# STOCK MARKET PREDICTION WITH SENTIMENT ANALYSIS

MANUEL GARCIA ACOSTA, PAPA FALL, VIVSWAN SHAH

Note: To run the program please refer to "README.md" text file in the zip folder.

## 1. Introduction

The stock market is an equity market where buyers and sellers can trade shares of stocks. It allows a wide array of individuals to invest their capital and take part in the economy's growth. A stock market is often seen as one of the most important indicators of the economic strength of a country since an increase in the price of stocks can be thought of as an increase in investment in companies of such country. Because of the importance of the stock market, it is not surprising to notice that stock return predictability is of the most important concern for investors.

In recent times "investors are exposed to an ever-increasing number of new facts, data and statistics every minute of the day. Assessing the predictability of stock returns requires formulating equity premium forecasts on the basis of large sets of conditioning information, but conventional statistical methods fail in such circumstances" [1]. To access all this data and build models that have accurate stock return predictability machine learning has emerged as a valuable resource. Machine learning models have gained popularity in recent times regarding the study of large-scale data sets, and the financial sector has not been impervious to this. Several ML models have been used to study stock returns by using lagged (past) stock returns as features and the future stock return as the output values.

## 2. Project description and goals

The main objective of our project is to predict the daily stock price on the US stock market and, specifically, a selection of 50 companies in the S&P 500 (Standard and Poor's 500) index. We chose this index, which tracks the performance of 500 leading companies in the US economy, since it is one of the most used indices for tracking the performance of the US stock market. "Directional prediction of stock returns is based on forecasting whether returns are greater than some pre-specified threshold. Previous research mainly focuses on sign prediction, where this threshold is equal to zero (i.e., whether the return is positive or negative)" [2].

For our study we will be using neural networks with data from the years 2014-2019 to train models that predict the next day's return (price). We do not consider the most recent years because of the large effect of the COVID-19 pandemic on the global economy. Choosing a year such as 2020 would be difficult due to the increased volatility –as compared to a normal scenario- that the pandemic had on the stock market.

We will be using LSTM (Long short-term memory) neural networks with sentiment analysis with pre-trained BERT (Bidirectional Encoder Representations from Transformers) using information from the years 2014-2018 to train models that predict the return for 2019. The features we will use are the lagged entries of each stock and sentiment analysis of headlines where the company is mentioned. We intended to follow Nevasalmi methodology, who states that lag lengths beyond ten trading days are found to be uninformative [2] but ended using 50 lagged entries due to the improvement in the results. For the performance metric we will be measuring the accuracy of the models using the Mean Absolute Percentage Error (MAPE).

## 3. Data Preprocessing

To train a neural network model like LSTM, data needed to be normalized properly. The process of the data preprocessing was as follows:

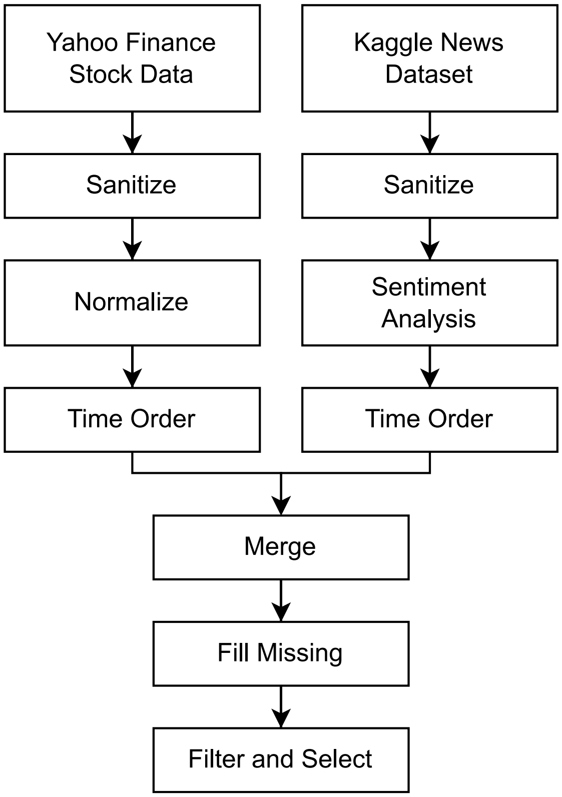

*Figure 1. Data Collection and Data Preprocessing*

1. Collecting Sentiment Analysis data.
   a. Download news headline data (the dataset "analyst_ratings_processed.csv" was acquired from *kaggle* [3]).
   b. Sanitizing by removing any double spaces and removing any invalid or unreadable characters.
   c. Distributing or sorting news by stock tickers.
   d. Sorting news by reverse chronological order.
   e. Passing news headlines to FinBERT (Financial Bidirectional Encoder Representations from Transformers) network to get sentiment data.
2. Collecting Stock data.
   a. Download daily stock data for each ticker from Yahoo Finance API.
   b. Sanitizing by removing any black spaces or empty rows.
   c. Normalizing data by L1 and L2 normalization (this creates two data sets).
   d. Sorting news by reverse chronological order.
3. Merging sentiment data and stock data.
   a. Connecting sentiment data and stock data by date and tickers and if multiple news or sentiment data in same date then merging them by taking a mean.
   b. Filling the missing sentiment data by interpolating using a decreasing exponential curve ($e^{-0.5x}$).
   c. Filtering and selecting data from 2014-2019.

4. Now this data can be used to train LSTM models and predict the next day's stocks price.

## 4. Framework: FinBERT Networks

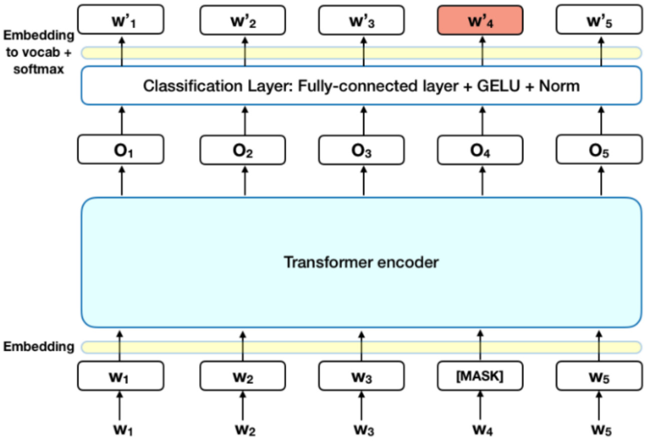
w'₁
w'₂
w'₃
w'₄
w'₅
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O₁
O₂
O₃
O₄
O₅
Transformer encoder
Embedding
w₁
w₂
w₃
[MASK]
w₅
w₁
w₂
w₃
w₄
w₅

*Figure 2. BERT Model [6]*

FinBERT is a deep neural network model based on BERT by Google, which is a language model. FinBERT is specifically trained in sentiment analysis on finance related datasets. Unlike a LSTM neural network, which is not fine tunable, BERT is a fine tunable model. Fine tunable models do not require complete retraining from scratch to learn a new related dataset or task. That is, while BERT was designed to predict the next word or draft essays, since its fine tunable it can be used to predict sentiment with just little more training. A new sentiment dataset can be added to a pre-trained BERT network without needing to be retrained from scratch on such dataset, this feature made possible the creation of the FinBERT network.

FinBERT takes text as input and return values from positive, negative, and neutral sentiments. The output values range from 0 to 1, where 0 is no inclination and 1 is full inclination towards that sentiment. In this project a pre-trained FinBERT model from HuggingFace  was used to convert news headlines to sentiment data which combined with the stock data was used to train LSTM networks.

## 5. Framework: LSTM Networks

LSTM [8] is a commonly used neural network model used to training and predicting time series data like language, stocks, videos, etc. LSTMs are one of the best time-series learning models, they are not fine tunable and can take long to train. While new types of neural networks models like transformers (used in BERT) are being developed and can surpass these flaws, none are of general purpose (i.e., can be trained on any type of time series data). For example, BERT only works for language-based datasets.

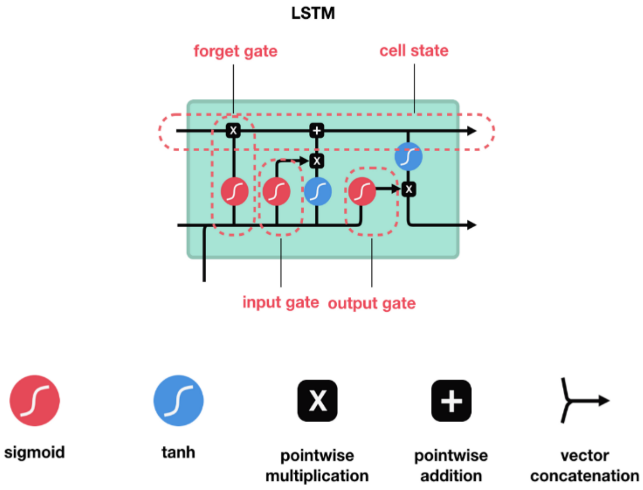

*Figure 3. Internal Design of a LSTM Layer*

For our project, the neural network model is shown in Figure 4. We used 1 to 4 LSTM layers combined along with Dropout layer (rate of 0.33 or 33%). Dropout layers are used to make sure that no neuron is overtaking the layer, that is, no single neuron is making all the decision for the layer. The dropout is done by randomly turning off a set percentage (rate) of neurons in each step of training, this way the information is distributed throughout the network. Dropout layers also help with addressing overfitting. Finally, a dense layer was used to give the output in a proper format.

The input was made up of the last 50 days of normalized stock data combined with its respective sentiment data to give the output for the next day stock in normalized format.

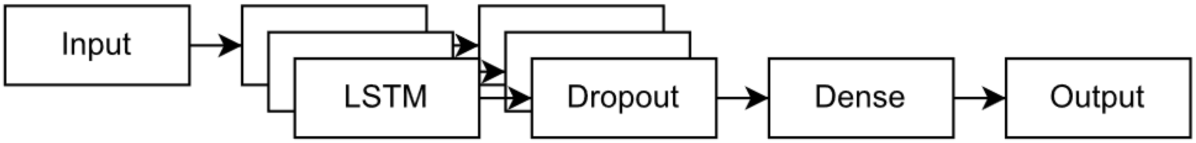

*Figure 4. Design of the Full LSTM Network used to predict next day stocks.*

## 6. Training and testing of the model

For training the model, the data (merged stock/sentiment data after preprocessing) was divided into an 80%-20% split in chronological order for training (80%) and testing (20%).

The parameters are:

- Time-step: 50 days (number of previous days data as input to get next day prediction as output).
- Optimizer: Adam.
- Learning rate: 0.001.
- Loss: Mean-Squared Loss.
- Batch Size: 64.

The hyper parameters (the parameters which were varied between models to evaluate different models) are:

- Number of LSTM layers: 1-4.
- Sentiment data: True and False (we tested the model with sentiment data and without. sentiment data to see if the addition of sentiment data makes any difference in the accuracy of the model).
- Epochs (number of times the model was trained before testing): 50, 100 and 150.

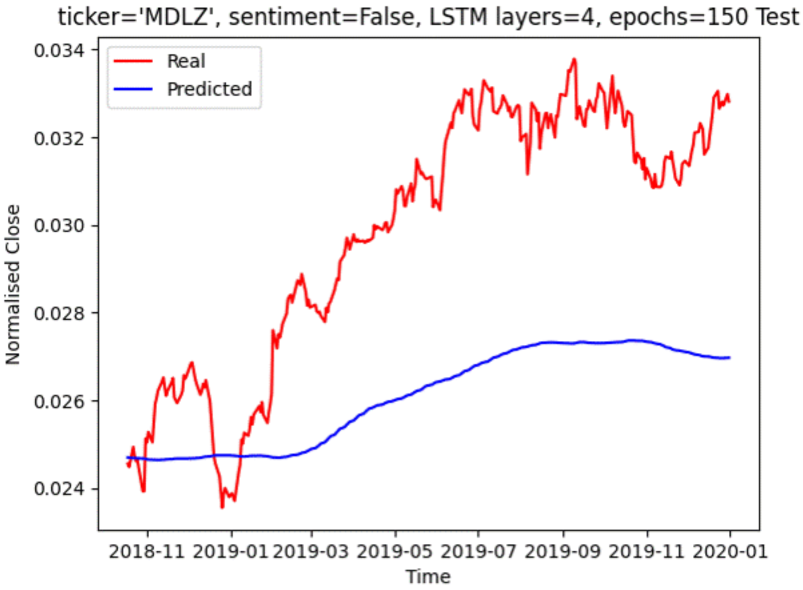

*Figure 5. Results of the LSTM Network next day predictions filtered by epochs = 150*

## 7. Analysis of results/Discussion

While our Neural Networks were trained using the MSE, we used MAPE for evaluating the test errors.

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

$A_t$ = Observed value of time series at time *t*

$F_t$ = Predicted value of the time series at time *t*

One of the reasons we did not use MAPE for training was that it penalizes more heavily the errors where the predicted value is larger than the observed value ($A_t < F_t$). [9] This results in models which predictions are biased low. Another reason was that the computational time required to train models with MAPE far exceeded the runtime for models trained with MSE. Finally, MSE was not used to compare testing errors because the scale in the time series of the companies we selected were quite different. For example, a share of The Coca-Cola Company (KO)[1] was well below 70 USD while a share of The Home Depot exceeded 100 USD in the considered timeframe.

Models were trained and compared across the 50 companies that we selected using both L1 normalization and L2 normalization. Normalization with L1 norm resulted in models with less

[1] KO is what we call a ticker, this is a string that identifies a security. For our study, the ticker identifies the stock of the companies we are working with.

accuracy (and thus possible real-world applications) than normalization with L2 norm. The top and worst 3 performers for L1 normalization are shown in the next figure.

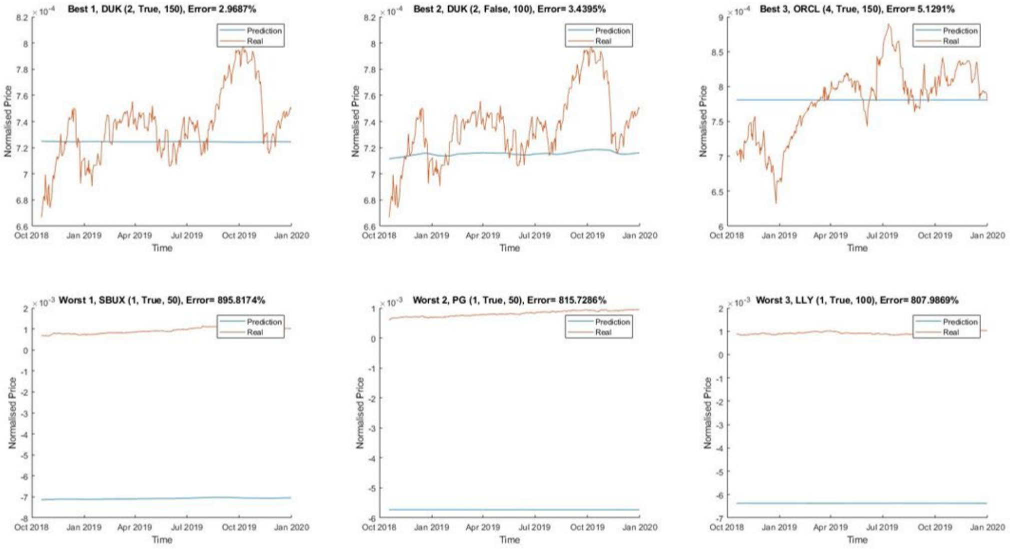

*Figure 6. Best and worst performers with L1 normalization processed data.*

The best performers were

1. Duke Energy Corporation (DUK): 2 hidden LSTM layers, sentiment analysis and 150 epochs.
2. Duke Energy Corporation (DUK): 2 hidden layers, no sentiment analysis and 100 epochs.
3. Oracle Corporation (ORCL): 4 hidden LSTM layers, sentiment analysis and 150 epochs.

The worst performers were

1. Starbucks (SBUX): 1 hidden LSTM layer, sentiment analysis, 50 epochs.
2. Procter & Gamble (PG): 1 hidden LSTM layer, sentiment analysis and 50 epochs.
3. Eli Lilly and Company (LLY): 1 hidden LSTM layer, sentiment analysis, 100 epochs.

As can be seen, the predicted time series -in blue- for the models with L1 normalization appear to behave as constant functions. This behavior is interesting because it is similar to what we would expect to get while using an Autoregressive Integrated Moving Average (ARIMA) model in a time series with high volatility such as the time series of the stock of a company. What we saw here is that this normalization resulted in effectively no learning.

On the other hand, L2 normalization provided better results. The next image portrays the 3 best/worst performers under this regularization

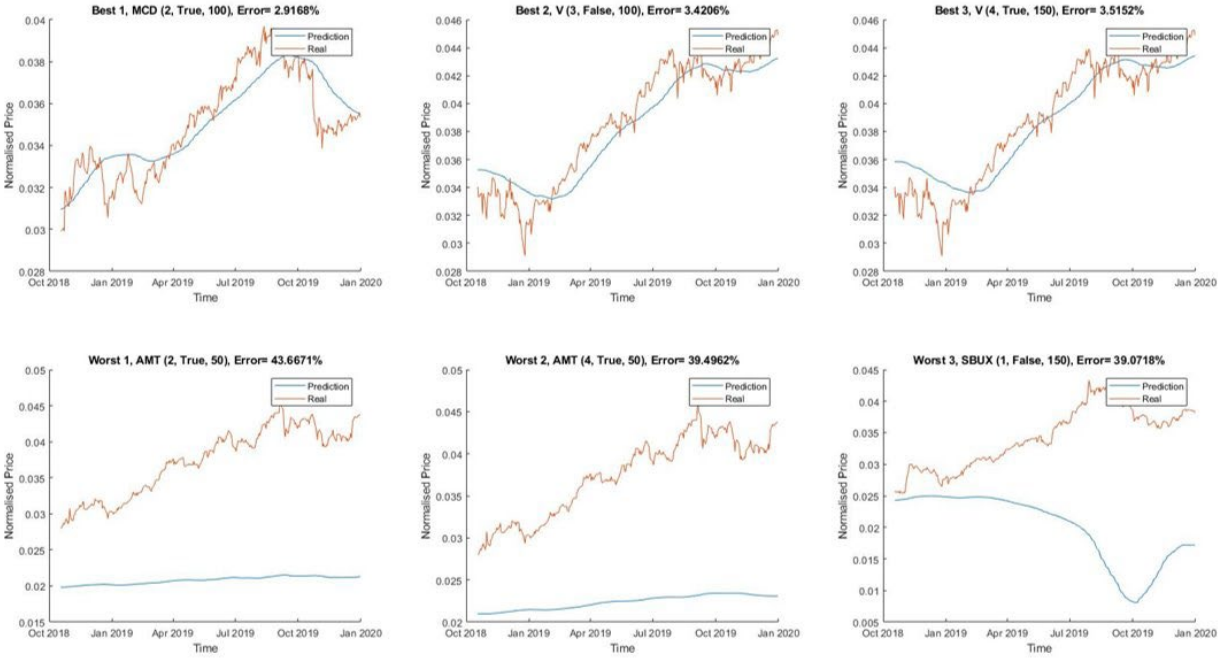

*Figure 7. Best and worst performers with L2 normalization processed data.*

The best performers were

1. McDonald's (MCD): 2 hidden LSTM layers, sentiment analysis, 100 epochs.
2. Visa (V): 3 hidden LSTM layers, no sentiment analysis, 100 epochs.
3. Visa (V): 4 hidden LSTM layers, sentiment analysis, 150 epochs.

The worst performers were

1. American Tower Corporation (AMT): 2 hidden LSTM layers, sentiment analysis, 50 epochs.
2. American Tower Corporation (AMT): 4 hidden LSTM layers, sentiment analysis, 50 epochs.
3. Starbucks (SBUX): 1 hidden LSTM layer, no sentiment analysis, 150 epochs.

While the worst performers predictions might resemble the ones from L1 normalization, it can be seen that the best models were able to learn in an acceptable manner the complex structure of these time series. This shows promise in terms of real-world applications of our models such as creating a trading strategy based on the predicted values of a given stock (or set of stocks).

While measuring the performance of the trained models some trends can be observed. First, the variation in the error for stocks with high volatility was larger than the error variation in stocks with low volatility. Secondly, models with 3 LSTM layers were able to achieve the smallest errors. Networks with 1 and 2 LSTM layers had large errors and these errors decreased in the networks with 3 layers to be increased in the networks with 4 LSTM layers. This might be an indicator of 3 LSTM layers being a sweet spot between underfitting and overfitting models.

It can be seen also that networks with more epochs performed better, in our case we used 50, 100 and 150 epochs. Continuing to increase the number of epochs will most likely lead to overfitting the training data, and thus, the test errors will begin to increase. Finally, models with sentiment analysis were subject to more variation in the test error than the models with no sentiment analysis. We think that the reason for this is that the news data set we used proved to be too small. The following figure includes graphs that give us a visual representation of these results.

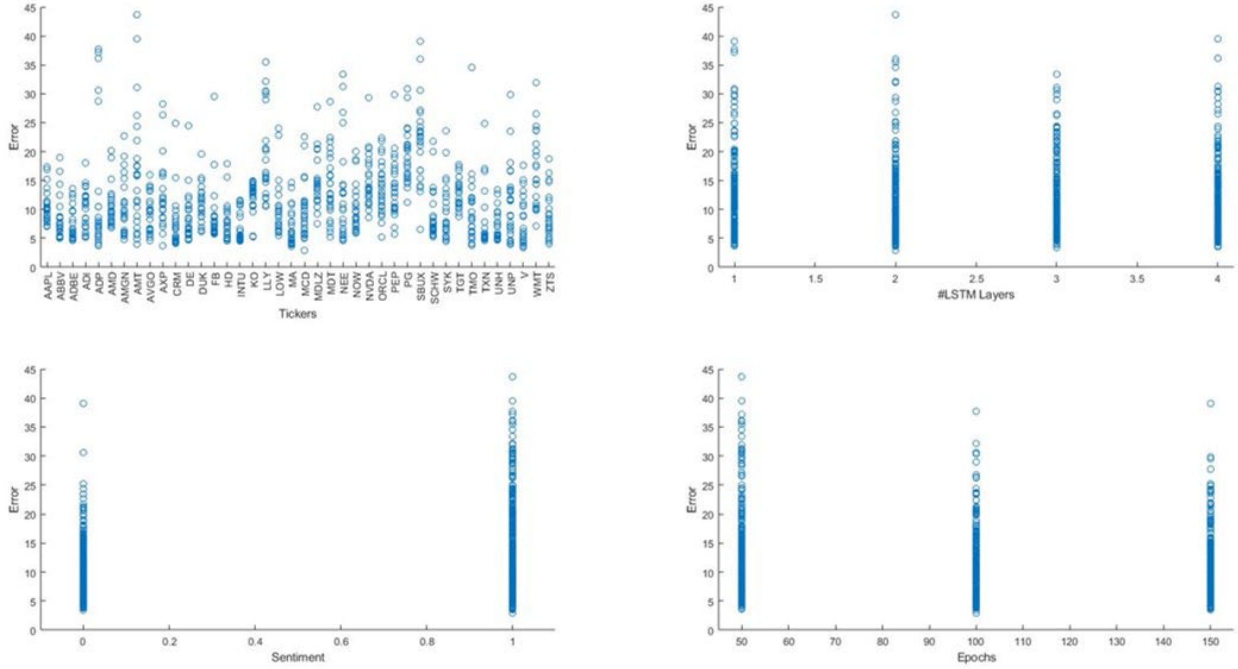

*Figure 8. Top-left, test errors by ticker. Top-right, test errors by number of LSTM layers. Bottom-left, test errors if model used sentiment analysis or not. Bottom-right, test errors by number of epochs.*

## Conclusion

From our results we can conclude that LSTM networks provide a promising approach to analyzing stock market trends. Immediate future directions for this work are expanding the number of companies included to cover all the companies in the S&P 500. Also, future studies might add more input factors to the neural networks such as stock splits, macroeconomics variables or indexes such as the VIX index, the NASDAQ index, the Dow Jones index or even the S&P 500 index itself. These new variables will need to be subject of model selection to train the neural networks with input variables that are relevant while trying to predict the next day's price of stocks.

As mentioned previously, networks using sentiment analysis were subject to more variability in the test errors and this might be because of the size of the data set currently used. If this is the case, then using a larger dataset from a news dedicated API such as *mediastack, The New York Times*

API or *News API* (from Google) might improve the results. If the core of the issue resides elsewhere then sentiment analysis needs to be addressed differently when trying to ensemble it with LSTM neural networks.

## References

[1] A. G. Rossi, "Predicting stock market returns with machine learning," Georgetown *University,* 2018

[2] L. Nevasalmi, "Forecasting multinomial stock returns using machine learning methods," *The Journal of Finance and Data Science*, vol. 6, pp. 86–106, Nov. 2020, doi: 10.1016/j.jfds.2020.09.001.

[3] "Daily Financial News for 6000+ Stocks | Kaggle." https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests (accessed Apr. 28, 2022).

[4] "Yahoo Finance - Stock Market Live, Quotes, Business & Finance News." https://finance.yahoo.com/ (accessed Apr. 28, 2022).

[5] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," *arXiv:1908.10063 [cs]*, Aug. 2019, Accessed: Apr. 28, 2022. [Online]. Available: http://arxiv.org/abs/1908.10063

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019, Accessed: Mar. 21, 2022. [Online]. Available: http://arxiv.org/abs/1810.04805

[7] D. Araci and Z. Genc, "ProsusAI/finbert · Hugging Face." https://huggingface.co/ProsusAI/finbert?text=Stock+Market+Prediction+with+Sentiment%0AAnalysis (accessed Apr. 28, 2022).

[8] A. Moghar and M. Hamiche, "Stock Market Prediction Using LSTM Recurrent Neural Network," *Procedia Computer Science*, vol. 170, pp. 1168–1173, 2020, doi: 10.1016/j.procs.2020.03.049.

[9] S. Makridakis, "Accuracy measures: theoretical and practical concerns," *International Journal of Forecasting*, vol. 9, no. 4, pp. 527–529, Dec. 1993, doi: 10.1016/0169-2070(93)90079-3.