**Exploring the Possibility of Forecasting Stock Prices**

Reuben Lopez

Fall 2020

In Partial Fulfillment of

Data 4395-Senior Project

Project Advisor: Dr.Vasilis Zafiris

Department of Mathematics and Statistics

|  |  |
| --- | --- |
| Faculty Advisor: |  |
| Dr. Vasilis Zafiris |  |
| Committee Member: |  |
| Dr. Timothy Redl: |  |
| Committee Member: |  |
| Dr. Dexter Cahoy: |  |
| Department Chair: |  |
| Dr. Ryan Pepper |  |

**Abstract**

Stock forecasting is the process of predicting the future prices of a stock based on previous historical data. With the introduction of machine learning techniques and the use of Big data, traders have claimed to have begun having success in predicting the stock market.

The main idea of this work will be to highlight if machine learning can be used to forecast stock prices. In the process we will examine the theory of each algorithm and view the effectiveness of each algorithm

We will be implementing 3 Algorithms on the feature dataset:

* Linear Regression
* Decision Tree
* Recurrent Neural Network

**Table of Contents**

1. Dataset Overview
2. Linear Regression
   1. Theory
   2. Problems when implementing Linear Regression
   3. Implementation & Results
3. Decision Tree
   1. Theory
   2. Problems when implementing Decision Trees
   3. Implementation & Results
4. Recurrent neural networks (**LSTM**)
   1. Theory
   2. Problems when implementing Recurrent Neural Networks
   3. Implementation & Results
5. Conclusion

**Dataset Overview**

The feature dataset used for this work is Apple’s stock prices from 01/14/2012 - 11/29/2019 or 1258 rows of data. This dataset was sourced from Yahoo Finance, and contains columns such as Date, Open, High, Low, Closed, Adj Closed, and Volume.There was no missing data in this dataset. The feature columns used for the machine learning models are Date, Closed, and Volume. We define and view the columns of our data below.

**Date**: Date of the trading day

**Open**: Price of the stock at the start of the trading day.

**High**: Highest price point the stock was traded at during the trading day.

**Low**: Lowest price point the stock was traded at during the trading day.

**Closed**: Final price the stock was traded at once the markets closed.

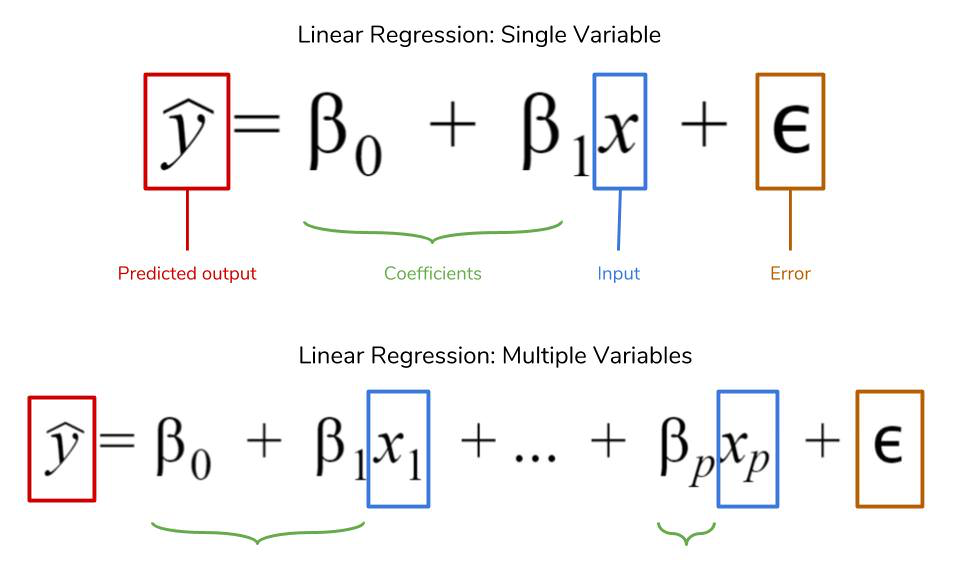
**Adj** **Closed**: The closing price of a stock after accounting for any corporate actions.

**Volume**: The physical number of shares traded of that stock for the trading day.

**Linear Regression**

**Theory:**

One of the more basic machine learning algorithms that can be used for forecasting data is linear regression. Linear regression can be summarized as a linear approximation of a relationship between two or more variables. The goal is to approximate a line that best fits the data.



The (y) variable is considered the dependent variable with (x) being the explanatory variable. In other words, x explains the movement of y. The relationship of x & y is modeled with an error term which captures how far off the predicted linear equation is from each data point, with the goal being to minimize the sum of the errors.

The great advantage of regression models is that they can be used to capture important relationships between the forecast variable of interest and the predictor variables. In the context of machine learning, a linear regression model can be one of the simpler algorithms to model.

**Problems when implementing Linear Regression**

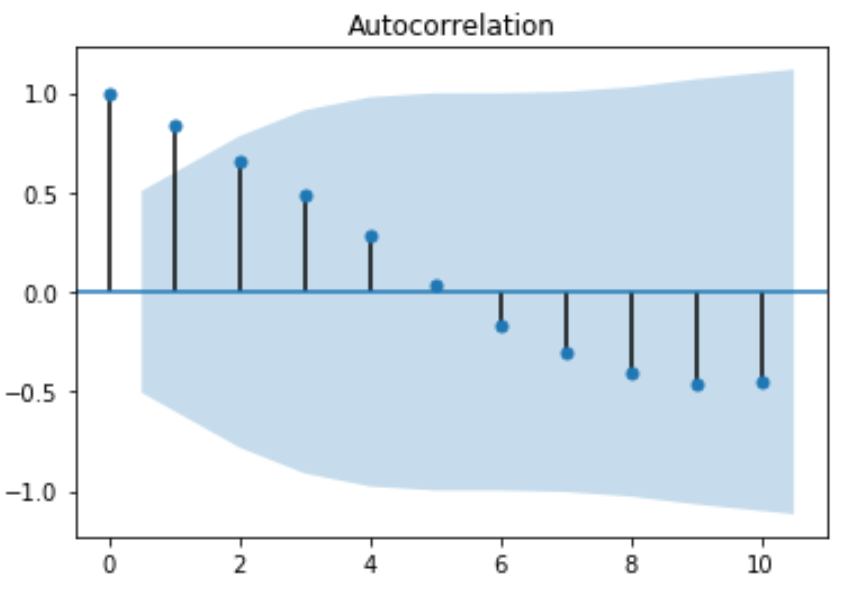
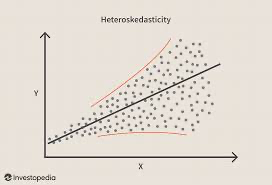
There are multiple problems that can arise when forecasting using regression models. These problems can have a significant impact on the accuracy and significance of your model, we will highlight just a few

Autocorrelation is the degree of correlation between values of the same variable. While not exclusive to time series data, it is the most common when dealing with this type of data. The Durbin-Watson statistic can be used to identify autocorrelation.

Heteroscedasticity refers to the variability being unequal across a set of second predictor variable, this can result in biased coefficients. A way to check for this is with a scatter plot, if your data resembles a cone then it is likely that your data is Heteroscedastic. Using a weighted least square, or a transformation of your variables can correct this.

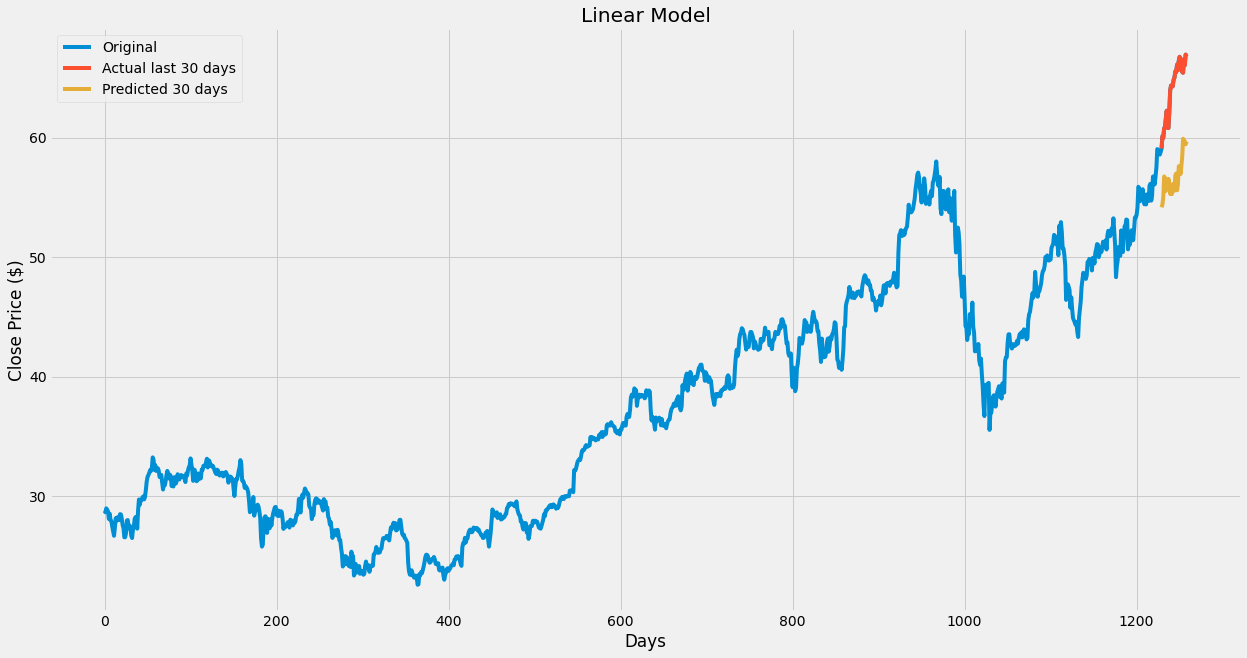
While overfitting is an issue for most algorithms, under-fitting is very common when implementing linear regression.Under-fitting occurs when the linear regression model fails to capture the data properly. Since linear regression assumes a linear relationship between the input and output variables, it fails to fit complex datasets properly. In most real life scenarios the relationship between the variables of the dataset isn't linear and hence a straight line doesn't fit the data properly. In such situations a more complex function can capture the data more effectively

Linear regression models are also extremely sensitive to outliers. Having outliers in the dataset can drastically impact the accuracy of the model often leading to misinterpretations .



**Implementation & Results:**

This model was implemented using the python libraries SKLearn and Keras, both are machine learning modeling libraries. The goal for our model is to predict the next 30 days of trading based on 4 years of historical trading data or 1258 days of trading. The data was split with 75% of the data being used to train the model & 25% for the testing set. The results of the model indicate that using linear regression to predict the stock price in this scenario would not be the best choice. The model predicted a 30-day price trend significantly lower than what the actual last 30-day trend was. Below I have included a code snippet explaining how the data was split into the test and training sets, as well as how the model was built using SKLearn. In conclusion, linear regression can be a great method to use when analyzing relationships between variables however in this scenario it seems that linear regression would be too simple of a model to use when forecast stock prices.

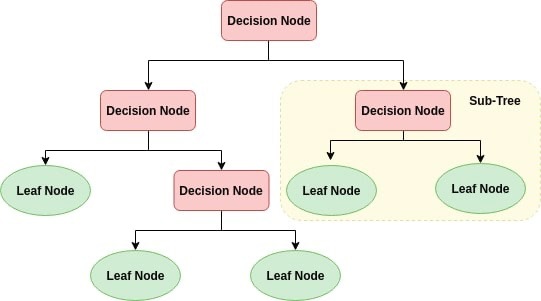


**Decision Trees**

**Theory:**

Classification and regression tree (CART) method, known also as decision trees is a powerful data mining technique used in medicine, economy and many other areas.Generally, CART is a supervised learning algorithm capable to process very large or small datasets of arbitrary type of predictor variables to regress a given dependent variable.CART can handle locally linear, nonlinear effects and interaction terms in a rule sequence and build the easily interpreted models.The algorithm takes observations about an item to then make a prediction about the items value. The optimization of a decision tree is when a tree can represent the most data with fewest number of levels. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity. The main idea behind tree methods is to recursively partition the data into smaller and smaller strata in order to improve the fit as best as possible

Some advantages of using the CART method are that CART models can handle numerical data, and categorical data, and combinations of these really well. Also, there are no special requirements for the distribution and specific properties of the variables. CART models are also able to detect complex nonlinear dependencies. Another important advantage is that they are simple to understand and interpret. The structure of a CART tree consists of a root node, internal nodes and terminal nodes, each of them representing a subset of cases falling within a subregion.



**Problems when implementing Decision Trees**

Overfitting can be an issue for most algorithms; however, decision trees are extremely sensitive to this issue. Decision trees are more liable to overfitting because of the likelihood of having a complicated long decision chain are high. If a data point satisfies all the rules in the decision chain, then a decision can be made. This type of rule training is very specific to the training set making it difficult to generalize new data. This becomes even more problematic for higher dimension datasets.

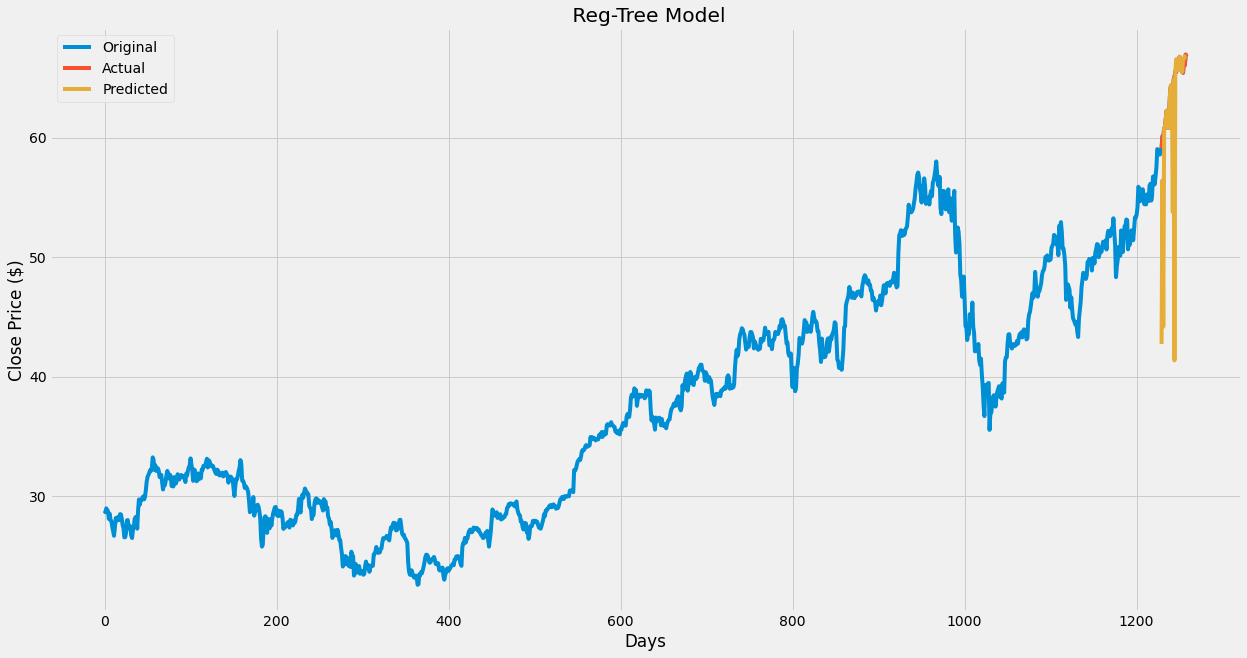
A popular method for addressing the issue of overfitting in decision trees is “Pruning”. In pruning, you trim off the branches of the tree that are not critical or that are redundant. This reduces the complexity of the tree which in turn improves the accuracy of the decision tree.

With Random Forest specifically, an issue in dealing with time-series data is that they do not bring forward information when evaluating data like linear models do. It is essential to use previous information to forecast when dealing with time-series data. To overcome this, the introduction of lagging variables can be used to help evaluate information from the past.

**Implementation & Results**:

This model was implemented using the python libraries SKLearn and Keras, both are machine learning modeling libraries A regression tree was used for this decision tree model. The data was split with 75% of the data being used to train the model & 25% for the testing set. The results were a bit surprising. The regressor tree did a much better job than the linear regression model. However, its predicted 30-day was way more volatile than the actual 30-day trend was. While the trend was not accurate, the final price was not far off.As with our linear model, below I have included a code snippet explaining how the data was split into the test and training sets, as well as how the regression tree model was built using SKLearn.



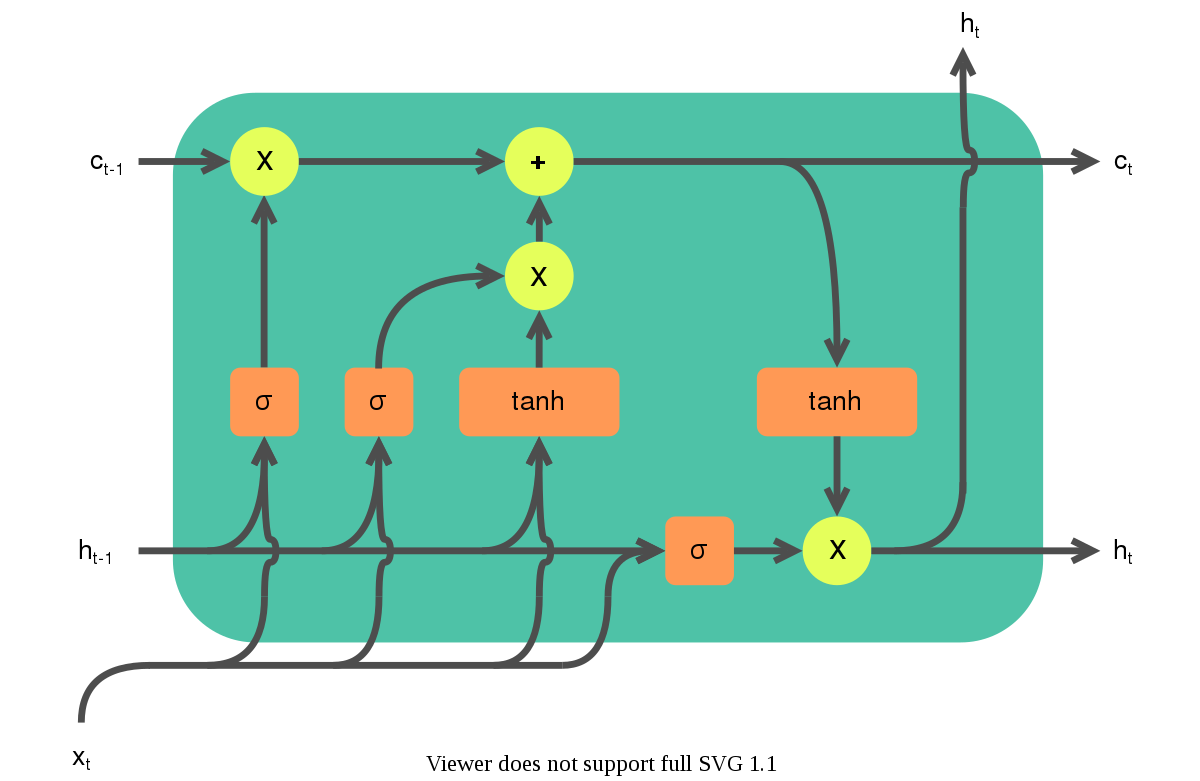


**Recurrent neural networks (LSTM)**

**Theory**:

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input. In short, Recurrent Neural Networks use their reasoning from previous experiences to inform the upcoming events.

Long Short-Term Memory is an artificial recurrent neural network (RNN) architecture. Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.



**Problems when implementing RNN**

Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they will have a challenging time carrying information from earlier time steps to later ones. In the case of So if you are trying to process a paragraph of text to do predictions, RNN’s may leave out vital information from the beginning

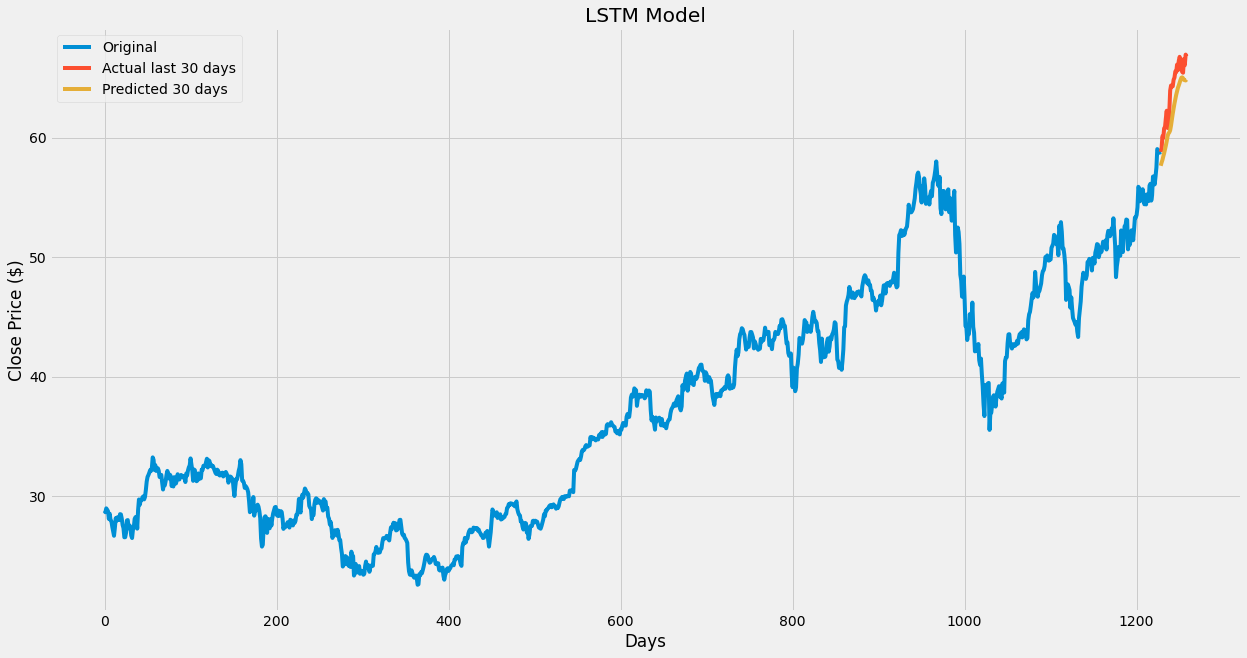
Along with the short term memory problem, exploding gradients can also become a huge problems with dealing with RNN’s. An error gradient is calculated during the training process which is then used in order to update the network weights. When using recurrent neural networks, These gradients can accumulate during the updating process thus resulting in large gradients which result to large updates. Large updates to network weights can result into having an unstable network.The explosion occurs through exponential growth by repeatedly multiplying gradients through the network layers that have values larger than 1.

Some signs that the model might have exploding gradients are, when weights in the models become large quickly, also large changes in loss for each update iteration indicating an unstable model. One of the most basic yet common approaches to fixing the exploding gradient problem is a simple redesign of the model. Another solution is the LSTM model.

LSTM networks are well-suited to classifying, processing, and making predictions based on time series data, since there can be lags of unknown duration between notable events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNN. For this reason, LSTM, a variation of a RNN, will be used in our modeling

**Implementation & Results:**

This model was implemented using the python libraries SKLearn and Keras, both are machine learning modeling libraries. As with our other models, the data was split with 75% of the data being used to train the model & 25% for the testing set. The results proved that LSTM was the best of the three algorithms when forecasting stock prices. The last 30-day trend was not as volatile as the regressor tree predicted, and much more accurate than the linear regression model. The model’s predictions were quite close to the actual last 30-days prices. It is possible that the LSTM model could have made a more accurate prediction by adding more layers to the LSTM.



**Conclusion**

In conclusion, we have showed how machine learning can be used to forecast a stock's price, as well as showing how some algorithms forecast better than others. In our case the more complex an algorithm, the better the prediction. Many have said that predicting the price of stock is an impossible task due to the many variables involved in the stock market itself. While machine learning may never be able to be used to predict the exact price of a stock, it can be useful tool that can maybe give insight on the future movement of a stock.

**References**

Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5-6), 594-621. https://doi.org/ 10.1080/07474938.2010.481556

Cummins, F., Schmidhuber, J., & Gers, F. (n.d.). Learning to forget: continual prediction with LSTM.

Gers, F. A., Eck, D., & Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. *Perspectives in Neural Computing*, 193-200. https://doi.org/10.1007/978-1-4471-0219-9\_20

Gocheva-Ilieva, S. G., Voynikova, D. S., Stoimenova, M. P., Ivanov, A. V., & Iliev, I. P. (2019). Regression trees modeling of time series for air pollution analysis and forecasting. *Neural Computing and Applications*, *31*(12), 9023-9039. https://doi.org/10.1007/s00521-019-04432-1

Klapper-Rybicka, M., Schraudolph, N. N., & Schmidhuber, J. (2001). Unsupervised learning in LSTM recurrent neural networks. *Artificial Neural Networks — ICANN 2001*, 684-691. https://doi.org/ 10.1007/3-540-44668-0\_95

Moisen, G. (2008). Classification and regression trees. *Encyclopedia of Ecology*, 582-588. https://doi.org/10.1016/b978-008045405-4.00149-x

Olah, C. (2015). Understanding LSTM Networks. https://colah.github.io/posts/ 2015-08-Understanding-LSTMs/

Staudemeyer, R., & Morris, E. (2019). Understanding LSTM. *a tutorial into Long Short-Term Memory Recurrent Neural Networks*.

Tan, C., Feng, X., Long, J., & Geng, L. (2018). FORECAST-CLSTM: A new Convolutional LSTM network for cloudage Nowcasting. *2018 IEEE Visual Communications and Image Processing (VCIP)*. https://doi.org/ 10.1109/vcip.2018.8698733