# Stock Price Prediction

**Comparing Neural Network and Long Short-Term Models for Time Series Data with Reduced-Order Modeling and Sentiment Analysis**

**Arushi Sadam, Alana Gaughan, Dylan McIntyre**

*Final Project for COE 379L: Introduction to Machine Learning and Data Science*
*Advisor: Dr. Tan Bui Thanh*

**Abstract:** This article compares several methods for stock price prediction using neural networks and classification algorithms. We compare results for both conventional neural networks and Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory or LSTM models. To improve predictions, we add sentiment analysis of news articles as a feature. Wavelet functions (Dynamic Mode Decomposition, DMD) are implemented to smooth data and ensure models are not trained on arbitrary error. Overall, the standard neural network outperformed the LSTM. Short-term predictions performed better while using less features while long-term predictions performed better with a greater number of features. Finally, classification algorithms were used to predict stock price increases based on sentiment data.

## 1. INTRODUCTION

The problem of predicting stock prices is notoriously difficult. Being able to accurately and reliably predict future stock prices has obvious applications for individual profit. Furthermore, the prices of major stocks have an enormous impact on the economy and society. Because stock price valuations are not just based on a company's economic output but also on the general's public opinions, getting an idea of the overall sentiment surrounding a company can allow analysts to better predict future stock prices.

Several research papers used Twitter post data to enhance 1-day stock price predictions. We trained a model on a financial tweet database to perform sentiment analysis. Then, we use classification and regression methods to predict the stock price with sentiment data features.

In order to expand on previous research, we've applied several existing methods to compare their results and develop new models. First, this paper compares LSTM neural network predictions with those of standard fully connected neural networks for predictions of 1 day or 20 days (with different lookback windows). Second, a wavelet function is introduced to smooth the price data and decrease prediction error (existing method, different application). Third, we use Hankel Matrix decomposition (existing method, different application) and Black-Scholes pricing (new method) to develop new features. Finally, we use various classification models to compare our new sentiment analysis function with the prebuilt Vader sentiment analysis model. These additions expand on existing solutions to the problem of predicting stock prices.

## 2. DATA ACQUISITION

Our sentiment analysis tool was created using a neural network trained on finance tweet data from Hugging Face. The database includes more than 38,000 tweets (in English) about financial topics, labeled as bearish (2, negative), bullish (1, positive), or neutral (0).

NewsAPI was used to access news headline data for the past 30 days. Any more data was prohibited for users with the free version of the service. We were able to save some headlines from mid-October when we ran our model in mid-November. This increased our total number of days with news headline data to 46.

Yahoo Finance's API was used to access close price, open price, volume, and more features for the chosen companies: Tesla, Apple, NVIDIA, Google, Amazon, Intel, Johnson & Johnson, Walmart, and Microsoft. This API was chosen because it was free to use and contained plenty of data to create our regression models.

## 3. SENTIMENT ANALYSIS

### *3.1 Data preprocessing*

To preprocess the Twitter financial tweet database to train the sentiment analysis model, all special characters, links, punctuation, and numeric values were removed, leaving only the text content. The dataset was then tokenized by assigning a unique number to each of the 46,650 unique words present. Tweets were converted into sequences of these numerical tokens and uniformly padded with zeros to ensure consistent input size for the neural network model. This pipeline standardized the data, enabling effective model training and analysis.

### *3.2 Sentiment analysis model training*

The NN model uses 5 hidden layers (2 dropout layers, and 3 dense layers with ReLU activation). The model uses an Adaptive Moment Estimator (Adam) optimizer and a categorical cross-entropy loss function. After manually adjusting the number of epochs, batch size, and dropout parameters, we achieved a 78% validation accuracy.

## 4. TIME SERIES MODEL

We evaluate three different prediction ranges: 1) a model that takes a 180-day sequence and outputs the next 30 days of stock prices (trained on 3 years of data), 2) a model that takes a 3-day sequence and predicts the next day of the stock price (trained on 30 days of data), and 3) a model that takes a 5-day sequence and outputs the next day of stock price (trained on 30 days of data) with a Dynamic Mode Decomposition-smoothed data set. Each of these 3 prediction periods are evaluated for the 2 different models (defined below, NN & LSTM) (6 combinations total). We have 8 features in total; for each of the 6 combinations, we evaluate using either 1, 7, or 8 different features. The top two models in the 5-day sequence to 1-day prediction are evaluated with sentiment analysis.

### *4.1 Model Types*

Two model types were evaluated: a Recurrent Neural Network (Long Short-Term Memory) and a Fully Connected Neural Network.

The architecture for a **LSTM Neural Network** (assumed 180 day input data, 20 day prediction) is defined as follows:

- **LSTM Layer**: A Long Short-Term Memory (LSTM) layer with 180 units:
$$\mathbf{h}_t, \mathbf{c}_t = \text{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}),$$
where $\mathbf{h}_t$ is the hidden state, $\mathbf{c}_t$ is the cell state, and $\mathbf{x}_t$ is the input at time $t$.
- **Dense Layers:**
    - · Dense layer with 180 units and Leaky ReLU (Rectified Linear Unit) activation. Leaky ReLUs introduce a small non-zero gradient for negative inputs, eliminating the vanishing gradient problem with standard ReLU activation functions.
    - · 2 Dense layers with 360 units and Leaky ReLU activation. For additional model complexity (expanded feature space), we introduce 2 more layers.
    - · Output Dense layer with 20 units; maps the processed features to 20 predicted future prices.
- **Optimization**: The model uses the Adam optimizer:
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\partial L}{\partial \mathbf{w}_t},$$
where $\eta = 0.001$ is the learning rate, $\mathbf{w_t}$ is a weight at a particular training step, $L$ is the Mean Squared Error (MSE) loss.

The fully connected neural network is similar, but lacks the LSTM layer.

### *3.2 Physics-Informed Features*

The model leverages domain-specific features to enhance predictive accuracy:

- **Technical Indicators:**
    - · Simple Moving Averages (SMA) (n = 30, n = 100) (**SMA**):
$$\text{SMA}_n = \frac{1}{n}\sum_{i=0}^{n-1} \text{Close}_{t-i}.$$
    - · **Volatility**: Annualized ( 252 trading days per year) rolling standard deviation of percentage changes:
$$\text{Volatility} = \sqrt{252} \cdot \sigma\left(\Delta \text{log}(\text{Close})\right).$$
- **Black-Scholes Option Pricing Model:** Using the Black-Scholes formula:
$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T},$$
the call option price is:
$$C = S\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$
where $S$ is the spot price, $K$ is the strike price (105% of current price), $T$ is the time to maturity (assumed 1 month), $r$ is the risk-free rate (1%), $\sigma$ is the volatility, and $\Phi$ is a cumulative distribution function (CDF).
- **Hankel Matrix and SVD:**
    - · Hankel matrix construction (m = sliding window size = 50):
$$\mathbf{H} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ x_2 & x_3 & \dots & x_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \dots & x_{m+n-1} \end{bmatrix}.$$
    - · SVD decomposition:
$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$
where $\mathbf{U}$ contains modes, and $\mathbf{\Sigma}$ contains singular values.

The top two (largest) modes are used as features; these represent the static patterns of the data.

### *3.3 Feature Preprocessing*

Features undergo preprocessing as follows:

- **Normalization**: Scaled to $[0, 1]$ using a MinMaxScaler:
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}.$$
- **Feature Combination**: Processed features and Hankel-derived modes are combined into a single feature set of up to 8 features per day: Close (stock price at day-end), Volume (number of traded shares), SMA30, SMA100, Volatility, 2 modes from Hankel decomposition, and Black-Scholes option price.

Additionally, we also use a Dynamic Mode Decomposition on our training data to ensure our model does not learn day-to-day noise, but rather the true oscillations/trend of the stocks.

The process of Dynamic Mode Decomposition (DMD) for a 1-feature time series can be written as follows:

1. Organize the data into matrices:
$$X = [x_1 \ x_2 \ \cdots \ x_{n-1}], \quad X' = [x_2 \ x_3 \ \cdots \ x_n]$$
2. Perform Singular Value Decomposition (SVD):
$$X = U\Sigma V^*$$
3. Project and compute the approximation of $A$:
$$A_r = U_r^* X' V_r \Sigma_r^{-1}$$
4. Eigen-decomposition of the reduced $A$:
$$A_r W = W\Lambda$$

Here, $\Lambda$ contains the eigenvalues, and $W$ contains the eigenvectors.

5. Reconstruct the time series:

$$\hat{X}(t) = W \exp(\Lambda t) b, \quad b = W^{-1} x_1$$

We use PyDMD, a python package, for this approach. This decomposition is also referred to as using a wavelet transformation.

### *3.4 Results*

Ultimately, 6 types of models are tested:

- 1. LSTM with 1 feature (close price)
- 2. LSTM with 7 features (close, volume, SMA30, SMA100, Volatility, 2 modes from Hankel decomposition)
- 3. LSTM with 8 features (close, volume, SMA30, SMA100, Volatility, 2 modes from Hankel decomposition, option price)
- 4. Above 3 models, but instead with a fully connected neural network (rather than LSTM).

For each model, we evaluate the Mean Absolute Percentage Error (MAPE) of 10 different stocks. For long-term predictions (20-day predictions), only the errors of one 20-day prediction (each day-to-day value is considered) per stock are used. For short-term predictions (1 day), we do multiple evaluations on the testing data set (about 5 days) to compute the MAPE for each model; we do not do more evaluations for each model since our total data set is only about 30 days long (when implementing sentimental analysis of the news data).

The below model represents a long-term prediction; it uses about 3 years of data with 180 days of input data and 20 days of prediction.

| **Model** | **1 Feature** | **7 Features** | **8 Features** |
|---|---|---|---|
| NN | 5.875 | 6.167 | **5.375** |
| LSTM | 6.889 | 8.331 | **6.581** |

**Table 1:** 180-Day Input, 20-Day Prediction.

The top two models (bolded in Table 1) are shown graphically in Figure 1a and 1b (figures in Appendix, pg 5).

For short-term predictions (1 day input, 1 day prediction), we follow a similar pattern. This time, however, instead of SMA30 and SMA100 we use a 3-day and a 5-day moving average as features. The Hankel matrix is constructed with a 5 day window, as well.

| **Model** | **1 Feature** | **7 Features** | **8 Features** |
|---|---|---|---|
| NN | **0.925** | 2.584 | 2.201 |
| LSTM | **1.149** | 1.491 | 2.311 |

**Table 2:** 1-Day Input, 1-Day Prediction.

The top two models (bolded in Table 2) are shown graphically in Figures 2a and 2b.

Finally, for the second type of short-term predictions (5 day input, 1 day prediction), we follow a similar pattern. We evaluate the best two models of the above short-term predictions (NN with 1 feature, LSTM w/ 1 feature), and add combinations of Wavelet transformations of data and the news article sentiments.

| **Model** | **Close** | **Close + Sentiment** | **Wavelet Close** | **Wavelet Close + Sentiment** |
|---|---|---|---|---|
| NN | 1.972 | 1.782 | **1.130** | 1.394 |
| LSTM | 1.854 | 2.589 | 1.489 | **1.102** |

**Table 3:** 5-Day Input, 1-Day Prediction with DMD and Sentiment Analysis.

The top two models (bolded in Table 3) are shown graphically in Figures 3a and 3b. Note that the predictions are far smoother than in previous models, as a result of the DMD implementation.

## 5. CLASSIFICATION METHODS FOR NEXT DAY PREDICTION

The goal of using the classification methods was to find whether a given stock price would increase or decrease the next day. We used our sentiment model and also the pre-built Vader model to find sentiment scores for each news headline in our dataset.

### *5.1 Data Preprocessing*

In order to get a target related to a binary class, we calculated the next day price increase as the target and converted it to a 1 if it was positive and 0 if it was negative or neutral.

We used the news headlines from News API to get an average daily sentiment score. This API was able to give us 100 headlines per day that mentioned the company name in the article, however this means that not every headline was relevant to the company. We had to filter out irrelevant news headlines that were collected from our query. We used keywords like the company name, the CEO name, and the company abbreviation to ensure all headlines were relevant. We averaged the sentiment scores for all headline data in one day to get the total combined sentiment scores for the day.

To define our targets, we looked at three components of the daily sentiment data: positive, negative, and neutral scores. We also decided to use the rolling means of the sentiment scores as separate features. This would hopefully provide some insight into overall trends in sentiment, not just the sentiment of the day before. Finally, we applied standard scaling to these components to help with the radial basis function kernel and polynomial kernel models.

### *5.2 Model Types*

We used 4 supervised classification models in total.

(1) Logistic Regression is similar to linear regression, but uses the logistic function to draw the decision boundary, resulting in a curve that divides different classes.

(2) Gaussian Discriminant Analysis (GDA) model is a statistical model that assumes that the classes follow a normal distribution. Then it uses Bayes's theorem to compute the conditional probability that a certain data point belongs to a class given the known features.
(3) $5^{th}$ degree polynomial kernel with the support vector machine (SVM) draws a nonlinear decision boundary using a fifth degree polynomial.
(4) Radial basis function kernel with SVM also draws a nonlinear decision boundary. This kernel measures similarity based on distances between points.

### *5.3 Evaluation Metric*

We used the F1 score to evaluate each model. The F1 score is based on the model's precision and recall scores. Precision is the ratio of true positives to total positives, and recall is the ratio of true postives to the sum of all samples that should have been positive (true positives and false negatives).

Since our dataset was relatively sparse, we used k-fold cross validation to ensure we could accurately evaluate each model. This process involves sectioning the dataset into k even parts (in our case k=6), then using each part as the holdout data for one model while the rest of the parts are used for training data. Once a given model was evaluated, it's F1 score was recorded, the model was discarded, the next section of data was used as the holdout, and the process was repeated.

### *5.4 Results*

Below are the tables of our results. The model number corresponds to the number listed in section 5.2.

| **Model** | **1 day** | **2 days** | **3 days** | **4 days** | **5 days** |
|---|---|---|---|---|---|
| **1** | 0.2952 | 0.3750 | 0.3750 | 0.3154 | 0.2222 |
| **2** | 0.4083 | 0.4083 | 0.3949 | 0.3313 | 0.2629 |
| **3** | 0.3444 | 0.3333 | 0.4333 | 0.3015 | 0.3095 |
| **4** | 0.2167 | 0.4841 | 0.3451 | 0.2222 | 0.2062 |

**Table 4:** Mean F1-score after 6-fold cross validation (Vader Model).

| **Model** | **1 day** | **2 days** | **3 days** | **4 days** | **5 days** |
|---|---|---|---|---|---|
| **1** | 0.3444 | 0.2730 | 0.2869 | 0.2738 | 0.3472 |
| **2** | 0.3656 | 0.4220 | 0.3472 | 0.3412 | 0.3656 |
| **3** | 0.3175 | 0.1905 | 0.4048 | 0.4266 | 0.3250 |
| **4** | 0.2611 | 0.1508 | 0.2361 | 0.4107 | 0.4127 |

**Table 4:** Mean F1-score after 6-fold cross validation (Our Model).

These F1 scores are all relatively low, although models 3 and 4 (the polynomial and RBF kernel with SVM models) did a bit better overall. From the fact that these F1-scores were highly variable, we determined that there is not a direct enough correlation between sentiment data and price increase. In order to improve classification models in the future, more statistical and financial features should be included.

Some sources of error include that the data was really sparse. We only had 46 days worth of headline data, and only the weekdays were usable as data points, which brought our final number of points down to 33 days. Furthermore, we had even less data as the number of days used for the rolling mean features increased. Another source of error might be irrelevant headlines included in the dataset. If an article mentioned a company breifly, the headline was included in the dataset, but the headline itself might not be relevant enough to determine general sentiment towards the company.

## 6. CONCLUSION

In previous works, researchers used social media posts to determine daily sentiment values and made 1 day predictions on stock price. In this project, news data is used to determine a daily sentiment value. For the 1-1 predictions, the model with the least features outperformed the higher feature models. For the 180-20 day predictions, the higher feature predictions performed better. For the 5-1 day predictions, implementing the wavelet greatly improved the error. Feature engineering improved long-range predictions the most.

Also, NN models outperformed LSTM models in most graphs, which was not initially expected per the literature review. This might be because of short time dependencies or short windows (LSTM is more suited for long range, time-dependent patterns).

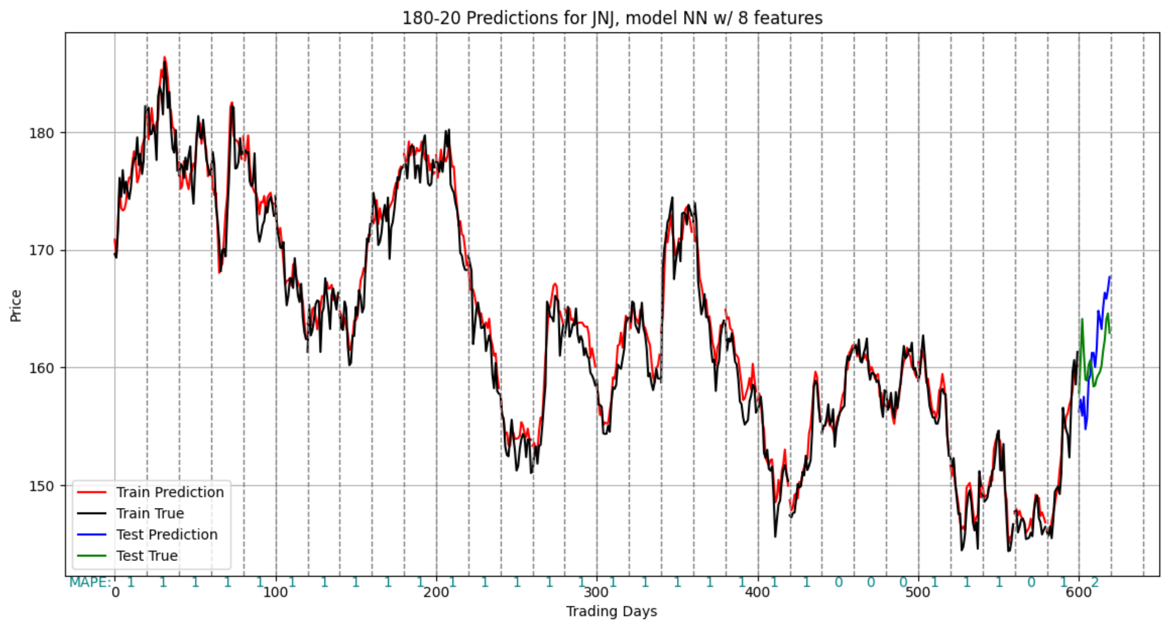
Generally, the addition of sentiment data as a feature reduced accuracy. This might be due to limitations of the data sources utilized in this project. Headline data is also much sparser than social media post data, and free API's were typically limited. All sentiment analysis models used were trained on twitter data, so they may not accurately characterize the sentiment scores of headlines.

## REFERENCES

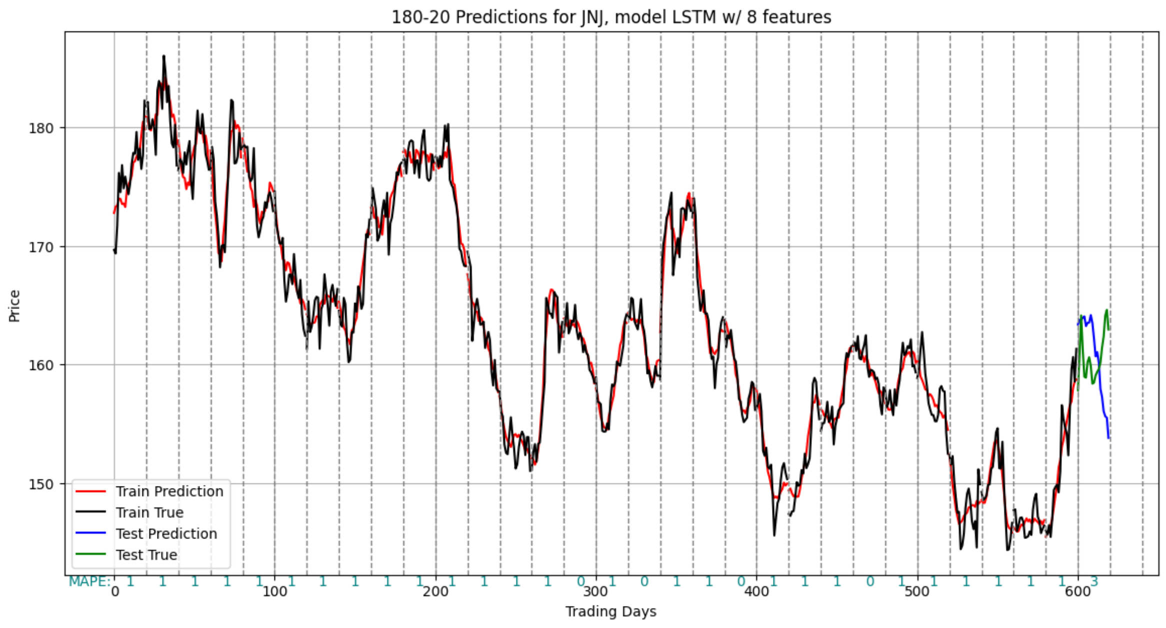
[1] Salim Olinyola, *How To Train a Neural Network for Sentiment Analysis*, November 28, 2022, `https://www.digitalocean.com/community/tutorials/how-to-train-a-neural-network-for-sentiment-analysis`.

[2] Tim Koornstra, *Financial Sentiment Analysis Dataset*, HuggingFace, `https://huggingface.co/datasets/TimKoornstra/financial-tweets-sentiment`.

## 7. FIGURES/GRAPHS

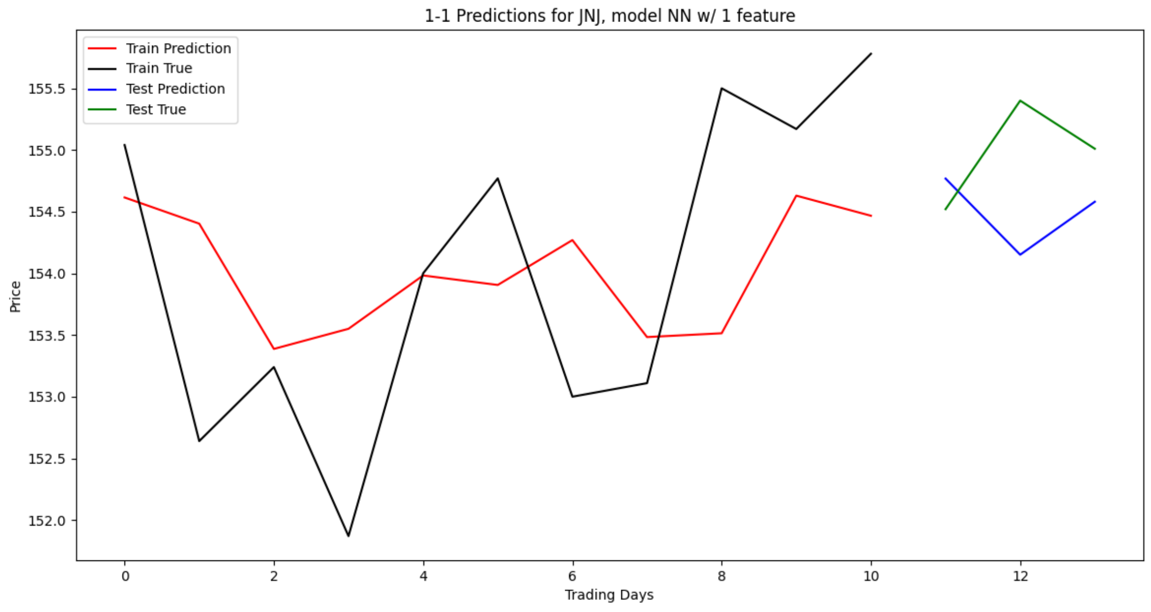


(a) 180-20 predictions with NN, 8 features

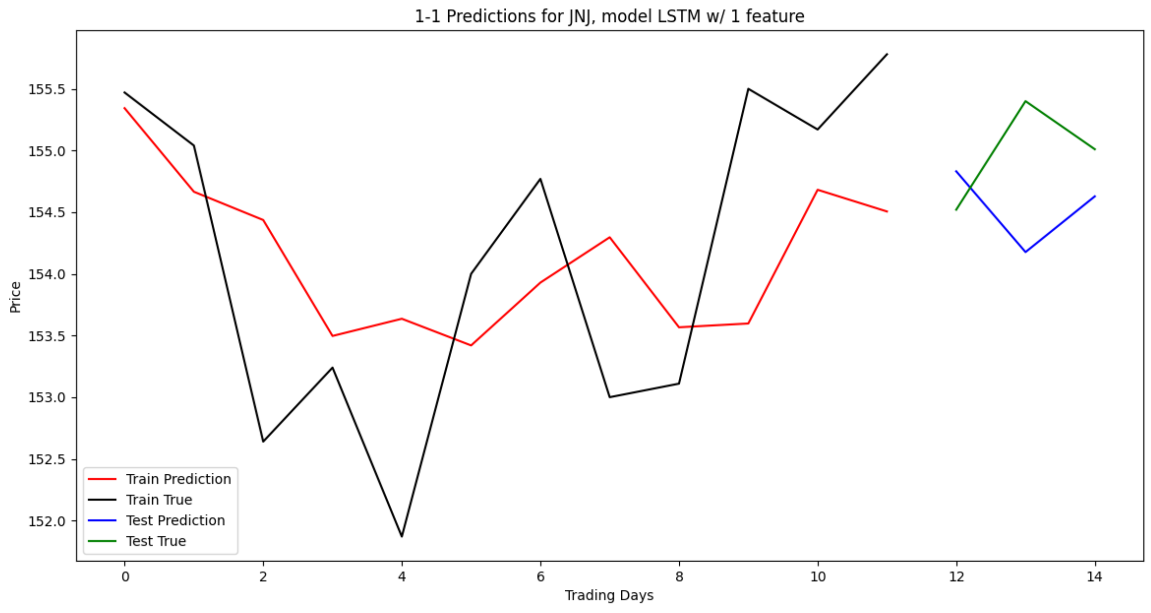


(b) 180-20 predictions with LSTM, 8 features

Fig. 1. 180-20 Day Predictions

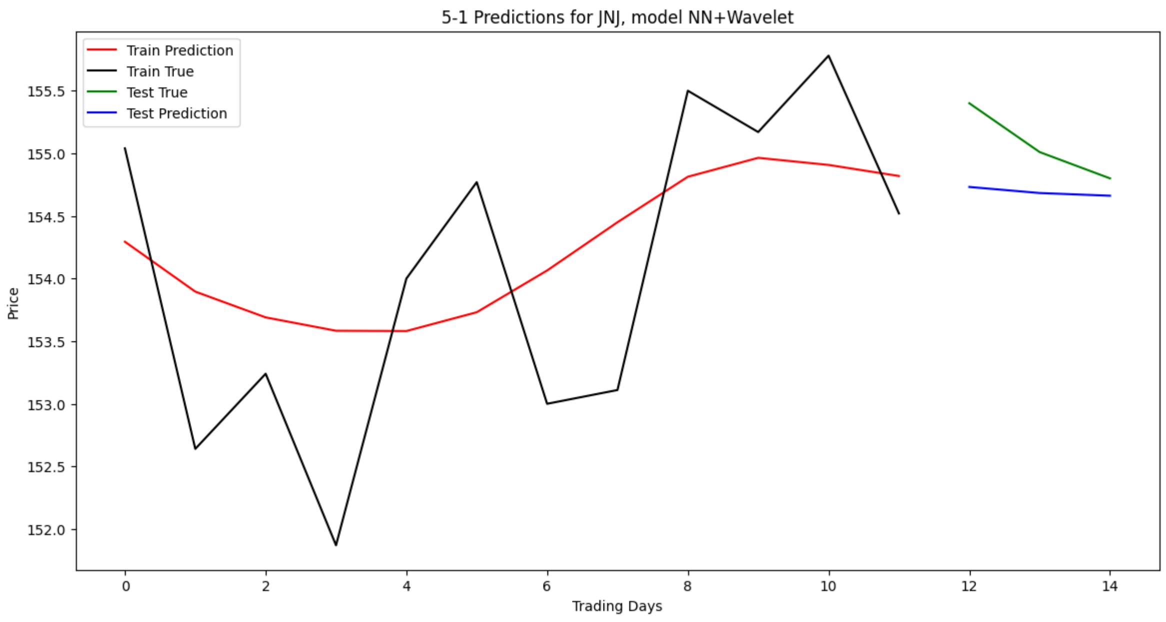


(a) 1-1 predictions with NN with 1 feature, Closing Price

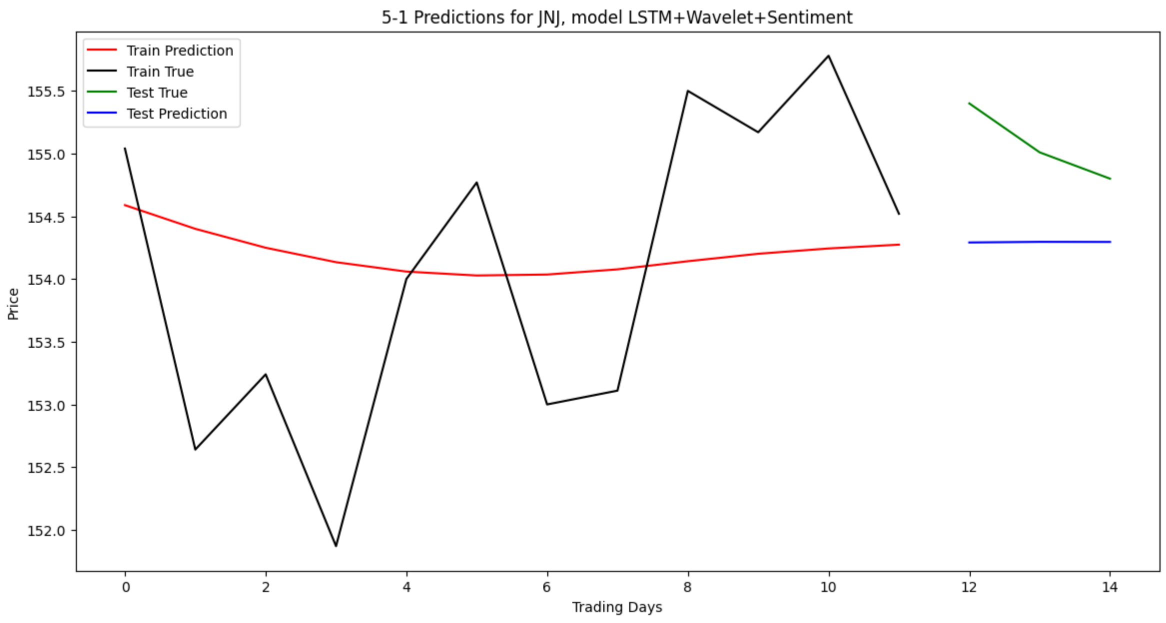


(b) 1-1 predictions with LSTM with 1 feature, Closing Price

Fig. 2. 1-1 Day Predictions



(a) 5-1 predictions with NN, Wavelet transformed data



(b) 5-1 predictions with LSTM, Wavelet transformed data, and Sentiment Analysis

Fig. 3. 5-1 Day Predictions