A Study on Stock-Return Forecasting with Sentiment Analysis using Distributed Deep Learning

Abhinav Mandava
B00841453
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia
Email: Abhinav.Mandava@dal.ca

***Abstract*—Sentiment Analysis is a powerful technique which leverages natural language processing, text analysis, and computational linguistics to extract and quantify the sentiment or public opinion on a subject. Sentiment analysis can be used for social media monitoring, customer support management, analyzing customer feedback, and market analysis. Knowing the aggregated sentiment on any subject can be helpful in improving and optimizing it. In this paper, an end-to-end sentiment analysis pipeline is built which demonstrates how a sentiment classification model can be built with Recurrent Neural Networks (uses LSTM), and is applied to experiment with stock market which is highly stochastic and is one of the major areas of research in time-series forecasting. The paper also introduces distributed deep learning which is highly essential in this era of big data, to train deep learning models on huge data setss.**

I. Introduction

Sentiment analysis uses natural language processing at its core. Natural language processing is an area in computer science that deals with the ability of computers to understand human language. Hence, the first step is to devise a technique which can represent the words in the language mathematically such that the semantic information is preserved. For example, words with similar meaning, or context should have similar representations. This is made possible by formulating techniques which represent words as *feature vectors*. Once we have the vectors, we can do text analysis, or apply machine learning techniques to achieve the objectives. The main text vectorization techniques are: *Bag of Words*, *TF-IDF*, abd *Word Embedding*. Here, word embeddings were generated using Word2Vec model. Some other techniques are also efficient like neural embeddings where the vector representations are learned in the *Embedding* layer of a neural network

In this paper, a sentiment classifier was built using an LSTM (*Long Short-Term Memory*) network, trained on 1.6 million tweets from the *Sentiment140* data set. A Spark cluster with two nodes and *PySpark* API were used to preprocess the tweets. *Horovod* framework was used for distributed training of the neural network on Spark cluster. With the trained model, the sentiment was obtained for the tweets and news related to four companies, *Apple, Tesla, Microsoft,* and *Google* for the last one year. The sentiment score was then added as an additional feature to the daily stock data obtained from Yahoo finance using which two models were built: *Model A* which does not use daily sentiment information, and *Model B* which uses daily sentiment information. Thus, the workflow for this study is shown in fig 1.

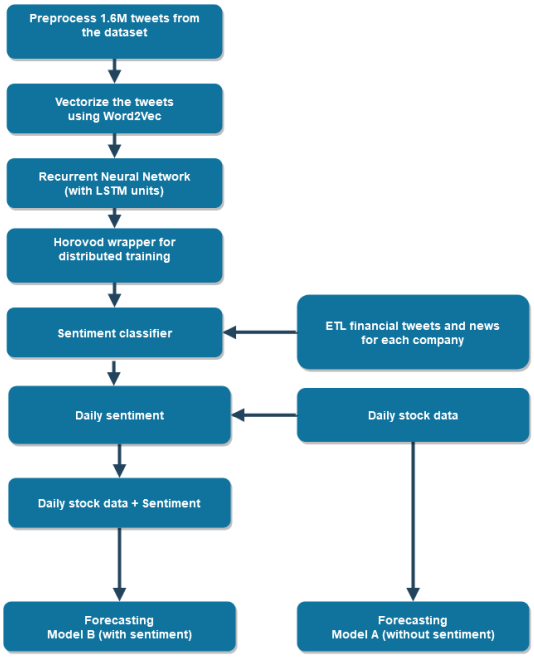


Fig. 1. Workflow

II. Data Preprocessing

In this phase, the Sentiment140 dataset with 1.6M tweets was uploaded onto the Spark cluster provided by *Databricks*. It uses *Databricks File System* which is distributed file system just like HDFS. Once the dataset was stored on DBFS, using PySpark API, processed the tweets by removing URLs, smileys, non-ASCII characters, special symbols, and stopwords. Tokenized the tweets and stored them back in DBFS. The preprocessing took 3.43 seconds on the cluster.

III. Word Embedding with Word2Vec

Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. It can be obtained using two methods (both involving Neural Networks): Skip Gram and Common Bag Of Words (CBOW)[1].

Continuous Bag-of-Words Model predicts the middle word based on surrounding context words. The context consists of a few words before and after the current (middle) word. This architecture is called a bag-of-words model as the order of words in the context is not important[2].

Continuous Skip-gram Model which predict words within a certain range before and after the current word in the same sentence. A worked example of this is given below[2].

Here, Word2Vec model from Apache Spark's MLlib was used to obtain the word embeddings for the tweets. This feature extraction step took 8.42 minutes on the cluster.

IV. Neural Network for Sentiment Classification

There are several pretraiend sentiment classifiers with high accuracy scores. Some popular ones are:

- **NLTK's Vader:** This sentiment analysis tool uses a bag of words approach (a lookup table of positive and negative words) with some simple heuristics (e.g. increasing the intensity of the sentiment if some words like "really", "so" or "a bit" are present)[3]. The advantage of this approach is that sentences containing negated positive words (e.g. "not happy", "not good") will still receive a negative sentence sentiment.

- **Flair**: This model is based on a character-level LSTM neural network which takes sequences of letters and words into account when predicting[3]. The network learned to take negations into account

- **DeepMoji:** It predicts emojis for a sentence. It is slightly sophisticated (a deep bi-LSTM with an attention mechanism) and can detect emotion as well.

Apart from these, there are many other highly sophisticated pretrained models like Google's BERT, XLNet, Binary Partitioning Transfomer (BPT), ERNIE etc. However, in this paper, a basic neural classifier for sentiment is built by using LSTM units so that the crux of sequence models and how they are suitable for natural language processing can be elucidated.

A. Recurrent Neural Networks for Sequence Learning

A recurrent neural network (RNN) uses sequential data(like sentences) or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, time series forecasting, and image captioning. They are incorporated into popular applications such as Siri, voice search, and Google Translate[5]. They are

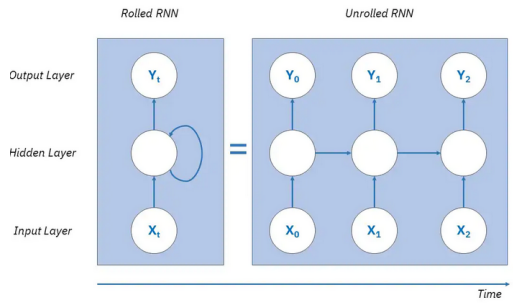


Fig. 2. An RNN unit

$$p(y_T = k \mid x_1, x_2, x_3, \ldots, x_T)$$

Fig. 3. Probabilistic model for an RNN

distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. Thus, they are autoregressive in nature. In a probabilistic framework, RNNs model the probability of the current element given the sequence of previous elements.

B. LSTM Network

RNNs suffer from the vanishing gradient problem which makes them forget the data in the sequence if the sequences are long. This poses a problem when we have long sequences like a tweet or a small paragraph. This problem is solved by *Long Short-Term Memory* networks which comprise forget gates and update gates that are basically neuron units which learn what data to forget and what to keep as they train on long sequences.

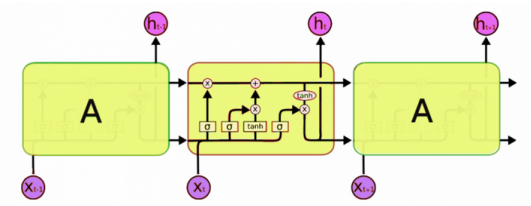


Fig. 4. An LSTM unit

Thus, an LSTM unit will have two inputs, one is the *cell state* which gives the probability of forgetting or updating the data from the previous time step, and the other is the input from the previous time step.

Thus, for the task of sentiment classification (essentially NLP) which involves sequences of tweets, LSTMs are a great choice. Many sophisticated, state-of-the-art NLP models like BERT, XLNet, GraphStar etc. are complex models which use LSTMs internally. Here, the network for sentiment classifica-

tion is shown in fig 5.

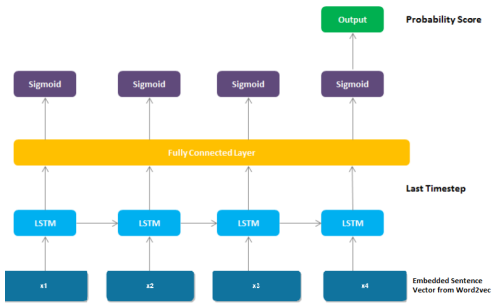


Fig. 5. Sentiment classifier model architecture

V. Distributed Deep Learning

When it is not desirable to store the whole data-set or a model on a single machine, it becomes necessary to store the data or model across multiple machines.

- **Data parallelism:** Data is distributed across multiple machines. This can be used in case data is very large and not practical to be stored on a single machine or to achieve faster training[6].

- **Model parallelism:** If the model is too big or complex to be fit into a single machine, it can be distributed across multiple machines. For example, a single layer can be fit into the memory of a single machine and forward and backward propagation would involve communication of output from one machine to another in a serial fashion[6]. We resort to model parallelism only if the model cannot be fit into a single machine and not so much to fasten the training process.

  In this paper, data parallelism is used since the dataset is large (1.6M tweets), and we want faster training time.

A. Frameworks for distributed training

Though Apache Spark has a complete machine learning lirary, MLlib for training machine learning models on large datasets, it does not have support for deep learning and artificial neural networks. However, there are several distributed deep learning training frameworks available like:

- Distributed TensorFlow.
- Apache SystemDS (formerly SystemML).
- Horovod

1) Distributed TensorFlow:: *tf.distribute.Strategy* is a TensorFlow API to distribute training across multiple GPUs, multiple machines or TPUs. Using this API, one can distribute the existing models and training code with minimal code changes[8]. There are several types of strategies for distributed training. *tf.distribute.MirroredStrategy* is the commonly used strategy that supports synchronous distributed training on multiple GPUs. It creates one replica per GPU device. Each variable in the model is mirrored across all the replicas. Together, these variables form a single conceptual variable called MirroredVariable. These variables are kept in sync with each other by applying identical updates. Efficient *all-reduce* algorithms are used to communicate the variable updates across the devices. All-reduce aggregates tensors across all the devices by adding them up, and makes them available on each device. Also, there are several implementations of the all-reduce algorithm, depending on the type of communication available between devices. MirroredStrategy uses NVIDIA NCCL as the all-reduce implementation by default[7].

2) Apache SystemDS:: Apache SystemDS was formerly known as SystemML and was developed by IBM. It can be run on top of Apache Spark or standalone, where it automatically scales data, determining whether the code should be run on the driver or an Apache Spark cluster[9]. SystemDS includes additional deep learning with GPU capabilities such as importing and running neural network architectures and pre-trained models for training.

3) Horovod:: Horovod is an open-source distributed deep learning framework developed by Uber. Horovod provides two APIs for running Horovod on Spark: a high level Estimator API and a lower level Run API. Both use the same underlying mechanism to launch Horovod on Spark executors, but the Estimator API abstracts the data processing (from Spark DataFrames to deep learning datasets), model training loop, model checkpointing, metrics collection, and distributed training. HorovodEstimator facilitates distributed, multi-GPU training of deep neural networks on Spark DataFrames, simplifying the integration of ETL in Spark with model training in TensorFlow[9]. HorovodEstimator thus provides good user experience, where it provides an abstraction on our traditional model, and deals with the distributed training part at low-level. All we have to do is create our Keras(or PyTorch) model and wrap it inside the respective HorovodEstimator(eg: KerasEstimator), and seamlessly work with Spark dataframes(which are distributed in nature) to train the model.

Here, Horovod framework was used because of its simplicity and support for Databricks. The test accuracy of the sentiment classifier was found to be 81.92%. The training time was observed to be 22.86 minutes.

VI. ETL OF TWEETS AND NEWS DATA

In this phase, the daily tweets from twitter accounts pertaining to finance and investment like *Stocktwits, Benzinga, CNBC, BreakoutStocks, bespokeinvest, WSJmarkets, StephanieLink, nytimesbusiness, IBDinvestors, and WSJDealJournal* were extracted for the four companies for the last one year. Also, news from New York Times API was obtained for the last one year.

These data were processed by removing special symbols, URLs, smileys, stop-words, and non-ASCII data. These processed tweets and news were then combined as per the dates, and were synced up with the dates in the stocks dataset. This daily info (tweets and news) data was fed to the sentiment classifier which predicted the sentiment between 0(negative) and 1(positive). This predicted value was used as the sentiment score. The sentiment score for a given date is the average of the scores of all the tweets or news items on that day. Thus, the sentiment scores for the dates in the stock dataset were obtained. The dates for which there were no tweets and news were given a score equal to the average score over the year. An example of stock data with and without the sentiment score is shown below:

	Stock	Date	Open	High	Low	Close	Volume
141	MSFT	2020-07-01	203.139999	206.350006	201.770004	204.699997	32061200
142	MSFT	2020-07-02	205.679993	208.020004	205.000000	206.259995	29315800
143	MSFT	2020-07-06	208.830002	211.130005	208.089996	210.699997	31897600
144	MSFT	2020-07-07	210.449997	214.669998	207.990005	208.250000	33600700
145	MSFT	2020-07-08	210.070007	213.259995	208.690002	212.830002	33600000

Fig. 6. Stock data without sentiment score

	Stock	Date	Open	High	Low	Close	Volume	Sentiment Score
141	MSFT	2020-07-01	203.139999	206.350006	201.770004	204.699997	32061200	0.414730
142	MSFT	2020-07-02	205.679993	208.020004	205.000000	206.259995	29315800	0.414596
143	MSFT	2020-07-06	208.830002	211.130005	208.089996	210.699997	31897600	0.414596
144	MSFT	2020-07-07	210.449997	214.669998	207.990005	208.250000	33600700	0.414658
145	MSFT	2020-07-08	210.070007	213.259995	208.690002	212.830002	33600000	0.414859

Fig. 7. Stock data with sentiment score

VII. FORECASTING MODELS

Forecasting models are autoregressive. They use the data from the previous time steps, and predict the next value. The dimensionality of data for a forecasting model just like any sequence model, is NxTxD, where N is the number of time steps(number of sequences in total), T is the length of the time step(sequence length), and D is the dimensionality of the data. For instance, in the stock dataset, N would be the size of the dataset, T is the length of the window(time step) for forecasting, and D is the number of dimensions (High, Low, Open, Close,..).

Forecasting models often work on time series data which is just another type of sequence data. Hence, recurrent neural networks can be levaraged to build forecasting models as well. Here, LSTM network was used to build the forecasting models A, and B.

Stock forecasting uses the features: Open, Close, High, Low, Volume. The target feature for the forecast can be any of those features since forecasting is autoregressive. However, in this paper, the we are interested in the 'Return' which is calculated by:

Return = (Close(t) - Close(t-1))/Close(t-1)

In both the models A, and B, the goal is to predict if there would be positive return or not, and the numerical value of return is not the concern. In this experiment, for each of the companies, the forecasting model A does not use daily sentiment and the forecasting model B uses daily sentiment. The dataset split is 80:20.

Stock data is stochastic and this can be inferred from the following figures 8 and 9 which are the error, and accuracy on train and test sets respectively. Though these belong to the stock 'TSLA', the forecasting models corresponding to the other stocks also exhibit the same behaviour. The test error goes up indicating that the model is fitting to the noise which is inevitable from the fact that stock data is random.

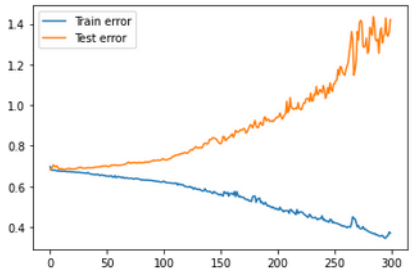


Fig. 8. Error

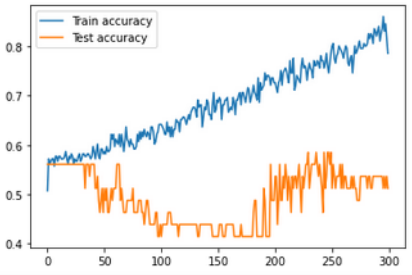


Fig. 9. Accuracy

The following figures show the trends of the stocks of the respective companies. Below the trend of each company is the evaluation of the two forecasting models A, and B for that stock. The models predict whether or not there is a positive return on the stock. The evaluation of the models show how the models are predicting the probability of return. From the evaluation metrics, it can be observed that for 3 out of 4 companies(TSLA, GOOGL, and AAPL), the model with sentiment does a better job of predicting if there would be a return.

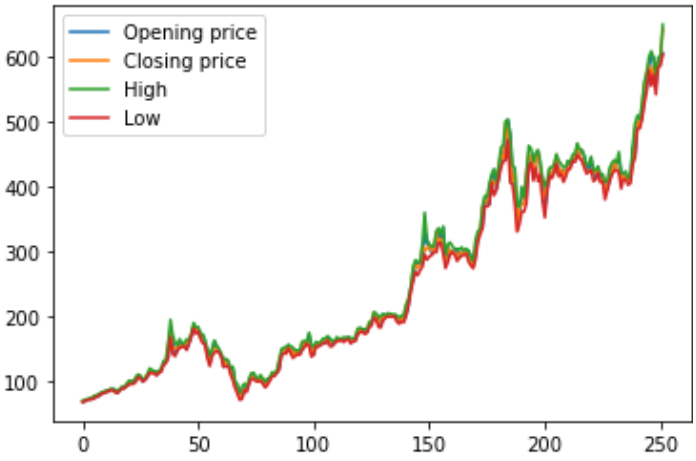
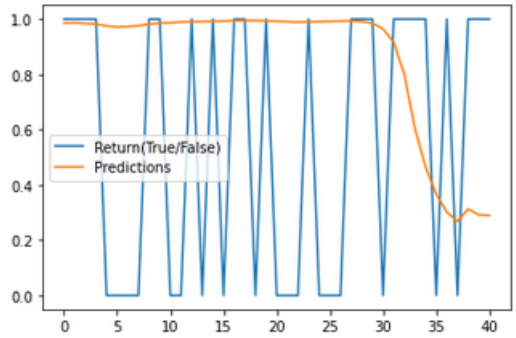


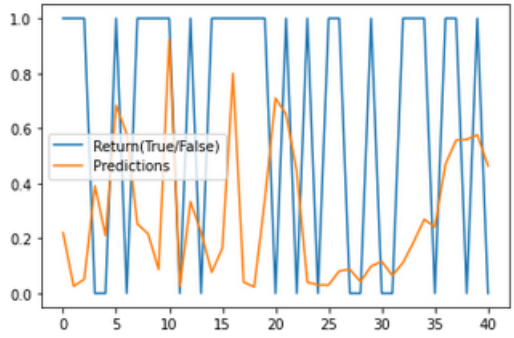
Fig. 10. 'TSLA' stock trend

Fig. 13. 'GOOGL' stock trend



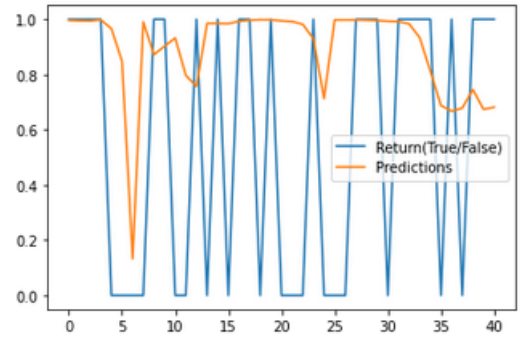

```
Precision:  0.40756302521008403
Recall:  0.4468599033816425
F1 score:  0.3957894736842105
Accuracy:  0.4878048780487805
```

Fig. 11. Metrics for model-A(no sentiment). Stock: 'TSLA'



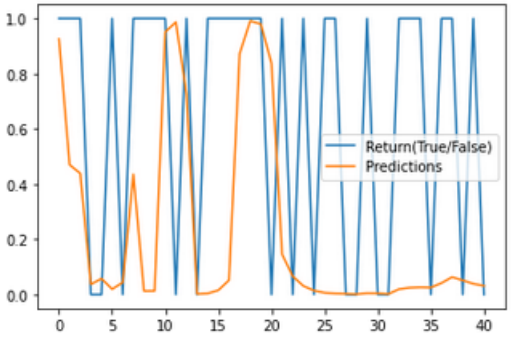

```
Precision:  0.5208333333333333
Recall:  0.5153846153846154
F1 score:  0.42674772036474173
Accuracy:  0.43902439024390244
```

Fig. 14. Metrics for model-A(no sentiment). Stock: 'GOOGL'




```
Precision:  0.7875
Recall:  0.5277777777777778
F1 score:  0.4177109440267335
Accuracy:  0.5853658536585366
```

Fig. 12. Metrics for model-B(with sentiment). Stock: 'TSLA'




```
Precision:  0.571969696969697
Recall:  0.5487179487179488
F1 score:  0.4473039215686274
Accuracy:  0.4634146341463415
```

Fig. 15. Metrics for model-B(with sentiment). Stock: 'GOOGL'

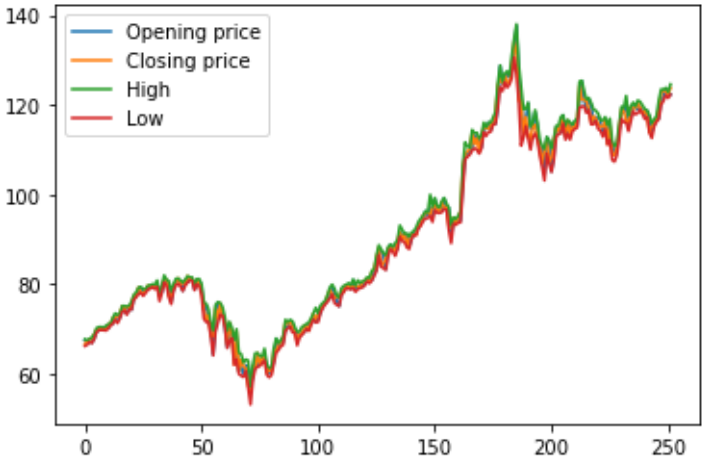


Fig. 16. 'AAPL' stock trend

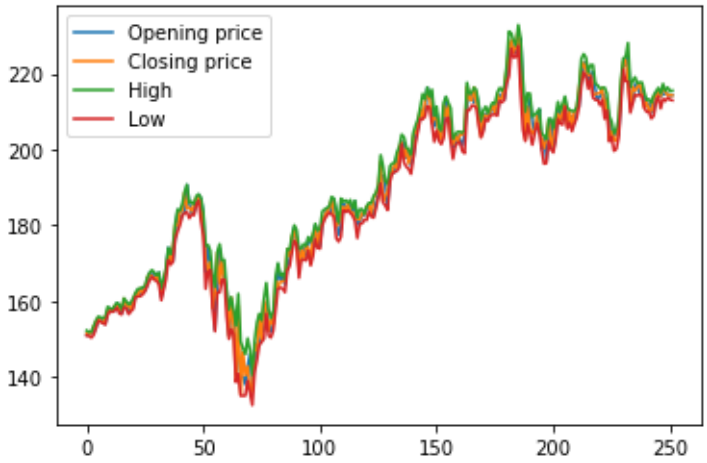
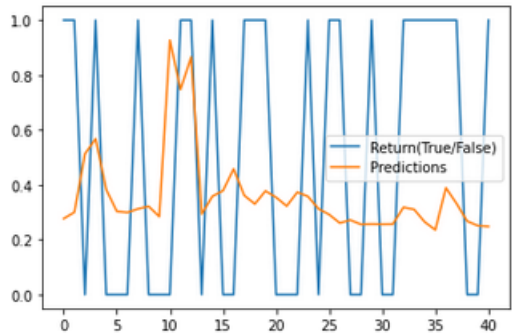
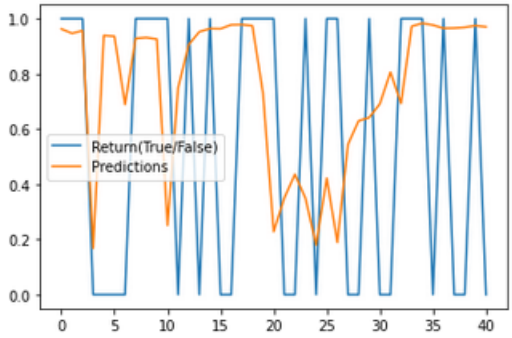


Fig. 19. 'MSFT' stock trend



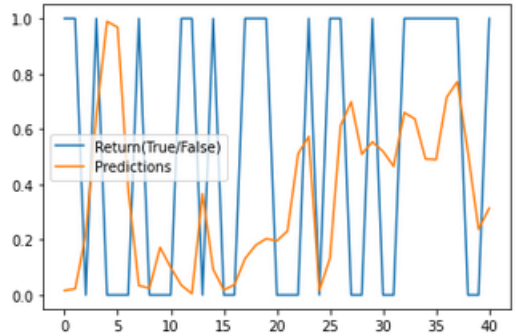

```
Precision:  0.55
Recall:  0.5214285714285715
F1 score:  0.4368131868131868
Accuracy:  0.5121951219512195
```

Fig. 17. Metrics for model-A(no sentiment). Stock: 'AAPL'



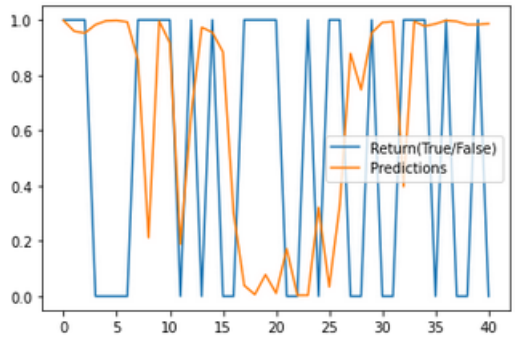

```
Precision:  0.4878472222222222
Recall:  0.4916267942583732
F1 score:  0.4576719576719577
Accuracy:  0.5121951219512195
```

Fig. 20. Metrics for model-A(no sentiment). Stock: 'MSFT'




```
Precision:  0.5166666666666666
Recall:  0.5154761904761904
F1 score:  0.5048309178743962
Accuracy:  0.5121951219512195
```

Fig. 18. Metrics for model-B(with sentiment). Stock: 'AAPL'




```
Precision:  0.4193121693121693
Recall:  0.42703349282296654
F1 score:  0.41682127396413104
Accuracy:  0.43902439024390244
```

Fig. 21. Metrics for model-B(with sentiment). Stock: 'MSFT'

VIII. Conclusion and future scope

From the above forecasting models it can be seen that stock data is non-deterministic and very random. However, market sentiment can have an impact on forecasting accuracy to a certain extent but it cannot be completely relied upon especially when there is not enough data. From the models A, and B, it can be observed that the model which uses sentiment can have slightly higher accuracy but it may not be true in all cases. Here, 3 out of 4 stocks(TSLA, GOOGL, AAPL) had slightly better forecasting results when sentiment was used. However, there is a lot of future scope in the domains of time series analysis, and natural language processing. With more and more data, and sophisticated model architectures, more accurate sentiment analysis and time-series forecasting can be achieved. However, the challenges of stochasticity in this domain makes it one of the prime areas of research.

References

[1]"Introduction to Word Embedding and Word2Vec", Medium, 2020. [Online]. Available: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. [Accessed: 18- Dec- 2020].

[2]"Word2Vec — TensorFlow Core", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/tutorials/text/word2vec. [Accessed: 18- Dec- 2020].

[3]"Natural Language Processing: Text Data Vectorization", Medium, 2020. [Online]. Available: https://medium.com/@paritosh30025/natural-language-processing-text-data-vectorization-af2520529cf7. [Accessed: 18- Dec- 2020].

[4]"What are Recurrent Neural Networks?", Ibm.com, 2020. [Online]. Available: https://www.ibm.com/cloud/learn/recurrent-neural-networks. [Accessed: 18- Dec- 2020].

[5]"A step by step Guide on Sentiment Analysis with RNN and LSTM", Medium, 2020. [Online]. Available: https://medium.com/@lamiae.hana/a-step-by-step-guide-on-sentiment-analysis-with-rnn-and-lstm-3a293817e314. [Accessed: 18- Dec- 2020].

[6]S. Upadhyaya, "Parallel approaches to machine learning—A comprehensive survey", Journal of Parallel and Distributed Computing, vol. 73, no. 3, pp. 284-292, 2013. Available: 10.1016/j.jpdc.2012.11.001.

[7]"Distributed training with TensorFlow — TensorFlow Core", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/guide/distributed_training. [Accessed: 18- Dec- 2020].

[8]A. SystemDS, "Apache SystemDS - Declarative Large-Scale Machine Learning", Systemds.apache.org, 2020. [Online]. Available: https://systemds.apache.org/. [Accessed: 18- Dec- 2020].

[9]"HorovodEstimator: distributed deep learning with Horovod and Apache Spark MLlib — Databricks Documentation", Docs.databricks.com, 2020. [Online]. Available: https://docs.databricks.com/applications/machine-learning/train-model/distributed-training/horovod-estimator.html. [Accessed: 18- Dec- 2020].