Stock Price Movement Prediction aided by News Sentiment Analysis

Poulinakis Konstantinos
National Technical University of Athens, NTUA
Athens, Greece
poulinakis.kon@gmail.com

Depastas Vasileios
National Technical University of Athens, NTUA
Athens, Greece
vasileiosdepastas@mail.ntua.gr

***Abstract*—Stock price prediction is a complex and long lasting research field in the field of finance. The stock market environment is very uncertain as the stock prices fluctuate based on a plethora of factors. Traditional prediction methods use historical data for predicting future stock prices, however, in recent years machine learning and deep learning techniques have been increasingly utilized for such tasks. News articles are considered to be among the factors that influence stock prices as they can affect peoples' sentiments regarding specific stocks or the overall stock market. This paper focuses on devising distinct machine learning algorithms, including deep learning, that take news articles as input, extract the sentiment of those and output a prediction on the S&P 500 stock market index movement. The efficiency of each algorithm as well as the ability to predict such movements through news articles is evaluated.**

I. Introduction

Stock price movement prediction is a crucial aspect of investment companies' work, such as Hedge Funds. However, stock prices are affected by a huge amount of factors, including both stock-specific factors like earnings [1] as well as macroeconomic factors such as interest rates and exchange rates [2] among others. Moreover, stock markets are affected by news through a variety of means, such as their rapid spreading through social media which affects investors' sentiment [3].

On the other hand, based on the Efficient-Market Hypothesis (EMH), share prices reflect all information and consistently beating the market is impossible. According to this theory, stocks always trade at their fair value on stock exchanges and thus it's not possible for an investor to purchase undervalued stocks or sell stocks for an inflated price. This is making it impossible to beat the overall market via stock selection or timing the market and the only way for an investor to yield higher returns is by pursuing riskier investments.

News and their effect on stock prices has been subject to several studies in the literature, since they can be collected and analyzed in order to help predict stock prices. According to [4], there is considerably more evidence of a strong relationship between stock price changes and information. However, it has traditionally been a challenge for the researcher to be able to parse through which news stories are relevant and which are not. Given that there is vast amount of news stories to work through, this results in a computational challenge. Advancements in textual analysis, though, have allowed better identification of relevant news.

In this work, we experimented with data collection for news, through GoogleNews API [5] and NYTimes API [6] in order to gather news data for a 10-year period, between 2010 - 2020. However, the GoogleNews API has been deprecated and scraping GoogleNews without an API can result in IP banning. Experimenting with the NYTimes API resulted in successful news stories data collection for the selected time period. Nevertheless, the returned results were found to be very noisy and in many cases irrelevant to the stock under consideration, given that the search query for the results is being searched in title as well as the news body. This is why, after experimentation, we collected a dataset of directly-scraped raw news headlines from [7]. The dataset contains headline, URL, publisher, date and stock ticker and was filtered to contain news for the top 35 stocks in the S&P500 index which account for roughly 48% of the index's value. Since these companies alone represent almost half of the index, we speculate that sentiment analysis conducted on news related to these 35 companies roughly represents the overall market sentiment in a given day. We then proceed to collect financial data for the S&P500 market index through Yahoo! finance API [8] for the selected time period. For any given day, we create a sentiment score that is derived as the average news sentiment score of the 35 companies mentioned above. We then proceed to link these scores with the S&P500 index price for all stock market active days inside the selected time period of 2010 to 2020. Several machine learning classifiers are trained on the sentiment scores in order to predict the stock's price movement.

Finally, we model again the problem as a time-series learning task using an LSTM network. First, the only input given to the LSTM network is the closing price of previous days. Then we proceed to also include the sentiment scores as features. Using the LSTM we approached the problem both as a regression and a classification problem. Performance in all cases is compared.

II. Related Work

The news sentiment stock price or movement prediction problem has been thoroughly examined in the literature. [10] examines the predictive power of traditional machine learning

classifiers such as SVM, Random Forest and Naive Bayes on stocks based on sentiment extracted from news articles. News documents are tokenized, stop-words are removed and stemming is applied to the text. Then, a bag of words technique is followed and a polarity dictionary is created in order to count the words matching to positive polarity or negative. The count of positive polarity words minus the count of negative polarity words in a news document gives the final sentiment outlook of the news article, which is positive if the differnce is greater or equal than 0, otherwise it is considered negative. By taking financial and news data about Apple Inc for 3 years and feeding the sentiment scores to a classifier, the authors claim to have achieved over 75% accuracy scores in each case. In [11], four different sentiment dictionaries are used to extract sentiment vectors from news articles. Those are SenticNet 5, SentiWordNet 3.0, Vader and LoughranMcDonald Financial Dictionary 2018. Then an SVM classifier and an LSTM network are devised for stock future trend classification. Based on accuracy and F1 scores, the authors conclude that LSTM generally performs better than the SVM, with the prediction scores varying depending on the sentiment dictionary used. In [12] sentiment analysis of financial news is utilized, along with features extracted from historical stock prices to predict the future stock market behaviour. The authors use a Naive Bayes, an SVM and a K-NN algorithm for the classification task, considering different news types related to companies, markets and financial reports. Textual information is first preprocessed and then n-gram feature extraction is used along with the tf-idf feature weighting method resulting in a single news polarity for each news article, positive or negative. The authors examine 3 different companies, Microsoft, Yahoo and Apple and result in very different accuracy scores depending on the company and the classification model with scores ranging from 58-86%. The authors on [13], study the use of time-series data along with NLP techniques to extract information about an individual stock's chance of closing higher or lower than its opening price utilizing an LSTM network. News data from reuters.com were gathered and VADER was utilized as a sentiment analyzing tool to extract news sentiment. By experimenting with a number of individual stocks and related news articles, the study concludes that more data might be needed for the LSTM to avoid overfitting, as the authors experiments' yielded a 50% prediction score. Another observation suggested that sentiment analysis on individual stocks may have better prediction rates with aspects of the stock other than price such as volume or liquidity.

III. METHODOLOGY

A. Dataset and Feature Creation

Our Dataset consists of 3 sentiment values (Positive, Neutral, Negative), the previous day's closing price and a price movement label for almost every day from 2010-01-01 up to 2020-01-01 for the the S&P500 index. First, news and financial articles for the top-35 S&P500 companies are gathered. Then, for each one of them a sentiment analysis is executed using the NLP Transformer FinBERT [9] which is a fine-tuned version of BERT [14] NLP transformer. FinBERT is a state-of-the-art sentiment analysis model that has been re-trained, using transfer learning, on financial texts and articles. For each date we average the sentiment scores to create a single 3 valued vector of $[Positive, Neutral, Negative]$. The values always add up to 1. This sentiment vector represents, with some error, the overall sentiment for the market at each date. Hence we can consider that with these features we can achieve predictions about the index's price movement. The label is computed as:

$$label = sign(close\_price_t - close\_price_{t-1}) \quad (1)$$

A negative sign represents a decline in price while a positive sign represents an increase. We represent decrease with 0 and increase with 1.

We can also incorporate the previous day's close-price as an added feature. Previous day's close-price is first Normalized using the min-max transformation

$$X_t^n = \frac{X_t - min\{X\}}{max\{X\} - min\{X\}} \quad (2)$$

where X_t^n the value of feature X during timestep t after normalization. Normalized data is effectively transformed into the [0, 1] range.

After the pre-processing takes place we are left with 2608 data points, consisting of 4 features and 1 label. Out of those, 2608 data points (54.7%) belong to the positive (rising-price) class and the rest (45.3%) to the negative class.

Traditional M.L. algorithms are trained using 70% of the dataset (1825 dates) and tested on the remaining 30% (783 dates), with the dataset shuffled before splitting. However, for the LSTM the first 1825 data points (70%) are used, without shuffling, in order to train the network while the next 391 consecutive data points (15%) are used for validation and the last 391 (15%) points are used for testing.

B. Traditional Machine Learning

We trained 4 different machine learning algorithms and tested them on the 783 (30%) unseen dates. The algorithms were trained using only the sentiment scores as features, leaving past price information unused since the previous day's close-price does not offer any valuable information to the algorithms on its own.

We trained Logistic Regression, K-Nearest-Neighbor, SVM and Random Forest classifiers. All of the classifiers gave results better than a random coin toss and better than 54.7% accuracy which a dummy classifier always choosing the most frequent class (rising price) would achieve.

In order to choose the best possible parameters we use 5-fold cross-validation [15] using only the training set so that no information about the test set is leaked. The parameter combination that provides the best overall scores on the training set is used for the final algorithm. Finally, we evaluate all algorithms on the unseen test set. The results are presented in table I.

Logistic Regression is a classification algorithm that works by trying to learn a function approximating a conditional probability $P(Y|\vec{X})$. Logistic Regression assumes that probability

$P(Y|\vec{X})$ can be approximated by a sigmoid function on a linear combination of the input features. This is for a single training sample point $(\vec{x}, y)$ where y is the class label:

$$P(Y = 1|\vec{X} = \vec{x}) = \sigma(z) \tag{3}$$

$$z = w_0 + \sum_{i=1}^{m} w_i x_i = \vec{w}^T \vec{x} \tag{4}$$

where

$$\vec{x}^T = (1, x_1, x_2, ..., x_m), and, \vec{w}^T = (w_0, w_1, w_2, ..., w_m)$$

Therefore, another form of that assumption is the following:

$$\begin{aligned} P(Y = 1|\vec{X} = \vec{x}) = \sigma(\vec{w}^T \vec{x}) = \\ = \sigma(w_0 + w_1 x_1 + ... + w_m x_m) \end{aligned} \tag{5}$$

$$\begin{aligned} P(Y = 0|\vec{X} = \vec{x}) = 1 - P(Y = 1|\vec{X} = \vec{x}) = \\ = 1 - \sigma(\vec{w}^T \vec{x}) = 1 - \sigma(w_0 + w_1 x_1 + ... + w_m x_m) \end{aligned} \tag{6}$$

Utilizing the above equations for the conditional probabilities of the two classes $y = 0$ and $y = 1$, the algorithm select values of $\vec{w}$ maximizing that probability for all training data sample points. The way parameters are chosen by the algorithm is using Maximum Likelihood Estimation. This is equivalent to writing the log-likelihood function and finding the values of $\vec{w}$ that maximize that function. Since the prediction task is binary and the logistic regression function is $P(Y = 1|\vec{X} = \vec{x})$, we can interpret the logistic regression model as a Bernoulli random variable $Y|\vec{X} = \vec{x} \sim Ber(p)$, where $p = \sigma(\vec{w}^T \vec{x})$. Therefore:

$$P(Y = y|\vec{X} = \vec{x}) = \sigma(\vec{w}^T \vec{x})^y [1 - \sigma(\vec{w}^T \vec{x})]^{1-y} \tag{7}$$

Assuming each data point is drawn independently, we can write the conditional likelihood of all the data:

$$\begin{aligned} L(\vec{w}) = \prod_{i=1}^{n} P(Y = y^{(i)}|\vec{X} = \vec{x}^{(i)}) \\ = \prod_{i=1}^{n} \sigma(\vec{w}^T \vec{x}^{(i)})^{y^{(i)}} [1 - \sigma(\vec{w}^T \vec{x}^{(i)})]^{1-y^{(i)}} \end{aligned} \tag{8}$$

where the number of independent and identically distributed training datapoints is n:

$$(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), ..., (\vec{x}^{(n)}, y^{(n)})$$

Taking the log of the previous function, we derive the log-conditional likelihood and can maximize the log likelihood instead. Using an optimization algorithm, the best values of can be determined. The partial derivative of log-likelihood with respect to each parameter w_j of $\vec{w}^T = (w_0, w_1, w_2, ..., w_m)$ can be calculated as:

$$\frac{\partial LL(\vec{w})}{\partial w_j} = \sum_{i=1}^{n} [y^{(i)} - \sigma(\vec{w}^T \vec{x}^{(i)})] x_j{}^{(i)} \tag{9}$$

Finally, gradient ascent can be utilized in order to compute the parameter values, by taking small steps in the direction of the gradient, leading eventually to a local maxima. This is an iterative process that adjusts the parameter values in every step according to the following equation:

$$w_j^{new} = w_j^{old} + \eta \cdot \frac{\partial LL(\vec{w}^{old})}{\partial w_j^{old}} \tag{10}$$

In the previous equation, η is the magnitute of the taken step size. By iteratively updating w_j values, the algorithm converges to the best parameter values. Once the best parameter values have been defined, then a new test data sample is classified to one of the two classes based on the conditional probabilities on (7) and (8).

Another classifier devised for the classification task was kNN or k-Nearest Neighbors. kNN is considered one of the simplest supervised algorithms as its training phase only consists of storing the training data.

Then, to make a prediction of a test sample's class, the kNN algorithm finds the k nearest neighbors of a test data sample from the stored training data samples. The default distance metric used is the Euclidean distance, which computes the distance between two data points $\vec{x}^{(i)}, \vec{x}^{(j)}$:

$$d(\vec{x}^{(i)}, \vec{x}^{(j)}) = \sqrt{\sum_{k=1}^{m} (x_k^{(i)} - x_k^{(j)})^2} \tag{11}$$

Once, the k nearest neighbors are identified, the kNN model then predicts the target class label of a test data sample as the class label most often represented among the k nearest neighbors training examples.

Another classifier utilized was SVM. The initial algorithm proposed a method for classification and linear regression, that has becomes a very popular and powerful algorithm in the machine learning space. An SVM is typically a binary classifier, separating multiple data points into two classes. A supervised SVM training is done by placing a line for 2-D dimensional data or a hyperplane for N-D dimensions between two different classes by maximizing the margin between all data points. SVMs try to minimize the classification generalization loss loss by choosing a hyperplane, which has the maximum margin to the separated datapoints.

As some data is not linearly separable, it may be required to move the data into a higher dimensional space. Cover's theorem states that a non-linearly separable space can be transformed to a new space where samples are linearly separable with a high probability, given that the transform is non-linear and the new space has sufficient dimensionality size.

Given a set of pairs $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), ..., (\vec{x}^{(n)}, y^{(n)})$, with $y^{(i)} = -1$ if $\vec{x}^{(i)} \in$ Class 0 and $y^{(i)} = 1$ if $\vec{x}^{(i)} \in$ Class 1, the parameters $\vec{w}$ are calculated such as:

$$\begin{aligned} g(\vec{x}^{(i)}) = \vec{w}^T \phi(\vec{x}^{(i)}) + w_0 >= 0, \text{if } y^{(i)} = 1 \\ g(\vec{x}^{(i)}) = \vec{w}^T \phi(\vec{x}^{(i)}) + w_0 < 0, \text{if } y^{(i)} = -1 \end{aligned} \tag{12}$$

where $\phi(\vec{x}^{(i)})$ is a feature mapping ϕ: $\vec{x}^{(i)} \rightarrow \phi(\vec{x}^{(i)})$ from $\mathbb{R}^m \rightarrow \mathbb{R}^d$, where generally $d >= m$, and $g(\vec{x}^{(i)}) = 0$ is the separating hyperplane's linear equation separating the two classes and $\vec{w}^T = (w_1, w_2, ..., w_d)$, $(\vec{x}^{(i)})^T = (x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)})$.

In the case that no transformation is required and the data are already linearly separable, then $\phi(\vec{x}^{(i)}) = \vec{x}^{(i)}$. In all cases, an objective function is minimized to derive the parameters $\vec{w}$. $g(\vec{x}^{(i)})$ can be rewritten in a form that contains pairs of $\phi(\vec{x}^{(i)})^T \cdot \phi(\vec{x}^{(j)})$ along with some restrictions. This dot product is called a kernel:

$$k(\vec{x}^{(i)}, \vec{x}^{(j)}) = \phi(\vec{x}^{(i)})^T \cdot \phi(\vec{x}^{(j)}) \tag{13}$$

Some of the most popular kernels are linear, polynomial, gaussian radial basis function and the sigmoid kernel. By using a kernel, $\phi(\vec{x}^{(i)})$ can remain unknown and any calculations made are simplified.

C. LSTM

LSTMs [16] are considered a great algorithm when working with timeseries data. Since there is a dependency between a stock's present price and yesterday's price our problem can be modelled as a time series problem. We try to tackle the problem both as a regression and a classification task.

First, we only feed the network with the price information. Sequences of the normalized close-prices are fed into the LSTM network during training while parameter tuning is driven by the validation loss. Final performance is evaluated on the last 391 dates, the test set. When doing regression we use Mean Squared Error Loss and performance is evaluated with MSE Loss, Pearson and Spearman [17] correlations. When switching to classification we use Binary Cross Entropy Loss and performance is evaluated with the accuracy metric.

Subsequently, we feed the LSTM network with the news sentiment scores along with the past prices and repeat the process above. Detailed results are presented in table II.

IV. Results

Training traditional machine learning algorithms, such as SVMs, on the news sentiment scores proves to be an effective technique offering satisfactory results. All classifiers performed better than a coin toss and than a dummy classifier that would always choose the most frequent class (54.7%). Detailed results for each algorithm are presented in table I. A support vector machine trained with a radial basis function (rbf) provides 59.132% accuracy offering effective predictions that could help somebody beat the market. Logistic Regression and 100-Nearest-Neighbors algorithms also provide strong results with 58.365% and 58.493% respectively. Lastly, Random Forest classifier with entropy criterion and 15 leaf nodes correctly predicts 57.599% of the 783 test dates. All classifiers are consistently better than pure luck, which could help someone in consistently beating the markets.

Table II sums up all the experiments ran with LSTM networks. Training an LSTM using only past closing price data does not provide enough information in order to extract any meaningful patterns when used as a classificator for the next day's price movement. It performs worse than a coin toss with a mere 44.91%. However, when used as a regressor the LSTM provides excellent correlation scores, meaning that an LSTM is able to follow prices effectively but not time price trend changes.

The inclusion of news' sentiment scores as an input feature in the LSTM seems to have very beneficial effects for classification. For the task of classification, news's sentiment provided about 9.2% increase in accuracy. This hints that news is a rich source of information for the classifier. However, regression results are more complicated. On the one hand Mean Squared Error has been substantially decreased, meaning that predicted values are closer to the real ones. On the other hand, Pearson correlation has been decreased by 2.7% and spearman is almost steady. It seems that for the task of regression, news' sentiment does not provide as much information.

V. Conclusions

In this paper we tested various methods of exploiting news' sentiment in order to make predictions on the S&P500 price movements. We evaluated and compared 4 different machine learning models and LSTM models for both classification and regression purposes.

Traditional machine learning classifiers like SVM and KNN trained on just the news sentiment score prove to be the most accurate technique on making predictions about the price movement of S&P500 index. SVM managed to reach an accuracy of 59.132%, way above a random guess. However, when trying to predict price movement based on past price data using an LSTM results are dissapointing, achieving worse than coin toss results. As stated by the "Efficient Market Hypothesis", a stock's current price reflects its fair market value at any time and so past prices do not offer any valuable information about future movements.

Overall, it seems that news and financial articles hide a lot of precious information, possibly because they affect traders'

TABLE I
ML Classifiers Performance Evaluation

Model	***Accuracy (%)***
Logistic Regr.	58.365
Nearest-Neighbors	58.493
SVM	59.132
Random Forest	57.599

TABLE II
LSTM Performance Evaluation

Model Type	**Metric**			
	MSE	***Spearman (%)***	***Pearson (%)***	***Accuracy (%)***
LSTM	16195.6	97.914	99.926	44.91
LSTM with news	12520.6	97.893	97.270	54.126

* Spearman and Pearson refer to the respective correlation measures

decisions in semi-predictable ways. Hence, leveraging news' sentiment as a supplementary tool to one's financial analysis and combining it with other Machine Learning classifiers can provide very promising results on price movement predictions.

REFERENCES

[1] Tandon, Kamini, and Nidhi Malhotra. "Determinants of stock prices: Empirical evidence from NSE 100 companies." International Journal of Research in Management Technology 3.3 (2013): 2249-9563.

[2] Oluseyi, ADENIJI Sesan. "An empirical investigation of the relationship between stock market prices volatility and macroeconomic variables' volatility in Nigeria." European Journal of Academic Essays 2.11 (2015): 1-12.

[3] Basilone, Ryan. "The Impact of News on Stock Market Investors." Available at SSRN 3908662 (2021).

[4] Boudoukh, Jacob, et al. Which news moves stock prices? A textual analysis. No. w18725. National Bureau of Economic Research, 2013.

[5] https://pypi.org/project/GoogleNews/

[6] https://github.com/michadenheijer/pynytimes

[7] https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests?select=raw_partner_headlines.csv

[8] https://pypi.org/project/yfinance

[9] Araci, Dogu. " FinBERT: Financial Sentiment Analysis with Pre-trained Language Models.

[10] Kalyani, Joshi, Prof Bharathi, and Prof Jyothi. "Stock trend prediction using news sentiment analysis." arXiv preprint arXiv:1607.01958 (2016).

[11] Li, Xiaodong, Pangjing Wu, and Wenpeng Wang. "Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong." Information Processing Management 57.5 (2020): 102212.

[12] Khedr, Ayman E., and Nagwa Yaseen. "Predicting stock market behavior using data mining technique and news sentiment analysis." International Journal of Intelligent Systems and Applications 9.7 (2017): 22.

[13] Prosky, Jordan, et al. "Sentiment predictability for stocks." arXiv preprint arXiv:1712.05785 (2017).

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

[15] Refaeilzadeh P., Tang L., Liu H. (2009) Cross-Validation. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_565

[16] Hochreiter, S. Schmidhuber, J"urgen, 1997. Long short-term memory. Neural computation, 9(8), pp.1735–1780.

[17] Spearman, C.: The Proof and Measurement of Association between Two Things. Am. J. Psychol. 15, 72–101 (1904)

[18] Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh (1992)

[19] Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)

[20] Cortes, C., Vapnik, V.: Support-vector network. Mach. Learn. 20, 273–297 (1995)