## Introduction

Stock forecasting is a challenging task, difficult at best and impossible at worst. The closing price of a stock is highly sensitive to a number of factors, known and unknown. It is the latter that makes this quantity highly volatile. External factors include interest rates, inflation, GDP, political events and other socio-economic factors. Since the stocks are a valuation of a company's output, there are also internal factors to the market: competition, reputation of the company, and supply and demand. There may also be unknown factors; unforeseen circumstances such as natural disasters, war, a recession or even a pandemic. It therefore would be wise to suggest that a technical analysis of stock prices is naive [1], however we sought to test how limiting this hypothesis can be.

## Experimental setting

This study involves stock forecasting using two machine learning methods. Through analysing historical data, future prices can be predicted. This is a regression problem, ultimately involving a time series. First, a regression-based model will be used, followed by an artificial neural network (ANN). Using the *yfinance* dataset, we will analyse the stock prices of the FAAMG group [2]. We will be selecting a time range of 2015-2021. The year 2020 is particularly volatile, so provides an interesting point of discussion. Our first candidate is the Auto-Regressive Integrated Moving Average (ARIMA) model. This is a classic example of a traditional regression model, developed in the 1970's [3] as an improvement over earlier models [4]. The second candidate is the Long Short-Term Memory network, an ANN which belongs to the class of recurrent neural networks (RNNs). LSTMs can predict entire sequences of values, outperforming other networks [5]. LSTMs have already replicated trading strategies involving market indicators [6], so an application in technical analysis should not be unfamiliar.

### Data analysis

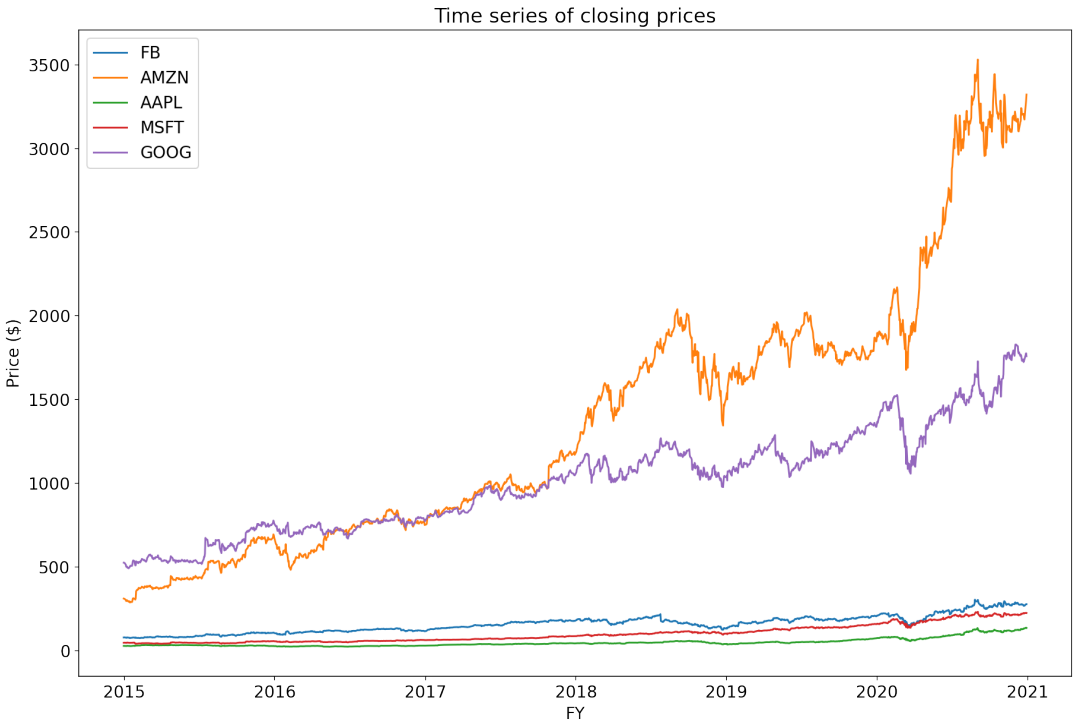



***Figure 1. The closing prices of the stocks. All stocks displayed a bullish trend.***

At first glance, AMZN and GOOG appear to outperform the competition. However, both have an inflated pricing for their stocks [7] [8]. There is an obvious trend: AMZN skyrocketed from early 2020; this is likely due to *working-from-home* becoming common. Naturally, the world's largest online marketplace would benefit from this [9].

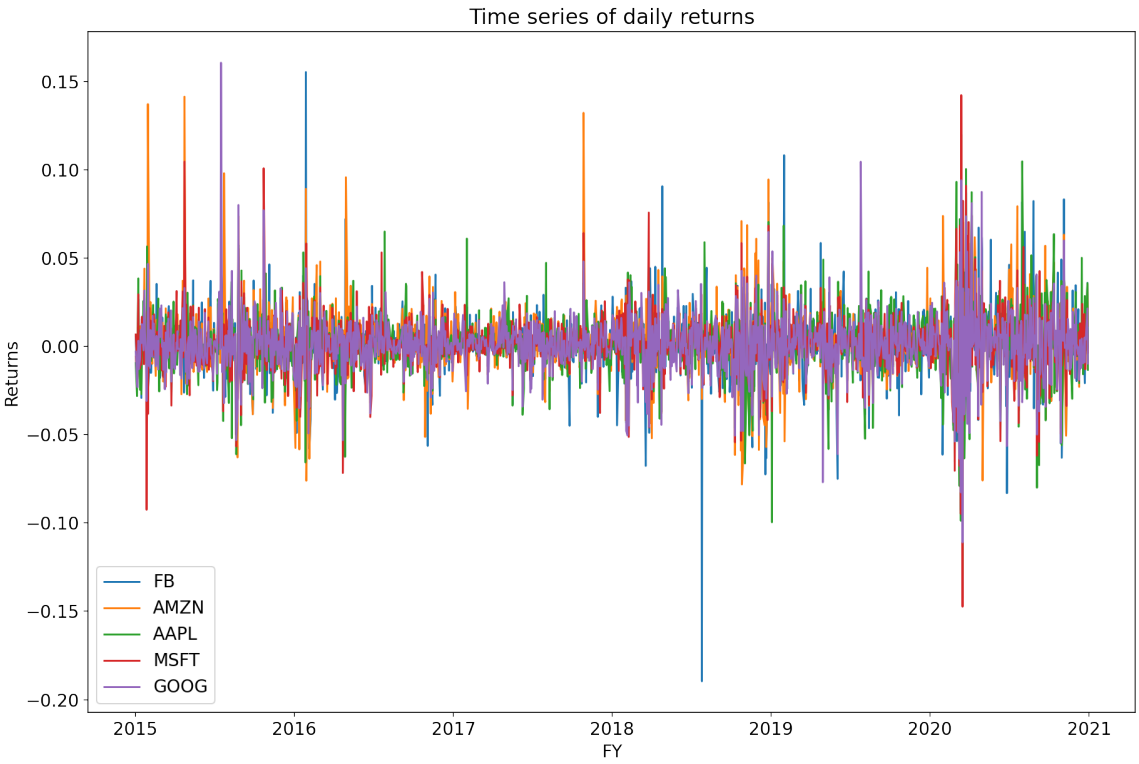



***Figure 2. The daily returns of the stocks shows more volatility, particularly in early 2020.***

Conducting a statistical exploration of the data, for both closing prices and returns, we found interesting conclusions for the latter. Although the data initially hinted towards a bell-curve distribution, further investigation exposed the data as being non-Gaussian.

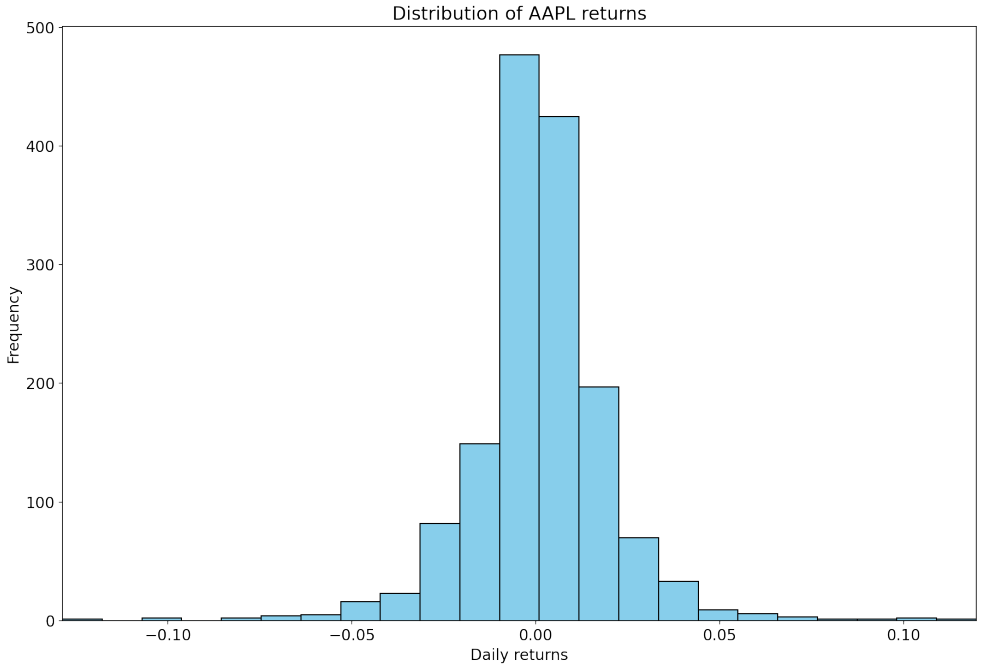



***Figure 3. AAPL returns. There appears to be a bell-curve distribution.***

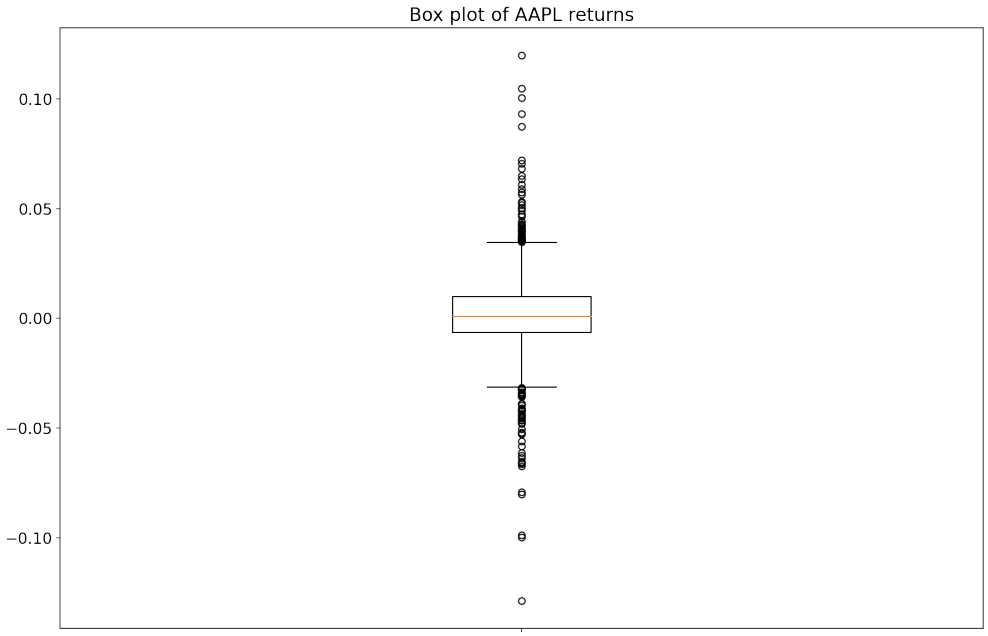



***Figure 4. The corresponding box plot. There are several outliers present.***

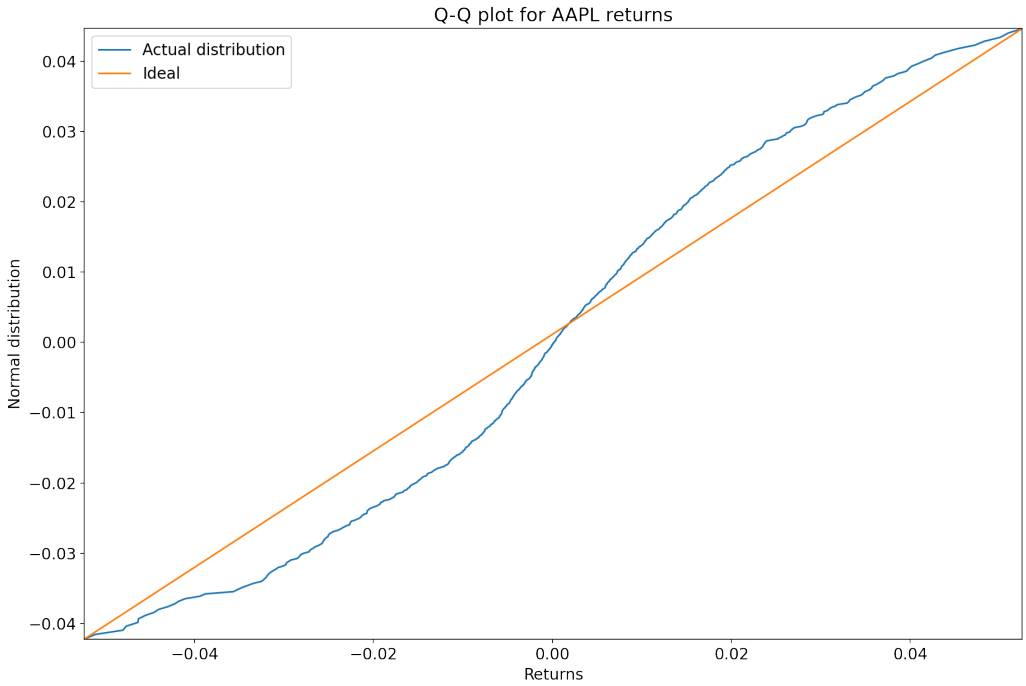



***Figure 5. The QQ plot for returns. The fat tails of the distribution may be a result of cluster volatility (see below).***

A 30-day rolling standard deviation (SD) displays the volatility of the stocks. There are hints of seasonality from peak festive periods. This is known as *cluster volatility*, with a changing variance throughout the time period, rendering the series unstable [10].

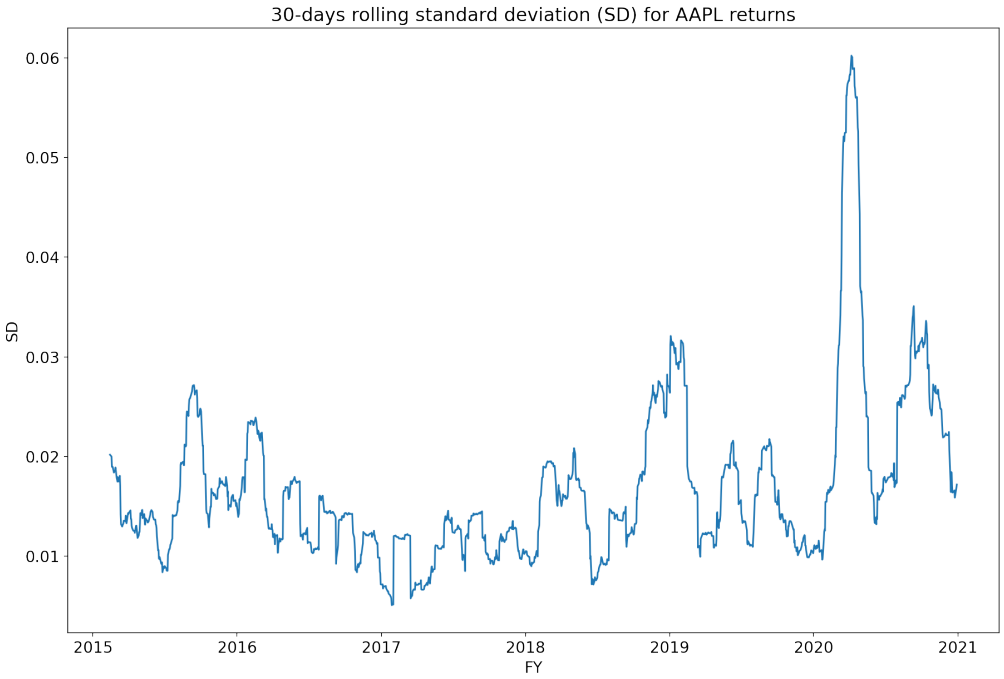



***Figure 6. The 30 day rolling SD. The peak SD is reached in early 2020, as expected.***

The PACF was also plotted for highlighting compatibility with ARIMA modelling [11].

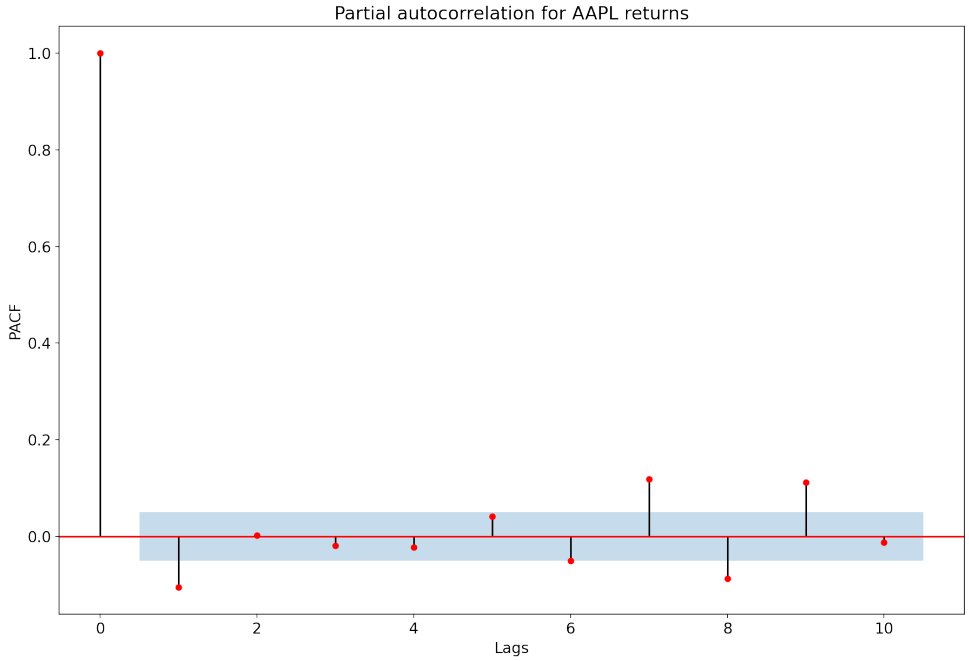



***Figure 7. The PACF. There was an autoregressive term within the data of order 1, confirming suitability for ARIMA modelling [12].***

**On missing values:** this dataset appeared complete (after a check for null data), however please note that trading only commences on *trading days*. This did not become an issue, because the stocks all traded on the same days.

## Performance metrics

There are three primary metrics which are used to characterise the performance of a regression model [13].

**Mean absolute error** is the absolute difference between a prediction and its true value.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x| \tag{1}$$

**Mean squared error** is the average squared difference between a prediction and its true value.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \left(x_i - \hat{x}_i\right)^2 \tag{2}$$

**Root mean squared error** is root of MSE. Analogous to standard deviation, hence sometimes known as *standard* error.

$$\mathrm{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{\left(\hat{x}_i - x_i\right)^2}{n}} \tag{3}$$

**Mean absolute percentage error** is an additional metric [14] which expresses the accuracy of a dataset containing predicted values $F_t$ with respect to the dataset of true values $A_t$ as the following ratio:

$$\mathrm{MAPE} = \frac{100}{n}\sum_{t=1}^{n} \left|\frac{A_t - F_t}{A_t}\right| \tag{4}$$

## Proposed experiments

Both models were evaluated on five stocks. Performance metrics would be averaged between stocks. A training split of 60/40 for ARIMA and 60/20/20 for LSTM were chosen. It is important to bear in mind that the ARIMA model uses a larger test set, so the graph plot ranges would differ. When comparing the two, it is important to make sure that the same time periods are used[1].

ARIMA tuning would be carried out by the **auto.arima** function (see next section). The search values of $(p, d, q)$ are in the range [0, 3].

For the LSTM model, two versions would be run: a CPU runtime, and a GPU-accelerated version. The latter is expected to outperform the former in all areas. Tuning is carried out by the Keras Tuner, and a Bayesian search is used to find the optimum hyper model [15]. The LSTM would be run with 50 epochs, a batch size of 50, and 64 neurons. A recurrent dropout would be used to prevent overfitting from excess epochs. Activation functions are automatically trialled in the GPU accelerated version.

---

[1] Since the LSTM skips the first half of the 40% split (using it for validation), only the second half of the ARIMA predictions are suitable for comparison.

## Methodology

Let us refresh our terminology. The *closing prices* are the price changes of a stock before the end of the day [16]. The daily stock *returns* are the percentage changes between prices, and are analogous to their first order differences [17]. We will now discuss model theory.

### ARIMA

The ARIMA model consists of the parameters $(p, d, q)$ which are the orders of the autoregressive (AR) model, differencing (I) and moving average (MA) respectively. The key concept here is that the observed value $Y_t$ is related to preceding 'lags' in the series $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$. Similar to linear regression, the regressor in this case is the *same* variable, a few lags behind. The data must be stationary, so differencing will be used. Regression is completed by the MA component, which uses error terms from previous values.

For a datapoint $Y_t$ in a time series with AR parameters $\alpha_i$, MA parameters $\theta_i$ and i.i.d. random error terms $\epsilon_t$, the forecasting can be given by

$$Y_t - \alpha_1 Y_{t-1} - \cdots - \alpha_{p'} Y_{t-p'} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \tag{5}$$

The polynomial will have a unit root of multiplicity $d = p - p'$ such that there are $d$ unit roots. In this scenario, the **auto.arima** function is used to process multiple stocks. This mechanism is far quicker than finding the values of $(p, d, q)$ manually, with an automatic approach [18] taken to minimise the Akaike Information Criteria (AIC) [19]. A popular measure for model selection [20], the AIC can be related to these values for non-seasonality through

$$\mathrm{AIC} = -2\log(L) + 2(p + q + k) \tag{6}$$

Here, $L$ is the likelihood of the data, $k$ the model intercept. This way, the PACF will not have to be calculated every time. The model utilises the **ndiffs** function to determine the order of differencing required, performing a joint ADF-KPSS test [21] to check for stationary. The model is then iterated for each timepoint in the test set.

#### Data preprocessing

A non-seasonal ARIMA model was used. Seasonality is important in time series analysis, and can obscure trends if not handled properly [22]. These features were removed prior to modelling, by subtracting the *log returns* [23] from the training set. A validation set was not required by **auto.arima** hence the 60/40 split.

## LSTM

The advantage of RNNs over ANNs is the ability to save output of a given layer, and feed it back to input. Aptly named, this results in a *recurrent* process. The predecessors to LSTMs are the Hopfield [24], Elman [25] and Jordan networks [26]. These employed a hidden layer vector $h_t$ and an output vector $y_t$, represented by the following

$$h_t = \sigma_h \left( w_h x_t + u_h y_{t-1} + b_h \right) \tag{7}$$

$$y_t = \sigma_y \left( w_y h_t + b_y \right) \tag{8}$$

The drawback of these networks is the problem of the vanishing gradient [27], occurring during stochastic gradient descent [28]. A workaround was devised by Hochreiter and Schmidhuber in 1997, named the Long Short-Term Memory (LSTM) network [29]. The introduction of the Constant Error Carousel (CEC) was proposed to avoid the vanishing gradient, truncating all delta errors except those over the memory state. This results in delta errors arising at the gates and cell input, but not being propagated back [30]. An LSTM unit consists of a cell, an input gate, an output gate and a forget gate. It can 'remember' values in a given time interval, with the combination of gates regulating the flow of information.

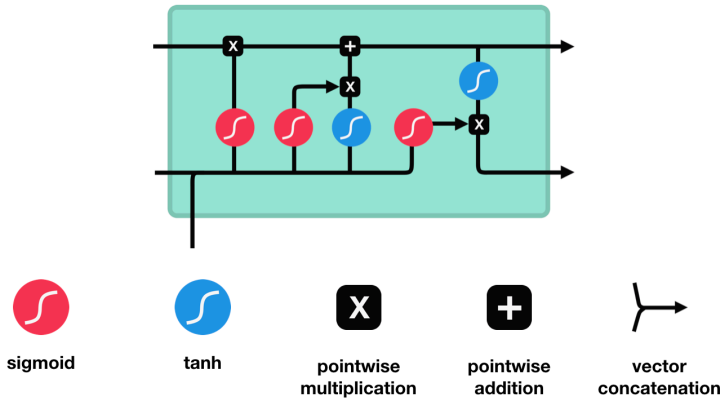



***Figure 8. An LSTM cell. Source: TowardsDataScience [31].***

A forward pass in an LSTM unit is described by these relations

(9-14)

$$f_t = \sigma_g \left( W_f x_t + U_f h_{t-1} + b_f \right)$$
$$i_t = \sigma_g \left( W_i x_t + U_i h_{t-1} + b_i \right)$$
$$o_t = \sigma_g \left( W_o x_t + U_o h_{t-1} + b_o \right)$$
$$\tilde{c}_t = \sigma_c \left( W_c x_t + U_c h_{t-1} + b_c \right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \sigma_h \left( c_t \right)$$

Where $c_0 = 0$, $h_0 = 0$, and the operator $\circ$ is the Hadamard Product [32] (returns an output of the same dimension as its inputs). The matrices $W_q$ and $U_q$ represent the weights of input and recurrent connections, while lowercase variables represent gate vectors [33].

### Data preprocessing

There were three stages. Firstly, the data was normalised to [0,1] using the MinMax scaler. This was followed by a reshape, to convert the data into a 3D tensor (batch size, time steps $t = 50$ and features $n = 1$ i.e. closing prices).

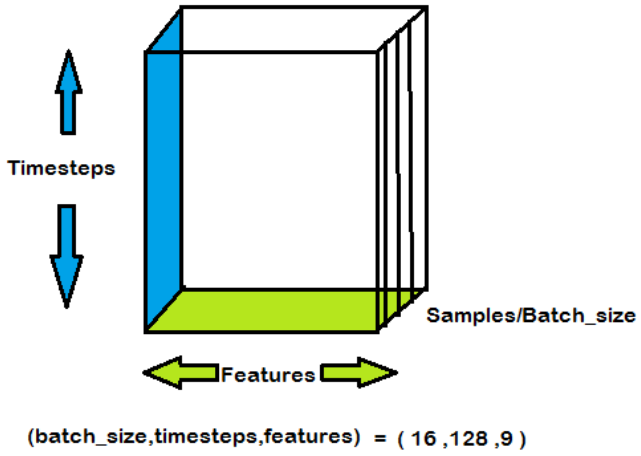



***Figure 9. An illustration of the 3D tensor used by Keras, with example sizes. Source: Medium [34].***

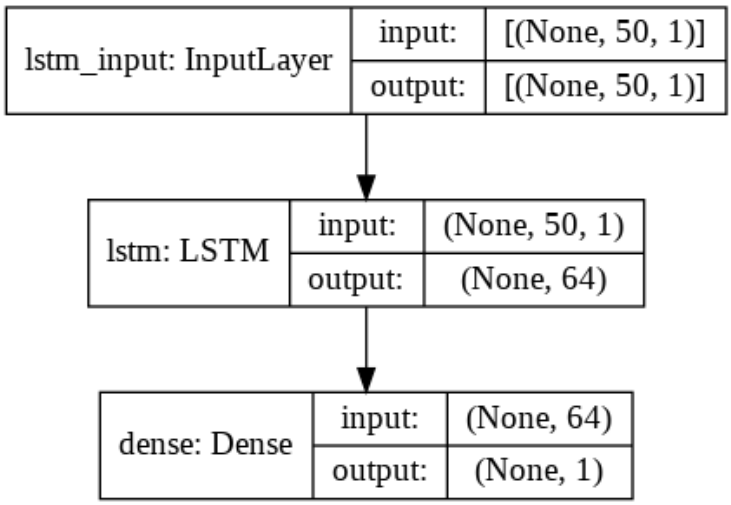



***Figure 10. The LSTM layer schematic, generated from our program.***

## Results and analysis

### ARIMA

In almost all cases, the optimum values of $(p, d, q)$ were $(3,1,0)$ with the exception of FB, preferring $(3,2,0)$. The AIC-minimising polynomials of best fit were

```
ARIMA(3,1,0)(0,0,0)[0]

ARIMA(3,2,0)(0,0,0)[0]
```

For precise results, please see **Appendices A-B.**

### LSTM

The GPU-accelerated application outperformed the CPU runtime in all areas. For a sense of scale, see the findings below

```
For evaluation with the same data and parameters, the difference in execution time is
380.11s. The GPU-accelerated LSTM ran 1181.76% faster than the CPU runtime version.
```

An enormous increase of > 1000% computation speed, shows the formidable capabilities of a GPU-powered neural network.

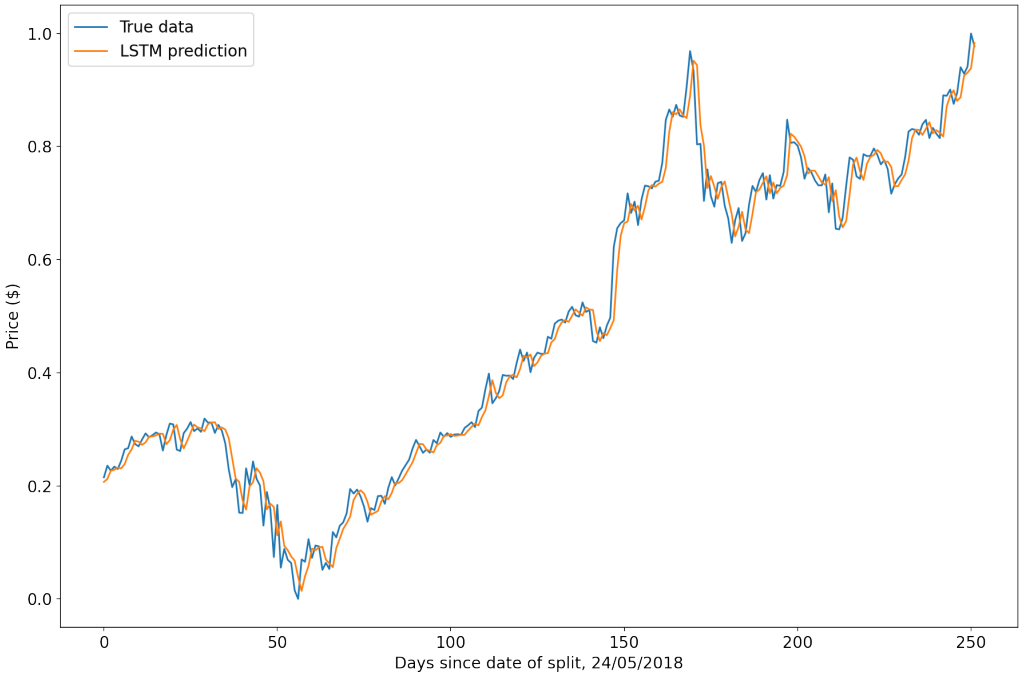
For precise results, please see **Appendices C-G.**

### Comparing the models

The LSTM's performance was far more erratic than the ARIMA's. Being highly stochastic in nature, the LSTM algorithm would only stabilise if enough epochs and trials were being run. The Bayesian search was an effective tuner for model selection, and scaled well to the GPU application. The overall performances on all fives stocks for each model are shown below

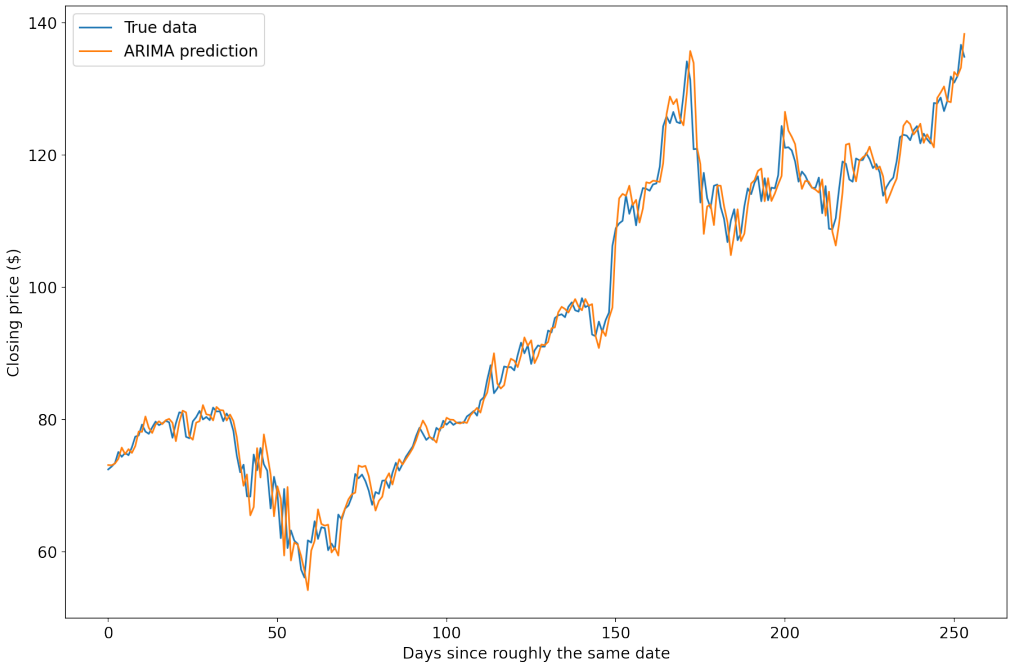
```
ARIMA performance for the FAAMG group:
Averages for MAE=11.47140, MSE=604.56885, RMSE=16.76263, MAPE=0.01548% and
execTime=65.678s

LSTM performance for the FAAMG group:
Averages for MAE=91.12595, MSE=35655.89410, RMSE=102.54812, MAPE=6.65438% and
execTime=53.786s
```

Whilst the LSTM performance does not appear impressive, it is important to note that this class contained more outliers. The ARIMA applications were more consistent in comparison.




```
LSTM metrics:
MAE=1.87004
MSE=6.71871
RMSE=2.59205
MAPE=2.03850%
execTime=33.63s
```

***Figure 11. The output of the LSTM for predicting 250 days into the future.***




```
ARIMA metrics:
MAE=1.3182
MSE=4.2378
RMSE=2.0586
MAPE=0.01830%
execTime=103.69s
```

***Figure 12. The same data, predicted by the ARIMA model.***

## Conclusion

The ARIMA model is very accurate for short term forecasting, and outperforms the LSTM in most metrics. The LSTM however, has greater potential for longer term predictions due to its more advanced mechanism [35]. LSTMs are more adaptable to new features that may need to be analysed, and potential is supercharged by GPU capabilities. ARIMA's model accuracy is high, however when applying to longer term data it is known to overfit. This is a consequence of multicollinearity.

Future improvements could include further optimising training split sizes for each model, and plotting a series of splits as a function of performance to find the optimum value. Varying the number of epochs, batch sizes and neurones in the LSTM could also provide interesting results. Both models could even be combined to form an ensemble [36], where the merits of each would focus on different parts of the dataset.

Finally, the most effective use of an ANN is using multiple feature sets. The true potential of the LSTM can not be realised from merely forecasting a stock with technical analysis. The network could be adapted to account for other factors such as observing internet trends and articles hinting at market information [37], or text mining [38] (which is becoming increasingly promising with developments in NLP).

Time-based cross validation would be a better means of training/testing the data in the future, and using more financial data for parameters i.e. *High, Low, Adjusted Close* prices would also be a good move.

# Bibliography

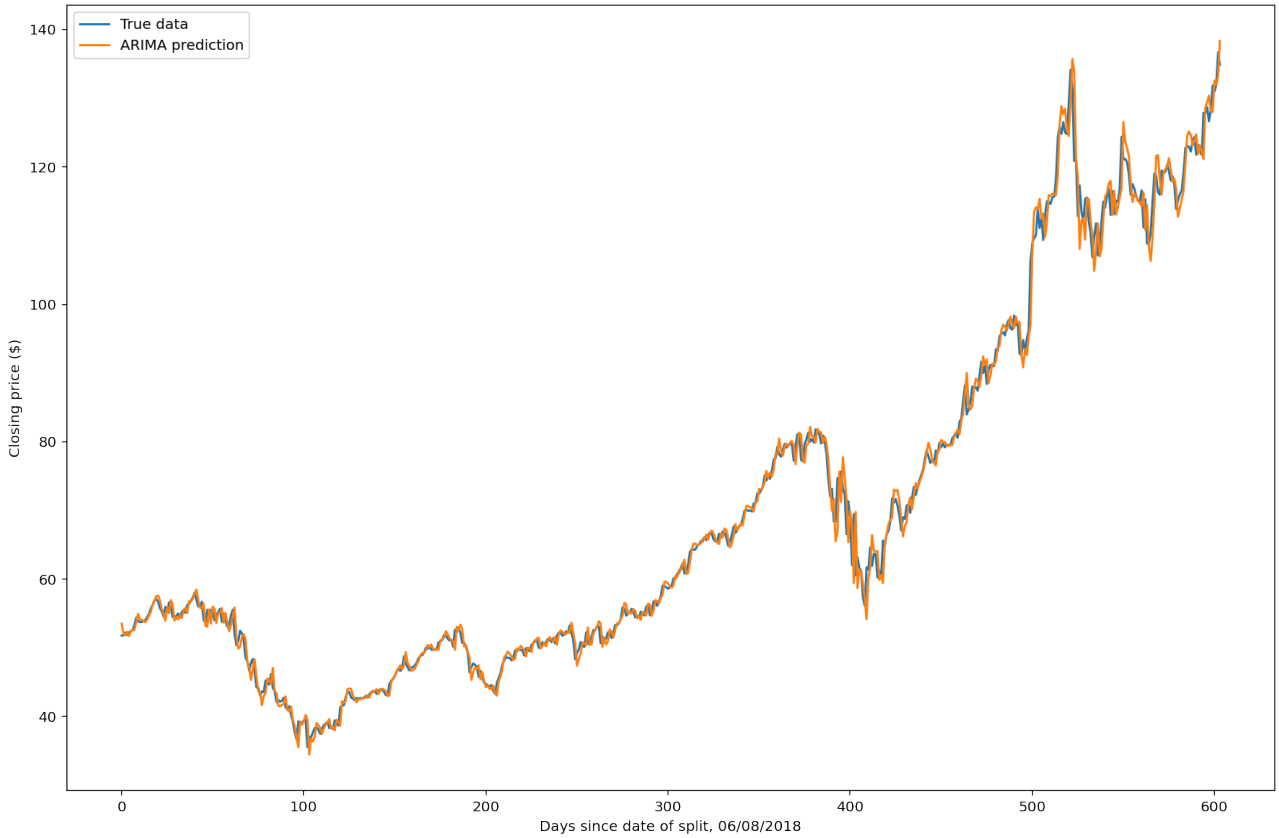

[1] A. Hayes. "Technical Analysis Definition." Investopedia. https://www.investopedia.com/terms/t/technicalanalysis.asp (accessed Aug 8, 2021).
[2] A. Hayes. "FAAMG Stocks." Investopedia. https://www.investopedia.com/terms/f/faamg-stocks.asp (accessed 8 Aug, 2021).
[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
[4] P. Whittle, "Hypothesis Testing in Time Series Analysis, 1951," *Almquist and Wiksell, Upssala,* 1951.
[5] N. K. Manaswi, "RNNs and LSTMs," in Deep Learning with Applications Using Python: Springer, 2018, pp. 115-126.
[6] L. Troiano, E. M. Villa, and V. Loia, "Replicating a trading strategy by means of LSTM for financial industry applications," *IEEE transactions on industrial informatics,* vol. 14, no. 7, pp. 3226-3234, 2018.
[7] J. Ballard. "Why Is Amazon's Stock So Expensive? | The Motley Fool." The Motley Fool. https://www.fool.com/investing/2021/02/19/why-is-amazons-stock-so-expensive/ (accessed 1 Aug, 2021).
[8] Zacks. "Why Google's Stock has a Higher Valuation than Apple's (AAPL)." Nasdaq. https://www.nasdaq.com/articles/why-googles-stock-has-higher-valuation-apples-aapl-2016-02-02 (accessed 1 Aug, 2021).
[9] Trefis. "Amazon's Stock Is Booming Despite The Coronavirus Pandemic, But What's Next?" Forbes. https://www.forbes.com/sites/greatspeculations/2020/04/27/amazons-stock-is-booming-despite-the-coronavirus-pandemic-but-whats-next/ (accessed 2 Aug, 2020).
[10] R. Cont, "Volatility clustering in financial markets: empirical facts and agent-based models," in *Long memory in economics*: Springer, 2007, pp. 289-309.
[11] J. Brownlee, "A Gentle Introduction to Autocorrelation and Partial Autocorrelation - Machine Learning Mastery," 2017. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/.
[12] Minitab. "Interpret the partial autocorrelation function (PACF)." https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/time-series/how-to/partial-autocorrelation/interpret-the-results/partial-autocorrelation-function-pacf/ (accessed 8 Aug, 2021).
[13] J. Brownlee. "Regression Metrics for Machine Learning - Machine Learning Mastery." https://machinelearningmastery.com/regression-metrics-for-machine-learning/ (accessed 2 Aug, 2021).
[14] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing,* vol. 192, pp. 38-48, 2016.
[15] GCBC. "Hyperparameter Search With Bayesian Optimization for Keras (CNN) Classification and Ensembling - Machine Learning Applied." https://machinelearningapplied.com/hyperparameter-search-with-bayesian-optimization-for-keras-cnn-classification-and-ensembling/ (accessed 7 Aug, 2021).
[16] A. Hayes. "What Is a Closing Price?" Investopedia. https://www.investopedia.com/terms/c/closingprice.asp (accessed 8 Aug, 2021).
[17] A. Hayes. "Return." Investopedia. https://www.investopedia.com/terms/r/return.asp (accessed 8 Aug, 2021).
[18] T. Smith. "PMDARIMA 1.8.2 Documentation." Alkaline-ML. https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html (accessed 7 Aug, 2021).
[19] A. Keshvani. "Using AIC to Test ARIMA Models." CoolStatsBlog. https://coolstatsblog.com/2013/08/14/using-aic-to-test-arima-models-2/ (accessed.
[20] S. Hu, "Akaike information criterion," Center for Research in Scientific Computation, vol. 93, 2007.
[21] P. Kębłowski and A. Welfe, "The ADF–KPSS test of the joint confirmation hypothesis of unit autoregressive root," *Economics Letters,* vol. 85, no. 2, pp. 257-263, 2004.
[22] R. J. Hyndman, "The interaction between trend and seasonality," *International Journal of Forecasting,* vol. 20, no. 4, pp. 561-563, 2004.
[23] L. Quigley and D. Ramsey, "Statistical analysis of the log returns of financial assets," *Financial mathematic, University of Limerick,* vol. 32, 2008.

[24] J. J. Hopfield, "Hopfield network," *Scholarpedia,* vol. 2, no. 5, p. 1977, 2007.
[25] S. C. Kremer, "On the computational power of Elman-style recurrent networks," *IEEE Transactions on neural networks,* vol. 6, no. 4, pp. 1000-1004, 1995.
[26] J. Jordan, "Evaluating a machine learning model.," 07-21 2017. [Online]. Available: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/.
[27] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013: PMLR, pp. 1310-1318.
[28] D. Saad, "Online algorithms and stochastic approximations," *Online Learning,* vol. 5, pp. 6-3, 1998.
[29] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation,* vol. 9, pp. 1735-80, 12/01 1997, doi: 10.1162/neco.1997.9.8.1735.
[30] M. Alom *et al.*, "A state-of-the-art survey on deep learning theory and architectures. Electronics," ed: Multidisciplinary Digital Publishing Institute, 2019.
[31] M. Phi. "Illustrated Guide to LSTM's and GRU's: A step by step explanation." TDataScience. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (accessed 7 Aug, 2021).
[32] R. A. Horn, "The hadamard product," in *Proc. Symp. Appl. Math*, 1990, vol. 40, pp. 87-169.
[33] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation,* vol. 12, no. 10, pp. 2451-2471, 2000.
[34] R. Shinde. "Understanding sequential/TimeSeries data for LSTM…." Medium. https://medium.com/@raman.shinde15/understanding-sequential-timeseries-data-for-lstm-4da78021ecd7 (accessed 9 Aug, 2021).
[35] S. Chatterjee. "ARIMA/SARIMA vs LSTM with Ensemble learning Insights for Time Series Data." https://www.datasciencecentral.com/profiles/blogs/arima-sarima-vs-lstm-with-ensemble-learning-insights-for-time-ser (accessed 8 Aug, 2021).
[36] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing,* vol. 50, pp. 159-175, 2003.
[37] T. Preis, H. S. Moat, and H. E. Stanley, "Quantifying trading behavior in financial markets using Google Trends," *Scientific reports,* vol. 3, no. 1, pp. 1-6, 2013.
[38] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo, "Text mining for market prediction: A systematic review," *Expert Systems with Applications,* vol. 41, no. 16, pp. 7653-7670, 2014.

# Appendices

## A. ARIMA evaluated on a single stock

```
Performing stepwise search to minimize aic
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=6942.283, Time=0.06 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=6792.194, Time=0.14 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=1.38 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=6940.284, Time=0.06 sec
 ARIMA(2,2,0)(0,0,0)[0] intercept   : AIC=6207.580, Time=0.29 sec
 ARIMA(3,2,0)(0,0,0)[0] intercept   : AIC=6127.899, Time=0.48 sec
 ARIMA(3,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.79 sec
 ARIMA(2,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.12 sec
 ARIMA(3,2,0)(0,0,0)[0]             : AIC=6125.901, Time=0.18 sec
 ARIMA(2,2,0)(0,0,0)[0]             : AIC=6205.582, Time=0.13 sec
 ARIMA(3,2,1)(0,0,0)[0]             : AIC=inf, Time=2.17 sec
 ARIMA(2,2,1)(0,0,0)[0]             : AIC=inf, Time=1.10 sec

Best model:  ARIMA(3,2,0)(0,0,0)[0]
Total fit time: 10.922 seconds
```




```
MAE=1.3182, MSE=4.2378, RMSE=2.0586, MAPE=0.01830%, execTime=61.12s
```

## B. ARIMA evaluated on all stocks

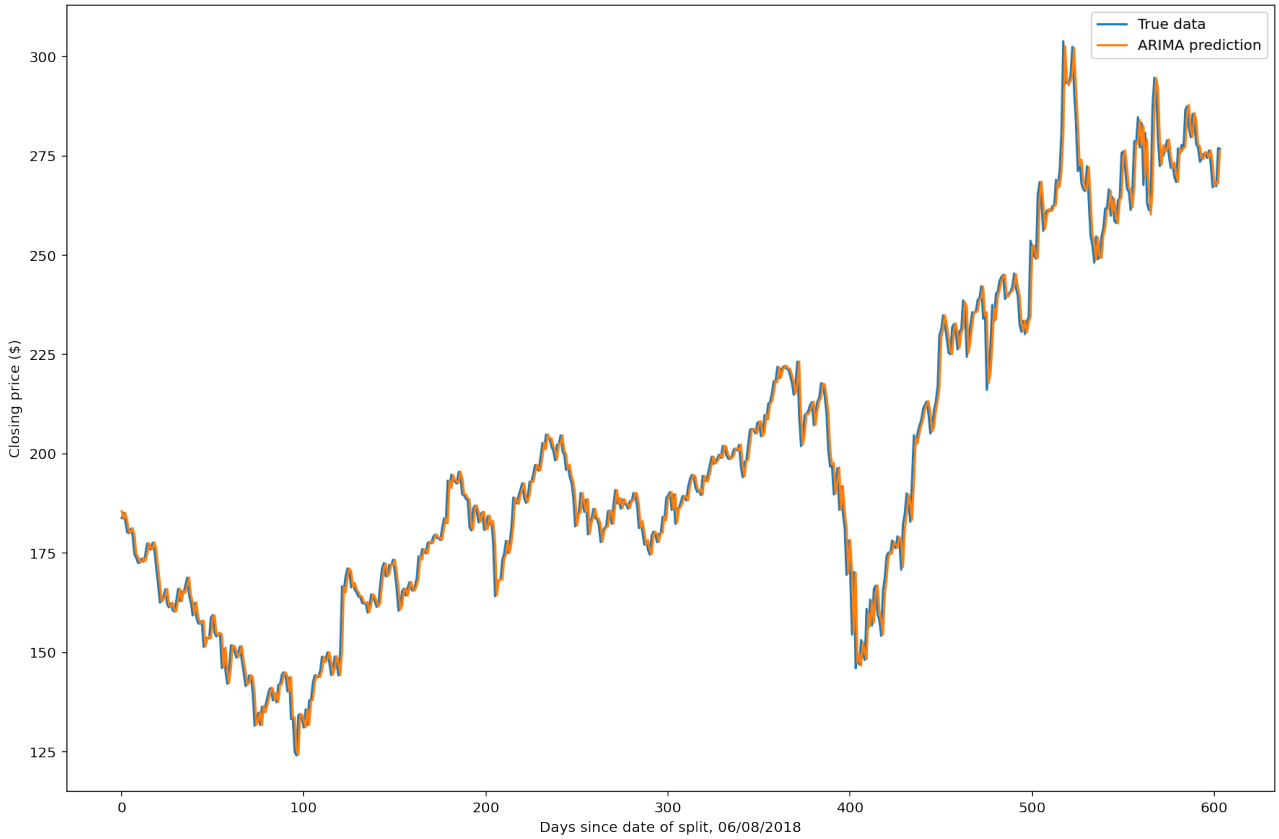
```
    ARIMA performance for the FAAMG group:
    Averages for MAE=11.47140, MSE=604.56885, RMSE=16.76263, MAPE=0.01548% and
execTime=65.678s
```

---

```
FB
```

---

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=9264.942, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=8732.518, Time=0.13 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=1.37 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=9262.942, Time=0.05 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=8616.192, Time=0.27 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=8474.003, Time=0.37 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.73 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.20 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=8472.005, Time=0.16 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=8614.194, Time=0.12 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=inf, Time=1.61 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=inf, Time=1.09 sec

Best model:  ARIMA(3,1,0)(0,0,0)[0]
Total fit time: 10.183 seconds
```




```
MAE=3.2168, MSE=22.0032, RMSE=4.6908, MAPE=0.01634%, execTime=42.35s
```

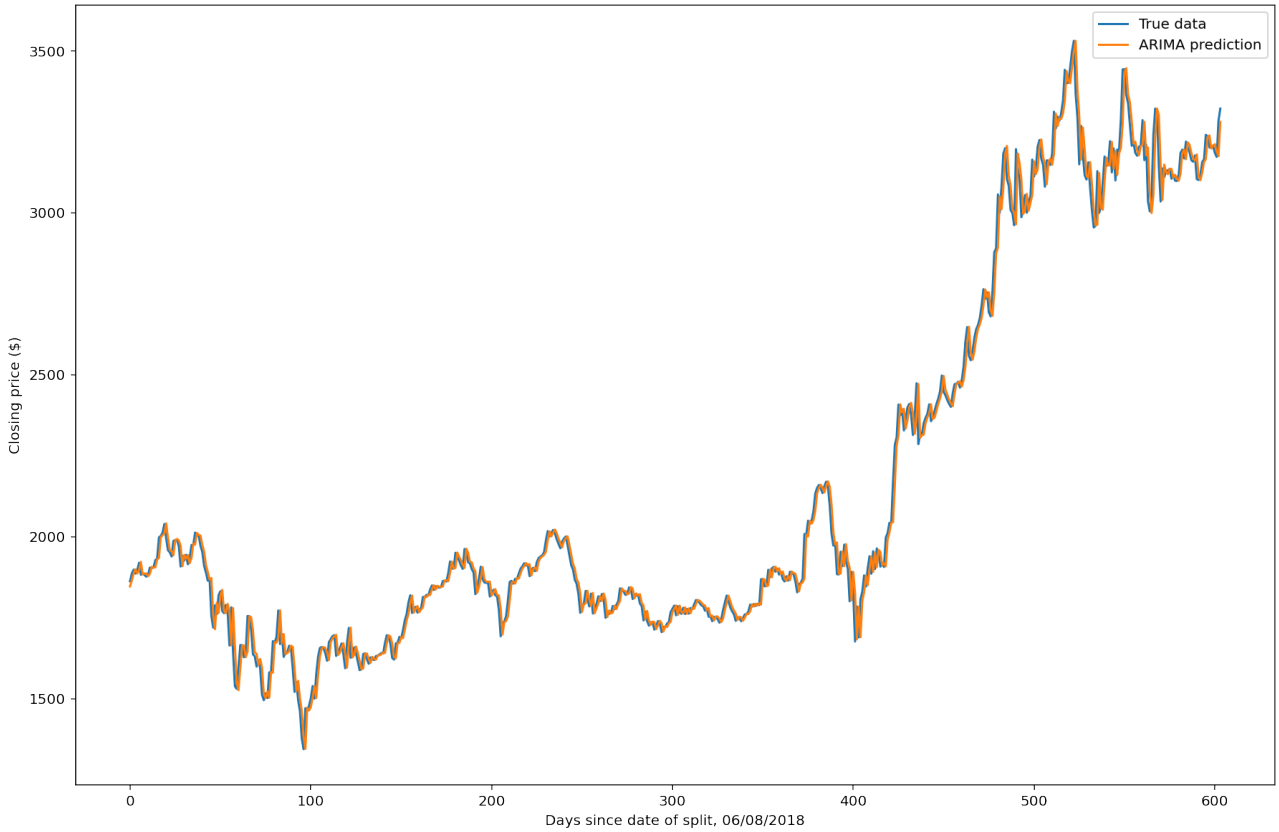
---

```
AMZN
```

---

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=15937.322, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=15415.424, Time=0.15 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=1.25 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=15935.322, Time=0.04 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=15298.881, Time=0.19 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=15166.239, Time=0.27 sec
```

```
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=1.24 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.18 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=15164.246, Time=0.11 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=15296.888, Time=0.08 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=inf, Time=0.65 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=inf, Time=1.01 sec

Best model:  ARIMA(3,1,0)(0,0,0)[0]
Total fit time: 7.232 seconds
```




```
MAE=33.3348, MSE=2339.8125, RMSE=48.3716, MAPE=0.01537%, execTime=63.27s
```

---

```
AAPL
```

---

```
Performing stepwise search to minimize aic
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=6942.283, Time=0.06 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=6792.194, Time=0.15 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=1.35 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=6940.284, Time=0.05 sec
 ARIMA(2,2,0)(0,0,0)[0] intercept   : AIC=6207.580, Time=0.29 sec
 ARIMA(3,2,0)(0,0,0)[0] intercept   : AIC=6127.899, Time=0.47 sec
 ARIMA(3,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.72 sec
 ARIMA(2,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.17 sec
 ARIMA(3,2,0)(0,0,0)[0]             : AIC=6125.901, Time=0.19 sec
 ARIMA(2,2,0)(0,0,0)[0]             : AIC=6205.582, Time=0.13 sec
 ARIMA(3,2,1)(0,0,0)[0]             : AIC=inf, Time=2.23 sec
 ARIMA(2,2,1)(0,0,0)[0]             : AIC=inf, Time=1.16 sec

Best model:  ARIMA(3,2,0)(0,0,0)[0]
Total fit time: 10.992 seconds
```

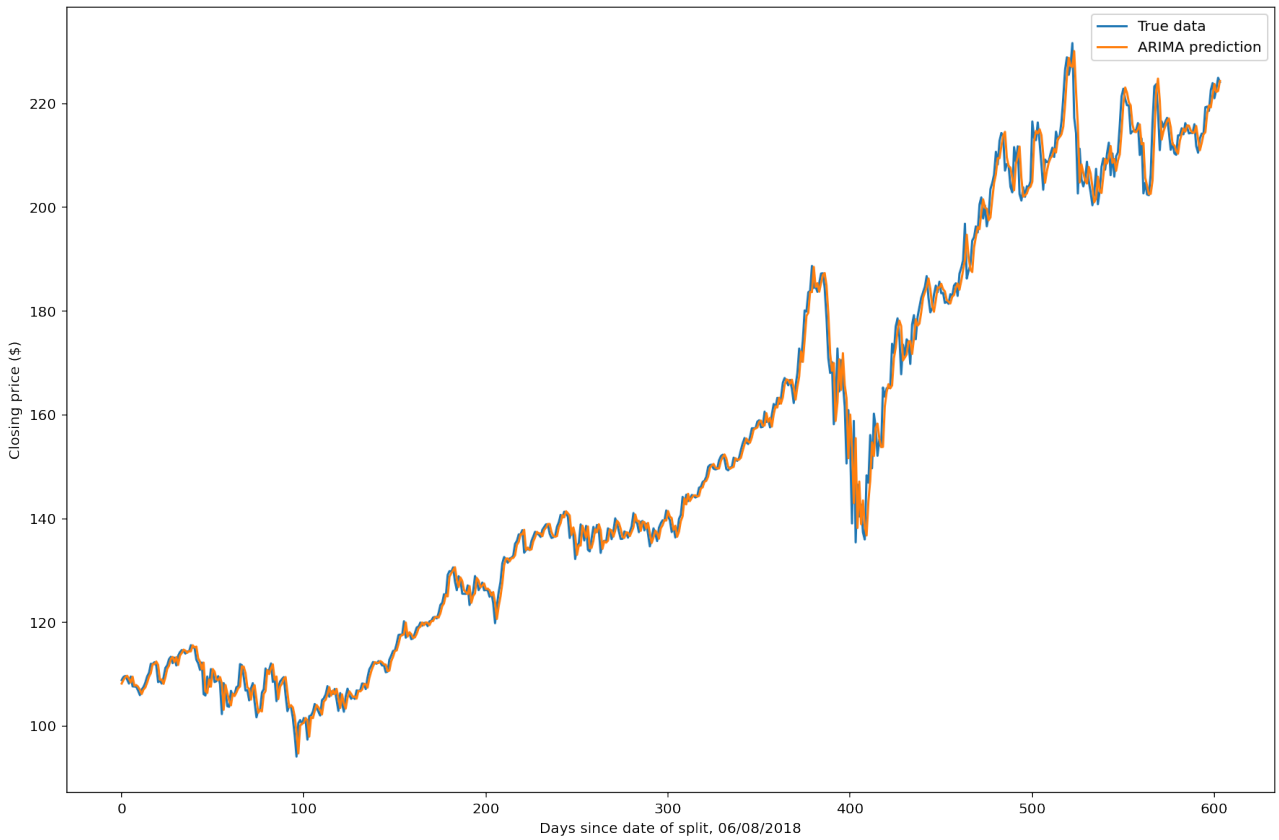

```
MAE=1.3182, MSE=4.2378, RMSE=2.0586, MAPE=0.01830%, execTime=117.61s
```

---

```
MSFT
```

---

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=8145.299, Time=0.07 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=7341.078, Time=0.13 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=1.37 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=8143.299, Time=0.05 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=7046.900, Time=0.24 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=6970.105, Time=0.31 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.48 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.22 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=6968.106, Time=0.15 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=7044.901, Time=0.14 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=inf, Time=0.89 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=inf, Time=0.67 sec

Best model:  ARIMA(3,1,0)(0,0,0)[0]
Total fit time: 8.719 seconds
```




```
MAE=2.1468, MSE=10.7212, RMSE=3.2743, MAPE=0.01386%, execTime=44.09s
```

---

GOOG

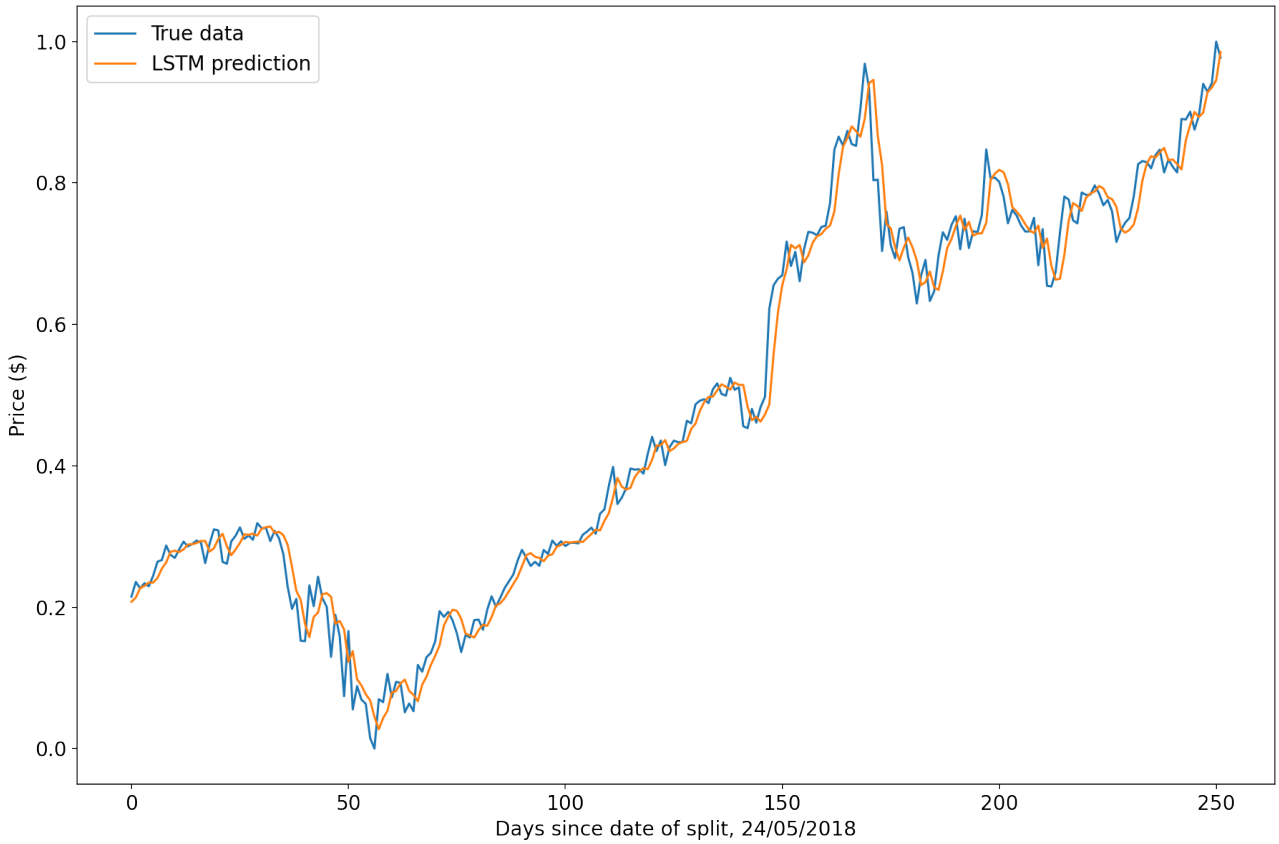
---

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=14292.510, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=13716.148, Time=0.13 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=1.34 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=14290.510, Time=0.04 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=13504.110, Time=0.42 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=13419.040, Time=0.54 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.81 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=2.05 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=13417.042, Time=0.23 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=13502.111, Time=0.18 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=inf, Time=1.46 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=inf, Time=1.41 sec

Best model:  ARIMA(3,1,0)(0,0,0)[0]
Total fit time: 10.682 seconds
/usr/local/lib/python3.7/dist-packages/statsmodels/base/model.py:568:
ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals
```


MAE=17.3403, MSE=646.0697, RMSE=25.4179, MAPE=0.01354%, execTime=61.08s

## C. LSTM evaluated on a single stock, CPU runtime

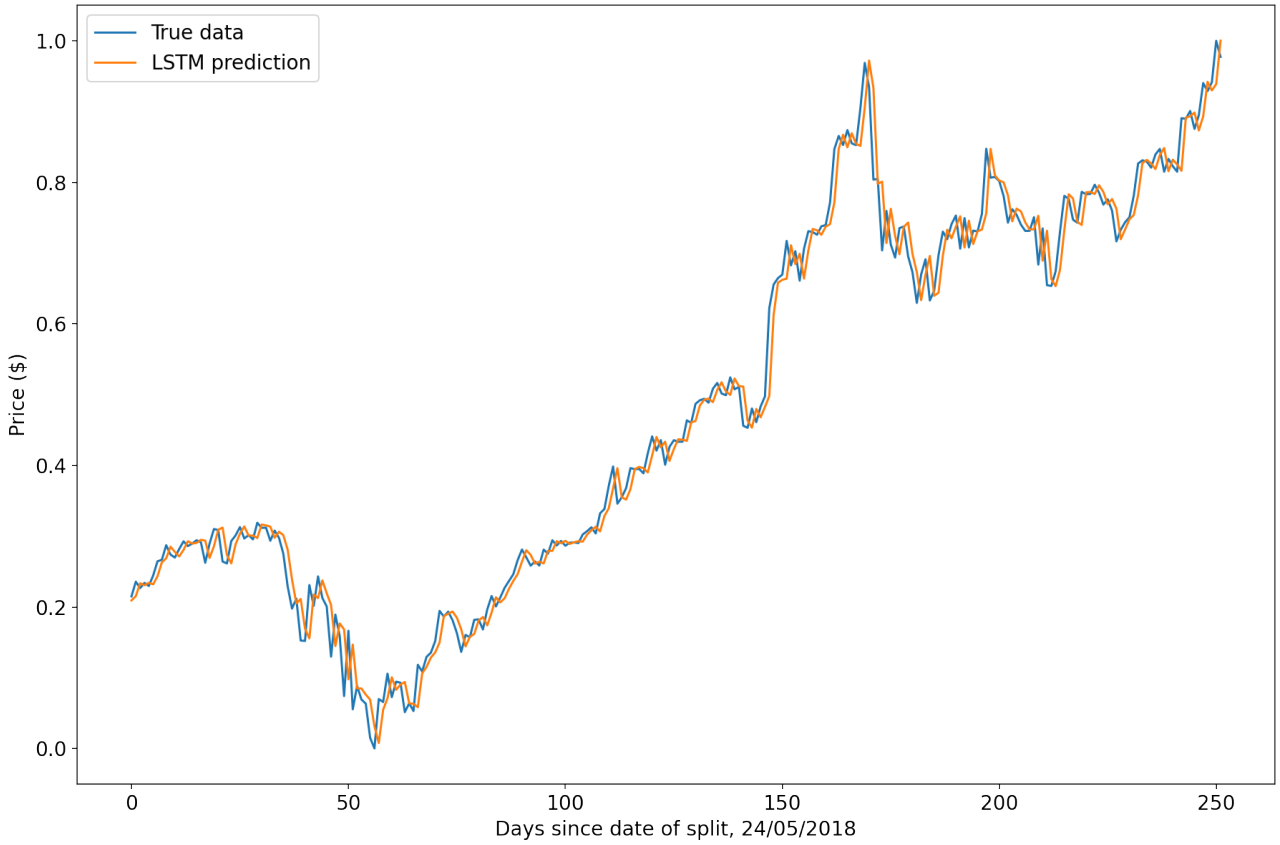
```
    Trial 3 Complete [00h 01m 24s]
    val_loss: 0.0006167953833937645

    Best val_loss So Far: 0.0006167953833937645
    Total elapsed time: 00h 05m 54s
    INFO:tensorflow:Oracle triggered exit
    WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the
criteria. It will use a generic GPU kernel as fallback when running on GPU.
```




```
    Important parameters:
    Stock analysed: AAPL
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%

    Metrics:
    MAE=1.91132
    MSE=7.34491
    RMSE=2.71015
    MAPE=2.08542%
    execTime=415.24s
```

## D. LSTM GPU evaluated on the same stock

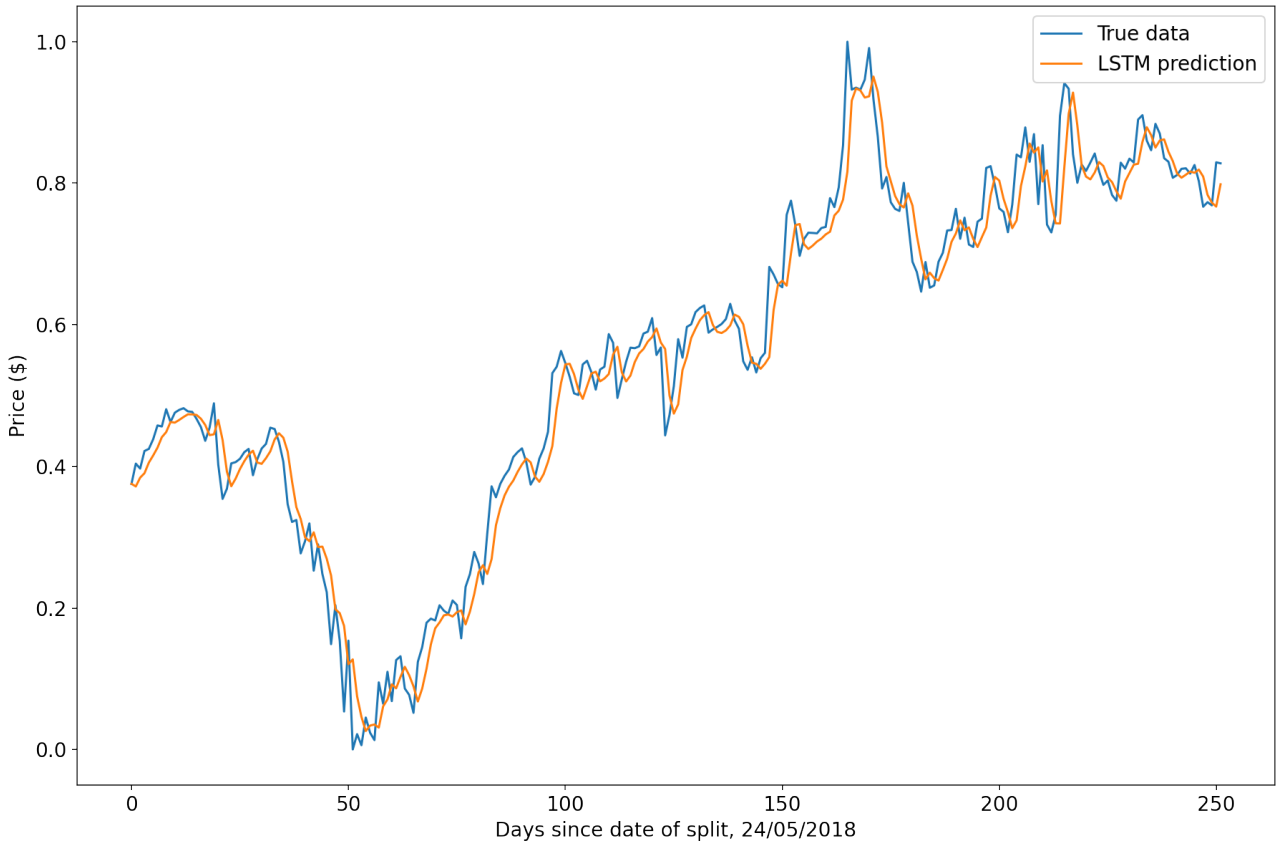
```
Trial 3 Complete [00h 00m 11s]
val_loss: 0.00047602152335457504

Best val_loss So Far: 0.00047602152335457504
Total elapsed time: 00h 00m 33s
INFO:tensorflow:Oracle triggered exit
```




```
Important parameters:
Stock analysed: AAPL
Batch size = 50
Epochs = 50
Neurons = 64
Training split 60.0%
Validation split 20.0%
Testing split 20.0%

Metrics:
MAE=1.81755
MSE=6.46691
RMSE=2.54301
MAPE=1.97878%
execTime=35.14s
```

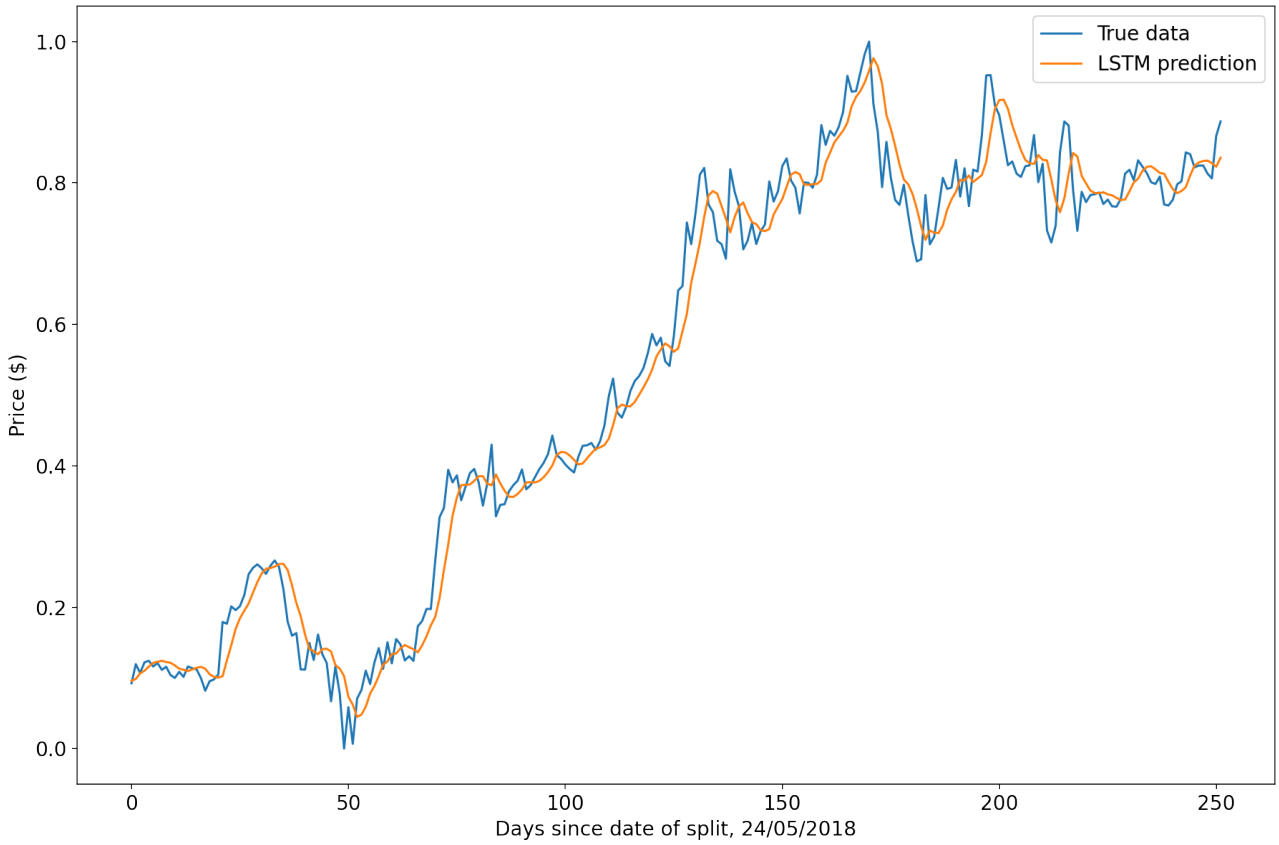
## E. LSTM evaluated on all stocks (round 1)

```
    LSTM performance for the FAAMG group:
    Averages for MAE=81.29282, MSE=27745.10372, RMSE=91.68440, MAPE=6.13263% and
execTime=53.786s

    ----------------------------------------------------------------------

    FB

    ----------------------------------------------------------------------

    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```

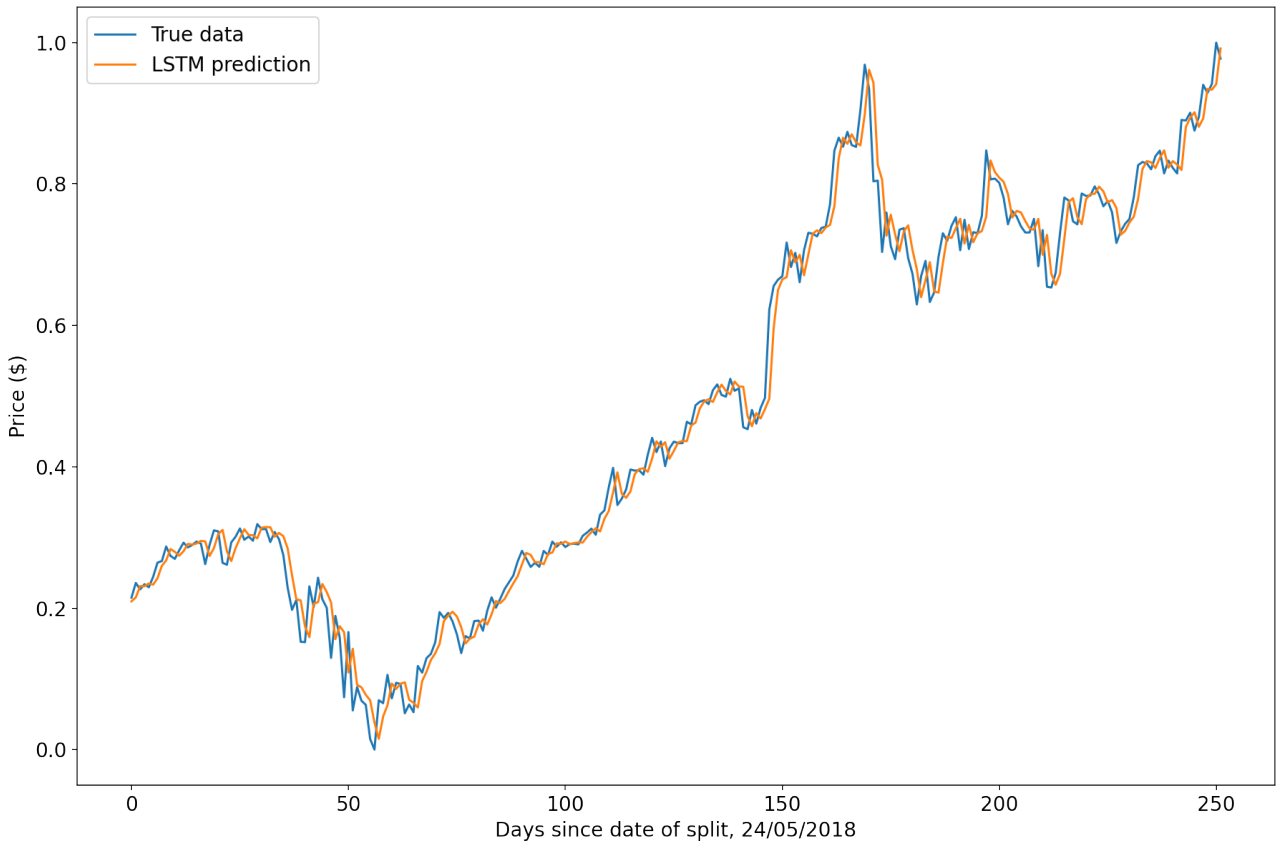



```
    Important parameters:
    Stock analysed: FB
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%

    ----------------------------------------------------------------------

    AMZN

    ----------------------------------------------------------------------

    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
```

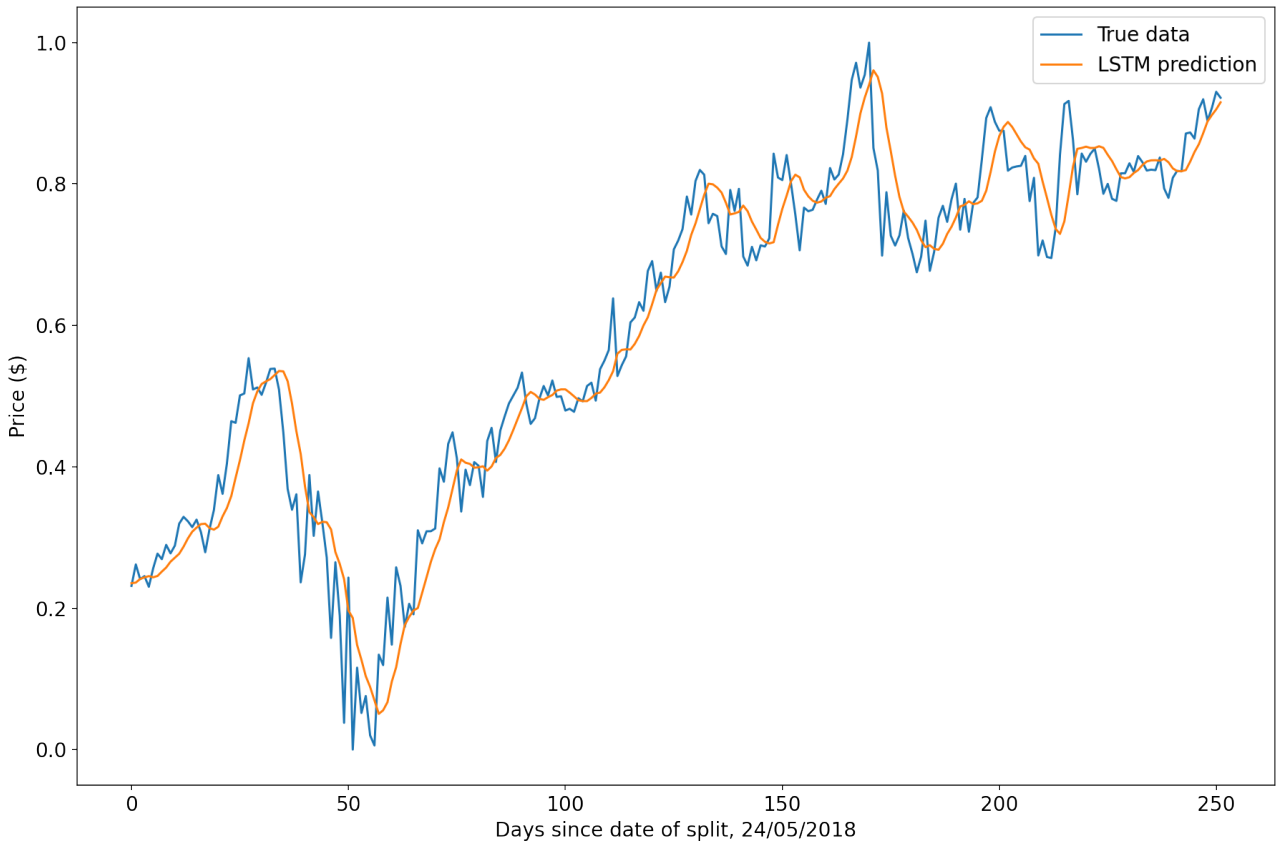
```
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```



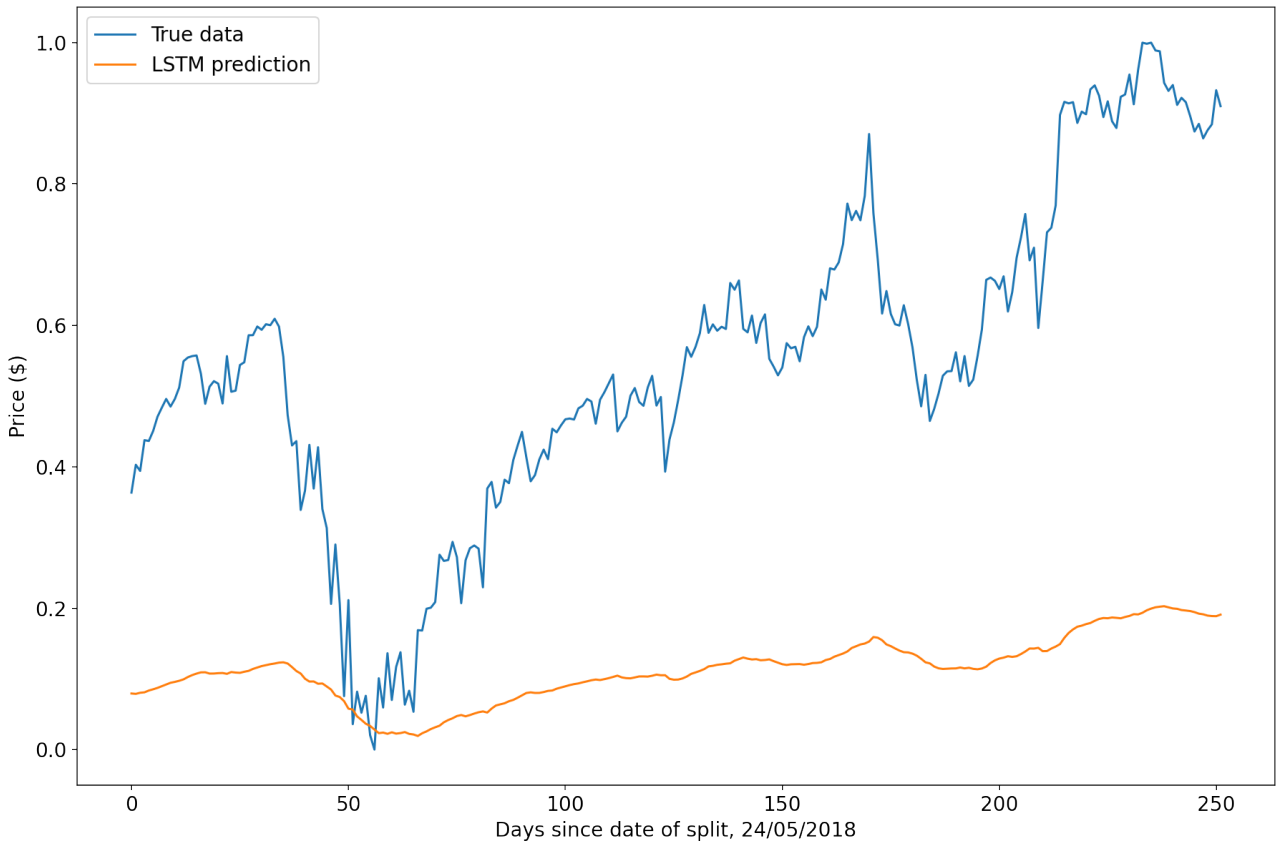

```
    Important parameters:
    Stock analysed: AMZN
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------------
    AAPL
    ----------------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```




```
    Important parameters:
    Stock analysed: AAPL
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------
    MSFT
    ----------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```



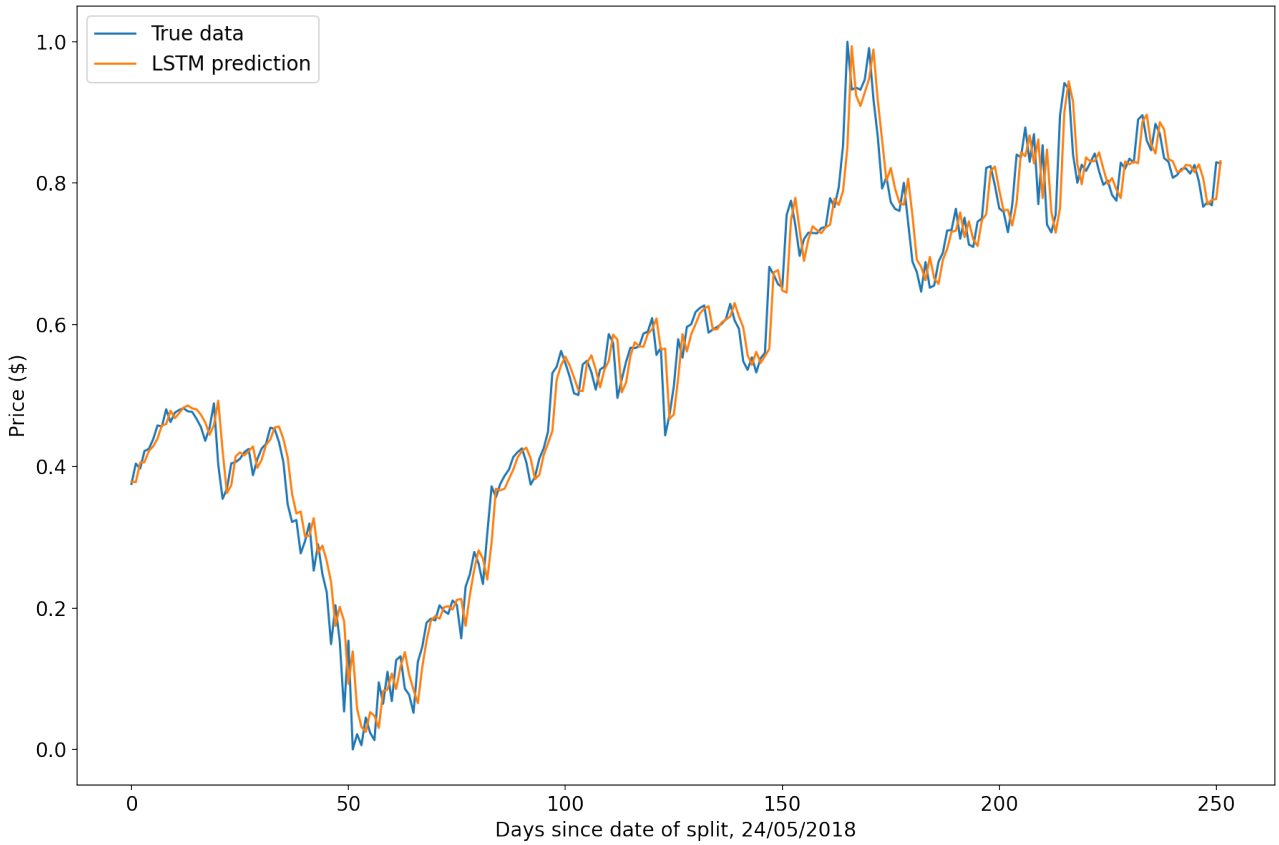

```
Important parameters:
Stock analysed: MSFT
Batch size = 50
Epochs = 50
Neurons = 64
Training split 60.0%
Validation split 20.0%
Testing split 20.0%
--------------------------------------------------------------------------------
GOOG
--------------------------------------------------------------------------------
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
INFO:tensorflow:Oracle triggered exit
```




```
Important parameters:
Stock analysed: GOOG
Batch size = 50
Epochs = 50
Neurons = 64
Training split 60.0%
Validation split 20.0%
Testing split 20.0%
```

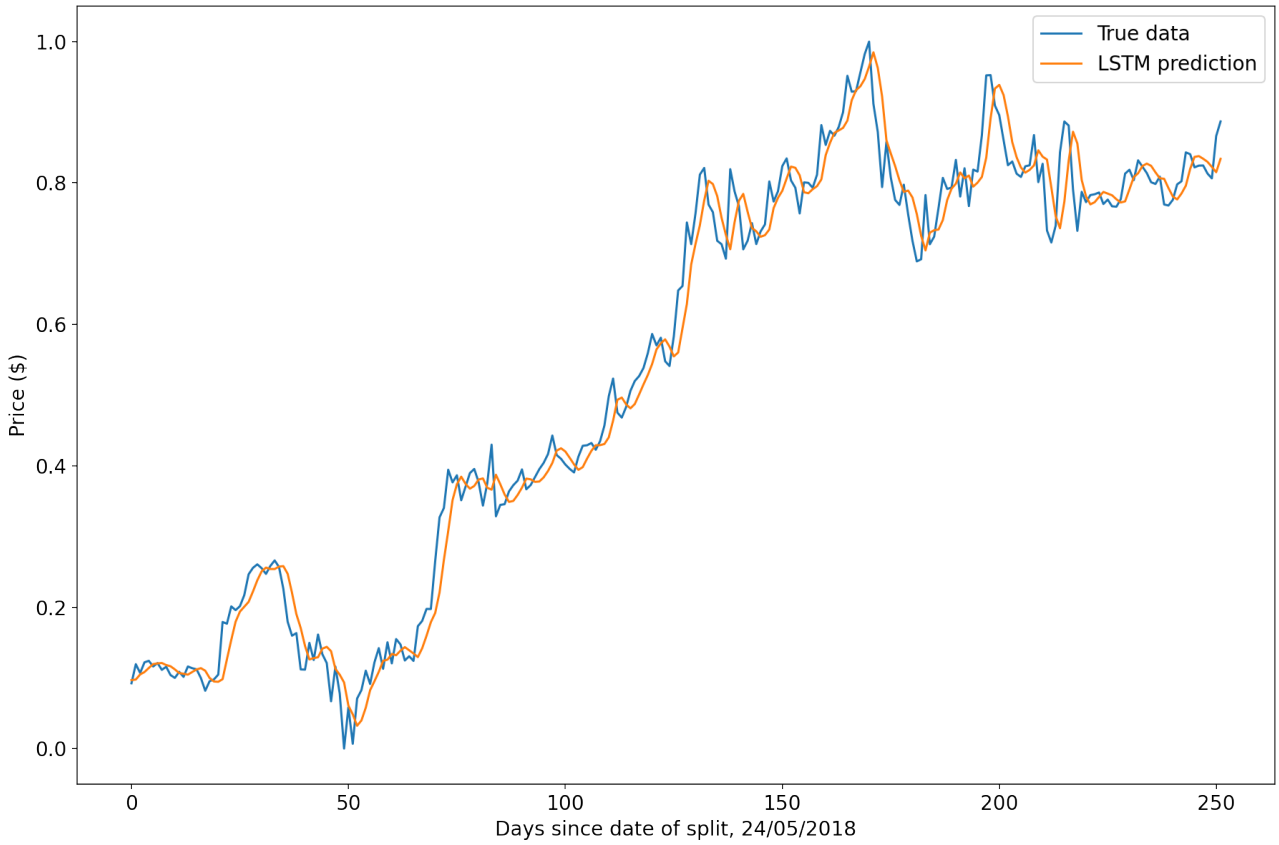
## F. LSTM evaluated on all stocks (round 2)

```
    LSTM performance for the FAAMG group:
    Averages for MAE=17.88987, MSE=1331.88777, RMSE=24.07842, MAPE=1.96128% and
execTime=53.786s

    ----------------------------------------------------------------------

    FB

    ----------------------------------------------------------------------

    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```

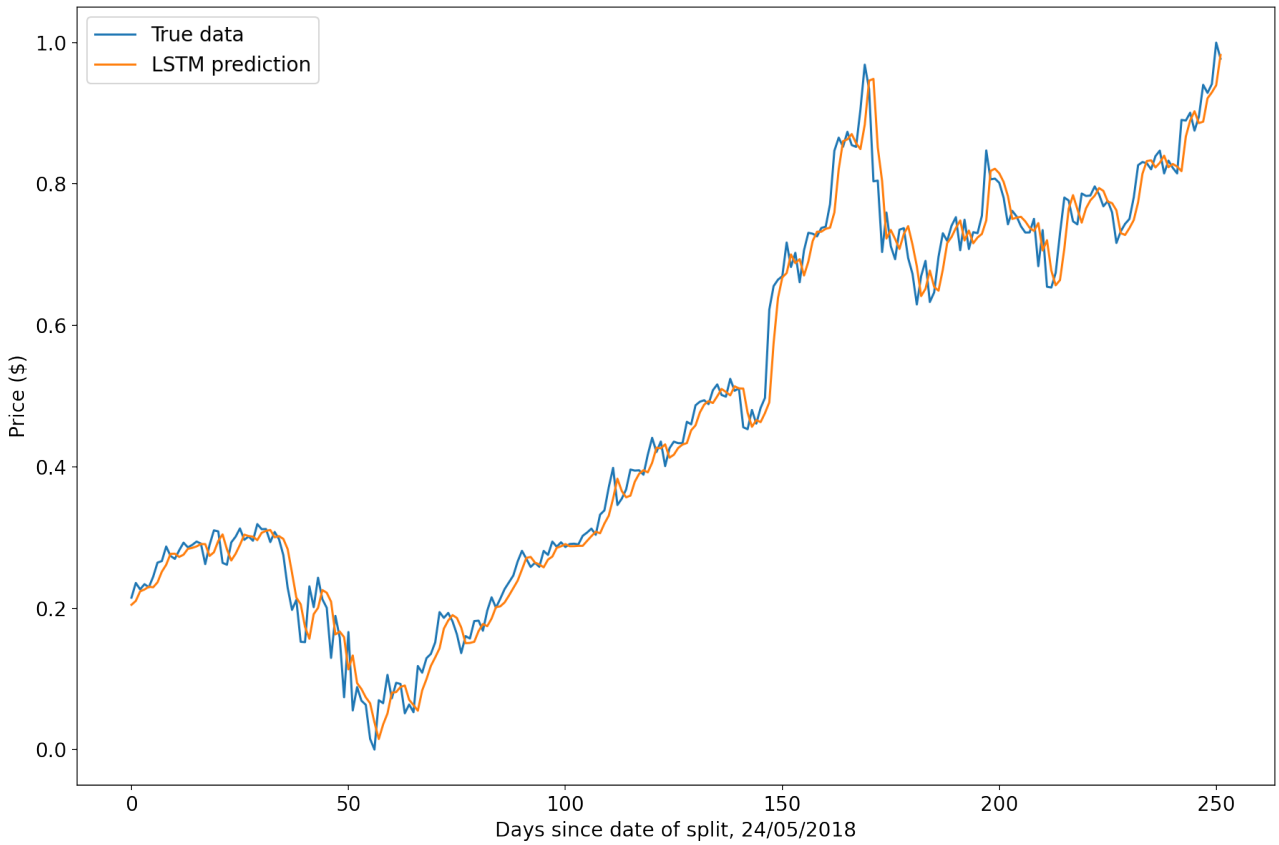



```
    Important parameters:
    Stock analysed: FB
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%

    ----------------------------------------------------------------------

    AMZN

    ----------------------------------------------------------------------

    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
```

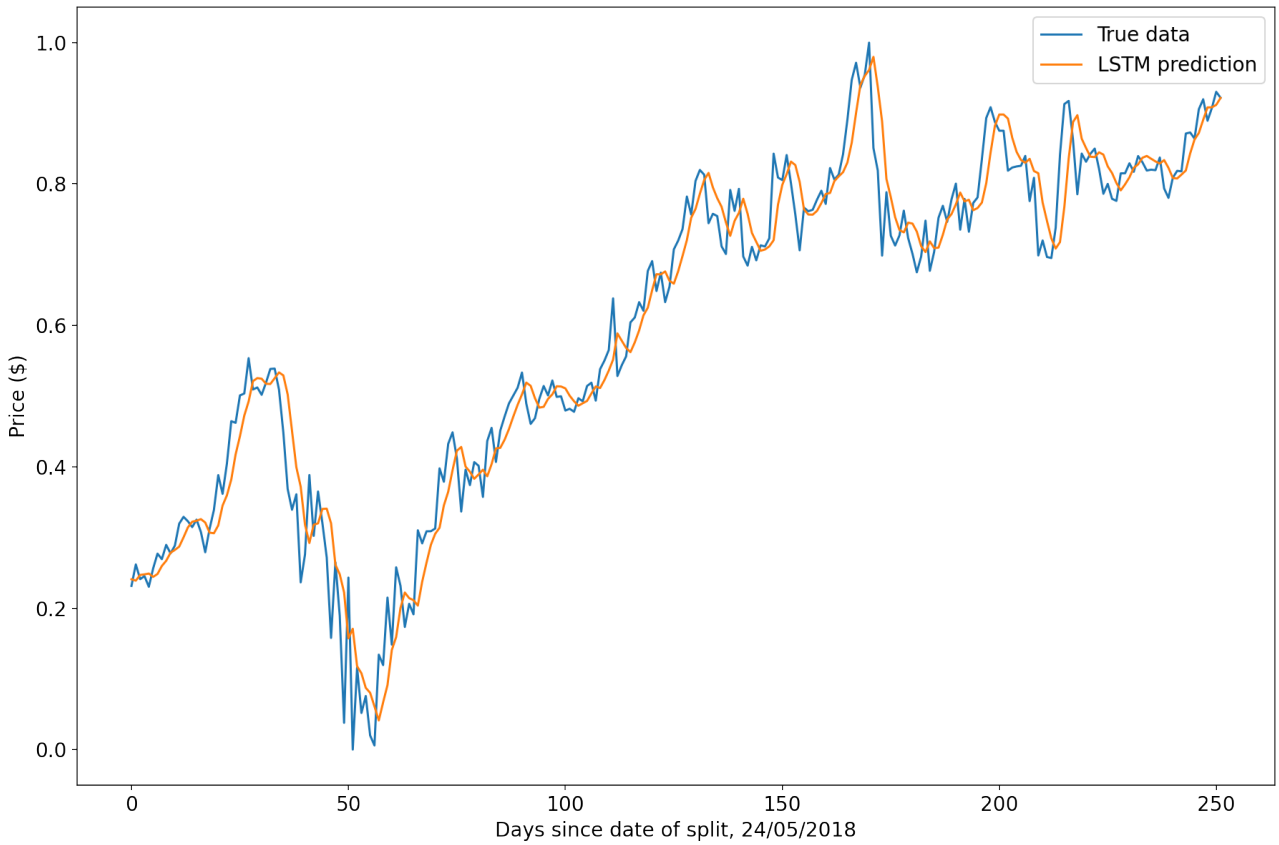
```
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```



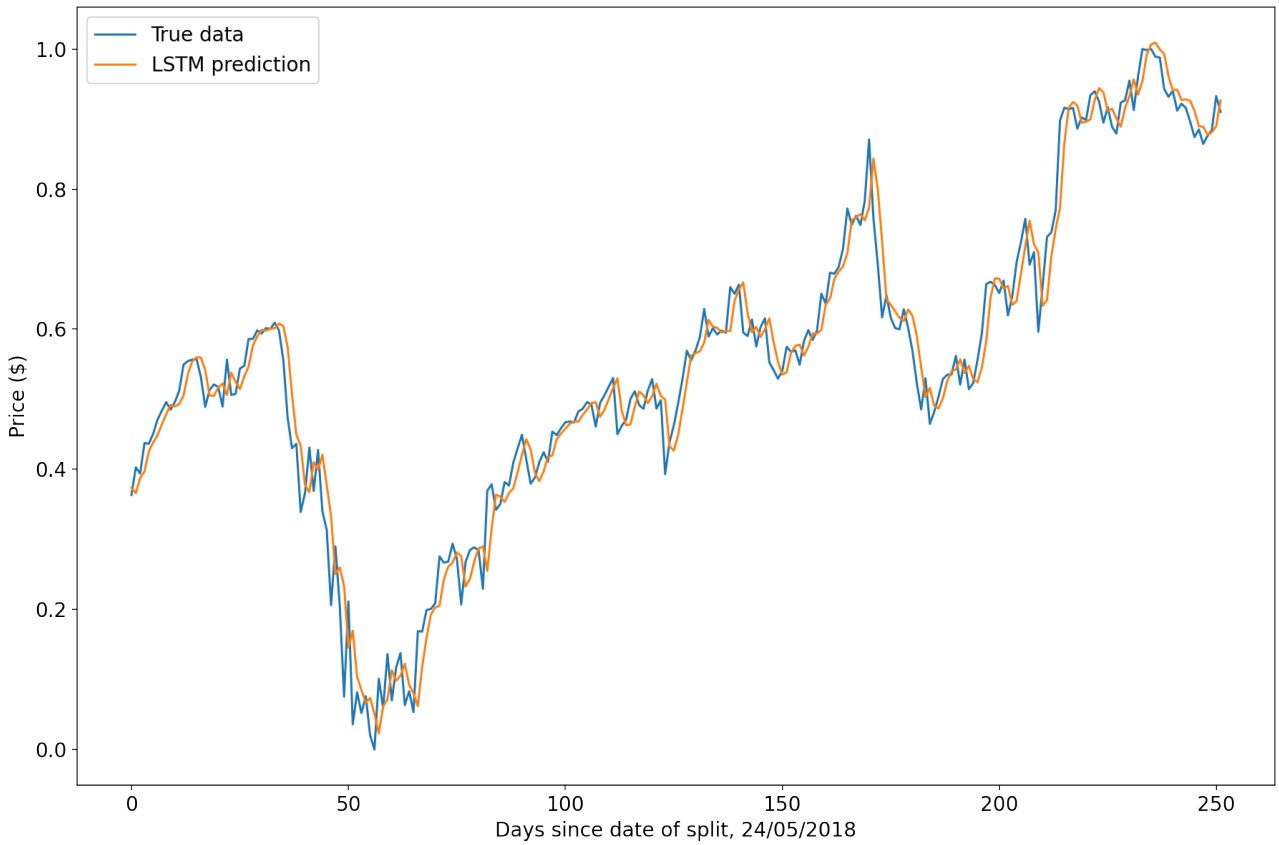

```
    Important parameters:
    Stock analysed: AMZN
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------------
    AAPL
    ----------------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```




```
    Important parameters:
    Stock analysed: AAPL
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------------
    MSFT
    ----------------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```



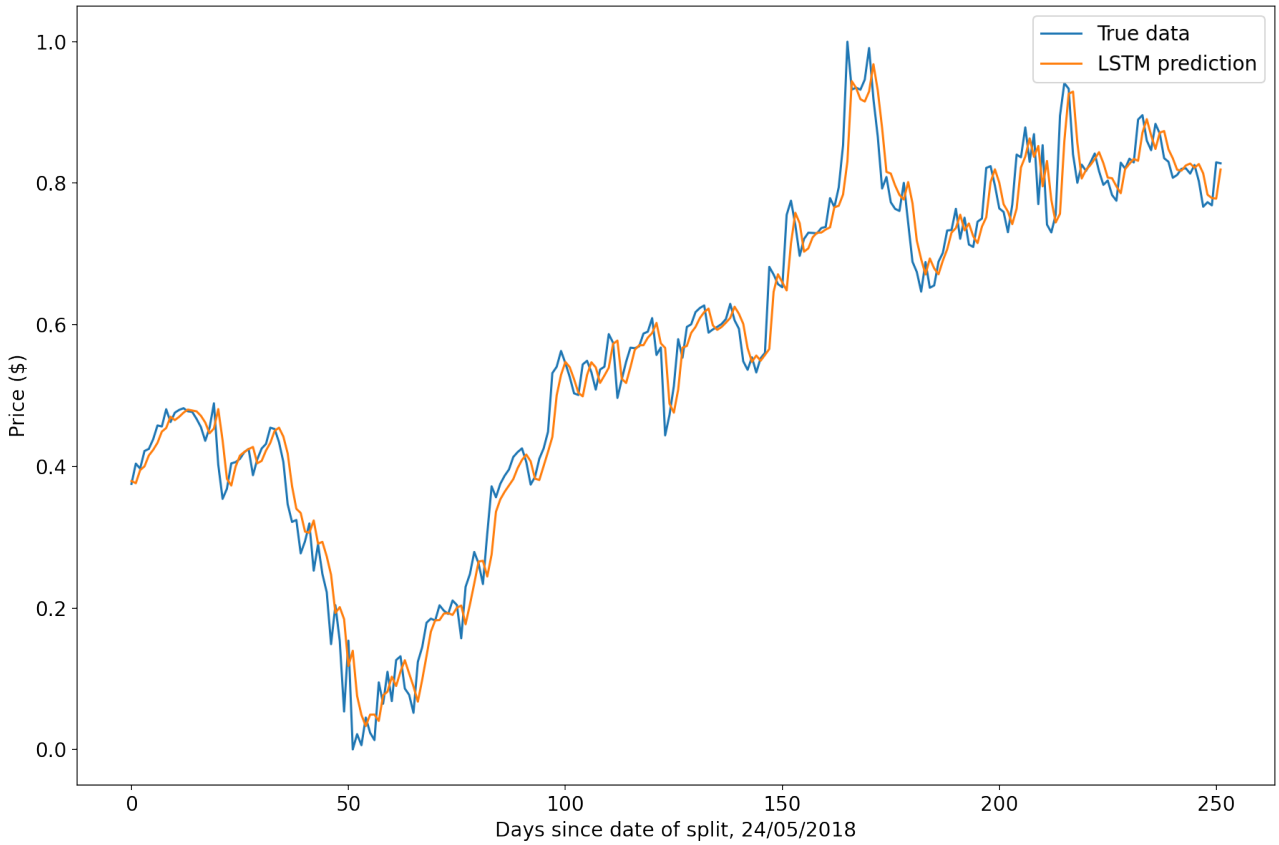

```
    Important parameters:
    Stock analysed: MSFT
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------
    GOOG
    ----------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```




```
Important parameters:
Stock analysed: GOOG
Batch size = 50
Epochs = 50
Neurons = 64
Training split 60.0%
Validation split 20.0%
Testing split 20.0%
```

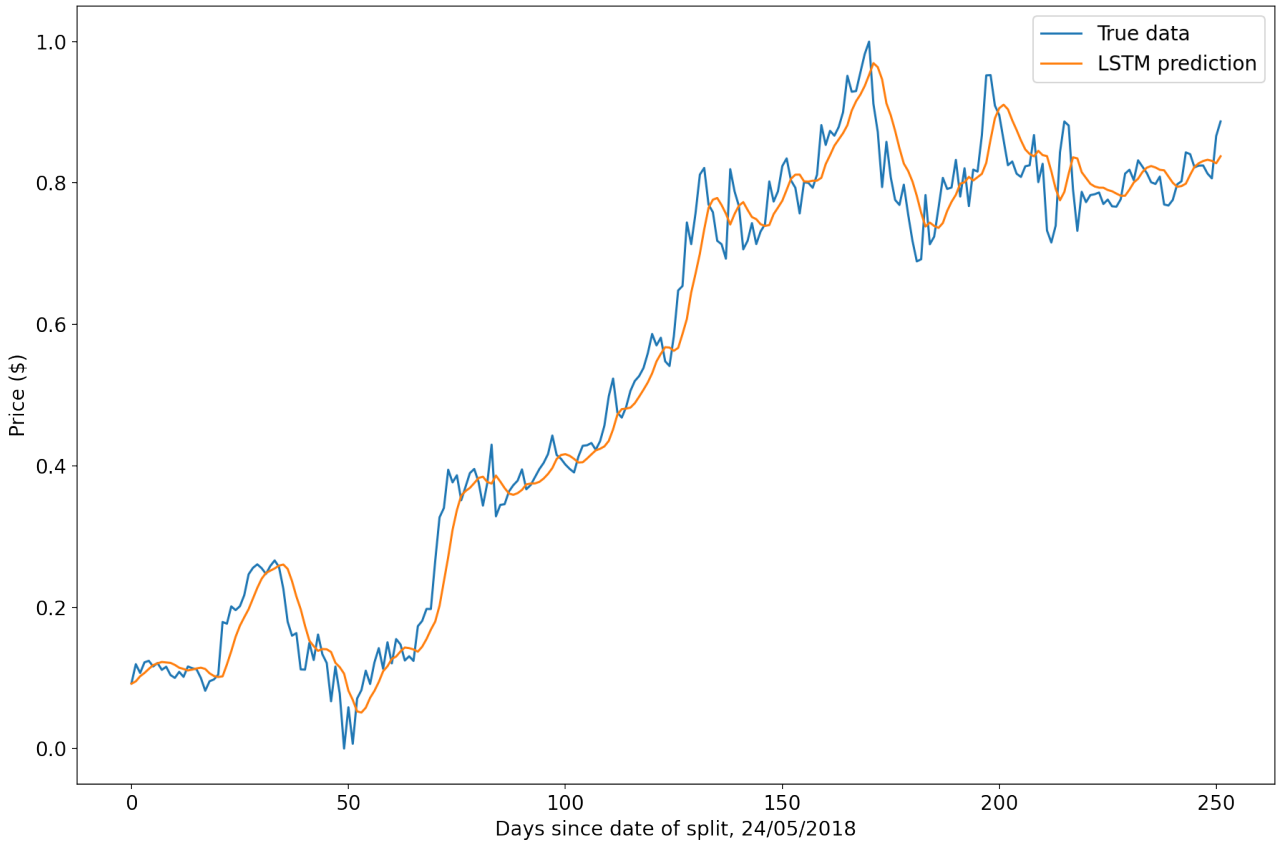
## G. LSTM evaluated on all stocks (round 3)

```
    LSTM performance for the FAAMG group:
    Averages for MAE=91.12595, MSE=35655.89410, RMSE=102.54812, MAPE=6.65438% and
execTime=53.786s

    ----------------------------------------------------------------------------
    FB
    ----------------------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```

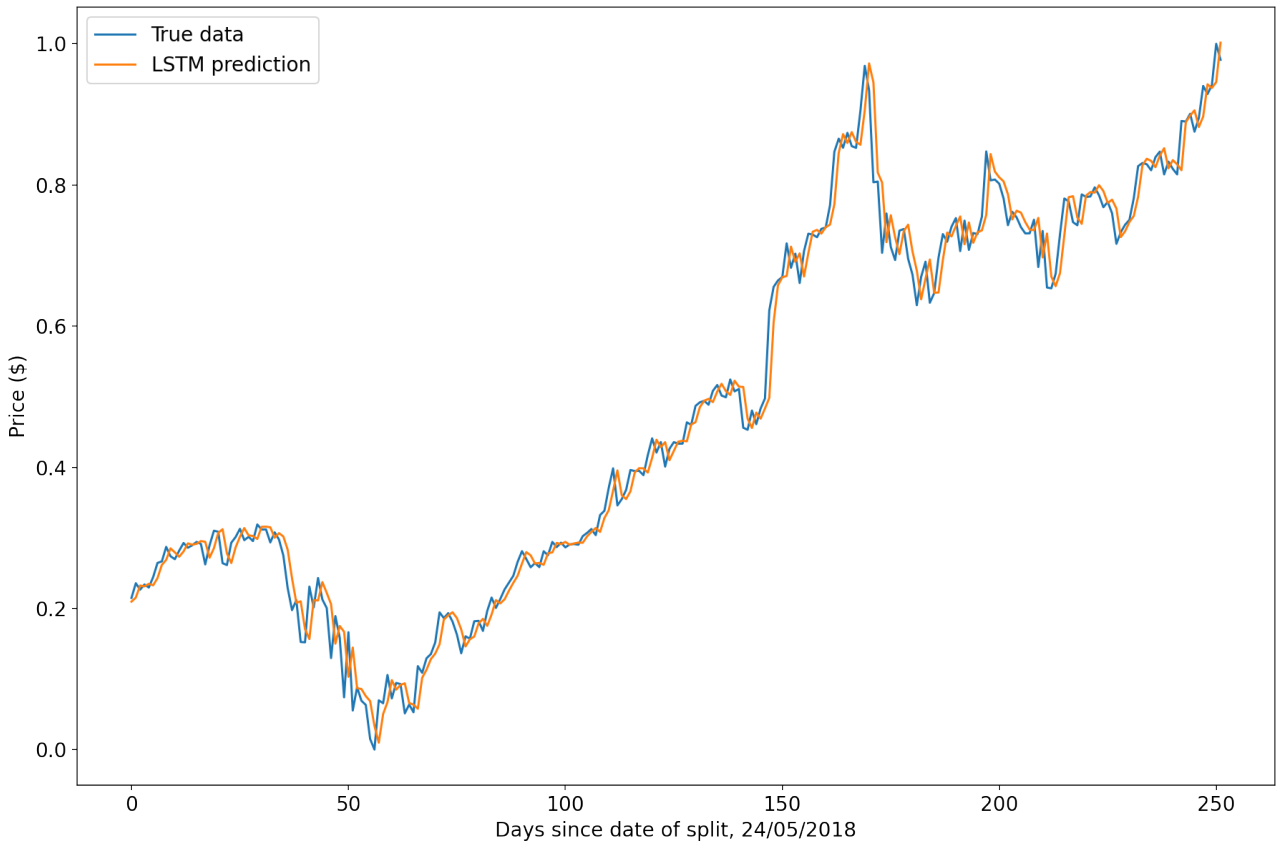



```
    Important parameters:
    Stock analysed: FB
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------------------
    AMZN
    ----------------------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
```

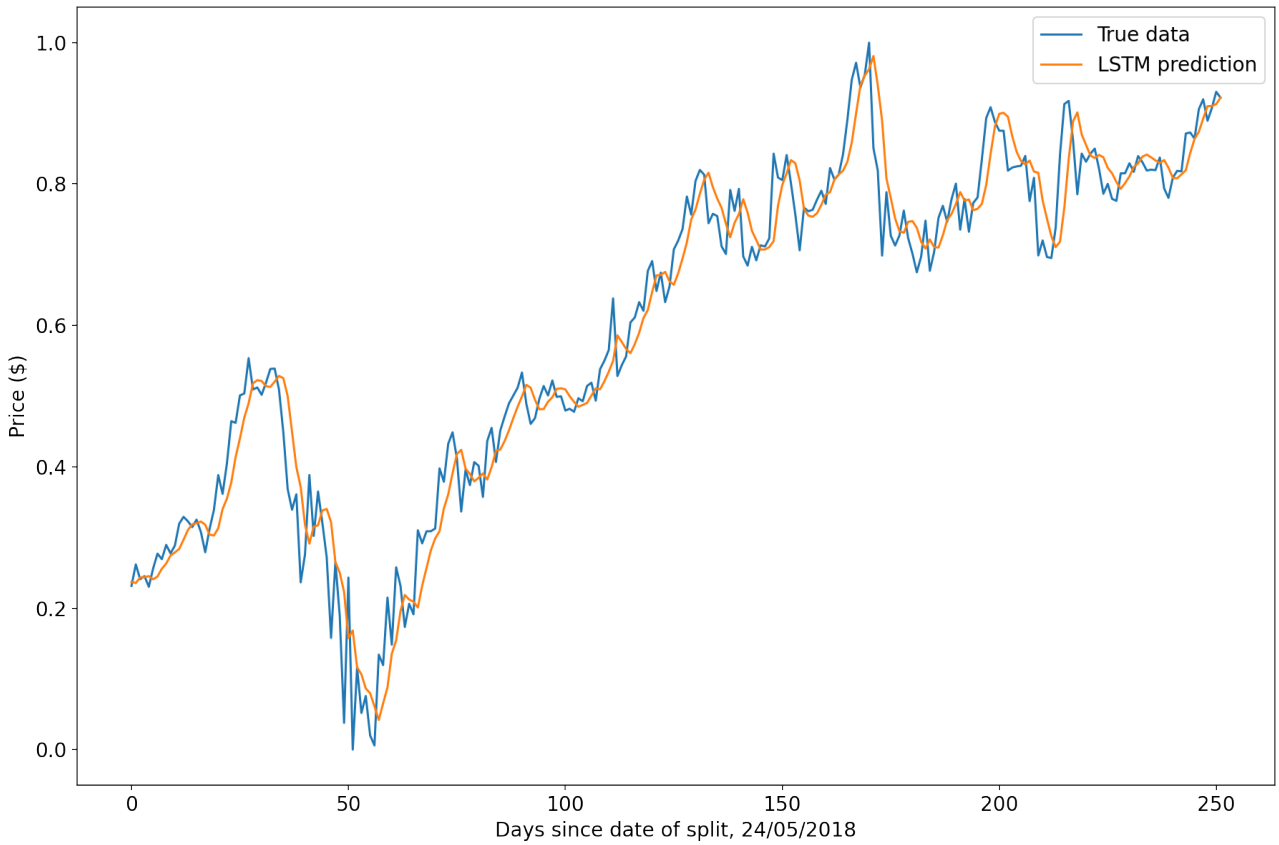
```
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```



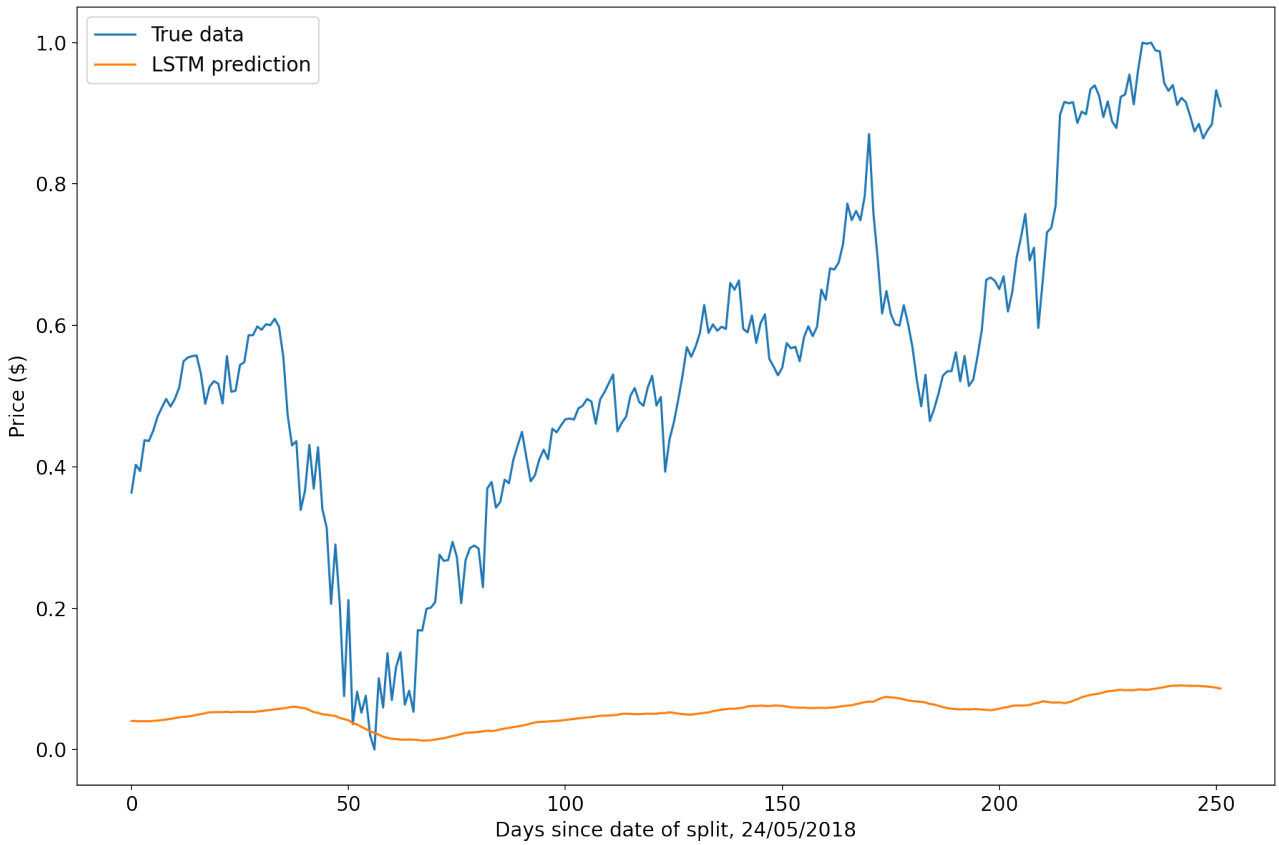

```
    Important parameters:
    Stock analysed: AMZN
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------------
    AAPL
    ----------------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```




```
    Important parameters:
    Stock analysed: AAPL
    Batch size = 50
    Epochs = 50
    Neurons = 64
    Training split 60.0%
    Validation split 20.0%
    Testing split 20.0%
    ----------------------------------------------------------------------
    MSFT
    ----------------------------------------------------------------------
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
    WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
    WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
    INFO:tensorflow:Oracle triggered exit
```




```
Important parameters:
Stock analysed: MSFT
Batch size = 50
Epochs = 50
Neurons = 64
Training split 60.0%
Validation split 20.0%
Testing split 20.0%
--------------------------------------------------------------------------------
GOOG
--------------------------------------------------------------------------------
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or
tf.keras.Model.load_weights) but not all checkpointed values were used. See above for
specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(...).expect_partial(), to silence these warnings, or use
assert_consumed() to make the check explicit. See https://www.tensorflow.org/guide/
checkpoint#loading_mechanics for details.
INFO:tensorflow:Oracle triggered exit
```




```
Important parameters:
Stock analysed: GOOG
Batch size = 50
Epochs = 50
Neurons = 64
Training split 60.0%
Validation split 20.0%
Testing split 20.0%
```