Exploring Stock Price Predictions with Machine Learning

Ihsaan Malek, Jonathan Frey, and Uriel Bitton

Abstract

The goal for the project was to determine whether Machine Learning models could accurately predict stock price data. Two deep learning models which are suitable for time series forecasting were explored, a Long Short Term Memory model and a Temporal Convolution Network. Initial results were insignificant, but after careful model tuning and the addition of features our models were able to improve and gave some satisfying results in certain interesting cases that we will showcase in this report.

1. Introduction

Forecasting stock prices remains an elusive goal that many try to achieve. Due to the unpredictability of the data and the almost endless amount of variables that affect it, the task is extremely difficult, and financial firms spend a huge amount of resources trying to successfully achieve it. With financial data being noisy and available at such a high frequency, the goal of this project was to determine if there are any machine learning models that would be able to predict the stock price accurately.

1.1 Data

The standard used for equities price data is OHCL, or Open, High, Low, Close. These refer to the stock's price when the market opens, the day's high, the day's low and the price the stock ends at when the market closes. In addition to OHCL, daily trading volume (quantity of shares that underwent a transaction) is provided. Typically, the data is pre-cleaned to take in account of weekends and U.S. holidays as the Equities Market is closed. Secondly, we will be using data that will pre-adjust for stock splits (for example APPL currently trades approx. $120, therefore the past data will be scaled down to account for the splits). This will help prevent the models from reacting poorly due to an unobserved event.

There were four main stocks that we used, separated in three categories. The S&P 500, which is a market index fund of the 500 biggest companies listed in the American stock market and is typically used as a benchmark to compare return performance. Apple as a benchmark stock which closely follows the movements of the overall market (S&P500). Tesla was used as an example of a highly volatile stock, to determine how well our models would perform on unpredictable data. Lastly, AT&T was used as an example of a low volatile stock, to test how well a model could predict very narrow ranged data.

1.2 Models

In this project machine learning is explored to perform temporal sequence predictions. The two main deep learning models looked at are a LSTM (Long Short-Term Memory) and a TCN (Temporal Convolution Neural Network). We have included Linear Regression as a baseline when comparing model results.

Recurrent Neural Networks (RNN) would normally be good to use for time series forecasting and learning the dependency on past data, however they run the risk of vanishing or exploding gradients, therefore, to mitigate this risk an LSTM model was used instead. Indeed, an LSTM model has the added advantage of feedback connections which enables the model to selectively remember patterns for a long duration of time and their validity not being affected by arbitrary gap length between two important events. This is useful when predicting stock prices as patterns often re-emerge in the data. LSTM is made up of a cell, an input gate, an output gate and a forget gate. It's the cell's memory unit that gives the ability of the model to encapsulate the notion of forgetting part of its previously stored memory, as well as to add part of the new information.

In April 2018, a research paper was published exploring a CNN architecture for temporal modeling and suggested that it would outperform across a broad range of sequence modeling task. Bai et al. (2018) described a generalized model architecture and named it the Temporal Convolution Network (TCN). The two main principles around a TCN is the fact that the output will have the same length as the input and the second main point is that there is no leakage of data from the future into the past. A TCN's architecture consists of a 1D fully - convolutional network plus causal convolutions. To ensure that the output length remains equal to the input length, each hidden layer is the same length as the input layer and zero padding of size *(kernel_size -1)* is applied to the left of the input data. The causal convolutions allow the output at *time t* to be dependent only on present and past data from previous network layers. The proposed architecture also suggests using dilated convolutions to create a larger receptive field without having to linearly increase the number of layers in the model (Bai et al.). The proposed architecture was used without modifications for this experiment.

2. Methodology & Experimental Results

2.1 Time Series Forecasting and Sampling

To predict the stock prices, we use time series forecasting, which uses past time values to predict future values. There are a couple different methods to perform time series forecasting, there is recursive forecasting, time horizon forecasting and Univariate/Multivariate-In-Single-Out. For recursive forecasting, you take a model's predicted output and use it as part of the new input. This method was not used due to the fact that this can reintroduce new errors to model. Time horizon forecasting implies that the predicted output is of multiple time periods. This method was not used because of the nature of the data at hand. Stock price data is particularly volatile; thus, the model will not be able to account for the volatility properly. In addition, there are often surprise shocks events that cause the stock price to have sharp movements. Thus, we have used the third method which takes in data of length time steps and outputs the next day's prediction. For both the LSTM and TCN, this is done by using the sliding window method, if the input is of length 5, the input will have values {*t0, t1, t2, t3, t4*}, the target will slide over one step and be {*t1, t2, t3, t4, t5*}. This trains the model to predict the value of time step + 1, in the example that would *t5*. For the linear regressions, it is much simpler, the input remains the sample, but the target value will be *t5.* Another important thing to mention about time series forecasting is that a cross-validation like K-fold will not work because the model is dependent on the past sequences to predict the future. Instead the data is split into various periods that are sequential. For this experiment, the models have been trained from the start of 2018 until February 2020, the remainder of 2020 was used to test the models predictive ability.

In the beginning, we built our models as univariate. Meaning we would only input the Close price to output predictions of Close price. Then we improved upon this and made our models multivariate. We did so by computing two additional features: the Relative Strength Index (RSI) indicator and the Average True Range (ATR) indicator. RSI is a momentum oscillator that measures the speed and change of price movements. It oscillates between zero and 100. At 70 RSI signals a peak in momentum and at 30 a low in momentum. It is calculated using the Close price and by computing a ratio between average gain and average loss. The ATR is a volatility indicator that measures the range of how much the stock price moves during a given period. It is calculated by taking the difference between the high and the low price data of a stock.

2.2 Model Tuning

For the LSTM preprocessing, a Min Max scaler was used to scale our features. Four features were experimented with: Closing price, Volume, RSI and ATR.

Initially a simple model with a single hidden layer and one output layer was tested to make a prediction. The prediction results were improved by modifying the architecture to have two hidden LSTM layers and two output layers. After testing, we found that the optimal number of units for the two hidden layers to be 50, additional units made insignificant changes. For the hidden layers, the first layer contained 25 units and the second had one unit. When stacking LSTM layers, we must output a sequence rather than a single value for each input so that the subsequent LSTM layer can have the required 3D input. We did this by setting the return_sequences argument to True.

When compiling the model, mean squared error was used as a loss function and we found that the best optimizer was adam. Fitting the model, we first started out with only one epoch, and while the results were acceptable on the univariate model, increasing the number of epochs to 5 made significant improvements to the multivariate models and additional epochs made very small improvement to the mean squared error loss

Hyper-parameter tuning was attempted for the TCN with the use of the Optuna package. Testing was done to try and find the best optimizer and learning rate that would result in the lowest average validation Mean Squared Error (MSE). The list of optimizers looked at were AdamW, Adam, RMSprop, Rprop, and ASGD. An issue encountered while using Optuna is that there are few parameters that can be searched for. A notable example is for Adam, the betas were not able to be optimized. By hand tuning the learning rate, betas and weight loss parameters of the Adam optimizer, the validation loss of the model was able to be lowered, signally that Optuna may not return the best possible TCN training hyper- parameters.

2.3 Results

For model testing, various combinations of our four features were tested. We have decided on using Root Mean Square Error as the metric to evaluate the accuracy of our model's predictions. For both the LSTM and TCN, when testing the benchmark stocks, S&P500 and Apple, any combination of features did not provide a significant difference in the RMSE. With the AT&T stock, results were similar and any combination of features did not impact on the score. However it is important to note that whatever features were inputted, the models performed very well with a score between 0.625 and 1.25. When testing for Tesla, a high volatile stock, the results from the LSTM and TCN diverge. For the LSTM, in the univariate case where closing price was the sole feature, the model's predictions had a RMSE of 43. When introducing RSI, the RMSE increased drastically to 105. Our interpretation of this is that the model is trained on data up to 2020, where Tesla had low momentum and stayed in a stable range. Thus, the model could not map out any significance from the RSI values when trying to predict the volatility during the testing period. However, when adding ATR as a feature to the LSTM, the RMSE was the lowest at 35. Our interpretation of this is that since the ATR values give the range of how much the stock price moves in one day, the model was able to learn that Tesla was extremely volatile and managed to adapt its prediction to fit this volatility. (See Appendix for graphs of these results) As for the TCN, the model's RMSE stayed in a range between 16-17 irrespective of the combination of features.

3. Conclusions

We were not able to build a model that could reliably predict stock prices if deployed in a real-time situation. However, we were able to examine the importance that feature engineering takes when dealing with this type of data. Generally, the TCN performs better than the LSTM which confirms the findings of Bai et al. (2018) research. Due to the surprise shocks and unpredictability of real life events, the task of predicting future stock prices remains highly difficult. Perhaps with further domain expertise these models can perform better with a different map of features. One suggestion would be to build an event handler that would weigh recent events (news, changes in fundamental performance of the company, etc..) to supply the model with information related to surprise shocks.

Bibliography

Bai, Shaojie, et al. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling." ***arXiv*****, vol. v2, no. 803.01271, 2018, pp. 1-5.** ***arXiv*****, https://arxiv.org/pdf/1803.01271.pdf.**

Brownlee, Jason. "How to Develop LSTM Models for Time Series Forecasting." ***Machine Learning Mastery*****, 2018, https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/. Accessed 2020 November 20.**

Srivastava, Pranjal. "Essentials of Deep Learning : Introduction to Long Short Term Memory." ***Analytics Vidhya*****, 2017, https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/. Accessed 20 November 2020.**

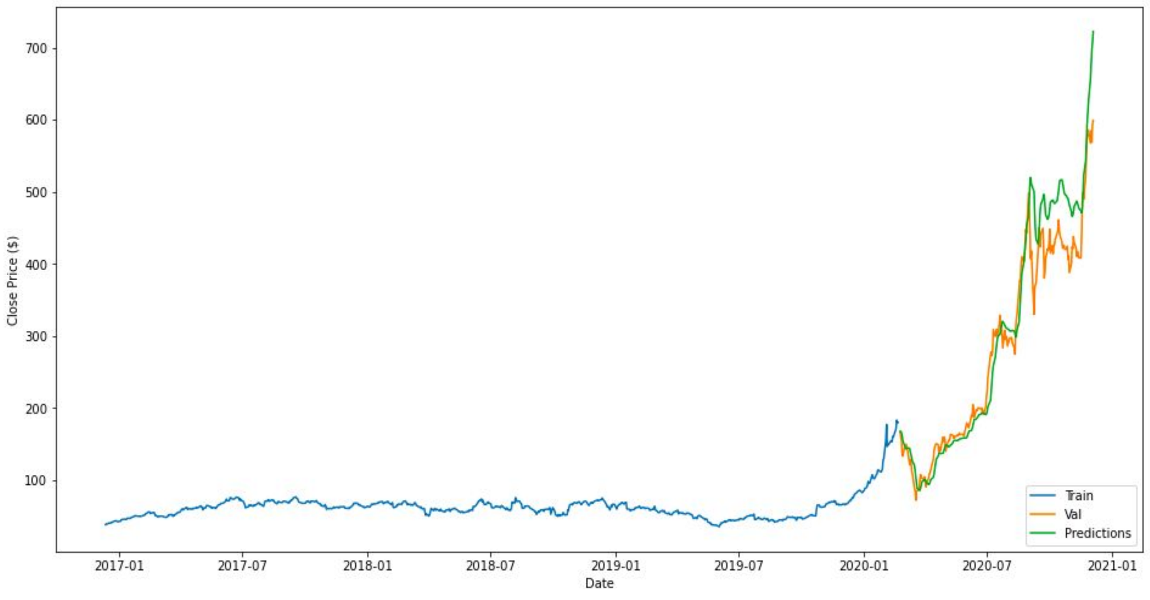
Appendix : Results

Testing Results

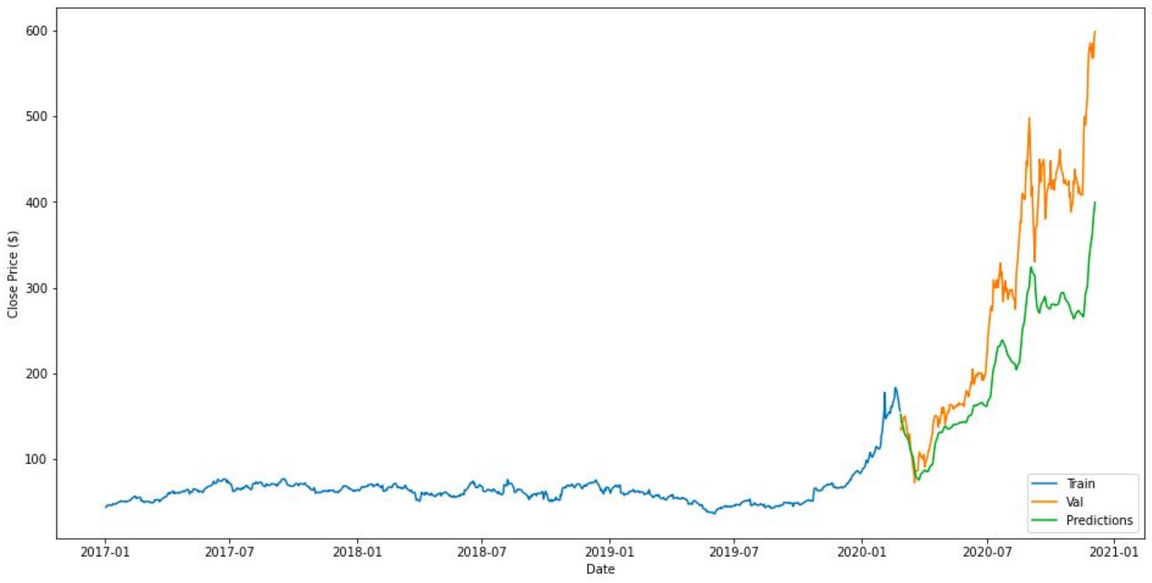
Model	S&P500	Apple	AT&T	Tesla
	RMSE	RMSE	RMSE	RMSE
Linear Regression	66.7	55.2	27.5	163.2
LSTM(U)	7.72	5.72	0.74	43.39
LSTM (M with RSI)	7.96	4.02	1.05	105.61
LSTM(M with ATR)	8.02	4.48	0.98	35.35
TCN (U)	6.38	2.73	1.25	15.71
TCN (M)	5.75	2.63	0.625	16.9

Table 1: Results from best performing models, where U and M signifies Univariate and Multivariate(Close and RSI) respectively

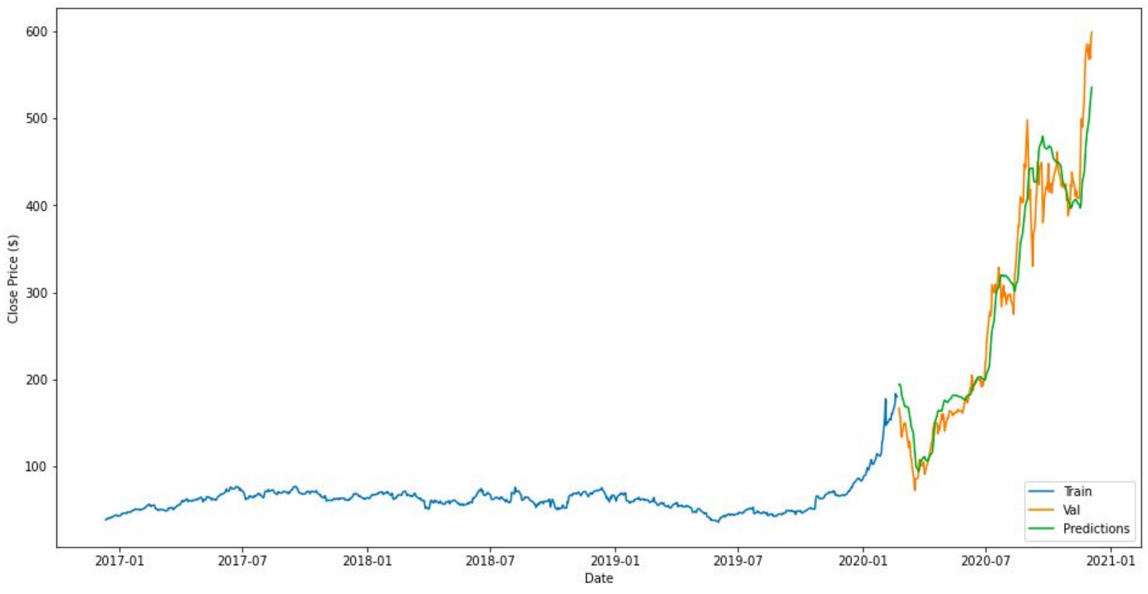
LSTM Model predictions for Tesla with univariate Close price



LSTM Model prediction for Tesla with Multivariate Close + RSI values



LSTM Model Predictions for Tesla with Multivariate Close + ATR values


Close Price ($)
Date
Train
Val
Predictions