ADVANCED MACHINE LEARNING

WESTERN UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Stock Prediction Via Sentiment Analysis and Hybrid Deep Learning Models

---

Authors:
Sepehr Asgarian (Sasgaria@uwo.ca)
Rouzbeh Meshkinnejad (rmeshkin@uwo.ca)

Last Update: December 17, 2021

Contents

1 Introduction

In this project, we aim to enhance the prediction of stock market movements using sentiment analysis, as well as technical features directly extracted from stock prices. Sentiment analysis is the process of taking in textual data and determining the sentiment, opinion, or emotion. With the growth of social media and its popularity, there are increasingly more data that could be utilized for stock prediction. In order to effectively extract opinions and sentiments, we need intelligent natural language processing systems. Recently, with the introduction of transformer-based models [17], sentiments can be extracted more accurately than before, prompting us to make use of these models and the abundance of data to predict stock price movements.
We divide the effort in this project into four phases. In the first part, we aim to find as much textual data in the form of tweets, comments, etc., as possible. We then process, transform and structure this data so that our models can be trained on it. During the second phase, we make use of pre-trained language models to generate sentence-level embeddings for each of the samples in our dataset and save these embeddings on disk. In the third part, we train a simple but capable classifier that takes in the embeddings and predicts sentiments. As there are many tweets for a stock symbol on a particular day, we also aggregate the predicted sentiments to generate a single number indicating how positive the sentiment has been for that stock on that particular day. We save these predicted and aggregated sentiments for each stock symbol and day on disk. Finally, in the fourth phase, we extract price fluctuations for each stock symbol in each day and compute technical features. Appending the new technical features to the sentiment predictions, we then find the best features and train various hybrid deep learning models to take in these features for a window size before the current day and predict the stock price movement for tomorrow.

2 Related Work

While investment in financial markets is very attractive due to possibly high returns, uncertainty and fluctuations in prices pose a risk that leave investors unsettled. As a result, a model that reduces the risk and helps investors on when to buy, sell, or hold their shares, is considered to be invaluable. Consequently, research towards developing such a model has been on going for at least a decade, a complete summary of which is described in [15]. Here we will briefly touch on select works that we considered to be more relevant to the task in hand.
In [14], Nguyen et al. suggest a new method for prediction of stock prices. They use various predictors together with various sentiment analysis methods. Most notably, they use a topic distillation approach where they find a few topics in each sentence and predict sentiments for each of those topics instead of a single sentiment indicator for the whole sentence. In [9] Jeong et al. use a pipeline similar to our work that first filters

fake information, then assesses credit risk in a semi-supervised manner with sentiment analysis, detects risk signals using the obtained credit risk scores, and finally tries to predict movements and establish a strategy for investments.
As mentioned in [15] the accuracy of the machine learning model is highly dependent on the chosen algorithms, the quality of the data, and the infrastructure [2].
The prospects of analyzing sentiments have been looking much better with the emergence of large pre-trained transformer-based language models [17]. However, there have been drawbacks on using these models for analysis of stock related social media tweets. These limitations include the different vocabulary used in these tweets compared to what the language models have seen during training, limited access to accurate sentiment and opinion labels, and the computing resources needed to effectively run these language models.
More recently, in [8] Jaggi et al. try to solve the limitation of scarce sentiment labels by fine-tuning a pre-trained language model on tweets from stocktwits website. First they produce labels based on the prices changes in the same and next day of a tweet, and then try to classify them using their language model. This effort, however, did not provide a better performing language model as BERT [6] still showed to provide better results.
In this project, instead of fine-tuning a large language model, we use an existing language model to produce embeddings for a small-scale dataset of labelled tweets from Yahoo Finance Message Board, and train a simple classifier, to see whether this results in a better stock prediction accuracy.
Artificial intelligence has established a place in scientific communities for predicting similar time series problems. Artificial intelligence techniques effectively forecast many phenomena and occurrences, such as weather [13], and disease [18, 3].

3 Proposed Model

3.1 Sentiment Analysis

The process to extract sentiment from each tweet consists of two phases. First, we feed each tweet into BERT to get sentence-level embeddings. Next, we trained a classifier to predict the sentiments of tweets given their embeddings.

3.1.1 BERT

Among pre-trained language models that could fit in our GPU's memory, we chose BERT as it persistently outperformed other pre-trained models in [8]. Bidirectional Encoder Representations from Transformers, or BERT for short, is a language representation model introduced in 2018. Unlike language models before it, BERT learns deep representations from both left and right contexts at each layer. As for BERT's architecture, it

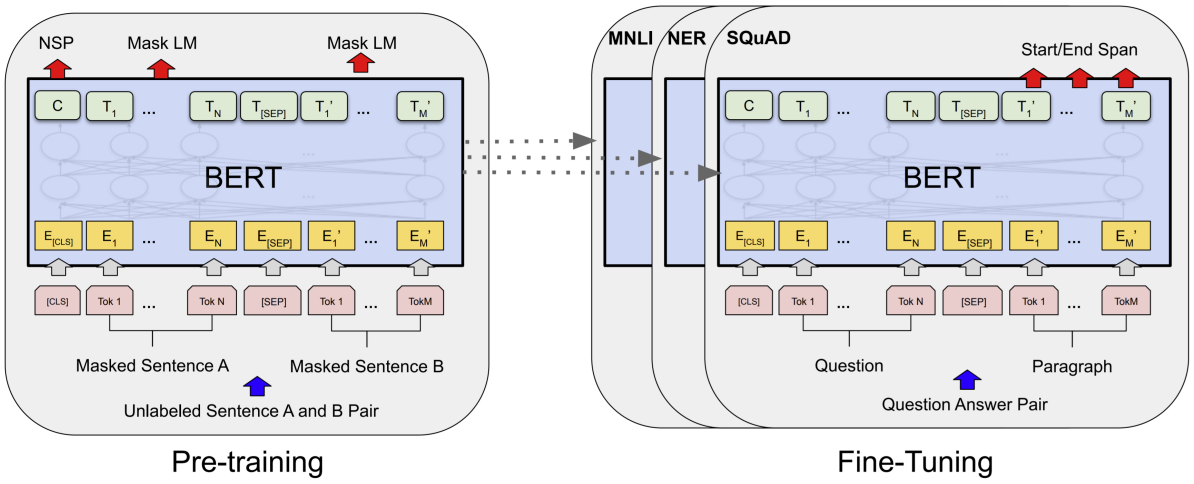

Figure 1: BERT pretraining and fine-Tuning [6]

is the same as the transformer architecture described in [17]. BERT is pretrained simultaneously on two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). After pretraining, BERT could be used for downstream tasks by introducing only a new additional layer. A diagram of BERT's pretraining and usage in downstream tasks can be seen in 1. In the MLM task, 15 percent of input tokens are randomly masked and the model is trained to predict masked tokens. Since downstream tasks do not have any special tokens called [MASK] in BERT's pretraining, each token chosen to be masked is masked only 80 percent of the time and remains unchanged 10 percent of the time, while it is replaced with a random token in the remaining 10 percent of the time. In the NSP task, two sentences A and B are fed to the model and the model is to output whether A actually follows B. 50 percent of the time B is the sentence that actually follows A, while the rest of the time B is randomly sampled from the sentence corpus. BERT is also fed in a special [CLS] token before any other token, output of which is interpreted as whether B is the next sentence that comes after A or not. As described in the original paper, the NSP task has been significantly helpful for representation learning.

3.1.2 Neural Network Classifier

After extracting sentence embeddings from our pre-trained language model, we train a neural network to predict each tweet's sentiment given its embedding.
The overall schematic of the classifier can be seen in Figure 2. The hyperparameters used for training are also described in Table 1. The classifier has two hidden layers, each of them consisting of 250 units. The activation for each layer is ReLU [12]. Dropout with rate = 0.5 is also applied in the second layer. The output of the second layer is given

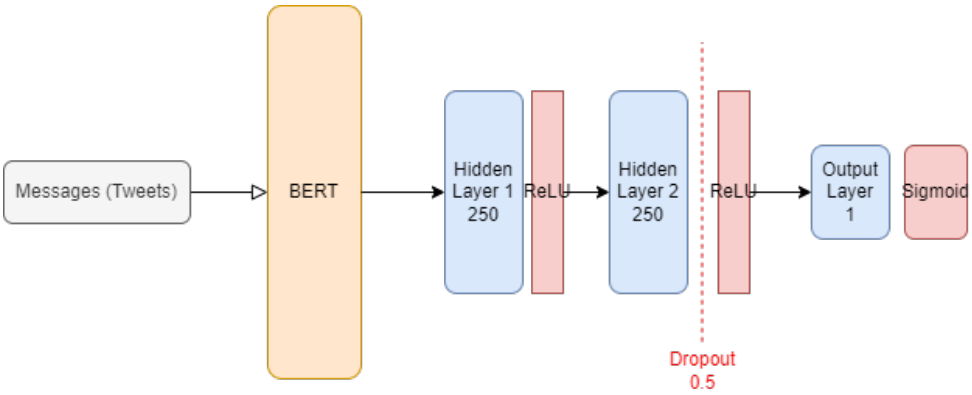

Figure 2: Sentiment Analysis model and process

to the output layer that predicts the probability of a tweet having a positive sentiment via a sigmoid activation. As we have only two labels (1 for positive sentiment and 0 for negative sentiment), we chose our loss function to be binary cross entropy loss.

3.2 Price Model

In this section, we first describe our preprocessing for price features, including how we develop our features, normalization, and feature extraction. Following that, we discuss two deep learning approaches, namely the Long Short Term Memory or LSTM for short [7] and then Convolutional Neural Networks (CNN) [11]. After that, in the following section, we will discuss our suggested hybrid architecture for combining the two methods.

3.2.1 Preprocessing for Price Features

The price dataset includes High, Low, Open, Close price, and volume attributes. To improve accuracy, we extracted technical and statistical features, normalized all the features, and used feature engineering to find the best features for each stock.

Developing Features Initially, Open, High, Low, and Close prices were employed in the data processing stage. Several additional features were developed using the most popular technical and statistical indicators: Simple and exponential moving averages, including several window sizes, moving average convergence/divergence (macd), relative strength index (rsi), stochastic, momentum, Bollinger band (containing high, low, and middle Bollinger bands), Ichi muko (containing tenkensen, kijun sen, senquan span a, senquan span b, and chikou-span), and volume. The obtained 21 features were then sent into the XGBoost algorithm, which extracted the best fitting and outstanding features for the hybrid deep learning model.

Normalization and Feature Engineering Due to the fact that the extracted features are not normalized and have a wide range of values, we modified the values of all features to be between 0 and 1. Moreover, Since all the created features are not helpful for the prediction, we have used the feature engineering technique to extract the best features. For instance, some developed features may be correlated and are not helpful for our task. The XGBoost algorithm aided in delicate feature extraction by providing for each feature a relevance score, allowing us to select the features according to their impact on a more accurate final prediction. Hence, for each specific stock, XGBoost chose the features that had the most prominent effect in prediction.

3.2.2 Deep Learning Model

In this section, we discuss deep learning techniques that attempt to predict the future using existing historical data. CNNs are the first of such techniques, while Recurrent Neural Networks (RNN) are the second approach that will be explained in this section. Following that, we discuss our suggested hybrid design for combining these two methods.

3.2.3 Convolutional Neural Network

CNN was offered for the first time by [11]. As a result of CNN's remarkable performance in image classification, the concept of utilizing it to forecast time series data developed. Several researchers have previously employed this approach. Unlike LSTM, CNN's input data cannot record extended data sequences because of the architectural differences. CNN, on the other hand, is more capable in extracting local features. Max-pooling is incorporated into the design to decrease the data's complexity/size.

3.2.4 Recurrent Neural Network

Recurrent neural networks (RNNs) are a type of artificial neural network whose decision relies on the concepts that they have understood before. They, however, suffer from the problem of vanishing or exploding gradients.
LSTM models, a special kind of RNNs, solve this problem and achieve better accuracy. LSTM is preferred to normal RNN since it has a memory cell with three gates that enables LSTM to model the past information better. LSTM is used in some prevalent problems such as stock prediction [4], text processing [16], and speech recognition [10]. This model and the underlying architecture was first proposed by [7]. Input gate, output gate, and forget gate are the foundational gates that construct an LSTM cell.
The following formulas describe the updating pattern of the cell state, which is based on the outputs of the gates, in a mathematical manner:

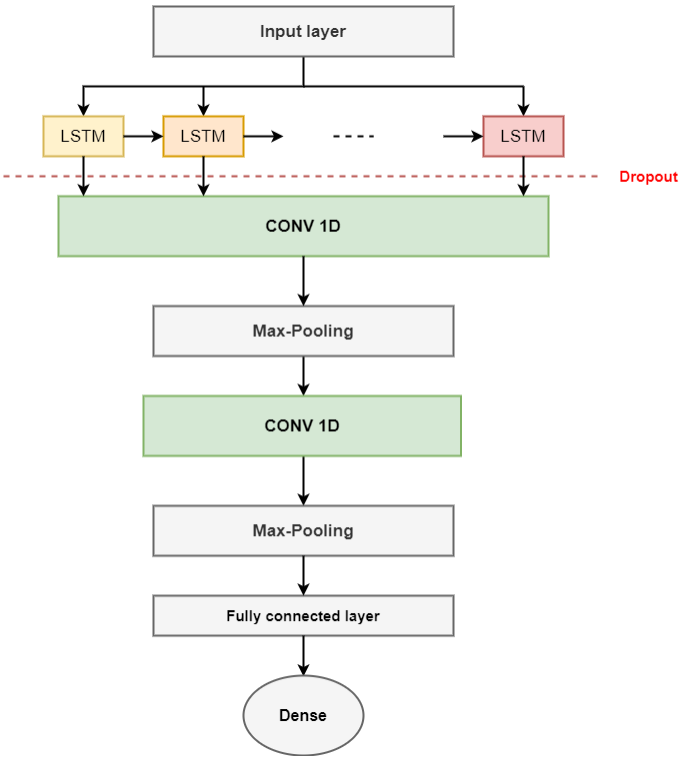

Figure 3: The architecture of the LSTM-CNN

$$
\begin{aligned}
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \\
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \sigma(W^{(c)}x_t + U^{(c)}h_{t-1} + b^{(c)})
\end{aligned}
\tag{1}
$$

In the above equations, x_t is the input vector, h_t is the output vector, c_t is the cell state vector, f_t is the forget gate vector, i_t is the input gate vector, o_t is the output gate vector, W and U are weight matrices, and b is the bias vector.

3.2.5 Hybrid Deep Learning model

Furthermore, we employed a hybrid model known as LSTM-CNN. This model is a combination of the previously described LSTM and CNN models in an attempt to improve performance. CNN layers are able to extract unique features, while LSTMs are capable of identifying excellent sequential features; when combined, both models can significantly enhance prediction performance. The architecture of the LSTM-CNN is illustrated in Figure 3. As can be seen, the time series data is received by the input LSTM layer

first in this chain of layers. The generated prediction, denoted by the variable f_{lstm}, is sent to two CNN layers through max-pooling. The result of this operation, f_{cnn}, goes through a fully connected layer and one dense layer at the end.

3.3 Methodology

This project was implemented in Python. Several libraries, such as NumPy, pandas, min-max scaler, XGBoost, Matplotlib, Keras, Tensorflow, Pytorch, and Huggingface were applied. Table 1 summarizes the chosen parameters.
The sentiment analysis classifier was implemented in PyTorch. We used the uncased version of BERT-BASE from huggingface library [1] to generate embeddings. BERT and classifier are used separately: we generate embeddings using BERT in eval mode and save them on disk. For training the classifier, we load the embeddings from disk and feed them to the neural network. As a result, BERT is not fine-tuned for the sentiment analysis task. We use 90 percent of the sentiment analysis data for training and the remaining 10 percent for testing.
One of the approaches taken in this project is to demonstrate the value of sentiment analysis in stock prediction. As a result, we performed two experiments to predict stocks. In the first experiment, as illustrated in 5, we utilized our suggested model without sentiment analysis; in the second experiment, as illustrated in 4, we added the sentiment analysis feature to the best features generated by feature engineering.

3.3.1 Price Sentiment Hybrid model

To begin, we created technical and statistical features for our model using real-time stock price data. Following that, we normalized features and applied feature engineering to identify the best features. The derived features were then appended to the extracted sentiment analysis information. Finally, we sent this data into our deep hybrid model as input. The proposed model is shown in Figure 4.

3.3.2 Price Hybrid model

As shown in Figure 5 this model is without sentiment analysis.

4 Evaluation

4.1 Dataset

In this project, two sets of datasets are used. The first set consists of the data that we have used for sentiment analysis. The second set includes the stock market price data.

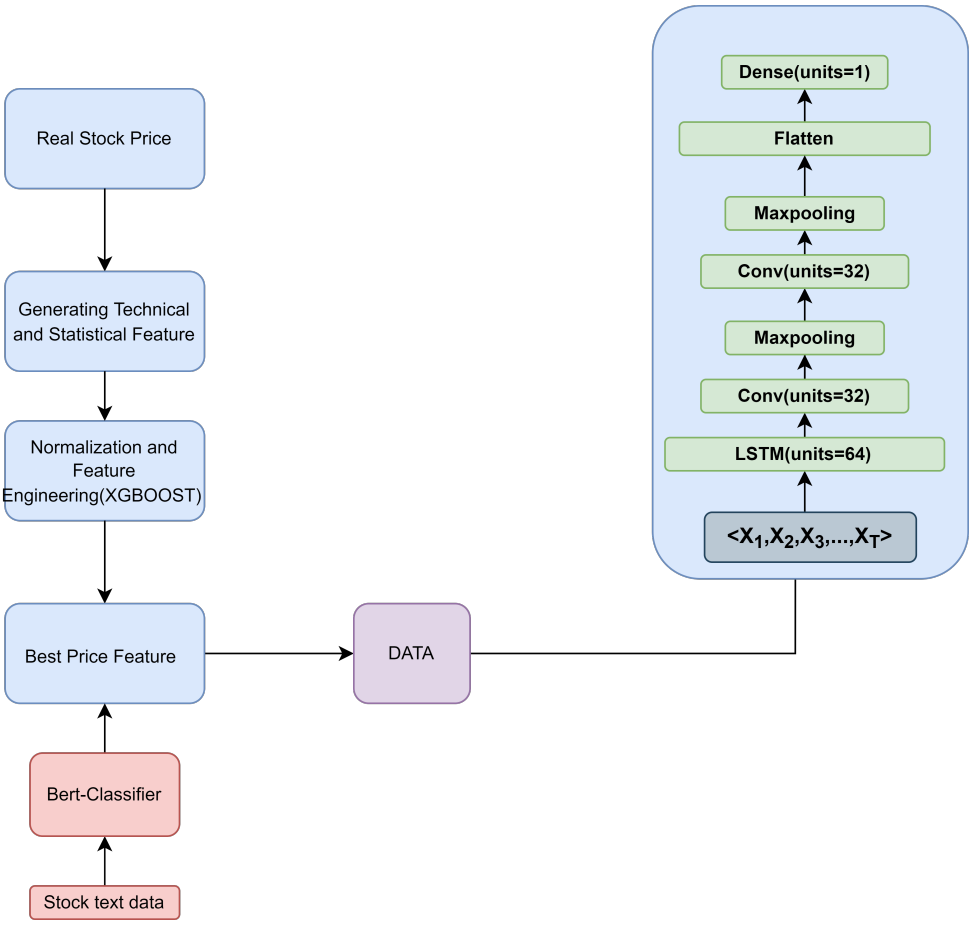

Figure 4: The architecture of the Price Sentiment Hybrid model

4.1.1 Sentiment Analysis Dataset

As training the hybrid deep learning model with sentiments needed at least a few years of financial data and tweets, we started looking for a suitable dataset. At first, we found the dataset used in [8]. However, it was removed by the time we wanted to start implementing the models. We asked the authors to make the dataset available again and, in the meantime, contacted stocktwits.com directly. We did not hear back from stocktwits.com, but the authors of [5] generously provided us with their dataset. We will call this dataset the StockTwits dataset. This dataset contains more than 6 million tweets from the stocktwits.com website for 24 symbols, including TSLA, AAPL, NFLX, QQQ, AMZN, DIS, DIA, VIX, BAC, SPY, GOOG, T, HD, V, VZ, JNJ, ADBE, BRK.B, PG, UNH, INTC, and WMT. Each sample in this dataset has information regarding its symbol, message, date, time, userid, and label. A complete description of this dataset is available in Table 2.

As the labels in the StockTwits dataset were generated using price movement of the

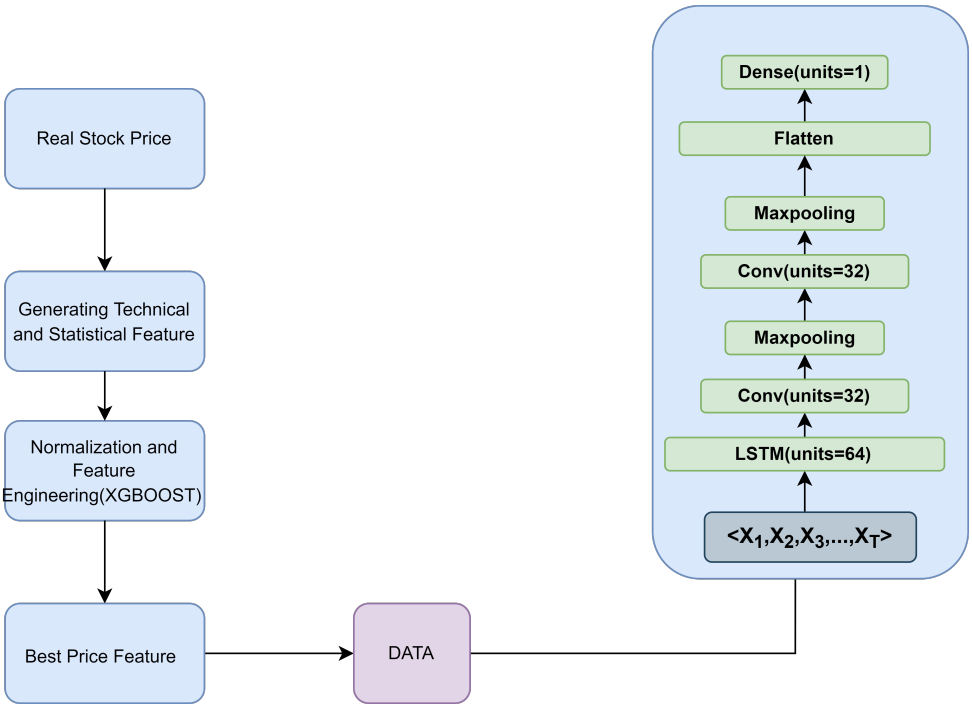

Figure 5: The architecture of the Price Hybrid model

same day or the next day, we concluded the labels are not accurate enough to train our classifier on. As a result, we started looking for another dataset and found a study for sentiment analysis [5]. The authors of this study also generously provided a dataset we will call YahooFinanceDataset. This dataset was originally in XML format. In order to use it more easily, we first converted this dataset to XML format and discarded the parts we did not need (thumbs up attribute of each message, userid, etc.) The YahooFinance-Dataset contained more than a million tweets from symbols AMZN, BA, BAC, CSCO, EBAY, IBM, GOOG, INTC, YAHOO, MSFT, NVDA, ORCL, T, XOM. The information for each sample was the message data, date, time, and label. The label attribute of each sample could hold any of the following values: Strong Buy, Buy, Hold, Sell, Strong Sell, and there were some samples without a label. We discarded those samples with labels equal to Hold and those that did not have a label. We also replaced Strong Buy and Buy with 1 and Sell and Strong Sell with 0 to simplify labels further. A more specific description of this dataset can be seen in Table 3.

We trained our classifier on the YahooFinanceDataset. We then applied the model to the larger StockTwitsDataset to generate sentiments for each sample. We finally generated aggregate sentiments for each symbol in each day by taking the mean of each sample's sentiment for that symbol and day. In the subsequent sections, we refer to this newly generated dataset as SentimentDataset.

Models	Batch Size	Epoch	Hyperparameters
LSTM-CNN	128	200	L1: LSTM Units=64 Dropout rate= 0.2 L2: Filters= 32 Kernel size= 4 pool size= 3 L3: Filters= 32 Kernel size= 3 pool size= 2 L4: Dense= 1 Optimizer: Adam
BERT-NN Classifier	128	100	L1: Neural Network units= 250 L2: Neural Network units= 250 Activation= ReLU Dropout rate= 0.5 Learning rate= 1e-3 Optimizer: Adam

Table 1: Parameter setting for the hybrid model and BERT-NN Classifier for sentiment analysis (L stands for the layer.)

4.1.2 Price Dataset

We have used Yahoo finance public API(https://ca.finance.yahoo.com) for getting price data. Based on the predicted SentimentDataset from the last section, the start date is 2013-01-01, and the end date is 2020-07-21. In the next step, we have merged the data predicted in the sentiment analysis step with the price features generated after the feature engineering part. Finally, we divided our dataset into 70 percent train data, 20 percent test data, and 10 percent validation.

4.2 Evaluation Metrics

As mentioned before, for training the sentiment analysis classifier, labels were either 1 for positive sentiment or 0 for negative sentiment. We trained the classifier using the binary cross entropy loss, formula of which is:

$$BCE = -\sum_{\text{samples}} y_i log(p_i) + (1 - y_i) log(1 - p_i) \tag{2}$$

where ys are ground truth labels and ps are predicted probabilities.
The loss function for our hybrid deep learning model is the Mean Squared Error (MSE), which is defined as the sum of squared errors. The formula for MSE is as follows:

Stock	Stock name	Number of tweets
TSLA	Tesla Inc.	1798534
AAPL	Apple Inc.	1231448
NFLX	Netflix Inc.	666001
QQQ	Invesco QQQ Trust Series 1	592253
AMZN	Amazon.com Inc.	450379
DIS	Walt Disney Co.	245472
DIA	SPDR Dow Jones Industrial Average ETF Trust	188598
VIX	CBOE Volatility Index	188401
BAC	Bank of America Corp.	164952
SPY	SPDR S&P 500 ETF Trust	151984
GOOG	Alphabet Inc. CI C	135365
FB	Meta Platforms Inc.	111100
GOOGL	Alphabet Inc. CI A	107908
T	AT&T Inc.	70091
HD	Home Depot Inc.	52425
V	Visa Inc. CI A	50100
VZ	Verizon Communications Inc.	49403
JNJ	Johnson & Johnson	38931
ADBE	Adobe Inc.	36582
BRK.B	Berkshire Hathaway Inc. CI B	33550
PG	Procter & Gamble Co.	31413
UNH	UnitedHealth Group Inc.	16049
INTC	Intel Corp.	15328
WMT	Walmart Inc.	13513

Table 2: Statistics of the StockTwits dataset [8]

$$MSE = \frac{1}{n}\sum (y - \hat{y})^2 \tag{3}$$

Where $(y - \hat{y})^2$ is the squared difference between the real and the predicted values.
However, because this task is a classification problem, once the model has been trained, each transaction day assigns an up or down label depending on the difference between the current day's price and the previous day. As a result, all expected prices are assigned either a label of 1 or 0. Accuracy(ACC) is the evaluation metric that is used. The Accuracy measurement is calculated as follows:

$$Accuracy = \frac{t_p + t_n}{(t_p + t_n + f_p + f_n)} \tag{4}$$

Where t_p indicates true positive, t_n true negative, f_p false positive, and f_n false negative

Stock	Stock name	Min	Median	Mean	Max	# texts (mean)
AMZN	Amazon.com Inc.	24	154	192	1 963	28
BA	The Boeing Company	46	173	203	1 053	16
BAC	Bank of America Corporation	94	282	343	1 366	49
CSCO	Cisco Systems Inc.	69	247	274	972	10
EBAY	eBay Inc.	1	17	29	267	3
IBM	IBM Inc.	3	14	20	195	3
GOOG	Google Inc.	10	69	93	1 305	16
INTC	Intel Corporation Corporation.	37	177	200	958	29
YAHOO	Yahoo! Inc. Corporation.	22	121	141	860	27
MSFT	Microsoft Corporation.	27	139	172	815	16
NVDA	NVIDIA Corporation.	10	65	80	410	11
ORCL	Oracle Corporation.	5	67	79	372	6
T	AT & T Inc.	10	52	59	251	8
XOM	Exxon Mobil Corporation.	10	37	44	202	4

Table 3: Statistics of the YahooFinanceDataset for each transaction date [5]

5 Results

5.1 Sentiment Analysis Result

After successfully training our classifier on the YahooFinanceDataset, we evaluated the accuracy. We achieved training set accuracy of 97.91 percent and test set accuracy of 88.68 percent.

5.2 Price Prediction Result

Table 4 summarises the accuracy of the two proposed models. Two models are compared in this table. The overall average accuracy of the Price Sentiment Hybrid model across all stocks is 54.3 percent, while the Price Hybrid model is 51.03 percent. Additionally, as can be seen, the results for these individual companies are very promising; VIX, PG, and QQQ stocks all have an accuracy of greater than 58 percent. Furthermore, the overall average of the Price Sentiment Hybrid model is 3.3 percent more than the Price Hybrid method. The Price Sentiment Hybrid model performs better in 21 out of 24 stocks than the Price Hybrid model. This can be a primary reason why textual information on social media can help improve stock market prediction.

Stock	Price Hybrid	Price Sentiment Hybrid
AAPL	0.523	**0.579**
ADBE	0.547	**0.566**
BAC	0.478	**0.506**
AMZN	0.503	**0.524**
DIA	0.503	**0.506**
DIS	0.48	**0.506**
BRK.B	0.523	**0.558**
FB	0.468	**0.500**
GOOG	0.468	**0.549**
HD	0.512	**0.515**
INTC	0.514	**0.559**
JNJ	0.466	**0.509**
PG	0.597	**0.632**
QQQ	**0.626**	0.603
SPY	0.528	**0.534**
T	**0.594**	0.587
TSLA	0.519	**0.567**
NFLX	0.486	**0.511**
UNH	0.461	**0.506**
VIX	0.521	**0.588**
V	0.513	**0.543**
WMT	0.484	**0.547**
VZ	**0.481**	0.476
GOOGL	0.524	**0.551**
Average Accuracy	0.513	**0.543**

Table 4: Results of accuracies of stocks for Price Hybrid and Price Sentiment Hybrid model

6 Conclusion

In recent years, both academics and industry have been interested in forecasting stock markets using sentiment analysis, machine learning, and deep learning models. This project presents a hybrid model that uses both LSTM and CNN. In this study, the proposed model is assessed using two distinct types of inputs. The first is a Price Hybrid model that utilizes only optimized price features, while the second is a Price Sentiment Hybrid model that incorporates optimized price features and sentiment features. These approaches are used to a total of 24 stocks in this project. Additionally, these two approaches are contrasted. Our experiments demonstrate the Price Sentiment Hybrid model's advantage over the Price Hybrid model. In general, our findings indicate that

sentiment analysis helped the Hybrid model learn and forecast more accurately in the stock market.

References

[1] Huggingface bert webpage. pages 9

[2] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020. pages 4

[3] Hossein Bonakdari, Jean-Pierre Pelletier, and Johanne Martel-Pelletier. A reliable time-series method for predicting arthritic disease outcomes: New step from regression toward a nonlinear artificial intelligence method. *Computer methods and programs in biomedicine*, 189:105315, 2020. pages 4

[4] Eunsuk Chong, Chulwoo Han, and Frank C Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017. pages 7

[5] Ali Derakhshan and Hamid Beigy. Sentiment analysis on stock social media for stock price movement prediction. *Engineering applications of artificial intelligence*, 85:569–578, 2019. pages 10, 11, 14

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. pages 4, 5

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. pages 6, 7

[8] Mukul Jaggi, Priyanka Mandal, Shreya Narang, Usman Naseem, and Matloob Khushi. Text mining of stocktwits data for predicting stock prices. *Applied System Innovation*, 4(1):13, 2021. pages 4, 10, 13

[9] Yujin Jeong, Sunhye Kim, and Byungun Yoon. An algorithm for supporting decision making in stock investment through opinion mining and machine learning. In *2018 Portland International Conference on Management of Engineering and Technology (PICMET)*, pages 1–10. IEEE, 2018. pages 3

[10] Max WY Lam, Xie Chen, Shoukang Hu, Jianwei Yu, Xunying Liu, and Helen Meng. Gaussian process lstm recurrent neural network language models for speech recognition. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 7235–7239. IEEE, 2019. pages 7

[11] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. pages 6, 7

[12] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. pages 5

[13] Sravani Nalluri, Somula Ramasubbareddy, and G Kannayaram. Weather prediction using clustering strategies in machine learning. *Journal of Computational and Theoretical Nanoscience*, 16(5-6):1977–1981, 2019. pages 4

[14] Thien Hai Nguyen, Kiyoaki Shirai, and Julien Velcin. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, 42(24):9603–9611, 2015. pages 3

[15] Mehtabhorn Obthong, Nongnuch Tantisantiwong, Watthanasak Jeamwatthanachai, and Gary Wills. A survey on machine learning for stock price prediction: algorithms and techniques. 2020. pages 3, 4

[16] K Sangeetha and D Prabha. Sentiment analysis of student feedback using multi-head attention fusion model of word and context embedding for lstm. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–10, 2020. pages 7

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. pages 3, 4, 5

[18] G Wang, W Wei, J Jiang, C Ning, H Chen, J Huang, B Liang, N Zang, Y Liao, R Chen, et al. Application of a long short-term memory neural network: a burgeoning method of deep learning in forecasting hiv incidence in guangxi, china. *Epidemiology & Infection*, 147, 2019. pages 4