DEPARTMENT OF ENGINEERING MATHEMATICS

Time series analysis using SARIMAX and LSTM models

Philip Harrison

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Wednesday 21st September, 2022

Supervisor: Dr. James Pope

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Philip Harrison, Wednesday 21st September, 2022

Contents

List of Figures

List of Tables

Abstract

Sudden famine is a potential existential risk to humanity. One main method of predicting famine is to monitor food prices for sudden spikes. However, food price modelling methods are typically not optimised towards predicting these spikes accurately. This thesis looks into using SARIMAX models and LSTM (Long Short Term Memory) models to try and predict future food prices. SARIMAX models are a traditional method of time series analysis that are used in many fields. LSTM models are deep learning models whose application in time series analysis is more recent. The models are investigated in two parts. Initially, the models are trained with synthetic data in order to understand how best to build them to learn underlying patterns. Then, the models are trained on real-world data, to see how well they perform in predicting real-world prices. The models perform well on the synthetic data, but have more trouble with the real world data, being unable to track the target price variable accurately.

The oral part of this thesis can be found at https://youtu.be/N-ea5ZgEkm4

Code for this thesis can be found at https://github.com/philhteoma/Msc-Data-Science-Thesis

After careful consideration and discussion with my advisor, it has been decided that this project does not require ethical review.

Supporting Technologies

All model training and data preparation was carried out on my own personal machine.
The following public-domain python libraries were used:

- pandas, a library for loading and manipulating data[21].
- numpy, a library of mathematical tools, used to structure data for input into models[8].
- PyTorch, a deep learning library used to construct LSTM models[15].
- statsmodels, a library used primarily for the included SARIMAX model[17].
- matplotlib, a graphing library for python, used to produce visual images for the thesis[10].
- sklearn, a library with machine learning tools, used for some miscellaneous tools[16].

I used LaTeX to format my thesis, via the online service *Overleaf*.

Acknowledgements

With thanks to Dr. James Pope, my supervisor, whose enthusiasm, knowledge and experience provided inspiration for completing this project. With thanks to Noah Wescombe, for his insights and ideas, his willingness to share them with me, and for giving me the chance to work with ALLFED for my thesis. With thanks, of course, to ALLFED, for their support during this thesis. With thanks to Gavin Leech, an old friend, who pushed me forward when I needed to be pushed. With thanks to my family, for listening to me talk at length about subjects they do not have any interest in. Their patience was invaluable. And with thanks to Corrie, beloved family dog, who passed away aged 15 years during the writing of this thesis. She was, by far, the kindest animal I have ever known.

Chapter 1

Introduction

Many charities in the world are focused on improving well-being in the near term, especially those that focus on global well being. E.g. Recently, however, much attention has been given to safeguarding humanity against potential extinction events – often called "existential risks".

ALLFED (Alliance to Feed the Earth in Disasters) is one such charity. It is focused on finding methods that, in times of sudden famine, could be used to successfully feed the world's population until the crisis has passed. Much of ALLFED's work focuses on novel methods of creating food in low sunlight scenarios, events that may occur through nuclear war or asteroid strike. ALLFED has looked into extracting fats from petroleum[13], rendering leaf biomass suitable human consumption[7], and synthesising carbohydrates using carbon dioxide [12].

Not all food shortages require such novel solutions. Locust swarms are a perennial problem in northern Africa - they can occur suddenly and unpredictably, and devour huge quantities of crops. ALLFED has looked into creating a novel financial product to help combat locust outbreaks[9].

Food price shocks are an important factor in determining areas at risk of famine. Basic economic theory dictates that, as a given commodity become more scarce, it's price rises. Traditional modelling of food prices often assumes consistent, predictable cycles. Commonly used models include SARIMA (Seasonal Auto-Regressive Integrated Moving Average) and Non-Parametric Regression.

Famine, however, is often sudden and unexpected. In dire circumstances, food prices can rise dramatically over a few months. An example of this can be seen in the price of grain due to the recent war in Ukraine. Confidently predicting price spikes in advance would allow both preventative measures to be taken in affected areas, and give time for the mobilisation of disaster response if the crisis cannot be averted.

1.1 SARIMAX Models

SARIMA models have been used to model food prices before, with some success. SARIMA and SARIMAX models can serve as a good baseline for predicting food prices in a vacuum, which further models can then build on.

To be able to predict price spikes, traditional models must be adjusted, or new models implemented. The SARIMAX model is a modified SARIMA model which incorporates exogenous data. Well chosen exogenous data sources could allow the model to respond to price shocks more accurately.

A second possible method of improving SARIMA models is to switch from a parametric linear regression to a Fourier series regression. This could allow the model to more easily track seasonal changes in food price, along with more frequent cycles that may occur.

Another possibility for improving SARIMA models is to incorporate modern neural networks. Long Short-Term Memory (LSTM) models. LSTM's are a variant of recurrent neural networks (RNN). RNN's keep track of the values in their hidden layers with each successive input, and use these stored values when calculating the hidden layer of next input. This is a useful feature to have when data is ordered, as it allows row of the input data to influence the rows ahead of it. In this case, LSTM's may make a good standalone model for predicting food price shocks.

Alternatively, an LSTM - or possibly another deep learning model - could be used to predict the residuals of an ARIMA model. Food prices would be modelled as a simple mathematical trend, and a variety of exogenous variables could be used to predict the difference between this trend and real-world observations.

SARIMAX models are a combination of five different underlying parts. The auto-regressive model assumes that the value at each time step is a function of the values of previous times steps, and follows a form of:

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} ... + \beta_n y_{t-n} + \varepsilon_t$$

where y is a time-series vector, β is a vector of weights, and ε_t is the error at time t. The moving average model assumes that the value at each time step is a function of the error at the previous time step, plus some constant mean:

$$y_t = \mu + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} ... + \phi_n \varepsilon_{t-n} + \varepsilon_t$$

These two models can be combined into an ARMA model:

$$y_t = \beta_0 + \beta_1 y_{t-1} + \phi_2 \varepsilon_{t-1} ... + \beta_n y_{t-n} + \phi_n \varepsilon_{t-n} + \varepsilon_t$$

If the time series does not have a static mean, the model can be adjusted to compensate for this. If the mean is increasing linearly, the method involves taking the difference between each time point and its predecessor, and substituting this into the ARMA model. One possible method for this would be to

$$yt = Y_t - Y_{t-1}$$

Where Y_t is the initial time series.

The SARIMA model takes seasonality into account for forecasting. Expressing this in a compact form requires the use of a backshift operator B, which is defined as

$$B^n y_t = y_{t-n}$$

Using this, a SARIMA model with an AR, MA and I orders of 1 and a seasonality order or n would take the form:

$$(1 - \phi_1 B)(1 - \Phi_1 B^n)(1 - B)(1 - B^4) y_t = (1 + \theta_1 B)(1 + \Theta_1 B^n)\varepsilon_t$$

1.2 Long Short Term Memory Models

Long Short Term Memory models (LSTM's) are modified implementations of Recurrent Neural Networks (RNNs), which are in turn modifications of Deep Neural Networks (DNNs).

In its most basic form, a DNN consists of an input layer, some number of intermediate layers, and an output layer. The input layer is a vector with n members, where n is the number of features being used as input to the model. The output layer is another vector of size m, where m corresponds to the number of features the model is outputting. In many cases, this value can be 1, if the model is predicting only a single value.

The intermediate layers are where the bulk of the complexity in a DNN lies. One of the simplest types of layer is known as a hidden layer. It consists of a series of nodes, with each node having a connection to each node in the layer before it. Each connection has an associated weight. The value of each node in the layer is determined by taking the sum of the products of the values of the nodes in the previous layer and the respective weight of the connection and then passing the result through an activation function. The activation function maps the input to a value between 0 and 1, and usually follows a sigmoid curve. This can be represented as:

$$\mathbf{h} = \mathbf{g}(\mathbf{W^{(1)}}\mathbf{x})$$

Where $\mathbf{h}$ is the hidden layer, $\mathbf{W}_t$ are the connection weights, $\mathbf{x}$ is the input vector, and g is the activation function.

After the input has been passed through the network, the values in the output layer are recorded, and the loss is calculated using one of many functions. The loss is then used to modify the connection weights in the neural nets. This is done by computing the gradient of the loss with respect to each weight, and then adjusting that weight in the opposite direction of the gradient.

An RNN expands on this by tracking the values of the hidden layer as new inputs are run through the model. Instead of solely relying on the input to calculate the hidden state h, the model also uses the values of the hidden state from the previous input, referred to as $\mathbf{h_{t-1}}$. The equation for determining the hidden state becomes:

$$\mathbf{h_t} = \mathbf{g}(\mathbf{W^{(1)}x_t} + \mathbf{Uh_{t-1}})$$

Keeping track of the hidden state in this manner allows the model to accommodate previous inputs when calculating the next hidden state. This is useful for tasks such as text analysis, where prior words in a sentence can change the meaning of other words [2].

RNN's can have trouble transmitting information over large distances, however. Updating weights using backpropagation has an inbuilt issue where gradients get smaller when moving backwards through time, as each step multiplies partial derivatives together.

LSTM models attempt to solve this issue by including a memory cell. This cell is separate from the hidden layer, and its contents is primarily determined by three gates - a forget gate, an add gate, and an output gate Figure 1.1

The forget gate controls what information is removed from the cell at each time step. It only looks at the previous cell state when making modifications.

The add gate takes information from the next input to the model and adds it to the cell. It functions much like a standard neural net layer.

The output gate simply passes the current cell state through an activation function.

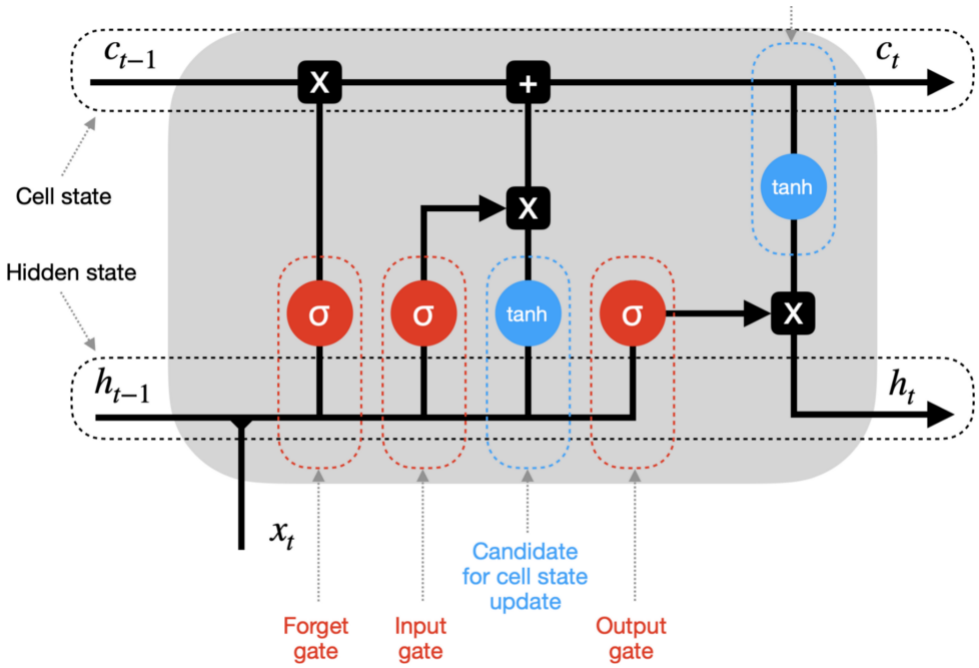

Figure 1.1: Visual depiction of the structure of a Long Short Term Memory layer. Image courtesy of towardsdatascience.com[4]

In a SARIMAX model, the auto-regressive part of the model is based on the assumption that the value of the time series at a given step is some function of its values at previous steps. The structure of an LSTM allows the model to take advantage of this relationship, by letting earlier entries in the time series be contextualised and influence predictions later in the time series.

This setup also allows for easy extrapolation of the input data far into the future. SARIMAX models often have trouble predicting more than a few points into the future, as their predictions normally tend towards a mean. LSTM models are able to avoid this issue and are more capable of learning underlying patterns and extrapolating them into future predictions.

Additionally, LSTM models are easily able to incorporate exogenous variables into their training. In many cases, this will simply mean adding extra features to the input and letting the model work out how best to incorporate the extra data.

Chapter 2

Related Work

Forecasting Commodity Price Index of Food and Beverages in Kenya Using Seasonal Autoregressive Integrated Moving Average (SARIMA) Models

Wanjuki *et al.*[20] explore using SARIMA to model the Kenyan food and beverage price index. They explore multiple different model orders, settling on SARIMA$(1,1,1)(0,1,1)_{12}$. The final model has a result with an RMSE of 3.37 and a MAPE of 1.62. Figure 2.1 shows the residuals of the model predictions. Notably, the model does a poor job at predicting a price crash in 2009. There is, therefore, room for improvement on this model regarding its ability to predict price shocks.

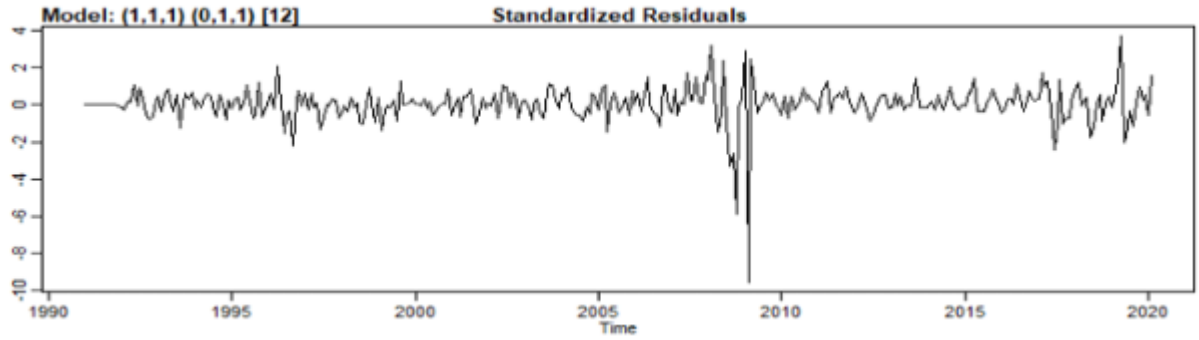

Figure 2.1: Wanjuki *et al.* [20]

Predicting Staple Food Materials Price Using Multivariables Factors (Regression and Fourier Models with ARIMA)

Asnhari *et al.*[1] investigate fitting linear regression and Fourier regression models to ARIMA food price models. Fourier regression slightly underperforms multiple linear regression in most cases. In two cases - Green Cayenne Pepper and Garlic prices - linear regression performs extremely poorly, while Fourier regression has results similar to its performance on other commodities. Results are shown in Figure 2.2. Fourier regression may help provide a better underlying model for food prices, which can then be used to help predict price shocks.

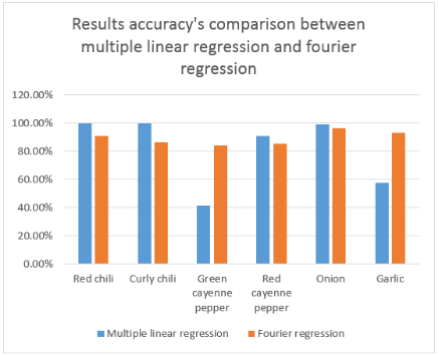

Figure 2.2: Asnhari *et al.* [1]

Comparing the Performance of Seasonal ARIMAX Model and Nonparametric Regression Model in Predicting Claim Reserve of Education I

Ulyah *et al.* [19] also explore using Fourier series to fit time series models. They attempt to model claim reserves of education insurance companies. Their results are strikingly different to Asnhari *et al.* - their Fourier series is unable to pick up on the seasonality is the data. It

does manage to capture the rising mean, however. They also fit a SARIMAX model, which performs significantly better.

Forecasting economic and financial time series : ARIMA VS LSTM

Siami-Namini[18] *et al.* compare forecasting financial time series using both ARIMA and LSTM. They find striking results, with LSTM achieving 80-90% reductions in RMSE on multiple data sets. LSTM performs noticeably more poorly on both the Medical Care and Food and Beverages data sets, though in both cases still ties or beats ARIMA.

National-scale electricity peak load forecasting: Traditional, machine learning, or hybrid model?

Lee *et al.*[11] explore using explore combining SARIMAX models with machine learning algorithms to predict electricity usage in South Korea. They find that Long Short Term Memory (LSTM) models significantly outperform the SARIMAX model. Notably, the LSTM is significantly better at predicting spikes or falls in electricity usage. They note that a hybrid SARIMAX/LSTM model performs better than the LSTM model in some circumstances, but not others.

Forecasting Indonesia Exports using a Hybrid Model ARIMA-LSTM

Dave *et al.*[3] explore using a hybrid ARIMA/LSTM model to predict exports in Indonesia. They use seasonal decomposition to separate the data into trend, seasonal and residual components. They then model the trend with ARIMA, and the seasonality and residuals with LSTM models. They also try modelling the data with LSTM and SARIMAX separately. They find a hybrid model shows an improvement in MAPE (7.38, compared to 8.56 for LSTM and 9.38 for SARIMAX). Figure 2.3 shows the hybrid models predictions against actual values. We can see the model is not predicting spikes particularly well.

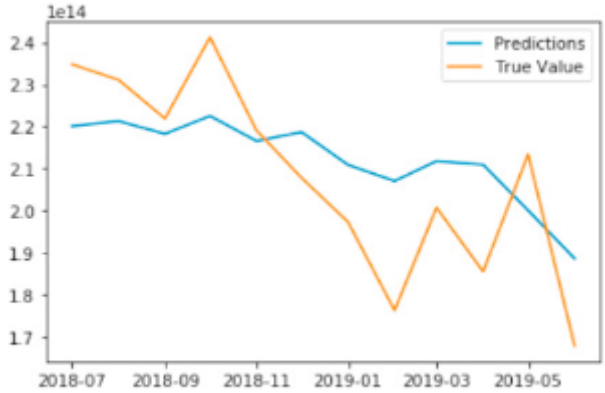

Figure 2.3: Dave *et al.* [3]

Time Series Forecasting Using LSTM Networks: A Symbolic Approach

Elsworth and Güttel[6] investigate using a symbolic approach to improve LSTM performance on time series analysis. They use a compression algorithm called ABBA, which they developed and documented in a prior paper[5]. The ABBA compression algorithm reduces the time series to a series of tokens, with each token representing a specific length and height. The number of unique tokens used is variable and decided at the time of compression, though a maximum can be specified.

Chapter 3

Execution

3.1 LSTM Design

The basic LSTM design was heavily inspired by a tutorial by Charlie O'Neill on towardsdatascience.com[14]. It is a standard PyTorch model training loop using LSTM layers. The LSTM model and training loop were designed to be easily modifiable. Modifiable aspects included:

- The number and size of the LSTM layers
- The training algorithm and loss measurement used
- The learning rate of the training algorithm
- The number of batches to split the input data into

The model was designed to take a list of integers as an argument when initialised. This list corresponds to the size of the hidden layers in the model. For example, if passed a list of [64, 32, 16], the resulting model would have three hidden LSTM layers of sizes 64, 32 and 16 respectively. An argument for the number of input features is also passed in.

The forward method of the model has also been altered from the template. The generation of the initial hidden layers and cell states is now based on the number and size of the hidden layers and is generated at runtime. The forward method is also modified to allow vectors to be passed in as input, allowing for multiple features to be used in the LSTM.

A feature carried over from the template is the ability for the model to continue to predict values after it has finished using the input it has been given. It achieves this by using its own output as a new input for each step. When using inputs with multiple features, this involves removing the oldest feature from the input and then appending the model's output.

3.2 Simulated Data

Exploratory model design was initially done on simulated data. The simulated data initially consisted of simple sine waves (Figure 3.1). After the data was generated, the points were randomly offset using a standard distribution to add noise to the data (Figure 3.2).

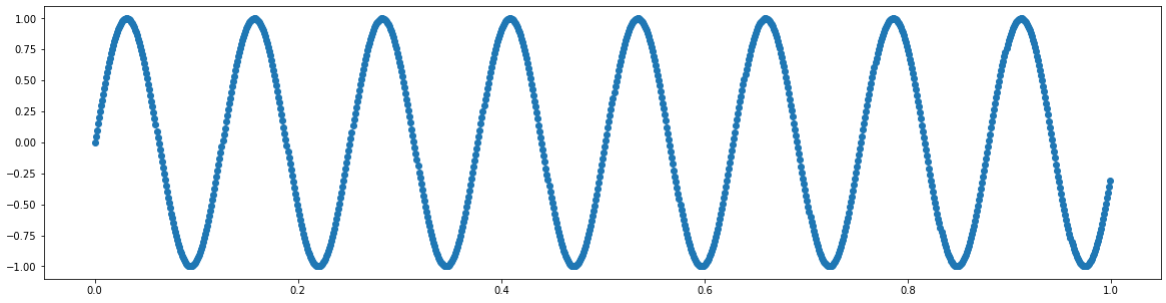

Figure 3.1: Simulated data consiting of a sin wave

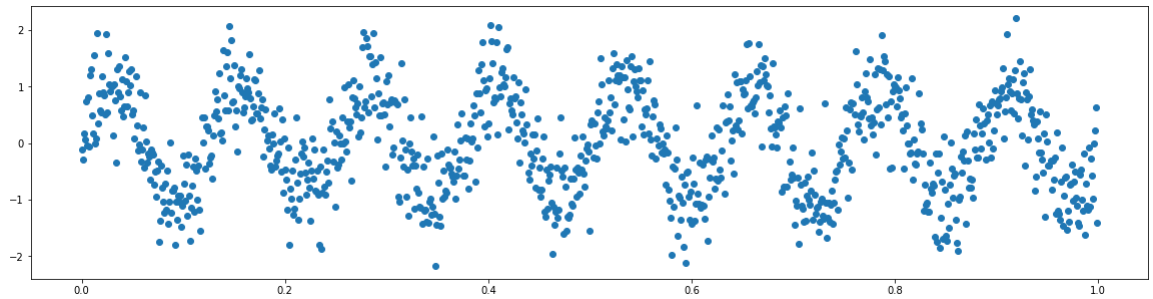

Figure 3.2: Simulated data with added noise

Model	Last Value	SARIMA	Single Feature LSTM	25 Feature LSTM
Loss	0.514	0.218	0.225	0.167

Table 3.1: The training losses of the different models.

3.3 Initial Models

This simulated data was fitted to four models:

- A simple model that simply guesses the previous value in the sequence
- A SARIMA model of order $(1, 0, 1)(1, 0, 1)_{100}$
- An LSTM model with one input feature (The previous value in the sequence)
- An LSTM model with 25 input features (The 25 previous values in the sequence)

The models were trained on the first 2000 points of the training set and were then used to extrapolate the remaining 1000 points. These were compared against the true underlying pattern without the added noise. The losses on training on the training set were calculated using RMSE. (Table 3.1). Calculating loss on the extrapolated data was more tricky, as a model that simply predicts one value for every future point is arguably less useful than a model predicts a sin wave half a period out of sync (which would have twice the error). Instead, the models individual absolute errors for each extrapolated point were graphed and then compared (Figure 3.3).

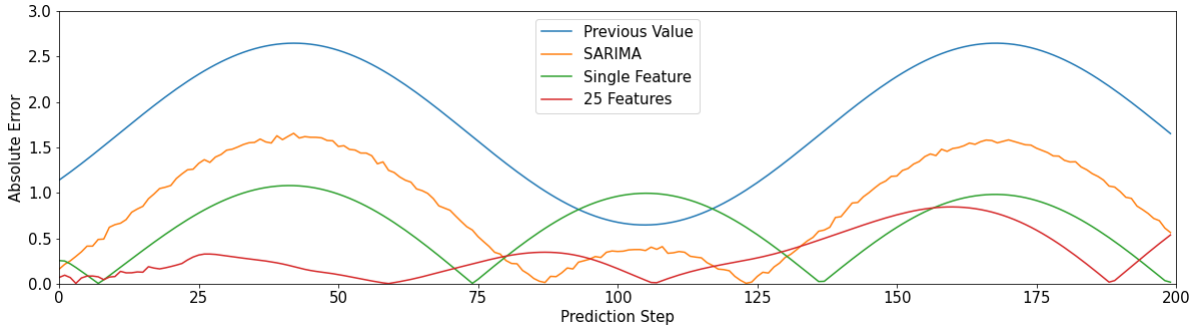

Figure 3.3: Error in predicted future values of sin wave. Despite always guessing the same value, the "Previous Value" model is never correct, as random noise added to the last point of the dataset pushes that value to be greater than any point of the data without the noise

For every model except the 25 feature LSTM, the model starts to guess a single value very quickly, resulting in the sin wave patterns in the error. The 25 feature LSTM, however, maintains an error much closer to zero than the other models, as its predictions form a sin wave (Figure 3.4)

A more complicated periodic time series was then obtained by summing two different sin waves (Figure 3.5). Noise was then added in the same manner as the previously generated data (Figure 3.6). The same four models were then trained on the first 2000 points of this data and left to predict the last 1000 points. The losses on the first 200 predicted points for each model were charted (Figure 3.7).

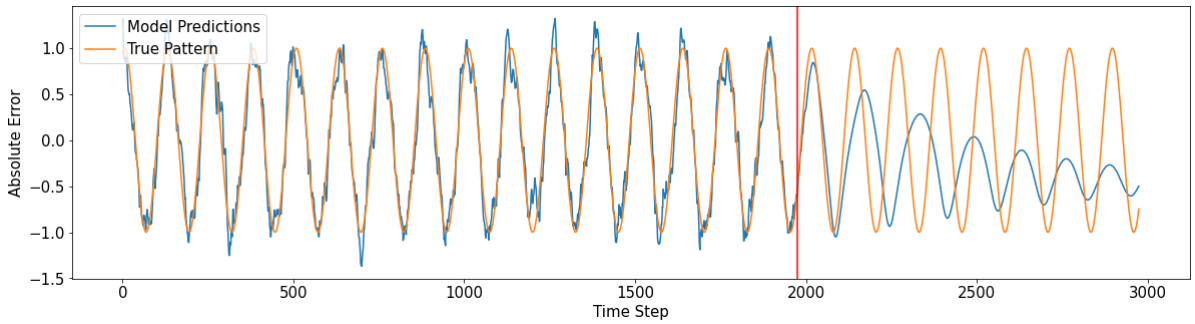

Figure 3.4: Predictions of 25 Feature LSTM against the true underlying pattern. The data prior to the red bar on the graph was used for training. The blue line after the red bar represents the model's extrapolation once the training data has been finished.

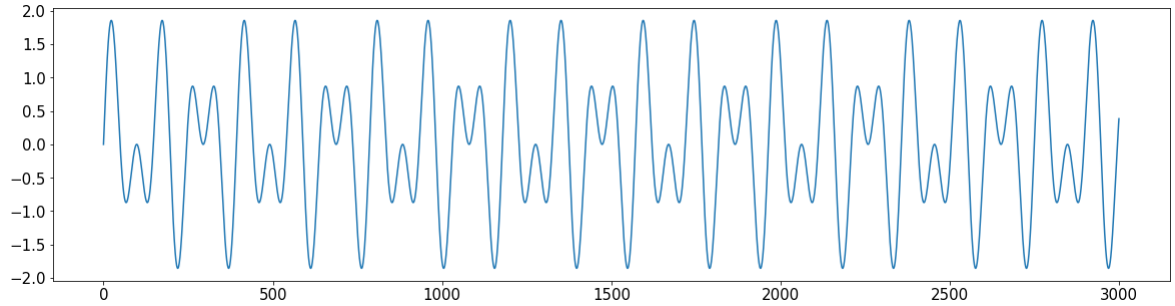

Figure 3.5: Generated Fourier Series

Only the 25 feature LSTM was able to track the Fourier series at all, and even then it did so poorly. It did, however, outperform the other models.

3.4 Hyperparamter Exploration

After identifying the multi-feature LSTM as a promising candidate for time series prediction, tests were performed to attempt to optimise the model's performance on the simulated data. The following hyperparameters were looked at:

- The number and size of the hidden LSTM layers
- The number of features (previous time values) in the input
- How many batches the training data was split into
- The learning rate of the model

3.4.1 Hidden Layer Size

A number of models were trained with various configurations of hidden LSTM layers. These models were then compared by their performance in predicting future data. The same set-up of using 2000 points of training data and 1000 points of test data was used to evaluate the models. The other hyperparameters were fixed at 25 input features, 1 batch of training data, and a learning rate of 0.08. Visualising the performance of these models is difficult, as they tended to either tend towards predicting a large sin wave, or just the same value repeatedly (Figure 3.8).

The best performing models used only a single, large, LSTM layer. Models with multiple layers would tend towards predicting a single value much faster than those using only a single layer.

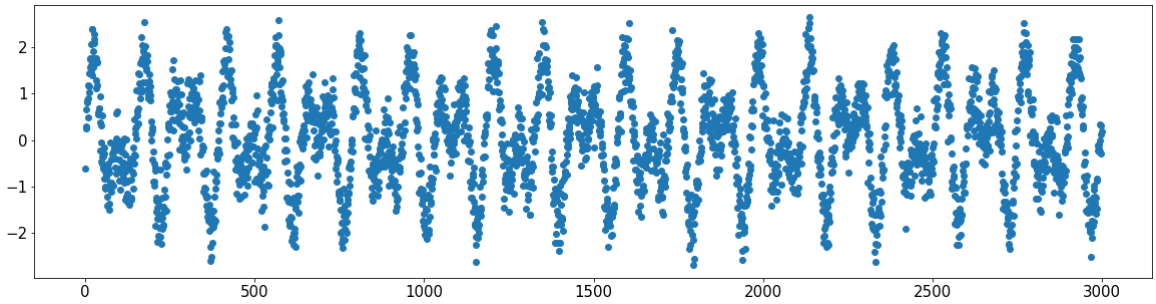

Figure 3.6: Generated Fourier series with added noise

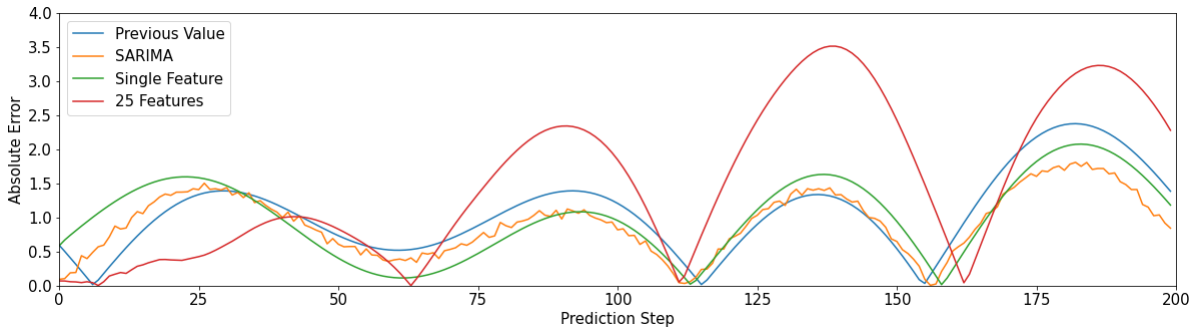

Figure 3.7: Error in predicted future values of the Fourier series between different models.

3.4.2 Feature Count

A number of models were then trained using varying numbers of features in their inputs. As before, models were compared on their performance on predicting unseen points in the Fourier series (Figure 3.9). Hyperparameters other than the number of features were kept the same as prior, with the hidden layers now being fixed at a single layer of size 128.

The models perform significantly better with more features in their inputs. Notably, at 100 features, the model settles into a state where it is able to consistently predict the Fourier series indefinitely (Figure 3.10).

3.4.3 Training Data Batches

A method was devised to split the training data into an arbitrary number of batches when training. To split the data into n batches, $n-1$ random integers are generated between 0 and the length of the input data. These integers are used as markers of where to split the training data. A parameter can be passed into the training loop to enforce a minimum size of each batch. The data is then partitioned into these calculated batches, and the batch order randomised. For every epoch of training, new batches are randomly generated from the training data.

Models were trained as before, with the number of batches varying from 1 to 5 (Figure 3.11). The number of features used when training was set to 100 for each model.

Since the models are now extrapolating the data accurately, it is now feasible to use more traditional methods of error calculation to compare the performance of the different hyperparameters. In this case, the models can be compared on the RMSE of the predicted values of the training data (Figure 3.11).

While the overall effect is slight, there is a noticeable reduction in loss when using 3-4 batches when training the model.

3.4.4 Learning Rate

Models were trained using varying values for the learning rate of the LBFGS optimiser (Figure 3.12). The hyperparameters were set as before, with the training data being split into three batches for each epoch.

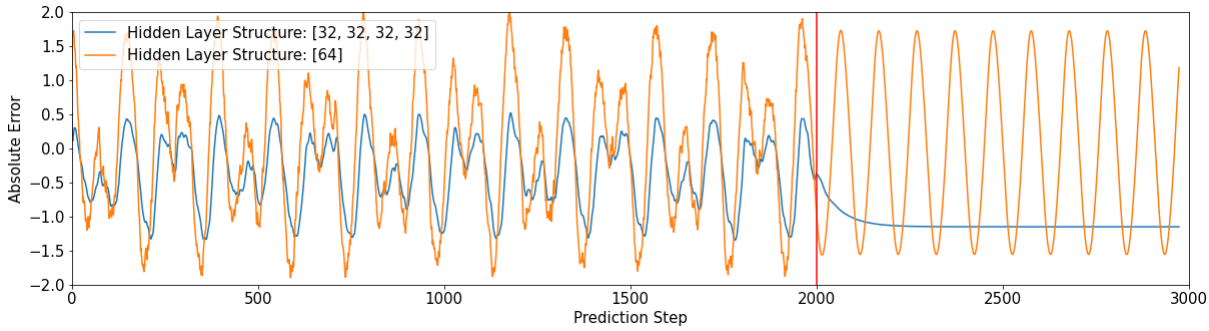

Figure 3.8: Predictions of two types of model - one with multiple small LSTM layers, and one with a single large layer.

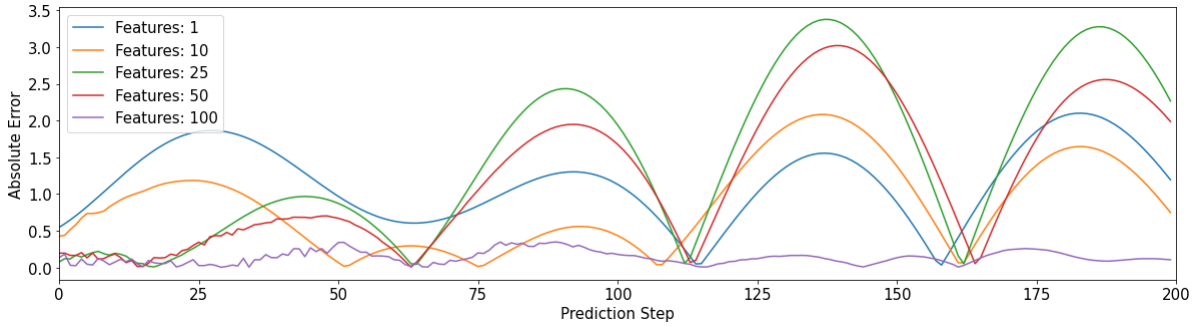

Figure 3.9: Error in predicted future values of the Fourier series when varying the number of features in the input.

3.5 Complex Fourier Series

To test the model's applicability on more complex data sets, a more complicated Fourier series was constructed (Figure 3.13). Noise was again added to the series to create the training set(Figure 3.14).

An LSTM model with the previously identified well-performing hyperparameters was trained on this data. This was a model with a single hidden LSTM layer of size 128, a learning rate of 0.08, with an input size of 100 features. The training data was partitioned randomly into three separate batches for each epoch of training. Additionally, a second model was trained using 300 features instead of 100 features.

3.6 Training on Future Data

All previous models had been trained in the same manner - providing the model with a list of values from the training data leading up to point n in the time series, and comparing the model's prediction for the value of $n + 1$ with the true value. An alternate way to train the model is to instead allow the model to see the training data, and then predict some number of steps into the future. For example, if training on a batch of 500 data points, the model could be allowed to see the first 300 and asked to predict the last 200. The loss would then be calculated between the predicted values and the actual values.

Two models were fit with this method to the complex Fourier series shown in Figure 3.6 and Figure 3.14. These models were trained with 300 input features, a single LSTM layer of size 128, and a learning rate of 0.08. The training data was not split into batches. The model weight were adjusted based on the model's ability to predict 1000 points into the future. The results of fitting the model to the simpler Fourier series are shown in Figure 3.17, and the results of fitting to the more complicated series are shown in Figure 3.18.

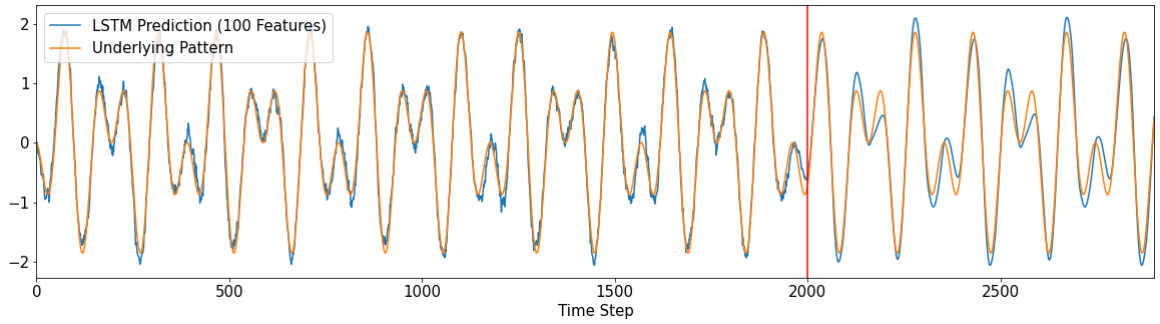

Figure 3.10: Full predictions of an LSTM with 100 features as input, compared against the true underlying Fourier Series.

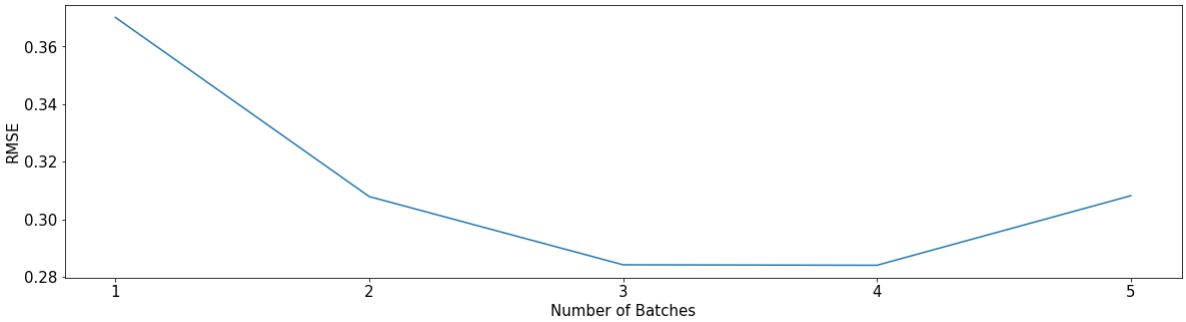

Figure 3.11: RMSE between the models predicted values and the underlying Fourier series.

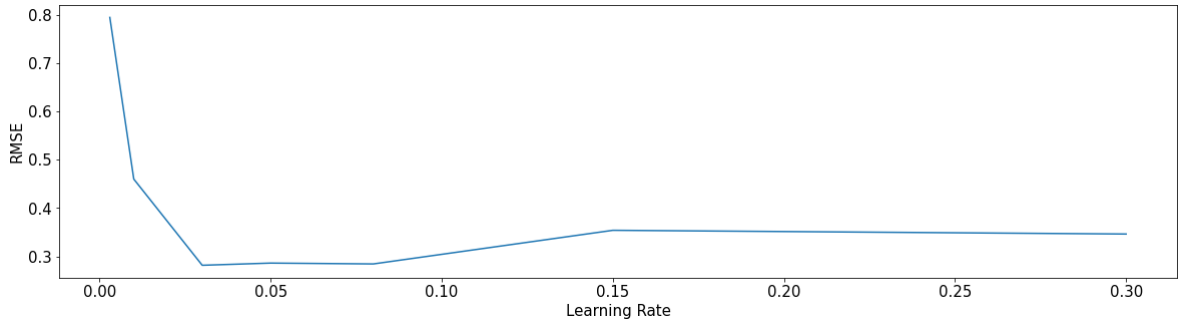

Figure 3.12: RMSE between the model's predicted values and the underlying Fourier series.

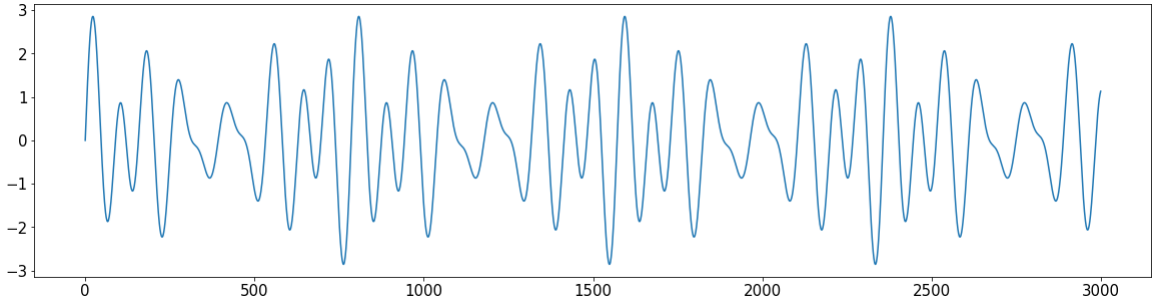

Figure 3.13: Complex Fourier series for testing LSTM models.

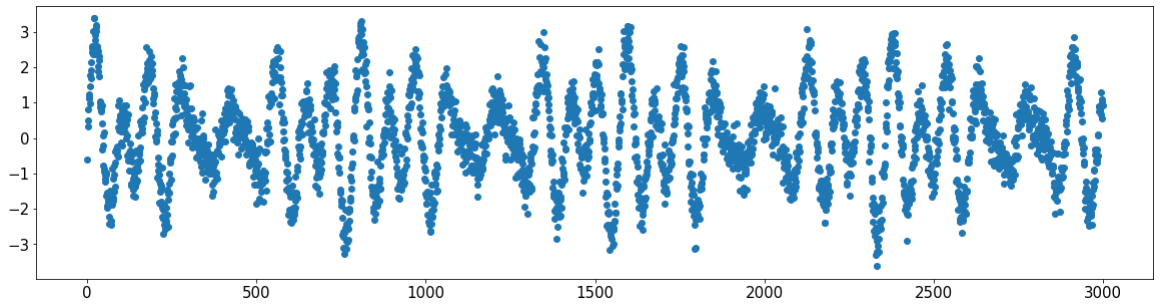

Figure 3.14: Complex Fourier series with added noise.

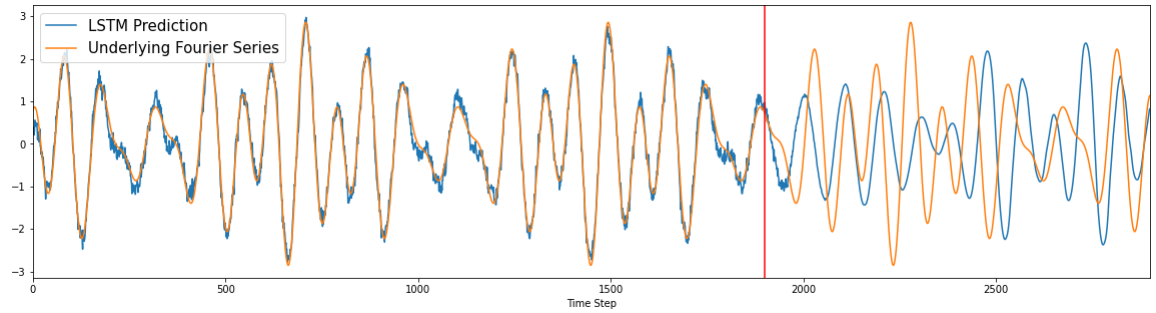

Figure 3.15: LSTM fit to a more complicated Fourier series. Points after the red line were extrapolated by the model without seeing input data.

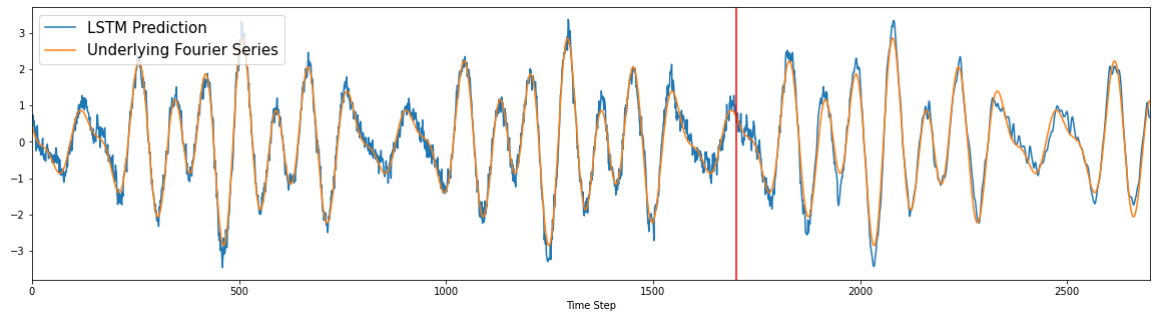

Figure 3.16: A model trained as in Figure 3.15, except using 300 input features instead of 100

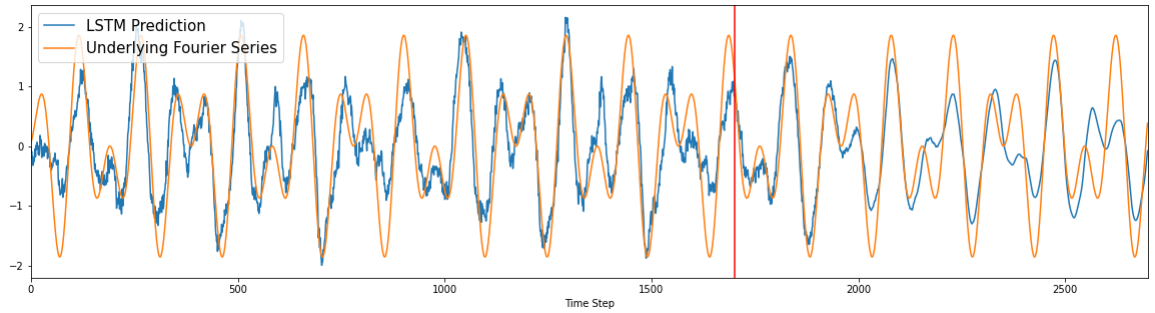

Figure 3.17: LSTM fit to a more complicated Fourier series with loss being calculated on the model's ability to extrapolate 1000 points into the future. Points after the red line were extrapolated by the model without seeing input data.

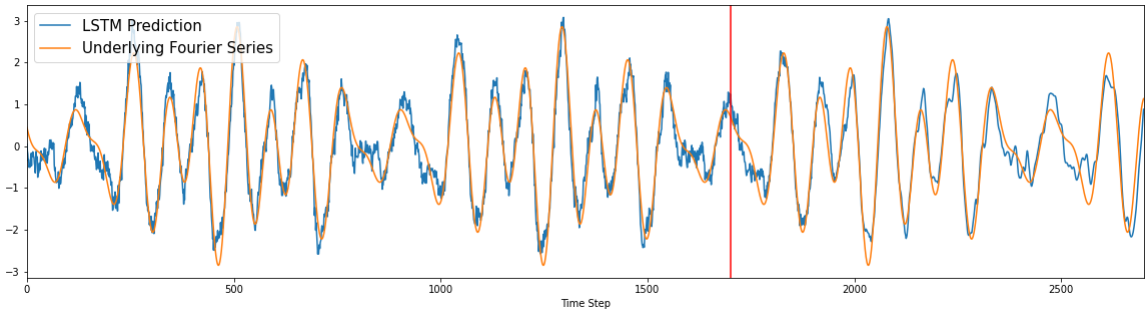

Figure 3.18: A model trained as in Figure 3.17, except using a more complicated Fourier series as training data

Chapter 4

Real World Testing

4.1 Sourcing Data

Once a suitable model structure had been identified, testing on real-world data could begin. The initial testing showed that a large volume of data would help with model performance, as it would allow for more features to be used in the input. While many sites offer comprehensive, fine-tuned data for download, it was often locked behind paid subscriptions.

Three data sets were sourced. The first was a daily tracker of rice futures in US dollars per hundredweight (Figure 4.1). The second was a daily tracker of brent oil price per barrel in Europe, also in US dollars (Figure 4.2). The third was a track of the rainfall in china, subdivided by month (Figure 4.3).

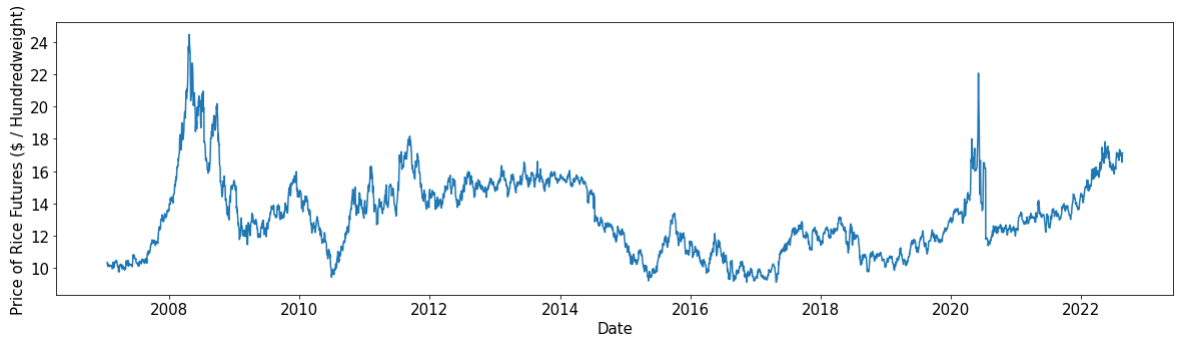

Figure 4.1: Rice future price over time. Data sourced from https://uk.investing.com.

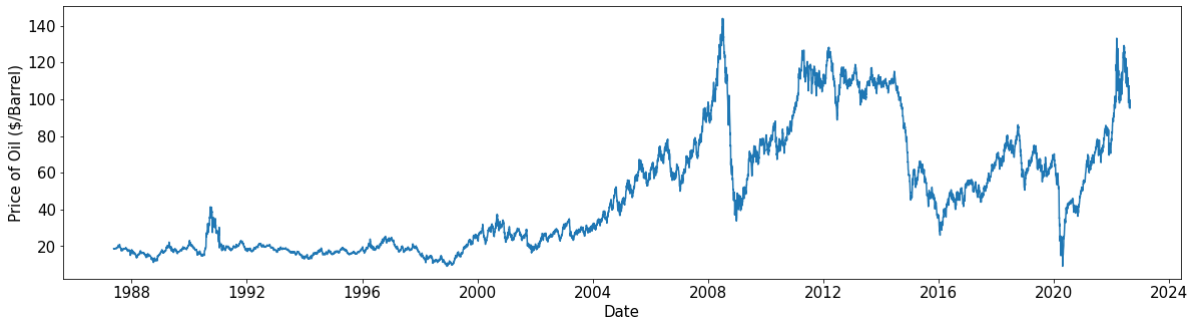

Figure 4.2: Oil price over time. Data sourced from https://www.eia.gov.

Converting the data into a usable time series format was simple for the rice futures and oil data sets, as these were provided in a CSV format, with one entry per day. The rice and oil data were joined together on the date column to obtain a single dataset.

The rainfall data was more complicated to convert, as the table structure included a year column and twelve month columns. To extract relevant information, a python function was built to grab the rainfall from the table, using year and month as arguments. This function was then applied to the rice and oil

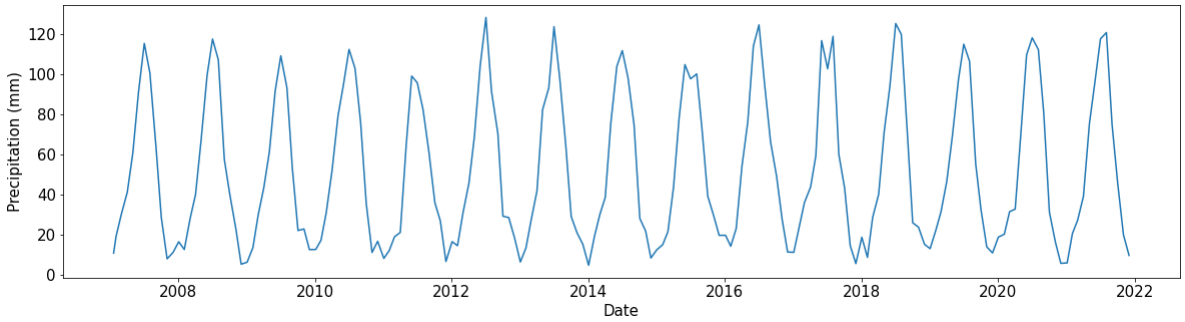

Figure 4.3: Rice future price over time. Data sourced from https://climateknowledgeportal.worldbank.org.

Date	Rice Price	Oil Price	Rainfall
30/01/2010	13.90	70.65	12.61
31/01/2010	14.20	71.20	12.61
01/02/2010	14.15	71.58	17.2
02/02/2010	14.45	73.94	17.2

Table 4.1: Formatted time series table, ready for use in training LSTM models.

table, extracting years and months from that table's date column. The result was a DataFrame of daily entries for rice prices, oil prices and rainfall (Table 4.1). This format was chosen as converting pandas DataFrame columns into tensors is a relatively simple process, allowing for quick subsetting of data for training various different models.

4.2 Initial Modelling

An ARIMA and an LSTM model were trained using only the rice futures data as input (Figure 4.4. The ARIMA model was structured with an order of (1, 0, 1). The LSTM model used hyperparameters derived from the testing on simulated data - a single hidden LSTM layer of size 128, a learning rate of 0.08, and 100 input features. Additionally, for both models, the data was adjusted to have a mean of 0 by subtracting the mean of the whole data set from each point in the set.

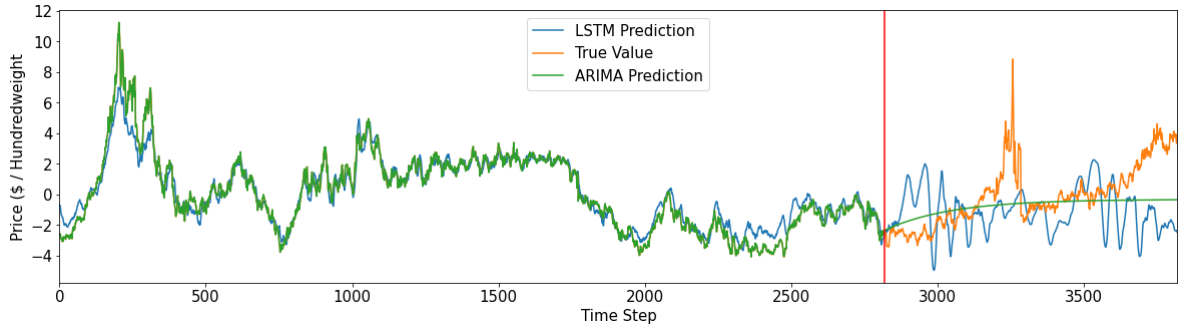

Figure 4.4: Results of ARIMA and LSTM models when trained using only rice futures data. Predictions prior to the red line were made point by point with previous data. The ARIMAX predictions prior to the red line are close enough to reality that they overwrite the line. Predictions after the red line were extrapolated

4.3 Modelling with exogenous variables

Oil prices were then incorporated into the model. This was simple with the ARIMAX model, as it allows for the exogenous data to be passed in as a separate argument. For the LSTM model, this required a rework of the data preparation. The input data was still partitioned into windows n data points. This

process was applied for both variables - both endogenous and exogenous. These windows were then concatenated together. For example, when constructing the input data where $n = 50$, the resulting input would have 100 features, 50 from each of the input variables.

The model's mechanics for predicting future values also had to be altered, as it could no longer rely solely on its own output. This is due to the model not predicting the exogenous variable, only the endogenous variable. The model's forward function was modified to accept an argument containing future exogenous data, instead of an integer specifying how many future predictions to make. Once the model had finished predicting data using both endogenous and exogenous variables, it would then start using the provided extra exogenous data, while updating the endogenous data with its own predictions. Since the input data was one continuous vector, this required splitting the input vector into the endogenous and exogenous parts, updating the endogenous data, and then concatenating the new endogenous data with the next exogenous data point.

Using this method, an ARIMAX and an LSTM were constructed using the rice and oil data. This was done twice - once using only the first 3000 data points of the set, to avoid including a price shock to both rice and oil in 2020 (Figure 4.5), and once using the full data set (Figure 4.6).

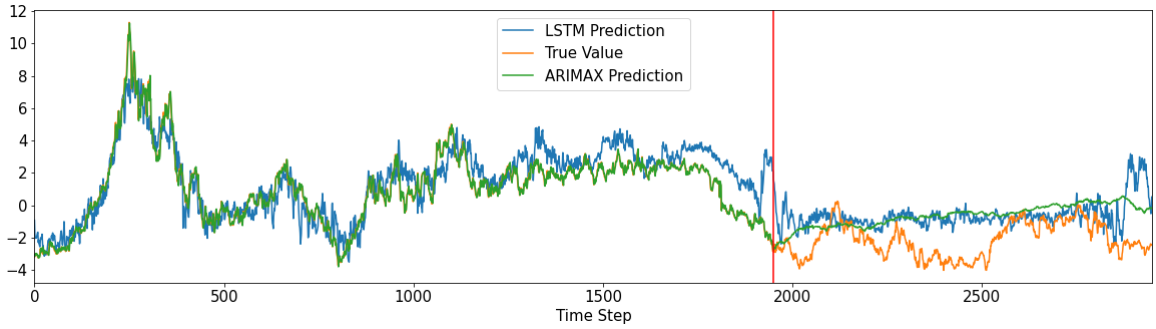

Figure 4.5: Predictions of ARIMAX and LSTM models on Rice Futures data, using oil price as an exogenous variable. Training data was limited to the first 3000 data points, in order to exclude a shock in the rice and oil prices in 2020. Predictions prior to the red line were made point by point with previous data. The ARIMAX predictions prior to the red line are close enough to reality that they overwrite the line. Predictions after the red line were made with oil price data only, with rice future price data extrapolated.

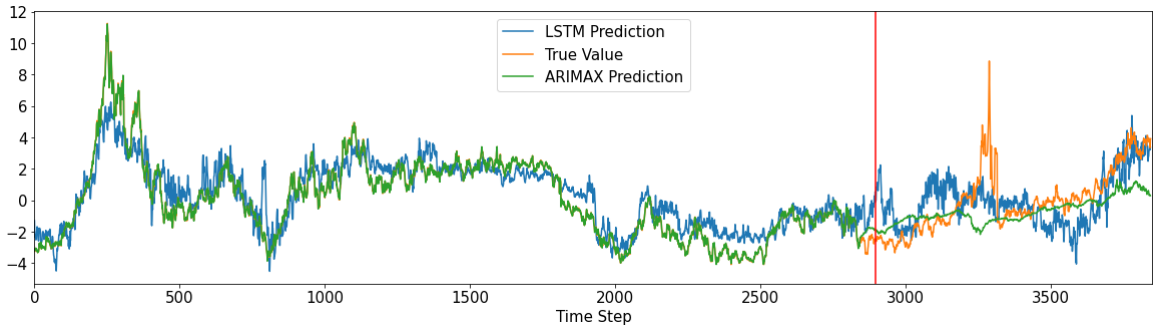

Figure 4.6: Results of ARIMAX and LSTM model training as in Figure 4.5, except using the full dataset instead of the first 3000 values.

The input data was then expanded to include the rainfall data, and models were trained as previous. Two sets of results were obtained, one from training using only the first 3000 data points (Figure 4.7), and another from training on the full data set (Figure 4.8).

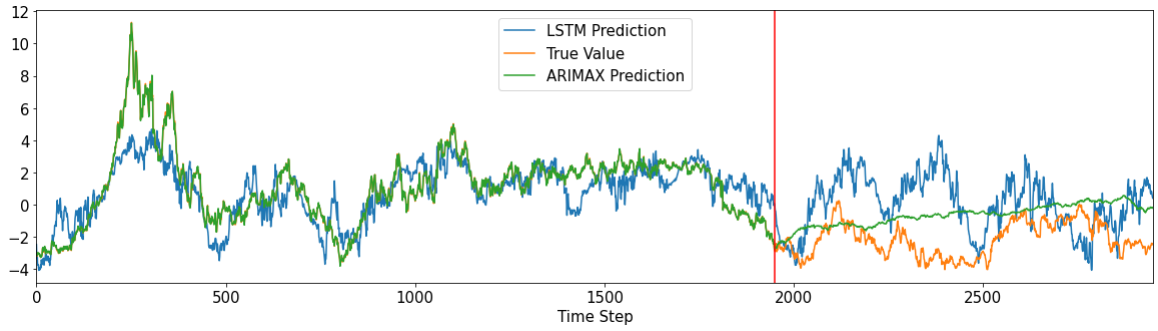

Figure 4.7: Predictions of ARIMAX and LSTM models on Rice Futures data, using oil price and rainfall as exogenous variables. Training data was limited to the first 3000 data points, in order to exclude a shock in the rice and oil prices in 2020. Predictions prior to the red line were made point by point with previous data. The ARIMAX predictions prior to the red line are close enough to reality that they overwrite the line. Predictions after the red line were made with oil price data only, with rice future price data extrapolated..

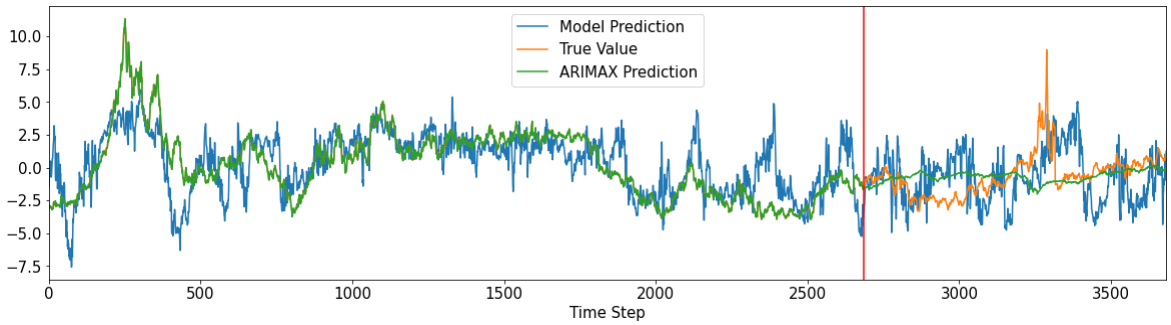

Figure 4.8: Results of ARIMAX and LSTM model training as in Figure 4.8, except using the full dataset instead of the first 3000 values.

Chapter 5

Critical Evaluation

5.1 Simulated Data

The LSTM models performed well on the simulated data once appropriate hyperparameter configurations were found. LSTM models were able to learn complicated patterns when trained with appropriate configurations, as shown in Figure 3.10 and Figure 3.16. They were able to continue to predict these patterns even in the absence of input data, simply extrapolating from their own outputs. This is a task that SARIMA models struggle with, as they will normally tend towards a mean value in the absence of extra input data.

There were multiple aspects of LSTM construction I was unable to test, however. I find it notable that increasing the size of the time window that the LSTM is exposed to (by adding more features) has such a drastic effect on performance. This can be seen in Figure 3.15 and Figure 3.16, where tripling the number of features allowed the LSTM to learn the underlying pattern. It is possible that using a more conventional deep neural net would produce similar results if it had access to this much input data. Due to time constraints, I was not able to build and test models of this nature.

I was initially excited about the idea of calculating a models loss using only extrapolated data, as in Figure 3.17 and Figure 3.18. This goal is more closely related to what I want the model to be able to do, so I had hoped it might produce better results. However, while the model was still able to predict the Fourier series, it did significantly worse than when trained with a more standard method.

5.2 Real World Data

The LSTM data performed more poorly than I had hoped on the real-world data. The LSTM showed, in general, little improvement in predictive power over an ARIMAX model. It could be argued that the model depicted in Figure 4.6 does a better job at following the real rice price than the ARIMAX model - however, I do not think the correlation is strong enough to state this confidently and could be due primarily to chance.

Time constraints affected this section much more than the simulated data section, and I was not able to investigate nearly as many things I would have liked to, including:

- More endogenous data sets. I feel that the rice futures set is good, but a comparison between other sets would provide a lot of insight
- More exogenous data sets. I only used two exogenous data sets here, only one of which was likely to correlate well with the endogenous data set. Comparisons of the effects of different exogenous data sets on model performance could have provided insights into not only the model but also how well the exogenous data relates to the endogenous data.
- Better hyperparameter tuning. This is true not only for the LSTM, but also for the ARIMAX model - due to performance issues on my machine, I was unable to generate anything more complicated than a basic ARIMAX model of order (1, 0, 1). One method to avoid this could have been to run the models on a cloud computing service such as AWS. Alternatively, I could have modified the data set to have fewer points (perhaps grouping the data by month).

- Testing using the method of training the LSTM using extrapolated points, as in section 3.6. While this did not show promising results on the simulated data, this could possibly have worked well on real-world data.

I was also interested in potentially working with symbolised time series data, using the method described by Elsworth and Güttel[6]. Some models were built using this method - however, useful predictions were not able to be obtained before the deadline for this thesis came due. Notably, accurately assessing the models proved difficult, as there was no clear way to compare the output to the input - since the length of each symbol varied, it was quite possible for one time series to be longer than another. When charting the models predictions as a whole, earlier errors would propagate into the future and would often cause the model to veer wildly off course. With more time, I feel I could have explored this avenue more thoroughly.

5.3 Future Work

One aspect of the modelling I did not investigate in the simulated data section were exogenous variables. A potential approach to this would be to generate other data sets that correlated with the Fourier series I produced to some degree, and include these in the model using the methods developed in chapter 4. Alternatively, data could be included from one or more of the separate sin waves used to generate the Fourier series.

Additionally, there is room for improvement in the models themselves, especially with regard to the hidden layer setup. My models only used LSTM layers and a single linear layer for output. PyTorch has many more layers built in, which could allow for better performing models to be built.

One potential use for this technology would be forecasting based off predicted exogenous variables. For example, if a strong model was built for predicting rice futures using oil prices as exogenous data, synthetic oil data could be created and passed into the model to see the potential effects on future rice prices. This could allow for the simulation of world events, such as price shocks or supply shortages, and the corresponding effect on the endogenous variable - the rice price - could be studied. With multiple exogenous variables, this could potentially allow for complex world events to be studied and prepared for ahead of time.

Chapter 6

Conclusion

6.1 Summary

The performance of the LSTM models on the synthetic data was excellent. The models were able to understand the underlying patterns well once properly configured and were able to extrapolate the pattern well into the future.

The performance on the real-world data was poorer than hoped, but the models still showed some signs of learning underlying patterns and predicting new prices. When incorporating the oil price variable, the models were able to better extrapolate the rice price data.

6.2 Project Status

There is plenty of further work to be done on this topic. While the success of the models on real-world data is limited, they performed well on the simulated data. More work would hopefully provide better results on real data. After this thesis is complete, I intend to continue working on some of the remaining issues with ALLFED. Notably, sourcing and incorporating new exogenous variables into the models is a primary goal of this project which was somewhat neglected during this thesis.

6.3 Closing Thoughts

Going into this project, I knew nothing about time series analysis. I had never heard of a SARIMAX model, or any of its constituent parts. My experience with LSTM models was limited to a brief exploration during a group project earlier in the year, in an entirely different context. In short, I had a lot to learn and implement in a short time.

I also struggled with motivation and depression during this semester, including the loss of a beloved family pet. This led to the time crunch which left the latter half of the project under-completed.

However, I am still proud of my ability to learn these new models and implement them in the time frame that I had. I am looking forward to doing more work on this topic with ALLFED.

Bibliography

[1] Said Fadlan Asnhari, PH Gunawan, and Yanti Rusmawati. Predicting staple food materials price using multivariables factors (regression and fourier models with arima). In *2019 7th International Conference on Information and Communication Technology (ICoICT)*, pages 1–5. IEEE, 2019.

[2] Noopur Ballal and Sri Khetwat Saritha. A study of deep learning in text analytics. In *Social Networking and Computational Intelligence*, pages 197–205. Springer, 2020.

[3] Emmanuel Dave, Albert Leonardo, Marethia Jeanice, and Novita Hanafiah. Forecasting indonesia exports using a hybrid model arima-lstm. *Procedia Computer Science*, 179:480–487, 2021.

[4] Saul Dobilas. Lstm recurrent neural networks - how to teach a network to remember the past, Mar 2022. https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e.

[5] Steven Elsworth and Stefan Güttel. Abba: adaptive brownian bridge-based symbolic aggregation of time series. *Data Mining and Knowledge Discovery*, 34(4):1175–1200, 2020.

[6] Steven Elsworth and Stefan Güttel. Time series forecasting using lstm networks: A symbolic approach. *arXiv preprint arXiv:2003.05672*, 2020.

[7] Tim Fist, Adewale A Adesanya, David Denkenberger, and Joshua M Pearce. Global distribution of forest classes and leaf biomass for use as alternative foods to minimize malnutrition. *World Food Policy*, 7(2):128–146, 2021.

[8] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[9] Michael Hinge, Edward Wilkinson, Sahil Shah, and Noah Wescombe. The Potential for a Novel Financial Product to Safeguard Communities Against Future Locust Outbreaks. Technical report, ALLFED, November 2021.

[10] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[11] Juyong Lee and Youngsang Cho. National-scale electricity peak load forecasting: Traditional, machine learning, or hybrid model? *Energy*, 239:122366, 2022.

[12] Juan B García Martínez, Kyle A Alvarado, Xenia Christodoulou, and David C Denkenberger. Chemical synthesis of food from co2 for space missions and food resilience. *Journal of CO2 Utilization*, 53:101726, 2021.

[13] Juan B García Martínez, Kyle A Alvarado, and David C Denkenberger. Synthetic fat from petroleum as a resilient food for global catastrophes: Preliminary techno-economic assessment and technology roadmap. *Chemical Engineering Research and Design*, 177:255–272, 2022.

[14] Charlie O'Neill. Pytorch lstms for time-series data, Jan 2022. urlhttps://towardsdatascience.com/pytorch-lstms-for-time-series-data-cd16190929d7.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[18] Sima Siami-Namini and Akbar Siami Namin. Forecasting economics and financial time series: ARIMA vs. LSTM. *CoRR*, abs/1803.06386, 2018.

[19] SM Ulyah, MFF Mardianto, et al. Comparing the performance of seasonal arimax model and nonparametric regression model in predicting claim reserve of education insurance. In *Journal of Physics: Conference Series*, volume 1397, page 012074. IOP Publishing, 2019.

[20] Teddy Mutugi Wanjuki, Adolphus Wagala, and Dennis K Muriithi. Forecasting commodity price index of food and beverages in kenya using seasonal autoregressive integrated moving average (sarima) models. *European Journal of Mathematics and Statistics*, 2(6):50–63, 2021.

[21] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.