The predictive power in online investing chatter.

Analysis of the r/WallStreetBets.

Master thesis

submitted by

Piotr Antoniak

Student no. 187829

Barcelona, 7th June, 2021

Contents

List of Figures

List of Tables

Predictive power in online investing chatter.

Piotr Anatoniak

7th June, 2021

Abstract

This work investigates predictive power of sentiment of Reddit forum users - r/WallStreetBets. Using data from 2016 to May 2021, I extract such sentiment from submissions and comments and check whether a quantitative model predicting 5 days return of stocks incorporating such sentiment outperforms models that do not have access to it. I find that the approach using both past prices of a stock and sentiment extracted from the Reddit is able to outperform its peer trained on past prices only. This suggests that comments and submissions posted on WallStreetBets convey information useful in making predictions. Those striking results are especially visible when investigating the latest Minsky moment - series of liquidations in late February and first half of March 2020.

1 Introduction

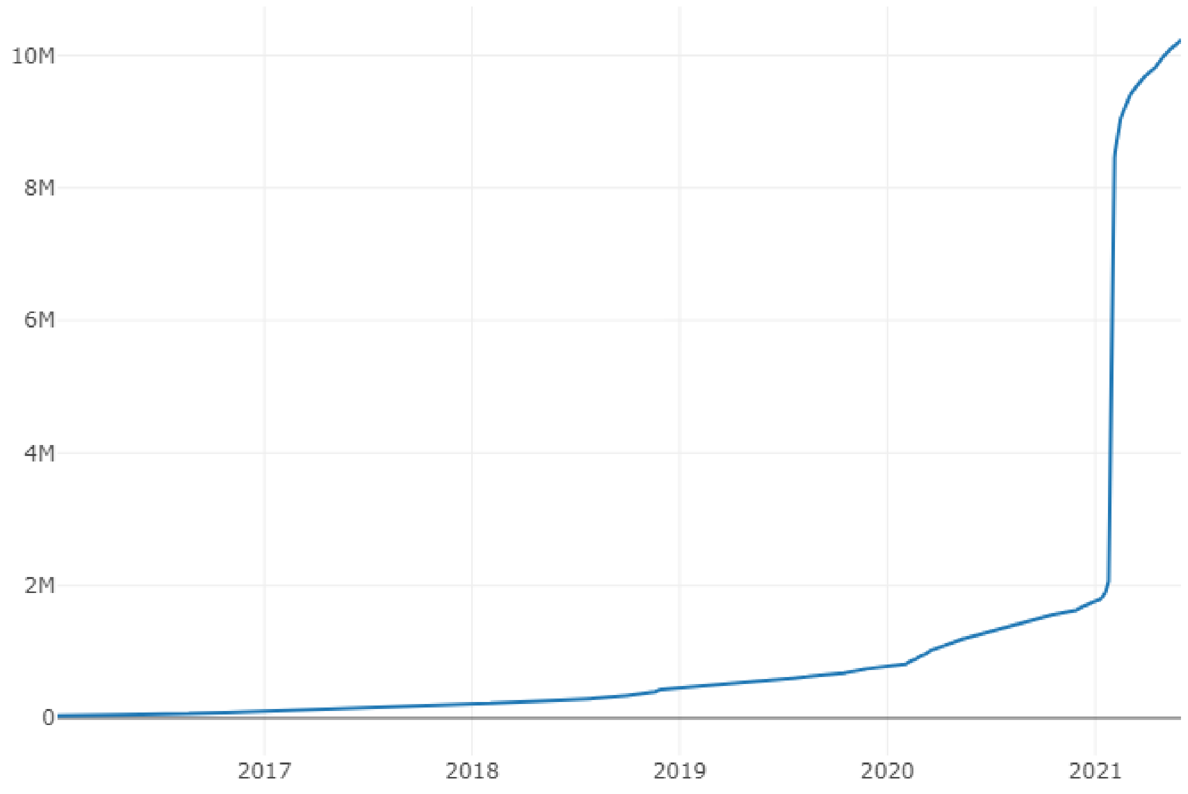
Following the GameStop (further referred to also as GME) short squeeze in late January of 2021, the Reddit forum r/WallStreetBets (WSB) caught the attention of mass media around the world, celebrities and politicians. Shortly after the January events, on the 18th of February, Congress called upon Vlad Tenev, the CEO of Robinhood - popular among retail traders brokerage, Gabe Plotkin - founder of Melvin Capital - the fund that lost over half of its value during the squeeze, Kenneth Griffin - the CEO of Capital Securities and Keith Gill - a user of WSB and one of the earliest GME "bulls". The theme of the hearing revolved around the possibility of stock manipulations and addressed two concerns:

- Various brokerages, including Robinhood, blocked the ability to buy GME stock on the day it's price peaked,
- The explosion of WSB users - from 1.75 million at the end of 2020 to 9 million in a span of a month (see **Figure 1**). This raised the possibility that WallStreetBets may become some sort of "pump&dump" scheme hurting not very sophisticated retail traders.

However, while WallStreetBets will be probably forever associated with the events of January, it wasn't the first time that financial media in the USA allocated significant attention to discuss the developments within the forum.

The first such event happened at the start of 2019 when one of WSB users, by exploiting the faulty algorithm calculating the leverage, forced the Robinhood to entirely ban a box-spread strategy on it's platform. Roughly ten months later, Robinhood brokerage faced another problem when one of the WSB users managed to create a position with over 50 times leverage (the maximum legal is two times). He quickly shared his strategy which was followed by multiple people building multi

Figure 1: The number of subscribers to r/WallStreetBets



sourced from https://subredditstats.com/r/wallstreetbets

million positions while posting miniscule collateral. Although there was no damage done to the functioning of the market nor Robinhood didn't face any liquidity issues, various users started to plan "Robinhood funded hostile takeovers" of smaller companies during the few hours the exploit was functioning. Following this event, the American program CNBC allocated air time for experts to explain what was happening on a niche forum of stock pickers. One of them, controversially called the WSB users "psychopaths".

The next wave of attention can be tracked to early 2020. On 26th of February, the Bloomberg Business featured the Reddit mascot on the cover of its weekly issue and went into detailed explanation how retail traders may be forcing the market makers to bid up popular stocks by exploiting the funds that use delta hedging strategies.

Although the perception of the WSB changed overtime - from people taking on high risk and not understanding what they may be dealing with, to people taking on the high risk and not understanding what they may be dealing with, while trying to punish hedge funds that have sold lots of stock that they like. By some, the short squeeze that GME have experience is refereed to as a fight of David vs Goliath (for a discussion see [Aharon et al., 2021]). The people posting on WallStreetBets like to share their ideas about their current portfolio composition. This is often followed by "gain or loss porn" - a high payout from a highly leveraged position using options, especially weeklys (contracts that expire within a week), or the total wipe out of the account.

Given that the creators of posts and comments like to usually express their opinion towards the stocks and how the price of the asset will change in the near future, I suspect that this sentiment may convey some information that could be quantified and used in a predictive model.

To investigate it, I utilize the latest techniques in Natural Language Processing to classify all relevant comments and posts into one of the three categories: "bullish", "neutral" and "bearish". Next, I use this sentiment as an input to a classifying model trying to predict the sign of stock return five days ahead. By comparing the model that was trained only on the past prices (further refereed to as $model_T$) and the model that had access to both past prices and features extracted from comments ($model_N$). I find that introduction of sentiment features improves the performance of the model.. Evaluated in the period July 2020- May 2021, $model_N$ is able to significantly outperform $model_T$. However, the results are due to it's ability to correctly predict the return sign of very volatile assets that have become WallStreetBets favourites. When such stocks are excluded from the sample, addition of sentiment degrades the performance of model.

The popularity of r/WallStreetBets has also caught the attention of researches as the forum can viewed as populated by retail traders with high appetite for the leverage. However, so far the results do not paint a single picture. For example, Long et al. [C. Long et al., 2021] investigate comments and submissions for each minute during the recent GME short squeezes and find that sentiment is a poor measure predictor of Game Stops returns on 1min time frame. This is similar with conclusions of Eaton et al. [Eaton et al., 2021] that Robinhood traders, as proxied by r/WSB, are mostly noise traders. However, this evidence goes in contrast of work of Bradley et al. [Bradley et al., 2021]. By investigating over two thousands of "Due Dilligence" posts, they find that following positive submission about the stock, the retail trading of assets mentioned rises and the abnormal 2 days returns are roughly 1.12% while the cumulative, one-quarter return, following positive submissions is 6%.

However neither of the above mentioned works attempted to label each relevant comment into one of three categories: "bearish", "neutral" and "bullish" nor uses such labels as an input to a classifier thus making my work quite novel. This approach, instead of focusing on the quality of the posts and submissions, looks more into the herd behaviour of the users of WallStreetBets. While such analysis using sentiment data is not something new as the literature around this topic is quite rich (see for examples: [Fisher and Statman, 2000],[Schmeling, 2009],[Arias et al., 2014]), I am not aware of any previous work trying to deploy recent breakthroughs in Natural Language Processing.

2 Data

2.1 Textual Data

Textual data used in this work can be split into two types: Submission, a post created by a user that usually guides the discussion; Comments, replies to either submission or other comments made under the submission.

The main difference is usually their length, especially when a submission is called as "DD", short for "Due Diligence" - a lengthy investigation why, given the current macroeconomics conditions, state of the market or the quality of the firm, certain stock is either overvalued or undervalued. While one could distinguish between a submission and comments and give them different weights based on:

- their popularity as measured by the amount of up-votes they receive - an equivalent of reddit "likes",

- the number of comments they receive, whether they are direct responses to the submission or responses to other comments,
- the number of awards a post received,

my analysis is invariant to those variables. In detail, my work not only ignores the above mentioned differences but also ignores all the comments that do not include the ticker of a stock traded in Russell 3000 index. For example, consider two comments:

- "With the amount of vaccines and chips being administered daily, Microsoft must be making bucks, 260c next month."
- "With the amount of vaccines and chips being administered daily, MSFT must be making bucks, 260c next month."

It is clear that both convey the same message about the same stock. However, the first comment is rejected and not taken into account as it does not contain the ticker of the stock Microsoft - MSFT, but rather the name of the company. Therefore, from now on, I will use word "comment" or "submission" as interchangeable with the meaning "a body of text that contains a ticker of a stock traded publicly, listed in the Russell 3000, that can be labeled as either bearish, bullish or neutral" while "bullish", "neutral" and "bearish" will refer to "amount of comments each day that were labeled to belong to one of those three categories".

While this approach may seem to reject lots of data, a nice rule of WSB - "position or ban" - ensures that most comments and submissions that initiate the discussion about the stock also convey the ticker of the company discussed. On top of that, some tickers, that coincide with words that can be found in english dictionary are also rejected. For example, tickers suchs as SHOP (Shopify), EDIT (Editas Medicine) or WU (The Western Union Company) are discarded.

Therefore, while the entire WSB is made of around 50 millions submissions and comments, my final database is made of 5 millions rows, each containing a ticker of a stock, the label associated with the comment and the date on which the comment was made. The textual data was collected using Pushshift API [Baumgartner et al., 2020].

2.2 Historical Prices

Required in the analysis is also historical daily data on prices of stocks, which was acquired using the AlphaVantage API (https://www.alphavantage.co/). The data consists of roughly 9 million rows of observations each representing the price of a stock at the open, close, daily high and daily low. Using those four financial time series and VIX, I create 80 price indicators. I do it using a public Technical Analysis Library [Padial, 2018]. The full list of the indicators used in this work is presented in Appendix A. Further, each of the series, when necessary, is adjusted. The adjustment process follows:

- Consider a series A, made of positive, non-infinity numbers. The series B is just $2A$.
- If the distribution of the indicator calculated using series A and series B is exactly the same, do nothing.
- Else, divide the series of values of this indicator by the series used to calculate it.

This process assures that if two stock prices series are perfectly correlated and move together, the indicators obtained from those two series will be identical. The use of this approach is dictated by the need to make the generated features independent of the price level and only represent it's evolution over the past.

3 Methodology

In Appendix B, some of the technical details and Deep Learning terminology are briefly explained. To make the explanations very clear, I draw some comparisons to the methods that are commonly used in the field of economics.

3.1 Extraction of the Sentiment - Bidirectional Embedding Representation from Transformers

To extract the sentiment of each comment deemed relevant for the task, that is, having a ticker of a stock listed on Russell 3000, I use the Bidirectional Embedding Representation from Transformers (BERT, [Devlin et al., 2018]) model for automated classification of each of 5 million documents. Published by the Google AI language research team in 2018, BERT has become a milestone in Natural Language Processing by obtaining State-Of-The-Art results on the most popular NLP tasks that act as a benchmark for testing models.

The discussion of how BERT is able to achieve its results is beyond the scope of this work. Therefore, I will limit myself to only describe how I utilize this model. However, some intuition is provided to make a reader familiar with why this model was chosen and how it manages to classify the comments.

The BERT can be thought of as a model that, given an input word, returns it's representation in a multidimensional form. Therefore, given two different words, it is possible to represent how closely related they are. This can be used to represent simple concepts such as King - Man + Woman -> Queen and Madrid - Spain + Germany -> Berlin.

However, the out-of-the-box BERT is not capable of extracting sentiment because:

- Language used within the WallStreetBets forum is quite different from "normal" daily English.
- The BERT itself is just a model that understands how one word can be related to another.

To deal with that, I first "adapt" the BERT to the domain specific language, that is WSB. This can be achieved by further training the model on a corpus of text - 600MB of sentences sourced from WallStreetBets. The process of adapting the model is no different than training and the only difference is in the data fed into the model. Therefore, the process is quite simple:

- Feed into the model a batch of documents that has some words randomly (20%) hidden from the model - MASKED, that is, replaced by a token MASK; and ask for prediction of the concealed word. On top of that, BERT has to predict whether the current sentence is from the same document as the previous one, or a different document.
- Calculate the gradient with respect to the input and backpropagate it through the network.

This process is repeated for many steps with the intention to adapt the BERT into domain specific language. In my case, I do this for 50000 steps using 16 documents each time and gradient accumulation set to eight.

To evaluate the performance of an adapted model to its original, one can ask to predict MASKED words in a sentence that is representative for the specific domain. For example, a not adapted BERT predicts that [MASK] in the sentence “Any thoughts about $SNAP calls before [MASK]?” is a word “dinner” with the probability of 15% while the adapted BERT correctly predicts the word “earnings” with the probability of 51%. When the word “calls” is masked, the first model predicts a word “per” with probability of 25% while the adapted one, once again, correctly returns “calls” with probability of 54%.

Having the BERT that understands the jargon of WallStreetBets, the next task is to train it on a set of labeled sentences sourced from the forum. To do so, I extend the model with two Feed Forward Layers and the softmax output layer which returns probabilities of a comment belonging to one of the three classes. The model is trained for 2 epochs on 30000 labeled comments and achieves an accuracy of 96% on unseen dataset consisting of 10000 comments. A sample of BERT classification of text sourced from WallStreetBets is presented below, in the **Table 1**.

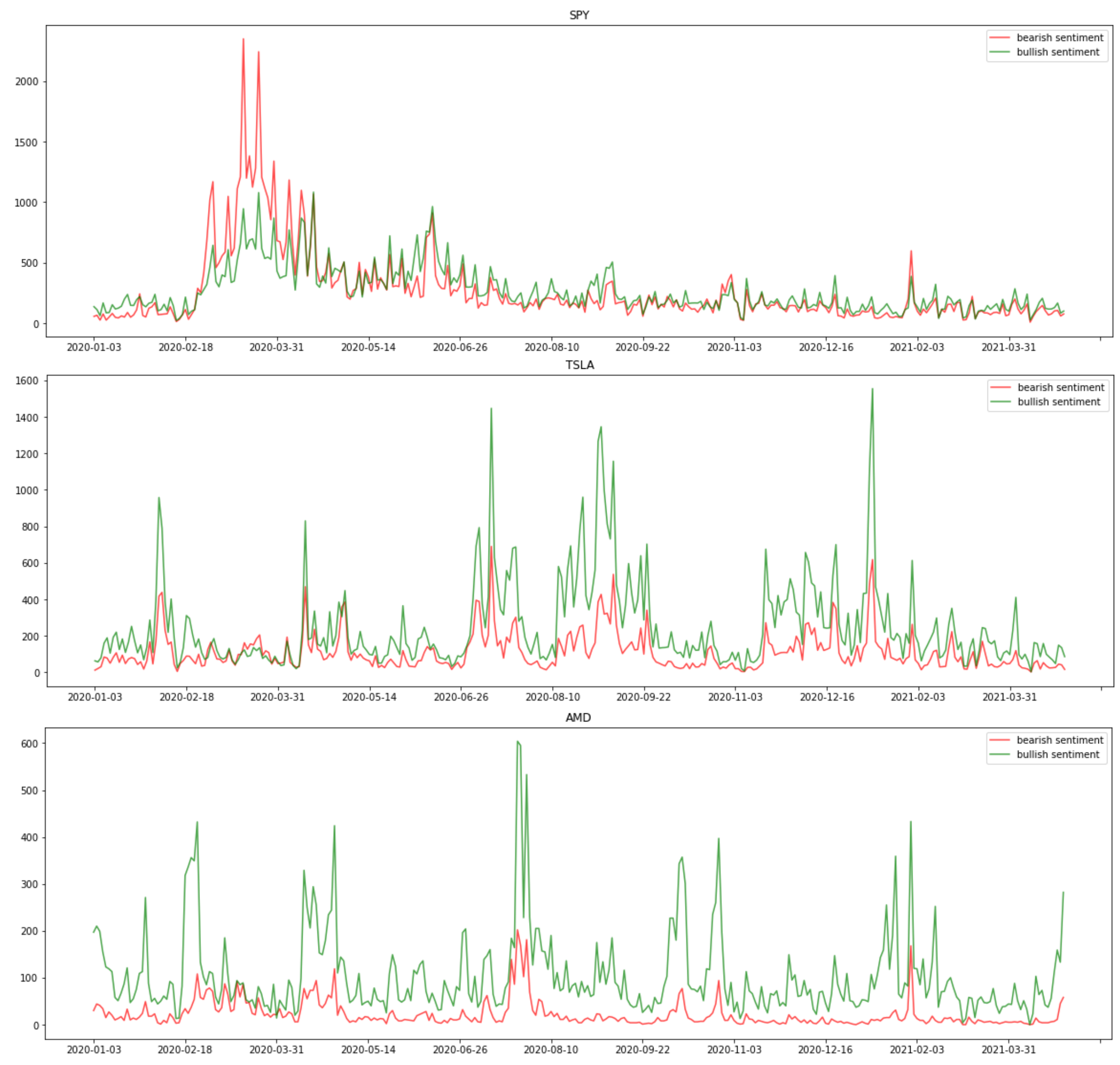
Table 1: Some examples of BERT classification.

Text	Label	Probability
“Any thoughts about $SNAP before earnings?”	Neutral	99.9%
“People are increasingly scared to go to the gym. LA Fitness just closed all of its gyms. If PTON can lower their prices or release a cheaper model, they could clean up up with all the stressed out at-home fitness people.”	Bullish	60%
“Damn PACB hitting its 52 week high. Good stuff, will have to research into this.”	Neutral	94%
“I’m down 14%, NVDA is saving me. Up 34% in the last 3 months.”	Bullish	92%
“Picked up 32 12/21 AMD puts”	Bearish	96%
“TSLA time to recoup the basis and moon past”	Bullish	92%
“Didn’t sell my SPY calls”	Bullish	35%
“Sold my SPY puts”	Neutral	67%
“Are you holding onto you SPY puts? I have some too, hoping to buy more tomorrow”	Bearish	99%
“Containers prices are up again. 3 times the historical mean. Someone give me a play.”	Neutral	80%

Text column contains the body of text of various comments sourced from r/WallStreetBets, the Label column tells which label was assigned to each of comments while the Probability (table reports the highest probability) column tells how certain the model was about the label.

One way to visualize the sentiment is to just plot it. The **Figure 2**, plots the evolution of sentiment for SPY (SP500 etf), TSLA (Tesla) and AMD (Advanced Micro Devices) for period 2020 - middle of the April 2021. Comments made during time of the market being open are mapped into the close on that day. Comments made while the market was closed, are mapped to the next close. Since I’m dealing with only daily returns, this approach ensures that there is no leakage.

Figure 2: Bullish and Bearish sentiment for few stocks



Red lines represent the amount of comments and submissions classified as bearish, green - bullish.

3.1.1 Creation of features representing the current sentiment and its past evolution.

Having this extended, domain-adapted BERT, it is now possible to extract sentiment for each comment. However, having only 3 sentiment features may not be enough as it does convey any information about the past sentiment and it's evolution over time. Therefore, using the bullish, neutral and negative labels, I create a wide range of features for each type of label:

- Slow Moving Average (SMA) with the length of the window of 30 and 200 trading days, the minimum period is set to 5 and 20 days, respectively,
- Fast Moving Average (FMA) with the length of the window of 10 and 50 trading days, the minimum period is set to 5 and 10 days, respectively,
- Exponential Moving Average (EMA) with the span of 10 and 30 trading days and minimum period of 5 days,

- By subtracting the SMA (30 days) from FMA (10) and EMA(30 days) from EMA(10) I create an indicator called MACD,
- The ratio of number of labeled comments to it's lagged values (at 1, 5 and 20 days),
- The ratio of SMA and FMA (both 30 and 10 days) to the total number of labeled comments,
- The ratio of of labels to lagged values of the total number of labeled comments,
- Ratios: bearish to bullish, bearish to neutral and bullish to neutral,
- Rolling window of 20 days of standard deviation for all 3 classes.

Which results in 60 features. On top of that, all SMAs, FMAs and EMAs were adjusted by dividing them by the series used to calculate them. I do it for exactly the same reason why some stock prices series had to be adjusted. For example, the adjusted Slow Moving Average with a window of 30 days takes the form:

$$SMA_{30} = (1/30\sum_{i=0}^{29} x_{t-i})/x_t \tag{1}$$

3.2 Choice of predictive models

To investigate whether the sentiment extracted from the forum has any predictive power, I propose to compare two models - $model_T$, trained only on the past prices and the features extracted from their evolution.The second model,$model_N$ - is an extension of a previous one containing features extracted from labeled comments. On top of that, I use a third model, further refereed to as a $model_{transfer}$ that plays a crucial role in making the comparison between $model_T$ and $model_N$ possible.

Given the amount of data in possession and the high dimensionality of it - 80 features extracted from the past prices and 60 from labeled comments, the choice is to use a Neural Network (NN) model, specifically the Feed Forward Network(FNN). The NN is a model that tries to approximate some function f^*. A FFN that maps input x into category y, by defying a mapping $y = f^*(x, \theta)$ is a classifying model.

The Feed Forward Network is built from a few elements. Let σ be an activation function. A simple choice for such function is sigmoid:

$$\sigma = 1/(1 + exp(-x)) \tag{2}$$

Then let x be an matrix of observations, W is a matrix of Weights and b, a vector of biases.
Then the function:

$$f^0(x) = \sigma(x^T W^0 + b^0) \tag{3}$$

is a simple Neural Network and, at the same time, a Logistic Regression. When the sigma activation function is replaced with a linear function:

$$f^0(x) = x \tag{4}$$

the NN turns out to be Linear Regression.

By combining the multiple functions (3) in a chain, one creates a Deep Feed Forward Network. For example:

$$f^*(x) = \sigma(\sigma(\sigma(x^T W^0 + b^0)^T W^1 + b^1)^T W^2 + b^b) \tag{5}$$

or

$$f^*(x) = f^2(f^1(f^0(x))) \tag{6}$$

Both represent the same model that is created by combining 3 functions. In equation (5), $\sigma(x^T W^0 + b)$ and f^0 in equation (6) are called called the first layer of the network. $\sigma(x^T W^1 + b^1)$ and f^1 are called the second layer. The last layer of the network is called an output layer. Because we are usually only interested in the output of the last layer, and the values of the intermediate layers are of less importance, they are called hidden layers.

While the x is known, the coefficients W and b have to be estimated. The process to find them is very similar to finding the coefficients of Linear Regression models using gradient descent. However, in my case, I'm not dealing with a continuous response but a binary class. Therefore, the loss function takes the form of:

$$J_{loss} = -1/N \sum_{i=1}^{N} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i)) \tag{7}$$

The minimum of this function can still be found[1] using the gradient descent. In practice, stochastic version of the algorithm is preferred, more specifically, some variants of it. Here, the Adam optimizer [Kingma and Ba, 2014] is used.

3.2.1 Stock returns and noise

When dealing with stock prices series, the amount of noise makes it impossible to distinguish whether trading is done on the true information or just noise [Black, 1986].

This problem can be tackled, although to only some extent, by using denoising methods. Thus, another architecture I use in this work is a Denoising Autoencoder (DAE). Autoencoders are the family of models that given some input x try to learn a function $f(x) = x$ under some constrain. This type of network is usually viewed as made of two functions:

$$h = f(x) \tag{8}$$

encoder, that tries to learn some representation, h of the data x and

$$r = g(h) \tag{9}$$

[1]Although it is not guaranteed that the local minimum found will coincide with the global minimum.

decoder, which tries to reconstruct the original data from h. The loss function is then mean squared error between x and r.

In more general, the family of autoencoders can be represented as as some function that minimizes:

$$L = (x, g(f(x))) \tag{10}$$

Usually, autoencoders are constructed in a way that the job of the encoder becomes mapping input into lower dimensional space. This is achieved by connecting a few layers, shrinking in size. The smallest one is called a “bottleneck” layer. Then, this layer is followed by set of increasing layers with the final one being the same size as the input layer. A comparison can be made between autoencoders and PCA - the former is constrained by the lower amount of dimensions that the data has to be represented and then reconstructed while the PCA finds the directions maximizing the explained variance while being constrained to keep the components orthogonal.

The Denoising Autoenoder is an autoencoder that receives artificially corrupted data. For example, one can add isotropic Gaussian noise (N($0,\sigma * I_p$), where σ is some small number varying each batch) to the first hidden layer. Such autoencoder learns to be resistant to small perturbations of the input - noise. I’m only interested in the encoder part that produces latent variables - therefore, after training the autoencoder, the decoding part is rejected. Further, I ensure that the latent variables are not only good at mapping the input into lower dimensions but also develop some properties useful in the classification task. To achieve that, the bottleneck layer is connected to another dense unit followed by the output layer. By imposing such architecture, there are two loss functions - Binary Cross-Entropy (see equation 7) and the reconstruction error (Mean Squared Error).

3.3 Fair-play enforcement architectures of $model_{transfer}$, $model_T$ and $model_N$ and the training process.

To compare the outcome of the two models, there is a need to make sure that they are both trained on the same set of rules and the difference is due to adding the sentiment features to the $model_N$. On top of that, given that I am in possession of the data for the past prices starting from the 2000 while the comment data taken into consideration in this analysis starts from 2016, instead of rejecting 16 years of observations, I deploy a technique called transfer learning. In a nutshell, transfer learning is a process of estimating one model then using it for some very similar tasks after short fine tuning. This technique is a backbone of training virtually all natural processing language models, including the BERT.

Therefore, the training process of all three models takes the following form:

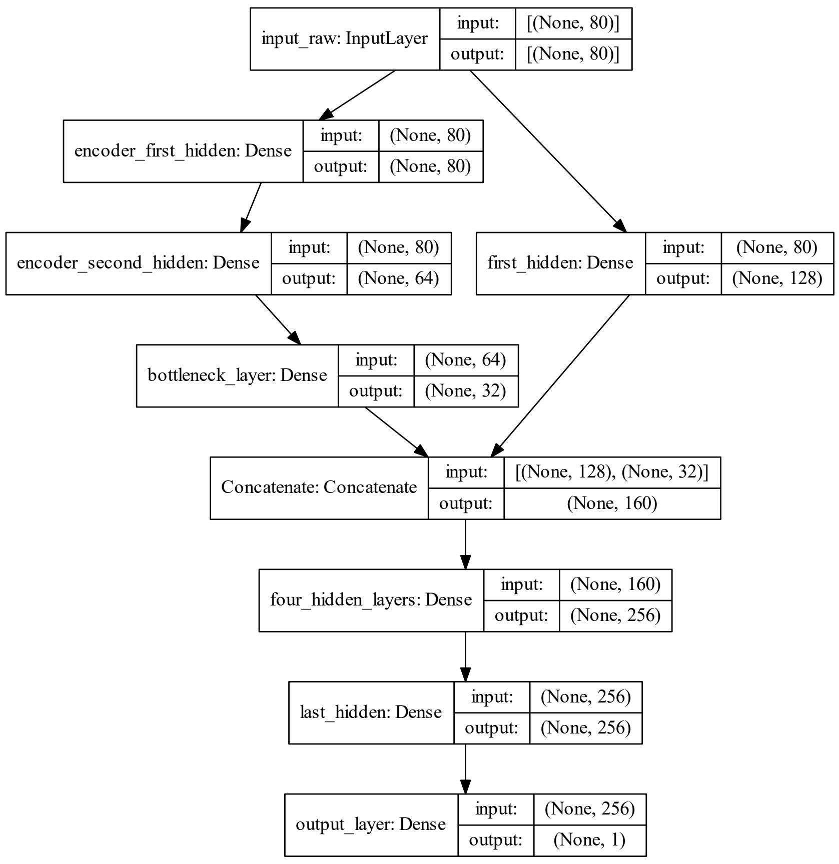
- First, I train $model_{transfer}$, using past prices data for years 2000-2016. This model combines the encoder and feed forward layers (see **Figure 3** for simplified view). Encoder is trained before the rest of the network and his weights are then frozen. The task the $model_{transfer}$ is trained on, is prediction of the sign of the stock return in 5 days.
- Second, I use data between 2016 and July 2020 to train $model_T$. This model takes as an input 80 features derived from stock prices and the is combination of the feed forward layers and the $model_{transfer}$, specifically the output of the last hidden layer - vector of 256 features

associated with each of observations. During the training of $model_T$, $model_{transfer}$'s weights are also updated. This model is also trained to predict the sign of the stock return in 5 days.

- Lastly, using the sentiment data from 2016 to July 2020 I train $model_N$. This model accepts 160 features as an input and is a combination of feed forward layers and the $model_T$, specifically the output of the last hidden layer - vector of 384 features associated with each of observations. As for the 2 models above, $model_N$ predicts the probability of the stock return being positive or negative in 5 days. To avoid assets that are not widely discussed, I require that the ticker has to be mentioned more than 25 times during a day.

Figure 4 shows stylized flowchart of how $model_{transfer}$, $model_T$ and $model_N$ are related to each other. **Figure 5** and **Figure 6** in appendix C present architectures of $model_T$ and $model_N$, respectively.

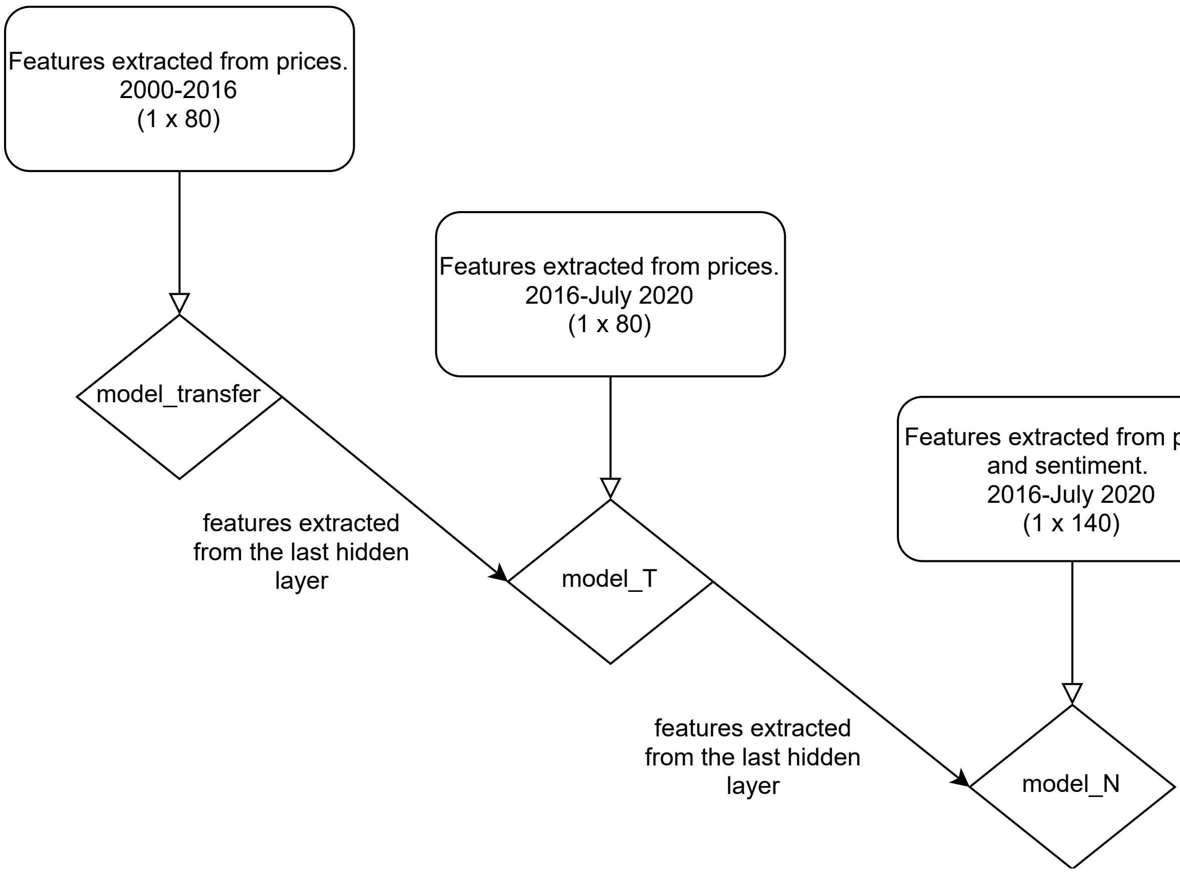
Figure 3: The architecture of $model_{transfer}$.


input_raw: InputLayer | input: [(None, 80)] | output: [(None, 80)]
encoder_first_hidden: Dense | input: (None, 80) | output: (None, 80)
encoder_second_hidden: Dense | input: (None, 80) | output: (None, 64)
first_hidden: Dense | input: (None, 80) | output: (None, 128)
bottleneck_layer: Dense | input: (None, 64) | output: (None, 32)
Concatenate: Concatenate | input: [(None, 128), (None, 32)] | output: (None, 160)
four_hidden_layers: Dense | input: (None, 160) | output: (None, 256)
last_hidden: Dense | input: (None, 256) | output: (None, 256)
output_layer: Dense | input: (None, 256) | output: (None, 1)

The left side of the model - layers named encoder first layer, second hidden and bottleneck layer are encoder part of the DAE. They receive the raw input in for of 80 features and encode it into 32 dimensional space. Those latent variables are then concatenated with the output of the first hidden layer of the FNN and fed further into the network. The Dropout layers and Activation functions are not shown, four hidden layers are represented by a single block: ”four hidden layers”.

The above described process is a result of making sure that neither of models has architecture advantage over the another. That is why the $model_{transfer}$ is a crucial element of the entire process. From the point of view of $model_T$, the $model_{transfer}$ plays exactly the same role as the

Figure 4: Flowchart of three models


Features extracted from prices.
2000-2016
(1 x 80)
model_transfer
features extracted from the last hidden layer
Features extracted from prices.
2016-July 2020
(1 x 80)
model_T
Features extracted from prices and sentiment.
2016-July 2020
(1 x 140)
features extracted from the last hidden layer
model_N

$model_T$ from the point of view of the $model_N$.

While training $model_{transfer}$, $model_N$ and $model_T$, to prevent overfitting, I have used a Combinatorial Purged Cross-Validation [Lopez de Prado, 2018]. The CPCV method splits the data into not overlapping, respecting the time order, train-validation sets. Between all hidden units of Feed Forward Network there is a Dropout layer. The activation function for each hidden dense unit is *mish* function [Misra, 2019]:

$$f(x) = x * (exp^{ln(1+exp^x)} - exp^{-ln(1+exp^x)})/(exp^{ln(1+exp^x)} + exp^{-ln(1+exp^x)}) \tag{11}$$

Encoder model uses *relu* activation functions - $max(0, x)$ and no Dropout layers. All models were trained using tensorflow [Martın Abadi et al., 2015] framework and the hyperparameters were found using kerastuner's Bayesian Optimisation [O'Malley et al., 2019]. Based on validation AUC during training, three best performing models of each type are chosen and then trained until the EarlyStoping triggers. During the training, I have used undersampling to maintain the balanced response.

By keeping the process of training of $model_N$, and $model_T$ to be the same - constrained by the size of the dense units due to different size of the inputs, I enforce that the output of both models is different only because of stochastic nature of the neural networks and the additional input that the model N has access to.

4 Evaluation

Both $model_T$ (model using only past prices features), and $model_N$ (model that also makes use of sentiment), give the same type of output - a vector of probabilities of a stock return in 5 days being positive. This allows us to directly compare models performance. One way to do it, is to look on the accuracy.

Accuracy comparison implies a usage of some arbitrarily given threshold. For example, given a vector of probabilities, I can assign that all predictions with probability higher than 0.5 means that the return on the stock in 5 days is positive. By changing this threshold, it is possible to ignore the stocks - and trading opportunities - that the model associates with lower probability of the return being positive[2]. By rejecting some trades, the coverage ratio, which is defined as "a number of the trading opportunities that the model would take given some threshold divided by total possible trades" will change. Let's illustrate it with a simple example. Consider the histogram (see **Figure 7** in Appendix D) of 9.7 million probabilities produced by $model_{transfer}$ for period between 2000 and 2016.

When the threshold is set to 0 - coverage rate of 100% - the accuracy of the model is just 54%. By setting the threshold such that it rejects all trades with probability being between 0.4 and 0.6, the coverage rate drops to just 6.778%. When the coverage rate is just 0.56% - that is, on average, model makes only 20[3] trades each day - the accuracy rises to 85%. This inverse relationship between coverage and accuracy is summed up below.

Hypothesis 1: Trade-off between coverage ratio and accuracy of a classifier. 1 *Let's assume that the output of a classifier is ∼N(x,σ). Furthermore, let's assume that the probability of correctly classifying each observation is a function increasing in the absolute distance from the mean. Then, by rejecting the observations closer to the mean, the accuracy of the classifier has to rise at the cost of lower coverage rate.*

In period between the 1st of July 2020 and 27th of April 2021, there was 688 unique stocks mentioned more than 25 times on WallStreetBets - giving 8322 trading opportunities (for scatter plot of those opportunities see **Figure 6** in Appendix C).**Table 2** shows the accuracy and average performance[4] for different levels of coverage. Due to the stochastic nature of Neural Networks, I present outputs of 3 models of both types ($model_T$ and $model_N$) chosen based on performance during the training period (based on validation AUC during the training period). For each model (names of columns), each row gives two numbers: accuracy and average performance at certain coverage ratio (first column). The "—" indicates that due to the distribution of outcomes, relevant coverage ratio can not be obtained by changing threshold.

The **Table 2** presents some interesting results. While the family of $model_T$ has a visible trade-off between coverage ratio and accuracy, this does not only doesn't hold for first $model_N$, but for third $model_N$, lower coverage is accompanied by lower accuracy. Yet, for 10%, 5% and 1% coverage, models leveraging the sentiment perform surprisingly well when looking at their average performance. However, those gains are mostly driven by positions in reddits favourites. To investigate

[2]The same thing goes the other way, if a probability of a stock being up in 5 days is very low, the trade taken can be shorting it and claiming the inverse of the stock return.

[3]0.0086*9700000/252/16

[4]Average performance is gross hypothetical gain that the model would achieve if it was to either go long 1 unit whenever it believes that the return in 5 days is positive, or go short 1 unit otherwise.

Table 2: Performance of 2 types of models at different levels of coverage.

coverage	model_T^0	model_N^0	model_T^1	model_N^1	model_T^2	model_N^2
100%	0.543, 1.0137	0.505, 1.0090	0.542, 1.0130	0.509, 1.0086	0.542, 1.0134	0.490, 1.0028
75%	0.552, 1.0167	0.511, 1.0106	0.555, 1.0161	0.510, 1.0102	0.553, 1.0145	0.489, 1.0034
50%	0.564, 1.0131	0.511, 1.0124	0.565, 1.0180	0.511, 1.0122	0.569, 1.0149	0.477, 1.0015
25%	0.599, 1.0182	0.500, 1.0165	0.607, 1.0199	0.515, 1.0192	0.605, 1.0184	0.473, 1.0016
10%	0.656, 1.0244	0.509, 1.0421	0.685, 1.0279	0.514, 1.0378	0.643, 1.0230	0.495, 1.0088
5%	0.697, 1.0303	0.524, 1.0553*	0.750, 1.0344	0.528, 1.0717	0.733, 1.0312	0.463, 1.0125
1%	0.869, 1.0607	—	0.830, 1.0802	0.591, 1.2562	0.797, 1.0706	0.451, 1.1020

During that period, average 5 days net return was 1.58% and it's standard deviation was 0.179 .* indicates the performance at 7% coverage.

this phenomena in detail, **Table 3** focuses only on "meme stocks"[5] (2228 trading opportunities). **Table 4** takes into account 25 most popular stocks[6] on WSB, after excluding "meme stocks" (3027 trading opportunities).

Table 3: Performance of 2 types of models on "meme stocks".

coverage	model_T^0	model_N^0	model_T^1	model_N^1	model_T^2	model_N^2
100%	0.553, 1.0267	0.507, 1.0221	0.530, 1.0205	0.498, 1.0276	0.537, 1.0242	0.460, 1.0011
75%	0.567, 1.0335	0.500, 1.0221	0.545, 1.0275	0.500, 1.0293	0.550, 1.0294	0.455, 1.0012
50%	0.577, 1.0462	0.500, 1.0318	0.568, 1.0427	0.505, 1.0412	0.562, 1.0309	0.439, 0.9959
25%	0.5763, 1.029	0.471, 1.0521	0.591, 1.0456	0.515, 1.0584	0.585, 1.0350	0.436, 1.0000
10%	0.610, 1.0352	0.477, 1.1194	0.596, 1.0381	0.529, 1.1279	0.596, 1.0343	0.453, 1.0377
5%	0.694, 1.0521	—	0.696, 1.055	0.569, 1.2497	0.670, 1.0484	0.482, 1.0783
1%	0.909, 1.1101	—	0.909, 1.154	0.595, 1.2911*	0.810, 1.1261	0.472, 1.1125**

During that period, average 5 days net return was 4.2% and it's standard deviation was 0.315.* and ** indicate the performance at 1.9% and 3.7% coverage, respectively.

Comparison of the average performance and accuracy between $model_T$s and $model_N$s shows that the sentiment can convey information useful in predicting future returns. However, this information only improves the forecasting abilities when considering "meme stocks". In case of 25 most popular stocks (measured by the average number of comments, after the exclusion of "meme stocks") the sentiment highly degrades the $model_N$. While $model_T$ is able to approach the benchmark return (average 5day return of 2.71%) for coverage rates of 5% and beat it at 1%, this is not the case for model using sentiment.

Another way to compare the performance of those two models is to investigate how they would

[5]GME, AMC, TSLA, PLTR, NIO, ICLN, PLTR, PTON, SLV, PLUG, FSLY, SNAP, ARKK, KODK, MU, BYND, ROKU, MVIS, RKT, CRSP, GLD, DKNG, BBBY, TWTR, MU."Meme stock" is an asset that has:

- experienced rise of trading volume among the retail investors and increase in popularity on WallStreetBets - as proxied by comments referring to it,
- highly volatile,
- usually unprofitable.

Such stocks are frequently mentioned on WSB - even when there is no new news regarding the asset. The exact definition of a "meme stock" is hard to formulate and creating a test that would allow to classify assets to this group is probably impossible. Therefore, the decision how to create the basket of those assets is slightly arbitrary and based on my previous experience as an user of the forum.

[6]SPY, AAPL, AMD, AMZN, MSFT, BABA, QQQ, FB, NVDA, BA, WMT, SQ, INTC, GM, ATVI, DIS, NFLX, PFE, ARKG, TSM, VXX,CRSR, PRPL, NET, CRM.

Table 4: Performance of 2 types of models on popular stocks excluding "meme stocks".

coverage	model^0_T	model^0_N	model^1_T	model^1_N	model^2_T	model^2_N
100%	0.554, 1.0071	0.509, 1.0029	0.544, 1.0073	0.510, 1.0016	0.552, 1.0071	0.514, 1.0036
75%	0.547, 1.0080	0.514, 1.0033	0.550, 1.0080	0.511, 1.0018	0.555, 1.0088	0.515, 1.0043
50%	0.565, 1.0098	0.511, 1.0026	0.571 1.0103	0.503, 1.0000	0.574, 1.0110	0.523, 1.0062
25%	0.592, 1.0133	0.511, 1.0000	0.605, 1.0142	0.509, 1.0005	0.585, 1.0120	0.530, 1.0076
10%	0.647, 1.0189	0.544, 1.0040*	0.661, 1.0200	0.532, 1.0065	0.655, 1.0187	0.541, 1.0101
5%	0.671, 1.0216	—	0.768, 1.0280	0.583 1.0145	0.744, 1.0241	0.487, 1.0025
1%	0.870, 1.0551	—	0.813, 1.0513	0.585, 1.0101	0.871, 1.0543	0.441, 0.9950

During that period, average 5 days return was 2.71% and it's standard deviation was 0.236.* and ** indicate the performance at 9% and 3.7% coverage, respectively.

perform in some rare environment. An example of it is the sell-off took place in the first quarter of 2020. During that time, and more specifically, between late February and the middle of March, SP500 has dropped from almost 3400$ to just above 2100$. This move was accompanied with multiple other assets experiencing extreamly sharp collapse in value and even price of WTI Oil trading for negative 37$.

To investigate the performance of models in this exact scenario, the training data does not include the period from 15th of the February 2020 to the 1st of July 2020. This ensures that the strategies developed by $model_T$ and $model_N$ are ignorant of the events of 2020. Results of this comparison are presented in **Table 5**.

Table 5: Performance of 2 types of models between 15th of February and 1st of April 2020.

coverage	model^0_T	model^0_N	model^1_T	model^1_N	model^2_T	model^2_N
100%	below 0.5 and 1	0.560, 1.0532	below 0.5 and 1	0.595, 1.0589	below 0.5 and 1	0.503, 1.0223
75%	below 0.5 and 1	0.574, 1.0605	below 0.5 and 1	0.613, 1.0661	below 0.5 and 1	0.505, 1.0273
50%	below 0.5 and 1	0.580, 1.0558	below 0.5 and 1	0.630, 1.0721	below 0.5 and 1	0.497, 1.0349
25%	below 0.5 and 1	0.553, 1.0575	0.581, 1.0254	0.679, 1.0934	below 0.5 and 1	0.488, 1.0354
10%	below 0.5 and 1	0.596, 1.0600	0.587, 1.0434	0.668, 1.1040	below 0.5 and 1	0.494, 1.0744

During that period, average 5 days return was -3.85% and it's standard deviation was 0.22.

Table 5 show that the $model_N$s outperforms greatly $model_T$s during the sell-off of 2020. It appears that the usage of sentiment improved the performance during this Black-Swan event. This outcome is not dictated by the sentiment leveraging models being "short" all the time during that period. Actually, at the 25% coverage rate, the long to short ratios for $model_N$s were 0.73, 0.27 and 0.79.

5 Conclusions

In this work I have analysed the herd behaviour of users of online investing forum called WallStreetBets. By extracting sentiment from comments and posts, I investigated whether they have any predictive power that can be used to improve models trying to forecast sign of the 5 days return of stocks. This is achieved by:

- First, extracting the data from 2016 to May 2021 and labeling them into one of three categories: "bearish", "neutral" or "bullish",
- Second, training two very similar Neural Networks - one having access to only past prices

($model_T$) and one that besides past prices, uses features extracted from classified comments ($model_N$),

- Lastly, comparing the outputs of the models.

I find that purely quantitative approach focusing on the numbers of posts, comments and their sentiment, can extract information improving the performance of $model_N$. This shows that there is at least some sense in the herd behaviour of WallStreetBets users. This improvement during the period between July 2020 and May 2021 is mainly driven by correctly predicting the returns on "meme stocks". This manages to explain the contradicting results of previous studies. On average, the sentiment is nothing but noise, but when it is used to forecast returns of Reddit's favourites, it yields useful information.

Additional striking difference can be also found during the latest Minsky moment. I found that between February 15th and April 2020, sentiment leveraging model is outperforming its peer greatly.

For the future research, I suggest to take a slightly different approach to the extraction of sentiment. Submissions that are highly voted or receive multiple awards should be given higher weight than comment that nobody has read. This approach may reduce the noise significantly. On top of that, one could also deploy a different algorithm to find relevant comments. My approach has discarded over 90% of data - possibly some of it could have improved the actual performance of $model_N$. For example, given that it is possible to extract the *parentID* from reddit comments, one could treat answers to a submission or a comment as related and possibly conveying sentiment related to the previously mentioned asset. However, this may call for Language Model of higher complexity, trained with the different task in mind, than an adapted and fine-tuned BERT.

Another possible approach that can be undertaken is to not craft a set of features that represent the evolution of past prices and sentiment but use sequence-to-output modeling or Reinforcement Learning.

Since the WallStreetBets is all about leverage through the use of options, it is also possible to not focus on stock returns, but the options chains of assets frequently discussed within the forum and how they behave following increased discussion around a stock.

References

Aharon, David Y. et al. (2021). *Did David Win a Battle or the War Against Goliath? Dynamic Return and Volatility Connectedness between the GameStop Stock and the High Short Interest Indices.* DOI: dx.doi.org/10.2139/ssrn.3788155.

Arias, Marta, Argimiro Arratia, and Ramon Xuriguera (Jan. 2014). "Forecasting with Twitter Data". In: *ACM Trans. Intell. Syst. Technol.* 5.1. ISSN: 2157-6904. DOI: 10.1145/2542182.2542190. URL: https://doi.org/10.1145/2542182.2542190.

Baumgartner, Jason et al. (2020). *The Pushshift Reddit Dataset.* arXiv: 2001.08435 [cs.SI].

Black, Fischer (1986). "Noise". In: *The Journal of Finance* 41.3, pp. 528–543. DOI: https://doi.org/10.1111/j.1540-6261.1986.tb04513.x.

Bradley, Daniel et al. (2021). *Place Your Bets? The Market Consequences of Investment Advice on Reddit's Wallstreetbets.* DOI: http://dx.doi.org/10.2139/ssrn.3806065.

Devlin, Jacob et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* arXiv: 1810.04805 [cs.CL].

Eaton, Gregory W. et al. (2021). *Zero-Commission Individual Investors, High Frequency Traders, and Stock Market Quality.* DOI: http://dx.doi.org/10.2139/ssrn.3776874.

Fisher, Kenneth L. and Meir Statman (2000). "Investor Sentiment and Stock Returns". In: *Financial Analysts Journal* 56.2, pp. 16–23. ISSN: 0015198X. URL: http://www.jstor.org/stable/4480229.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization.* arXiv: 1412.6980 [cs.LG].

Long, Cheng, Brian M. Lucey, and Larisa Yarovaya (2021). *I Just Like the Stock' versus 'Fear and Loathing on Main Street' : The Role of Reddit Sentiment in the GameStop Short Squeeze.* DOI: http://dx.doi.org/10.2139/ssrn.3822315.

Lopez de Prado, M. (2018). *Advances in financial machine learning.* John Wiley Sons.

Martın Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* URL: https://www.tensorflow.org/.

Misra, Diganta (2019). *Mish: A Self Regularized Non-Monotonic Activation Function.* arXiv: 1908.08681 [cs.LG].

O'Malley, Tom et al. (2019). *Keras Tuner.* https://github.com/keras-team/keras-tuner.

Padial, Darío López (2018). *Technical Analysis Library in Python.*

Schmeling, Maik (2009). "Investor sentiment and stock returns: Some international evidence". In: *Journal of Empirical Finance* 16.3, pp. 394–408. ISSN: 0927-5398. DOI: https://doi.org/10.1016/j.jempfin.2009.01.002. URL: https://www.sciencedirect.com/science/article/pii/S0927539809000048.

Appendices

Appendix A:

The list of features derived from financial time series. Default values of indicator's hyper parameters were used, indicator written in italics means that it's value had to be adjusted using the process described before.

- VIX and VIX SMA(10days),
- Chaikin Money Flow,
- Money Flow Index ,
- Ease Of Movement (EMV) and SMA EMV,
- Average True Range,
- Bollinger Channel Percentage Band (BCP),
- two dummy variables whether price is outside the BBP (and on which side),
- Keltner Channel Percentage Band (KCP),
- two dummy variables whether price is outside the KCP (and on which side),
- Donchian Channel Percentage Band (DCP),
- two dummy variables whether price is outside the DCP (and on which side),
- Ulcer Index,
- Positive and Negative Directional Movement Index,
- Vortex Indicator,
- Trix,
- Mass Index,
- Commodity Channel Index,
- KST oscillator,
- Aroon Indicator,
- Parabolic Stop and Reverse,
- Schaff Trend Cycle,
- RSI and Stochastic RSI,
- True Strength Index,
- Ultimate Oscillator,
- Stochastic Oscillator,
- Stochastic Oscillator Signal,
- Williams %R,
- Rate of Change,
- Percentage Price Oscillator,
- Percentage Price Oscillator Signal Line,
- Percentage Price Oscillator Histogram,
- Daily Return,
- Daily Log Return,
- Daily High to Daily Close,
- Daily Low to Daily Close,
- Daily Open to Daily Close.

And adjusted indicators (following the process described before):

- Volume Weighted Average Price,
- Bollinger Bands (Mean, Low, High, Width),
- Keltner Channel (Mean, Low, High, Width),
- Donchian Channel (Mean, Low, High, Width),
- SMA, FMA, and SMA(200 days)
- SEMA, FEMA,
- Ichimoku Kinkō Hyō,
- MACD,
- Kaufman's Adaptive Moving Average,
- Detrended Price Oscillator.

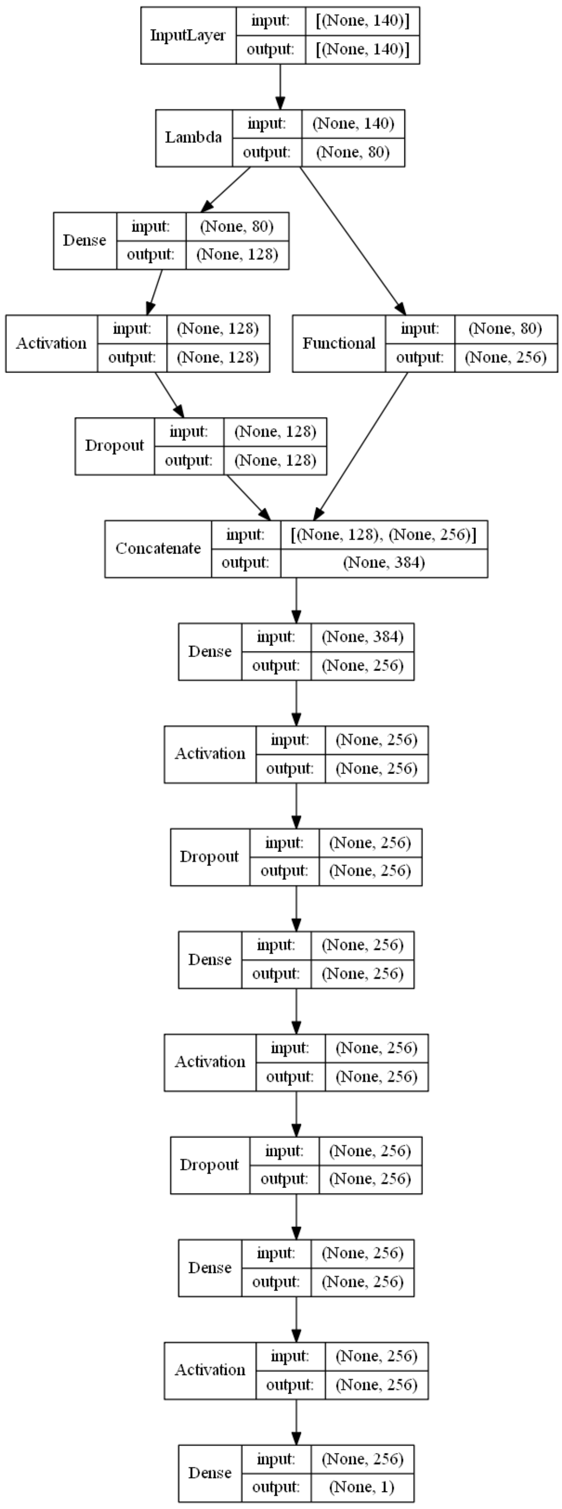
Appendix B:

A short explanation of commonly used terms in the DeepLearning literature:

- forward pass - a process of feeding the data into the model and calculating the total loss of an algorithm. Equivalent to calculating MSE for a Linear Regression model.
- backpropagation - a process of updating the weights of the Neural Network. Equivalent of adjusting the coefficients of Linear Regression model using gradient descent.
- epoch - a single pass of the entire training data through the algorithm.
- batch - when a training set is large, the data is usually split into smaller samples called batches.
- gradient accumulation - process of postponing the backpropagation until a certain number of forward passes happens. Used to deal with memory constrains.
- Dropout layer - layer that during the training of Neural Networks randomly, with some probability, sets the input to 0. This practice prevents the network from just remembering the training examples and improves generalization.
- Dense layer - tensor of predetermined size that acts as an input to the next layer.

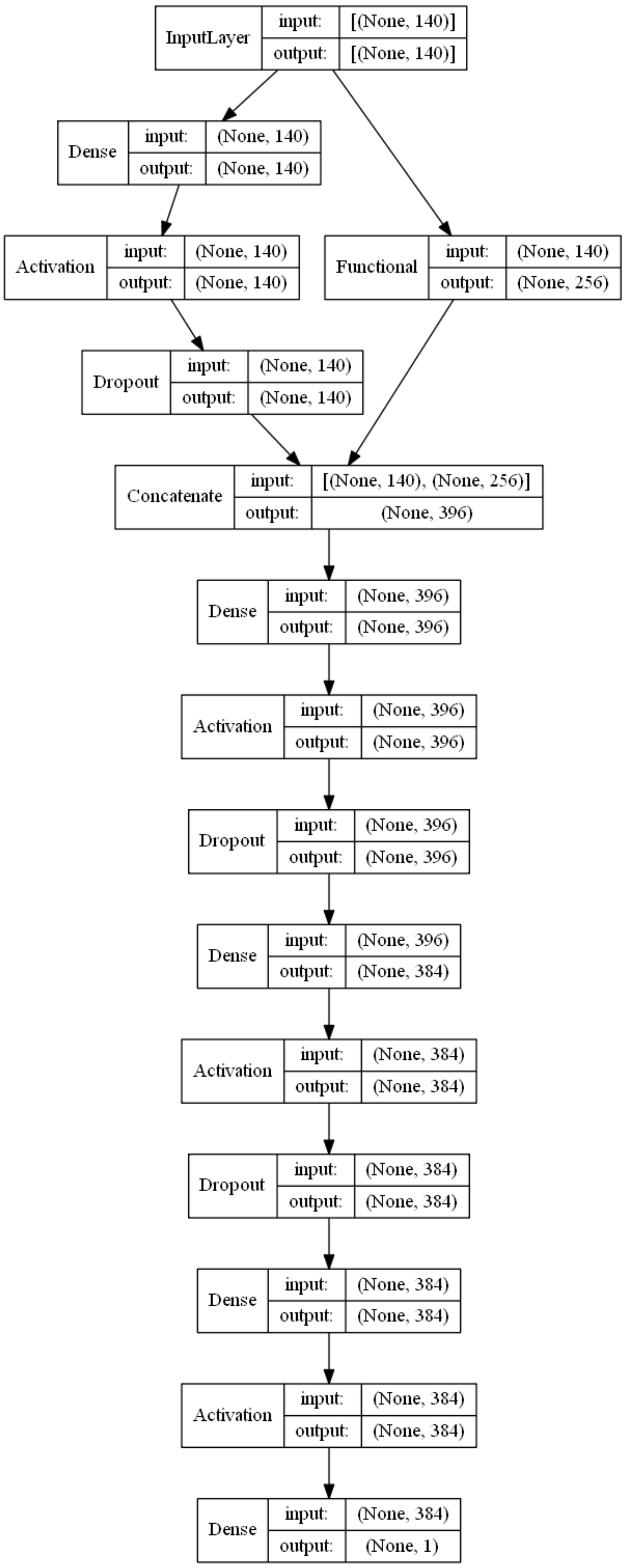
Appendix C:

Figure 5: Architecture of $model_T$.



$model_T$ is constructed from set of feed forward layers and $model_{transfer}$ (represented by block named Functional).

Figure 6: Architecture of $model_N$.

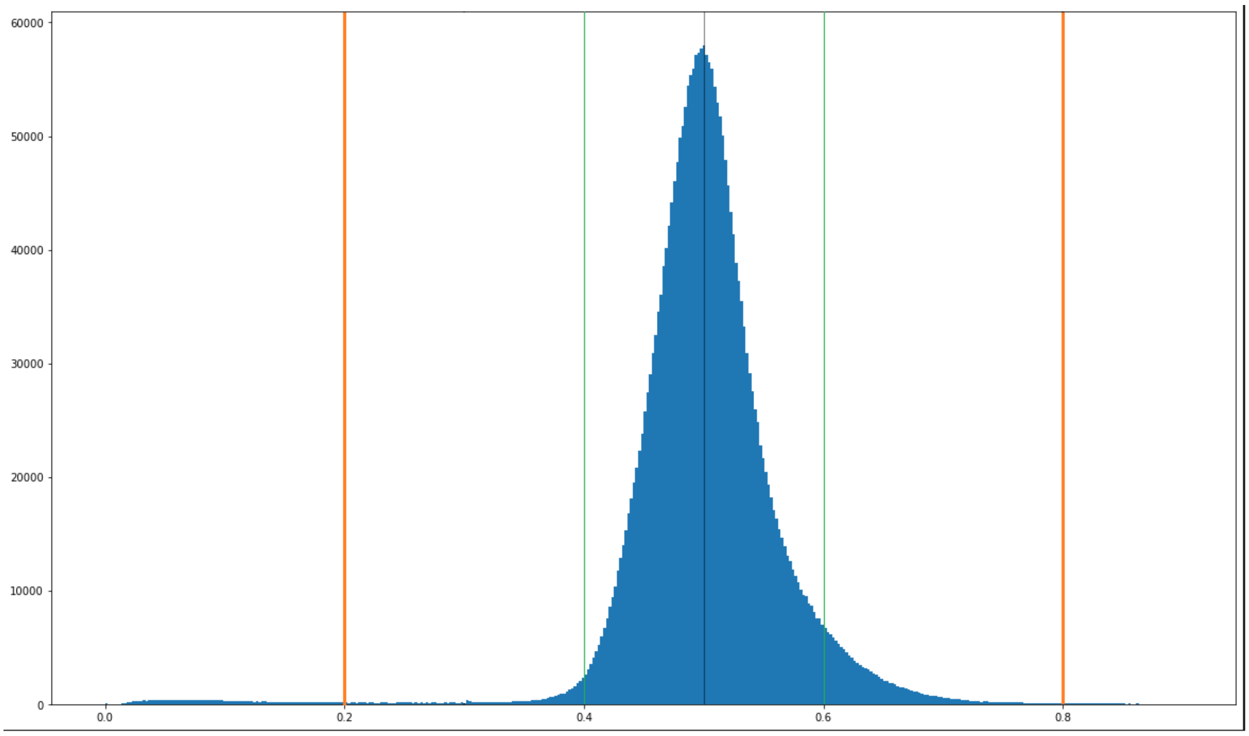


InputLayer | input: [(None, 140)] | output: [(None, 140)]
Dense | input: (None, 140) | output: (None, 140)
Activation | input: (None, 140) | output: (None, 140)
Functional | input: (None, 140) | output: (None, 256)
Dropout | input: (None, 140) | output: (None, 140)
Concatenate | input: [(None, 140), (None, 256)] | output: (None, 396)
Dense | input: (None, 396) | output: (None, 396)
Activation | input: (None, 396) | output: (None, 396)
Dropout | input: (None, 396) | output: (None, 396)
Dense | input: (None, 396) | output: (None, 384)
Activation | input: (None, 384) | output: (None, 384)
Dropout | input: (None, 384) | output: (None, 384)
Dense | input: (None, 384) | output: (None, 384)
Activation | input: (None, 384) | output: (None, 384)
Dense | input: (None, 384) | output: (None, 1)

$model_N$ is constructed from set of feed forward layers and $model_T$ (represented by block named Functional).

Appendix D:

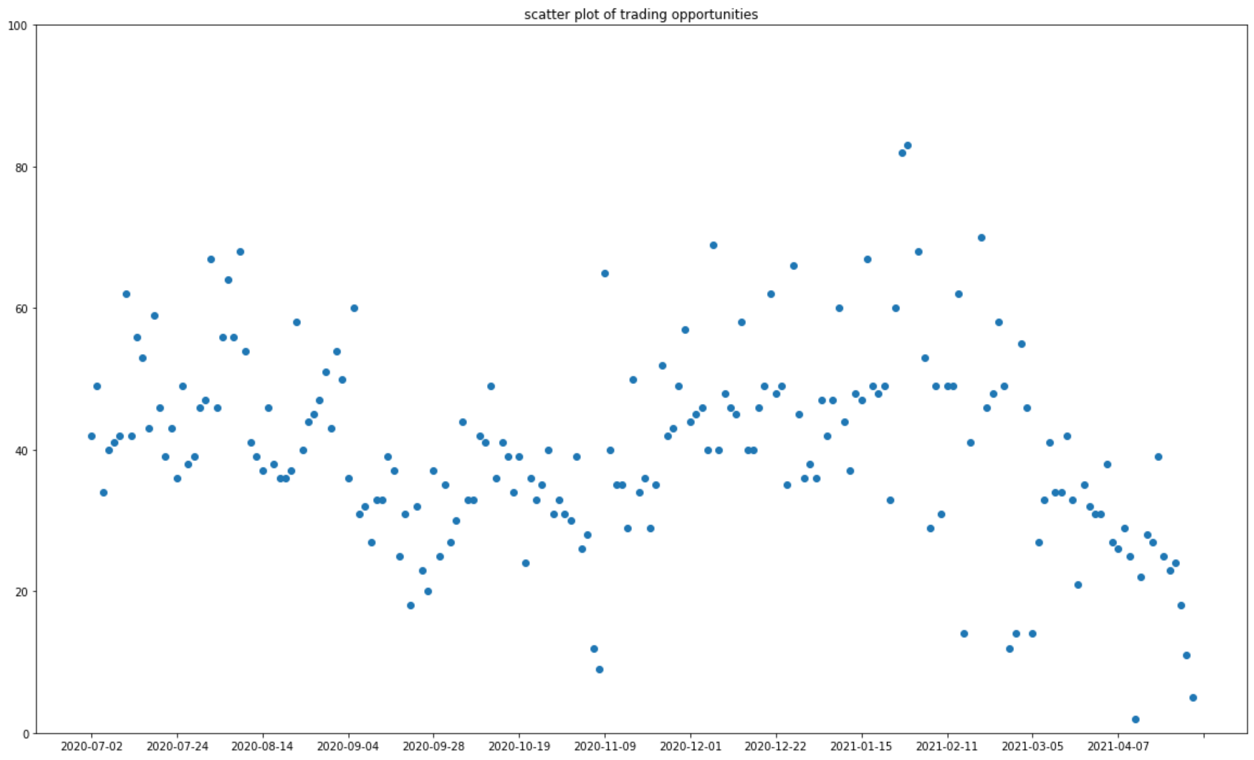
Figure 7: Histogram of probabilities of a stock return being positive in 5 trading days.



This particular pdf of outcomes looks similar to a normal distribution. In practice there is no guarantee it will always happen. However, in this case, it is easier to illustrate the example using this particular output from one of the models. The vertical green and orange lines, indicate the two thresholds - 0.1 and 0.3, respectively. When the threshold of 0.1 is used, all trading opportunities with probability that lies between 0.4 and 0.6 are rejected. As a result, the model has the coverage ratio of 6.778% and the accuracy of 67%. When 0.3 is considered as a threshold, that is, stocks with probability of giving the positive return in 5 days are lower than 0.2 are shorted. At the same time, stocks with probability higher than 0.8 are bought. This results in the coverage ratio of 0.86% and accuracy of 85%.

Appendix E:

Figure 8: Scatter plot of trading opportunities (more than 25 mentions between July 2020 and 27th April 2021)



Axis y represents the number of stocks that have been mentioned more than 25 times within one day.