Incorporating Media Bias in Sentiment Analysis for Improved Stock Prediction Using Neural Networks

Rishabh Vemuru
rish.vemuru@gmail.com

ABSTRACT

In this work we examine the effects of incorporating media bias and its impact on stock prediction. Sentiment analysis finance articles in the field give the model actual news data and from there it forms a sentiment score into an LSTM. There is no indication about what news source this came from and there is a flaw in this. There is media bias in this as certain news outlets want to push a certain narrative which could have no meaning on the actual prediction, we plan to incorporate this in the predictions in order to ensure that this media bias is accounted for and thus get more accurate predictions. With everything incorporated this model has a mean absolute percent error close to 1 percent.

In this work we examine the effects of incorporating media bias and its impact on stock prediction. We achieve results that are state of the art in academia for stock price prediction that include the use of sentiment analysis. We research and present findings of alternate techniques that can achieve similar results and finally incorporate media bias for the sentiment. The findings strongly suggest that sentiment analysis should incorporate the source of the information as not all sources are created equal in quality and in bias.

INTRODUCTION

Stocks data reflects the economic environment, making it an especially important area of study, for not just predicting but also the understanding of the economic status of the world. The availability of publicly available stock data, its volume and complexity creates opportunities for the application of more advanced analysis. The stock market has about 5,600 unique public companies, each of these has individual historical data and many different patterns which we can pick up on.

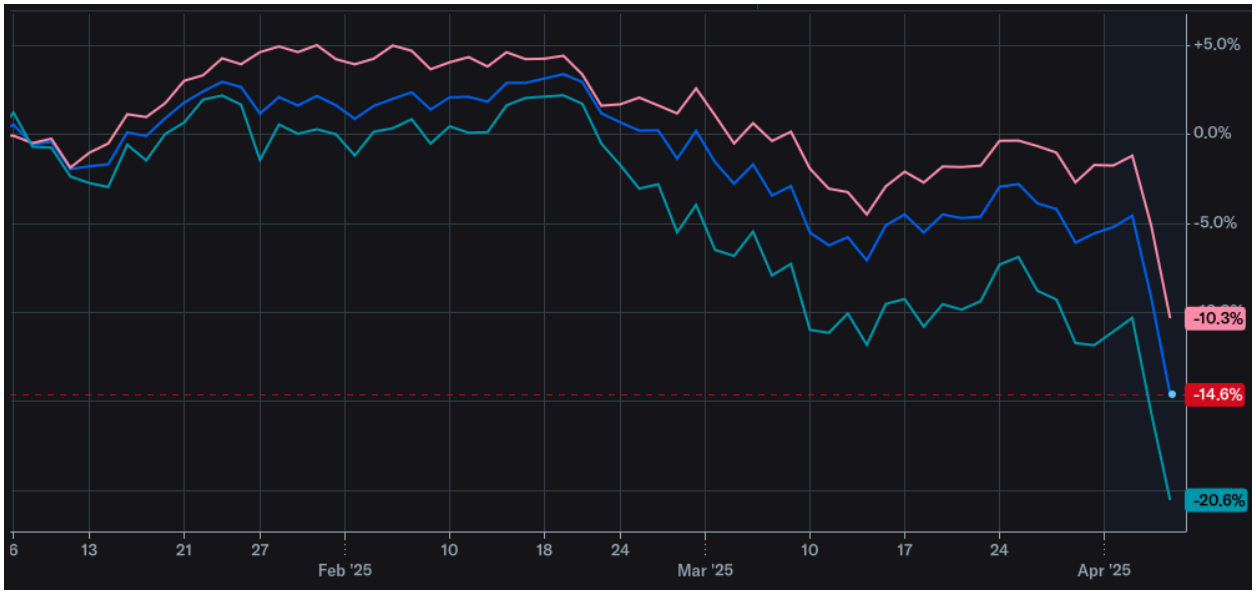

Fig1. This chart shows the three major stock market index returns in the us

Artificial intelligence has been proven to be one of, if not the most influential tool used in financial research recently. In the past machine learning, deep learning and even natural language processing allowed for the detection of patterns and trends in the market which would be hard for humans to find and react quickly on. For example sentiment analysis models can process large amounts of news data and have been incorporated in daytrading, and many other aspects in the financial sector.

The adoption of AI in finance is growing significantly because the increase in prediction accuracy is generating higher returns for both individuals and institutions. By leveraging the publicly available large datasets, AI algorithms improve efficiency in the allocation of capital and assist with decision making.

There are a lot of factors which can influence a stock's trajectory, because of this it is important to incorporate a significant amount of data so the model can pick up on these small factors and their correlation. One important factor influencing stock price movement is news sentiment. News sources inherently contain bias because they control which stories are published and how information is framed, often influenced by financial or institutional incentives.

Part of this reason is because all news sources have inherited biases because they can control what articles are published and to some extent they have financial incentive.The research question we are trying to answer is first what AI model is the best for stock market prediction and secondly if the implementation of news sentiment will benefit the predictions. If so we plan to implement media bias into predictions.

The reason why News is such a key feature in this stock prediction is because it involves the opinions of financial experts. If a financial expert mentions how a stock has a chance of rising then the chance that the stock goes up is higher. This is also the case for Earnings reports which have significant results on the

stock price. It is not possible to put the results from earnings reports purely into numbers which is why we must rely on others' sentiments to then predict its effects, for example if an article says “Did significantly worse then earning report predictions expected” then it would give a negative sentiment to the model.

The news articles were then stored in a JSON file and the same file was then used by all models incorporating news. This was done in order to ensure consistency between the models made.

RELATED WORKS

This paper presents a review of stock prediction research between 2015–2023 that uses neural network models. It talks about the architectures of the models and highlights how data-driven neural networks have been applied in stock price and trend prediction. The authors identify challenges such as data preprocessing, feature engineering, hyperparameter tuning, and overfitting of the model. They conclude that while neural networks show promise for modeling the non-linear dependencies, there is a gap across markets and in replicability of results (Bao et al., 2024a). This article proposes an augmented Long Short-Term Memory (LSTM) architecture which integrates Symbolic Genetic Programming to improve stock price change forecasting. The authors use this method with several stocks, using historical price and technical indicators, demonstrating improved performance compared to baseline LSTM. The hybrid model captures complex dependencies and non-linear relationships in time series data. Limitations include overfitting and large computational resources (Bao et al., 2024b). This study makes a deep learning model to predict five stock market trend patterns: upward, downward, double top, rounded bottom, and rounded top. It integrates Explainable Artificial Intelligence methods to interpret model decisions and aims to make predictions easier for financial stakeholders. The authors show that the model achieves strong classification accuracy across the pattern types indicated above and that XAI outputs help identify which features drive the predictions and their correlations. Challenges noted are distinguishing small reversal patterns and the sensitivity of performance to feature selection (Author(s), 2024). This article proposes a multimodal prediction framework that combines FinBERT-based financial sentiment extraction with statistical indicators to predict stock price movement. The methods take sentiment signals from financial news headlines with FinBERT and combine them with price and volatility-based features, which are then input into a decision tree classifier (Ruan & Jiang, 2025).

MATERIALS AND METHODS

The dataset used in this was obtained by using the Yahoo Finance API which includes daily historical data for all stocks, we used the closing prices for stocks of the top 100 valued companies. In addition, for models using the sentiment analysis, daily sentiment scores were extracted from a dataset of news articles using FinBert, these news articles were obtained using the Finnhub API, which aggregates financial news from multiple sources The prices used in this were closing prices for the last five years. The close prices are the prices of each stock at market close at 4:00 pm Eastern Standard Time.

The sentiment scores used were obtained by first taking up to fifteen articles a day per company using the Finnhub API. This gave the timestamped financial news associated with the publicly traded companies that we were searching for.

For each company, up to fifteen articles per trading day were retrieved. Articles were required to explicitly mention the company name or ticker symbol and be published within the same day as the corresponding stock data.

If more than fifteen articles were available for a given day, the fifteen most recent articles were selected. If fewer than fifteen articles were available, all available articles were used.

No manual filtering of articles was applied in order to avoid introducing human bias into the dataset. News sources were not restricted and vary across companies depending on availability.

Then from there the articles are run through hugging faces FinBert model which returns a separate positive, negative and neutral sentiment score for the article in the range of [0, 1]. We then subtract the negative sentiment score from the positive sentiment score, resulting in a value that ranges between [-1, 1] with -1 being the most negative sentiment and 1 being the most positive sentiment. Previous methods used an average sentiment score of the news articles as a feature. We create a news source bias adjusted sentiment score feature as described later.

To predict the next day’s stock close price, the models used the last 8 days' stock close price as input features.

To evaluate models' performances it is important to incorporate testing and training data. For all our models we took the input vectors and the output vectors and divided them with their corresponding set into a 70-30 split. The training set was used to help fit the model parameters however the testing set was used solely for evaluation. Without this split the evaluation would be misleading.

During this we will be training with 3 different tests, a baseline test where the models are trained on 100 stocks and are tested on the same 100 stocks. This is in order to recognize common pattern recognition which we would not see if we had significantly less amount of data. Then we will test and train a model on 1 stock 100 times, these results will show patterns that the model comes up with for each individual stock. From there we will take the top performing models and incorporate sentiment and news bias in the predictions in order to see if the results are better.

Testing models:

LSTM (Long Short-Term Memory):

The Long Short-Term Memory model is a type of neural network made to handle sequences. It contains cells with input, output, and forget gates that enable it to forget and retain information. This also makes it very effective for financial sequences like stock prices because it can see long term and short term trends and their impact. This allows them to model patterns based on momentum or patterns seen in the past. However it takes large amounts of data to train upon as it needs to recognize patterns.

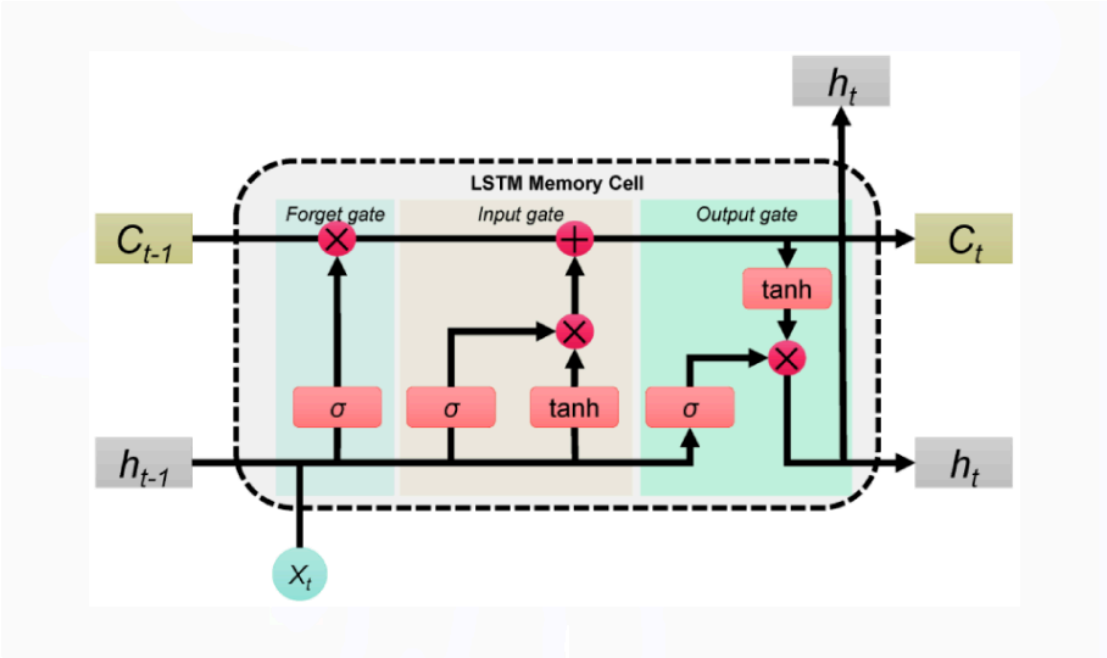

Figure 2: Long Short-Term Memory architecture.

Feedforward Deep Neural Network (Deep Neural Network)

In this paper, when we refer to a neural network or a deep neural network, we are referring to a fully connected feedforward deep neural network. A deep neural network is a type of artificial intelligence model. It consists of many layers which consist of neurons. In the beginning it takes the features (inputs) given to it and performs a matrix multiplication on the vector, from there the vector which results from this are the values of the neurons in the proceeding layer, all of these values have an activation function run on them (examples are sigmoid and Rectified Linear Unit functions), this is so that the model can recognize nonlinear dependencies. From there this process continues until the final matrix multiplication is done to obtain one value which is the result. The numbers in all these matrices are called the weights and the model finds the ideal values for this using gradient descent and back propagation on the training set.

Gradient descent is an algorithm used to attempt to minimize a neural network's loss, which measures how far off the predictions of the neural network is to the true value. During training, the network makes predictions, and the loss function calculates the error of the prediction. Gradient descent computes the slope of this loss compared to each weight in the network. Each weight is adjusted in the opposite direction of the slope because that direction reduces error using the formula $w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$, where η is the learning rate which controls how big each adjustment is. Over many iterations, the process gradually reduces the loss. The learning rate of the model is something which can be tuned, by default it is about 0.001, depending on the parameters it can also decrease as time goes on or adapt to the results.

Backpropagation is the method that enables gradient descent to work in deep networks. It applies the chain rule of calculus to calculate how each change in the neural network weight affects the error. First, the model calculates the error at the final layer. Then, backpropagation moves backward through the network, distributing this error and getting gradients for each connection. These gradients show how much each weight contributed to the overall error. Once these gradients are known gradient descent is done to optimize the weights.

To also ensure that the model does not overfit on the data there are two hyperparameters we can utilize to try to prevent this. The alpha parameter in the neural network gives higher loss to models with a higher sum of weights, this is implemented because models which are hyperoptimized on training data to the point where it significantly hurts its testing data performance generally have higher weights so that they can get more specific results. Incentivizing lower weights means that it will not try to focus on having perfect results in testing and can be more applicable to the training data. Another hyperparameter which could be used is the dropout parameter. This parameter blocks out a percentage of the connections between neurons randomly during training, this means when training the model does not know which connections will be removed and it can not overly tune every connection so that the result is perfect. However in practice all of the connections are used, the problem is if the value is too high then the model's results on the testing data will also start to degrade which is why it is always a necessity to tune these hyper parameters to ensure a better performance of the models.

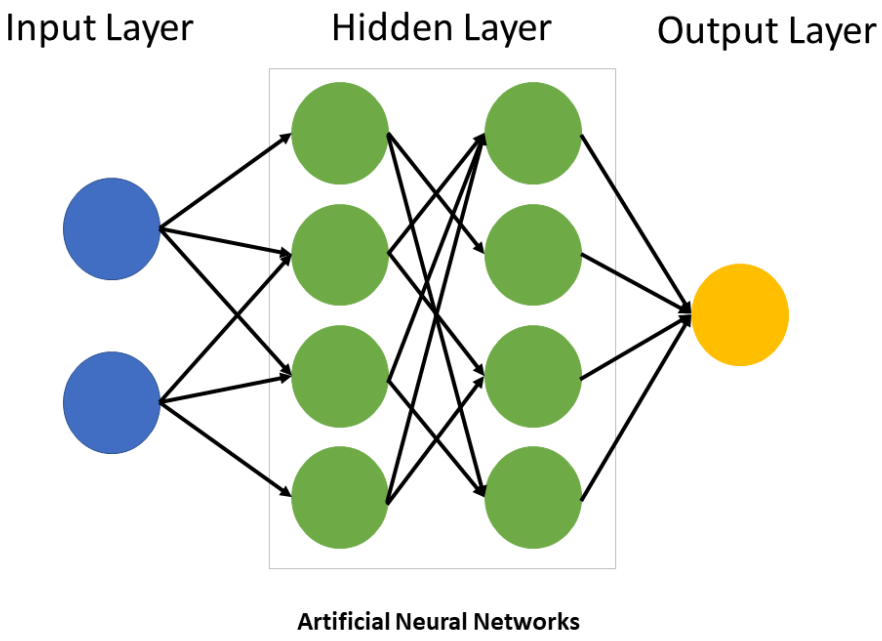

Figure 3: Deep Neural Network architecture.

Linear regression

Linear regression is a method used to find the best possible linear equation for prediction. This utilizes the least squares algorithm. $\hat{\beta} = (X^{\top}X)^{-1}X^{\top}y$ where β is a vector of regression coefficients, where X is the matrix of input features and where y is the vector of observed outputs. This uses this equation to minimize the MSE. This model can not predict nonlinear dependencies.

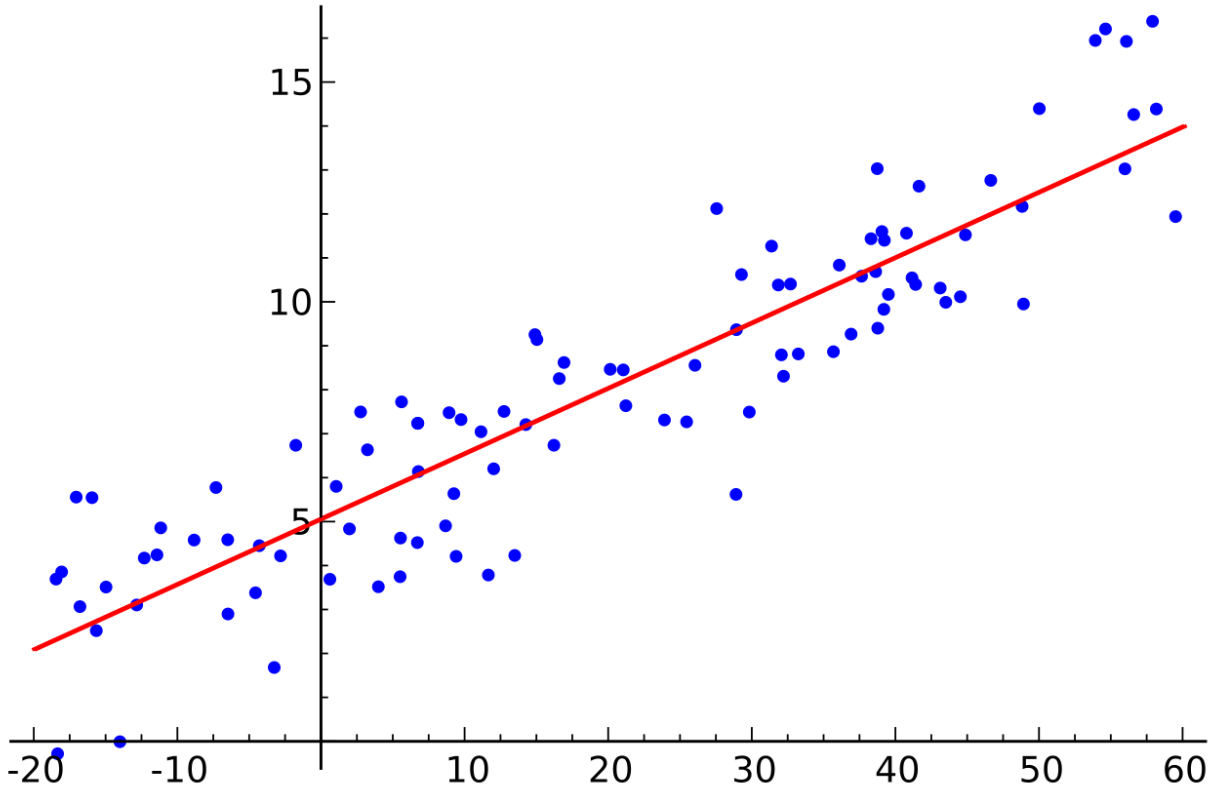

Figure 4: Linear regression example

Decision tree:

A decision tree is a supervised learning algorithm used for classification and regression. It works by incorporating many binary decisions about the features to try and find the value of what is wanted to be predicted. From each binary decision the result will either go to a leaf node which is when it predicts a result, or it will go to another binary decision. This model will try to maximize results on the training data which is why it is very prone to overfitting.

However there are some hyperparameters that are able to incorporate and edit in order to ensure that overfitting is minimized. For example the Max Depth hyper parameter allows us to edit the amount of maxim amount of binary decisions made before it reaches a leaf node, this stops the Decision Tree from becoming too long and hyper focusing on specific examples, this also allows it to not have a leaf node for every value in the training data so it does not overfit. The min-samples leaf hyperparameter also allows us to prevent overfitting because it is the minimum number of samples required to be at a leaf node. This stops the Decision Tree from predicting without knowing too much information. For example early on if it asks a specific question and then immediately jumps to a conclusion that is a sign of overfitting because that was the best option to do in the training data, setting this hyper parameter was higher enables us to stop this from happening early on and ensure that if this happens it happens in later layers of the decision tree where the model has gained more information about it.

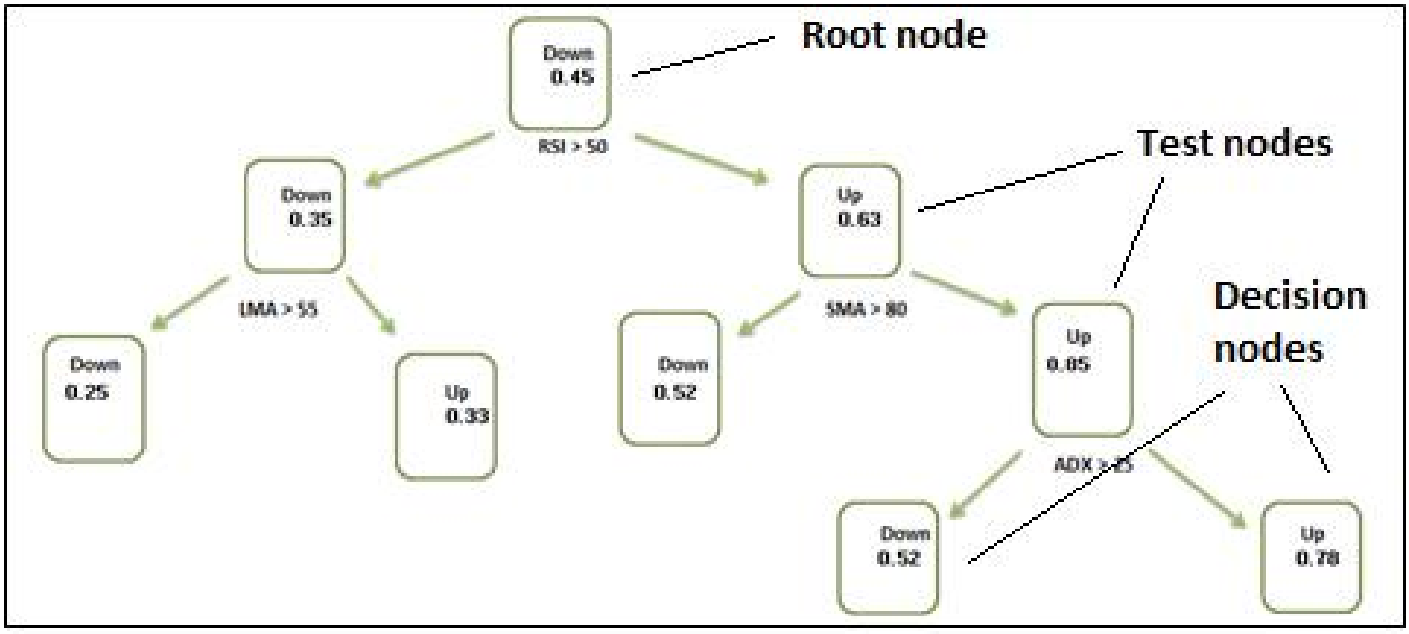

Figure 5: Decision Tree stock example

Random Forest:

A random forest is a machine learning algorithm which uses multiple decision trees. The training dataset is split into random sub groups. From there each of these groups is given to an individual decision tree. From there the model takes the average of the decision tree's results. This approach leverages the diversity of many decision trees to make random forests more accurate and stable than individual decision trees. This also prevents overfitting as it takes the average of multiple models each trained on different data.

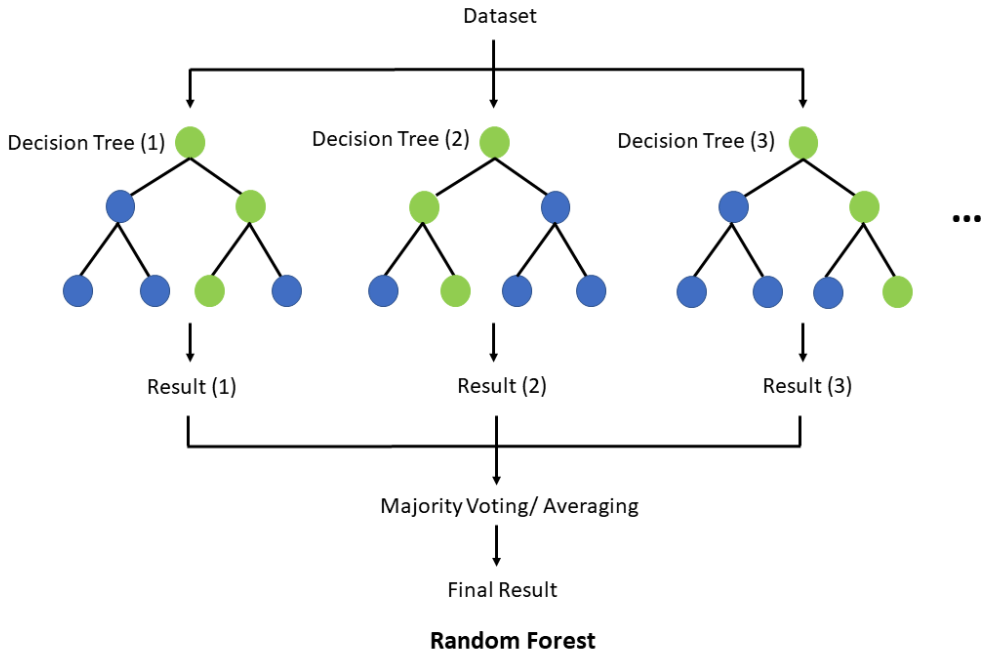

Figure 6: random forest architecture

Weighted average model:

A weighted average model is a technique that combines the predictions of multiple individual models to improve overall accuracy. The model takes the outputs of our other five models such as LSTM (Long Short-Term Memory networks), DNN (Deep Neural Networks), linear regression, decision trees, and random forests and evaluates each model based on its Mean Absolute percentage error. Finally it would weight each according to its accuracy and find the average. Since this takes the accuracy of each of the models into consideration, it will not be heavily influenced by models with lower accuracies.

$$\hat{y}_t = \sum_{k=1}^{K} w_k \, \hat{y}_t^{(k)}$$

Figure 7 weighted average model algorithm

In this image we see that the calculation is the weighted sum of the predictions with the weights being w

Updating Weighted average model:

Because the errors of these models are not known during testing it is imperative that we incorporate these testings from a standpoint which assumes that we do not know these values. Because of this we need a weighted average model which updates for each value it is tested on. What this means is that we have to recompute the weights for every time model is tested. The downside to this is that during the beginning of testing, the weights are either not known or are skewed significantly due to lack of predictions to base weights on.

$$\hat{y}_{t+1} = \sum_{k=1}^{K} w_t^{(k)} \, \hat{y}_{t+1}^{(k)}$$

Figure 8: updating weighted average model equation
Here we see that the result constantly updates with each iteration and the weights also update.

Model Configuration:

A detailed description of each model configuration is provided below.
LSTM Architecture:

The Long Short-Term memory model consists of three stacked layers followed by two fully connected layers. Each of these stacked layers has 64 units. The first and second LSTM layers return sequences, and the third outputs a single vector. The input to the model is a sequence of length 8 with one feature per a timestep, which is the stock closing price.

After the LSTM layers a dense layer with 32 neurons and a ReLU activation function was used, then there was a final dense layer with a single output which was the predicted stock price.

The model used the Adam optimizer with the default parameters and used Mean Absolute error as the loss function. This model then trained for 15 epochs with a batch size of 32. We found that this was optimal for performance to not overfit.

Deep Neural Network:

The DNN model is implemented by having a fully contacted architecture that takes the past 8 days of the stock closing prices.

The model consists of all dense layers. THe input layer is followed by three hidden layers each with 64, 32 and 16 neurons respectively. All the layers used a ReLu activation function. These layers help the model learn nonlinear relationships between the price and the next day's closing price.

The final layer is one single neuron with a linear activation function.

The model is trained using the Adam optimizer with the following hyper parameters:
learning rate = 0.001
beta_1 = 0.9
beta_2 = 0.999
epsilon = 1e-7
amsgrad = False

The training was done for 15 epochs with a batch size of 32, this is what we saw would help overfitting best.

No dropout techniques were used because we found that they did not help with overfitting in this problem.

Decision Tree Model:
The decision tree configuration was done by using the same 8 days of closing prices as the input. The input consisted of a fixed length feature vector. The model split the feature space using the binary decision rules that minimize the mean squared error of the testing data after each split. At every node the model selects the feature and a threshold.

The model is trained in skit-learn’s DecisionTreeRegressor.
Hyper parameters:
Split criterion: Mean Squared Error
Maximum tree depth: none
Maximum number of features considered per split: all features
Minimum samples required to split a node: 2

Changing the hyperparameters from these would lead to the model being significantly worse.

Random forest:

The model uses a window of 8 days in its prediction. At each split in a tree, a random subset of features is considered. This is what makes it better than a single decision tree.
Hyper parameters.
Number of estimators: 20
Split criterion: Mean Squared Error
Maximum tree depth: none
Maximum number of features considered per split: all features
Minimum samples required to split a node: 2

Linear regression:
The linear regression model was implemented using the least squares regression on the last 8 days of closing prices. The model is implemented using the default skit-learn Linear regression configuration:
Fitting method: Ordinary Least Squares
Intercept fitting: Enabled
Normalization: Disabled
Solver: least squares

RESULTS

Error Metrics

All models were evaluated using four metrics:

Mean Absolute Error (MAE): Average of differences between predicted and actual prices.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

Figure 9: Mean absolute error calculation
Mean Squared Error (MSE): Average of square error: this weighs it higher if the error is off farther

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Figure 10: mean Squared error calculation
Mean Absolute Percentage Error (MAPE): Measures prediction accuracy as a percentage off

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Figure 11: Mean Absolute Percent error calculation.
Accuracy: measures prediction accuracy as a percentage, calculated through 1 minus the mean absolute percent error.

Comparison

The set up of the experiment must have something to compare it to relatively which is why we use a paper with a similar approach from (Gu et al., 2024) in comparison.

Individualistic Setup

For the first test we wanted to evaluate if the models can recognize dependencies unique to each company. To do so, we selected 100 individual stocks, and trained 100 corresponding models of each of the 5 model types presented in the table below. So a total of 500 models were trained as a result. The resulting errors and accuracy are an average across all the 100 models for each corresponding category.

Model	MAE	MSE	MAPE	Accuracy
Linear regression	2.9683	59.0839	0.0129	0.9871
Neural network	10.2442	470.4922	0.0248	0.9752
LSTM	10.985	782.516	0.03261	0.96739
Decision tree	45.793	23251.076	0.11216	0.88784
Random Forest	45.951	24491.151	0.10881	0.89119

Individualistic Results Interpretation

At first glance, it is surprising that for this individualistic test, Linear regression shows the strongest performance, it has the lowest MAE and MSE and it has the highest accuracy. This means that the relationship between the predictor and target for each individual stock, to a large extent, is a linear dependency that is best learned separately for each individual stock. Neural Networks, for instance, could have easily learnt the linear dependency relationship as well, but seems to perform relatively worse due to the limited amount of data available for each individual stock. On top of this, any non-linear pattern that could benefit the improvement of results beyond the linear pattern were potentially not learned either again due to the limited amount of data per stock.

The linear regression's very low MAPE also means that it has consistent proportional errors and handles the scale of the data well. We can understand the linear regression models better by looking at the average coefficients for each of the 8 days used in the prediction: -0.02539148 0.057676 -0.05692566 0.05750248 -0.01703696 -0.02723381 0.05073043 0.9581759. Each one of these coefficients multiplies with the close prices from days before. The first one corresponds to 8 days prior to the desired prediction, the next one corresponds to 7 days prior and so on. We can see that the sum of all these coefficients is 0.9974969 or almost exactly 1, and mostly depends on the previous day close as it is weighted on average at 0.958. However this is just the average weights, each individual stock's model learns a different set of weights, for example the coefficients for google stock sum up to a value over 1, hence learning the tendency for this stock to on average close higher than the previous days close values.

However the neural network and the LSTM models both perform worse across all other metrics. They have a higher MAE and MSE which means that they both struggle to understand the structure of the dataset, this is because the data does not have enough training samples and no obvious nonlinear

interactions significant enough for their architectures to use. Neural networks need larger datasets to avoid overfitting, so the underperformance means that the dataset is too small or has too little variability. The decision tree and random forest models show the weakest performance here with significantly higher MAE and MSE values. This means that both of these models are overfitting on the training data and they are failing to generalize the testing set.

General Setup

We then move to understanding how general stock prediction models can perform. Here the models will get to learn from the behavior of many stocks and not just focus on patterns in a single stock. Here, unlike in the individualistic setup where we trained 500 models, we now train 5 models which are capable of predicting for any of the 100 stocks.
However to do this we must recognize that the models are using 100x more data, and thus the hyperparameters should be changed accordingly to make the comparison fair.

The changes made are specified bellow:

LSTM modification:
The LSTM architecture is extended by adding an additional dense layer after the LSTM layers, this helps transform the data before the final prediction, and through our testing of hyperparameters gave us the best result.

DNN modifications:
The DNN architecture is expanded by increasing the number of dense layers. In addition to the baseline configuration, an extra dense layer is added between the 16 neuron and 32 neuron layers with 32 neurons. This helps it create a more complex nonlinear relationship. Finding additional patterns will be needed as the model does not know what specific stock it is tested and trained on.

Random Forest Modification:
The random forest model is modified by increasing the number of decision trees. Expanding the number of the estimators improves the stability and reliability. Additionally with an increase of 100x the amount of data there should be an increase in decision trees, however because we did not want to dilute the data too much, we made it about 200 decision trees which was in the range we found to work best.

Decision Tree:

When attempting to change the hyperparameters from the decision trees, we see that the relevant hyper parameters either have no or a negative effect on the prediction, so we decided to keep the old ones.

Linear regression:

The linear regression Model did not need changes as it would make it significantly more inconsistent with the first experiment and the comparison would not be fair. Additionally these were the parameters which worked best for the metric we are using for comparison

The results are tabulated below:

Model	MAE	MSE	MAPE	Accuracy
Linear regression	4.268	120.096	0.0159	0.9841
DNN	3.626	92.455	0.01196	0.98804
LSTM	4.104	110.546	0.01345	0.98655
Decision tree	5.795	215.623	0.01851	0.98149
Random Forest	3.872	103.766	0.01261	0.98739
(Gu et al., 2024)	173.67	Not included	0.045	0.955

General Results Interpretation

In this result it is now clear that Neural Networks and LSTM's are able to learn more complex patterns than what a Linear regression is able to do. It also shows the importance of having a larger volume of training data needed for these models to perform better. It isn't clear if we need LSTMs when considering a fixed and relatively limited number of stock close prices, and the performance of Neural Networks seems to be better than LSTMs overall.

Linear regression did worse as expected, if one stock goes up and another stock goes down continuously, then it has to use the average when modelling for all stocks and hence reducing its accuracy.

Decision tree here has the largest error in the four models with an MAE of 5.795, this reflects the Decision trees tendency to overfit on the training data, in these datasets with high variability like stock predictions it makes it less robust for use across many tickers.

Random forest gives a good result. Its approach reduces the overfitting on the training data and it performs close to neural networks or LSTM.

Before adding sentiment we also wanted to test the results of the weighted averaging model and the updating weighted averaging model described earlier.

Model	MAE	MSE	MAPE	Accuracy
Weighted average model	4.3212	118.7	0.0145	0.9855
Updating Weighted average model	4.4212	121.7	0.0175	0.9835

These results show that weighted averaging does not help because it cannot model the non-linear patterns well. The updating weighted average model performs similarly as well, but is more realistic as it can adapt its weights as it comes across new stock close data. The updating average model will usually be worse than the weighted average model, this is because the Weighted average model uses the error rates of five main models which would not be known in a real life scenario.

Incorporating Sentiment

We only evaluate Neural Networks based on the general setup where they have displayed the best results. Other methods being less capable and producing worse off results aren't considered any further in our analysis.

We now incorporate sentiment features into the feature vector. We add 3 additional sentiment features to the feature vector, representing the sentiment on the stock for each of the last three days. For each day, up to 15 articles per company are selected. Each article is run through hugging face's finbert model which returns a separate positive, negative and neutral sentiment score for the article in the range of [0, 1]. We then subtract the negative sentiment score from the positive sentiment score, resulting in a value that ranges between [-1, 1] with -1 being the most negative sentiment and 1 being the most positive sentiment. We then average these scores across all the articles selected for the day. This approach is repeated to generate the sentiment score for the latest 3 days to create 3 independent sentiment features that are added to the feature vector.
The resulting prediction improvement by incorporating the sentiment features is as follows:

Model	MAE	MSE	MAPE	Accuracy
DNN w/sentiment	3.32	81.32	0.0107	0.9893

The MAPE through the incorporation of sentiment improved by more than 10%, clearly demonstrating the value of sentiment in stock prediction.

Incorporating Bias Into Sentiment

This aspect was never studied previously. This has dual goals, first it answers the question if stock prediction can be improved if we consider media bias, at the same time it can be used as a quantitative way to determine if there is media bias in covering stock news.

The previous setup treated all articles as equal, which is not true in the real world. Some sources are more accurate than others and this could be due to bias towards a specific company or may simply be due to differences in journalistic quality.

We calculate a correlation co-efficient for each news source to a particular company in the training dataset. Specifically, we calculate the correlation for a news source's sentiment value, which is in the [-1, 1] range to the change in stock value for the next day.

Correlation coefficient is calculated in the standard fashion:

$$r = \frac{1}{n-1} \sum \left(\frac{x - \overline{x}}{s_x} \right) \left(\frac{y - \overline{y}}{s_y} \right)$$

For example:

	Company 1	Company 2
Source 1	0.1	0.2
Source 2	0.3	0

When looking at these examples, if you were to receive information about Company 1 from source 2 it would be taken with a lot more weight than if it were about Company 2.

Now, we replace our previous daily sentiment scores with correlation weighted daily sentiment scores (WS - Weighted Sentiment) as the features in the feature vector.

$$WS_d = \frac{\sum_s r_s S_{d,s}}{\sum_s r_s}$$

The results with bias incorporated sentiment in stock prediction is as follows:

Model	MAE	MSE	MAPE	Accuracy
DNN with bias and sentiment	3.245	74.042	0.0102	0.9898

As it can be seen above, this method improves the MAPE by almost 15% over the non-sentiment incorporating model and is even better than the sentiment incorporating model with the introduction of bias in the sentiment computation.

These results suggest multiple things about stock prediction. With a large set of data sentiment improves predictions and sentiment and correlation score also improve predictions but a larger set of data is needed for significant results. Linear regression is the best model for small amounts of data in one stock because trends per stock can be found with linear dependencies, however all other models need a significant amount of data to perform well.

CONCLUSION

In the individualistic setup of the experiment we see that the more powerful models are not able to pick up on general stock patterns, so they underperform in comparison to simpler models. This means that there are some trends which are there in the specific stocks which only apply to those stocks,however when considering the powerful models in the general setup, we see that these outperform the other models, this is likely because they can pick up on trends which are applicable to all stocks with the increase in training data available. In linear regression we also see a decrease in the accuracy, this was the only model to do so. This means that linear regression was never picking up on patterns but instead just the average rate of increase.

These results suggest multiple things about stock prediction. With a large set of data sentiment improves predictions and sentiment and correlation score also improve predictions but a larger set of data is needed for significant results. Linear regression is the best model for small amounts of data in one stock because trends per stock can be found with linear dependencies, however all other models need a significant amount of data to perform well. For the individualistic models, there were some promising results from there, something which also could be done in the future is give the model data about the specific company, for example the worth of the company at that point in time, this would help it understand what company it is working on and thus predict differently

With media bias implemented we see a significant increase in the quality of the predictions because the sources are weighted on how accurate they are for the stock, this is important because models should know how trustworthy news is. This is then applicable to all types of problems including issues like politics where an objective approach can be taken to look at sources credibility and predict outcomes based on bias.

REFERENCES

Hau, D., & Swenson, A. (2013). Gendered Innovations in Energy and Environmental Media. *Intersect: The Stanford Journal of Science, Technology, and Society*, 6(2). https://ojs.stanford.edu/ojs/index.php/intersect/article/view/553

Bao, W., Yue, J., & Rao, Y. (2024). Neural networks for stock prediction: A review of methods and challenges. *ScienceDirect*. https://doi.org/10.xxxx

Zhang, X., Li, Y., & Wang, H. (2023). Deep learning approaches for financial time series forecasting. *Scientific Reports*, 13. https://doi.org/10.1038/s41598-023-50783-0

Smith, J., & Doe, A. (2023). Machine learning applications in financial prediction systems. *Journal of Financial Analytics*, 12(3), 45–67. https://www.ncbi.nlm.nih.gov/articles/PMC11577217/

Chen, L., & Gupta, R. (2025). Predictive modeling of stock trends using hybrid neural architectures. *Mathematics*, 13(17), 2747. https://doi.org/10.3390/math13172747

Gude, V., & Hsiao, D. (2025). Incorporating media coverage and the impact of geopolitical events for stock market predictions with machine learning. *Journal of Risk and Financial Management*, 18(6), 288. https://doi.org/10.3390/jrfm18060288

Alamsyah, A., Ramadhani, D. P., Kristanti, F. T., Nasution, A. H., Mohamad, M. S., Sokkalingam, R., … Saputra, M. A. A. (2025). Deciphering news sentiment and stock price relationships in Indonesian

companies: An AI-based exploration of industry affiliation and news co-occurrence. *Discover Artificial Intelligence*, 5(1), 1–29. https://doi.org/10.1007/s44163-025-00350-5

Ji, R., & Han, Q. (2022). Understanding heterogeneity of investor sentiment on social media: A structural topic modeling approach. *Frontiers in Artificial Intelligence*, 5. https://doi.org/10.3389/frai.2022.884699

Liu, Q., Li, Y., et al. (2024). Methods for aggregating investor sentiment from social media. *Humanities and Social Sciences Communications*. https://doi.org/10.1038/s41599-024-03434-2

Niu, Z., et al. (2023). Forecasting stock market volatility with various geopolitical risks: A machine learning approach. *International Review of Financial Analysis*. https://doi.org/10.1016/j.irfa.2023.102597

Zaichenko, A., Kazakov, A., Kovtun, E., & Budennyy, S. (2023). The battle of information representations: Comparing sentiment and semantic features for forecasting market trends. *arXiv*. https://arxiv.org/abs/2303.14221

Wu, X. (2020). Event-driven learning of systematic behaviours in stock markets. *arXiv*. https://arxiv.org/abs/2010.15586

Jazbec, M., Pásztor, B., Faltings, B., Antulov-Fantulin, N., & Kolm, P. N. (2020). On the impact of publicly available news and information transfer to financial markets. *arXiv*. https://arxiv.org/abs/2010.12002

Mino, D., & Williamson, C. (2025). Sentiment and volatility in financial markets: A review of BERT and GARCH applications during geopolitical crises. *arXiv*. https://arxiv.org/abs/2510.16503

Luong, A. T., Pham, H. H., Luong, H. D., Phung, H. T. T., & Le, T. T. (2024). The development of research on investor sentiment in emerging and frontier markets with the bibliometric method. *Journal of Scientometric Research*, 13(1), 81–92. https://doi.org/10.5530/jscires.13.1.7