MIT 6.864 Spring 2020

LSTMs for Financial Time Series Forecasting

Arlene Siswanto

May 8, 2020

Abstract

With the widespread availability of data enabled by the online transition of information, we can now generate informed models that provide us with insights and predictions about our data. Though statistical techniques for time series forecasting, including ARIMA and exponential smoothing, have existed for decades, recent advances in machine learning have also been shown to have potential in the forecasting space. In particular, *long short-term memory* or LSTM models have a natural temporal dependence in which long range knowledge about the past is taken into account in determining future outputs. In this paper, we implement LSTMs to model and mimic the movement of stock prices in the DJIA index over time and analyze the accuracy of these predictions. We compare the technique to existing statistical modeling techniques in an effort to determine forecasting abilities of the various techniques. Our work shows that LSTMs perform roughly within the same benchmark as the statistical techniques, though the results depend on the structure and quantity of data as well as the parameters used in tuning.

1 Introduction

Advancements in machine learning allow for improved classification, regression, and generation. A clear extension is to use these advancements to make predictions about values in future points in time. *Time series forecasting*, which is based on using models to predict future values based on previously observed values, has a wide range of practical applications which include predicting consumer pattern shifts, financial market movements, and air quality fluctuations. LSTMs, originally proposed in 1997 by Hochreiter [1] but popularized in recent years from use in Google Translate, Siri, and Amazon Alexa, are natural candidates for such sequence prediction. We implement an LSTM to predict future stock market data, and compare the technique with implemented baseline statistical techniques.

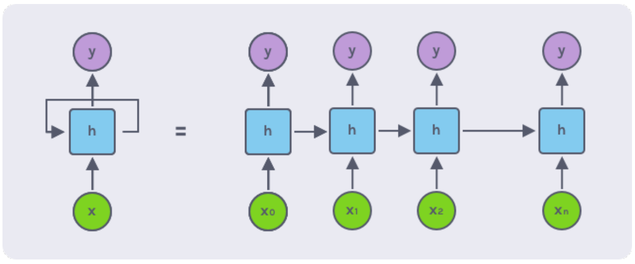

Figure 1: **Vanilla RNN.** An unrolled depiction of a recurrent neural network. LSTMs are variants of RNNs.

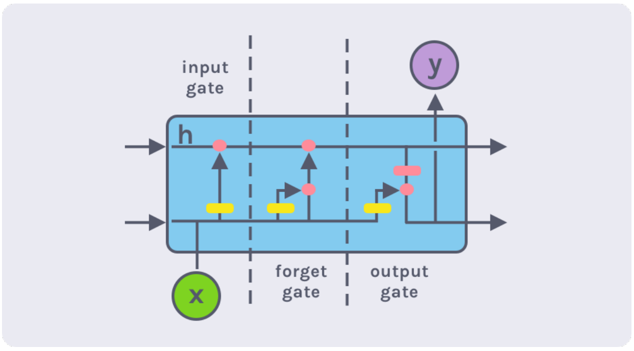

Figure 2: **LSTM cell.** Each LSTM cell contains input, forget, and output gates which control the flow of old and new information in the recurrent neural network.

1.1 LSTM

LSTM networks are a type of recurrent neural network with temporal dependencies. As we can see in Figure 1, an RNN takes in an input sequence in value-by-value and calculates the output by combining the input element with a hidden *latent* state, which is updated for use by the next element. The LSTM, a recent and more advanced update of the vanilla RNN, was developed to combat the vanishing gradient problem in which previous inputs would not be remembered after a large time lag.

The architecture of an LSTM cell can be found in Figure 2. The horizontal line spanning the top of the cell contains the cell state, which is passed in from the previous cell. The LSTM continuously regulates the contents by using gates. Generally, an LSTM cell contains an input gate, output gate, and forget gate through which information flows. This formation allows the cell to remember values across arbitrary time intervals, making the LSTM well-suited to tackle the challenge of analyzing and predicting time series data.

1.2 Related Work

Recent work in this financial domain has been completed, and most work is largely focused on the accuracy of the models on time series rather than their applications to trading strategies. Sezera et al. [8] compile the existing literature of machine learning techniques, including CNNs, DBNs, LSTMs on financial data. Liu et al. [3] discuss applying data denoising techniques prior to predicting using LSTMs. Qiu et al. [7] and Kim et al. [2] discuss using LSTMs with attention mechanism to predict stock market data. The previously mentioned studies were conducted in 2019. In 2018, Makridakis et al. [5] reported empirical results from open competitions that utilized machine learning techniques for time series analysis. It is clear that LSTMs are a new addition to the field of financial modeling, and that there is more work to be done in comparing results to simpler existing methods.

2 Dataset

We will be forecasting the historical daily values of the large cap companies listed on the Dow Jones Industrial Average or DJIA [11], a stock market index consisting of 30 large companies in the United States. Alongside other market indices such as the S&P 500 Index and NASDAQ composite, this index can reveal much about how the economy of a geographic region or particular sector is performing at a specific time. Trends in industry shifts tend to be represented by these indices. The dataset used, which was retrieved from Kaggle [10], contains the open, high, low, close, and volume of trades of stocks listed in the DJIA from 2006 to 2018. For optimal forecasting performance, we may only use some subset of this data as relevant to the forecasting problem.

3 Methodology

Applying the LSTM to historical stock data consists of several steps including data preprocessing, modeling, and result analysis. Our implementation is inspired by work done by Orac [6].

3.1 Preprocessing

First, we must convert the data to a usable format. The data ideally should be spaced in regular time intervals to be used by the LSTM for consistency. Since the data contains historical daily high prices, the data is already properly spaced. Plotting this, Figure 3 provides us with a high level understanding of what the series looks like: In general, there exists an upward trend, potential seasonality, random fluctuations, and changepoints that shift underlying patterns. Due to the property that LSTMs capture both long term and short term information and decides on its own which components are important, all of the historic data from 2006 - 2018 will be used in our training.

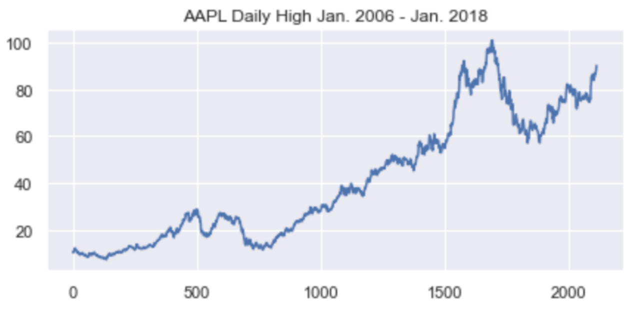

Figure 3: **Stock time series - AAPL.** Historical high daily values, 2006 - 2018, of DJIA stock AAPL. Varying time horizons of this data are used depending on forecasting technique.

The next step is to scale the data by using `StandardScaler` from `sklearn`, which removes the mean and scales to unit variance. From the literature in neural networks, non-standard data with inconsistent variance may make the estimator unable to learn as expected due to the effects of overwhelming magnitudes. Then, we mold our data into a typical many-to-many RNN input format: for each timestep i with a sequence length of n on sequence (s), our expected input is $s_i s_{i+1} \ldots s_{i+n-1}$ and our expected output is $s_{i+1} s_{i+2} \ldots s_{i+n}$. Then, the data is split into training, validation, and test data such that test comes later in time than validation comes later in time than training.

3.2 Modeling

The LSTM is Pytorch based and takes in input size, hidden size, and output size. Since we input one value at a time and output one value at a time, the input and output sizes are 1. The hidden size is up to experimentation. Although a larger hidden size may cause the network to overfit to the training data due to memorization of previous values, we found that with a smaller hidden size (roughly 20) the corresponding outputs suffered from high losses. With a larger hidden size (roughly 300) we found that the losses were much less significant.

We utilized mean squared error loss as well as the Adam optimizer, and trained over 20 epochs. Despite using various combinations of parameters, the validation loss was unable to match the training loss, which may be an expected observation due to characteristics of the stock market.

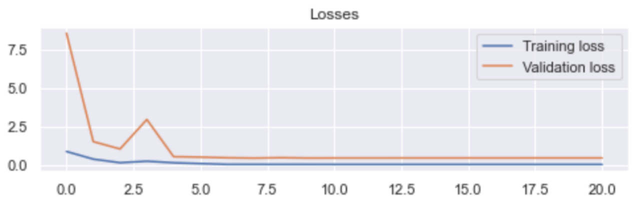

Figure 4: **LSTM loss.** Loss over training epochs.

3.3 Results

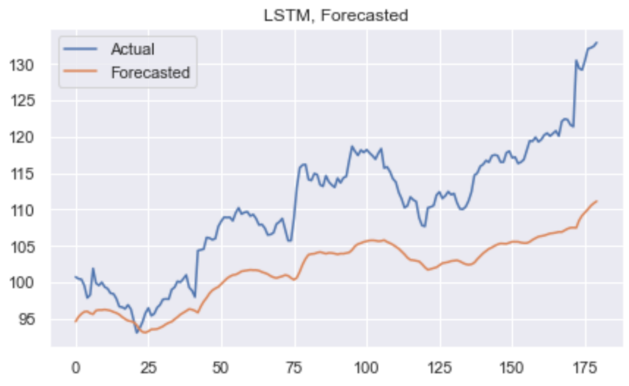

Figure 5: **Forecasted LSTM data.** Results from one run of the LSTM on training data. The forecasts varied significantly based on the chosen hidden size.

Figure 5 displays the generated forecasts for the next 180 days of the market. A first glance can tell us that the results do capture the general trends in a data in a smoother and less pronounced manner. The values are not perfectly aligned which may be expected from any model, but the model seems to capture more information than by, for example, simply averaging the data. We have a few metrics with which we can quantify the accuracy of these predictions:

- **Root Mean Squared Error: 10.0**
  $RMSE = \sqrt{\frac{1}{n}(y-y')^2}$.
- **Mean Average Percentage Error: 7.8%**
  $MAPE = \frac{100}{n}\sum\frac{|y-y'|}{y}$
- **Symmetric MAPE: 8.1%**
  $sMAPE = \frac{100}{n}\sum\frac{2|y-y'|}{|y|+|y'|}$

These values give us quantifiable results from our model, yet they are difficult to contextualize with no baseline. In the next section, we introduce several existing statistical techniques to provide context for LSTMs versus standard techniques.

From an implementation standpoint, LSTMs are more complex and require more computational power than statistical models. In addition, as with most machine learning models, they benefit from large amounts of data, so they are not suitable for problems in which the quantity of data is limited. Contrary to some statistical methods which may perform best when applied to a more local portion of the data, the long memory and prioritization ability of an LSTM may incentivize data across longer periods of time.

4 Statistical Baselines

Machine learning properly applied to time series forecasting is a relatively new phenomenon, with attempts in the past including convolutional neural networks and support vector regression. The use of LSTMs in this domain is still in its infancy, and its particular applicability to financial time series data is up for debate. In fact, an empirical study done in 2018 by Spyros Makridakis et al. [4] comparing the multiple techniques submitted to the M4 forecasting competition claims that LSTMs did not reveal a higher accuracy than existing statistical baselines. Compared to "simpler NNs like RNN and MLP," the LSTM reported "better model fitting but worse forecasting accuracy." Further studies will expand the understanding of usage of this network in time series analysis and forecasting.

In this section, we introduce several existing statistical approaches that serve as competitors to LSTMs in forecasting.

4.1 ARIMA

The **autoregressive moving average ARMA model**, popularized by Box and Jenkins, and its variants including *integrated* ARIMA, *seasonal* SARIMA, and *exogenous variable* ARIMAX feature a combination of autoregression (AR) and moving average (MA). Autoregression involves regressing future values as a function of its own lagged values. Moving average involves predicting future values based on linear dependence of values of a specified window size. The basic `ARMA(p, q)` model with `p` autoregressive terms and `q` moving average terms is:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p}\phi_i X_{t-i} + \sum_{i=1}^{q}\theta_i\epsilon_{t-i}$$

4.1.1 Stationarity

From Figure 3 it is visually apparent that the stock time series is not *stationary* – i.e. it does not exhibit constant mean and variance over time. We analytically confirm this statement using the Dickey-Fuller and KPSS statistical tests. To make a time series stationary, we may use differencing, log transforms, and/or subtracting out the rolling mean to create a stationary representation of the original. The *integrated* component of ARIMA allows us to specify a degree of differencing that can take care of the trend and seasonality of the model, so we will use this fact to manage the stationarity of the data. Prior to using the data in the ARIMA model, we use a log transform, since we find through differencing that the data seems to be *heteroskedastic*, or the variance varies over time.

4.1.2 Parameters

Autocorrelation plots and partial autocorrelation plots can help us determine which parameters to use for AR and MA. These plots summarize the strength of the relationship between one observational value to those occurring prior to that observation. Viewing the ACF plot in Figure 4, we see that there is a strong relationship between an observation and the past 8 or so

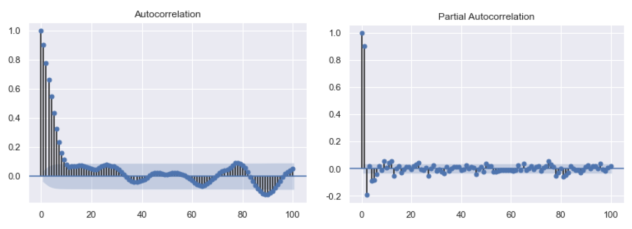

Figure 6: **Autocorrelation (ACF) and partial autocorrelation (PACF) plots.** The ACF plot tells us the moving average `MA(q)` term should be roughly 8, while the PACF plot tells us the autoregression `AR(p)` term should be 2 or 3.

data points. As a result, using the moving average parameter of 8 is appropriate for this problem. Partial autocorrelation, which removes the effect of indirect correlations per step, can tell us how many values have a direct correlation with an observation. Thus, the PACF plot tells us that the autoregression parameter of 2 or 3 is appropriate for this problem.

4.1.3 Results

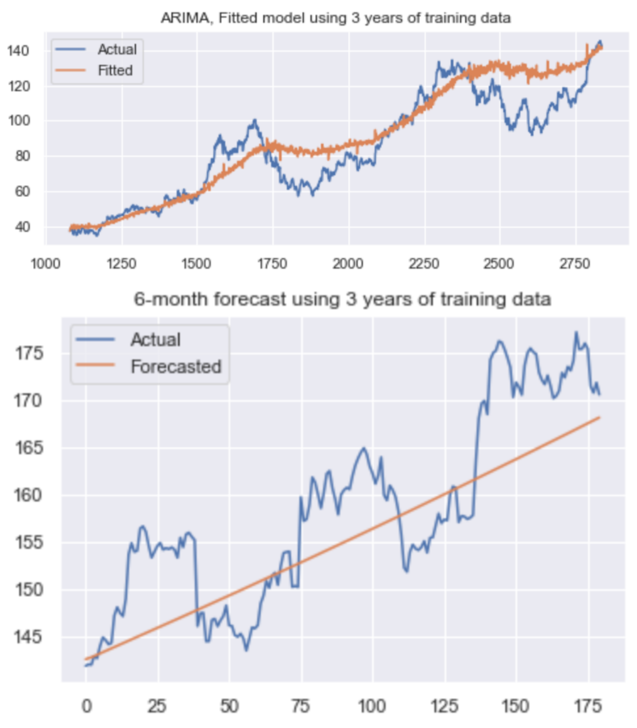

Figure 7: **Fitted and forecasted ARIMA data.** The top displays the fitted values of the ARIMA model on three years of training data. The bottom displays a seemingly linear forecast of the next 6 months of trading.

With `ARIMA(2, 2, 7)` on three years of training data, our model forecasts the next 6 months with the following results:

- $RMSE = 6.2$
- $MAPE = 3.3\%$
- $sMAPE = 3.3\%$

While modeling, ARIMA tended to catch the underlying patterns of the data and was generally able to fit historic data to some reasonable extent. As can be shown in Figure 5, the values forecasted by the model also tended to be *linear* rather than following a complex pattern.

However simplistic, it did seem as though the accuracy of ARIMA was at least quantitatively comparable to that from the LSTM if not better. Testing this on different datasets with various parameters, we found that the fit of the model to historic and forecasted data depended significantly on amount of training data used and the parameters of the model. As a result, it is difficult to form bold claims about the forecasting and modeling ability of ARIMA. However, the quality of predictions from a well-tuned ARIMA does seem to be roughly in the same range as a well-tuned LSTM.

4.2 Exponential Smoothing

Moving averages weight each observation in the past equally and assign an average. However, the observation that more recent values in a time series tend to have a greater relationship to an observed value brings to mind a different weighting scheme: exponential smoothing.

4.2.1 Simple Exponential Smoothing

Simple exponential smoothing is a realization of this idea, and it assigns exponentially decreasing weights as the observations get older. Using a smoothing constant $0 < \alpha \leq 1$ and time period t, the smoothed value S_t is computed by the formula:

$$S_t = \alpha y_{t-1} + (1-\alpha)S_{t-1}$$

As can be seen in Figure 8, simple exponential smoothing does an extremely strong job in modeling historical data. However, it does not have informative forecasting ability because it simply predicts the last computed S_t and uses that constant value for future forecasts. The metrics are:

- $RMSE = 18.8$
- $MAPE = 10.8\%$
- $sMAPE = 12.5\%$

4.2.2 Holt-Winters Method

Simple exponential smoothing assumes no overall patterns in the data, explaining its forecasted results. To handle this, double exponential provides us a method to incorporate trends into the model, while triple exponential further incorporates seasonality into the model. The resulting equations are known as the Holt-Winters method after those who introduced the idea. Essentially, three equations are introduced.

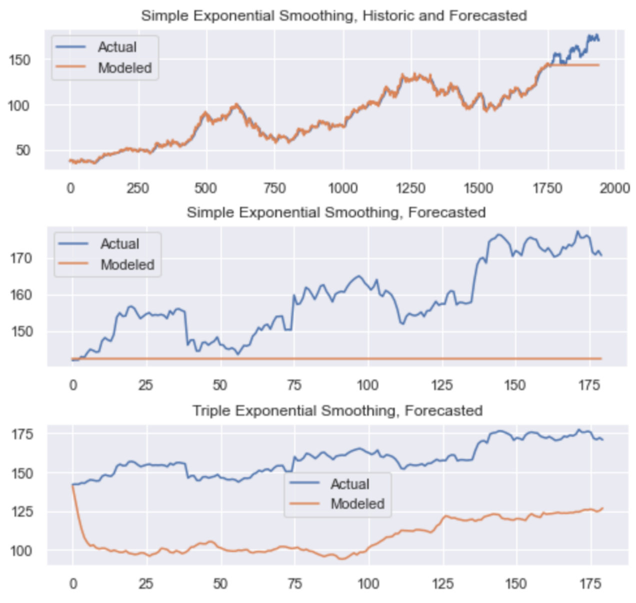

Figure 8: **Exponential smoothing.** *Top:* Historical and forecasted data from simple exponential smoothing. *Center:* The forecasted data remains at a constant value for simple exponential smoothing. *Bottom:* The forecasted value exhibits more pattern for triple exponential smoothing due to trend and seasonal components.

Overall smoothing:

$$S_t = \alpha \frac{y_t}{I_{t-L}} + (1-\alpha)(S_{t-1} + b_{t-1})$$

Trend smoothing:

$$b_t = \gamma(S_t - S_{t-1}) + (1-\gamma)b_{t-1}$$

Seasonal smoothing:

$$I_t = \beta \frac{y_t}{S_t} + (1-\beta)I_{t-L}$$

Due to the added complexity from trend and seasonal components, the forecasted results do not remain constant over time in the same way they do in simple exponential smoothing, as can be seen in Figure 8.

- $RMSE = 40.9$
- $MAPE = 28.5\%$
- $sMAPE = 35.7\%$

Quantitatively this performs worse than a simple straight line, though we can envision (and have seen) time series data for which this would not be the case. Overall, however, the exponential smoothing methods seem to have lower predictive abilities than ARIMA, so ARIMA may be the superior forecasting technique.

4.3 Prophet

A more recent statistical model with a similar goal is Facebook's implementation of Prophet [9]. It is a "procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects," and is stated to work best on data with strong seasonal effects. Essentially, Prophet is a modular model that uses a maximum a posteriori estimate to optimize equations that take into account trends, seasonality, and changepoints.

4.3.1 Implementation

Prophet offers a straightforward library that generates results that are quite intuitive to understand. A collection of dates and values are necessary, and additional components such as changepoints may be specified, allowing the model to be human-tunable to specific needs and industry knowledge. Due to model construction the data needs to be neither evenly spaced nor standardized, which makes its use very simple. As non-experts in the financial field, our implementation does not utilize the key point that the model can incorporate domain knowledge. As a result, the results may not be as strong as they could be with more information.

4.3.2 Analysis

From Figure 9, Prophet seems to emulate its underlying data well, capturing underlying trends and seasonality. Unlike the other models we have explored, this model offers a range for expected lower and upper bounds. This provides us with a better sense of the certainty of the model on historical and forecasted data.

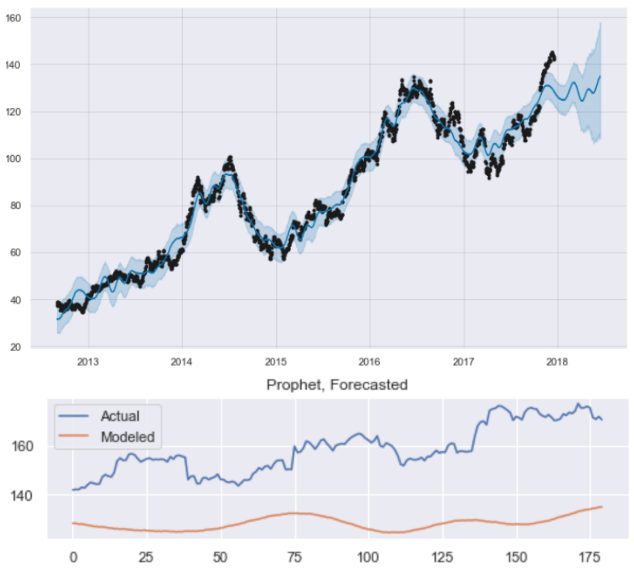

Figure 9: **Fitted and forecasted Prophet data.** Notice that all forecasting data takes the same value: the smoothed value prior to forecasting.

As a combination of several modular components, Prophet is also useful in visually demonstrating the impacts of trend, yearly seasonality, and weekly seasonality found in the underlying data. The plots in Figure 10

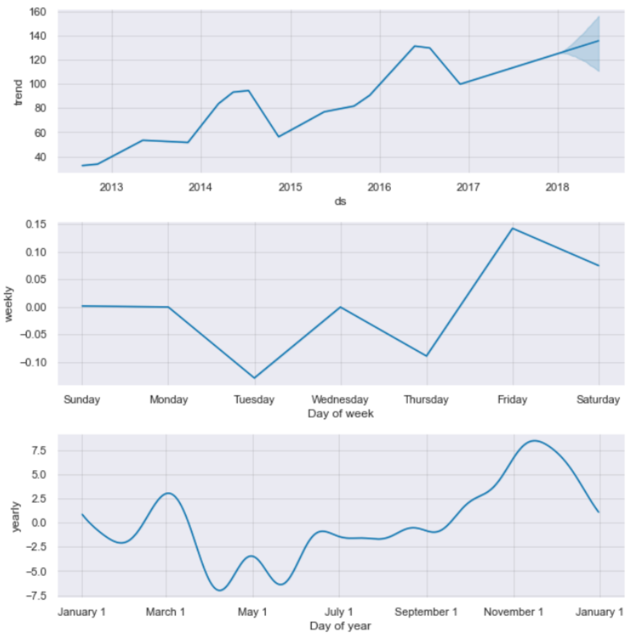

Figure 10: **Modular Prophet components.** *Top:* Overall data trend. *Center:* Weekly seasonality. *Bottom:* Yearly seasonality.

display that the time series features an overall upward trend, with specific spikes in 2014 and 2016. It shows there are seasonal yearly patterns in which prices are highest around November and lowest around May. It also shows that prices on some days of the week, such as Friday, are higher while on others, such as Tuesday, they are lower.

4.3.3 Results

Intuitively, Prophet seems like a reasonably effective and interpretable way to analyze and predict time series data. The metrics are as follows:

- $RMSE = 31.6$
- $MAPE = 18.8\%$
- $sMAPE = 20.9\%$

The model features the ability to incorporate domain knowledge into it, a substantial addition to the forecasting space. We did not add elements, such as changepoints, due to lack of domain expertise. As a result, the results may not have been as optimized as possible.

4.4 Discussion

ARIMA, exponential smoothing, and Prophet are all statistical techniques to model time series data. Each technique has its own strength. Though ARIMA does not fit its inputted data tightly, it seems to capture global trends well leading to overall reasonable, yet simplistic, predictions about the future. Exponential smoothing can mimic the original data extremely well, but seems to have little predictive power when separated from real data. Prophet can help modularize trend, yearly, and weekly seasonality, although it was not shown to have the strongest predictive abilities.

5 Conclusion

Of the models studied, LSTMs and ARIMA were shown to perform most effectively in predicting the future values of time series. The LSTM, a machine learning newcomer to the field, shows promise because its accuracy is within the range of that from ARIMA, yet additions such as denoising and attention mechanisms may provide it with a push toward improved accuracy. ARIMA, a lightweight and well-studied statistical procedure, provides us with reasonable forecasts that may be useful for an initial estimate.

Future work. In a realistic setting, there exists more data than the stock market data in order to detect trends. Interesting work could include using multivariate time series data to explore the performance of LSTMs. Other insider information regarding industry shifts may also be used to fine tune the model for techniques such as Prophet. In addition, a constructed ensemble of results from several techniques may perform more favorably than the results from a single technique – a claim described by Makridakis et al. [5].

References

[1] Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780. Available: `https://www.bioinf.jku.at/publications/older/2604.pdf`

[2] Kim, S., Kang, M. (2019). Financial series prediction using Attention LSTM. *Preprint arXiv: 1902.10877.* Available: `https://arxiv.org/abs/1902.10877`

[3] Liu, J., Chao, F., et al. (2019). Stock Prices Prediction using Deep Learning Models. *IEEE.* Available: `https://arxiv.org/abs/ 1909.12227`

[4] Makridakis, S., Spiliotis, E., et al. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS One.* Available: `https://doi.org/10.1371/journal.pone.0194889`

[5] Makridakis, S., Spiliotis, E., et al. (2018). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting.* Available: `https://www.researchgate.net/publication/325901666`

[6] Orac, R. (2019). LSTM for time series prediction. `https://romanorac.github.io/machine/learning/2019/09/27/time-series-prediction-with-lstm.html`

[7] Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. `PLoS ONE 15(1): e0227222`. Available: `https://doi.org/10.1371/journal.pone.0227222`

[8] Sezera, O., Gudeleka, M., Ozbayoglua, A. (2019). Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005-2019. *arXiv: 1911.13288*. Available: `https://arxiv.org/ abs/1911.13288`

[9] Taylor, S. & Lentham, B. (2017). Forecasting at Scale. *The American Statistician*. Available: `https://research.fb.com/publications/forecasting-at-scale`

[10] DJIA 30 Time Series. `https://www.kaggle.com/szrlee/stock-time-series-20050101-to-20171231`

[11] Dow Jones Industrial Average. `https://www.marketwatch.com/investing/index/djia`