Using a LSTM Model to Predict the Price of Apple Stock Prices

Candidate Number: 033666

November 25, 2020

Abstract

In this project we aim to make use of a Long Short-Term Memory (LSTM) model as a tool for predicting stock prices based on historical data. We set out to predict the stock price from day-to-day, and compare it to the actual figures for those dates. Recurrent neural networks (RNNs) have been shown to be powerful when processing time series data and LSTM has been one of the most successful architectures. Using such a model, we were able to make reasonable predictions of the stock price even when exposed to large amounts of turbulence due to the COVID-19 pandemic.

I certify that all material in this project which is not my own work has been identified.
Signed: 033666

Contents

1 Introduction

1.1 Contextualisation: Predicting Stock Price

This project will focus on predicting the price of AAPL stocks (Apple), a well known technology company with extensive data on its stock price history. Stock prices are dependent on time so present an interesting problem with regards to machine learning techniques.

As part of this project, we will be making use of historical data on the company's stock prices. There are a number of factors that can be considered:

- **Opening Price**: The price at which a security first trades when an exchange opens for the day.
- **Closing Price**: The price at which a security last trades before an exchange closes for the day.
- **High Price**: The highest price at which a security trades that day.
- **Low Price**: The lowest price at which a security trades that day.

There are a great number of financial indicators which make the data difficult to interpret and predict, hence there is a high amount of fluctuation. Prices can also be affected by external factors, such as public interests and the COVID-19 pandemic we are currently experiencing.

1.2 Motivation

Predicting the price of stock is an interesting problem, as it is a time-series. This means that previous data can potentially provide valuable information about future information. As such it seems pertinent to use a model that can use the data from previous days in its predictions.

Developing a model for the prediction of stock prices can also be a useful tool for analysts, allowing them to predict trends in the market before investing in different securities. This can lead to increased returns, an increase in the speed of analysis, and a reduction in the number of analysts that a firm may require.

The model may also aid in helping analysts recognise the most important indicators among the data. Thus, reducing the time spent on this part of analysis.

1.3 Project Aims

The goal of the project is to utilise a LSTM Neural Network to predict the price of a stock on dates based on previous performance of the securities. We will make use of the security's historical data, including opening, closing, high, and low price, for a series of dates. We aim to make good predictions of how the stock will perform and compare it to how it actually performed on the given dates. We will make comparisons for AAPL (Apple) shares as it has a wealth of historical data.

2 Literature Review

Stock prediction is a popular problem in both academics and industry as shown by the wealth of literature relating to the subject. A number of models have been proposed and tested in the past, in recent times many of these have involved machine learning techniques.

Roondiwala *et al.* made use of a LSTM model for a similar purpose in 2017 [1], they made use of historical stock data from Quandl for their training and testing data. They did some preprocessing, including discretization, normalization, and cleaning. Following this they trained the neural network using a selection of features. They measured the efficiency of their system by making use of the Root Mean Squared Error (RMSE). They ran experiments with a number of different parameter sets, each run with both 250 and 500 epochs, which can be seen in Figure 1. From their work they found that the parameter set High/Low/Open/Close gave the best results.

Selvin *et al.* ran comparisons between three different neural network architectures: LSTM, Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN). They experimented on minute-wise prices for a great many companies, making use of data from July 2014 to June 2015. The companies were from either the IT or Pharma sector. Their results (shown in Figure 2) indicate that CNNs gave the greatest results and note that this is due to the fact that CNNs did not rely on previous information for their prediction, whereas the RNN and LSTM architectures did. Due to the volatile nature of the stock market, previous dynamics and patterns do not always apply which causes learning errors in RNN and LSTM [2]. They also compared the results to ARIMA, a linear model that is also used for forecasting, and all of the neural network architectures gave much better results.

Parameters	No. of Epochs	Training RMSE	Testing RMSE
Open/ Close	250	0.01491	0.01358
Open/ Close	500	0.01027	0.00918
High/Low/Close	250	0.01511	0.014
High/Low/Close	500	0.01133	0.01059
High/Low/Open/ Close	250	0.0133	0.01236
High/Low/Open/ Close	500	0.00983	0.00859

Figure 1: *Results of Roondiwala et al. 's findings.* [1]

COMPANY	RNN	LSTM	CNN
Infosys	3.90	4.18	2.36
TCS	7.65	7.82	8.96
Cipla	3.83	3.94	3.63

Figure 2: *Results of Selvin et al. 's findings.* [2]

Some have even expanded on the number of considered factors, for example Zhuge *et al.* [3] combined behavioural data with emotional data to increase the effectiveness of predictions, with results showing to be effective on a number of stocks from the Shanghai Composite Index [4]. The results show that using emotional data alongside exchange data provides more accurate results.

3 Method

Stock price prediction is very complex and can be affected by financial and external factors, as such it is difficult to construct an analytical model, though attempts have been made with models such as ARIMA [5]. As such, machine learning methods appear to be appropriate. Data is rather simple to come by, as there are many sites that provide historical data on a multitude of stocks.

3.1 Recurrent Neural Networks (RNNs)

RNNs are a number of Neural Network architectures for dealing with series input, with no predetermined size limit. This means that it can make use of the past to learn and influence its decisions. In addition to using its training data, it also makes use of prior input when generating output. They have been used in a number of areas including machine translation, speech recognition, and time series prediction.

Given the nature of the data we are working with, the ability to deal with time series data is useful and provides an interesting insight into the theory that "past results are no guarantee of future success" [6]. One problem to consider is that, due to the recurrent nature of the networks, computation can be slow. With the amount of data being used and the power of the environment being used, this should not be an issue.

3.1.1 Long-Term Dependencies Problem

One of the large advantages of RNNs over traditional Artificial Neural Networks (ANNs) is the ability to use past data to influence decisions. Traditional RNNs perform well when using information in the recent past, the problem comes when the useful context is many time steps past. The longer in the past that the relevant context exists, the more difficult it is for the RNN to use it. At a certain distance in the past, the RNN becomes unable to use the context for its decision.

In theory, this problem can be overcome by RNNs. The problem is explored in more detail by Hochreiter [7] and Bengio [8].

3.2 Long Short-Term Memory (LSTM) Neural Networks

LSTMs were introduced by Hochreiter Schmidhuber in 1997 [9]. They are an architecture of the RNN family that are capable of learning long-term dependencies. They have been shown to be well-suited to classification, processing, and predictions on time series data.

LSTMs are designed in such a way as to avoid the long-term dependency problem that is described in §3.1.1. LSTMs consist of a series of LSTM cells, their structure is shown in Figure 3. This cell allows the LSTM to remove or add information using structures called gates, which optionally let information through. Gates are made of a sigmoid layer and a pointwise multiplication. There are three such gates in a cell.

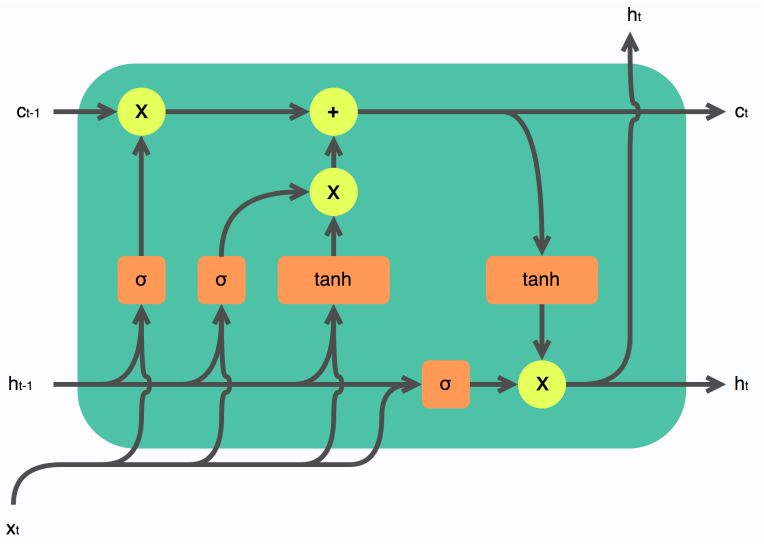


Figure 3: *Diagram of an LSTM cell from [10]*

The first step of the cell determines what information is going to be lost (or "forgotten"). This decision is made by the forget gate layer, it determines for each entry of C_1 whether to retain the information or forget it. It does so using the following equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The subsequent step is to determine what new information is going to be stored in the cell's state. Firstly, the input gate layer determines which values will be updated. Subsequently, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Next, the cell state is updated. We take C_t as the new state and C_{t-1} as the previous state. As before we take f_t as the forget gate layer output, i_t as the input gate layer outputs and $\tilde{C}_t$ as the candidate values.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

The final step of the cell is to determine its output. The output is a filtered version of the cell state. A sigmoid layer is run on the parts of the state that are going to be output. The cell state is passed through a tanh function and multiplied by the output of the sigmoid layer.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

3.2.1 Hyperparameters

Whenever building machine learning models, it is always pertinent to consider the hyperparameters of the model. For a LSTM, we need to consider the number of nodes and layers, and the dropout layer sensitivity. The dropout layer probabilistically removes inputs to a layer so as to make the nodes in the network more robust to inputs. After some research, it was found that 20% was regularly used as a compromise for retaining accuracy and preventing overfitting. After some testing, we settled on using 4 LSTM layers of 50 nodes.

3.3 Data

Yahoo! Finance has a wealth of historical data on stocks, we shall be using the historical data for Apple Inc. (AAPL) which can be found at [11]. The data contains daily entries for the stock. Each entry has a date stamp, high price, low price, open price and close price. It also contains a couple of other pieces of data, namely the adjusted closing price and the volume. These two factors are not considered in this project. The data used is from 12/11/2015 to 11/11/2020 and appears to be complete on an initial inspection, though standardisation will aid in the model accuracy.

4 Discussion

A number of experiments were run regarding the training and testing of the model with varying numbers of training epochs and batch sizes. Tables 1, 2 show the Root Mean Squared Error (RMSE) in testing and training of the model with each set of hyperparameters.

Epochs	Batch Size	Scaled Training RMSE
100	32	0.0375
100	64	0.0393
100	128	0.0483
250	32	0.0335
250	64	0.0341
250	128	0.0359
500	32	0.0278
500	64	0.0295
500	128	0.0319

Table 1: Training RMSE for different experiments

It can be seen from Table 1 that, in general, increasing the batch size increases the training error and increasing the epochs reduces it. This makes sense as it is becoming better fit to its testing data the more iterations it goes through.

Epochs	Batch Size	Testing RMSE
100	32	14.313
100	64	10.327
100	128	9.516
250	32	18.192
250	64	10.563
250	128	11.193
500	32	30.228
500	64	22.621
500	128	14.585

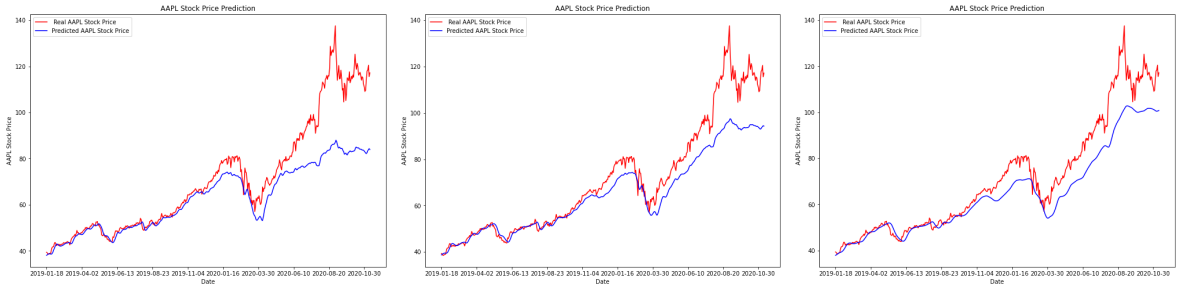
Table 2: Testing RMSE for different experiments

Conversely to training, it can be found from Table 2 that the testing error decreases with larger batch sizes, suggesting that the model is overfitting to the training data with smaller batches. A similar thing seems to be occurring with the increased number of epochs. The results indicate that the best model in testing was with 100 epochs on a batch size of 128. This can be further seen in Fig. 4.

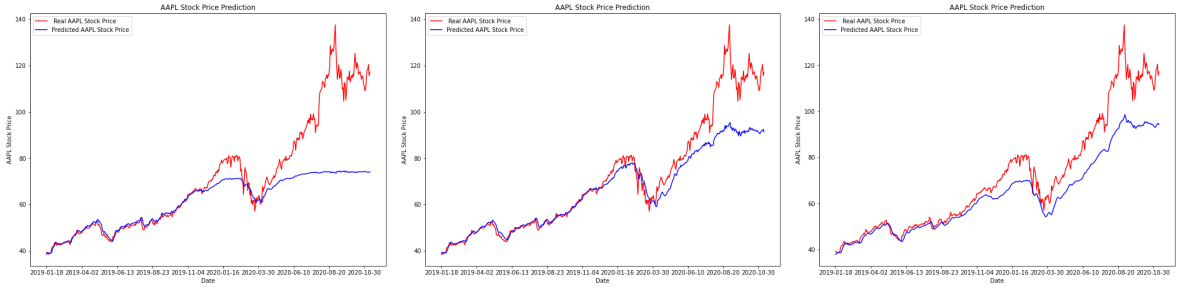
Looking at Figs. 4g 4h 4i it can be seen that 500 epochs gives very poor predictions when the spike due to the pandemic occurs. Meanwhile, models with only 100 epochs react to the change rather well.

5 Conclusions

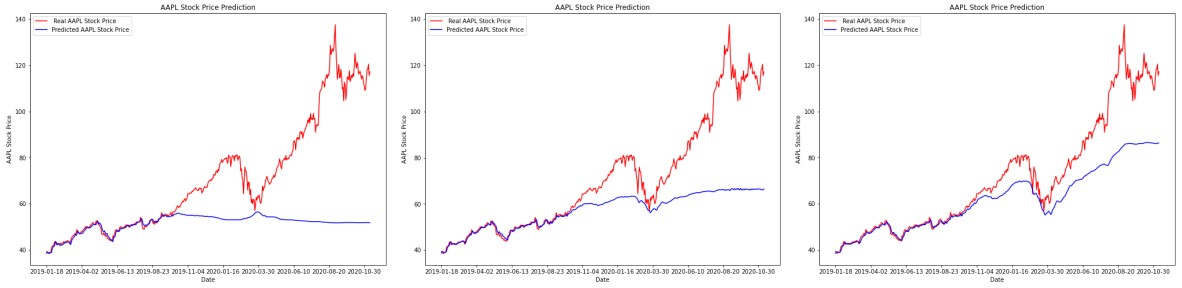
In this project, we aimed to utilise an LSTM network to predict the price of Apple stocks based on previous performance. Further to this, we looked to perform experiments to find patterns between hyperparameters and accuracy, as measured by the model's RMSE. We managed to generate a model that dealt well with unforeseen leaps in price that were caused by the ongoing COVID-19 pandemic, which indicates the method is rather robust and could work well at predicting further real data when trained on the relevant historical data. Further work to improve this work would be to test different model structures that take into account different combinations of attributes from the data, thus narrowing down which features are the most influential on the overall price. Further experimenting with different numbers of epochs and batch sizes could also yield an optimal set of hyperparameters given the time. Overall, the results that the model has produced have been satisfactory and could potentially aid in giving insights on when to invest and when to sell.



(a) 100 Epochs, Batch Size 32 (b) 100 Epochs, Batch Size 64 (c) 100 Epochs, Batch Size 128



(d) 250 Epochs, Batch Size 32 (e) 250 Epochs, Batch Size 64 (f) 250 Epochs, Batch Size 128



(g) 500 Epochs, Batch Size 32 (h) 500 Epochs, Batch Size 64 (i) 500 Epochs, Batch Size 128

Figure 4: Predictions from each experiment

References

[1] Murtaza Roondiwala, Harshal Patel, and Shraddha Varma. "Predicting stock prices using LSTM". In: *International Journal of Science and Research (IJSR)* 6.4 (2017), pp. 1754–1756.

[2] Sreelekshmy Selvin et al. "Stock price prediction using LSTM, RNN and CNN-sliding window model". In: *2017 international conference on advances in computing, communications and informatics (icacci)*. IEEE. 2017, pp. 1643–1647.

[3] Qun Zhuge, Lingyu Xu, and Gaowei Zhang. "LSTM Neural Network with Emotional Analysis for prediction of stock price." In: *Engineering letters* 25.2 (2017).

[4] *SHCOMP: Shanghai Composite Index Overview*. URL: https://www.marketwatch.com/investing/index/shcomp?countrycode=cn.

[5] Selva Prabhakaran. *ARIMA Model - Complete Guide to Time Series Forecasting in Python*. Sept. 2020. URL: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/.

[6] Lawrence Carrel. *Study Proves Past Results Don't Predict Future Results*. Dec. 2020. URL: https://www.forbes.com/sites/lcarrel/2020/02/15/study-proves-past-results-dont-predict-future-results/.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[8] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[9] Sepp Hochreiter and Jürgen Schmidhuber. "LSTM can solve hard long time lag problems". In: *Advances in neural information processing systems*. 1997, pp. 473–479.

[10] *Long short-term memory*. Jan. 2021. URL: https://en.wikipedia.org/wiki/Long_short-term_memory.

[11] *Apple Inc. (AAPL) Stock Historical Prices Data*. Jan. 2021. URL: https://finance.yahoo.com/quote/AAPL/history?period1=1447286400&period2=1605139200&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true&guccounter=1.