AI Stonkarithm

Final Project Report

By Josh Fish & Anthony Elkommos Youssef

INTRODUCTION

In recent times, the significance of social media could be described as an aspect that has grown at an exponential rate. Today, it certainly has a crucial role in connecting people and providing a platform for discussions and communication with massive audiences. In fact, social media has become so influential that it played a significant role in triggering the Arab Spring, influencing large numbers of people to join the action, and facilitating the communication among the participants of the political protests.

Today, social media has a direct effect on the stock market and has the ability to cause huge swings on the price of stocks. From time to time, we have seen how popular figures such as Elon Musk and Donald Trump have caused huge changes in stock prices with just a tweet. De facto, the modern investor and the day trader have had to become active on Twitter and turn on the tweet notifications for important and popular Twitter accounts in order to be able to keep up with the latest news.

More recently, struggling and slowly going out of business, Gamestop was not able to keep up with the competition from digital distribution services and was greatly affected by COVID-19. As Reddit users began realizing that the stock was strongly shorted by major hedge funds, they began investing their money into the company and sharing posts about the rise of the stock and the decline of the hedge funds and short sellers. This led to a short squeeze that resulted in the rise of the stock price, which at its height reached nearly 30 times its $17.25 valuation at the beginning of that month.

The purpose of this paper is to implement a machine learning algorithm, monitoring and performing a sentiment analysis on posts in social media websites in order to more accurately predict the direction of stock prices. However building a model that predicts the price of stock is

no easy task, many challenges need to be overcome in order to come produce an algorithm that could be considered good enough to replace a human investor. Such challenges include collecting the posts which was difficult due to restrictions on the APIs provided by these social media websites, performing the sentiment analysis which is very difficult task due to many reasons such as the fact that users could be talking about a rising stock and a declining stock within the same post, among other problems due to the machine learning aspect of the project.

RELATED WORK

After the rise of social media platforms, Bloomberg decided in 2014 to implement a sentiment analysis tool on Twitter to which they gave their Bloomberg Portal users access. The tool allowed "[u]sers can set up an alert to watch, for example, all of the tickers in the S&P 500 and be alerted to those with significantly positive or negative social sentiment" (Bloomberg, 2014).

Lubitz, M. (2017) discussed a sentiment analysis tool which was performed on articles which were shared on Reddit communities. Supervised machine learning techniques were used in order to collect the desired posts. The predicted sentiment analysis was, then, compared to the corresponding closing value of the S&P 500 stock index. The number of votes and comments on such posts were also taken into consideration and included in the calculations. "News published in the Financial Times [was used] as a baseline" (Lubitz, 2017). At best, an accuracy of 56.49% in predicting the direction of the stock prices was achieved. The author concluded that the predictive power of Reddit was not much more significant than the standard newspaper analysis and that the results of this research "corresponds to the results of recent academic research" (Lubitz, 2017).

Rao, T. and Srivastava, S. (2012) investigated the relationship between posts on Twitter

and the stock market data. Sentiment analysis was performed on about 4 million tweets which were all posted on Twitter between the dates of June 2010 and July 2011. The authors found a high correlation of up to 0.88 for returns between the sentiment of tweets and stock prices. Granger's Causality Analysis was used to validate the movement of stock prices, and an Expert Model Mining System was utilized in order "[t]o show the performance of tweet features in [their] prediction model" (Rao & Srivastava, 2012). The authors concluded that the approach used in their paper is far more superior than any previously discussed method due to its high directional accuracy of 91%.

Ren, R., Wu, D., and Liu, T. (2019) developed a method through which stock market movement was directionally predicted through the use of financial market data as well as sentiment analysis which had been incorporated with investor psychology. A web crawler was built in order to download daily news documents with which sentiment indexes were constructed. The authors found that through their predictive models, "the accuracy of forecasting the movement direction of the SSE 50 Index can be as high as 89.93% with a rise of 18.6% after introducing sentiment variables" (Ren et al., 2019).

APPROACH

In order to collect the necessary data required for the machine learning algorithm which was used in this paper, stock data was easily collected from the Yahoo Finance API. However, collecting the data from Twitter and Reddit was not as simple of a task. A Reddit API named PRAW was first tested, however, the API did not allow for searching through posts as desired due to a limit of 25 posts per search which was enforced by the API. Multiple different APIs have then been tested, however, only the Pushshift Reddit API allowed for the desired Reddit search capabilities. Through this API, a JSON of the up to 100 most upvoted posts on any given

day was produced. The post titles were, then, easily extracted from the JSON.

In order to collect the tweets, the Twitter API was first used. However, it was highly limited as it limited both the number of posts to be collected through a search and it also limited the number of posts that were accessible through the API to only posts which have been published within the previous seven days. Given this knowledge we attempted to create a predictive model that used this data from 7 days and stock price information per minute over the same 7 days, however, as one may have expected the results were not ideal. Our predictions were not very close to the actual price and in general 7 days data was simply not enough for a training and testing set.

The Twint API was then tested. However, due to recent Twitter changes, the API returned errors every time a search was performed. Then, another version of the Twint API, which was developed in order to resolve the errors returned by the original Twint API, was installed using the following command:

```
pip3 install --user --upgrade git+https://github.com/twintproject/twint.git@origin/master#egg=twint
```

Now that we had our tweet data we had to convert this into a decimal value that could represent the sentiment score on a given date. We could then use this numerical value as a parameter to our neural network so that it took this score into account when predicting the price. To do this we used another third party library called NLTK and it's "vader_lexicon" library which given a list of text will derive a compound score of the sentiment. When using this however we noticed certain key stock terms and social media "meme" terms were not included in the lexicon so we then added these terms with a weight on the sentiment they would carry. Below are all the terms we added and their sentiment score. For reference the scores in the lexicon all ranged from -3 to 3 so we made these values slightly higher to give them more

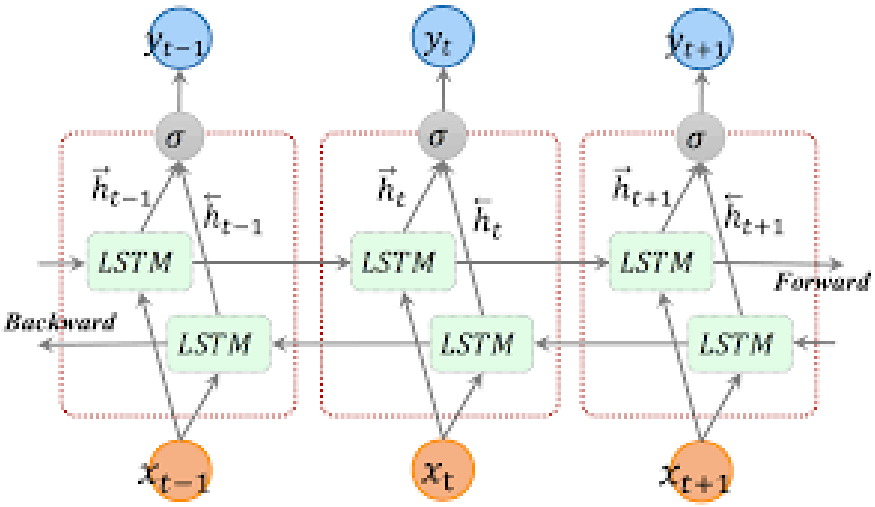
weight.

```
stock_meme_words = {
    'moon': 4.0,
    'mooning': 4.0,
    'DD': 2.8,
    'BTFD': -2.8,
    'mars': 0.5,
    'tendies': 3.5,
    'rocket': 2.5,
    'hold': -2.5,
    'diamond': -2.0,
    'paper': -3.0,
    'yolo': 1.0,
    'f': -4.8,
    'sold': -3.0,
    'sell': -3.0,
    'buy': 2.5,
    'buying': 2.5,
    'bought': 2.5,
    'pullback': -4.0,
    'bullish': 3.8,
    'bearish': -3.8,
    'dip': -4.0,
    '🚀': 3.0,
    '💎': 1.5,
    '📈': 3.0,
    '📉': -3.0
}
```

After computing sentiment scores from our twitter data we were now ready to start building our data and neural network. To start we knew that for our data we would have to split it into training and testing datasets as well as construct a dataframe to store it in an easy to access and understandable way. Thus with some research into the Pandas python library we went about building a method that would load our stock data from yahoo finance and get the following fields: *'adjclose', 'volume', 'open','close', 'high', 'low'*. Now using this data and the date index that came along with it we ran our code to get our sentiment score and add it to the dataframe at the

correct date. With this setup we now added variable to set the ‘number of days prior”, “days ahead” and “training ratio” which represented the number of days prior to the target price to look at the stock and sentiment data, the number of days ahead to predict the price, and the ration of training to testing sample sizes. In the case of this report the values we chose for these were 30 days prior, 1 day ahead, and 80% training to 20% testing sample sizes.

Moving on now that our data was organized correctly, we went about building our neural network. Once again we used a third party library called Keras which allowed us to build a configurable Sequential model. We decided to use Bidirectional Recurrent neural network with LSTM because LSTM works best with sequential data which is perfect for stock market data and because the Bidirectional aspect allowed the networks to have both backward and forward information about the sequence at every time step and this gave our model more information about the input which would help it make a better prediction for the output. For our loss function we decided to use mean square error which finds the average squared difference between the estimated values and the actual value. We chose this because it seemed like a good option for finding the error between a predicted stock price and actual stock price. Lastly, we used an “Adam” optimizer when compiling our model and this is an algorithm for stochastic gradient descent for training deep learning models.



$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

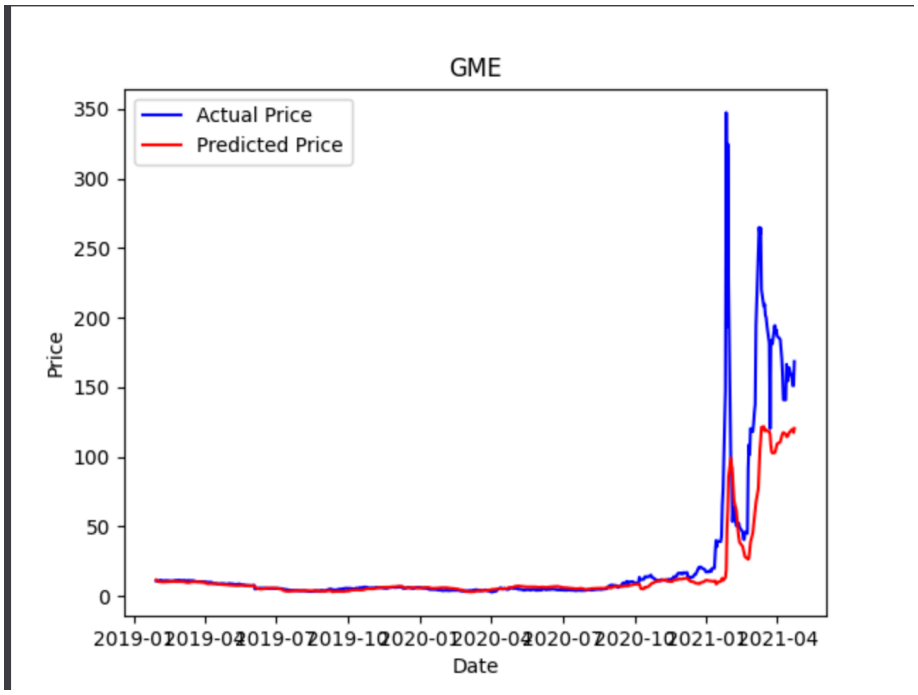
Now that our model and data was complete we were finally ready to train our model and make predictions using this trained model. When training our model there were three main parameters we messed around with to try to find an ideal output. These values were “Units”, “Batch Size”, and “Epochs”. These values represented the number of hidden perceptrons used in each layer, the number of training examples utilized in one iteration, and the number of iterations ran. An issue we found with these is that as we increased all of these values the time it took for our model to run became increasingly longer, so much so that it became impractical for us to sit around for hours to get a result. However we do think if we were able to increase these values our results may have been more accurate as it would give the network more information to work with. Another issue regarding time was that it took our sentiment analysis quite a bit to read all tweets from a date and then compound a mean sentiment score from them so the run time for our model was already kind of an issue. Looking past that, once our model did complete

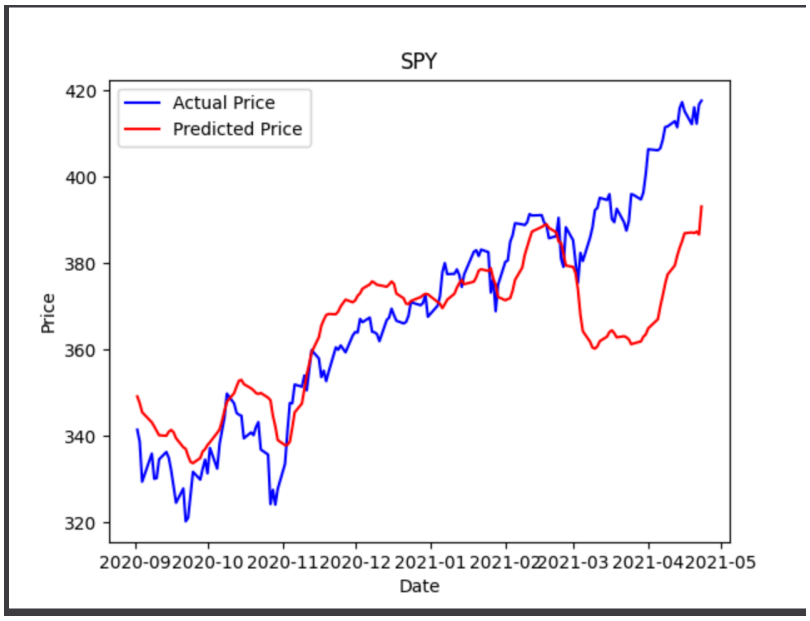
training we used the test data to predict prices a day ahead for the “X_test” data. Once we had these predicted prices we then compared them with the true values of the “Y_test” data. We plotted this data to help us visualize it but we also decided to calculate the profit we would have made if we actually followed our models predictions. We did this by looking at the value of the predicted price and seeing it was higher or lower than the actual price on that date. If the price was higher we would proceed to “buy” the stock and now looked at the actual price of the stock the next day. If this actual price followed our model and went higher we then recorded the profit we would have made and if the true price actually went down then our loss. We followed this same process for when our model predicted the stock would go down and added to our “sell” profit if the true price went down.

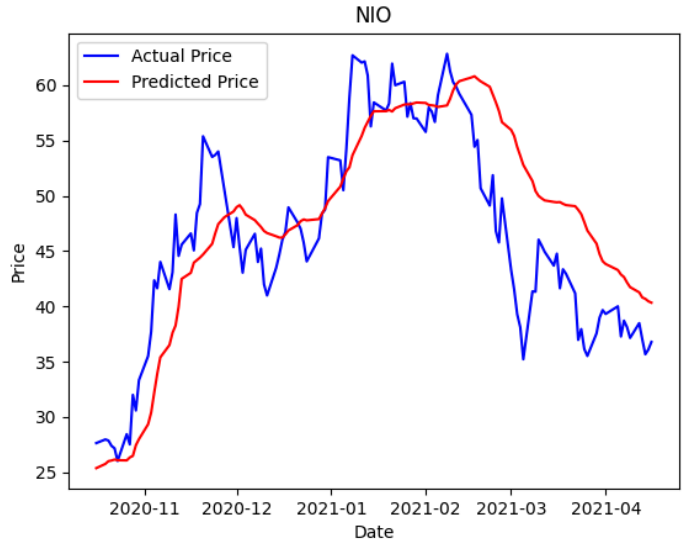
	open	high	low	close	adjclose	volume	ticker	sentiment	predicted_adjclose_1	true_adjclose_1	buy_profit	sell_profit
11/13/20	355.269989	358.8999939	354.7099915	358.1000061	355.4159546	62892200	SPY	0.008047619	359.38773	359.852478	4.436523438	0
11/16/20	360.980011	362.7799988	359.5899963	362.5700073	359.852478	74541100	SPY	-0.02559	362.86334	357.9170532	-1.935424805	0
3/1/21	385.5899963	390.9200134	380.5700073	389.5799866	388.308197	104945700	SPY	0.194977778	379.12433	385.2781372	0	3.030059814
3/2/21	389.8200073	390.0700073	386	386.5400085	385.2781372	79389200	SPY	0.3018125	377.69897	380.1748352	0	5.103302002
3/3/21	385.7900085	386.8299866	381.3099976	381.4200134	380.1748352	119482700	SPY	0.419085714	373.82922	375.4702454	0	4.704589844
3/4/21	381.2200012	384	371.8800049	376.7000122	375.4702454	182856500	SPY	0.419085714	368.03012	382.3776245	0	-6.90737915

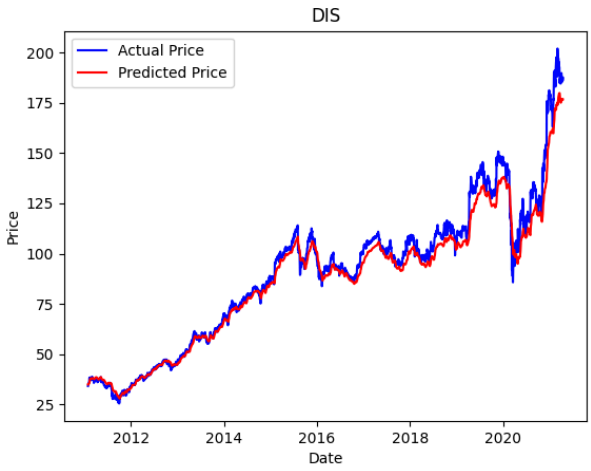
EVALUATION & RESULTS

As stated above we evaluated our results in a few different ways. The first was visually as we graphed our predicted price against the true price of the test data.


GME
Actual Price
Predicted Price
Price
Date
350
300
250
200
150
100
50
0
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04


SPY
Actual Price
Predicted Price
Price
Date
420
400
380
360
340
320
2020-09
2020-10
2020-11
2020-12
2021-01
2021-02
2021-03
2021-04
2021-05





When looking at the graphs our results seem pretty accurate and the predicted price does follow the true prices curve fairly well which was an encouraging sign however when we looked at some of our other evaluations we found some other observations. Another way we evaluated our results was using the profit that would have been made if our price predictions were followed, and using this information we found the total sum of profit that would have been made, the mean squared error of our results, and the accuracy of our model that recorded each correct prediction we made. We also decided to take the last sequence of data we had and use our trained

model to predict the price one day away from the current date and record this down. The results are below:

GME

```
Future price after 1 days is 123.93$
mse loss: 0.008155892603099346
Mean Squared Error: 5.611417771566952
Accuracy: 0.5230496453900709
Total buy profit: 6.368034124374367
Total sell profit: -151.29711842536935
Total profit: -144.929084300995
```

SPY

```
Future price after 1 days is 400.06$
mse loss: 0.006740475073456764
Mean Squared Error: 220.76688089210046
Accuracy: 0.4968944099378882
Total buy profit: 37.849029541015625
Total sell profit: -26.157470703125057
Total profit: 11.691558837890568
```

After observing these results we saw that our accuracy value was only around 45% to 50% which is not terrible given how difficult it is to predict the stock market, but not good enough to actually put real money into our predictions or really give us confidence in our model. We also saw that for tech stocks and blue chip companies we did make a pretty substantial profit, but that is probably due to the “bullish” market that has been happening over the past few years

in which it is hard to lose money investing in stocks.

CONCLUSION

The model described in this paper model predicted the direction of stock price at a satisfactory rate. However, it had not been deemed great enough to be consulted when making investments. This seemed to be a recurring theme when observing other stock market projects work and it goes to show just how hard it is to predict the stock market even if you try to account for social media sentiment. For future projects maybe adding some more sentiment data from news headlines or more financial indicators could help improve our model. Also an alternate approach could be to target one person's tweets such as Elon Musk and use that to predict the price of Tesla stock.

This project has provided a lot of insight into Python, the stock market, machine learning, neural networks, APIs, and the completion of real world projects which led to great takeaways.

Acknowledgements

The project was split up in a way such that Anthony worked on retrieving the data from Twitter and Reddit. Anthony worked on performing the sentiment analysis, however, due to the problems with MonkeyLearn, Josh implemented the current sentiment analysis model. Josh mostly worked on integrating the sentiment analysis results into the machine learning algorithm. The presentation and report were mostly focused on by Anthony, however, Josh made sure to integrate his experience in the approach, results, and conclusion.

REFERENCES

Bloomberg. (2014, February 20). Trending on Twitter: Social Sentiment Analytics. Bloomberg.

https://www.bloomberg.com/company/press/trending-on-twitter-social-sentiment-analytics/.

Brownlee, J. (2019, July 24). Your First Deep Learning Project in Python with Keras Step-By-Step. Machine Learning Mastery. https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/.

Cunningham, A. (2018, November 9). Simple Stock Price Prediction with ML in Python - Learner's Guide to ML. Towards Data Science. https://towardsdatascience.com/simple-stock-price-prediction-with-ml-in-python-learners-guide-to-ml-76896910e2ba.

Ganti, A. (2020, December 28). Adjusted Closing Price Definition. Investopedia. https://www.investopedia.com/terms/a/adjusted_closing_price.asp.

Lubitz, M. (2017). Who drives the market? Sentiment analysis of financial news posted on Reddit and Financial Times (thesis). University of Freiburg, Freiburg im Breisgau.

Rao, T., & Srivastava, S. (2012). Analyzing Stock Market Movements Using Twitter Sentiment Analysis.

Ren, R., Wu, D., & Liu, T. (2019). Forecasting Stock Market Movement Direction Using Sentiment Analysis and Support Vector Machine. IEEE SYSTEMS JOURNAL, 13(1), 760–770. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8326522&casa_token=HYizvk8Pe6sAAAAA:7PQNjYJls02B9n4rQWxDN2xZZVJRdi2-hezz0qu7qyT6NYOsDiLC_N0JhVUbLRxEuEnQQgz-bAw&tag=1

Treadway, Andrew. (2021, March 29). Yahoo_fin Documentation. Open Source Automation. http://theautomatic.net/yahoo_fin-documentation/#get_data.

T. *Twint API*. GitHub. https://github.com/twintproject/twint

Sentiment Analysis with Python - A Beginner's Guide - AlgoTrading101 Blog. (2020, July 7). Quantitative Trading Ideas and Guides - AlgoTrading101 Blog. https://algotrading101.com/learn/sentiment-analysis-python-guide/

Installing NLTK Data — NLTK 3.6.2 documentation. NLTK. https://www.nltk.org/data.html

Bohen, T. (2020, December 16). *40 Stock Market Terms You Need to Learn + [Infographic]*. StocksToTrade. https://stockstotrade.com/40-trading-terms-beginners-infographic/

Brownlee, J. (2021, January 12). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. Machine Learning Mastery. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:%7E:text=Adam%20is%20a%20replacement%20optimization,sparse%20gradients%20on%20noisy%20problems.

Brownlee, J. (2021, January 17). *How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras*. Machine Learning Mastery. https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/#:%7E:text=Bidirectional%20LSTMs%20are%20an%20extension,LSTMs%20on%20the%20input%20sequence.

Rockikz, A. (2020, January 8). *How to Predict Stock Prices in Python using TensorFlow 2 and Keras*. The Python Code. https://www.thepythoncode.com/article/stock-price-prediction-in-python-using-tensorflow-2-and-kera