Forecasting Stock Prices using Tweets

Jiacheng Wang

Data Science and Business Analytics Dept. HEC Montreal, Canada

E-mail:jiacheng.wang@hec.ca

Abstract. Stock market price prediction is a challenging problem since the market is an immensely complex, stochastic and dynamic environment. There are many studies from various areas aiming to improve the performance of prediction and analysis of public emotion has been the focus of one of them. We use information shared over Kaggle, an online community of data scientists and machine learning practitioners, to better understand and predict stock prices of Tesla. This article studies the methods to preprocess tweets and to tune models so that neural network models and linear regression can adapt to the preprocessed tweets. According to previous authors, one way to preprocess tweets is to keep the tweets of smart user, and output can be the next-day close prices or the next-day return. For that goal, prediction models (CNN-LSTM, LSTM and linear regression) were built and modified step by step and their results were analyzed by Mean Square Error and Mean Absolute Error. Finally, the LSTM model, with close value and weighted labels as input features and return as output feature, wins the prediction of stock price of Tesla among other candidate models with different input and output features.

Keywords: CNN-LSTM, LSTM, FinBERT, linear regression, twitter, smart user, TF-IDF, sentiment analysis

1 Introduction

Stock price predictions have been a study field of high interest to investors as it presents them with an opportunity to benefit financially by investing their resources in shares and derivatives of various companies. However, the movement of a stock price is a chaotic system; meaning the behavioural traits of share prices are unpredictable and uncertain. To make some sort of sense of this chaotic behaviour, some researchers focused on finding good factors relating to financial knowledge to explain the stock movement, such as the Fama French model which is a financial model that calculates the expected rate of return for an asset or investment. Some researchers such as Li Bing et al. (2014) focused on using sentiment analysis to explain the effect of public emotion on stock price prediction[7]. Moreover, Yumo Xu et al. (2018) introduced a new neural network model called StockNet which was used to predict stock movement with a high accuracy of 58.23%[18].

More specifically, the method using sentiment analysis, one popular approach, proposed by Johan Bollen et al. (2011), is based on neural networks to learn from tweets in score form processed by sentiment analysis tools and historical stock data to predict future prices[2]. And another method, used by Scott Coyne et al. (2017), is using linear regression to learn the effect of each word in the form of a TF-IDF vector[4], which will bed explained in detail later in Section 3. This paper goes in that direction in studying a method to improve the performance of prediction by using tweets and evaluating their performances.

The first question is "how to find smart users?". Scott Coyne et al. (2017) suggest manually labelling tweets as bullish, bearish, or neutral[4]. After removing the neutral class, they compared the sentiment to how the stock moved that day to generate each user's accuracy.

The second question is "how to use public emotion as input?". Scott Coyne et al. (2017) concatenate all smart tweets per day for each stock and vectorize daily smart tweets by term frequency–inverse document frequency (TF-IDF), a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus, which is used as input features of linear regression[4]. Johan Bollen et al. (2011) convert daily tweets to score form by Opinion Finder or Google-Profile of Mood States, which is a psychological rating scale used to assess transient, distinct mood states and use the score as the input of a Self-organizing Fuzzy Neural Network model[2].

In this paper, I contend that Scott, Praveen and Joseph were correct in the methodology of finding smart users and the method of using TF-IDF vectors as input to predict return by linear regression and that Johan, Huina and Xiaojun were correct in the way of using sentiment scores. I will first focus on the possibility to reduce the time to find smart users by using machine learning algorithms, in order words, automatically labelling all tweets and then test whether by using the adjusted daily sentiment score of smart tweets as one of the input features can improve the performance of the model with all tweets as one of the input feature for the prediction of close price of the stock of Tesla.

The remainder of this paper is organized as follows. Section 2 provides a literature review to summarize related works and Section 3 discusses preliminaries and tools we use moving forward. This leads into Section 4 where we discuss the result of our methods and the effect of using smart users' tweets in prediction. Finally, Section 5 provides our conclusion and ideas for future work.

2 Literature Review

In the field of stock prediction, researchers are mainly interested in two types of predictions, one is stock movement prediction and the other one is stock value prediction. However, the approaches to achieve the two types of predictions has some common feature. Specifically, to improve the performance of prediction, researchers focus on different aspects such as data processing and machine learning methods.

Sima Siami-Namini et al. (2018) made a comparison of a rolling Autoregressive Integrated Moving Average model (ARIMA) and a rolling Long Short-Term Memory (LSTM) in Forecasting a historical monthly financial time series from Jan 1985 to Aug

2018 from the Yahoo finance Website[14]. The input and output of the experiment are Adjusted Close of stock indexes for LSTM and ARIMA. The authors' ARIMA model is built with p (lag order) =5, d (degree of differencing) =1 and q (order of moving average) =0, without seasonality. For LSTM, the mean squared error and ADAM are used as the loss function and the optimization algorithm, respectively. The authors mentioned their batch size is 1 and the ratio of the size of the training set to validating set is 7:3, but no details of the architecture of LSTM such as batch size, activation function, the range of the date of each set, etc. The monthly data included Nikkei 225 index (N225), NASDAQ composite index (IXIC), Hang Seng Index (HIS), S&P 500 commodity price index (GSPC) and Dow Jones industrial average index (DJ). The assessment metric they used was Root-Mean-Square Error. They concluded that the LSTM-based algorithm improved the prediction by 85% on the average price of these stock indexes compared to ARIMA.

David M. Q. Nelson et al. (2017) applied LSTM neural networks to predict the price movement of five stocks that are part of the IBovespa index from the BM&F Bovespa stock exchange[10]. Except that the target of prediction become price movement, the input of the LSTM model was also modified. First, a log-return transformation was performed as means of normalization as well as to stabilize the mean and variance along the time series[3]. Secondly, exponential smoothing was performed on stock price to reduce random variation and noise. Finally, with the Technical Analysis Library (TA-Lib library), which is widely used by trading software developers required to perform technical analysis of financial market data, price data was used to generate 175 technical indicators for each period so that there are 180 input features for each period. For training, it is used the last 10 months of trading before the current day, and the model performance is validated by using the data of the past week. On the following day, all the predictions will be done using the most recent model. The assessment metrics were accuracy, precision, recall, F-measure (a harmonic mean between precision and recall), rate of return, etc. They conclude that there is a significant improvement in terms of accuracy when comparing LSTM to Pseudo-random, which predicts stock movement based on probabilities according to the class distribution, in five datasets of stocks, but the accuracy of LSTM is only larger than that of random forest in 2 datasets out of 5.

Wenjie Lu et al. (2020) used a Convolutional Neural Networks-Long Short-Term Memory-based model (CNN-LSTM-based model) to forecast the close value of the Shanghai Composite Index with a dataset from July 1, 1991, to August 31, 2020[8]. Each piece of data contains the opening price, highest price, lowest price, closing price, volume, turnover, ups and downs, and change. In this model, "CNN is used to extract the time feature of data, and LSTM is used for data forecasting". The author concluded CNN-LSTM is more suitable for stock price forecasting than multilayer perceptron (MLP), CNN, Recurrent Neural Networks (RNN), LSTM and CNN-RNN. Specifically, the prediction of the CNN-LSTM model has the smallest MAE (27.56) and RMSE (39.69) and has the largest R square (0.96).

Anshul Mittal et al. (2011) used Twitter sentiment analysis to predict stock value, it raised the question that is how to process tweets[9]. In Anshul et al.'s paper, they mentioned that some sentiment analysis tools such as OpinionFinder and SentiWordnet are inadequate and/or inefficient, because “this may however ignore the rich, multi-dimensional structure of human mood” indicated by Johan Bollen et al. (2011)[2]. Thus, they used a methodology relating to the Profile of Mood States questionnaire. In this methodology, the authors first establish a word list where each word is mapped on one of the six standard Profile of Mood States (POMS) moods. Based on the authors' equation, they then calculated the score of each word in a day and use the scores of words to generate the scores of moods. However, we cannot get access to the authors' word list and the algorithm and code of POMS are not publicly available online.

Many sentiment analysis tools can detect the polarity (i.e. positivity or negativity) of a short text. Filipe N Ribeiro et al. (2016) compared several sentiment analysis methods, where we can see that SentiStrength is the best method among 11 out of 18 datasets with 2 classes and that Umigon is the best method among 6 datasets among 13 datasets with 3 classes[12].

However, the sentiment analysis tools above are not designed for financial datasets. Dogu Tan Araci et al. (2019) introduced a pre-trained NLP model called FinBERT to analyze the sentiment of financial texts[1]. It is built by further training the BERT language model in the finance domain, using a large financial corpus and thereby fine-tuning it for financial sentiment classification. They achieved state-of-the-art result on FiQA sentiment scoring and Financial PhraseBank. Moreover, for the classification task, they increased the state-of-the-art accuracy by 15%.

Yumo Xu and Shay B. Cohen (2018) developed a method called StockNet to predict the movement of 88 stocks from 9 industries (Basic Materials, Consumer Goods, Healthcare, Services, Utilities, Conglomerates, Financial, Industrial Goods and Technology) from January 1st, 2014 to January 1st, 2016[18]. The input features are the Twitter dataset and a historical price dataset. The model comprises three primary components which were Market Information Encoder (MIE), Variational Movement Decoder (VMD) and Attentive Temporal Auxiliary (ATA). In detail, MIE encodes information from social media and stock prices to enhance market information quality and outputs the market information input X for VMD, VMD infers and decodes the latent driven factor Z and the movement y from the encoded market information X, ATA integrates temporal loss through an attention mechanism for model training. Finally, the authors concluded that StockNet performs better than a set of baselines, such as ARIMA and Random Forest on a comprehensive dataset.

Scott Coyne et al. (2017) focused on the idea that the prediction based on tweets of smart users can be better than that of all users[4]. The authors first labelled every twit as bullish, bearish, or neutral, then use a multilayer perceptron classifier to predict labels. Then they removed the neutral class in the forecasted labels, and they compared the forecasted labels and how the stock moved to generate accuracy for each user. The users who had at least 80% of posts accurate were labelled smart users. Then the tweets of the smart users were converted into TF-IDF form to predict the return, of each stock, which was converted into price movement. The assessment metric is the accuracy of price movement and the accuracies of the method using smart users of stock movement of 8 out of 8 stocks are higher than that of the method using all users. However, filtering smart users will cause a problem from a small sample, where small sample sizes can cause bias. To overcome the bias caused by a small sample, the authors use random train-test splits and a high number of runs. Specifically, "for each stock, a random hundred days of the year were chosen for testing days and the rest for training. Training days were used to identify smart users, and then we looked at smart user postings over the training set. This random split and test were performed twenty-five times per stock, and we averaged results to calculate accuracy for each".

3 Literature Review

3.1 Data-Preprocessing

Data.

In this experiment, Tesla stock value[17] and Tweets[15] about Tesla are selected as the experimental data. Data from 720 trading days from January 1, 2017, to December 31, 2018, are obtained from Kaggle. Each piece of stock value data contains five items, namely, opening price, highest price, lowest price, closing price and volume. In the dataset of Tweets, I take the data of the first 438 trading days as the training set and the data of the next 146 trading days as validating set and the data of the last 146 trading days as the testing set. The time step of each set will be set to be a number N, in other words, the data of N consecutive trading days will be used to forecast the close value at the N+1 day. If the sequence length is less than N days, padding will be applied to maintain the sequence length by adding multiple first observations of the sequence at beginning of the sequence. The parameter dictionary for each neural network model is the one performing with the best MAE in validating set, and these models are used to predict the next-day close value of the stock of Tesla in the testing set.".

Historic data preprocessing.

a) Sentences Preprocessing.

I lower all letters, remove all web links, tickers(@ and $), digits, symbols and punctuations, and retrieve some abbreviations(n't to not, etc.). Finally, I drop duplicate sentences

b) Filtering Tweets.

The number of preprocessed tweets relating to Tesla from 2017 to 2018 is 415398. It takes about 1 second to label 10 tweets by FinBert and 12 hours to label 415398 tweets, which costs too much time. Thus, I keep tweets which has more than 1 comment, 1 retweet and 1 like, so that the number of filtered tweets is 51232. Table I is an example of the tweets being kept.

Table 1. A example of keeping tweets

	Comment number	**Comment number**	**Comment number**	**Is kept?**
Tweet 1	*1*	*1*	*1*	Yes
Tweet 2	1	1	0	No

c) Weighted Labels and Scores.

To generate daily emotion labels and scores, we should generate the weight, weighted score and weighted label for each tweet in order. Then generate daily weighted scores and labels by weighted scores and labels of each tweet. The formulas to get daily weighted scores and labels are shown below:

$i: tweet\ i$

$weights_i = comment\ number_i + retweets\ number_i + like\ number_i$

$weighted\ scores_t = E[score_i \times weights_i\ ], for\ day, where\ E[X]\ \ is\ the\ mean\ of\ variable\ of\ X$

$$weighted\ labels\_t = E[label\_i \times weights\_i\ ], for\ day\ t$$

d) Target variable (Price or Return).

If the target variable is close price, standardization will be applied to it which will be described in the next section. If it is return, then $eturn\_t = (P_t - P_{(t-1)})/P_{(t-1)}$, where P_t is the close price of the stock of Tesla at time t.

e) Standardization on predictor variables.

$X_t = \frac{X_t - \bar{X}}{std(X)} \quad \forall\ t, where\ std\ means\ standard\ deviation$

Standardization is performed on each predictor variable to make them contribute equally and to help the training of our neural networks as the different features are on a similar scale, which helps to stabilize the gradient descent step, allowing us to use larger learning rates or help models converge faster for a given learning rate.

f) Technical Analysis Library.

Technical Analysis Library (TA-Lib) is a market analysis tool which includes 200 functions used to generate technical indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands etc. Himanshu Sharma (2023) wrote a blog named Exploring the Best Indicators in TA-Lib: Technical Analysis of Stocks using Python- Part 1 , and mentioned that the effectiveness of these indicators may vary depending on the asset and market conditions, so I randomly choose the following 5 indicators, mentioned by Himanshu Sharma, which are the 5-day Exponential Moving Average (ema), the 5-day Simple Moving Average(sma), the 20-day Bollinger Bands(low band, mid band, up band), the 14-day Relative Strength Index(rsi) and the 20-day Average Directional

Movement Index (adx) as inputs of neural network models by TA-Lib[13]. Their formulas are shown below:

$\mathrm{sma}_t(\mathrm{x}, \mathrm{N}) = \frac{\sum_{i=1}^{N} x_{t-i+1}}{N}$, where x is the sequence on which the simple moving average algorithm is applied, and N is the number of days in the smoothing period.

$\mathrm{ema}_t(\mathrm{x}, \mathrm{N}) = \left(x_t - \mathrm{ema}_{t-1}(\mathrm{x}, \mathrm{N})\right) \times \frac{2}{N+1} + \mathrm{ema}_{t-1}(\mathrm{x}, \mathrm{N})$, where x is the sequence on which the exponential moving average algorithm is applied, and N is the number of days in the smoothing period.

$$low\ band_t(N) = sma_t(TP, N) - 2 \times \sigma_t(TP, N)$$

$$mid\ band_t(N) = sma_t(TP, N)$$

$$up\ band_t(N) = sma_t(TP, N) + 2 \times \sigma_t(TP, N)$$

where $\mathrm{TP}_t = \frac{\text{High value}_t + \text{Low value}_t + \text{Close value}_t}{3}$, and $\sigma_t(\mathrm{TP}, \mathrm{N})$ is the standard deviation of the sequence $\mathrm{TP}_{t-N+1}, \dots, \mathrm{TP}_t$, and N is the number of days in the smoothing period.

$rsi_t(x, N) = 100 - \frac{100}{1+RS_t(x,N)}$, where $RS_t(x, N) = \frac{\sum_{i=1}^{N} 1_{[change_t(x)>0]}}{\sum_{i=1}^{N} 1_{[chang\ \ _t(x)<0]}}$ and $hange_t(x) = \frac{x_t - x_{t-1}}{x_{t-1}}$.

$adm_t(N) = \frac{adm_{t-1}(N) \times (N-1) + DX_t(N)}{N}$, where $DX_t(N) = \frac{|DI_{+t}(N) - DI_{-t}(N)|}{|DI_{+t}(N) + DI_{-t}(N)|} \times 100$

$$DI_{+t}(N) = 100 \times \frac{sma_t(DM_+, N)}{ATR_t}$$

$$DI_{-t}(N) = 100 \times \frac{sma_t(DM_-, N)}{ATR}$$

N is the number of days in the smoothing period

ATR_t: *Average True Range at time t*

$$DM_{+t} = Max(UpMove_t, 0)\ if\ UpMove_t > DownMove_t$$

$$DM_{-t} = Max(DownMove_t, 0)\ if\ DownMove_t > UpMove_t$$

$$UpMove_t = High_t - High_{t-1}$$

$$DownMove_t = Low_{t-1} - Low_t$$

$High_t$: *High value at time t*

Low_t: *Low value at time t*

TF-IDF.

TF-IDF is a statistic which evaluates how relevant a word is to its body of text. It is done by multiplying two metrics, which are term frequency and inverse document frequency. Its formula is shown below[16]:

$$tf(t,d) = \log\big(1 + freq(t,d)\big)$$

$$idf(t,D) = \log\left(\frac{N}{count(d \in D: t \in d)}\right)$$

$$tf\ idf(t,d,D) = tf(t,d) \times idf(t,D)$$ [1]

where t is the index of a word, d is the index of a document, and D is a collection of all documents.

In the formula, the TF score is larger if the frequency of the word t in the document (tweets at one day) d is higher, while the IDF score is larger if the number of documents (tweets at one day) d including word t in document set(tweets all times) D is small compared to the total number of document (tweets at one day) d in document set (tweets all times) D. Scott Coyne et al. (2017) used TF-IDF as a method of pre-processing of text[4]. They mentioned that TF-IDF can eliminate stop words automatically. Finally, the vectorized tweets will be used as the input of a machine learning method to predict the percentage change of stock value.

Smart users.

In the same paper, Scott Coyne et al. (2017) concluded that at least some limited number of twits have predictive power on the stock price[4]. The author manually labelled tweets which cost time, so to avoid spending time on manually labelling tweets, I import the FinBERT. I will test the effectiveness of the method to automatically label tweets.

Sentiment Analysis.

Sentiment analysis is an approach relating to natural language processing (NLP) that identifies the emotion of a text. The first reason why to use sentiment analysis is that some machine learning methods which is used to predict stock price cannot understand text directly, and the second reason is that the number of daily tweets is too large to be used as input of the machine learning methods that can understand text directly such as neural networks with word embedding layer. Therefore, by sentiment analysis tweets can be transferred into numeric vectors and the process to train a machine learning method is speeded up.

FinBERT.

Supervised methods require the step to train models, and it is often not easier to get labelled data. However, it allows us to be very specific about the definition of the labels

[1] https://monkeylearn.com/blog/what-is-tf-idf/

and we can determine the number of classes. In the literature, there is one article using supervised methods to do sentiment analysis. Scott Coyne et al. (2017) labelled tweets as three moods (bullish, bearish, or neutral), and they use MLP to classify tweets[4]. Because the tweets in my dataset are not labelled, a pre-trained model called FinBERT is used. I will use Transformer on Hugging Face, which is a platform that provides the community with APIs to access and use state-of-the-art pre-trained models available from the Hugging Face hub, to implement it.

3.2 Assessment Metric

The Mean-Square Error (MSE) and Mean-Absolute Error (MAE) are selected to assess the performance of the prediction of the close value of Tesla. They compare prediction errors of different models for particular data. The difference between them is that MSE penalizes each large single residual but MAE does not. The formulas for computing them are as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}|$$

Where N is the total number of observations, y_i is the actual value, and (y_i)^ is the predicted value.

3.3 Stock prediction models

The code and the results of all models are available in the link in the Appendix.

Benchmark.

My benchmark is a naïve method which takes the close price at day t as the predicted close price at day t+1. Its MSE is 105.69 and its MAE is 6.05.

Linear Regression.

Linear Regression is a basic machine learning method which is easier to interpret the output coefficients, and I use scikit-learn, which is a free software machine learning library for the Python programming language, to implement it. To ensure the accuracy of linear regression, its assumptions should be satisfied in the prediction of stock price. Its formula is shown below: $y_t = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n + \epsilon$ where each X_i is the TF-IDF vector of the word i in the document(day) t and y_t is the return of close price of the stock of Tesla. Therefore, then we can transfer the predicted return into predicted close price.

Neural Network Models.

The framework I used to construct neural network models is PyTorch which is based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella[11].

LSTM.

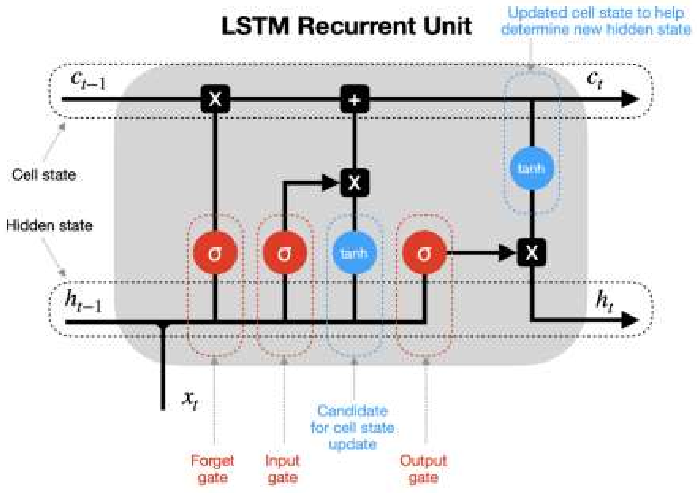

Fig. 1. Architecture of LSTM[5]

Unlike linear regressions, LSTM is a type of Recurrent Neural Network (RNN), that can handle the sequence of dependence among the input variables. LSTM differ from traditional feed-forward networks in the sense that they don't only have neural connections in a single direction, in other words, neurons can pass data to a previous or the same layer. LSTM was introduced by solving the vanishing gradient problem that recurrent networks would suffer when dealing with long time series. In other words, it has both long-term and short-term memory of the important information of inputs, which can provide better performance than RNN. In Figure 1, it achieves this through two special units called "gates" which are forget gate, controlling what information should be forgotten and input gate, identifying important elements that need to be added to the cell state.

I applied LSTM models with a different number of input features. The structures of the LSTM models are shown in Table II.

Table 2. Architecture of LSTM

Parameters	Value	Input Layer	Input	(64,L,C)
Number of LSTM layers	1		output	(64,L,C)
Number of hidden units in LSTM	64	LSTM	Input	(64,L,C)

LSTM layer activation function	tanh		output	(64,64)
Time step	10	Dense	Input	(64,64)
Batch size	64		output	(64,1)
learning rate	0.001			
Optimizer	Adam			
Loss function	MAE			

Where L is the length of the sequence and C is the number of input features.

CNN.

CNN has many layers and parameters including convolutional layers, pooling layers, fully connected layers, number of filters in each layer, dropout rate, size of filters in each layer and initial representation of input data which should be chosen wisely to get the desired outcomes[6]. In detail, convolutional layers simplify complex problems and reduce the number of parameters by filters, pooling layers introduce invariance to local translations and reduce the number of hidden units in hidden layers and fully connected layers convert extracted features in the previous layers, such as convolutional layers or pooling layers, to the final output. Moreover, to avoid overfitting, dropout can be implemented during the training phase. The structures of the CNN model I implemented are shown in Table III.

Table 3. architecture of CNN

Convolution layer filters	32	Input layer	Input	(64,L,C)
Convolution layer kernel size	1		Output	(64,I,C)
Convolution layer activation function	tanh	ConvID	Input	(64,I,C)
Convolution layer padding	Same		Output	(64,I,32)
Pooling layer pool size	1	Maxpooling1D	Input	(64,L,32)
Pooling layer padding	Same		Output	(64,I,32)
Pooling layer activation function	Relu	Dense	Input	(64,I*32)
			Output	(64,1)

Where L is the length of the sequence and C is the number of input features.

CNN-LSTM.

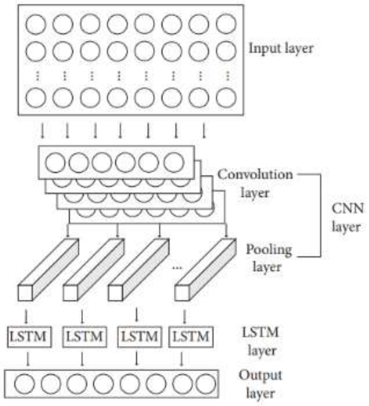

Fig. 2. Architecture of CNN-LSTM

CNN has the characteristic of paying attention to the most obvious features in the line of sight, so it is widely used in feature engineering. LSTM has the characteristic of expanding according to the sequence of time, and it is widely used in time series[8]. To test the findings of Wenjie Lu et al and explore more about the effect of architecture of CNN-LSTM, in Figure 2, on the result of prediction, I tried to modify parameters in their CNN-LSTM model, while their architectures are shown in Table IV.

Table 4. architecture of CNN-LSTM

Parameters	CNN-ISTM in paper	CNN-LSTM in paper		
Convolution layer filters	32		Input	(64, L, C)
convolutionlayer 1 kernel size	1	Input Layer	Output	(64, L, C)
convolution layer activation f	tanh		Input	(64, L, C)
Convolution layer 1 padding	Same	Conv1D	Output	(64, L,32)
Pooling lawer pool size	1		Input	(64, L,32)
Poolinq layer padding	Same	Maxpool-ing1D	Output	(64, L,32)
Pooling layer activation funct	Relu		Input	(64, L,32)
Number of LSTM lawers	Unknown	LSTM	Output	(64, 64)
Number of hidden units in each	64		Input	(64,64)
LSTM layer activation function	tanh	Dense	Output	(64,1)
Time step	10			
Batch size	64			
Learning rate	0.001			
Optimizer	Adam			
Loss function	MAE			
Epochs	100			

Where L is the length of the sequence and C is the number of input features.

Batch size and Epochs.

The batch size used in this paper is 64 which is large enough to speed up training steps. Instead of using the whole dataset to compute gradients, minibatch can avoid local minimum by adding noise. The number of epochs is 150 which is selected after checking the MAE vs epoch figure for all neural network models. The MAEs of validating set of all models stop decreasing at about 100 epochs.

4 Results

4.1 Granger Causality Test

A granger causality test, which is a statistical hypothesis test for determining whether a time series is useful for forecasting another, is performed on each predictor variable. Anshul Mittal et al. (2021) tested whether the mood values returned can be used to predict the future stock movements by granger causality test[9], so I applied it on close prices and all predictor variables. For neural network models, in the dataset including all users, 16 variables including weighted labels among 19 variables are selected as the candidates of the input features for forecasting the close prices on the next days. The three abandoned variables are weights, weighted scores and volumes. In the dataset including only smart users, both weighted labels and weighted scores are not significantly useful for forecasting the close values of Tesla on the next day. Because weighted labels represent the mood of public and it is the only difference between full dataset and dataset including only smart users, weighted labels, and the rest 15 technical indicators are selected as the candidates of the input features of models for dataset including only smart users. For the linear regression model, the candidates of input features are 15 technical indicators and all words in vocabulary list of TF-IDF vector form. The number of words in vocabulary is 28650 and 9063 for full dataset and dataset including smart tweets only respectively. The detail of each model will be abbreviated as (model name)_(input feature index)_(output feature)_(optimizer)_(dataset), where the corresponding combination of input feature to input feature index is shown in Table V.

4.2 Result of Models

Because the result of neural network models is not constant after each training-validating-testing procedure, I repeat the procedure for each neural network model 20 times and use average MSEs, average MAEs, standard deviations of 20 MSEs and standard deviation of 20 MAEs to evaluate their performance. For linear regression, the result is constant, so one round of the training-validating-testing process is enough. All results are reported in Table VI and Table VII.

Table 5. Index of Input features combination

Index of Input features combination	input features
1	close_value
2	close_value, weighted_label
4	close_value, high_value, low_value, open_value
5	close_value, high_value, low_value, open_value, volume
5_no_vol	close_value, high_value, low_value, open_value, weighted_label
6	close_value, high_value, low_value, open_value, volume, weighted_label
16	adm, close_value, ema_close, ema_high, ema_low, high_value , low_band, low_value, ma_close, ma_high, ma_low, mid_band, open_value, rsi, up_band, weighted_label
16_tfidf	adm, close_value, ema_close, ema_high, ema_low, high_value , low_band, low_value, ma_close, ma_high, ma_low, mid_band, open_value, rsi, up_band, weighted_label, tf-idf vectors of all words
tfidf	tf-idf vectors of all words

Table 6. Average MSE, Average MAE, Standard deviation of MSEs and Standard deviation of MAEs of neural network models with input feature index of 5, two types of outputs and full datasets

	Feature index	**Target**	**Optimizer**	**Dataset**	**Average MSE**	**Average MAE**	**STD MSE**	**STD MAE**
Benchmark	None	None	None	None	105.69	6.05	0	0
LSTM	5	close	adam	fu11	113.95	6.96	3.91	0.17
CNN_LSTM	5	close	adam	fu11	121.09	7.40	8.22	0.32
LSTM	5	return	adam	fu11	113.04	6.94	3.75	2.00
CNN LSTM	5	return	adam	fu11	107.78	6.70	4.40	0.19
LSTM	5	return	adam	fu11	105.51	6.42	2.00	0.09
CNN LSTM	5	return	adam	fu11	103.82	6.35	0.94	0.10

Table 7. 95% Confidence Interval of MSEs and MAES of neural network models with input feature index of 5, two types of outputs and full datasets

	Feature index	**Target**	**Optimizer**	**Dataset**	**95% lower bound**	**95% upper bound**	**95% lower bound**	**95 upper bound**
Benchmark	None	None	None	None	105.69	105.69	6.05	6.05
LSTM	5	close	adam	fu11	106.29	121.61	6.62	7.29
CNN_LSTM	5	close	adam	fu11	104.97	137.20	6.78	8.03
LSTM	5	return	adam	fu11	105.68	120.40	6.55	7.33

CNN LSTM	5	re-turn	adam	fu11	99.16	116.40	6.32	7.08
LSTM	5	re-turn	adag-rad	fu11	101.60	109.42	6.24	6.60
CNN LSTM	5	re-turn	adag-rad	fu11	101.97	105.67	6.15	6.54

Table 8. Average MSE, Average MAE, Standard deviation of MSEs and Standard deviation of MAEs of neural network models with all input feature index, return as output and all datasets

	Feature index	Tar-get	Opti-mizer	Da-taset	Average MsE	Average MAE	STD MSE	STD MAE
Bench-mark	None	None	None	None	105.69	6.05	0	0
CNN LSTM	1	return	adag-rad	fu11	105.98	6.05	0.30	0.05
CNN LSTM	2	return	adag-rad	fu11	107.32	6.48	2.69	0.13
CNN LSTM	4	return	adag-rad	fu11	105.90	6.06	0.34	0.03
CNN LSTM	5	return	adag-rad	fu11	103.82	6.35	0.94	0.10
CNN LSTM	5_no_vol	return	adag-rad	fu11	103.36	6.18	1.80	0.05
CNN LSTM	6	return	adag-rad	fu11	102.10	6.33	4.04	0.19
CNN LSTM	16	return	adag-rad	full	105.29	6.27	3.83	0.12
LSTM	1	return	adag-rad	fu11	106.00	6.03	0.11	0.01
LSTM	2	return	adag-rad	fu11	99.85	6.17	2.57	0.11
LSTM	4	return	adag-rad	fu11	105.65	6.06	0.42	0.02
LSTM	5	return	adag-rad	fu11	105.509964	6.422391	1.99653	0.09142
LSTM	5_no_vol	return	adag-rad	fu11	100.221305	6.202569	3.41496	0.15039
LSTM	6	return	adag-rad	fu11	98.92637	6.436547	6.62267	0.30707
LSTM	16	return	adag-rad	fu11	107.658646	6.594156	9.61233	0.34739
LSTM	2	return	adag-rad	smart	112.2514	6.341884	2.32763	0.04798
LSTM	5_no_vol	return	adag-rad	smart	111.203904	6.314734	3.29082	0.05656

Table 9. 95% Confidence Interval of MSEs and MAES of neural network models with all input feature index, return as output and all datasets

	Feature in-dex	Target	Opti-mizer	Da-taset	95% lower boun	95% up-per bounc	95% lower bouni	95 up-per boun
Bench-mark	None	None	None	None	105.69	105.69	6.05	6 05
CNN LSTM	1	return	adag-rad	fu11	105.39	106.56	5.95	6.15
CNN LSTM	2	return	adag-rad	fu11	102.05	112.55	6.23	6.73

CNN LSTM	4	return	adag-rad	full	105.23	106.56	6.00	6.11
CNN LSTM	5	return	adag-rad	full	101.57	105.67	6.16	6.54
CNN LSTM	5_no_vol	return	adag-rad	full	99.83	106.88	6.08	6.27
CNN LSTM	6	return	adag-rad	full	94.19	110.01	5.96	6.69
CNN LSTM	16	return	adag-rad	full	97.79	112.78	6.04	6.50
LSTM	1	return	adag-rad	full	105.79	106.21	6.01	6.05
LSTM	2	return	adag-rad	full	94.82	104.88	5.55	6.35
LSTM	4	return	adag-rad	full	104.82	106.05	6.01	6.11
LSTM	5	return	adag-rad	full	101.60	109.42	6.24	6.60
LSTM	5_no_vol	return	adag-rad	full	93.53	106.91	5.91	6.46
LSTM	6	return	adag-rad	full	86.01	111.97	5.83	7.04
LSTM	16	return	adag-rad	full	88.82	126.46	5.91	7.28
LSTM	2	return	adag-rad	smart	107.69	116.81	6.25	6.44
LSTM	5_no_vol	return	adag-rad	smart	104.75	117.65	6.20	6.43

Result of Neural Network Models.

Because many articles and blogs treat close prices as the output of neural networks, I first use all candidate models to predict close prices on the next day directly. However, after I run training-validating-testing one time for all neural network models, their MAEs and MSEs are all larger than the benchmark especially CNN with an MSE of 141.01 and an MAE of 8.06, so I exclude CNN from later implementation.

Then, I repeat the training-validating-testing procedure for the LSTM model and the CNN-LSTM model with input feature index 5 (LSTM_5_close_adam_full) and see whether it is true that neural network models with close value as input is worse. In Table VI, average MSEs and average MAEs of LSTM_5_close_adam_full which are 113.951383 and 6.957606 are worse than those of benchmark and those of LSTM model with input feature index 5 and return as output feature (LSTM_5_return_adam_full). This phenomenon also happens between CNN_LSTM_5_close_adam_full and CNN_LSTM_5_return_adam_full. Thus, I tried to use the neural network models to predict the return first and then convert it into the predicted close price. Although the results neural network model, predicting return first, were still worse than that of the benchmark, I keep predicting returns at first and convert them into close prices in later implementation.

However, all the MAE vs epoch graphs of the current LSTM and CNN-LSTM models show an unstable and divergent trend and their MAE reduces to around 0.0225. Their MAE vs epoch graphs are shown in Figure 3 and Figure 4 where 'valid' is the result of validating set and 'test' is the result of testing set:

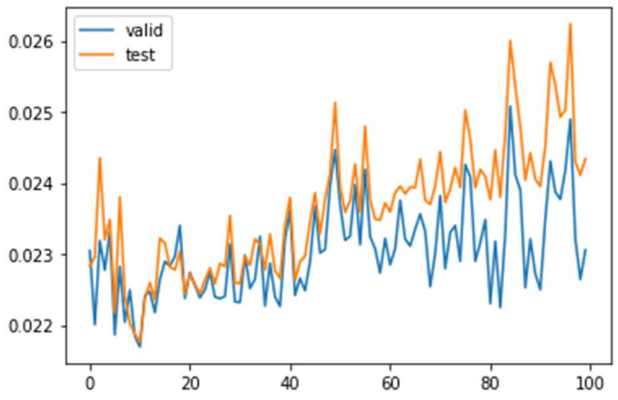

Fig. 3. MAE vs Epoch of CNN_LSTM_5_return_adam_full

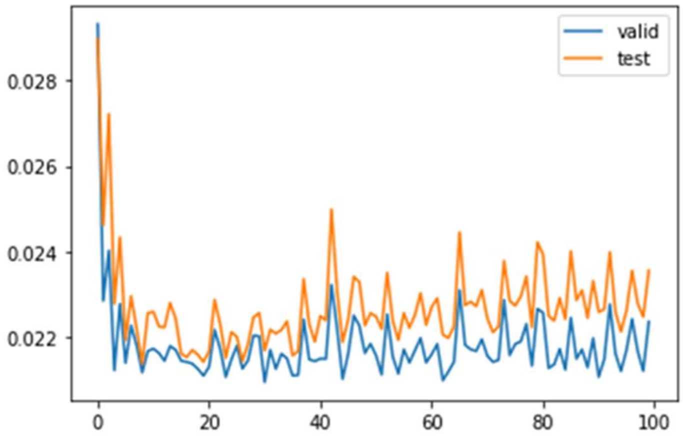

Fig. 4. MAE vs Epoch of LSTM_5_return_adam_full

Thus, I use Adagrad instead of Adam as the optimizer of the models. Adagrad (short for adaptive gradient) penalizes the learning rate for parameters that are frequently updated, instead, it gives more learning rate to sparse parameters, parameters that are not updated as frequently. In other words, the learning rate will be reduced after each iteration and the MAE will not fluctuate as epochs increase. In Figure 5 and Figure 6, of the LSTM and the CNN-LSTM model with 5 input features and adagrad as the optimizer, the volatilities of the lines decrease as the number of epochs increases and the line of validating set is more likely to be parallel to the line of the testing set.

In Table VI, average MSEs, average MAEs, standard deviation of MSEs and standard deviation of MAEs of LSTM_5_return_adagrad_full, which are 105.51, 6.42, 2.00 and 0.09, are smaller than those of LSTM_5_return_adam_full respectively. The average MSEs, average MAEs, standard deviation of MSEs and standard deviation of MAEs of CNN_LSTM_5_return_adagrad_full, which are 103.82, 6.35, 0.94 and 0.10, are smaller than those of LSTM_5_return_adam_full respectively.

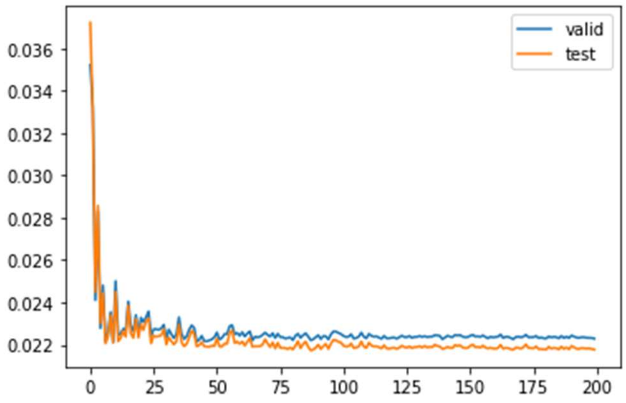

Fig. 5. CNN_LSTM_5_return_adagrad_full

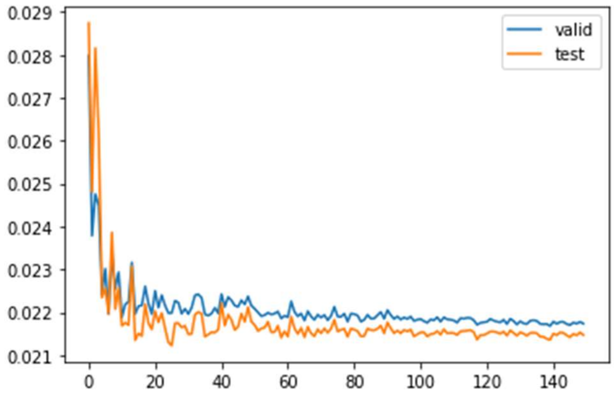

Fig. 6. LSTM_5_return_adagrad_full

Therefore, I get average MSEs, average MAEs, the standard deviation of MSEs, standard deviation of MAEs and 95% confidence interval of MSEs and MAEs for CNN_LSTM and LSTM with all combinations of input features which are shown in Table VIII and Table IX, and find that none of the models can have both average MSE and average MAE less than the MSE and MAE of benchmarks respectively. If MSE is the only evaluation metric, based on the 95% confidence interval, we can conclude that there is significant evidence that the average MSEs of LSTM_2_return_adagrad_full and CNN_LSTM_5_return_adagrad_full are lower than the constant MSE of Benchmark and that the average MAEs of LSTM_1_return_adagrad_full is lower than the constant MAE of Benchmarks. Moreover, for LSTM models, the average MSE of LSTM_6_return_adagrad_full is lowest and the average MAE of LSTM_1_return_adagrad_full is lowest; for CNN-LSTM models, the average MSE of CNN_LSTM_6_return_adagrad_full is lowest and the average MAE of CNN_LSTM_1_return_adagrad_full is lowest. So adding several technical indicators which even though pass Granger Causality test, in input features, cannot guarantee an improvement in the performance of LSTM and CNN-LSTM. Besides, the weighted label also cannot guarantee an improvement. For example, CNN_LSTM_2_return_adagrad_full has one more input feature (weighted label) than CNN_LSTM_1_return_adagrad_full, and the former has a larger average MSE and a larger average MAE, but LSTM_1_return_adagrad_full has larger average MSE and average MAE than LSTM_2_return_adagrad_full. In conclusion, the performance of CNN-LSTM models predicting the return first of the stock of Tesla does not dominate that of LSTM models, which is not consistent with the performance of CNN-LSTM in the paper of Wenjie Lu et al, even if predicting the return first.

Table 10. average mses, average maes, standard deviation of mses and standard deviation of maes of linear regression with all input feature index, two types of outputs and all datasets(linear regression has no randomness so std mses and std maes are zero)

	Feature index	Target	Optimizer	Dataset	Average MSE	Average MAE	STD MSE	STD MAE
Benchmark	None	None	None	None	105.69	6.05	0	0
linear regression	tfidf	return	None	full	111.48	6.96	0	0
linear reqression	16	return	None	full	107.57	6.82	0	0
linear reqression	16 _tfidf	return	None	full	110.49	7.32	0	0
linear reqression	tfidf	return	None	smart	128.12	7.61	0	0
linear regression	16	return	None	smart	109.90	6.89	0	0
linear regression	16 _tfidf	return	None	smart	128.76	7.84	0	0
linear reqression	tfidf	close	None	full	1147.12	28.62	0	0
linear regression	16	close	None	full	107.06	6.79	0	0
linear regression	16 _tfidf	close	None	full	109.44	7.23	0	0
linear regression	tfidf	close	None	smart	1264.37	29.54	0	0
linear reqression	16	close	None	smart	109.47	6.85	0	0
linear regression	16 _tfidf	close	None	smart	126.52	7.70	0	0

Result of Linear Regression.

In Table X, I compared the result of linear regression with different combinations of output feature and input features, where the best one for the full dataset was linear regression with 16 technical indicators as inputs and close value as output (Linear regression_16_close_full) even though Scott et al. used TF-IDF vectors as inputs. The MSE and MAE of the best ones are 107.06 and 6.79 which are worse than those of the benchmark. Furthermore, the smart tweets strategy based on FinBERT does not work for each type of input and output combination, which suggests FinBERT cannot be an automatic labelling tool to distinguish bearish, bullish or neutral emotions.

4.3 Result of Smart Tweets Strategy

According to Table X, by filtering out smart tweets, the MSE and MAE of linear regression are reduced from 113.26 and 7.02 to 104.39 and 6.64 respectively. Although its MAE is higher than that of Benchmarks, its MSE is lower. However, based on Table VIII it makes the prediction of the LSTM model worse, while MSE and MAE of LSTM_2 increase from 99.85 and 6.17 to 112.25 and 6.34 respectively.

4.4 Result of FinBERT

From the perspective of sentiment analysis tools, FinBERT, joining as weighted labels, improves the average MSE of tweets sentiment for LSTM with return as output and some CNN-LSTM such as CNN_LSTM_4_return_adagrad_full, according to Table VIII. Although it reduces the average MSE for some inputs combinations, it always increases the standard deviation of MSE for LSTM and CNN-LSTM, such as the pairs of (LSTM_1_return_adagrad_full, LSTM_2_return_adagrad_full), (CNN_LSTM_4_return_adagrad_full, CNN_LSTM_5_no_vol_return_adagrad_full)

5 Conclusion

Among all combinations of models, inputs and outputs, LSTM_1_return_adagrad_full and LSTM_2_return_adagrad_full are the only two which are significantly better than benchmarks in MAE and MSE respectively. Wenjie Lu et al.'s method of using CNN-LSTM to predict close returns, of the Shanghai Composite Index, is worse for the dataset of Tesla with the candidates of input features. Even if the output feature of CNN-LSTM is changed to return, its average MSE is larger than that of LSTM_2_return_adagrad_full and benchmark, and its average MAE is larger than that of LSTM_1_return_adagrad_full and benchmark. Also, Scott et al.'s method of using TF-IDF vectors of tweets to predict returns, of the stock price of Apple, cannot be applied to the dataset of Tesla because its performance on the dataset of Tesla is even worse than that of the benchmark. For weighted label, a public emotion variable, the result of LSTM with close value as input is improved by adding it in input significantly because the 95% confidence lower bound of LSTM_1_return_adagrad_full is larger than the 95% confidence upper bound of LSTM_2_return_adagrad_full.

All the conclusion above is based on the limited candidate input features. When the input feature index is 16, the average MSE, average MAE, the standard deviation of MSE and standard deviation of MAE of CNN-LSTM is better than LSTM, so CNN-LSTM can perform best with a particular combination of input features. With Turnover, Ups and downs and Change used by Wenjie Lu, but with no explanation, which cannot be generated for my dataset, CNN-LSTM might be a better model to predict stock price than LSTM. Moreover, the tweets of Tesla from Kaggle do not provide good information as TF-IDF vector for linear regression. It even includes some useless tweets such as "happy new year". However, Scott et al.'s datasets are from StockTwits which is a social media service for traders, investors and entrepreneurs to share ideas regarding stock information. Most importantly, StockTwits granted them partner-level access to their database, which might include a higher quality of information on tweets.

The limitations of this research point toward topics to be addressed in the future. The most important two are finding a good and comprehensive dataset with all necessary variables and testing the models on more datasets so that the conclusion of this paper can be generalized to all datasets.

References

1. Araci, Dogu. (2019). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models.
2. Bollen J, Mao H, Zeng X. Twitter mood predicts the stock market[J]. Journal of computational science, 2011, 2(1): 1-8.
3. Chng, Z. M. (2022, June 19). Using normalization layers to improve deep learning models. MachineLearningMastery. https://machinelearningmastery.com/using-normalization-layers-to-improve-deep-learning-models/
4. Coyne S, Madiraju P, Coelho J. Forecasting stock prices using social media analysis[C]//2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, 2017: 1031-1038.
5. Dobilas, S. (2022, March 5). LSTM Recurrent Neural Networks—How to Teach a Network to Remember the Past. Medium. https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e
6. Hoseinzade E, Haratizadeh S. CNNpred: CNN-based stock market prediction using a diverse set of variables[J]. Expert Systems with Applications, 2019, 129: 273-285.
7. Bing L, Chan K C C, Ou C. Public sentiment analysis in Twitter data for prediction of a company's stock price movements[C]//2014 IEEE 11th International Conference on e-Business Engineering. IEEE, 2014: 232-239.
8. Lu W, Li J, Li Y, et al. A CNN-LSTM-based model to forecast stock prices[J]. Complexity, 2020, 2020: 1-10.
9. Mittal A, Goel A. Stock prediction using twitter sentiment analysis[J]. Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittalStockMarketPredictionUsingTwitterSentimentAnalysis. pdf), 2012, 15: 2352.
10. Nelson D M Q, Pereira A C M, De Oliveira R A. Stock market's price movement prediction with LSTM neural networks[C]//2017 International joint conference on neural networks (IJCNN). Ieee, 2017: 1419-1426.
11. PyTorch - Wikipedia. (2016, September 1). PyTorch - Wikipedia. https://en.wikipedia.org/wiki/PyTorch
12. Ribeiro F N, Araújo M, Gonçalves P, et al. Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods[J]. EPJ Data Science, 2016, 5: 1-29.

13. Sharma, H. (2023, January 23). Exploring the Best Indicators in TA-Lib: Technical Analysis of Stocks using Python- Part 1. Medium. https://medium.com/mlearning-ai/exploring-the-best-indicators-in-ta-lib-technical-analysis-of-stocks-using-python-part-1-b7ad731aeeb2
14. Siami-Namini S, Tavakoli N, Namin A S. A comparison of ARIMA and LSTM in forecasting time series[C]//2018 17th IEEE international conference on machine learning and applications (ICMLA). IEEE, 2018: 1394-1401
15. Tweets about the Top Companies from 2015 to 2020. (2020, November 26). Kaggle. https://www.kaggle.com/datasets/omermetinn/tweets-about-the-top-companies-from-2015-to-2020
16. Understanding TF-IDF: A Simple Introduction. (2019, May 10). MonkeyLearn Blog. https://monkeylearn.com/blog/what-is-tf-idf/
17. Values of Top NASDAQ Companies from 2010 to 2020. (2020, November 26). Kaggle. https://www.kaggle.com/datasets/omermetinn/values-of-top-nasdaq-copanies-from-2010-to-2020
18. Xu Y, Cohen S B. Stock movement prediction from tweets and historical prices[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018: 1970-1979.