Stock Price Prediction of the S&P 500 Using Deep Learning Assisted by Sentiment Analysis

Niall Ryan
niall.ryan62@mail.dcu.ie

Oisin Duggan
oisin.duggan5@mail.dcu.ie

Cameron Ogiugo
cameron.ogiugo2@mail.dcu.ie

Sean Albert Dagohoy
sean.dagohoy2@mail.dcu.ie

***Abstract*— The Standard and Poor's 500 is the most famous financial benchmark in the world. This stock market index tracks the performance of the top 500 largest companies in the United states. It is well known that market sentiment is a good indicator of future market performance, and this paper aims to analyse the effect providing a LSTM model with natural language processed sentiment analysis has on its ability to predict stock prices.**

***Index terms*—Stock price prediction, Long-short term memory (LSTM), Sentiment Analysis, The Standard and Poor's 500 (S&P 500)**

I. Introduction

The stock market is a collection of markets where investors trade pieces of companies using various forms of analytics. Traditionally, there are two forms of using data to predict the future price of the market; fundamental analysis and technical analysis. Fundamental analysis evaluates a stock's intrinsic value based on economic factors like earnings and industry trends while technical analysis is purely concerned with using historical market data to predict price trends.

Much research has been performed into applying machine learning models on the stock market in order to predict future stock prices. Given deep learning's ability to recognise patterns in data, they are a natural choice. Classically, work in this field of machine learning has been mainly concerned with using models to analyse purely the historical data to make predictions about future price points, much akin to the way investors perform technial analysis on the data. When wanting to use real-world data to analyse the stock market, financial news headlines are an excellent reflection of the state of the market from a purely economic standpoint. This is why when looking to perform machine learning predictions with respect to economic factors, many researchers look to sentiment analysis to incorporate these factors into machine learning models [1].

The current state of machine learning is in a perfect position to be able to incorporate all of these factors to produce models that are well-equipped to predict the stock price. Development into recurrent neural network's ability to learn from historical data correctly has characterised itself into Long Short-Term Memory models [2], a specific type of recurrent neural network able to decide what information to retain without newer data disproportionally affecting the final model.

This paper aims to answer the question "What effect does providing a model with sentiment analysis data have on its performance?" This paper builds on unique pre-processing tactics and model creation explored in other papers to implement a high-quality model to answer that question, as will be discussed further.

II. Related Works

Given the natural profitability to be expected if a successful machine learning model is implemented, the research into this topic has been quite extensive. This paper builds on the work of others when deciding key factors in both the pre-processing of data and the specific model architecture in order to obtain the best results possible, while ensuring that we are still creating a model that performs as well as we can make it. Specifically, this research expands upon the work of two papers in particular, as expanded below.

A. *Paper One: Preprocessing, Architecture and Metrics*

In his paper, M. Jakhlal [3] explores several key concepts that resulted in impressive predictions. We built on a few ideas explored, specifically in the actions he did while preprocessing, and the specific model architecture. He also explored the effect the choice of optimiser and loss function has on your results, and inspiration was taken from the top performing model's selection.

When pre-processing his data, Jakhlal made interesting decisions that had very positive effects on the models he created. Firstly, he described how the selection of feature set was important, and that the only important one was closing price. A far more interesting concept explored was his process of expanding the knowledge that the model had access to by transforming every data point to have the context of the previous one hundred days of data, which allows the model to make much more generalised prdictions. He also explored the selection of hyperparameters of epochs, unit count, batch size and the best values he settled on to be the most perfomant. The final research from this paper that we built on was his extensive research into the effect that the loss function and optimiser has on your final model's performance. He showcased fourteen different combinations and we used his final discovery that the selection of Mean Squared Error (MSE) as the loss function and Adam as the optimiser resulted in the best perfomance to our advantage when building the model.

B. *Paper Two: Window Normalisation and Dropout Layers*

H. Hadhood [4] also provided very insightful research in his exploration of this field. Two key concepts that seemed to produce excellent results were his advanced normalisation techniques and the model architecture he used. Hadhood demonstrated a very successful form of normalisation of his data. As opposed to scaling the entire dataset to be understood properly by the model, he used a technique called window normalisation, where the dataset was segmented and every segment was normalised in respect to its own data. This is important when dealing with something like stock prices because there is a general trend upwards as time goes on. Because of this, if you perform normalisation on the entire dataset, earlier values are scaled close to zero and newer values are way too high.

When you normalise in windows, you capture the trend of the price in each window, but every window is properly normalised. This allows the model to be much more generalised to the general trend of the market instead of the overall performance.

The other concept implemented in the model that produced excellent results was the use of dropout layers. Dropout layers are a technique used in neural networks to combat overfitting. They achieve this by introducing randomness where some nodes in the layers are not transformed by the model every when a batch happens. This paper used a dropout rate of 0.2 which we used in our own model architecture.

C. *Inspiration Gained*

It is important to build upon the research of others when doing research of your own as this allows for collective advancement of the entire field. This is why when creating and implementing our own machine learning model, we made sure to heavily research our topic to understand the current standard of the field.

In terms of how we adapted these works into our own models, we coalesced the concepts explored in both papers to create a new hybrid of a model. From the first paper we implemented their units per layer and the specific loss function and optimser that they found to be the best. We also implemented most of their hyperparameters directly into the single-feature model as they were thoroughly tested and proven in his own paper. These included an epoch size of one hundred, a batch size of sixty and fifty-five units per LSTM layer.

The dataset preprocessing was inspired by both papers. We limited the selection of features to the one explored by M. Jakhlal to be the only relevant factor. When normalising the data, we combined both to produce a novel way of normalising your input data when dealing with stock prices. To do this we normalised the dataset into windows, like carried out in Hadhood's paper. However, instead of shifting the window completely over to properly section the dataset, we shift each window by only one day. By doing this, we obtained M. Jakhal's way of expanding the features to give extra context while also normalising it in respect to everything which allows the model to make general assumptions without a general trend upward.

We also took inspiration from H. Hadhood's technique of dropout layers with a 20% dropout rate in between every single LSTM layer, which he had proven to completely negate the risk of overfitting while remaining accurate.

III. Methods

A. *Datasets*

This research utilises four datasets in tandem for an ensemble method (See details in section B. Pre-Processing).

1) *S&P500 Prices:* A history of the daily S&P500 stock index provided by the NASDAQ website [5]. This dataset contains the last 10 years of data and the time span from the data we use is 22/01/2014-22/01/2024. This data has 2,532 rows (days) and contains 5 features (Date, Close, Open, High, Low). This is open-source data and our version was downloaded on 15/01/2024.

2) *Labelled News Headlines:* Various labelled news headlines [6]. This dataset contains data from 2009-02-14 to 2020-06-11. This dataset has 3 features (Date, headline and stock it pertains to). As each row represents a unique headline, this dataset has 3 columns and 843062 rows. This is open-source data and we used a version downloaded on 15/01/2024. The features selected to be used from this dataset were Date, Headline and Stock (The closing price of the index on that date.)

3) *Kaggle Stock Headlines:* Labelled stock headlines provided by kaggle [7]. This dataset contains headlines about stocks each of which are labelled Positive, Negative or

Neutral. This dataset contains data which is undated. This dataset has 3 features (Date, headline and label). The possible labels are again positive, negative and neutral. As each row represents a unique headline, this dataset has 2 columns and 26000 rows. This is open-source data and we used a version downloaded on 15/01/2024. The features selected to be used from this dataset were Headline and Label.

4) *Hugging Face Financial Phrases:* Financial phrases which are labelled provided by Hugging Face [8]. This dataset also contains headlines about stocks each of which are labelled Positive, Negative or Neutral. This dataset contains data which is undated. This dataset has 3 features (Date, headline and label). The possible labels are again positive, negative and neutral. As each row represents a unique headline, this dataset has 2 columns and 4850 rows. This is open-source data and we used a version downloaded on 15/01/2024. The features selected to be used from this dataset were Headline and Label.

B. Pre-Processing

We created two models, a 'Market Sentiment Model' and a 'Stock Market Prediction' model. (See Model Architecture) The purpose of the Market Sentiment Model was to interpret news headlines around the S&P500 and included companies and produce a 'sentiment' value to later be fed into the Stock Market Prediction model.

1) *Pre-Processing Data for the Market Sentiment Model:* Three datasets were used to train our Market Sentiment Model (see details in Model Architecture). These datasets were the previously defined, 'Labelled News Headlines', 'Kaggle Stock Headlines' and 'Hugging Face Financial Phrases'.

As the Hugging Face Financial Phrases was a text (.txt) file, it was converted to a pandas dataframe similar to all other datasets. This was then merged with the Kaggle Stock Headlines dataset and `pd.replace()` was used to convert all labels from 'positive', 'neutral' and 'negative' to 2, 1 and 0 for the model to understand.

The dataframe was reformatted with only positive and negative values selected. As there was an uneven distribution in the Kaggle Stock Headlines dataset - only 4803 positive, negative and neutral exemplars were selected to ensure the Market Sentiment Model did not produce skewed results.

As this is not sequential data, the function `pd.sample()` was used to randomly shuffle all data in the merged dataset. After this was performed, we utilised a 75%/25% training/testing split.

After the Market Sentiment Model was trained on these datasets, we performed inference on the 'Labelled News Headlines'. This was performed as the dataset contained 5 significant influencers (companies) on the S&P500 price (AAPL, NVDA, GOOGL, BRK and AMZN).

We then processed this outputted data, utilising a hashmap (python dictionary) to get rid of any duplicate sentiment for the same company on any same day and ensuring the sentiment provided was the average of any duplicate values. Finally, converting this hashmap to a `pd.DataFrame` instance.

To complete the dataset for training our Market Prediction Model, we then utilised the S&P500 dataset, combining it with the output of our Market Sentiment Model. This left us with a dataset of 1,170 rows (days) and 3 columns (Date, Sentiment and Closing Price). Notice how we lost 1,362 days from the original S&P500 dataset. This was due to the Market Sentiment datasets having clashing and generally smaller date ranges.

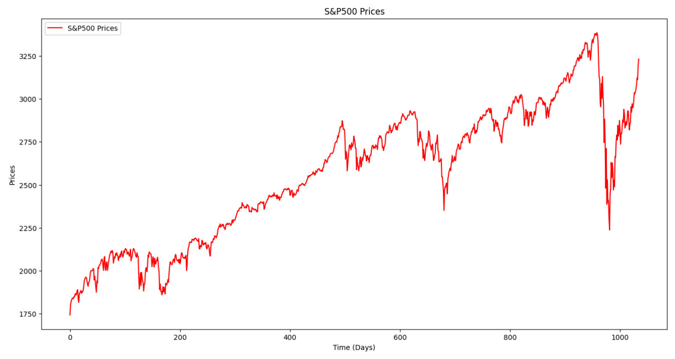


Figure 1: Final Dataset 'Closing Price' Column by Date

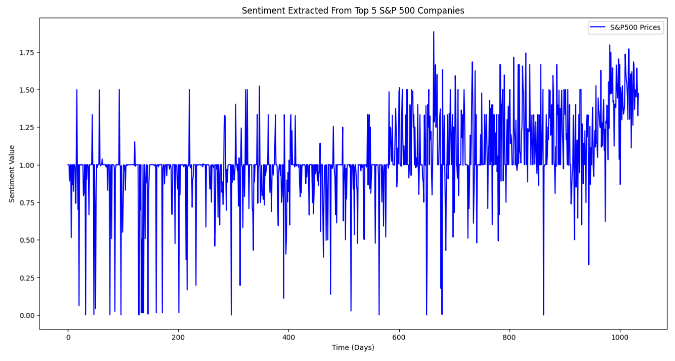


Figure 2: Final Dataset 'Sentiment' Column by Date

2) *Pre-Processing Data for the Stock Market Prediction Model:* Once we had the constructed dataset provided by combining the results of inference from the Market Sentiment Model and the actual S&P500 dataset we performed the 'Window Normalisation' Technique.

This included `shift`, `window_size` and `label_size` hyperparameters. We kept `label_size` to 1, as this was how many days ahead we wanted our model to predict - and the aim of our research is only to perform 'next-day-prediction'. We did experiment with various values of `shift` and `window_size`.

We performed windowing on both features ‘Closing Price’ and ‘Sentiment’ - ensuring to have their respective dates line up with eachother. This ensured better distribution among values (See Fig. 3 and 4.) and gave the Stock Market Model `window_size` days of context when predicting `label_size` days ahead.

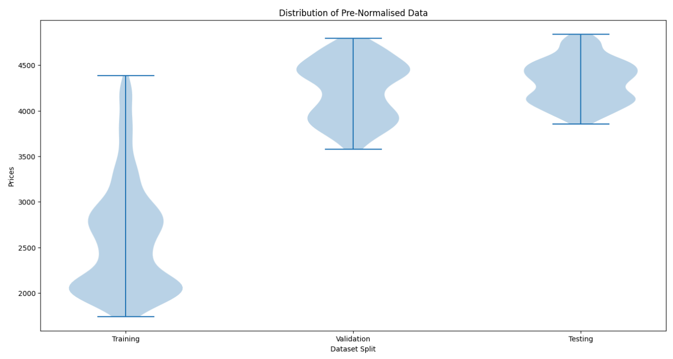


Figure 3: S&P500 ‘Closing Price’ column distribution by data split

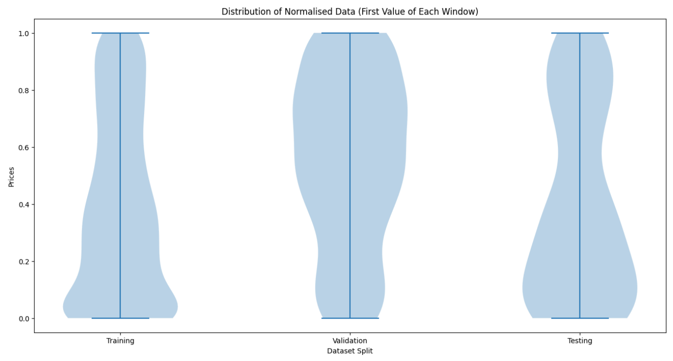


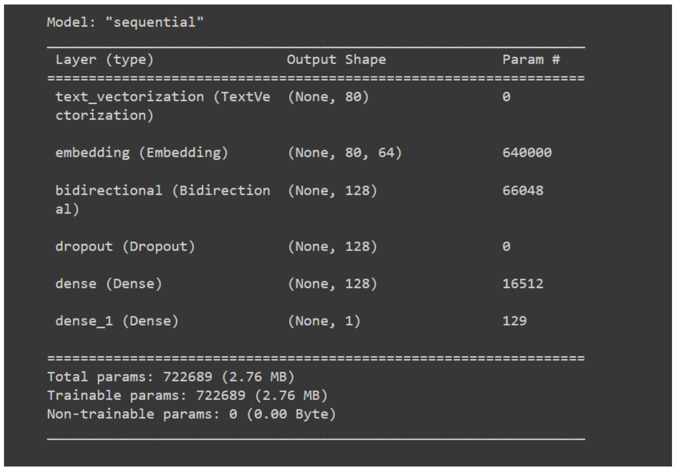
Figure 4: S&P500 ‘Closing Price’ column distribution after window normalisation

We then performed min max scaling on the ‘Closing Price’ windows to convert the values to a range of 0 - 1. We used SciKit-Learns MinMaxScaler for this [9].

In order to denormalise data from SciKit-Learns MinMaxScaler, we needed to pair the scaler with each window it was used to normalise. We created a `NormalisedPriceWindow()` object to help abstract this information.

C. *Model Architecture*

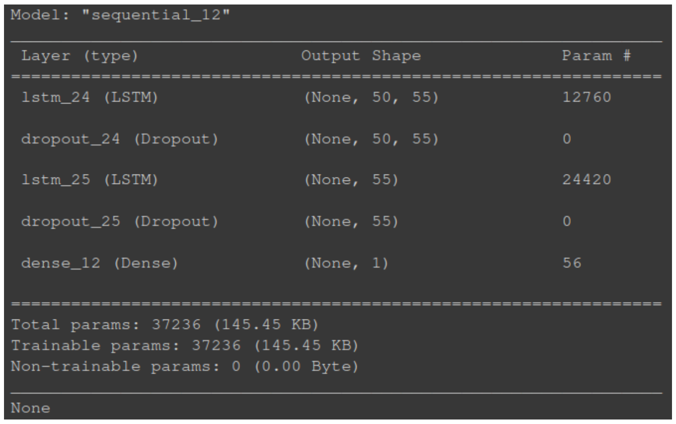
1) *Market Sentiment Model:* The Market Sentiment Model is a bidirectional Long-short Term Memory model (LSTM), which is a recurrent neural network used on natural language processing[10]. The Keras python library was used to create the bidirectional LSTM model. The model consists of 6 layers; a TextVectorization layer which is a preprocessing layer that processes and extracts meaning from textual data, an embedding layer which helps add meaning into each word, a bidirectional LSTM layer which has 64 neurons, a dropout layer at a rate of 0.2 to help fight overfitting, a dense layer with 128 neurons which has a “relu” activation function, and finally, an output dense layer which has a “sigmoid” activation function. The loss function used was BinaryCrossentropy and Adam for the optimisation function. This model was trained with 10 epochs. This model was adapted from TensorFlow documentation. [11]



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVe  (None, 80)                0
 ctorization)

 embedding (Embedding)       (None, 80, 64)            640000

 bidirectional (Bidirection  (None, 128)               66048
 al)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 128)               16512

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 722689 (2.76 MB)
Trainable params: 722689 (2.76 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 5: Market Sentiment Model architecture

2) *Stock Market Prediction Model:* The Market Prediction Model is a standard LSTM model, which is a form of recurrent neural network (RNN) which introduces the use of memory gates [2]. This was built using Tensorflow’s Keras and has hyperparameters `batch_size`, `epochs` and `units`. We utilised two LSTM layers with Dropout layers at the rate of 0.2. The loss function used was mean squared error and the optimiser used was Adam.



```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_24 (LSTM)              (None, 50, 55)            12760

 dropout_24 (Dropout)        (None, 50, 55)            0

 lstm_25 (LSTM)              (None, 55)                24420

 dropout_25 (Dropout)        (None, 55)                0

 dense_12 (Dense)            (None, 1)                 56

=================================================================
Total params: 37236 (145.45 KB)
Trainable params: 37236 (145.45 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Figure 6: Market Prediction Model architecture

IV. Results and Discussion

A. *Performance of Single Feature Market Prediction Model on S&P500 Dataset*

Evaluation of the Market Prediction Model was performed before introducing the combined dataset with two features 'sentiment' and 'closing price'.

We used the original S&P500 dataset containing 2,532 rows (days) and hyperparameters inspired from M. Jakhlal [3] and received performant metrics. (See table 1. and figure 7.)

	MSE	**RMSE**	**R-Squared**
Training	0.0107	0.1034	0.8373
Validation	0.0112	0.1058	0.8302

Table 1: Evaluation Metrics of the Market Prediction Model with a feature (closing price) on the S&P500 dataset

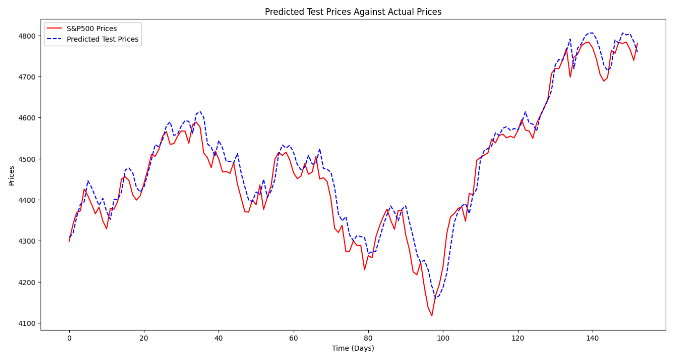


Figure 7: Predicted values plotted against actual values of the S&P500 testing dataset split.

Once we had built a high-performing model and confirmed absence of overfitting, we introduced the combined dataset with an additional feature (sentiment).

B. Performance of Two Feature Market Prediction Model on Combined Dataset

At first, we utilised the same hyperparameters used one the single feature S&P500 dataset. However, with our dataset substantially reduced to 1,170 rows from 2,532 days, we encountered issues with the window sizes and our data splits.

We maintained the hyperparameters of the single-feature experiment. However, as our data split was still 75%/15%/10% for training/validation/testing, this left us with 776/155/104 rows in each split respectively.

When we employed windowing on this data, we were left with 676/55/4 windows on each dataset split. This wouldn't work as our window sizes were simply too large, and training this model provided unsatisfactory results. (See Table. 2)

	MSE	**RMSE**	**R-Squared**
Training	0.0104	0.1019	0.8642
Validation	0.0059	0.0766	−1.6742

Table 2: Evaluation Metrics of the Market Prediction Model with two features (closing price, sentiment) on the combined dataset with a window size of 100.

So to combat this, we halved the window size to 50, and reduced the batch size to 30 to get better performance and more windows. (See figure 8 and table 3).

	MSE	**RMSE**	**R-Squared**
Training	0.0159	0.1262	0.8098
Validation	0.0152	0.1233	0.7562

Table 3: Evaluation Metrics of the Market Prediction Model with two features (closing price, sentiment) on the combined dataset with a window size of 60 and batch size of 30.

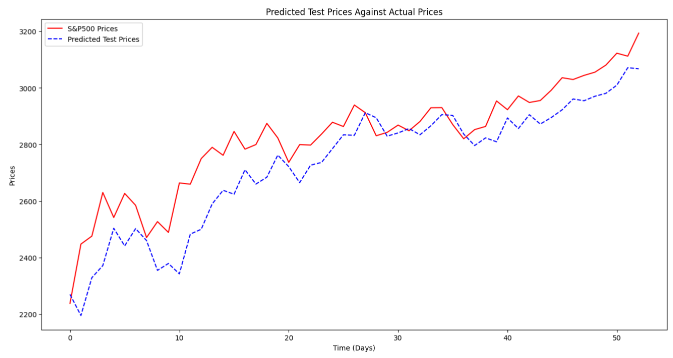


Figure 8: Predicted values plotted against actual values of the combined testing dataset split

After experimenting with multiple values, we discovered that we couldn't actually find any impactful improvement over the parameters set (see table 4) for the combined dataset with two features.

Hyperparameter	Value
Window Size	50
Shift	1
Label Width	1
Train Split	75%
Valid Split	15%
Test Split	10%
Batch Size	30
Epochs	100
LSTM Layer Units	55

Table 4: Optimal Hyperparameters for two feature combined dataset prediction

We then applied these exact same hyperparameters to the same combined dataset, but without the sentiment feature - leaving only the closing price feature. This was performed to compare performance of a model without sentiment to see if it was actually positively influencing the performance of the model's predictions or not.

Interestingly, we actually did perform better on the combined dataset when we omitted the sentiment feature. (See table 5, figure 9)

	MSE	RMSE	R-Squared
Training	0.0161	0.1267	0.8082
Validation	0.0123	0.01108	0.8030

Table 5: Evaluation Metrics of the Market Prediction Model with one feature (closing price) on the combined dataset with optimal hyperparameters

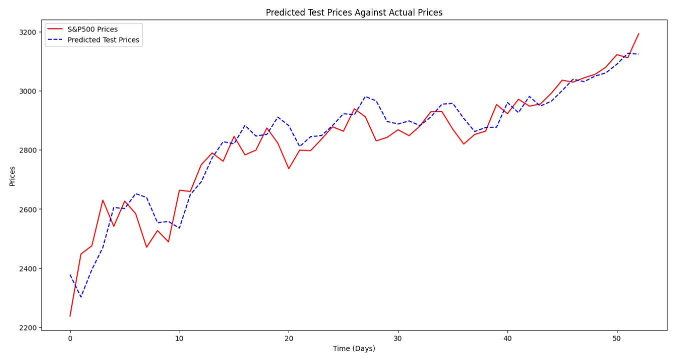


Figure 9: Predicted values plotted against actual values of the combined testing dataset split with one feature and optimal hyperparameters

This lead us to believe that the Market Prediction Model simply saw the sentiment feature as noise, and offered no further useful information.

V. Conclusion

It is an interesting observation that our Stock Market Model prediction proved to be better performant on the combined dataset when the sentiment feature was ommitted.

This leads us to believe that our methods for obtaining market sentiment aren't effective enough to provide insight into the S&P500 market's movement

To answer the paper's research question, "What effect does providing a model with sentiment analysis data have on its performance?", we would state that although our research and experiments conclude that it is only seen as noise, and make model performance worse, we believe our methods for extracting market sentiment could be improved.

We define our answer as inconclusive, as we firmly believe that including a market sentiment would have a positive influence on S&P500 market price predictions, but our methods for generating market sentiment aren't satisfactory.

We believe our research was stunted by our small timeframe. Had we enough time to experiment with different methods of extracting market sentiment, we believe we may have found far more satisfactory methods, and produced better results.

Additionally, had we more time, we could have performed far more experimenting with model hyperparameters, as although we did spend substantial time experimenting with them, it wasn't conclusive and there were more combinations we would have liked to try.

More obstacles included the fact that we were so limited with our dataset sizes for our Market Sentiment Model. The Market Prediction model trained and tested solely on the S&P500 dataset of 2,532 datapoints outperformed all other models - we believe this is due to it having far more data. Our combined dataset, as we were so restricted by our market sentiment data, only contained 1,170 datapoints. We also encountered many difficulties with splitting and window sizes due to this reduction. Had we more time, we could find better methods and datasets to maintain more data for training, testing and validation.

A. *Appendix*

1) *A. Google Colab Files:* Single-Feature Market Prediction Model: https://colab.research.google.com/drive/1oF6tJkmXdTrBst9v_pDMW_MQRrQd8iJB?usp=sharing

Double-Feature Market Prediction Model: https://colab.research.google.com/drive/1bba9-aO20smYvDBzoWFVA0GFPYa1TpcV?usp=sharing

Stock Sentiment Model: https://colab.research.google.com/drive/1qNKdtbsqW92KW5yZYCoT6DjhVbWSQyeq?usp=sharing

References

[1] V. S. Pagolu, K. N. Reddy, G. Panda, and B. Majhi, "Sentiment analysis of Twitter data for predicting stock market movements", in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 2016, pp. 1345–1350. doi: 10.1109/SCOPES.2016.7955659.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] M. Jakhlal, "Stock Price Prediction Using Deep Learning", 2022, [Online]. Available: https://arno.uvt.nl/show.cgi?fid=157645

[4] H. Hadhood, "Stock trend prediction using deep learning models LSTM and GRU with non-linear regression", 2022.

[5] NASDAQ, "SPX Historical Data", 2024, [Online]. Available: https://www.nasdaq.com/market-activity/index/spx/historical

[6] M. Aenille, "Daily Financial News for 6000+ Stocks", 2020, [Online]. Available: https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests

[7] Kaggle, "Labeled Stock News Headlines", 2023, [Online]. Available: https://www.kaggle.com/datasets/johoetter/labeled-stock-news-headlines

[8] P. Malo and A. Sinha, "Financial Phrasebank", 2014, [Online]. Available: https://huggingface.co/datasets/financial_phrasebank

[9] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] E. Zvornicanin, "Differences Between Bidirectional and Unidirectional LSTM", 2023, [Online]. Available: https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm

[11] TensorFlow, "Text classification with an RNN". [Online]. Available: https://www.tensorflow.org/text/tutorials/text_classification_rnn