentiment reduces future monthly returns by 0.198 standard deviations. However, the effect seems to be positive and stronger for individual investors rather than institutional investors. Utilizing Wavelet Coherence analysis on an AMC Theatres stock case study, Vasileiou and Tzanakis (2022) find that the link between sentiment and stock returns is cyclical rather than unidirectional. Specifically, the increase in the returns to AMC stock leads to 1) an increase in the number of Google searches relating to the company, 2) a periodic change in the Put-Call ratio (which is also subject to the influence of hedge funds' strategies), and 3) an increase in the stock's trading volume. AlZaabi (2021) also finds a positive relationship between social media sentiment and stock prices, although the effect is heterogeneous across different meme stocks. Specifically, the author focuses on the subreddit r/wallstreetbets and finds that the Reddit sentiment has a stronger influence on GameStop's price than on AMC Theatres'. Furthermore, Gupta (2021) experiments with the Convolutional Neural Network and Multilayer Perceptron Model and finds that the two models can explain 58% and 52%, respectively, of the stock's price (GameStop and Tesla) on day $t + 1$ with Reddit sentiment on day t as the predictor variable. Similarly, Dechow et al. (2021) focus on the so-called Olympics stocks (stocks that benefit from positive news in the summer Olympics). They find that during the summer Olympics, trading volume and price volatility of Olympic stocks are abnormally high on days when media outlets associate the issuing company with the Olympic games. Moreover, the effect is stronger for firms with a higher percentage of retail investors, suggesting that investors' sentiment has a causal effect on the stock categorization. Finally, in a similar vein, Buz and de Melo (2021) analyze all the investment advice in the r/wallstreetbets subreddit and assemble a portfolio based on these pieces of advice. Interestingly, they find that this portfolio significantly outperformed the S&P 500. In more detail, the 'r/wallstreetbets' portfolio's value doubled over a three-year period and increased nearly five times over the last year, even after accounting for the GameStop saga in early 2021.

Rather than focusing on the stock price per se, a few studies are only interested in the direction of the price movement (i.e., whether the future price will decrease or increase). For example, using VADER to analyze the sentiment of the r/wallstreetbets posts automatically, Wang and Luo (2021) find that adding semantic embeddings (i.e., the vectorized version of the text, using word2vec and BERT) increases the predictive power of social media sentiment by approximately 20%, compared to using only the VADER sentiment score. Furthermore, Dong et al. (2022) pit social media against mass media (which the authors define "as the media outlets that are traditional and run by organizations, such as Wall Street Journal, where news articles are written by professional journalists and curated by editors," (p. 1)) and find that social media and mass media contents have different power in predicting stock prices. Specifically, while mass media content has a stronger predictive power than social media content in a one-day period, social media content outperforms mass media in predicting stock prices in a 2- to 5-day period.

2.1.4. Our Research Questions and Contributions

Our goal with this research project is to advance our understanding of how meme stocks come into being. Specifically, we are interested in using social media sentiment to predict the stocks that are likely to go meme. It is also worth noting that we do not aim to make a normative claim with this study. In other words, we refrain from making a judgment regarding the potential (negative) consequences of the meme stock phenomenon. As has been clear from the literature review, most meme stock studies focus on a limited sample of one to five stocks within a few months to test either a detection model (Section 2.1.2) or the predictive power of social media sentiment on stock prices (Section 2.1.3). In this study, we expand 1) our sample to the universe of all stocks referred to in the subreddit 'r/wallstreetbets' and Twitter and 2) the time window (from several months to the whole calendar year of 2021). Method wise, we implement different methods and compare their classification results: from using the ground truth (aka, the list of stocks that faced the trading ban on Robinhood in early 2021) to train Random Forest, Logistic Regression, and Artificial Neural Network, to developing our own set of criteria (which includes four conditions that we will detail in Section 3.2). To the best of our knowledge, this study is the first to attempt to predict meme stocks using Change Point detection and Wavelet Coherence. In summary, our research question is: *using different methods, can we predict which stock will experience a 'mementum' on which day from its social media sentiment?* Last but not least, we reserve the disclaimer of financial

responsibility, which shall free us from financial consequences experienced by users, should they choose to make investment decisions as per the predictions suggested by our models.

2.2. Data Collection and Data Cleaning

For our study, a few data sources are required for classification and analysis, primarily social media and stock price data. Twitter and Reddit are social media platforms where users can post (or tweet) virtually anything on their minds. The attribute of 'free text' fields allowing people to convey ideas through string freely makes everyone's opinions critical and unique. Therefore, there is a need to subtract meaningful data from people's opinions about meme stocks. Such opinions have the potential to predict the stocks that will be more likely to go meme because they direct the capital flow injected into those stocks.

2.2.1. Collecting posts from the subreddit 'r/wallstreetbets'

For retrieving subreddit specific to 'wallstreetbets,' we use the Pushshift Reddit API wrapper in Python 'PMAW', which is used for the situation when we need to retrieve a large amount of data online. However, there is a connection disruption with Reddit API if we try to obtain more than 20,000 posts information at one time. So instead, we turn to retrieve 20,000 posts information from the first day to the day of each month and finally pull a total of 240,000 post information in the 2021 calendar year under the 'wallstreetbets' subreddit. To make the post information usable and meaningful, we extract certain variables that are critical elements to represent each post, including title, num_comments, score, upvote_ratio, selftext, and created_utc. In addition, some selftexts in posts that have been removed are marked with 'removed' and 'deleted', which may negatively affect the final sentimental scores. Therefore, we transform them into space strings by combining the two columns of 'title' and 'selftext' into a new column called 'content.' The last part of processing data is to utilize the NLTK package and regular expression operations to remove nonletters and spaces in the 'content' column to conveniently make sentimental scores for each post later. The next step is to obtain the stock symbols mentioned in the posts, so we need to know the existing stock symbols in the current market. We implemented it by breaking the content of each post into words and then performing words cross-matched against a specified list of Nasdaq ticker symbols. Since the stocks listed on the Nasdaq are considered growth-oriented and more volatile, it would be a suitable source as a matching list of stock symbols.

2.2.2. Collecting tweets on meme stocks

We utilized the Twitter API v2 to download all the tweets discussing meme stocks from January 1st to December 31st, 2021. The number of tweets that can be retrieved using the Twitter API depends on the account's access level. Twitter approved our application for the Academic Research access, which allows us to pull up to 10 million tweets per month. The list of keywords that we used in our search queries includes '#MemeStocks', '#PennyStocks', 'MemeStock', 'PennyStock', 'to moon', 'diamond hands', 'short squeeze', 'gamma squeeze', 'YOLO', and 'Options Yolo'. We specifically exclude retweets to make sure that we only pull original tweets. In addition, we are interested in only tweets published in English. We consulted the pagination and the query conventions provided by the Twitter documentation[1] to automatically download all the tweets meeting our search criteria and wrote the tweets into an updated csv file. In addition, after each page of results, we insert a brief interval of time sleep (5 seconds) to make sure that we do not abuse the Twitter API. Besides the tweet itself, we are also interested in several metadata, including the ID of the author of the tweet, the time (in ISO format) at which the tweet was published, the ID of the tweet, the language of the tweet (we selected English tweets only), how many times the tweet was retweeted, liked, and quoted, and how many likes the tweets received. Twitter also provides a field called 'cashtags,' which contains all the stock tickers mentioned in the tweet. Unfortunately, 'cashtags' are relatively new and unavailable for many tweets. Moreover, 'cash tags' miss a ticker symbol whenever the ticker is not preceded by the dollar sign. The complete code for retrieving Twitter data was performed in Python and can be found in the Appendix of this study.

2.2.3. Collecting daily stock prices

After collecting subreddit posts and tweets, the next step is to retrieve all the stocks mentioned in these social media outlets. To this end, we utilize the StockSymbol API to retrieve a complete list of tickers across all trading exchanges in the world. Currently, StockSymbol provides interested users with a free API key at https://stock-symbol.herokuapp.com/. We are interested in the tickers in the US market only. Next, we perform a simple regular expression search to find all the tickers with the following conditions: 1) the ticker starts with the dollar sign, or 2) any whole word that has anywhere between 3 and 6 letters in capital format. For example, the tweet "Are you buying into this Silver $SLV short squeeze this week?$GME $AMC $BB $NOK" will return a list that

[1] https://developer.twitter.com/en/docs/twitter-api/pagination
https://developer.twitter.com/en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-recent#Optional

contains ["SLV", "GME", "AMC", "BB", "NOK"], which is then compared against the complete list of tickers provided by StockSymbol. As it turns out, each of these strings is indeed a ticker. In other words, this example tweet mentions 5 stocks: SLV (iShares Silver Trust), GME (GameStop), AMC (AMC Theaters), BB (Blackberry), and NOK (Nokia). We repeat this step for both the sample of subreddit posts and the sample of tweets to get a complete list of all stocks mentioned in the subreddit r/wallstreetbets and in the tweets on meme stocks.

We then submit these lists of tickers to the API of the Center for Research in Security Prices (CRSP), the access to which is available in the Wharton Research Data Services (WRDS) suite, to which Georgia Tech has a subscription. The frequency of price data we can retrieve is at the daily level. For each ticker, we get the following information:

- '`permco`': a unique permanent identifier assigned by CRSP to all companies with issues on a CRSP file. This number is permanent for all securities issued by this company regardless of name changes.
- '`date`': the date on which the price was recorded.
- '`bidlo`': Bid or Low Price is the lowest trading price during the day or the closing bid price on days when the closing price is not available. The field is set to zero if no Bid or Low Price is available.
- '`askhi`': Ask or High Price is the highest trading price during the day or the closing ask price on days when the closing price is not available. The field is set to zero if no Ask or High Price is available.
- '`prc`': the closing price or the negative bid/ask average for a trading day. If the closing price is not available on any given trading day, the number in the price field has a negative sign to indicate that it is a bid/ask average and not an actual closing price.
- '`vol`': the total number of shares of a stock sold on a given date (rounded to the nearest hundred).
- '`ret`': Holding Period Return is the return for a sale of stock i on day t, supposing that the stock was purchased on the previous date $(t-1)$.
- '`bid`': the closing bid of the stock.
- '`ask`': the closing ask of the stock.
- '`openprc`': the open price of the stock.

- '`retx`': Holding Period Return without Dividends; in this returns calculation, Ordinary dividends and certain other regularly taxable dividends are excluded.
- '`spread`': this is an additional field that we created, using the formula 'spread = askhi – bidlo'.

The CRSP API returns the output in the csv format. Each row represents the price data of stock i on date t.

2.2.4. Data cleaning

If we look at those "free text" datasets extracted from Twitter and Reddit, it is easy to see that there are tons of repetitive posts and unhelpful data (Figure 1) since people can post the same posts as many times as they want on those social media platforms and the extracted data contains many advertisements and garbage posts (Figure 2). So, following the principle of *garbage in garbage out*, the very first thing to do is to clean the extracted data and keep the useful information as much as possible.

249	1.24E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
250	1.35E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
251	1.31E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
252	1.22E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
253	1.24E+18	2021-01-31 1	1.36E+18	en	0	0	0	2226 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS ROOKIE CARDS ARE " OUTPERFORMING " BITCOIN #BTC #Ethereum #cryptocurrency !!!NVESTMENT GRO,Ã¶
254	1.24E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
255	1.36E+18	2021-01-31 1	1.36E+18	en	0	0	0	2226 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS ROOKIE CARDS ARE " OUTPERFORMING " BITCOIN #BTC #Ethereum #cryptocurrency !!!NVESTMENT GRO,Ã¶
256	1.28E+18	2021-01-31 1	1.36E+18	en	0	0	0	2226 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS ROOKIE CARDS ARE " OUTPERFORMING " BITCOIN #BTC #Ethereum #cryptocurrency !!!NVESTMENT GRO,Ã¶
257	1.35E+18	2021-01-31 1	1.36E+18	en	0	0	0	2226 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS ROOKIE CARDS ARE " OUTPERFORMING " BITCOIN #BTC #Ethereum #cryptocurrency !!!NVESTMENT GRO,Ã¶
258	1.29E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
259	1.16E+18	2021-01-31 1	1.36E+18	en	0	0	0	2226 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS ROOKIE CARDS ARE " OUTPERFORMING " BITCOIN #BTC #Ethereum #cryptocurrency !!!NVESTMENT GRO,Ã¶
260	1.16E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
261	1.33E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
262	1.27E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
263	2255334513	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
264	1.21E+18	2021-01-31 1	1.36E+18	en	0	0	0	2226 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS ROOKIE CARDS ARE " OUTPERFORMING " BITCOIN #BTC #Ethereum #cryptocurrency !!!NVESTMENT GRO,Ã¶
265	1.32E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
266	1.35E+18	2021-01-31 1	1.36E+18	en	0	0	0	398 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS #138 ROOKIE CARDS ARE SUUUUUPER HOT! PRICE HAS WENT FROM $300 to $3,000 IN 60 DAYS!#KOBE ¦Ã¶
267	1.31E+18	2021-01-31 1	1.36E+18	en	0	0	0	2226 RT @MONEYGANGJAMES1: #KobeBryant 1996-97 TOPPS ROOKIE CARDS ARE " OUTPERFORMING " BITCOIN #BTC #Ethereum #cryptocurrency !!!NVESTMENT GRO,Ã¶

Figure 1. Unfiltered tweets with many duplicates

2215	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$ERFB new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 220
2216	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$ERFB new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 219
2217	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$ERFB new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 217
2218	1861049119	2021-01-28 2(	1.35E+18	ht	0	0	0	0	$BTDG new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 216
2219	1861049119	2021-01-28 2(	1.35E+18	tr	0	0	0	0	$SBFM new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 214
2220	1861049119	2021-01-28 2(	1.35E+18	ht	0	0	0	0	$BTDG new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 214
2221	1861049119	2021-01-28 2(	1.35E+18	ht	0	0	0	0	$BTDG new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 212
2222	1861049119	2021-01-28 2(	1.35E+18	tr	0	0	0	0	$SBFM new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 211
2223	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$XALL new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 211
2224	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$XALL new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 209
2225	1861049119	2021-01-28 2(	1.35E+18	tr	0	0	0	0	$SBFM new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 209
2226	1861049119	2021-01-28 2(	1.35E+18	ht	0	0	0	0	$BTDG new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 207
2227	1861049119	2021-01-28 2(	1.35E+18	tl	0	0	0	0	$HIHI new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 207
2228	1861049119	2021-01-28 2(	1.35E+18	tl	0	0	0	0	$HIHI new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 205
2229	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$XALL new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 204
2230	1861049119	2021-01-28 2(	1.35E+18	tr	0	0	0	0	$SBFM new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 203
2231	1861049119	2021-01-28 2(	1.35E+18	tl	0	0	0	0	$HIHI new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 202
2232	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$XALL new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 202
2233	1861049119	2021-01-28 2(	1.35E+18	tl	0	0	0	0	$HIHI new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 200
2234	1861049119	2021-01-28 2(	1.35E+18	tl	0	0	0	0	$WTII new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 200
2235	1861049119	2021-01-28 2(	1.35E+18	tl	0	0	0	0	$WTII new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 198
2236	1861049119	2021-01-28 2(	1.35E+18	tl	0	0	0	0	$WTII new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 196
2237	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$BTCS new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 195
2238	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$BTCS new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 194
2239	1861049119	2021-01-28 2(	1.35E+18	en	0	0	0	0	$BTCS new alert at https://t.co/knzARFId2i #Penny #pennystocks #PinkSheet #OTC #OTCBB #stocks 193

Figure 2. Unfiltered tweets with many advertisements

In order to cleanse the data, the first thing that we need to do is to browse all data entries since there is some obvious garbage data. Second, we need to do some Feature Engineering work to study what type of data is valuable and what is not in terms of Finance subjects and Stock Markets. After browsing all the extracted data, we found the following major garbage words:

1. https://t.co/knzARFId2i
2. #KobeBryant
3. @MONEYGANGJAMES1
4. https://t.co/
5. @AngelsTrading
6. #VotaYoloAventuras
7. @NandoAventurero

After filtering out all the posts containing these above seven garbage words, the original 4 million entries are reduced to around 2 million. Besides these seven garbage words, we note that Twitter and Reddit allow people to post emojis, which cannot be converted to text properly and turn to some unreadable codes instead. Since it is hard to detect a pattern of those unreadable codes, we retain them all because the number of entries that contain unreadable codes accounts for a very small proportion of the entire dataset. Moreover, such entries will be scored 0 anyway (details in

Section 2.3 on Scoring Sentiment Analysis) and thus will not impact our model accuracy. However, they do incur more expensive computation.

2.2.5. Prepare data in the panel format for analysis

At this point, we have four datasets: Reddit's social media posts, the daily price of the stocks mentioned on Reddit, Twitter's tweets, and the daily price of the stocks mentioned in these tweets. As a reminder, our research objective is to use social media data to predict whether a stock will experience a 'mementum' on a specific date. In other words, we would like to combine the social media data frame and the price data frame into one panel, in which each observation captures the social media data and the price of each stock on each day of 2021. However, the price data is already in the format of 'stock i on day t', and so our primary job is to reformat the social media data such that each row is also about 'stock i on day t'.

To this end, we employ several simple data frame preprocessing techniques. First, we explode each row of tweets/subreddit posts by the tickers. Recall the previous example, in which we have a tweet that mentions 5 stocks. 'Explode'[2] allows us to replicate this tweet 5 times so that each row contains the tweet itself and one unique stock (as a reminder, our unit of analysis is stock i on day t, not the tweet/post itself). We then reformat the date (from ISO format) to the same format date used in the price dataset to ensure that the two can be matched and merged later. We then group the rows by ticker and date and apply the following calculations:

- For sentiment score: we take the mean to have the daily average score for each stock.
- For the number of likes, the number of quotes, the number of replies, and the number of retweets: we take the sum to have the total number of likes, quotes, replies, and retweets a stock received on each day.
- For the upvote ratio (Reddit only): we take the mean to have the daily average value. As a note, the upvote ratio is the number of upvotes a post receives divided by the sum of upvotes and downvotes.
- 'post_count': we also created an additional field called 'post_count', which is the number of posts/tweets mentioning stock i on day t.

The final step is to simply merge the social media data frame and the price data frame by two columns: 'ticker' and 'date.' Figure 3 shows an example of the merged data frame that contains

[2] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.explode.html

daily information for the AACG stock (ATA Creativity Global). As a result, we now have two merged data frames—one for Reddit and one for Twitter ready for analysis.

	ticker	post_date	avg_sentiment_score	total_likes	total_quotes	total_replies	total_reweets	price	volume	sprea
63	AACG	02-04-2021	0.042818	58.0	2.0	6.0	5.0	14.30	315785055.0	16.300
64	AACG	02-05-2021	0.137333	14.0	1.0	5.0	5.0	6.33	18321709.0	4.519
65	AACG	06-29-2021	0.000000	1.0	0.0	0.0	0.0	3.08	111479.0	0.220
66	AACG	07-15-2021	0.096000	0.0	0.0	0.0	1.0	3.83	43776821.0	1.809

Figure 3. The merged data frame ready for analysis

2.3. Scoring Sentiment Analysis

After cleansing the dataset, we need to process every entry based on the sentiment of each word in a post via Natural Language Processing and score each post as one numerical feature to train our later model. For instance, one post might contain many positive words (e.g., Happy, Excellent and Fantastic, etc.), which means that the person who posts this is very confident that the stock to which his post relates will go up sooner or later. Therefore, the NLP algorithm might score this post as 0.9 (in our case, 1 is the most positive score, and 0 is the most negative score). Otherwise, if one post contains many negative words (e.g., Bad, Not Good, and Upset), this is likely indicating that this person predicts the relative stock will go down soon or that he has already lost a ton of money from it. Such types of posts will be given a low sentiment score.

The library we used for NLP is NLTK, which is a leading platform for building Python programs to work with human language data and it provides easy-to-use interfaces to over 50 corpora and lexical resources. Specifically, `SentimentIntensityAnalyzer` and the default corpus are mainly used. However, although NLTK is a powerful integrated library, it might not work very well for some specific cases because of many slang words used in the stock market (e.g., Diamond hand, Yolo and Short Squeeze, etc.). We can certainly create a customized corpus, but it is worth trying the standard one first. Surprisingly, the standard corpus does understand the stock market slang reasonably well. For example, as Table 1 shows, “Diamond hands” is scored 0.706, and “@elonmusk YOLO” is scored 0.739, which is higher than the score of “@blenderhd yolo.” This score difference is intuitive since Elon Musk is one of the most followed social media accounts in the present financial market. Therefore, we conclude that the default corpus understands contemporary Internet slang very well. Therefore, we decided to stick with this default corpus instead of customizing one since there is not enough reward that can be squeezed out from a customized corpus. Figure 4 shows what our dataset looks like after scoring, and we can see

there are some 0's due to those unreadable codes (emojis), but these 0's will not reduce our model's accuracy. Some of those 0's are worth keeping because those posts carry neutral text. However, those 0's because of unreadable codes will add unnecessary computational cost to our analysis (Refer to 2.2.4 for detail).

Table 1. Sentiment score given by NLTK for different contents

Posts	Scores
"Diamond hands"	0.706
"@blenderhd yolo"	0.677
"@elonmusk YOLO"	0.739

	author_id	created_a	tweet_id	like_count	quote_count	reply_count	retweet_	text	ticker	sentiment_scores
0	7.02E+17	2021-02-2	1.36E+18	2	0	0	0	Most who consider the GME shor	0	0.038
1	1.35E+18	2021-02-2	1.36E+18	0	0	0	0	@RyanSAdams forget diamond h	0	0.35
2	4.83E+09	2021-02-2	1.36E+18	0	0	0	0	omg wait do i post a moa selfie o	0	0
3	1.33E+18	2021-02-2	1.36E+18	0	0	0	0	AYDIA A MI PRIMO LE GUSTA YO	0	0.259
4	1.32E+18	2021-02-2	1.36E+18	1	0	0	0	que hace mi to moon selca en el t	0	0
5	1.32E+18	2021-02-2	1.36E+18	1	0	0	0	wiecie co wlosy odrosna a ja przy	0	0.11
6	1.89E+08	2021-02-2	1.36E+18	0	0	1	0	@motso_63 Inwele samma, YOL	0	0.486
7	88763246	2021-02-2	1.36E+18	0	0	1	0	@lowlearmonth Shal we jst do it	0	0.144
8	2.51E+09	2021-02-2	1.36E+18	1	0	0	0	@AYA__YOLO „Åä„ÇÅ„Åß„Å®„Å	0	0
9	1.9E+09	2021-02-2	1.36E+18	4	0	1	1	@StanTradingMan $UAVS big sh	[{'start	0
10	17727110	2021-02-2	1.36E+18	0	0	0	0	@danushman action during GME	0	0
11	4.67E+09	2021-02-2	1.36E+18	0	0	1	0	@trn04_yolo ÂõΩË™û„Åå„Çì„Å¬	0	0
12	9.30E+17	2021-02-2	1.36E+18	11	0	0	1	@Yolitatennis @Citizen1729 @D	0	0.174
13	4.67E+09	2021-02-2	1.36E+18	0	0	1	0	@trn04_yolo È†ëÂºµ„Å£„¶ÔºÅ	0	0
14	7.36E+17	2021-02-2	1.36E+18	0	0	0	5	RT @LifeNLasVegas: St Patrick's c	0	0
15	1.22E+18	2021-02-2	1.36E+18	1	0	0	0	You know.... There is no ugly to r	0	0.434
16	1.29E+18	2021-02-2	1.36E+18	2	1	1	0	Okay so here R the 3 simple strat	0	0.132
17	1.12E+18	2021-02-2	1.36E+18	2	0	0	0	@ebcapital Agree 100%, this stra	0	0.172
18	1.31E+18	2021-02-2	1.36E+18	0	0	1	0	Boys often concentrate in a few h	0	0.151
19	1.36E+18	2021-02-2	1.36E+18	0	0	0	0	@StonkstoMoon @MakingFrank	[{'start	0.365
20	7.21E+17	2021-02-2	1.36E+18	0	0	1	0	@theplanet_yolo ‡∏i‡∏≤‡°‡∏	0	0
21	1.26E+18	2021-02-2	1.36E+18	0	0	0	0	nombre yolo	0	0.677

Figure 4. Scored social media posts

3. Data Modeling Strategies

3.1. Random Forest, Logistic Regression, and Artificial Neural Network on the sample of stocks subject to the trading ban by Robinhood

Our first attempt to classify meme stocks is to rely on the ground truth. In the context of this study, we argue that a stock goes 'meme' if it faced a trading ban on the commission-free investing application, Robinhood, in the early 2021. The rationale for this argument is twofold. First, from the literature review, it is obvious that a 'mementum' is triggered and coordinated by a giant group of diffused, individual investors who, without access to a commission-free trading tool, might not be able to succeed in spearheading such a short squeeze. Second, for a fintech unicorn such as

Robinhood, implementing a trading ban is a very controversial and costly move that may cost the company its reputation as a pioneer in providing everyone with a fair, unlimited, and unexclusive access to the financial markets. As a consequence, we can expect that Robinhood does not take such a trading ban decision lightly and thus has to weigh the potential gains against its reputational cost very carefully. Indeed, the trading bans met with fierce opposition from millions of Robinhood investors, who believed that Robinhood had colluded with heavily shorted hedge funds to save the latter from further losses. Previous studies have also relied on these trading bans in building their samples of meme stock markets. For example, Aloosh et al. (2021) define the meme stock market as an equally weighted index of stocks whose purchase the Robinhood app restricted during the GameStop short squeeze episode.

We utilized the Robinhood stock ban list to define our ground truth of meme stocks during the period from January 27 to February 4, 2021. That is, if a stock was subject to the trading restrictions by Robinhood on one day between January 27 to February 4, 2021, we say it was a meme stock on that day. Otherwise, it was not considered a meme stock. Since the restrictions were implemented in multiple phases, the meme status of a stock, by definition, will vary from one day to the next during this period. However, because Robinhood had removed the list of restricted stocks from its official website, we could not get first-hand information and could only compile the ban list from the news media[3]. Figure 5 shows the complete list of stocks banned by Robinhood that we have assembled from multiple sources.

[3] The following sources were used to assemble the complete lists of stocks whose transactions were banned by Robinhood from January 27 to February 4, 2021: January 27th (Egan, 2021), January 28th (Fitzgerald, 2021a), January 29th (Fitzgerald, 2021b), February 1st (Li, 2021), February 2nd (Fitzgerald, 2021c), and February 4th (Murphy, 2021).

NO	Ticker	First Day	Last Day	NO	Ticker	First Day	Last Day	NO	Ticker	First Day	Last Day
1	AAL	28-Jan	31-Jan	18	GNUS	29-Jan	1-Feb	35	RLX	29-Jan	31-Jan
2	ACB	29-Jan	31-Jan	19	GTE	29-Jan	31-Jan	36	RYCEY	29-Jan	31-Jan
3	AG	29-Jan	31-Jan	20	HIMS	29-Jan	31-Jan	37	SBUX	29-Jan	31-Jan
4	AMC	27-Jan	2-Feb	21	INO	29-Jan	31-Jan	38	SHLS	29-Jan	31-Jan
5	AMD	29-Jan	31-Jan	22	IPOE	29-Jan	31-Jan	39	SIEB	29-Jan	31-Jan
6	BB	28-Jan	1-Feb	23	IPOF	29-Jan	31-Jan	40	SLV	29-Jan	31-Jan
7	BBBY	28-Jan	31-Jan	24	JAGX	29-Jan	31-Jan	41	SNDL	28-Jan	31-Jan
8	BYDDY	29-Jan	31-Jan	25	KOSS	28-Jan	1-Feb	42	SOXL	29-Jan	31-Jan
9	BYND	29-Jan	31-Jan	26	LLIT	29-Jan	31-Jan	43	SRNE	29-Jan	31-Jan
10	CCIV	29-Jan	31-Jan	27	MRNA	29-Jan	31-Jan	44	STPK	29-Jan	31-Jan
11	CLOV	29-Jan	31-Jan	28	MUX	29-Jan	31-Jan	45	TGC	29-Jan	31-Jan
12	CRIS	29-Jan	31-Jan	29	NAKD	28-Jan	2-Feb	46	TIRX	29-Jan	31-Jan
13	CTRM	28-Jan	31-Jan	30	NCTY	29-Jan	31-Jan	47	TR	28-Jan	31-Jan
14	EXPR	28-Jan	2-Feb	31	NOK	28-Jan	2-Feb	48	TRVG	28-Jan	31-Jan
15	EZGO	29-Jan	31-Jan	32	NVAX	29-Jan	31-Jan	49	WKHS	29-Jan	31-Jan
16	GM	29-Jan	31-Jan	33	OPEN	29-Jan	31-Jan	50	XM	29-Jan	31-Jan
17	GME	27-Jan	2-Feb	34	RKT	29-Jan	31-Jan	51	ZOM	29-Jan	31-Jan

Figure 5. The complete list of stocks banned by Robinhood in early 2021

With the stocks labeled using the Robinhood stock ban list, we need to perform some time series processing to generate features for supervised learning. For the social media variables, we calculate the mean values of the average variables like "avg_upvote_ratio" and the sum values of the total / count variables such as "post_count" during the selected time windows. For the market variables "price", "volume", and "spread ", we calculate the mean and standard deviation values during the selected time windows. The time windows selected include the previous day, the previous week, the previous two weeks, and the previous four weeks (or one month before). We split the labeled stocks into the training set and the testing set, where the testing size ratio is 0.2, for classifier training and testing after scaling the input features. To construct the classifiers, we adopt Random Forest, Logistic Regression, and ANN.

Random Forest is an algorithm that integrates multiple decision trees through the idea of ensemble learning. Its construction process is as follows. From the original training set, the bootstrapping method is used to randomly select m samples and perform $n_$trees times of sampling to generate $n_$trees training sets; For $n_$trees training sets, we train $n_$trees decision tree models separately. For a single decision tree model, we assume that the number of training sample features is n, then each split will select the best feature(s) for splitting according to a certain index. Each tree keeps splitting like that until all training examples for that node belong to the same class. No pruning is required during the splitting of the decision tree. The generated multiple decision trees

are grouped into a random forest. For classification problems, the final classification result is determined by the majority vote of multiple tree-classifiers.

Logistic regression is a generalized linear classifier to generate the probabilities of the predicted outcomes belonging to the binary classes. The purpose of logistic regression model is to minimize the error between the predicted value $\hat{y}$ and the true value y. Because $0 \leq \hat{y} \leq 1$, logistic regression model uses the sigmoid function $\left(\sigma(z) = \frac{1}{1+e^{-z}}\right)$ to translate the linear formulation $\hat{y} = w^T x + b$ to $\hat{y} = \sigma(w^T x + b)$. In fact, the sigmoid function is in the form of a parameterized logistic distribution (with $\mu = 0, \gamma = 1$), and the logistic regression model is exactly the following conditional probability distribution:

$$P(y = 1 \mid x) = \hat{y} = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

$$P(y = 0 \mid x) = 1 - \hat{y} = 1 - \sigma(w^T x + b) = \frac{e^{-(w^T x + b)}}{1 + e^{-(w^T x + b)}}$$

Finally, Artificial Neutral Network (ANN) is an extensive parallel interconnected network composed of adaptive simple units whose organization can simulate the biological nervous system's interactive response to real-world objects. The simplest neural network consists of three layers: an input layer i; a hidden layer j, and an output layer k. ANN is very flexible in the selection of hidden layers, activation function, and loss function. To construct a binary classifier, we would adopt the sigmoid function as the activation function for the output layer, and thus the simple unit (neuron) of the output layer would be the M-P neuron model.

3.2. A four-pronged application to detect meme stocks

A critical limitation of relying on the Robinhood trading bans to build our sample of meme stocks is that we end up with a rather small sample. Understandably, the trading ban was highly unpopular among Robinhood investors who accused the trading app to side with the "big guys" at the expense of the trading freedom which each individual investor on the platform was entitled to. Therefore, an important contribution of this study is our novel operationalization of identifying meme stocks. In particular, we develop the following four-pronged framework to determine whether a stock is experiencing a 'mementum' or not:

1) the stock must not be in the list of blue-chip stocks in 2021. In other words, it should not be included in the Standard & Poor's 500 Index.

2) the number of times it was mentioned in 2021 must be in the 90th percentile in our sample (at least 90% of the time).
3) the daily spread between its highest and lowest price in 2021 must be identified as an outlier.
4) there must be a co-movement detected between the stock's price and the sentiment score of discussions on the stock in social media (i.e., Twitter and Reddit).

The first two conditions are self-explanatory. The rationale for each of these conditions is as follows. First, a meme stock should not be a blue-chip one because our literature review has shown that the market price of a meme stock is not determined by any substantial improvements in the fundamentals or financial positions of the company. Birgersson and Carlén (2021) also find that S&P 500 stocks and meme stocks are fundamentally different from each other. We retrieve the list of 505 stocks that made up the S&P 500 as of December 31st, 2021 from Bloomberg (Figure 6). Second, the stock's popularity must be at least in the 90th percentile because being trendy on social media is the critical (if not only) driver sending the price of the meme stock "to the moon" (to borrow the lexicon of the meme stock trading community). Also, by design, all stocks in the sample were mentioned at least once, so setting the threshold at median (aka, 50%) is not robust. We vary the time window to compute the popularity of each stock relative to the whole sample on: the previous day $(t-1)$, the previous week $(t-7)$, the previous two weeks $(t-14)$, and the previous month, to test the sensitivity of our results to different time windows. Thirdly, for condition 3, we compute the Z-score to determine whether stock i's spread on day t is considered an outlier or not, relative to the Z-score of spread of all stocks on: the previous day $(t-1)$, the previous week $(t-7)$, the previous two weeks $(t-14)$, and the previous month. Following statistical conventions, we argue that the spread of a stock on a day is an outlier if its Z-score is greater than or equal to 3 (i.e., more than 3 standard deviations away from the mean should be identified as an outlier).

Lastly, we need a co-movement detection algorithm to check condition 4. To achieve this aim, we experiment with two different methods: the Change Point Detection algorithm (Schroth, Siebert, & Groß, 2021) and the Wavelet Coherence Analysis. The details of these two methods are presented below.

Figure 6. Member stocks in the S&P 500 as of December 31, 2021

3.2.1. Change Point Detection

In statistics, Change Point Detection (CPD) or change point analysis is a subset of anomaly detection problems. CPD aims to identify abrupt and significant changes in a time-series data. These models not only recognize if there is one (or more) significant change(s) but they also pinpoint when these changes occur. "Changes" here may refer to the variations in the mean of a time series, changes in the spread (variance), change in periodicity, or a combination of these types and more. Figure 7 illustrate a few examples of the types of changes measured by CPD models.

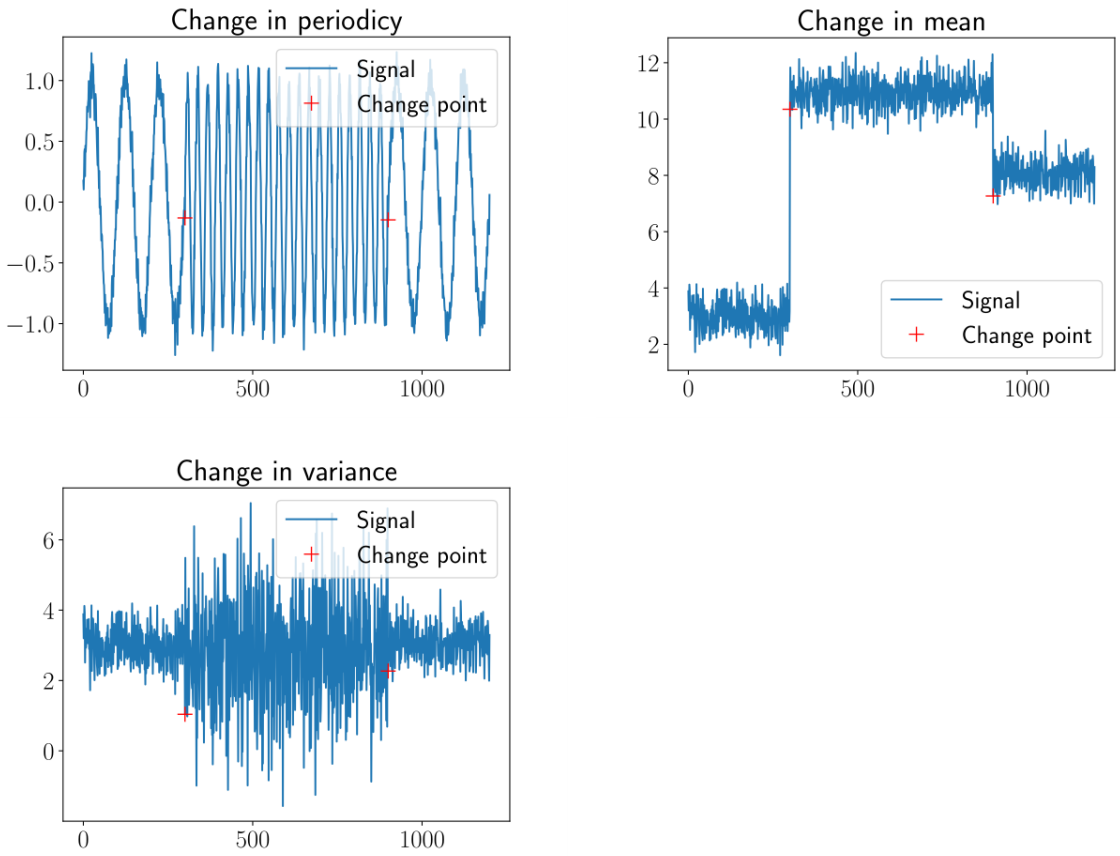

Figure 7. Different types of changes that can be detected by CPD (Source: https://www.iese.fraunhofer.de/blog/change-point-detection)

CPD problems are categorized in two types: *online* and *offline.* Offline analysis refers to problems where the whole data set is available. The algorithm identifies where the changes occurred in the past. Offline CPD is often considered post-hoc analysis. On the other hand, online analysis is concerned with detecting these changes as the observations are happening, such as detecting that a machine is failing in order to notify the mechanics as soon as possible. In the scope of this paper, we will focus on offline analysis. However, future research on "meme" stocks could be based on online CPD.

Barry and Hartigan (1993) introduced a specific way of measuring CPD, using the Bayesian approach. This method assumes that a sequence of observations may be divided into non-overlapping states partitions. The prior distribution of the mean of the block begins at position $i+1$ and ends at j, $\mu_{i,j}$, based on $\mathcal{N}\left(\mu_0, \frac{\sigma_0^2}{j-i}\right)$. A Markov chain is evaluated at each position i, where a value U_i is drawn from a conditional distribution given the partition of the data and the current partition. Recalling that each partition has a likelihood of being a change point. The

transition probability, p, for the conditional probability of a change point at the position $i+1$ is computed with the following ration:

$$\frac{p_i}{1-p_i} = P(U_i = 1 \,|\boldsymbol{X}, U_j, j \neq i) / \, P(U_i = 0 \,|\, \boldsymbol{X}, U_j, j \neq i)$$

Where $\boldsymbol{X}$ is the data, and U_i can be 0 or 1. This is equivalent to

$$\left(\frac{W_0}{W_1}\right)^{\frac{n-b-2}{2}} \cdot \left(\frac{B_0}{B_1}\right)^{\frac{b+1}{2}} \cdot \sqrt{\frac{W_1}{B_1}} \cdot \frac{\int_0^{\frac{\frac{B_1\lambda}{W_1}}{1+\frac{B_1\lambda}{W_1}}} p^{\frac{b+2}{2}}(1-p)^{\frac{n-b-3}{2}}\,dp}{\int_0^{\frac{\frac{B_0\lambda}{W_0}}{1+\frac{B_0\lambda}{W_0}}} p^{\frac{b+1}{2}}(1-p)^{\frac{n-b-2}{2}}\,dp} \cdot \frac{\int_0^{\gamma} p^b(1-p)^{n-b-1}dp}{\int_0^{\gamma} p^{b-1}(1-p)^{n-b}\,dp}.$$

where W_i and B_i are the within and between block sum of squares respectively. The tuning parameters γ and λ can take values from 0 and 1, depending on the specific needs of the model. Interested readers are encouraged to consult Erdman and Emerson (2008) for technical details. With this background, we will develop a CDP model using the social media data and use these detections as a proxy for labeling stocks as "meme".

3.2.2. Wavelet Coherence Analysis

Wavelet data mining techniques are an effective analysis methodology to understand various behaviors across a time series. Since stock variation is often nonlinear, wavelet coherence can be implemented to understand the rate of co-movement between stock variables that other methodologies fail to capture. It is especially effective at capturing behaviors, or signals, during a short oscillatory period in the data. In the case of meme stocks, their rises and falls often happen quite quickly, resulting in sharp nonlinear changes in the behavior of their variables. Wavelet analysis can take advantage of these events and use them to classify meme-stocks by identifying and scoring these events. The co-movement between a stock's sentiment and price across the time series of the dataset serves as the basis for the wavelet coherence analysis.

To perform the classification across the stocks in the dataset, we perform wavelet analysis using the Twitter and Reddit sentiment scores separately. First, we normalize the post entries to a uniform time series across 52 weeks of 2021. Next, the stock tickers with fewer than 9 post entries are removed from the analysis, as the Wavelet modeling does not have enough information to generate a good model. Then we implement the "mother" Morlet wavelet:

$$\psi(t) = \pi^{-\frac{1}{4}} e^{i\omega t} e^{-\frac{t^2}{2}}$$

which serves as the basis for the wavelet analysis. Then, we use the mother wavelet to generate a series of "daughters" with a translation and scaling factors shown as:

$$\text{Wave}(\tau, s) = \sum_t x_t \frac{1}{\sqrt{s}} \psi^* \left(\frac{t-\tau}{s}\right)$$

This equation constitutes the building block which the wavelet coherence is built upon. Wavelet coherence compares the signals of two factors by creating a composite of both input waves. Then the power of the composite wave is normalized and compared to the power of the two input waves to generate a coherence scoring value. For the purposes of this analysis, a strong coherence value across the time series indicates a strong correlation between price movement and sentiment movement and can identify cases of co-movement between the two. These values can be applied to all the stocks to classify meme stocks and will be used in the discussion and analysis of results.

4. Data Modeling/Analysis Results

4.1. RF, LR, and ANN

Since we have a heavily imbalanced dataset (2424:60), the accuracy could not be the proper metrics to evaluate the performance since by design, a classifier can obtain a high accuracy simply by predicting all points as 0. Therefore, we adopt the macro averaged $f1$-score as the main metric of accuracy. We realize that the macro average treats all classes equally while our main interest is in class 1 (meme stocks). Since the $f1$-score of the not-meme class is always near 1 because of the class size imbalance, the macro averaged $f1$-score can still meet our need.

We used Python function `StandardScaler` from the package `sklearn.preprocessing` to scale the input varaibles and the Python function `train_test_split` from the package `sklearn.model_selection` to reserve 20% of the dataset for testing purposes. For Random Forest, we adopted the Python function `RandomForestClassifier` from package '`sklearn.ensemble`'. We first set the parameters of the models as follows: n_estimators = 300, min_samples_leaf = 10, criterion = 'entropy'. The performances of the model on the training set and the testing set are shown below.

```
Random Forest on Train Set:
Predicted   0.0  1.0
Actual
0.0        2419    5
1.0          31   29
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      2424
         1.0       0.85      0.48      0.62        60

    accuracy                           0.99      2484
   macro avg       0.92      0.74      0.80      2484
weighted avg       0.98      0.99      0.98      2484

Accuracy: 0.9855072463768116
Cohen's Kappa: 0.6102102730411131
R^2 0.38514851485148527
precision for meme class 0.8529411764705882
recall for meme class 0.48333333333333334
```

```
Random Forest on Test Set:
Predicted  0.0  1.0
Actual
0.0        605    2
1.0          9    6
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       607
         1.0       0.75      0.40      0.52        15

    accuracy                           0.98       622
   macro avg       0.87      0.70      0.76       622
weighted avg       0.98      0.98      0.98       622

Accuracy: 0.9823151125401929
Cohen's Kappa: 0.5135788425991753
R^2 0.2485447556287752
precision for meme class 0.75
recall for meme class 0.4
```

We adopted the Python function `GridSearchCV` from package '`sklearn.model_selection`' to tune the hyperparameters for Random Forest using 5-fold cross validation and the macro averaged $f1$-score as scoring metrics. The tuned parameters are: n_estimators = 100, min_samples_leaf = 1, criterion = 'entropy'. The performances of the model on the training set and the testing set are shown below.

```
Tuned Random Forest on Train Set:
Predicted   0.0  1.0
Actual
0.0        2424    0
1.0           0   60
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2424
         1.0       1.00      1.00      1.00        60

    accuracy                           1.00      2484
   macro avg       1.00      1.00      1.00      2484
weighted avg       1.00      1.00      1.00      2484

Accuracy: 1.0
Cohen's Kappa: 1.0
R^2 1.0
precision for meme class 1.0
recall for meme class 1.0
```

```
Tuned Random Forest on Test Set:
Predicted  0.0  1.0
Actual
0.0        605    2
1.0          9    6
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       607
         1.0       0.75      0.40      0.52        15

    accuracy                           0.98       622
   macro avg       0.87      0.70      0.76       622
weighted avg       0.98      0.98      0.98       622

Accuracy: 0.9823151125401929
Cohen's Kappa: 0.5135788425991753
R^2 0.2485447556287752
precision for meme class 0.75
recall for meme class 0.4
```

For Logistic Regression, we adopted the Python function `LogisticRegressionCV` from package '`sklearn.linear_model`'. We trained the model with 5-fold cross validation and l1 regularization penalty. The performances of the model on the training set and the testing set are shown below.

```
Logistic Regression on Train Set:
Predicted   0.0  1.0
Actual
0.0        2421    3
1.0          30   30
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      2424
         1.0       0.91      0.50      0.65        60

    accuracy                           0.99      2484
   macro avg       0.95      0.75      0.82      2484
weighted avg       0.99      0.99      0.98      2484

Accuracy: 0.9867149758454107
Cohen's Kappa: 0.6389725701601395
R^2 0.43638613861386144
precision for meme class 0.9090909090909091
recall for meme class 0.5
```

```
Logistic Regression on Test Set:
Predicted  0.0  1.0
Actual
0.0        603    4
1.0          8    7
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       607
         1.0       0.64      0.47      0.54        15

    accuracy                           0.98       622
   macro avg       0.81      0.73      0.76       622
weighted avg       0.98      0.98      0.98       622

Accuracy: 0.9807073954983923
Cohen's Kappa: 0.5288473677565964
R^2 0.18023064250411847
precision for meme class 0.6363636363636364
recall for meme class 0.4666666666666667
```

Logistic regression generates the probability Pr(stock X goes meme), and the default predicted classes of Logistic regression use 0.5 as the threshold probability. To explore the robustness of logistic regression with different thresholds, we calculate the macro averaged $f1$-score on the training set and the testing set using different thresholds. The threshold that has the highest macro averaged $f1$-score on the training set is 0.3, and the one that has the highest macro averaged $f1$-score on the testing set is 0.65. This difference suggests that the trained model tends to have the False Negative results on the testing set if we use the default threshold of 0.5. The performance of the Logistic Regression models with threshold 0.3 and 0.65 on the training set and the testing set are shown below.

```
Logistic Regression on Train Set with threshold 0.3 :
Predicted     0   1
Actual
0.0        2413  11
1.0          23  37
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      2424
         1.0       0.77      0.62      0.69        60

    accuracy                           0.99      2484
   macro avg       0.88      0.81      0.84      2484
weighted avg       0.99      0.99      0.99      2484

Accuracy: 0.9863123993558777
Cohen's Kappa: 0.6782775644541964
R^2 0.41930693069306946
precision for meme class 0.7708333333333334
recall for meme class 0.6166666666666667
```

```
Logistic Regression on Test Set with threshold 0.3 :
Predicted    0  1
Actual
0.0        599  8
1.0          8  7
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       607
         1.0       0.47      0.47      0.47        15

    accuracy                           0.97       622
   macro avg       0.73      0.73      0.73       622
weighted avg       0.97      0.97      0.97       622

Accuracy: 0.9742765273311897
Cohen's Kappa: 0.45348709500274575
R^2 -0.09302580999450871
precision for meme class 0.4666666666666667
recall for meme class 0.4666666666666667
```

```
Logistic Regression on Train Set with threshold 0.65 :
Predicted     0   1
Actual
0.0        2423   1
1.0          34  26
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      2424
         1.0       0.96      0.43      0.60        60

    accuracy                           0.99      2484
   macro avg       0.97      0.72      0.80      2484
weighted avg       0.99      0.99      0.98      2484

Accuracy: 0.9859098228663447
Cohen's Kappa: 0.5915778792491122
R^2 0.40222772277227736
precision for meme class 0.9629629629629629
recall for meme class 0.43333333333333335
```

```
Logistic Regression on Test Set with threshold 0.65 :
Predicted    0  1
Actual
0.0        607  0
1.0          9  6
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       607
         1.0       1.00      0.40      0.57        15

    accuracy                           0.99       622
   macro avg       0.99      0.70      0.78       622
weighted avg       0.99      0.99      0.98       622

Accuracy: 0.9855305466237942
Cohen's Kappa: 0.5654401490451793
R^2 0.3851729818780888
precision for meme class 1.0
recall for meme class 0.4
```

For ANN, we adopt the Python function `keras.models.Sequentia` from package '`tensflow`'. We used the `ReLu` function as the activation function of the hidden layers and the sigmoid function as the activation function of the output layer. The performances of the model on the training set and the testing set are shown below.

```
ANN on Train Set with threshold 0.5 :
Predicted     0   1
Actual
0.0        2420   4
1.0          17  43
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00      2424
         1.0       0.91      0.72      0.80        60

    accuracy                           0.99      2484
   macro avg       0.95      0.86      0.90      2484
weighted avg       0.99      0.99      0.99      2484

Accuracy: 0.9915458937198067
Cohen's Kappa: 0.7994833710041976
R^2 0.6413366336633664
precision for meme class 0.9148936170212766
recall for meme class 0.7166666666666667
```

```
ANN on Test Set with threshold 0.5 :
Predicted    0  1
Actual
0.0        601  6
1.0          8  7
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       607
         1.0       0.54      0.47      0.50        15

    accuracy                           0.98       622
   macro avg       0.76      0.73      0.74       622
weighted avg       0.98      0.98      0.98       622

Accuracy: 0.977491961414791
Cohen's Kappa: 0.48854692822741685
R^2 0.04360241625480488
precision for meme class 0.5384615384615384
recall for meme class 0.4666666666666667
```

Our binary classifier ANN model also generates the decimal which are between 0 and 1 as the predicted values with 0.5 being the threshold of the decimal. To explore its performance with different thresholds, we calculate the macro averaged $f1$-score on the training set and the testing set, using different thresholds. The threshold that has the highest macro averaged $f1$-score on the training set is 0.25, and the one that has the highest macro averaged $f1$-score on the testing set is 0.85. This difference suggests that the trained ANN model tends to have the False Negative results

on the testing set if we use the default threshold of 0.5. The performance of the ANN models with threshold 0.25 and 0.85 on the training set and the testing set are shown below.

```
ANN on Train Set with threshold 0.25 :
Predicted     0   1
Actual
0.0        2414  10
1.0           9  51
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2424
         1.0       0.84      0.85      0.84        60

    accuracy                           0.99      2484
   macro avg       0.92      0.92      0.92      2484
weighted avg       0.99      0.99      0.99      2484

Accuracy: 0.9923510466988728
Cohen's Kappa: 0.8390555305479396
R^2 0.6754950495049505
precision for meme class 0.8360655737704918
recall for meme class 0.85
```

```
ANN on Test Set with threshold 0.25 :
Predicted    0   1
Actual
0.0        591  16
1.0          8   7
              precision    recall  f1-score   support

         0.0       0.99      0.97      0.98       607
         1.0       0.30      0.47      0.37        15

    accuracy                           0.96       622
   macro avg       0.65      0.72      0.67       622
weighted avg       0.97      0.96      0.97       622

Accuracy: 0.9614147909967846
Cohen's Kappa: 0.3494290943955374
R^2 -0.6395387149917631
precision for meme class 0.30434782608695654
recall for meme class 0.4666666666666667
```

```
ANN on Train Set with threshold 0.85 :
Predicted     0   1
Actual
0.0        2424   0
1.0          21  39
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00      2424
         1.0       1.00      0.65      0.79        60

    accuracy                           0.99      2484
   macro avg       1.00      0.82      0.89      2484
weighted avg       0.99      0.99      0.99      2484

Accuracy: 0.9915458937198067
Cohen's Kappa: 0.7837636173705417
R^2 0.6413366336633664
precision for meme class 1.0
recall for meme class 0.65
```

```
ANN on Test Set with threshold 0.85 :
Predicted    0  1
Actual
0.0        607  0
1.0          8  7
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       607
         1.0       1.00      0.47      0.64        15

    accuracy                           0.99       622
   macro avg       0.99      0.73      0.81       622
weighted avg       0.99      0.99      0.98       622

Accuracy: 0.9871382636655949
Cohen's Kappa: 0.630696155558854
R^2 0.45348709500274564
precision for meme class 1.0
recall for meme class 0.4666666666666667
```

We summarize the performances on the testing set of the above models in Table 2. Using macro averaged $f1$-score as the evaluation metrics, the best model would be ANN with 0.85 as the threshold. With the threshold increasing, the precision of both Logistic Regression and ANN increases, and the recall ratio slightly decreases to the Random Forest's meme class recall level for Logistic Regression while remaining unchanged for ANN. Finally, we note an important tradeoff in the accuracy offered by different methods. For instance, because investing in meme stocks is a speculation strategy that is considered highly risky, a risk-averse investor may prefer that the predicted meme stock is indeed a meme stock. In that case, we can use the LR and ANN models with up-adjusted thresholds or use the RF model. On the other hand, a risk-loving investor

may care more about not missing out on any meme stocks. In that case, RF and LR with high threshold values do not seem to be the best choice.

Table 2. Summary of performances of RF, LR, and ANN on the testing set

Method	Threshold	TP	FN	FP	TN	meme class			meme & not-meme class
						precision	recall	f1-score	macro avg f1-score
Random Forest	-	605	2	9	6	0.75	0.40	0.52	0.76
Logistic Regression	0.3	599	8	8	7	0.47	0.47	0.47	0.73
	0.5	603	4	8	7	0.64	0.47	0.54	0.76
	0.65	607	0	9	6	1.00	0.40	0.57	0.78
ANN	0.25	591	16	8	7	0.30	0.47	0.37	0.67
	0.5	601	6	8	7	0.54	0.47	0.50	0.74
	0.85	607	0	8	7	1.00	0.47	0.64	0.81

4.2. Change Point Detection

There are many R packages that perform different versions of CPD. A few examples include *segmented, changepoint, strucchange, mcp, cpm,* and *bcp*. Each package provides users with specific modeling flexibility and has different performance metrics. A summary of the pros and cons of these packages is provided by Otto (2019).

For this paper, we will be using *bcp* (Erdman & Emerson, 2008). This package is based on the work of Barry and Hartigan (1993), one of the few CPD packages designed using a Bayesian approach. It returns the posterior probability of a change point occurring at each time index in the series. Below is a snippet of scripts and the plot of a *bcp* model (Figure 8). Specifically, the red dots are the estimates for the average sentiment for the AMC (American movie theater chain AMC Entertainment Holdings, Inc) stock.

```
R < library(bcp);
R < set.seed(123)
R < x <- twitter[twitter$ticker == 'AMC',]$avg_sentiment_score
R < bcp_x <- bcp(x)
R < plot(bcp_x, main=("AMC Avg Sentiment Score (Twitter)\nPosterior mean and
probability of change", xlab="Date Index"))
```

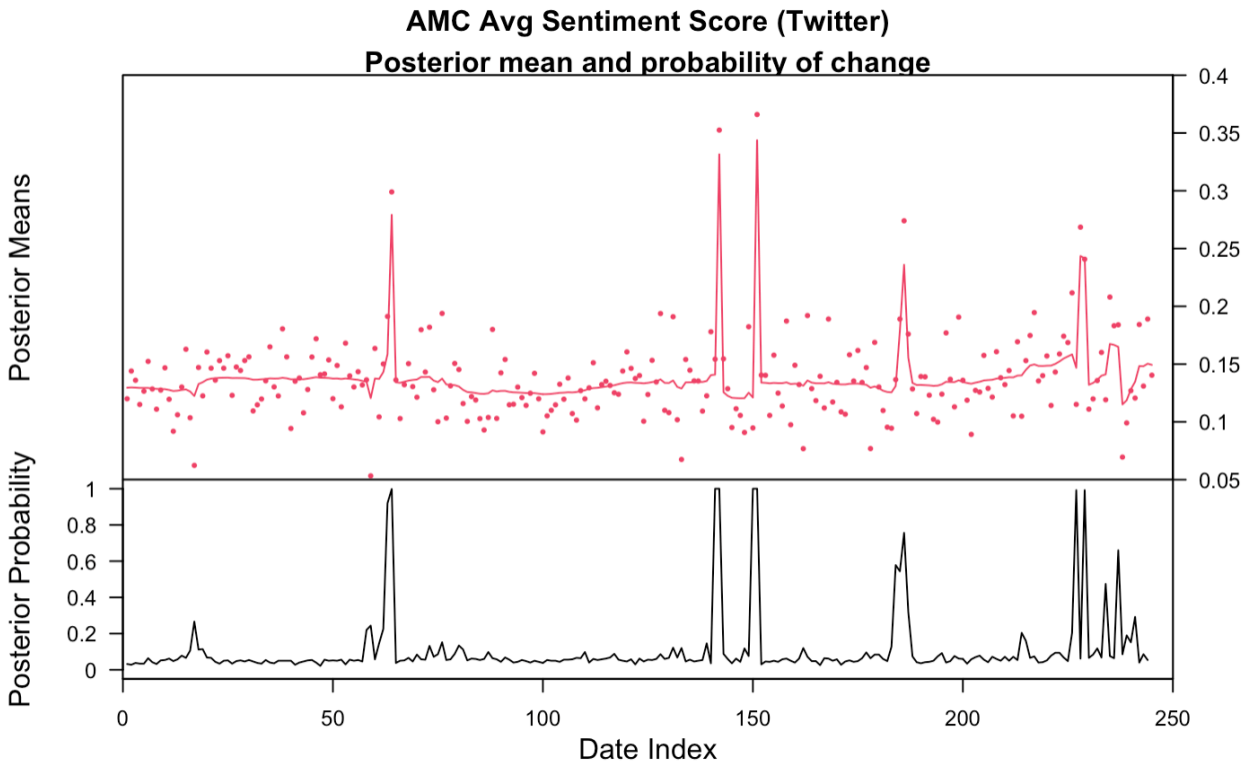

Figure 8. The posterior probability of a change point, produced by the 'bcp' model

The black line graph in the bottom represents the posterior probability of change for each data point. The x-axis of the graph, Date Index, represents continuous dates in the time series. For simplicity, the off-trading days, like weekends and holidays, are excluded from our sample. As a result, we assume that a Friday and the next Monday are consecutive. This method allows us to dynamically change the cutoffs for posterior probability. For example, the following script sets the cutoff at 0.85.

```
bcp_sum <- as.data.frame(bcp_x$posterior.prob)
bcp_sum$data <- x
bcp_sum$id <- 1:length(x);
sel <- bcp_sum[which(bcp_x$posterior.prob > 0.85),]
print(sel)
```

With this specific cutoff, we detect the following change points for AMC Theatres as follows (Table 3). Again, the change points below show where the mean of the data series experiences a dramatic change. Note that the prior probabilities are rounded to the nearest hundredth.

Table 3. Detected changes in AMC Avg. Sentiment Score (Twitter)

Index	Date	Avg. Sentiment Score	Prior Probability of Change
63	4/14/21	0.191	0.92
64	4/15/21	0.299	0.99
141	8/4/21	0.154	1.00
142	8/5/21	0.353	1.00
150	8/17/21	0.095	1.00
151	8/18/21	0.366	1.00
227	12/6/21	0.115	0.99
229	12/8/21	0.241	0.99

The same analysis was repeated on a 7-day moving average *MA(7)* on the average sentiment data. When using a moving average, the random noise of the dataset is significantly reduced, therefore the model perceives changes as more abrupt and categorizes them as change points. As we can see in Figure 9, MA(7) detects more change points than the daily example above. The result is that the posterior mean (red line) is significantly "bumpier". For this reason, no moving averages would be considered for our final categorization of meme stocks.

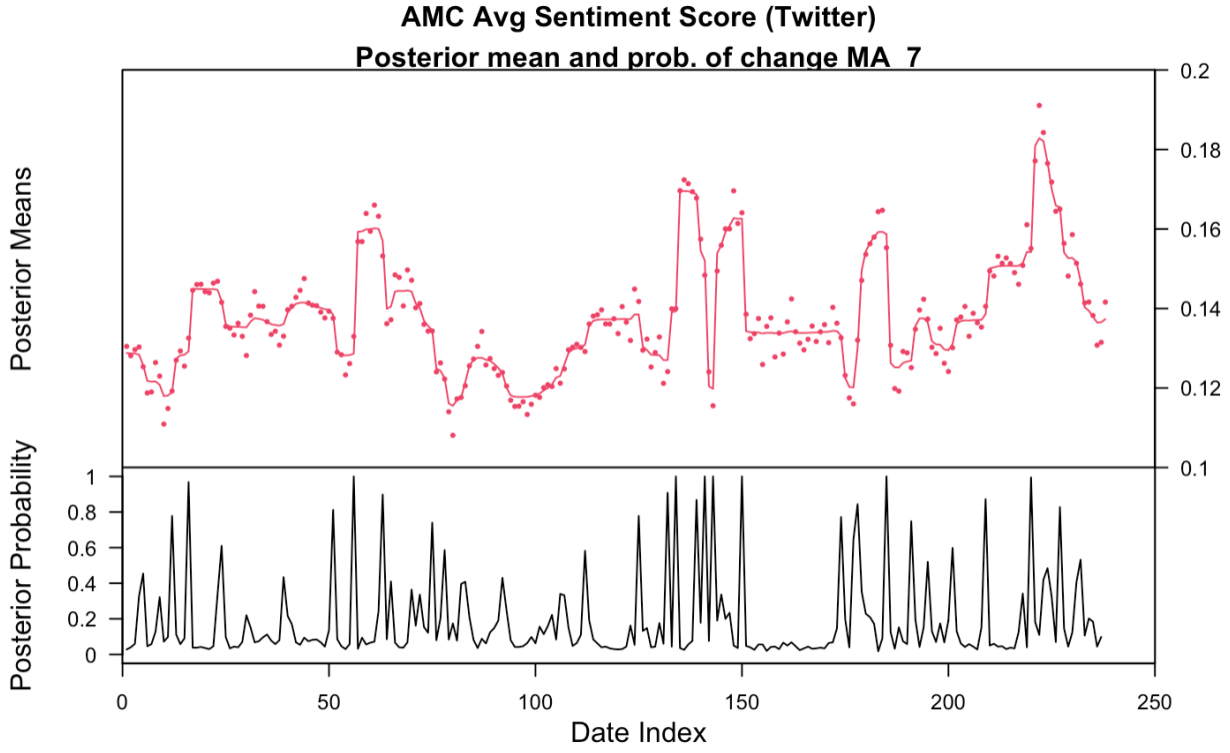

Figure 9. Posterior probability of changes using 7-day moving average

We ran a *bcp* model for every stock in the Twitter and Reddit datasets. A few tickers were excluded based on the lack of observations. As has been obvious from the literature review, the

detection of meme stocks largely depends on the aperiodic changes in the sentiment in social media and how frequently the stocks were mentioned. Therefore, the time series attributes selected for modeling are the average sentiment score and post count for the day (post volume). These two measurements were combined to ultimately label a ticker as a meme stock.

The model proved to be very sensitive to changes in the cutoff parameters. Small changes in the cutoff would include an increasingly number of tickers/days. The cutoffs parameters for each model were selected based on the maximum prior probability of change in 2021 of the stocks that are widely accepted as "meme stocks", like GameStop and AMC. We used the 90^{th} percentile max prior probability for the sentiment score and post volume. For Twitter and Reddit, the cutoff was set as 0.864 for average sentiment prior and 0.994 for post count. This yielded a good balance between over/under classifying each stock.

4.3. Wavelet Coherence Analysis

To perform Wavelet Coherence Analysis, we utilized the `WaveletComp` package in R. Following the methodology outlined in Section 3.2.2, the data for both the Reddit and Twitter posts were fit to apply the wavelet coherence methods. Using the `analyze.coherency()` function, every stock in both datasets was measured across a uniform time series representing the time window our data spans. Through the selection process, those stocks with a p-value of coherency above 0.7 were recorded. The results are shown in Table 4 below for the classification of both Twitter and Reddit data. We recall that co-movement coherence refers to the observed signal association between price and sentiment movement.

Table 4. Stocks whose coherence co-movement is greater than 0.7

Twitter	Reddit
"ADM", "AZRX", "BFRI", "CDE", "CFMS", "CLEU", "CRU", "CTT", "DS", "EEIQ", "FAN", "FCFS", "FNKO", "FSLR", "GENI", "LYFT", "MDGS", "MGM", "MOS", "PBTS", "PGEN", "POWW", "PRPO", "SINT", "TARA", "TIP", "TOPS", "USWS", "WEI", "XEC"	"AEZS", "APPH", "BIO", "CRCT", "DBX", "GET", "GOGO", "HIMS", "NYT", "OPAD", "POSH", "TEAM"

We observe several interesting patterns from these results. The most notable insight is that none of the "meme" classifications from Twitter overlaps with the classifications from the Reddit dataset. Additionally, these classifications do not align with other methodological classifications of meme stocks. While not initially intuitive, this behavior can be explained. If we examine the

wavelet coherence plot of the well-known meme-stock GME (Figure 10), we can observe the tight patches of co-movement between price and sentiment (i.e., the hot regions).

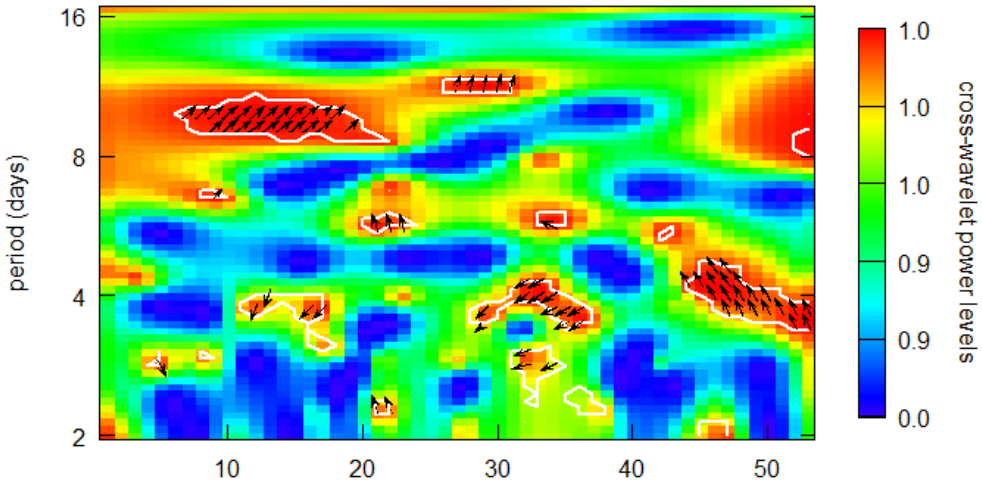

Figure 10. Wavelet Coherence plot of GameStop (GME)

The largest red patch in the upper left corner clearly shows a positive co-movement between the sentiment and price of GME in 2021, and this observation aligns with the initial boom of GME as a meme stock. As GME's price reached its peak, Twitter users used highly positive language with high sentiment values about the trading of GME. The upwards pointing arrow also indicates that price is leading sentiment, suggesting that the price has been increasing prior to the increase of people's opinions about GME. Later in the plot, we can see in quicker intervals that the sentiment is leading the price of the stock, such as around weeks 30-37. These smaller phases indicate that the two are in anti-phase, meaning that sentiment is increasing as price is decreasing. These smaller phases can be explained by traders “pumping” the stock's social media presence or encouraging others to buy the stock to combat the price drop and inflate the value of their shares. Later towards the end of 2021, we observe a high correlation between sentiment and price, with price leading sentiment where sentiment and price have an anti-phase relationship, explaining the eventual decline of GME's popularity corresponding with its price tapering off. In short, from one wavelet coherence plot, we can observe many complex interactions across the time series and understand the history of the stock's behavior. Similar observations can be drawn from every stock's plot in the dataset and can provide a deeper understanding of variable co-movement.

When it comes to classification using wavelet analysis, the outcomes observed from the graph become more difficult to quantify. In the case of the graph above for GME, we can observe

that sentiment and price do not always have a strong coherence, and only during certain portions can a relationship be pulled from the results. When creating a classification decision, the overall coherence of the entire dataset is considered, which explains why many of the well-known meme stocks were not selected by this method. Still, wavelet coherence remains an impressive method of visualizing the nuisances of variable and phase relationships and can help us mine a deeper understanding of price and sentiment co-movement within stock data.

5. Model Property Assessment

5.1. Model Evaluation

In this section, we compare the predicted "meme stock or not" results generated by Random Forest/ Logistic Regression/ANN and the predicted results generated by our own four-pronged detection method.

5.1.1. Prediction quality in the Reddit data set

To compare three machine learning methods applied in the Reddit dataset, we calculate the accuracy, which indicates the consistency of prediction results between two different methods. Among the three methods, ANN shows more similar prediction results to our four-pronged framework, with an accuracy of about 97%, whereas logistic regression results with an accuracy of 95% are less consistent with our own framework. Moreover, the results reveal that including change point detection to our criteria makes the framework more stringent, as can be seen from the fact that fewer stocks are qualified as 'meme stocks' once we add the fourth criterion based on the change point detection. In fact, the four-pronged framework identifies only one stock as meme, and this stock was not considered a meme stock in the other three machine learning methods. When excluding the change point detection (i.e., the fourth criterion) from our framework, more stocks are identified as meme. Take the result of ANN as an example. The framework without the change point detection and ANN identify 17 stocks as meme stocks at the same time. The other machine learning methods contrast starkly with the results yielded by our framework: specifically, RF and LR label more than 300 stocks as meme stocks.

- Compare tuned RF with our four-pronged framework:

Four-pronged Framework With Change Point Detection (Period = 7 and 14 days)

Tuned RF	0	1
0	12656	1
1	490	0

Accuracy = 0.962653

Four-pronged Framework Without Change Point Detection (Period = 7 days)

Tuned RF	0	1
0	12624	33
1	459	31

Accuracy = 0. 962577

- Compare LR with our four-pronged framework:

Four-pronged Framework With Change Point Detection (Period = 7 days)

LR th=0.65	0	1
0	12499	1
1	647	0

Accuracy = 0.950711

Four-pronged Framework Without Change Point Detection (Period = 7 days)

LR th=0.65	0	1
0	12480	20
1	603	44

Accuracy = 0. 952613

- Compare ANN with our four-pronged framework:

Four-pronged Framework With Change Point Detection (Period = 7 days)

ANN th=0.85	0	1
0	12825	1
1	321	0

Accuracy = 0.975508

Four-pronged Framework Without Change Point Detection (Period = 7 days)

ANN th=0.85	0	1
0	12779	47
1	304	17

Accuracy = 0. 973302

5.1.2. Prediction quality in the Twitter data set

Similarly, we compare three machine learning methods applied in the Twitter dataset by calculating their accuracy. We come to the same conclusion as obtained from the Reddit dataset. That is, among the three methods, ANN shows more similar prediction results to our four-pronged framework, with an accuracy of about 98%, whereas logistic regression exhibited the lower consistency with our framework, with an accuracy of 97%. Despite these differences, overall, each machine learning method shows higher consistency with our own framework in the Twitter than in the Reddit dataset.

Likewise, the four-pronged method in the Twitter dataset also demonstrates its stringency. It identifies only five stocks as meme, and only one of the stocks was also considered a meme stock in the three machine learning methods. When excluding the change point detection from our framework, 88 stocks were labeled as meme, slightly more than the results in the Reddit dataset. However, the number of meme stocks co-labeled by both methods is about the same. Again, take the result of ANN as an example. The framework without the change point detection and ANN identify 16 stocks as meme stocks at the same time. The other machine learning methods contrast starkly with our framework, labeling more than 500 stocks as meme stocks.

- Compare tuned RF with our four-pronged framework:

Four-pronged Framework With Change Point Detection (Period = 1, 7, and 30 days)

Tuned RF	0	1
0	30827	4
1	562	1

Accuracy = 0.981971

Four-pronged Framework Without Change Point Detection (Period = 7 days)

Tuned RF	0	1
0	30767	64
1	539	24

Accuracy = 0. 980793

- Compare LR with our four-pronged framework:

Four-pronged Framework With Change Point Detection (Period = 7 days)

LR th=0.65	0	1
0	30499	4
1	890	1

Accuracy = 0.971523

Four-pronged Framework Without Change Point Detection (Period = 7 days)

LR th=0.65	0	1
0	30462	41
1	844	47

Accuracy = 0. 971810

- Compare ANN with our four-pronged framework:

Four-pronged Framework With Change Point Detection (Period = 7 days)

ANN th=0.85	0	1
0	30896	4
1	493	1

Accuracy = 0.984169

Four-pronged Framework Without Change Point Detection (Period = 7 days)

ANN th=0.85	0	1
0	30828	72
1	478	16

Accuracy = 0. 982481

5.1.3. Comparison of model performances on the two data sets

To summarize the salient features of our analysis, several findings are of interest:

1. Compared with prediction in the Reddit data set, the prediction in the Twitter data set has a higher overall accuracy, possibly for reasons of a larger sample size and the inherent characteristics of tweets and Reddit posts.

2. The four-pronged framework, especially the one including the change point detection, tends to err on the side of caution and labels just a few meme stocks. For example, only one stock was considered as a meme stock by this method in the Reddit data set. Granted, this finding contradicts the fact that there are more than one stocks that were banned by Robinhood in early 2021. Future studies may explore other methods of co-movement detections or re-evaluate whether change point detection is a proper method in this context. Another possible explanation is that the market needs some time to reflect the changes in the social media sentiment on a particular stock. Relaxing the one-to-one mapping condition in change point detection to reflect such time lags may allow for more stocks to qualify as meme stocks.

3. When implementing our four-pronged framework, we experiment with different time windows in computing the popularity and spread of the stocks, namely 1 (the previous day), 7 (the week before), 14 (two weeks before), and 30 days (the month before). The results suggest that computing the popularity and spread of each stock relative to the whole sample in the previous week ($t-7$) yields the most accurate results.

4. In terms of the consistency among our four-pronged framework and conventional machine learning methods, all comparisons indicate a high consistency with at least 95% accuracy. We can conclude with certainty that the four-pronged framework we develop and conventional supervised ML methods can serve as cross-referencing tools for each other in identifying potential meme stocks.

5. The framework including change point detection and the ANN model with a threshold of 0.85 have the highest accuracy of 98.4%. However, the high accuracy comes at the cost of low recall ratio (very few meme stocks satisfy the criterion of change point detection). As a result, it would be rash to draw any unequivocal conclusion in this regard.

5.2. Strength and Weakness of Data Modeling Strategies

5.2.1. Robinhood's trading ban with RF, LR, and ANN

For the first criteria to define a meme stock, we argue that a stock will be defined as a 'meme' if it faces a Robinhood trading ban, so we use the Robinhood stock ban list as our ground truth of meme stocks during the event period, then bring it to classification with several models, including random forest, logistic regression, and artificial neural network. Though the ground truth can reveal the fact more accurately, the sample size of stocks with the trading ban on Robinhood is rather small since the trading ban was understandably unpopular and thus was soon lifted. To account for this small N problem, we develop our own set of criteria, the so-called four-pronged framework, to identify potential meme stocks.

5.2.2. A Four-Pronged Application on Detecting Meme Stocks

In our four-pronged framework, the first three criteria are simple and self-explanatory. Hence, here we focus only on the pros and cons of the two co-movement detection algorithms we use in validating the fourth criterion. As for change point detection (CPD), the CPD analysis highlights changes in the sentiment and popularity of stock in social media data. The `bcp` algorithm is also very efficient at O(n) complexity, making it very versatile for large data sets. However, CPD has no control regarding the source of the changes. If a publicly traded company has a favorable quarterly earnings report, then it is likely to be discussed on social media platforms due to this increased profitability. Depending on the magnitude of these changes in popularity, CPD methods could label such a stock as meme. Usually, we think of "meme stocks" as those that do not have sound financial fundamentals. Ultimately, it boils down to a problem of correlation vs. causation or a "chicken and the egg" situation. We do not know if people are talking about the stock because the stock's price increases (decreases), or vice-versa. Such a result is also observed in the Wavelet analysis. Additionally, change points are estimated regardless of the direction of the change. For example, the social media sentiment of a specific stock could drop significantly on a bad news day. This drop would trigger a CPD. However, the general idea is that meme stocks are "hyped", which should yield a higher sentiment score, therefore it might be against the general understanding of what a meme stock might be.

Regarding the Wavelet Coherence Analysis, the wavelet data mining technique aims to understand varied behavior across a time series, which helps to effectively capture nonlinear relationships such as stock variation in meme stock. Wavelet analysis graphs are impressively

informative and intuitive, providing users with a deeper understanding of price and sentiment co-movement. Nevertheless, we can only extract certain portions of the relationship from the cross-wavelet graph. The partial coherence between stock price and sentiment score also limits the effectiveness of using wavelet analysis on classification. Despite this limitation, the visualization of the cross-wavelet graph can still reveal valuable insights into the relationship between the stock's price and sentiment in different time periods, from which further strategies can be developed.

6. Conclusion and Avenues for Future Research

To discuss the results and avenues for meme stock modeling in the future, we must first understand the trend. Stocks have been a central component of economics for hundreds of years, but meme stocks are a relatively new development. As time goes on, we will better understand how meme-stocks are created and exactly what factors can lead to their huge breakouts. Further research on this topic can explore more detailed analysis across a longer period to better classify stocks as potential high risers.

Another possible avenue for future research is developing an online change point model. With this approach, "meme stocks" would ideally be detected before there is a significant increase in price, allowing the investor to profit from this information. For this to work, of course, the social media data would need to be streamed continuously. Developments to Twitter's open-source API are rumored to be in progress, and the future may see more free and shared information from social media sites. What becomes clear from this study is a complex push and pull relationship between stock price and social media sentiment, and an online change point detection model could be implemented to actively tell a user the next meme stock as seen from today. This tool would be a powerful way to identify good investments and may be used to give an investor the upper hand against competition.

In summary, there is a clear correlation between stock price and sentiment on the social media platforms of Twitter and Reddit. As time progresses and data becomes more accessible, it may be possible to implement the procedures demonstrated on live data and build models that can accurately predict meme stocks with decently high accuracy. Given the challenge and uncertainty in the financial market and the level of competition, this model would be a great tool for investors. In the end, a model that identifies a meme stock before its rapid growth phase using social media

sentiment analysis could change financial markets and be an effective demonstration of the power of machine learning in the 21st century.

APPENDIX

Collecting posts from subreddit 'r/wallstreetbets'

```
# Import libraries
import pandas as pd
from pmaw import PushshiftAPI
import datetime as dt
import re
import time
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from tqdm import tqdm

# instantiate
api = PushshiftAPI()

# Time in 2021
before = int(dt.datetime(2021,1,31,23,59).timestamp())
after = int(dt.datetime(2021,1,1,0,0).timestamp())

# wallstreetbets
subreddit = 'wallstreetbets'

posts_df = pd.DataFrame()
for i in tqdm(range(1, 13)):
    api = PushshiftAPI()
    if i in (1,3,5,7,8,10,12):
        day = 31
    elif i == 2:
        day = 28
    else:
        day = 30
    after = int(dt.datetime(2021,i,1,0,0).timestamp())
    before = int(dt.datetime(2021,i,day,23,59).timestamp())
```

```
    posts = api.search_submissions(subreddit = subreddit, limit=20000, before
= before, after = after)
    df = pd.DataFrame(posts)
    posts_df = pd.concat([posts_df, df])

# Example of few data
posts_df = posts_df.reset_index(drop = True)
posts_df.head(5)

# Export '2021_240000_posts.csv'
posts_df.to_csv('./Datasets/2021_240000_posts.csv', index = False)

posts_df = pd.read_csv('./Datasets/2021_240000_posts.csv')
# set empty selftext as ''
posts_df.loc[posts_df.selftext.isna(), 'selftext'] = ''
posts_df.loc[posts_df['selftext'] == '[deleted]', 'selftext'] = ''
posts_df.loc[posts_df['selftext'] == '[removed]', 'selftext'] = ''

# Select features we need
posts_df   =   posts_df[['title','num_comments',   'score',   'upvote_ratio',
'selftext','created_utc']]
posts_df.created_utc       =       posts_df.created_utc.apply(lambda       x:
dt.datetime.fromtimestamp(x)) # Convert time created

# New column 'content' with combined title and selftext(posts)
posts_df['content'] = posts_df['title'] + " " + posts_df['selftext']
posts_df = posts_df.drop(['title', 'selftext'], axis=1)
# remove non-letter or space
regex = re.compile('[^a-zA-Z ]')
posts_df['content'] = posts_df.content.apply(lambda x: regex.sub('', str(x)))
# Filter out
posts_df = posts_df[posts_df.content != ' ']
posts_df = posts_df.reset_index(drop = True)
posts_df = posts_df.sort_values('created_utc')
print(posts_df.shape)
posts_df.head(5)

# Export the modified data
```

```
posts_df.to_csv('./Datasets/2021_239020_posts_modified.csv', index = False)

# Get stock symbols existing in Nasdaq stock list & add new column with stock
ticker
posts_df = pd.read_csv('./Datasets/2021_239020_posts_modified.csv')
# Stock ticker list
stock = pd.read_csv("./Datasets/Nasdaq_stock.csv")
stock_symbols = list(stock.Symbol)
# Count the number of stock tickers
stop_words = set(stopwords.words('english'))
all_tickers = {}
ticker_col = []
for i, sentence in enumerate(posts_df['content']):
    word_tokens = word_tokenize(sentence)
    # tickers of each post
    ticker = []
    for w in word_tokens:
        if (w.lower() not in stop_words) and (w in stock_symbols): # word is
not a common words and is a stock symbol
            if w not in all_tickers:
                all_tickers[w] = 1
            else:
                all_tickers[w] += 1

            if w not in ticker:
                ticker.append(w)

    if ticker == []:
        ticker_col.append('NA')
    else:
        ticker_col.append(ticker)
# Sort all_tickers based on the number of stock symbol mentioned in the posts
sorted_all_tickers = sorted(all_tickers.items(), key=lambda kv: kv[1], reverse
= True)
symbols_list = pd.DataFrame(sorted_all_tickers, columns = ['Symbols',
'Counts'])
print(symbols_list.shape)
symbols_list.head(5)
```

```
# Export symbols_list
symbols_list.to_csv('./Datasets/2021_Symbols_list.csv', index = False)

# Export 'posts_with_ticker.csv'
posts_df['ticker'] = ticker_col  # add new column 'ticker'
posts_df = posts_df[posts_df.ticker != 'NA'] # remove data with no tickers
posts_df = posts_df.reset_index(drop=True)

# extra dataframe is to store the added data with the second, third, ...,
tickers
extra = pd.DataFrame()
for i in range(len(posts_df)):
    if len(posts_df.ticker[i]) > 1:
        for j in range(1, len(posts_df.ticker[i])):
            x = posts_df.iloc[i].copy()
            x.ticker = posts_df.ticker[i][j]
            extra = extra.append(x)

    # modify the first mentioned ticker in original posts_df after each
iteration
    posts_df.ticker[i] = posts_df.ticker[i][0]

# Combine original and extra dataframe
posts_df_ticker                    =                    pd.concat([posts_df,
extra]).sort_values('created_utc').reset_index(drop=True)
print(posts_df_ticker.shape)
posts_df_ticker.head(5)
# Export 'posts_with_ticker.csv'
posts_df_ticker.to_csv('./Datasets/posts_with_ticker.csv', index = False)
```

Collecting tweets on meme stocks using Twitter API

```
# For sending GET requests from the API
import requests
# For saving access tokens and for file management when creating and adding
to the dataset
import os
# For dealing with json responses we receive from the API
import json
# For displaying the data after
import pandas as pd
```

```
# For saving the response data in CSV format
import csv
# For parsing the dates received from twitter in readable formats
import datetime
import dateutil.parser
import unicodedata
#To add wait time between requests
import time

os.environ['TOKEN'] = 'insert the provided TOKEN'
def auth():
    return os.getenv('TOKEN')
def create_headers(bearer_token):
    headers = {"Authorization": "Bearer {}".format(bearer_token)}
    return headers
def create_url(keyword, start_date, end_date, max_results):

    search_url = "https://api.twitter.com/2/tweets/search/all" #Change to the
endpoint you want to collect data from

    #change params based on the endpoint you are using
    query_params = {'query': keyword,
                    'start_time': start_date,
                    'end_time': end_date,
                    'max_results': max_results,
                    'tweet.fields':
'id,text,author_id,created_at,lang,public_metrics,entities',
                    'next_token': {}}
    return (search_url, query_params)
def connect_to_endpoint(url, headers, params, next_token):
    params['next_token'] = next_token   #params object received from
create_url function
    response = requests.request("GET", url, headers = headers, params =
params)
    print("Endpoint Response Code: " + str(response.status_code))
    if response.status_code != 200:
        raise Exception(response.status_code, response.text)
    return response.json()
def append_to_csv(json_response, fileName):

    #A counter variable
    counter = 0

    #Open OR create the target CSV file
    csvFile = open(fileName, "a", newline="", encoding='utf-8')
    csvWriter = csv.writer(csvFile)

    #Loop through each tweet
    for tweet in json_response['data']:

        # We will create a variable for each since some of the keys might not
exist for some tweets
        # So we will account for that

        # 1. Author ID
        author_id = tweet['author_id']
```

```
        # 2. Time created
        created_at = dateutil.parser.parse(tweet['created_at'])


        # 3. Tweet ID
        tweet_id = tweet['id']

        # 4. Language
        lang = tweet['lang']

        # 5. Tweet metrics
        retweet_count = tweet['public_metrics']['retweet_count']
        reply_count = tweet['public_metrics']['reply_count']
        like_count = tweet['public_metrics']['like_count']
        quote_count = tweet['public_metrics']['quote_count']

        # 6. Tweet text
        text = tweet['text'].replace('\n', '').replace('\r\n', '')

        # 7. Ticker symbols
        try:
            ticker = tweet['entities']['cashtags']
        except KeyError:
            ticker = 0

        # Assemble all data in a list
        res = [author_id, created_at, tweet_id, lang, like_count,
quote_count, reply_count, retweet_count, text, ticker]

        # Append the result to the CSV file
        csvWriter.writerow(res)
        counter += 1

    # When done, close the CSV file
    csvFile.close()

    # Print the number of tweets for this iteration
    print("# of Tweets added from this response: ", counter)
#Inputs for tweets
bearer_token = auth()
headers = create_headers(bearer_token)
keyword = "#MemeStocks OR #PennyStocks OR #MemeStock OR #PennyStock -
is:retweet lang:en"
start_list = ['2021-01-01T00:00:00.000Z','2021-02-01T00:00:00.000Z','2021-03-
01T00:00:00.000Z','2021-04-01T00:00:00.000Z','2021-05-
01T00:00:00.000Z','2021-06-01T00:00:00.000Z','2021-07-
01T00:00:00.000Z','2021-08-01T00:00:00.000Z','2021-09-
01T00:00:00.000Z','2021-10-01T00:00:00.000Z','2021-11-
01T00:00:00.000Z','2021-12-01T00:00:00.000Z']
end_list = ['2021-01-31T23:59:59Z','2021-02-28T23:59:59Z','2021-03-
31T23:59:59Z','2021-04-30T23:59:59Z','2021-05-31T23:59:59Z','2021-06-
30T23:59:59Z','2021-07-31T23:59:59Z','2021-08-31T23:59:59Z','2021-09-
30T23:59:59Z','2021-10-31T23:59:59Z','2021-11-30T23:59:59Z','2021-12-
31T23:59:59Z']

max_results = 500
```

```
#Total number of tweets we collected from the loop
total_tweets = 0


for i in range(0,len(start_list)):

    # Inputs
    count = 0 # Counting tweets per time period
    max_count = 1000000 # Max tweets per time period
    flag = True
    next_token = None

    # Check if flag is true
    while flag:
        # Check if max_count reached
        if count >= max_count:
            break
        print("-------------------")
        print("Token: ", next_token)
        url = create_url(keyword, start_list[i],end_list[i], max_results)
        json_response = connect_to_endpoint(url[0], headers, url[1],
next_token)
        result_count = json_response['meta']['result_count']

        if 'next_token' in json_response['meta']:
            # Save the token to use for next call
            next_token = json_response['meta']['next_token']
            print("Next Token: ", next_token)
            if result_count is not None and result_count > 0 and next_token
is not None:
                print("Start Date: ", start_list[i])
                append_to_csv(json_response, "data.csv")
                count += result_count
                total_tweets += result_count
                print("Total # of Tweets added: ", total_tweets)
                print("-------------------")
                time.sleep(5)
        # If no next token exists
        else:
            if result_count is not None and result_count > 0:
                print("-------------------")
                print("Start Date: ", start_list[i])
                append_to_csv(json_response, "data.csv")
                count += result_count
                total_tweets += result_count
                print("Total # of Tweets added: ", total_tweets)
                print("-------------------")
                time.sleep(5)

            #Since this is the final request, turn flag to false to move to
the next time period.
            flag = False
            next_token = None
        time.sleep(5)
print("Total number of results: ", total_tweets)
```

Extracting tickers from social media posts

```
tweets = pd.read_csv('tweets.csv')
import ast
tweets['ticker'] = tweets['ticker'].apply(lambda x: ast.literal_eval(x))
from stocksymbol import StockSymbol

api_key = '6697474f-fe39-4a06-aaed-73fb397fe7fa'
ss = StockSymbol(api_key)
symbol_only_list = ss.get_symbol_list(market="US", symbols_only=True)
import re

def get_ticker(row):
    if row['ticker']==0:
        list1 = re.findall(r'(?<=\$)\w+|\b[A-Z]{3,6}\b', row['text'])
        return [x for x in list1 if x in symbol_only_list]
twitter['ticker_appended'] = twitter.apply(get_ticker, axis = 1)
def final_ticker(row):
    if row['ticker'] == 0:
        return row['ticker_appended']
    else:
        return [x.get('tag') for x in row['ticker']]

tweets['final_ticker'] = tweets.apply(final_ticker, axis = 1)
tweets = tweets.drop(['ticker', 'ticker_appended'], axis = 1)
tweets = tweets.rename(columns = {'final_ticker': 'ticker'})
tweets_filtered = tweets[tweets['ticker'].apply(lambda x: len(x) !=
0)].reset_index(drop = True)
tweets_filtered.to_csv('tweets_filtered.csv')
with open('stock_tweet.txt', mode='wt', encoding='utf-8') as myfile:

myfile.write('\n'.join(tweets_filtered["ticker"].explode().unique().tolist())
)
```

Collecting stock prices using CRSP API

```
import wrds
import pandas as pd
import csv
conn = wrds.Connection()


library = conn.list_libraries()

csvFile = open("handle.csv", "a", newline="", encoding='utf-8')
csvWriter = csv.writer(csvFile)

for j in library:
    tables = conn.list_tables(library=j)
    for i in tables:
        try:
            container = list(conn.get_table(library=j, table=i, obs=1))
            if 'tic' in container or 'ticker' in container or 'tick' in
container:
                res = [j, i, 'selected']
                csvWriter.writerow(res)
            else:
```

```
                res = [j, i, 'not selected']
                csvWriter.writerow(res)
        except:
            res = [j, i, 'no access']
            csvWriter.writerow(res)

csvFile.close()


symbols_reddit = pd.read_csv("2021_Symbols_list.csv")
import numpy as np
def match_permno(a):
    start = """select permno from wrdsapps_link_crsp_ibes.ibcrsphist where
ticker = """
    ticker = "'"+a+"'"

    x = conn.raw_sql(start+ticker).values
    try:
        x1 = np.unique(np.concatenate(x).ravel())
        permno = x1[~np.isnan(x1)]
    except ValueError:
        permno = np.nan
    except TypeError:
        permno = np.nan
    return permno
symbols_reddit['permno'] = symbols_reddit['Symbols'].apply(match_permno)


def get_price(row):
    start = """select permno, date, bidlo, askhi, prc, vol, ret, bid, ask,
openprc, retx from crsp.dsf where permno="""
    end = """ and date >='01/01/2021' and date <='12/31/2021'"""
    if row['num_matches'] == 1:
        permno = str(int(row['permno'][0]))
        price = conn.raw_sql(start+permno+end, date_cols = ['date'])
        if len(price) != 0:
            price.to_csv('Price_{}.csv'.format(row['Symbols']))
symbols_reddit.apply(get_price, axis = 1)
import glob
import os


df = pd.DataFrame()
for file in glob.glob(os.getcwd()+'\\Price_*.csv'):
    aux = pd.read_csv(file).drop('Unnamed: 0', axis = 1)
    aux['ticker'] = file[110:-4]
    df = df.append(aux)
df.to_csv('Price1.csv')
for file in glob.glob(os.getcwd()+'\\Price_*.csv'):
    os.remove(file)
```

Merging social media and price data frames

```
price_twitter = pd.read_csv('new_price_twitter.csv',
low_memory=False).drop(['NAMEENDT', 'DCLRDT', 'DLAMT', 'DLPDT', 'NEXTDT',
'PAYDT', 'RCRDDT', 'SHRENDDT', 'NWPERM', 'DLPRC', 'NUMTRD'], axis = 1)
price_twitter.head()
```

```
from datetime import datetime
price_twitter['date'] = price_twitter['date'].apply(lambda x:
datetime.strptime(x, "%m/%d/%Y").strftime("%m-%d-%Y"))
new_tweets_filtered['post_date'] =
new_tweets_filtered['created_at'].apply(lambda x: datetime.\

fromisoformat(x).strftime("%m-%d-%Y"))
new_tweets_filtered = new_tweets_filtered.explode('ticker').reset_index(drop
= True)


price_twitter['spread'] = price_twitter['ASKHI'] - price_twitter['BIDLO']
twitter_panel = pd.merge(a, price_twitter[['date', 'TICKER', 'PRC', 'VOL',
'spread']], left_on = ['ticker', 'post_date'], right_on = ['TICKER', 'date'],
how = 'inner').rename(columns = {'PRC': 'price', 'VOL':
'volume'}).drop(['date', 'TICKER'], axis = 1)
twitter_panel.head()


first_tweets = pd.read_csv('tweets_filtered_Zhelun.csv').drop(['Unnamed: 0',
'Unnamed: 0.1'], axis = 1)
first_tweets.head()


first_tweets['ticker'] = first_tweets['ticker'].apply(lambda x:
ast.literal_eval(x) if isinstance(x, float) == False else x)
first_tweets['post_date'] = first_tweets['created_at'].apply(lambda x:
datetime.\

fromisoformat(x).strftime("%m-%d-%Y"))
first_tweets = first_tweets.explode('ticker').reset_index(drop = True)
b = first_tweets.groupby(['ticker','post_date']).agg({'sentiment_scores':
'mean', 'like_count': 'sum', 'quote_count': 'sum', 'reply_count': 'sum',
'retweet_count': 'sum', 'tweet_id': 'count'}).reset_index().rename(columns =
{'sentiment_scores': 'avg_sentiment_score', 'like_count': 'total_likes',
'quote_count': 'total_quotes', 'reply_count': 'total_replies',
'retweet_count': 'total_reweets', 'tweet_id': 'post_count'})
first_price = pd.read_csv('Price_twitter.csv', low_memory = False)
first_price.head()


first_price['spread'] = first_price['ASKHI'] - first_price['BIDLO']
first_tweets.loc[~first_tweets['tweet_id'].isin(new_tweets_filtered['tweet_id
'])]
first_price['date'] = first_price['date'].apply(lambda x:
datetime.strptime(x, "%m/%d/%Y").strftime("%m-%d-%Y"))
first_panel = pd.merge(b, first_price[['date', 'TICKER', 'PRC', 'VOL',
'spread']], left_on = ['ticker', 'post_date'], right_on = ['TICKER', 'date'],
how = 'inner').rename(columns = {'PRC': 'price', 'VOL':
'volume'}).drop(['date', 'TICKER'], axis = 1)
first_panel.head()

c = pd.concat([first_panel, twitter_panel]).reset_index(drop=True)
c = c.drop_duplicates(subset = ['ticker', 'post_date'],keep =
'last').reset_index(drop = True)
c[c.duplicated(subset=['ticker', 'post_date'], keep=False)]
```

```
d = c.groupby(['ticker','post_date']).agg({'avg_sentiment_score': 'max',
'total_likes': 'sum', 'total_quotes': 'sum', 'total_replies': 'sum',
'total_reweets': 'sum', 'post_count': 'sum', 'price': 'max', 'volume': 'max',
'spread': 'max'}).reset_index()


d.to_csv('twitter_panel.csv')
c.to_csv('twitter_merged.csv')
```

Data cleaning

```
import numpy as np

import pandas as pd

df = pd.read_csv("tweets_additional_keywords.csv")

df_filtered =
df[df["text"].str.contains("https://t.co/knzARFId2i|#KobeBryant|@MONEY
GANGJAMES1|https://t.co/|@AngelsTrading|#VotaYoloAventuras|@NandoAvent
urero|Å")==False]

df_filtered.drop_duplicates(subset='text', keep='first', inplace=True)

df_filtered.to_csv("tweets_addition_filtered.csv")
```

Sentiment scoring

```
import nltk
import pandas as pd
import numpy as np

# Download Packages
nltk.download([
    "names",
    "stopwords",
    "vader_lexicon",
    "punkt"
])

# Set Stops Words as English
stopwords = nltk.corpus.stopwords.words("english")

# Create Sentiment Score Column for Reddit Dataset
df = pd.read_csv("reddit.csv")
df['sentiment_scores'] = 0
df.head()

# Import Sentiment Analyzer
from nltk.sentiment import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()

# Load Twitter Dataset
df_1 = pd.read_csv('tweets_addition_filtered.csv', index_col=0)
df_1['sentiment_scores'] = 0
df_1.drop(['Unnamed: 0.1', 'lang'], axis = 1, inplace = True)
```

```
df_1.head()

# Convert Data Type to String
df_1.index = range(len(df_1))
df_1['text'] = df_1['text'].astype(str)

# Split Dataset to 10 Smaller Datasets
df_split = np.array_split(df_1, 10)
df_split[0].index = range(len(df_split[0]))
df_split[1].index = range(len(df_split[1]))
df_split[2].index = range(len(df_split[2]))
df_split[3].index = range(len(df_split[3]))
df_split[4].index = range(len(df_split[4]))
df_split[5].index = range(len(df_split[5]))
df_split[6].index = range(len(df_split[6]))
df_split[7].index = range(len(df_split[7]))
df_split[8].index = range(len(df_split[8]))
df_split[9].index = range(len(df_split[9]))

# Scoring Sentiment for 10 Smaller Datasets
for i in range(len(df_split[0].index)):
        df_split[0].iloc[i, 9] =
sia.polarity_scores(df_split[0]['text'][i])['pos']

for i in range(len(df_split[1].index)):
        df_split[1].iloc[i, 9] =
sia.polarity_scores(df_split[1]['text'][i])['pos']

for i in range(len(df_split[2].index)):
        df_split[2].iloc[i, 9] =
sia.polarity_scores(df_split[2]['text'][i])['pos']

for i in range(len(df_split[3].index)):
        df_split[3].iloc[i, 9] =
sia.polarity_scores(df_split[3]['text'][i])['pos']

for i in range(len(df_split[4].index)):
        df_split[4].iloc[i, 9] =
sia.polarity_scores(df_split[4]['text'][i])['pos']

for i in range(len(df_split[5].index)):
        df_split[5].iloc[i, 9] =
sia.polarity_scores(df_split[5]['text'][i])['pos']

for i in range(len(df_split[6].index)):
        df_split[6].iloc[i, 9] =
sia.polarity_scores(df_split[6]['text'][i])['pos']

for i in range(len(df_split[7].index)):
        df_split[7].iloc[i, 9] =
sia.polarity_scores(df_split[7]['text'][i])['pos']

for i in range(len(df_split[8].index)):
        df_split[8].iloc[i, 9] =
sia.polarity_scores(df_split[8]['text'][i])['pos']

for i in range(len(df_split[9].index)):
```

```
        df_split[9].iloc[i, 9] =
sia.polarity_scores(df_split[9]['text'][i])['pos']

# Export All Files

df_split[0].to_csv('1.csv')
df_split[1].to_csv('2.csv')
df_split[2].to_csv('3.csv')
df_split[3].to_csv('4.csv')
df_split[4].to_csv('5.csv')
df_split[5].to_csv('6.csv')
df_split[6].to_csv('7.csv')
df_split[7].to_csv('8.csv')
df_split[8].to_csv('9.csv')
df_split[9].to_csv('10.csv')
```

Data preprocessing for RF/LR/ANN

```
import pandas as pd

import numpy as np

import random

import time

import scipy.stats as st

import datetime

import math

from scipy.stats import variation twitter_panel_


df = pd.read_csv('twitter_panel.csv')

reddit_panel_df = pd.read_csv('reddit_panel.csv')

print(len(twitter_panel_df),len(reddit_panel_df))

start_date = pd.to_datetime('1/26/2021', format='%m/%d/%Y')

total_panel_df =
twitter_panel_df.merge(reddit_panel_df,how='outer',on=['ticker','post_date','price','volume','spread','meme'])

total_panel_df['date'] = pd.to_datetime(total_panel_df['post_date'], format='%m/%d/%Y')

total_panel_df = total_panel_df.sort_values(['ticker','date'])

total_panel_df.to_csv('6total_panel.csv', sep=',', header=True, index=True)
```

```
values = {'avg_sentiment_score_x':0.5,'total_likes':0,'total_quotes':0,'total_replies':0,'total_reweets':0,'post_count_x':0,

'avg_sentiment_score_y':0.5,'avg_score':0,'comments_count':0,'avg_upvote_ratio':0,'post_count_y':0 ,
        'price':0,'volume':0,'spread':0}
total_panel_df = total_panel_df.fillna(value=values)
print(len(total_panel_df))
total_panel_df.drop_duplicates(subset=['ticker','date'],keep=False,inplace=True )
print(len(total_panel_df))
total_panel_df.reset_index(drop=True,inplace=True )

df_1_head=['1_index','1_ticker','1_date','1_mentionday_count','1_avg_sentiment_score_x','1_total_likes','1_total_quotes','1_total_replies','1_total_reweets','1_post_count_x','1_avg_sentiment_score_y','1_avg_score','1_comments_count','1_avg_upvote_ratio','1_post_count_y',
'1_price_avg','1_volume_avg','1_spread_avg','1_price_std','1_volume_std','1_spread_std']
df_1 = pd.DataFrame(columns=df_1_head)

df_7_head=
['7_index','7_ticker','7_date','7_mentionday_count','7_avg_sentiment_score_x','7_total_likes','7_total_quotes','7_total_replies','7_total_reweets','7_post_count_x','7_avg_sentiment_score_y','7_avg_score','7_comments_count','7_avg_upvote_ratio','7_post_count_y','7_price_avg','7_volume_avg','7_spread_avg','7_price_std','7_volume_std','7_spread_std']
df_7 = pd.DataFrame(columns=df_7_head)

df_14_head=['14_index','14_ticker','14_date','14_mentionday_count','14_avg_sentiment_score_x','14_total_likes','14_total_quotes','14_total_replies','14_total_reweets','14_post_count_x','14_avg_sentiment_score_y','14_avg_score','14_comments_count','14_avg_upvote_ratio','14_post_count_y','14_price_avg','14_volume_avg','14_spread_avg','14_price_std','14_volume_std','14_spread_std']
df_14 = pd.DataFrame(columns=df_14_head)
```

```
df_21_head=['21_index','21_ticker','21_date','21_mentionday_count','21_avg_sentiment_score_x','21_total
_likes','21_total_quotes','21_total_replies','21_total_reweets','21_post_count_x','21_avg_sentiment_score_y'
,'21_avg_score','21_comments_count','21_avg_upvote_ratio','21_post_count_y','21_price_avg','21_volume
_avg','21_spread_avg','21_price_std','21_volume_std','21_spread_std']

df_21 = pd.DataFrame(columns=df_21_head)


df_28_head=['28_index','28_ticker','28_date','28_mentionday_count','28_avg_sentiment_score_x','28_total
_likes','28_total_quotes','28_total_replies','28_total_reweets','28_post_count_x','28_avg_sentiment_score_y'
,'28_avg_score','28_comments_count','28_avg_upvote_ratio','28_post_count_y','28_price_avg','28_volume
_avg','28_spread_avg','28_price_std','28_volume_std','28_spread_std']

df_28 = pd.DataFrame(columns=df_28_head)


gaps=[1,7,14,21,28]

total_panel_df.loc[:, 'Oindex'] = 0

for gap in gaps:

    for i in total_panel_df.index.values:

        total_panel_df.loc[i, 'Oindex'] = i

        if total_panel_df.loc[i,'date'] > start_date:

#             try:

            a_sentiment_score_x = 0

            a_total_likes = 0

            a_total_quotes = 0

            a_total_replies = 0

            a_total_reweets = 0

            a_post_count_x = 0

            a_sentiment_score_y = 0

            a_avg_score = 0

            a_comments_count = 0

            a_avg_upvote_ratio = 0

            a_post_count_y = 0
```

```
        a_metion_day_count = 0
        price_series = []
        volume_series = []
        spread_series = []
        price_series.append(total_panel_df.loc[i,'price'])
        volume_series.append(total_panel_df.loc[i,'volume'])
        spread_series.append(total_panel_df.loc[i,'spread'])

        date_i=total_panel_df.loc[i,'date']
        ticker_i=total_panel_df.loc[i,'ticker']
        i_shiftgap = max(0,i-gap)
        for j in range(i_shiftgap,i):
            date_j=total_panel_df.loc[j,'date']
            ticker_j=total_panel_df.loc[j,'ticker']
            diff = (date_i-date_j)/pd.Timedelta(1, 'D')
            if diff>0 and diff<=gap and ticker_i==ticker_j:
                a_sentiment_score_x += total_panel_df.loc[j,'avg_sentiment_score_x'] *
total_panel_df.loc[j,'post_count_x']
                a_total_likes += total_panel_df.loc[j,'total_likes']
                a_total_quotes += total_panel_df.loc[j,'total_quotes']
                a_total_replies += total_panel_df.loc[j,'total_replies']
                a_total_reweets += total_panel_df.loc[j,'total_reweets']
                a_post_count_x += total_panel_df.loc[j,'post_count_x']
                a_sentiment_score_y += total_panel_df.loc[j,'avg_sentiment_score_y'] *
total_panel_df.loc[j,'post_count_y']
                a_avg_score += total_panel_df.loc[j,'avg_score'] * total_panel_df.loc[j,'post_count_y']
                a_comments_count += total_panel_df.loc[j,'comments_count']
                a_avg_upvote_ratio += total_panel_df.loc[j,'avg_upvote_ratio'] *
total_panel_df.loc[j,'post_count_y']
```

```
            a_post_count_y += total_panel_df.loc[j,'post_count_y']
            a_metion_day_count += 1
            price_series.append(total_panel_df.loc[j,'price'])
            volume_series.append(total_panel_df.loc[j,'volume'])
            spread_series.append(total_panel_df.loc[j,'spread'])

        #print(price_series.type)
        a_price_avg = np.mean(price_series)
        a_volume_avg = np.mean(volume_series)
        a_spread_avg = np.mean(spread_series)
        a_price_std = np.std(price_series)
        a_volume_std = np.std(volume_series)
        a_spread_std = np.std(spread_series)

        if a_post_count_x != 0:
            a_sentiment_score_x = a_sentiment_score_x/a_post_count_x
        if a_post_count_y != 0:
            a_sentiment_score_y = a_sentiment_score_y/a_post_count_y
            a_avg_score = a_avg_score/a_post_count_y
            a_avg_upvote_ratio = a_avg_upvote_ratio/a_post_count_y

        print(i,gap,date_i,ticker_i)
        if gap==1:
            df_1.loc[len(df_1)] = [i,ticker_i,date_i,a_metion_day_count,
a_sentiment_score_x,a_total_likes,a_total_quotes,a_total_replies,a_total_reweets,a_post_count_x,
a_sentiment_score_y,a_avg_score,a_comments_count,a_avg_upvote_ratio,a_post_count_y, a_price_avg,
a_volume_avg,a_spread_avg,a_price_std,a_volume_std,a_spread_std ]
        if gap==7:
```

```
            df_7.loc[len(df_7)] = [i,ticker_i,date_i,a_metion_day_count,
a_sentiment_score_x,a_total_likes,a_total_quotes,a_total_replies,a_total_reweets,a_post_count_x,
a_sentiment_score_y,a_avg_score,a_comments_count,a_avg_upvote_ratio,a_post_count_y,  a_price_avg,
a_volume_avg,a_spread_avg,a_price_std,a_volume_std,a_spread_std ]
        if gap==14:
            df_14.loc[len(df_14)] = [i,ticker_i,date_i,a_metion_day_count,
a_sentiment_score_x,a_total_likes,a_total_quotes,a_total_replies,a_total_reweets,a_post_count_x,
a_sentiment_score_y,a_avg_score,a_comments_count,a_avg_upvote_ratio,a_post_count_y,  a_price_avg,
a_volume_avg,a_spread_avg,a_price_std,a_volume_std,a_spread_std ]
        if gap==21:
            df_21.loc[len(df_21)] = [i,ticker_i,date_i,a_metion_day_count,
a_sentiment_score_x,a_total_likes,a_total_quotes,a_total_replies,a_total_reweets,a_post_count_x,
a_sentiment_score_y,a_avg_score,a_comments_count,a_avg_upvote_ratio,a_post_count_y,  a_price_avg,
a_volume_avg,a_spread_avg,a_price_std,a_volume_std,a_spread_std ]
        if gap==28:
            df_28.loc[len(df_28)] = [i,ticker_i,date_i,a_metion_day_count,
a_sentiment_score_x,a_total_likes,a_total_quotes,a_total_replies,a_total_reweets,a_post_count_x,
a_sentiment_score_y,a_avg_score,a_comments_count,a_avg_upvote_ratio,a_post_count_y,  a_price_avg,
a_volume_avg,a_spread_avg,a_price_std,a_volume_std,a_spread_std ]


df_time = df_7.merge(df_14,how='left',left_on=['7_index'],right_on=['14_index'])
df_time = df_time.merge(df_21,how='left',left_on=['7_index'],right_on=['21_index'])
df_time = df_time.merge(df_28,how='left',left_on=['7_index'],right_on=['28_index'])
df_time = df_time.merge(df_1,how='left',left_on=['7_index'],right_on=['1_index'])
df_time.to_csv('try2df_time.csv', sep=',', header=True, index=True)
total_panel_df.to_csv('try2.csv', sep=',', header=True, index=True)


df_time = df_7.merge(df_14,how='left',left_on=['7_index'],right_on=['14_index'])
df_time = df_time.merge(df_21,how='left',left_on=['7_index'],right_on=['21_index'])
df_time = df_time.merge(df_28,how='left',left_on=['7_index'],right_on=['28_index'])
df_time = df_time.merge(df_1,how='left',left_on=['7_index'],right_on=['1_index'])
df_time.to_csv('try2df_time.csv', sep=',', header=True, index=True)
```

```
total_panel_df.to_csv('try2.csv', sep=',', header=True, index=True)


from sklearn.metrics import confusion_matrix

from sklearn import metrics

from sklearn.metrics import r2_score

from sklearn.preprocessing import StandardScaler

from sklearn import preprocessing

from sklearn.model_selection import train_test_split


df = pd.read_csv('df_timeseries.csv')

df = df[~df['price'].isin([0])]


df['post_date'] = pd.to_datetime(df['post_date'], format='%m/%d/%Y')

df=df.drop(['Unnamed:
0','1_index','7_index','14_index','21_index','28_index','date','1_date','7_date','14_date','21_date','28_date','1
_ticker','7_ticker','14_ticker','21_ticker','28_ticker'], axis=1)

start_date = pd.to_datetime('1/27/2021', format='%m/%d/%Y')

end_date = pd.to_datetime('2/4/2021', format='%m/%d/%Y')

df = df[(df['post_date']>=start_date)]

df_labeled = df[(df['post_date']>=start_date)&(df['post_date']<=end_date)]


X_allyear = df.drop(['meme','ticker','post_date','Oindex'], axis=1)

X_allyear_Oindex = df['Oindex']

X_allyear_ticker = df['ticker']

X_allyear_date = df['post_date']

X_allyear_scaled = pd.DataFrame(preprocessing.scale(X_allyear))

X_allyear_scaled.columns = X_allyear.columns
```

```
X = df_labeled.drop(['meme','ticker','post_date','Oindex'], axis=1)
X_Oindex = df_labeled['Oindex']
X_ticker = df_labeled['ticker']
X_date = df_labeled['post_date']
Y = df_labeled['meme']
X_scaled = pd.DataFrame(preprocessing.scale(X))
X_scaled.columns = X.columns
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled,Y,test_size=0.2, random_state = 2)


n_train = X_train.shape[0]
print("n_train:",n_train)
n_test= X_test.shape[0]
print("n_test:",n_test)
n_allyear= X_allyear_scaled.shape[0]
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
def print_grid_search_metrics(gs):
    print ("Best score: %0.3f" % gs.best_score_)
    print ("Best parameters set:")
    best_parameters = gs.best_params_
    for param_name in sorted(parameters.keys()):
        print("\t%s: %r" % (param_name, best_parameters[param_name]))

parameters = {
    'n_estimators' : [100,200,300],
    'min_samples_leaf':[1,5,10],
    'random_state':[3],
```

```
    'criterion':['entropy']
}
Grid_RF = GridSearchCV(RandomForestClassifier(),parameters, cv=5, scoring='f1_macro' )
Grid_RF.fit(X_train, Y_train)
print_grid_search_metrics(Grid_RF)

tuned_rf_model = Grid_RF.best_estimator_

Y_train_pred_tunedrf = tuned_rf_model.predict(X_train)
Y_test_pred_tunedrf = tuned_rf_model.predict(X_test)

print('Tuned Random Forest on Test Set:')
print(pd.crosstab(Y_test, Y_test_pred_tunedrf, rownames=['Actual'], colnames=['Predicted']))
print(metrics.classification_report(Y_test,Y_test_pred_tunedrf))
print("Accuracy: " + str(metrics.accuracy_score(Y_test,Y_test_pred_tunedrf)))
print("Cohen's Kappa: " + str(metrics.cohen_kappa_score(Y_test,Y_test_pred_tunedrf)))
print("R^2",r2_score(Y_test, Y_test_pred_tunedrf))
classification_report_test = metrics.classification_report(Y_test,Y_test_pred_tunedrf,output_dict=True)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegressionCV
lr_model = LogisticRegressionCV(penalty='l2',solver='liblinear',max_iter=400,cv=5)
lr_model.fit(X_train, Y_train)

Y_train_pred_lr = lr_model.predict(X_train)
Y_test_pred_lr = lr_model.predict(X_test)
Y_train_pred_lr_prob = lr_model.predict_proba(X_train)
Y_test_pred_lr_prob = lr_model.predict_proba(X_test)
```

```
threshold_list = [0.05,0.1,0.15,0.2,
                  0.25,0.3,0.35,0.4,
                  0.45,0.5,0.55,0.6,
                  0.65,0.7,0.75,0.8,
                  0.85,0.9,0.95]

threshold_macrof1_dict_train = {}
threshold_macrof1_dict_test = {}

for i in threshold_list:
    Y_train_pred_adjlr = []
    Y_test_pred_adjlr = []
    for j in range(n_train):
        if Y_train_pred_lr_prob[j,1] > i:
            Y_train_pred_adjlr.append(1)
        else:
            Y_train_pred_adjlr.append(0)
    for j in range(n_test):
        if Y_test_pred_lr_prob[j,1] > i:
            Y_test_pred_adjlr.append(1)
        else:
            Y_test_pred_adjlr.append(0)

    Y_train_pred_adjlr = np.array(Y_train_pred_adjlr)
    classification_report_train =
metrics.classification_report(Y_train,Y_train_pred_adjlr,output_dict=True)
    threshold_macrof1_dict_train [i] = classification_report_train['macro avg']['f1-score']

    Y_test_pred_adjlr = np.array(Y_test_pred_adjlr)
```

```
    classification_report_test = metrics.classification_report(Y_test,Y_test_pred_adjlr,output_dict=True)
    threshold_macrof1_dict_test [i] = classification_report_test['macro avg']['f1-score']

index_tr_lr = list(threshold_macrof1_dict_train.keys())[list(threshold_macrof1_dict_train.values()).index(max(list(threshold_macrof1_dict_train.values())))]
index_te_lr = list(threshold_macrof1_dict_test.keys())[list(threshold_macrof1_dict_test.values()).index(max(list(threshold_macrof1_dict_test.values())))]
```

Artificial Neural Network

```
import tensorflow as tf

#Initialising ANN
ann = tf.keras.models.Sequential()
#Adding First Hidden Layer
ann.add(tf.keras.layers.Dense(units=6,activation="relu"))
#Adding Second Hidden Layer
ann.add(tf.keras.layers.Dense(units=6,activation="relu"))
#Adding Output Layer
ann.add(tf.keras.layers.Dense(units=1,activation="sigmoid"))
#Compiling ANN
ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
#Fitting ANN
ann.fit(X_train.values,Y_train.values,batch_size=32,epochs = 100)

Y_train_pred_ann_values = ann.predict(X_train.values)
Y_test_pred_ann_values = ann.predict(X_test.values)
```

```
threshold_list = [0.05,0.1,0.15,0.2,
                  0.25,0.3,0.35,0.4,
                  0.45,0.5,0.55,0.6,
                  0.65,0.7,0.75,0.8,
                  0.85,0.9,0.95]

threshold_macrof1_dict_train = {}
threshold_macrof1_dict_test = {}

for i in threshold_list:
    Y_train_pred_ann = []
    Y_test_pred_ann = []

    for j in range(n_train):
        if Y_train_pred_ann_values[j][0] > i:
            Y_train_pred_ann.append(1)
        else:
            Y_train_pred_ann.append(0)

    for j in range(n_test):
        if Y_test_pred_ann_values[j][0] > i:
            Y_test_pred_ann.append(1)
        else:
            Y_test_pred_ann.append(0)



    Y_train_pred_ann = np.array(Y_train_pred_ann)
    classification_report_train = metrics.classification_report(Y_train,Y_train_pred_ann,output_dict=True)
```

```
    threshold_macrof1_dict_train [i] = classification_report_train['macro avg']['f1-score']


    Y_test_pred_ann = np.a rray(Y_test_pred_ann)
    classification_report_test = metrics.classification_report(Y_test,Y_test_pred_ann,output_dict=True)
    threshold_macrof1_dict_test [i] = classification_report_test['macro avg']['f1-score']


index_tr_ann = list(threshold_macrof1_dict_train.keys())[list(threshold_macrof1_dict_train.values()).index(max(list(threshold_macrof1_dict_train.values())))]
index_te_ann = list(threshold_macrof1_dict_test.keys())[list(threshold_macrof1_dict_test.values()).index(max(list(threshold_macrof1_dict_test.values())))]
```

Predicting meme stocks: not S&P 500, high popularity, and big spread

```
import pandas as pd
import numpy as np
twitter = pd.read_csv("twitter_panel.csv")
sp500 = pd.read_excel("SPX as of Dec 31 20211.xlsx")
reddit = pd.read_csv("reddit_panel.csv")
twitter.head()
len(set(twitter['ticker'])), len(set(reddit['ticker']))


# # First, a meme stock can't be in the S&P 500 list
twitter['sp500'] = np.where(twitter["ticker"].isin(list(sp500['Ticker'])), 1, 0)
reddit['sp500'] = np.where(reddit["ticker"].isin(list(sp500['Ticker'])), 1, 0)


# # Second, the stock's popularity must be at least at the median level
# # Third, the stock's spread must be identified as outlier
twitter['date'] = pd.to_datetime(twitter['post_date'], format='%m/%d/%Y')
reddit['date'] = pd.to_datetime(reddit['post_date'], format='%m/%d/%Y')
```

```
from datetime import datetime, timedelta
from dateutil.relativedelta import relativedelta

def twitter_reference(i,gap):
    if gap != 30:
        time_lowbound = twitter.loc[i,'date']- pd.Timedelta(gap, 'D')
    else:
        time_lowbound = twitter.loc[i,'date'] + relativedelta(months=-1)

    twitter_i = twitter[(twitter['date']>= time_lowbound)&(twitter['date']<= twitter.loc[i,'date'])]
    grouped_twitter_i_post_count = twitter_i.groupby(['ticker'], as_index=False)['post_count'].median()
    post_count_series = grouped_twitter_i_post_count['post_count']
    spread_series = twitter_i['spread']
    post_count_pc90 = post_count_series.quantile(0.90)
    spread_mean = twitter_i['spread'].mean()
    spread_std = twitter_i['spread'].std()
    return (post_count_pc90,spread_mean,spread_std)

def reddit_reference(i,gap):
    if gap != 30:
        time_lowbound = reddit.loc[i,'date']- pd.Timedelta(gap, 'D')
    else:
        time_lowbound = reddit.loc[i,'date'] + relativedelta(months=-1)

    reddit_i = reddit[(reddit['date']>= time_lowbound)&(reddit['date']<= reddit.loc[i,'date'])]
    grouped_reddit_i_post_count = reddit_i.groupby(['ticker'], as_index=False)['post_count'].median()
    post_count_series = grouped_reddit_i_post_count['post_count']
```

```
    spread_series = reddit_i['spread']
    post_count_pc90 = post_count_series.quantile(0.90)
    spread_mean = reddit_i['spread'].mean()
    spread_std = reddit_i['spread'].std()
    return (post_count_pc90,spread_mean,spread_std)

gaps = [1,7,14,30]
for gap in gaps:
    column_name_popularity = "popularity_" + str(gap)
    column_name_spreadoutlier = "spreadoutlier_" + str(gap)

    twitter[column_name_popularity] = 0
    twitter[column_name_spreadoutlier] = 0
    for i in twitter.index.values:
        twitter_ref = twitter_reference(i,gap)
        if twitter.loc[i,'post_count'] >= twitter_ref[0]:
            twitter.loc[i,column_name_popularity] = 1
        else:
            twitter.loc[i,column_name_popularity] = 0
        if twitter.loc[i,'spread']  > (twitter_ref[1] + 3* twitter_ref[2]):
            twitter.loc[i,column_name_spreadoutlier] = 1
        else:
            twitter.loc[i,column_name_spreadoutlier] = 0

    reddit[column_name_popularity] = 0
    reddit[column_name_spreadoutlier] = 0
    for i in reddit.index.values:
        reddit_ref = reddit_reference(i,gap)
```

```
    if reddit.loc[i,'post_count'] >= reddit_ref[0]:
        reddit.loc[i,column_name_popularity] = 1
    else:
        reddit.loc[i,column_name_popularity] = 0
    if reddit.loc[i,'spread'] > (reddit_ref[1] + 3* reddit_ref[2]):
        reddit.loc[i,column_name_spreadoutlier] = 1
    else:
        reddit.loc[i,column_name_spreadoutlier] = 0


twitter.to_csv('twitter_3cri.csv', sep=',', header=True, index=True)

reddit.to_csv('reddit_3cri.csv', sep=',', header=True, index=True)
```

Co-movement detection: Change Point Detection

#installing appropriate libraries

```
#install.packages("bcp")
library(bcp);
#install.packages('RcppRoll') #for moving average
require(RcppRoll)
library(dplyr)
```

downloading & cleaning data

```
twitter <- read.csv("twitter_panel_labeled.csv")
reddit <- read.csv("reddit_panel_labeled.csv")
smp <- read.csv("SPX as of Dec 31 20211.csv")

#twitter_meme <- twitter[twitter$meme %in% 1,]
twitter_ticker <- unique(twitter$ticker)
twitter_index <- twitter_ticker %in% smp$Ticker
twitter_ticker <- twitter_ticker[!twitter_index]

twitter['sentim_posterior_daily'] <- NA
twitter['postvol_posterior_daily'] <- NA

#Filter meme Reddit posts and remove tickers from SMP500
#reddit_meme <- reddit[reddit$meme %in% 1,]
reddit_ticker <- unique(reddit$ticker)
reddit_index <- reddit_ticker %in% smp$Ticker
reddit_ticker <- reddit_ticker[!reddit_index]

reddit['sentim_posterior_daily'] <- NA
reddit['postvol_posterior_daily'] <- NA
```

```
candidates <- Reduce(intersect,list(twitter_ticker,reddit_ticker))
```

creating a function that performs CPD on on input data using bcp

```
func_CPD <- function(input_data_x, p0 = 0.75, graph = FALSE, moving_avg = 7)
{
  bol_detection = FALSE
  lenth_detection = 0
  x <- as.vector(na.omit(input_data_x))

  if (length(x) > (5 * moving_avg)) {
    x_ma <- roll_mean(x, n = moving_avg, align ='right', fill = NA)
    x <- tail(x_ma, -moving_avg)
  }
  else {
    #print(length(x))
    return_list <- list("moving_avg" = moving_avg, "bol_det" = NA, "len_det"
= NA, "max_prob" = NA, "posterior.prob" = NA)
    return(return_list)
  }

  bcp_x <- bcp(x, p0 = p0)#, return.mcmc = TRUE)

  if (graph){
    plot(bcp_x, main=paste("Posterior mean and probability of change, MA = ",
moving_avg))
  }

  bcp_sum <- as.data.frame(bcp_x$posterior.prob)
  bcp_sum$data <- x

  bcp_sum$id <- 1:length(x);
  sel <- bcp_sum[which(bcp_x$posterior.prob > p0),]

  if (nrow(sel) > 1){
    bol_detection = TRUE
    lenth_detection = nrow(sel)
  }
  return_list <- list("moving_avg" = moving_avg, "bol_det" =
as.integer(bol_detection), "len_det" = lenth_detection, "max_prob" =
max(na.omit(bcp_sum[,1])), "posterior.prob" = bcp_x$posterior.prob)
  return(return_list)
}
```

Sample of how CPD looks, using twitter data for ticker = AMC

```
set.seed(123)
input_data_x_vol <- twitter[twitter$ticker == 'AMC',]$avg_sentiment_score

cpd_res_vol = func_CPD(input_data_x_vol, 0.865, TRUE,1)

#Moving average creates more CPDs
cpd_res_vol = func_CPD(input_data_x_vol, 0.865, TRUE,7)
```

Preparing Twitter data to exclude tickers with less than 9 data points (days)

```
over_9 <- names(table(twitter$ticker))[table(twitter$ticker)>9]
twitter_over9 <- twitter[twitter$ticker %in% over_9,]

t_ticker <- unique(twitter_over9$ticker)

t_ticker <- Reduce(intersect,list(t_ticker,candidates))

set.seed(123)
```

Looping through all Twitter candidate stocks and performing BCP. Returning results in csv

```
results_twitt_MA1 <- data.frame(ticker = character()
                                ,sent_moving_avg = integer(), sent_bol_det =
logical(), sent_len_det = integer(), sent_max_prob = double()
                                ,vol_moving_avg = integer(), vol_bol_det =
logical(), vol_len_det = integer(), vol_max_prob = double())

for (i in 1:length(t_ticker)) {
#for (i in 1:10) {
  id <- t_ticker[i]
  cat(id,", ")


  #twitter sentiment data
  input_data_x_sent <- twitter[twitter$ticker == id,]$avg_sentiment_score
  n <- length(input_data_x_sent)
  cpd_res_sent = func_CPD(input_data_x_sent, 0.865, moving_avg = 1)

  cpd_res_sent$posterior.prob[n] <- NA

  twitter$sentim_posterior_daily[twitter$ticker == id] <-
c(cpd_res_sent$posterior.prob)

  #twitter post volume data

  input_data_x_vol <- twitter[twitter$ticker == id,]$post_count
  n <- length(input_data_x_vol)
  cpd_res_vol = func_CPD(input_data_x_vol, 0.994, moving_avg = 1)

  cpd_res_vol$posterior.prob[n] <- NA

  twitter$postvol_posterior_daily[twitter$ticker == id] <-
c(cpd_res_vol$posterior.prob)


  results_twitt_MA1[i,] = c(id
                  ,cpd_res_sent$moving_avg, cpd_res_sent$bol_det,
cpd_res_sent$len_det, cpd_res_sent$max_prob
                  ,cpd_res_vol$moving_avg, cpd_res_vol$bol_det,
cpd_res_vol$len_det, cpd_res_vol$max_prob)
}

results_twitt_MA1 <- na.omit(results_twitt_MA1)
#write.csv(results_twitt_MA1,"results_twitt_MA1.csv")
```

```
twitter <- twitter %>% mutate(CPD_IS_MEME =
if_else((twitter$sentim_posterior_daily + twitter$postvol_posterior_daily) >
1.9,1,0))

print_twitter <- twitter[!is.na(twitter$CPD_IS_MEME),]
write.csv(print_twitter,"CPD_results_twitt_DAILY.csv")
```

Preparing Reddit data to exclude tickers with less than 9 data points (days)

```
over_9 <- names(table(reddit$ticker))[table(reddit$ticker)>9]
reddit_over9 <- reddit[reddit$ticker %in% over_9,]

r_ticker <- unique(reddit_over9$ticker)


r_ticker <- Reduce(intersect,list(r_ticker,candidates))

set.seed(123)
over_9 <- names(table(reddit$ticker))[table(reddit$ticker)>9]
reddit_over9 <- reddit[reddit$ticker %in% over_9,]

r_ticker <- unique(reddit_over9$ticker)

r_ticker <- Reduce(intersect,list(r_ticker,candidates))

set.seed(123)
```

Looping through all Reddit candidate stocks and performing BCP. Returning results in csv

```
results_reddit_MA1 <- data.frame(ticker = character()
                            ,sent_moving_avg = integer(), sent_bol_det =
logical(), sent_len_det = integer(), sent_max_prob = double()
                            ,vol_moving_avg = integer(), vol_bol_det =
logical(), vol_len_det = integer(), vol_max_prob = double())

for (i in 1:length(r_ticker)) {
#for (i in 1:10) {
  id <- r_ticker[i]
  cat(id,", ")

  #twitter sentiment data
  input_data_x_sent <- reddit[reddit$ticker == id,]$avg_sentiment_score
  n <- length(input_data_x_sent)
  cpd_res_sent = func_CPD(input_data_x_sent, 0.865, moving_avg = 1)

  cpd_res_sent$posterior.prob[n] <- NA

  reddit$sentim_posterior_daily[reddit$ticker == id] <-
c(cpd_res_sent$posterior.prob)

  #twitter post volume data
  input_data_x_vol <- reddit[reddit$ticker == id,]$post_count
  n <- length(input_data_x_vol)
  cpd_res_vol = func_CPD(input_data_x_vol, 0.994, moving_avg = 1)
```

```
  cpd_res_vol$posterior.prob[n] <- NA

  reddit$postvol_posterior_daily[reddit$ticker == id] <-
c(cpd_res_sent$posterior.prob)


  results_reddit_MA1[i,] = c(id
                  ,cpd_res_sent$moving_avg, cpd_res_sent$bol_det,
cpd_res_sent$len_det, cpd_res_sent$max_prob
                  ,cpd_res_vol$moving_avg, cpd_res_vol$bol_det,
cpd_res_vol$len_det, cpd_res_vol$max_prob)
}

results_reddit_MA1 <- na.omit(results_reddit_MA1)

#write.csv(results_reddit_MA1,"results_reddit_MA1.csv")

reddit <- reddit %>% mutate(CPD_IS_MEME =
if_else((reddit$sentim_posterior_daily + reddit$postvol_posterior_daily) >
1.9,1,0))

print_reddit <- reddit[!is.na(reddit$CPD_IS_MEME),]
write.csv(print_reddit,"CPD_results_reddit_DAILY.csv")
```

Co-movement detection: Wavelet Coherence Analysis

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r}
#Libraries
library(biwavelet)
library(WaveletComp)
library(xts)
library(dplyr)
library(lubridate)
```

```{r}
twitter <- read.csv("twitter_panel_labeled.csv")
reddit <- read.csv("reddit_panel_labeled.csv")
smp <- read.csv("SPX as of Dec 31 20211.csv")

#Filter meme Tweets and remove tickers from SMP500
twitter_meme <- twitter[twitter$meme %in% 1,]
twitter_ticker <- unique(twitter_meme$ticker)
twitter_index <- twitter_ticker %in% smp$Ticker
twitter_ticker <- twitter_ticker[!twitter_index]

#Filter meme Reddit posts and remove tickers from SMP500
reddit_meme <- reddit[reddit$meme %in% 1,]
reddit_ticker <- unique(reddit_meme$ticker)
reddit_index <- reddit_ticker %in% smp$Ticker
reddit_ticker <- reddit_ticker[!reddit_index]
```
```

```
#Get list of stock tickers that are meme classified in both Reddit and
Twitter
meme_tickers <- twitter_ticker[twitter_ticker %in% reddit_ticker]



#Filter the data to only include non-SMP500 stocks that both Reddit and
Twitter classify as meme stocks
twitter_candidates <- twitter[twitter$ticker %in% meme_tickers,]
reddit_candidates <- reddit[reddit$ticker %in% meme_tickers,]

print(meme_tickers)
```
```{r}
#The wavelet function requires at lest 9 entries of each stock, so we filter
those that dont have 9 observations
over_9 <- names(table(twitter$ticker))[table(twitter$ticker) >= 9]
twitter_over9 <- twitter[twitter$ticker %in% over_9,]
t_ticker <- unique(twitter_over9$ticker)
t_ticker <- t_ticker[-c(34)]
t_ticker <- t_ticker[-c(874)]
t_ticker <- t_ticker[-c(928)]
t_ticker <- t_ticker[-c(1174)]
t_ticker <- t_ticker[-c(1231)]
t_ticker <- t_ticker[-c(1288)]
t_select <- c()


for (i in 1:length(t_ticker)) {
  id <- t_ticker[i]
  stock <- twitter_over9[twitter_over9$ticker == id,]
  stock <- stock %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
  stock <- stock %>%
  group_by(week = week(post_date))
  sentiment <- aggregate(stock$avg_sentiment_score, list(stock$week), FUN =
mean)
  price <- aggregate(stock$price, list(stock$week), FUN = mean)
  t <- data.frame(x = sentiment$x, y = price$x)
  wav <- analyze.coherency(t, my.pair = c("x","y"), dt = 1, lowerPeriod = 2,
upperPeriod = 16, verbose = FALSE)
  selection <- mean(wav$Coherence.avg.pval)
  t_select[[i]] <- selection
  print(i)
}

t_meme_selection <- t_select >= 0.7
t_meme_stocks <- t_ticker[t_meme_selection]
t_meme_stocks
```
```{r}
#The wavelet function requires at lest 9 entries of each stock, so we filter
those that dont have 9 observations
over_9 <- names(table(reddit$ticker))[table(reddit$ticker) >= 9]
reddit_over9 <- reddit[reddit$ticker %in% over_9,]
r_ticker <- unique(reddit_over9$ticker)
```

```
r_ticker <- r_ticker[-c(431)]
r_select <- c()


for (i in 1:length(r_ticker)) {
  id <- r_ticker[i]
  stock <- reddit_over9[reddit_over9$ticker == id,]
  stock <- stock %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
  stock <- stock %>%
  group_by(week = week(post_date))
  sentiment <- aggregate(stock$avg_sentiment_score, list(stock$week), FUN =
mean)
  price <- aggregate(stock$price, list(stock$week), FUN = mean)
  r <- data.frame(x = sentiment$x, y = price$x)
  wav <- analyze.coherency(r, my.pair = c("x","y"), dt = 1, lowerPeriod = 2,
upperPeriod = 16, verbose = FALSE)
  selection <- mean(wav$Coherence.avg.pval)
  r_select[[i]] <- selection
  print(i)
}

r_meme_selection <- r_select >= 0.7
r_meme_stocks <- r_ticker[r_meme_selection]
r_meme_stocks
```

```{r}
r_meme_stocks
t_meme_stocks
```

```{r}
AMC_twitter <- twitter_candidates[twitter_candidates$ticker == "AMC",]
AMC_twitter <- AMC_twitter %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
AMC_twitter <- AMC_twitter %>%
  group_by(week = week(post_date))

AMC_sent <- aggregate(AMC_twitter$avg_sentiment_score,
list(AMC_twitter$week), FUN = mean)
AMC_price <- aggregate(AMC_twitter$price, list(AMC_twitter$week), FUN = mean)

AMC_t <- data.frame(x = AMC_sent$x, y = AMC_price$x)

AMC_wav <- analyze.coherency(AMC_t, my.pair = c("x","y"))

AMC_tp <- wc.image(AMC_wav, which.image = "wc", n.levels = 250,
legend.params = list(lab = "cross-wavelet power levels"),
timelab = "", periodlab = "period (days)")
```
```{r}
```

```
LYFT_twitter <- twitter[twitter$ticker == "LYFT",]

LYFT_twitter <- LYFT_twitter %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
LYFT_twitter <- LYFT_twitter %>%
  group_by(week = week(post_date))

LYFT_sent <- aggregate(LYFT_twitter$avg_sentiment_score,
list(LYFT_twitter$week), FUN = mean)
LYFT_price <- aggregate(LYFT_twitter$price, list(LYFT_twitter$week), FUN =
mean)

LYFT_t <- data.frame(x = LYFT_sent$x, y = LYFT_price$x)

LYFT_wav <- analyze.coherency(LYFT_t, my.pair = c("x","y"), dt = 1,
lowerPeriod = 2, upperPeriod = 16, verbose = FALSE)

LYFT_tp <- wc.image(LYFT_wav, which.image = "wc", n.levels = 250,
legend.params = list(lab = "cross-wavelet power levels"),
timelab = "", periodlab = "period (days)")
```

```{r}
GME_twitter <- twitter_candidates[twitter_candidates$ticker == "GME",]
GME_twitter <- GME_twitter %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
GME_twitter <- GME_twitter %>%
  group_by(week = week(post_date))

GME_sent <- aggregate(GME_twitter$avg_sentiment_score,
list(GME_twitter$week), FUN = mean)
GME_price <- aggregate(GME_twitter$price, list(GME_twitter$week), FUN = mean)

GME_t <- data.frame(x = GME_sent$x, y = GME_price$x)

GME_wav <- analyze.coherency(GME_t, my.pair = c("x","y"))
GME_tp <- wc.image(GME_wav, which.image = "wc", n.levels = 250,
legend.params = list(lab = "cross-wavelet power levels"),
timelab = "", periodlab = "period (days)")
```

```{r}
KOSS_twitter <- twitter_candidates[twitter_candidates$ticker == "KOSS",]
KOSS_twitter <- KOSS_twitter %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
KOSS_twitter <- KOSS_twitter %>%
  group_by(week = week(post_date))

KOSS_sent <- aggregate(KOSS_twitter$avg_sentiment_score,
list(KOSS_twitter$week), FUN = mean)
KOSS_price <- aggregate(KOSS_twitter$price, list(KOSS_twitter$week), FUN =
mean)
```

```
KOSS_t <- data.frame(x = KOSS_sent$x, y = KOSS_price$x)

KOSS_wav <- analyze.coherency(KOSS_t, my.pair = c("x","y"))
KOSS_tp <- wc.image(KOSS_wav, which.image = "wc", n.levels = 250,
legend.params = list(lab = "cross-wavelet power levels"),
timelab = "", periodlab = "period (days)")
```

```{r}
NAKD_twitter <- twitter_candidates[twitter_candidates$ticker == "NAKD",]
NAKD_twitter <- NAKD_twitter %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
NAKD_twitter <- NAKD_twitter %>%
  group_by(week = week(post_date))

NAKD_sent <- aggregate(NAKD_twitter$avg_sentiment_score,
list(NAKD_twitter$week), FUN = mean)
NAKD_price <- aggregate(NAKD_twitter$price, list(NAKD_twitter$week), FUN =
mean)

NAKD_t <- data.frame(x = NAKD_sent$x, y = NAKD_price$x)

NAKD_wav <- analyze.coherency(NAKD_t, my.pair = c("x","y"))
NAKD_tp <- wc.image(NAKD_wav, which.image = "wc", n.levels = 250,
legend.params = list(lab = "cross-wavelet power levels"),
timelab = "", periodlab = "period (days)")
```

```{r}
NOK_twitter <- twitter_candidates[twitter_candidates$ticker == "NOK",]
NOK_twitter <- NOK_twitter %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
NOK_twitter <- NOK_twitter %>%
  group_by(week = week(post_date))

NOK_sent <- aggregate(NOK_twitter$avg_sentiment_score,
list(NOK_twitter$week), FUN = mean)
NOK_price <- aggregate(NOK_twitter$price, list(NOK_twitter$week), FUN = mean)

NOK_t <- data.frame(x = NOK_sent$x, y = NOK_price$x)

NOK_wav <- analyze.coherency(NOK_t, my.pair = c("x","y"))
NOK_tp <- wc.image(NOK_wav, which.image = "wc", n.levels = 250,
legend.params = list(lab = "cross-wavelet power levels"),
timelab = "", periodlab = "period (days)")
```

```{r}
SNDL_twitter <- twitter_candidates[twitter_candidates$ticker == "SNDL",]

SNDL_twitter <- SNDL_twitter %>%
  mutate(post_date = lubridate::parse_date_time(post_date, "mdy")) %>%
  arrange(post_date)
SNDL_twitter <- SNDL_twitter %>%
  group_by(week = week(post_date))
```

```
SNDL_sent <- aggregate(SNDL_twitter$avg_sentiment_score,
list(SNDL_twitter$week), FUN = mean)
SNDL_price <- aggregate(SNDL_twitter$price, list(SNDL_twitter$week), FUN =
mean)

SNDL_t <- data.frame(x = SNDL_sent$x, y = SNDL_price$x)

SNDL_wav <- analyze.coherency(SNDL_t, my.pair = c("x","y"))
SNDL_tp <- wc.image(SNDL_wav, which.image = "wc",  n.levels = 250,
legend.params = list(lab = "cross-wavelet power levels"),
timelab = "", periodlab = "period (days)")
```

REFERENCES

Aloosh, A., Choi, H.-E., & Ouzan, S. (2021). *Meme Stocks and Herd Behavior* (SSRN Scholarly Paper No. ID 3909945). Rochester, NY: Social Science Research Network.

AlZaabi, S. A. (2021). *Correlating Sentiment in Reddit's Wallstreetbets with the Stock Market Using Machine Learning Techniques*. 54.

Anderson, J. P., Kidd, J., & Mocsary, G. A. (2022). *Social Media, Securities Markets, and the Phenomenon of Expressive Trading* (SSRN Scholarly Paper No. 3834801). Rochester, NY: Social Science Research Network.

Barry, D., & Hartigan, J. A. (1993). A Bayesian Analysis for Change Point Problems. *Journal of the American Statistical Association*, *88*, 309–319.

Birgersson, J., & Carlén, S. (2021). *Investigating the Nature of Retail Investor Activity*. Retrieved from https://gupea.ub.gu.se/handle/2077/69113

Buz, T., & de Melo, G. (2021). Should You Take Investment Advice From WallStreetBets? A Data-Driven Approach. *ArXiv:2105.02728 [Cs, q-Fin]*. Retrieved from http://arxiv.org/abs/2105.02728

Cahill, D., Liu, Z. F., & Smales, L. A. (2022). *Is investor attention on r/wallstreetbets different?* (SSRN Scholarly Paper No. 4069356). Rochester, NY: Social Science Research Network.

Cao, H., & Liu, S. (2022, March 26). *The Consideration of Meme Stock*. 211–214. Atlantis Press.

Chiu, V., & Yahya, M. A. (2021). The Meme Stock Paradox. *Corporate and Business Law Journal*, *3*, 51–101.

Choy, J., Wang, B., AlShelahi, A., & Saigal, R. (2021). *Investor Impatience and Financial Markets: The Case of the Short Squeeze of Meme Stocks* (SSRN Scholarly Paper No. 3908732). Rochester, NY: Social Science Research Network.

Corporate Finance Institute. (2021). Short Squeeze. Retrieved April 18, 2022, from Corporate Finance Institute website: https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/short-squeeze/

Costola, M., Iacopini, M., & Santagiustina, C. R. M. A. (2021). On the "mementum" of meme stocks. *Economics Letters*, *207*, 110021.

Dechow, P., Lawrence, A., Luo, M., & Stamenov, V. (2021). *Media Attention and Stock Categorization: An Examination of Stocks Hyped to Benefit from the Olympics* (SSRN Scholarly Paper No. 3881333). Rochester, NY: Social Science Research Network.

Dong, H., Ren, J., Padmanabhan, B., & Nickerson, J. V. (2022). How are social and mass media different in relation to the stock market? A study on topic coverage and predictive value. *Information & Management*, *59*, 103588.

Egan, M. (2021, January 27). Stock market news today: Dow and S&P 500 updates. Retrieved April 25, 2022, from CNN Business website: https://www.cnn.com/business/live-news/stock-market-news-012721/index.html

Erdman, C., & Emerson, J. W. (2008). bcp: An R Package for Performing a Bayesian Analysis of Change Point Problems. *Journal of Statistical Software*, *23*, 1–13.

Fitzgerald, M. (2021a, January 28). Robinhood restricts trading in GameStop, other names involved in frenzy. Retrieved April 25, 2022, from CNBC website: https://www.cnbc.com/2021/01/28/robinhood-interactive-brokers-restrict-trading-in-gamestop-s.html

Fitzgerald, M. (2021b, January 29). Robinhood is still severely limiting trading, customers can only buy one share of GameStop. Retrieved April 25, 2022, from CNBC website: https://www.cnbc.com/2021/01/29/robinhood-is-still-severely-limiting-trading-gamestop-holders-can-only-buy-one-additional-share.html

Fitzgerald, M. (2021c, February 2). Robinhood eases trading limits on restricted stocks, customers can buy 100 GameStop shares now. Retrieved April 25, 2022, from CNBC website: https://www.cnbc.com/2021/02/02/robinhood-raises-trading-limits-on-restricted-stocks-customers-can-buy-100-gamestop-shares-now.html

Gianstefani, I., Longo, L., & Riccaboni, M. (2022). *The Echo Chamber Effect Resounds on Financial Markets: A Social Media Alert System for Meme Stocks* (SSRN Scholarly Paper No. 4053771). Rochester, NY: Social Science Research Network.

Gric, Z., Bajzík, J., & Badura, O. (2021). *Does Sentiment Affect Stock Returns? A Meta-analysis Across Survey-based Measures*. Retrieved from https://www.cnb.cz/en/economic-research/research-publications/cnb-working-paper-series/Does-Sentiment-Affect-Stock-Returns-A-Meta-analysis-Across-Survey-based-Measures-00001/

Gupta, A. (2021). *Wall Street vs r/wallstreetbets: Exploring the Predictive Power of Retail Investors on Equity Prices*. 8.

Haan, S. C. (2021). *Is American Shareholder Activism a Social Movement?* (SSRN Scholarly Paper No. 3974031). Rochester, NY: Social Science Research Network.

Hayes, A. (2022, February 22). Meme Stock. Retrieved April 18, 2022, from Investopedia website: https://www.investopedia.com/meme-stock-5206762

Li, Y. (2021, January 31). Robinhood to continue trading limits on Monday, customers can still only buy one GameStop share. Retrieved April 25, 2022, from CNBC website: https://www.cnbc.com/2021/01/31/robinhood-to-continue-trading-limits-on-monday-customers-can-still-only-buy-one-gamestop-share.html

Murphy, M. (2021, February 4). Robinhood lifts trading restrictions on GameStop, AMC stock. Retrieved April 25, 2022, from MarketWatch website: https://www.marketwatch.com/story/robinhood-lifts-trading-restrictions-on-gamestop-amc-stock-11612499971

Otto, S. (2019, September 28). Comparison of change point detection methods. Retrieved April 26, 2022, from https://www.marinedatascience.co/blog/2019/09/28/comparison-of-change-point-detection-methods//

Ricci, G., Sautter, C. M., & Alberto, S. (2021). *Corporate Governance Gaming: The Collective Power of Retail Investors* (SSRN Scholarly Paper No. 3815088). Rochester, NY: Social Science Research Network.

Schroth, C., Siebert, J., & Groß, J. (2021, September 7). Time Traveling with Data Science: Focusing on Change Point Detection in Time Series Analysis (Part 2) - Blog des Fraunhofer IESE. Retrieved April 21, 2022, from Fraunhofer IESE website: https://www.iese.fraunhofer.de/blog/change-point-detection/

Stiebel, J. H. (2021). *GameStop: Power to the Players?* (SSRN Scholarly Paper No. 3883295). Rochester, NY: Social Science Research Network.

Takagi, H. (2021). Exploring the Endogenous Nature of Meme Stocks Using the Log-Periodic Power Law Model and Confidence Indicator. *ArXiv:2110.06190 [q-Fin, Stat]*. Retrieved from http://arxiv.org/abs/2110.06190

Vasileiou, E. (2021). Does the short squeeze lead to market abnormality and antileverage effect? Evidence from the Gamestop case. *Journal of Economic Studies*, *ahead-of-print*. https://doi.org/10.1108/JES-04-2021-0210

Vasileiou, E., Bartzou, E., & Tzanakis, P. (2021). *Explaining Gamestop Short Squeeze using Intraday Data and Google Searches* (SSRN Scholarly Paper No. 3805630). Rochester, NY: Social Science Research Network.

Vasileiou, E., & Tzanakis, P. (2022). *Sentiment Analysis and Wavelet Coherence Analysis: The AMC Theatres Case* (SSRN Scholarly Paper No. 4004619). Rochester, NY: Social Science Research Network.

Wang, C., & Luo, B. (2021). Predicting $GME Stock Price Movement Using Sentiment from Reddit r/wallstreetbets. *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, 22–30. Online: -.

Yahya, M. A., & Chiu, V. (2021). *The Meme Stock Paradox* (SSRN Scholarly Paper No. ID 3918506). Rochester, NY: Social Science Research Network.

Zheng, X., Tian, H., Wan, Z., Wang, X., Zeng, D. D., & Wang, F.-Y. (2022). Game Starts at GameStop: Characterizing the Collective Behaviors and Social Dynamics in the Short Squeeze Episode. *IEEE Transactions on Computational Social Systems*, *9*, 45–58.