Forecasting South Africa's Inflation Rate using Deep Neural Networks

By

Phage, Kabo.

Student Number:29079617

Supervisor: Prof R van Eyden

Department of Economics

UNIVERSITY OF PRETORIA

An MSc mini-dissertation submitted to the University of Pretoria following the requirements of MASTER OF ESCIENCE in the Faculty of Economic and Management Sciences

January 14, 2022

Declaration of own work

I, Kabo Phage, declare that this mini-dissertation is my own, unaided work. It is being submitted for Master of Science in the field of e-Science at the University of Pretoria. It has not been submitted for any degree or examination at any other university.

Kabo Phage, January 14, 2022

Acknowledgement

The support of the DSI-NICIS National e-Science Postgraduate Teaching and Training Platform (NEPTTP) towards this research is hereby acknowledged. I would also like to thank my supervisor, Prof R van Eyden, for her incredible expertise and guidance during the study. As well as the program manager, Ms. Casey Sparke, my mother, Daphne Phage and sister Oratilwe Phage for their patience, support and encouragement throughout my studies. Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to the NEPTTP.

Abstract

Inflation forecasting is crucial for efficient monetary policy and decision-making in an economy. This paper examines the feasibility of including deep neural networks in the macroeconomic forecasting toolbox for the South African economy. This study focuses on South Africa's annual headline inflation rate and applies two different deep neural network architectures for forecasting. The deep neural network's performance is compared to the autoregressive integrated moving average (ARIMA) benchmark, where root mean squared error (RMSE) is used as a performance measure. The results show that the multiple layer perceptron (MLP) outperformed the benchmark and its peer, the *convolutional* recurrent neural network model. Admittedly, the *convolutional* long-short term memory network (CNN-LSTM) is sensitive to architectural design, especially when the amount of training data is in short supply. In conclusion, the study finds that the ARIMA model predicts inflation inconsistently in the presence of endogenous and exogenous structural breaks in the time-series and consequently gives non-unique forecasts. The MLP becomes a viable addition to the macroeconomic forecasting toolbox in such a case.

The number of words in the mini-dissertation: 11625 words.

Contents

1 Introduction

Inflation forecasting is crucial for efficient monetary policy and decision-making in an economy. The South African Reserve Bank (SARB) has relied heavily on forecasts and every forecast is an essential input when deciding on the appropriate monetary policy stance while considering past and future price developments[1]. Private investors rely on future inflation predictions to adjust their asset portfolios. Firms use aggregate inflation level forecasts to adjust their prices to maximise their profits. Finally, fiscal authorities use the anticipated inflation rate to adjust social grant payments and income tax brackets.

Researchers agree that forecasting inflation accurately is challenging (Stock & Watson, 1998). Classical time-series methods can generate relatively accurate forecasts; however, such approaches traditionally take a deductive approach. The researcher starts with assumptions needed for generalising model specifications in such approaches. For example, these may include "assumptions of stationarity, mixing or asymptotic independence, normality, and linearity" (Brillinger, 2001). The decisions based on these assumptions often force the researcher to take strong positions on the nature of the data generating process.

Advances in computational learning have created an opportunity for data-driven tools, such as machine learning (ML), to overcome the deductive approaches in forecasting. This paper investigates the use of machine learning algorithms found in literature, specifically deep neural learning, commonly referred to as deep neural networks (DNN), in forecasting South Africa's inflation rate. The objective of this study is to contribute to the existing literature by following Makridakis et al. (2018). The paper focused on DNN architectures, specifically the Multi-layered Perceptron (MLP) and the Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM).

[1] Bernanke and Woodford (1997) also argue "that monetary policy is best thought of in terms of an optimal targeting rule in which policy reacts to past, present, and forecast values of target variables, including inflation. Accordingly, inflation forecasting is a key determinant of the setting of policy."

Multiple breakpoint tests suggest that South Africa's inflation rate has potentially four structural breaks over the period 1973 to 2003. In this study, the breaks were categorised using 2000 as the period when the inflation targeting framework was formalised in South Africa; that is a pre-2000 and post-2000 period. The post-2000 period presents a dataset that will be used to compare the benchmark's performance to the DNN without placing too much reliance on the researcher's experience with time-series forecasting but rather focusing on the application of deep learning to macroeconomic forecasting.

Chakraborty and Joseph (2017) showed that DNN algorithms need to be benchmarked to validate their relevance in macroeconomic forecasting problems. However, the term "benchmark" is loosely defined in two ways for this study. Firstly, it is defined in terms of a quantitative performance measure that investigates the accuracy of the forecast. It is worth noting that both the benchmark and DNN models were optimised using hyperparameter tuning. Hyperparameter tuning aims to find the optimal combination of parameters that minimises a predefined sigmoid loss function. Moreover, it is also used to prevent over-fitting. Consequently, the ease of hyperparameter tuning becomes the second benchmark definition. Therefore this paper assumes that the model with the lowest performance measure and ease of configuration would be optimal and qualifies to be added to the macro forecasting toolbox.

The rest of the paper is structured as follows. Section 2 provides a review of the background and empirical literature. Section 3 discusses the theoretical framework underpinning the traditional and deep learning models. The data, which includes stylised facts about the evolution of inflation, is discussed in section 4, whilst section 5 describes the methodology. Section 6 reports the results of the forecasting comparison and discusses the relevance of the deep neural network in inflation forecasting and potential future avenues of research. Finally, section 7 concludes.

2 Literature Review

The main objective of time-series forecasting can be stated as "identifying the nature of the phenomenon represented by the sequence of observations and predicting future values of the time series variable" (Mills, 2013). The ability to accurately predict the next time step is vital for decision-making. The application of deep learning in time-series forecasting is not complete without reviewing popular forecasting paradigms. Generally, studies on time-series forecasting are divided into two main categories; structural and non-structural approaches (Stock & Watson, 1998).

A theoretical model of the economy is formulated and estimated (or calibrated) before producing an empirical forecast in a structural approach to modelling and forecasting. In this way, economic theory guides the specification of the forecasting model (Almosova & Andresen, 2019). This forecasting approach can produce conditional forecasts, which are macroeconomic indicator predictions in response to a specific change in policy. "Dynamic stochastic general equilibrium (DSGE) models dominate the contemporary structural approach" (Makridakis et al., 2018). Although beyond the scope of this research, King and Srianantha kumar (2015) proved deep neural networks could be helpful "in approximating DSGE model solutions".

In contrast to structural models, non-structural approaches do not explicitly attempt to approximate some economic theory but rely on the statistical relationships of the data instead. Consequently, non-structural approaches tend to forego causal inference in the interest of predictive accuracy and are suitable for unconditional forecasting. Creel (2017) expanded their use by showing that simple linear differenced equations driven by random stochastic shocks called autoregressions can provide a convenient and robust framework for forecasting economic time-series. Autoregressive type models are popular because they deal with stationary and non-stationary data (Adhikari & Agrawal, 2013). In addition, their dominance over other forecasting models such as the Holt-Winters exponential smoothing and the Grey forecasting model grew in literature because of the

ease of application and the solid theoretical framework. However, some studies dispute their efficacy in forecasting inflation for longer time horizons. In such a case, Lim and Zohren (2021) found that neural networks were superior because they could model structural breaks without removing them from the data.

Makridakis et al. (2018) found that deep neural network models are not suitable for structural models but can substitute for extant non-structural approaches because of the following features. Firstly, deep neural networks are data-driven and self–adaptive, requiring no prior assumptions about data patterns or functional relationships. Secondly, they can generalise by learning from present data to make predictions using unseen data. Thirdly, they can approximate any continuous function and model non-linear relationships. The combination of these features presents deep learning as a promising forecasting tool. This research, therefore, investigates the feasibility of deep learning in forecasting South Africa's inflation rate using data from January 1960 to December 2020.

Deep learning neural networks is a term that was coined by Chollet and Allaire (2018) for its "directed graphical models which act as universal non-linear function approximations, optimised through back-propagation of errors". In ML literature (Zhang et al., 1998), deep learning is well-thought-out as its subset. Specifically, it works with multi-layered interconnected artificial neural networks (ANN) designed to imitate how the human brain functions in learning and making predictions from large or small volumes of data. Several papers have investigated the feasibility of using deep neural networks in time-series forecasting.

Chakraborty and Joseph (2017) introduced deep learning as central banking and policy analysis framework. The study aimed to forecast the United Kingdom's inflation over a medium-term horizon of two years by using quarterly data from 1998 to 2015. Their results show that deep neural networks outperformed traditional non-structural linear forecasting approaches such as autoregressive (AR) models with lag orders specified between one and p. Furthermore, their results confirm that deep neural networks learned new relations from complex non-linear data structures. In that way, the accuracy of the forecasts did not deteriorate given the structural break presented by the 2007/8 global financial crisis.

In support, Goodfellow et al. (2016a) argues that including such models in the forecasting toolbox would be beneficial, especially in periods of structural changes. The paper subsequently investigated using a single hidden-layer feed forward ANN[2] to forecast the Polish headline inflation in periods of persistently low inflation. However, in recent times, Martin (2019) found that a single hidden layer ANN was outperformed by traditional statistical approaches, specifically the Autoregressive Integrated Moving Average (ARIMA) model. The authors argue that ML theoreticians should improve their models by hyperparameter tuning before assessing their viability as forecasting tools.

It is worth noting that a single hidden layered ANN is generally not thought of as DNN because of its architectural simplicity. Also, ANNs were criticised by Cook and Hall (2017) for their inability to capture sequential information in the initial input data. In other words, the models' ability to make predictions using time-series data is negatively affected because the model fails to generalise[3]. Researchers, therefore, conclude that ANNs are most useful in classification problems and not a regression-type predictive modelling problem.

Expounding on the idea of using ANN for time-series forecasting, Makridakis et al. (2018) applied four different DNN in forecasting the monthly United States unemployment rate. The four DNN were built on a different architecture having more than one hidden layer and configured to handle time-series data. The authors advocated that using deep neural networks improved the accuracy of the forecasts when applied to limited data for short time horizons forecasts. The authors concluded that DNN performed better forecasting than the standard single hidden layer ANN. Although their paper focused on the unemployment rate, their models can extend to other macroeconomic indicators

Considering the growing application of DNN in macroeconomic indicator forecasting, this research aims to contribute to the existing literature in two ways. Firstly, this research follows (Makridakis et

[2]Commonly referred to as Single-layer Perception.

[3]In ML literature if a model is trained too well on the training data, it will be unable to generalise.

al., 2018) by focusing on DNN architectures, specifically the Multi-layered Perceptron (MLP) and the Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM). To date, these approaches are not used yet in literature to forecast South Africa's inflation. Martin (2019) explored the use of ANN to predict South Africa's gross domestic product and found that it performed better than traditional approaches. Van Heerden et al. (2018) found that non-linear autoregressive neural networks were superior to the seasonally adjusted ARIMA in forecasting South Africa's inflation. Although the authors did not focus on the DNN, their neural network application suggests that the deep learning networks could also be superior.

Secondly, to assess if DNN qualifies to be included in the macroeconomic forecasting toolbox. This research, therefore, investigates the accuracy of the DNN. At the outset, a quantitative measure of the neural network's performance is compared to the ARIMA benchmark. The main reason for choosing the benchmark model is because it is widely used in the empirical literature. Moreover, using a performance measure in forecasting problems has proven sound by Cook and Hall (2017).

Gil-Alana (2011) showed using data from January 1970 to December 2008 that South Africa's inflation rate is a "covariance stationary process with an order of integration ranging in the interval (0, 0.5)". The order of integration (I) is a contentious issue in the literature. Some authors (Backus & Zin, 1993) claim that inflation is stationary I(0) because prices are generally expressed in logarithms in forecasting problems. While some authors (Gil-Alaña et al., 2011) argue that the inflation series is a non-stationary I(1) process for the same reason. Therefore, this study first assumes that the inflation series is a non-stationary I(1) process because of the presence of structural breaks

The intuition is that structural breaks render the past relationship between observations a poor guide to the future resulting in forecasting failures (Burridge et al., 2005). Consequently, quantitatively comparing the benchmark's performance against the DNN models is a futile but necessary exercise. It is pointless because stationarity is not a requirement for DNN models because they are self–adaptive, meaning they do not rely upon past observations to make a forecast. It is, however, necessary because it will expose the weakness of the benchmark model, particularly its inconsistencies when deciding on the order of integration and when testing for optimal lag structures. This

research, therefore proposes that DNN provide an alternative approach to forecasting inflation. Finally, the implementation of the DNN models will follow those described in Brownlee (2020).

3 Theoretical Framework

This section provides the theoretical conceptualisation of the forecasting models employed by this study. First, the description of the benchmark model is given in section 3.1, followed by the description of the deep learning architectures, where deep-learning is configured to work with time-series data, presented in section 3.2.

3.1 Benchmark Model

3.1.1 The ARIMA Framework

This study considers the ARIMA framework. According to Van Heerden et al. (2018), it is the most "widely used forecasting approach for univariate time-series" and the most common benchmark for deep neural networks found in the literature.

The framework can be applied to financial or economic time-series through the combination of the autoregressive (AR) and moving average (MA) models with a differencing factor (I). Equation 3.1 is the generalised ARIMA model for a given stochastic process y_t:

$$\triangle^d y_t = \delta + \sum_{i=1}^{p} \phi_i \triangle^d y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}, \tag{3.1}$$

where d is the number of times the series needs to be differenced before becoming stationary, known as the order of integration, the intercept term is δ, p is the number of past values or equivalently lag observations and ϕ's are the coefficients related to them. The contemporaneous error term is ε_t. Moreover, it is assumed to be independent and identically distributed (*i.i.d.*). Finally, the moving average lag-order and its coefficients are q and θ, respectively.

The ARIMA framework can decompose to its component models through specific choices of p and

q. For a given d, when $p > 0$ and $q = 0$, the series is modelled using past observations of itself, yielding the autoregressive (AR) model. Likewise, when $p = 0$ and $q > 0$, it reduces to the moving average (MA) model. In this case, the series is modelled using the error term based on past observations.

In this study, the optimal parameters p, d and q are determined using diagnostic plots such as the autocorrelation function (ACF) and partial autocorrelation function (PACF). The diagnostic plots are used in conjunction with heuristic rules. Hyndman and Athanasopoulos (2018) proposed the Akaike information criteria (AIC) as a rule that focuses on predictions. It is a procedure that finds the combinations of the parameters by minimising the AIC defined by equation 3.2 below:

$$AIC = -2log(L) - 2(p+q+k+1), \tag{3.2}$$

where L is the likelihood estimate, and k is an indicator taking the value one if the intercept coefficient takes a value greater than 0. The AIC gives a "larger penalty to a more complex framework and is often preferred for forecasting models" (Hyndman & Athanasopoulos, 2018). Whereas the Bayesian information criterion (BIC) defined in equation 3.3 is often preferred in literature and is defined as:

$$BIC = -2log(L) + klog(T) \tag{3.3}$$

where T is the number of observations in the time-series. The optimal model would be the one that minimises the AIC and BIC given by equations 3.2 and 3.3.

3.2 Deep Learning Algorithms

The basic architecture upon which DNN is built is the perceptron (Lim & Zohren, 2021). A perceptron is a neural network unit (an artificial neuron) capable of making input data computations. Figure 3.1 shows a perceptron's graphical depiction where $\mathbf{x}$ is a vector of macroeconomic variables and $\mathbf{w}$ is the corresponding vector of weights that gets updated through the training and learning process. The vector and weights are linearly combined and called the *net input function*. The net input function is an *activation function* used to forecast the macroeconomic indicator at some prediction horizon called the output.

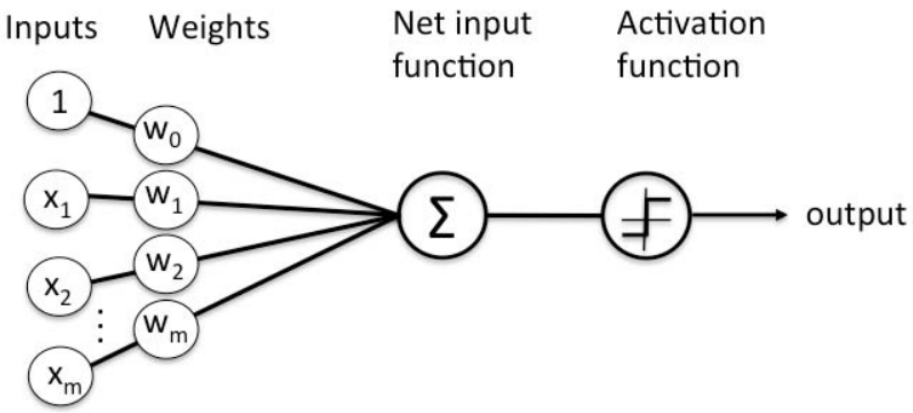


Figure 3.1: An image of a perceptron adopted from Rosenblatt (1958)

3.2.1 Multi-Layer Perceptron

MLP model is one type of supervised feedforward[4] neural network. It has three layers: input layer with N input nodes, output layer with m output nodes, and one or more hidden layers. Figure 3.2 shows the graphical architecture:

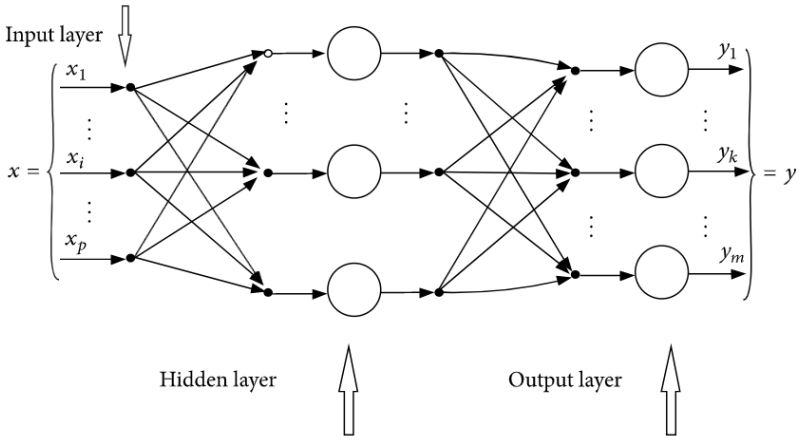


Figure 3.2: General architecture of multi-layer perceptron neural network model adopted from Azorin-Molina et al. (2017)

Equation 3.4 gives the mathematical function for a single timestamp[5] that is one pattern $(x^{(p)}, y^{(p)})$

[4] In a feedforward structure, information flows in one direction though the network from inputs to outputs.

[5] Technically,the model will view each time period as a separate feature instead of separate time periods.

where $p \in (1, \ldots\ldots P)$:

$$\hat{\mathbf{y}}_{\mathbf{k}}^{(\mathbf{p})} = f_m(\theta_k + \sum_{j=1}^{L} w_{kj} \cdot f_L(\theta_j + \sum_{i=1}^{N} w_{ji} x_i^{(p)})) \tag{3.4}$$

where p is the number of observations (timestamps or patterns), $\hat{y}_k^{(p)}$ is the output value of the output node k of the pattern P and $x_i^{(p)}$ is the input value of the i input node. The threshold of output node, hidden node j are respectively given by θ_k and θ_j; w_{kj} is the weight from hidden node j to output node k and w_{ji} is the weight from input node i to hidden node j.

The weights in this network are used to scale the individual inputs, representing the strength of the connection between nodes. For example, suppose the weight between the first and second node is more significant in magnitude. In that case, the first neuron has a more substantial influence over the second. Therefore, it decides how much information the input will have on the output. Finally, f_m and f_L are sigmoid activation functions of the output and hidden layers, respectively. The purpose of the activation functions is to help the deep neural network learn complex linear or non-linear patterns in the data.

The parameters of the MLP are given by the number of inputs used for the model, the number of hidden neurons provided by nodes, the learning rate, which is the number of training epochs, and the activation function. The grid search algorithm will be used to find the optimal parameters, according to Brownlee (2020). Section 5 describes the parameter choices and tuning. Once a neural network is trained, it can make predictions by providing input to the network. After that, a forward-pass is performed to generate an output that is used as a prediction.

3.2.2 Convolutional Neural Network-Long Short-Term Memory

The model combines the convolutional neural networks (CNN) and the long short-term memory (LSTM). The CNN-LSTM model has been used to predict stock market prices accurately (Mehtab & Sen, 2020). This forecasting method provides an alternative way that can be used to forecast price indexes. In addition, it provides practical experience for researchers to study financial or economic time-series data.

Firstly, CNN is a feedforward neural network commonly used for image processing and natural language processing. Cook and Hall (2017) found that it can be effectively applied to time-series forecasting problems. The model comprises two parts: the convolutional layer and the pooling layer. A convolution is a linear operation that involves multiplying a set of weights with the input data, similar to the MLP. However, the weight sharing found in the CNN reduces the number of parameters, therefore, improving the efficiency of the learning model.

The convolutional layer is the fundamental building block of a CNN, and it is where the most computation occurs. Each layer consists of learnable filters (or kernels) known as a *multitude of convolution kernels*, and its calculation formula is given by equation 3.5 below.

$$l_t = tanh(x_t * k_t + b_t), \tag{3.5}$$

The purpose of the kernel is to slide across the input data to give output weighted sums of a given number of the inputs. These outputs are called *feature maps*. The width of the kernel decides how many of the inputs are extracted at each timestamp. It is known as the *kernel size*, an additional hyperparameter of the network.

After the convolution operation, the features or patterns of the data are extracted. After that, the pooling layer is added to reduce the feature dimension or the number of parameters, where l_t is the output value after convolution, $tanh$ is the hyperbolic tangent activation function, x_t is the input vector. The weight of the convolution kernel is given by k_t, and b_t is its bias term.

Finally, CNN is characterised by its ability to pay attention to the most apparent data features or patterns in a time-series. Given that a univariate time-series has only one spatial dimension, this study considers a one-dimensional CNN model that is a one-dimensional kernel that learns by mapping an input sequence of data to an output value. Figure 3.3 gives the basic architecture of the CNN used in this study.

The LSTM component of the CNN-LSTM is a recurrent neural network capable of learning order dependence in sequence forecasting problems. In the LSTM network, information flows through a

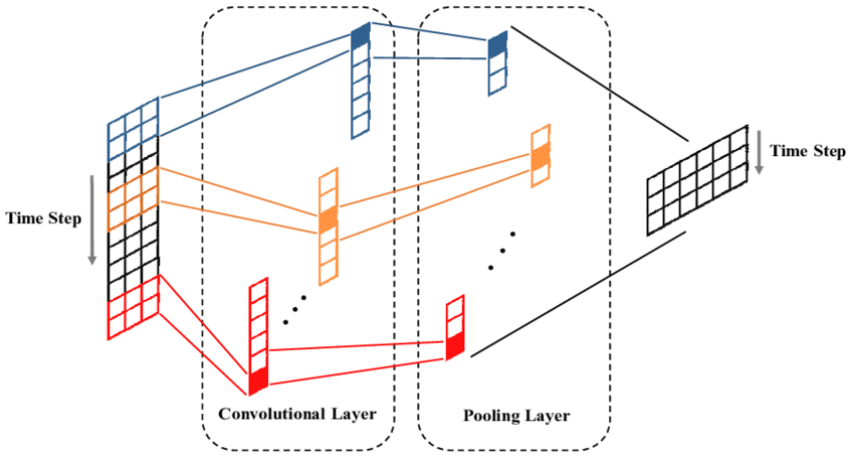


Figure 3.3: The basic structure of the one-dimensional CNN adopted from Yan et al. (2018)

mechanism known as a cell state. The cell state is similar to a conveyor belt running straight down the entire chain from c_{t-1} to c_t; as shown in figure 3.4, with only some minor linear interactions. The network can remove or add information to the cell state through regulated structures called gates.

Following figure 3.4, the first step is deciding what information must be omitted from the cell state. The forget gate layer is a sigmoid layer responsible for this decision. It does this by considering h_{t-1} and x_t to output a number between zero and one for each number in the cell state, c_{t-1}. Where one represents "keep information" and zero "discard it". The calculation of the forget gate layer is given by equation 3.6 below:

$$f_t = \sigma(w_t \cdot [h_{t-1}, x_t] + b_t), \tag{3.6}$$

In the second step, the network decides what new information it will store in the cell state. This step involves two parts. Firstly, a sigmoid input gate layer determines which values to update by creating a *tanh* layer containing new candidate values, $\tilde{c}_t$. These candidate values are added to the state. The final part involves combining the input gate values with the candidate values. The calculation of the input gate and candidate values is given by equations 3.7 and 3.8, respectively:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i), \tag{3.7}$$

$$\tilde{c}_t = tanh(w_c[h_{t-1}, x_t] + b_c), \tag{3.8}$$

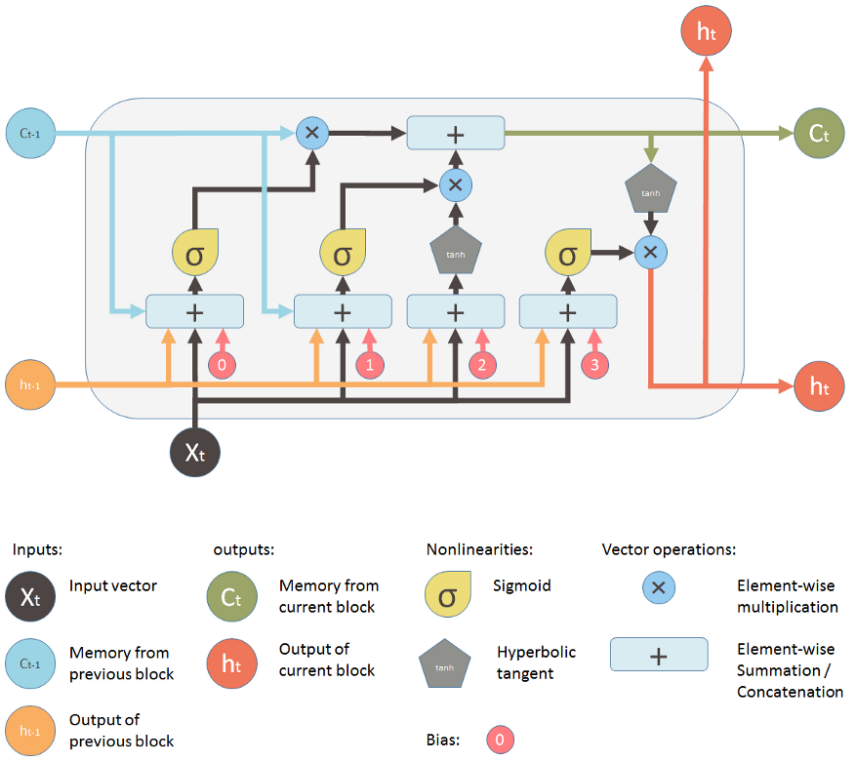


Figure 3.4: LSTM memory cell structure adopted from Yan et al. (2018)

The third steps involve updating the old cell state, c_{t-1} to a new cell state, c_t. The update is done by simply multiplying the old state by f_t. Recall, the old state is the state that forgets omits information (values) that should not enter the long-term memory and adds new candidate values, which are scaled by how much the update from the previous step was that is, $i_t * \tilde{c}_t$. Equation 3.9 summaries this step:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{3.9}$$

Finally, the network needs to decide what it is going to output. Equation 3.10 gives a sigmoid output layer and decides which part of the cell state it will output based on its filtered version:

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_0) \tag{3.10}$$

Then, the cell state passes through the *tanh* function, producing an output between negative one and positive one. This output is then multiplied by the same sigmoid output gate so that it only outputs the parts it decides to make predictions. Equation 5.1 gives the calculation for this final step:

$$h_t = o_t * tanh(c_t) \tag{3.11}$$

LSTMs offer native support for sequenced observations given the memory state cell operations. It does this by using a learnt mapping function consisting of inputs observed over time to produce an output. Contrasting the CNN that reads through the entire observed input vector, the LSTM model can read one timestamp of the sequenced observations and construct an internal state exemplification used as a *learned context* from which predictions are made.

Taking advantage of the characteristics of both the CNN and the LSTM mentioned above, this study uses the CNN-LSTM as a DNN forecasting model. The model architecture framework is shown in figure 3.5. Sequenced raw data is the input layer. After that, the model structure involves a one-dimensional convolution layer for feature extractions and parameter reductions on input data, coupled with a single hidden LSTM layer to support sequence prediction. Furthermore, consistent with the MPL model described above, the optimal parameters of the CNN-LSTM model will be found using a grid search algorithm. The parameters are presented in Section 5.

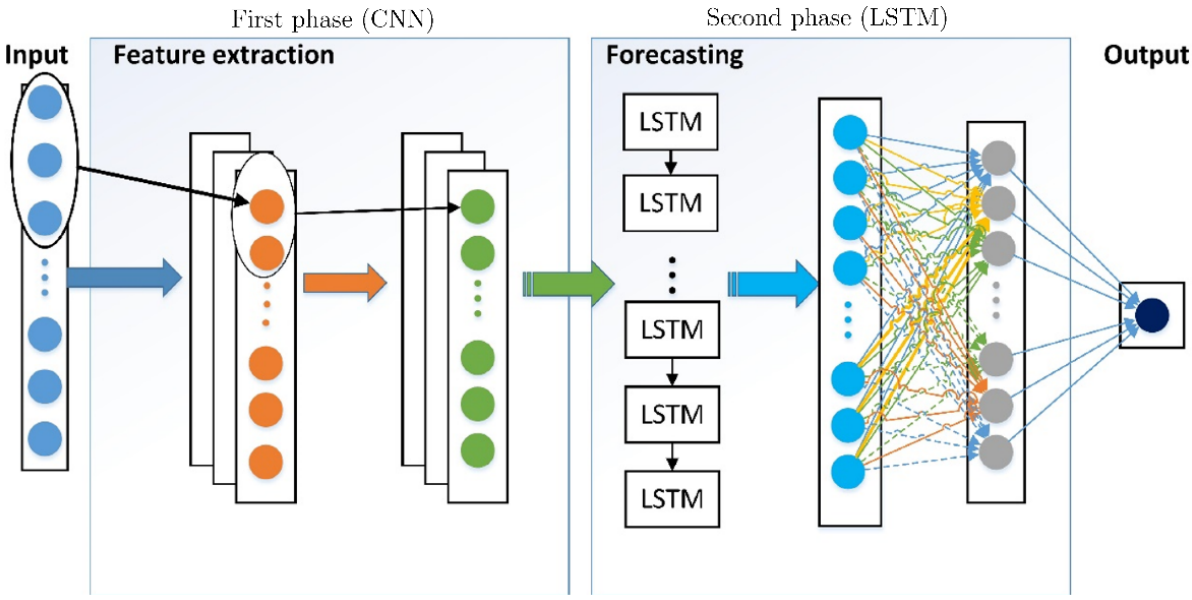


Figure 3.5: CNN-LSTM model framework adopted from Yan et al. (2018)

4 Stylised Facts and Data

The South African inflation rate is used to train each model discussed above. Specifically, monthly headline inflation from January 1960 to December 2020 was sourced from the International Monetary Fund's International Financial Statistics database.

Over the years, inflation developments in South African have primarily been driven by two factors, namely global price pressures as well as domestic economic conditions and domestic monetary policy. During the 1970s, inflation rose as a consequence of oil price shocks. After that, like most countries, South Africa's major trading partners experienced a steady decline in inflation, spreading over well into the 1980s. However, during the early 1990s, inflation in South Africa rose and remained high despite the disinflation experienced by its trading partners. The persistently high inflation resulted from a weaker monetary policy stance coupled with growth in broad money supply which was substantially higher in South Africa than in its trading partners.

South Africa, therefore, needed a monetary policy with relatively low-interest rates that would stimulate economic activity in the short term. Although this was possible, the country experienced economic sanctions and had a battery of exchange controls that reduced the mobility of international capital at the expense of higher inflation. From the early to mid-1990s, sanctions were lifted, and the gradual liberalisation of controls was associated with an increase in real interest rates followed by a decline in inflation. Monetary policy, therefore, slowly converged towards an informal eclectic inflation targeting approach.

It was only in 2000 that the inflation targeting was formalised. The SARB uses monetary policy tools such as controlling short-term interest rates to keep headline inflation within the 3-6% target range within this framework. From 2002 to 2003 and again in 2006 to 2008, there was a significant divergence from the target range. The SARB communicated that these deviations resulted from

exogenous shocks and introduced the flexibility justification to the inflation-targeting framework.

Since 2017, the central bank has communicated that it would generally like to see inflation close to the midpoint of the target range, notwithstanding its continuous and flexible nature. At this period and beyond that, the central bank accepted deviations from the target range conditionally. The condition was that inflation should return to its target range over a reasonable period usually specified as a year but no longer than two years. Finally, inflation targeting is always forward-looking, making inflation forecasting an important policy input.

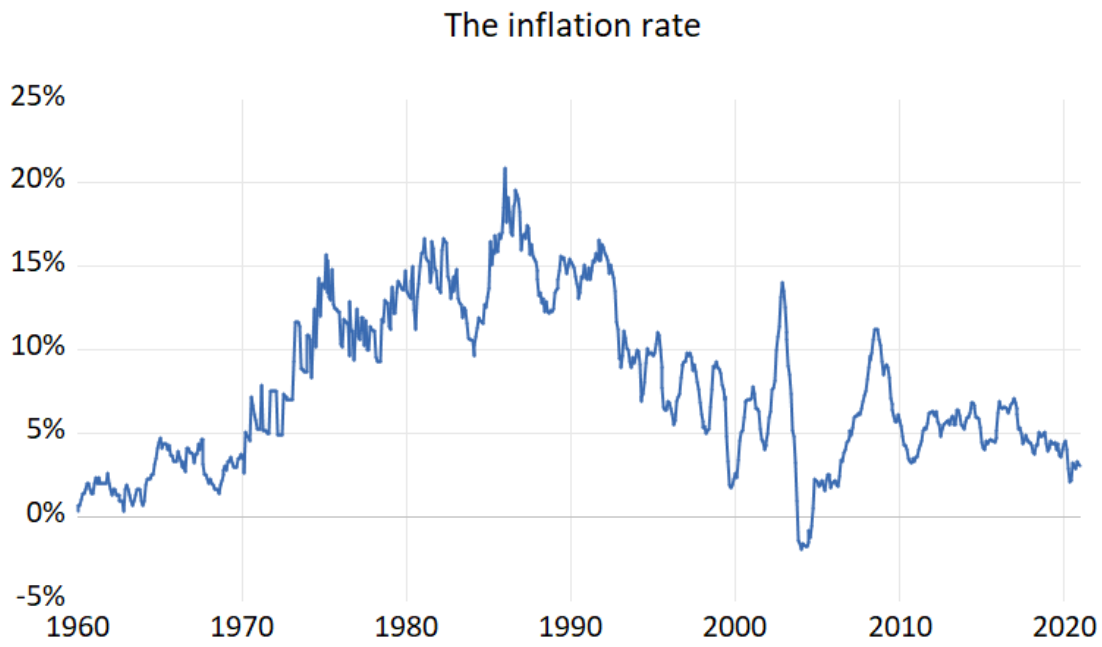


Figure 4.1: South African monthly inflation rate from 1960 to 2020

Figure 4.1 and the developments mentioned above show that South Africa's inflation has changed over time. Between 1970 and 1984, the series seems to have an increasing trend with relatively moderate peaks and troughs. A sudden decreasing trend with higher relative peaks and troughs started noticeably around 1982, which continued well into a year after formalising the inflation target framework in 2000. These changes are not compatible with a stationary process for the

1970-2000 period. Based on the plot, it can be concluded that the inflation series exhibit structural breaks.

A multiple breakpoint test was conducted following Yao (1988) to find the structural breaks statistically. The author showed that "the number of breaks that minimises the Schwarz information criterion is a consistent estimator of the actual number of breaks in a breaking mean model under relatively strong conditions". More generally, Liu et al. (1997) recommended using the modified Schwarz criterion LWZ to find the number of breaks in a regression structure. Table 4.1 shows that the Schwarz and the LWZ information criteria select four breaks identified by the following dates; March 1973, April 1982, November 1992, and July 2003. These breaks correspond to the above-mentioned stylised facts. Consequently, they can be categorised into pre- and post-formalisation of the inflation-targeting framework in SA. That is, a pre-2000 and post-2000 policy regime change. The latter includes the break identified in July 2003.

Breaks number	Schwarz Criterion	LWZ Criterion
0	3.156	3.169
1	2.884	2.926
2	1.874	1.943
3	1.738	1.835
4*	1.676*	1.800*
5	1.747	1.899

Table 4.1: Results from multiple breakpoint testing.

Deep learning neural networks do not require the stationarity assumption because of their self-regularising property. Moreover, they are flexible enough to handle stationary or non-stationary data even in the presence or absence of structural breaks, unlike the ARIMA model. Boot and Pick (2020) showed that the presence of structural breaks reduces the ARIMA model's forecasting accuracy. Consequently, Elliott and Müller (2014) suggested that to deal with structural breaks when the break-time is known, a common approach is estimating the ARIMA model before and

after the break. For the deep learning neural networks, this approach results in fewer total training observations in both samples, which deteriorates the overall accuracy of the forecast.

However, Pesaran et al. (2011) showed that ignoring breaks with known dates, as an approach, may lead to more accurate forecasts when evaluated using the root mean square error (RMSE) for both the ARIMA and deep learning models. So drawing from this finding, this research only estimates the models using the post-2000 period because the breaks identified in the pre-2000 period result from global price pressures, such as exogenous oil shocks. In addition, the year 2000 marks the period when the inflation targeting framework was formalised in South Africa. As a result, the post-2000 dataset remains relevant for the current monetary policy regime. The paper (Pesaran et al., 2011) also showed that the relevancy of exogenous shocks in forecasting is infrequent because the economy's structure is assumed to be unchanged over short forest horizons.

Box and Tiao (1975) recommended that at least 50 but preferably more than 100 observations be used for time-series forecasting employing monthly data. The post-2000 series contains 252 observations. Unit root testing suggest that the inflation series is non-stationary, I(1) (see Appendix A). Furthermore, to find the best set of parameters for the ARIMA model using the diagnostic plots are the starting point, the grid-search algorithm will optimize the criteria defined in section 3 and specified in the following sections.

5 Research Method

Time-series forecasting can also be framed as a supervised deep-learning problem because of the process an algorithm takes to learn from the training dataset. This process can be thought of as a "teacher supervising the learning" (Bontempi et al., 2013). The supervisor is assumed to know the “correct answers” made by the algorithm as it iteratively makes predictions on the training data. The learning process stops when the algorithm has realised a satisfactory level of the performance measure.

This study employs a naïve forecast as a supervised learning strategy that directly uses previous observations. It is often called the *persistence forecast* because the past observation is insistent. When the last time steps predict the next step, literature (Bontempi et al., 2013) commonly refers to this as the sliding window method, also called a lag or lag method in statistics and time-series analysis. The last time steps are called window width or lag length, which form the foundation of how any time-series dataset is transformed into a supervised learning problem. The time-series dataset is then converted into a data frame where the number of observations is used instead of the dates.

The following section establishes the evaluation methodology and configures the benchmark and deep learning grid search algorithm.

5.1 Model Performance Evaluation

There are many ways to evaluate the performance of a simple forecasting model. In supervised learning, a common approach is to split the data into training, and testing sets where model parameters are established on the training set and the performance is calculated on the test set. Cross-validation is a statistical method often used to estimate ML models’ performance (or accuracy)

on test sets. It is also a popular technique used to protect against overfitting, particularly in cases where the amount of data is limited, as in this research. The method is also used for hyperparameter tuning, which helps detect which parameters will result in the lowest test error.

In cross-validation, the procedure is to make a fixed number of folds (or partitions) of the data and run the analysis on each fold. The final step involves taking the average of the overall error estimates, which serves as a performance metric for the forecasting model. For example, in k-fold cross-validation, the dataset is randomly split into k equal-sized folds. Then the model is built iteratively on each of the $(k-1)$ folds and tested for effectiveness on the k^{th} fold. The average of the k^{th} captures the accuracy of the forecasting model and is called the cross-validation accuracy, which serves as the performance metric. According to Elhadad (2021), this method generally results in a less biased model than other approaches. Conversely, James et al. (2013) found that because the training algorithm has to iteratively run from scratch k times, it is computationally expensive and disruptive for timestamped data.

In the case of time-series data, cross-validation is not unimportant. Unlike k-fold cross-validation, a random sample cannot be chosen and assigned to either the test or train set because a model trained on future data may lead to look-ahead bias. "There is a temporal dependency between observations that must be preserved during testing" (Choudhary & Haider, 2008). For this reason, the method that can be used for cross-validating a supervised time-series forecasting model is walk-forward validation, commonly referred to as the rolling window analysis or rolling cross-validation.

As a starting point similar to cross-validation, the dataset must be split into train and test sets. In this way, the training set will train the model while predictions are made and evaluated using the test set. Depending on the number of observations, arbitrary splitting ratios of train:60%, test:40%; train:70%, test:30%, or train:90%, test:10% are often utilised by ML practitioners. Given that the data used in this study is monthly, the intention is to reserve the last three years of data for forecast evaluation. In other words, the last 36 observations will be used as the test data, and the resulting split will be approximately 85% for the training set and 14% for the test. The graphical presentation of the split dataset for the inflation series is shown in Appendix B.

The next step is the walk-forward validation approach implemented in Python[6], according to Brownlee (2020). However, the procedure is highlighted in the following steps. Firstly, the rolling window of size $m = 12$ is chosen because each window constitutes one year for monthly data. The rolling window is defined as the number of consecutive observations per rolling window. Secondly, given the forecasting strategy discussed in section 5, a one-step forecasting horizon is chosen, that is, a forecast horizon where $h = 1$ is chosen. Thirdly, because the number of increments between successive rolling windows is one period, the entire data set is partitioned into $N = T - m + 1$ subsamples, where $T = 36$ is the number of observations in the test set. The first rolling window contains observations for the first period through m, and the second includes observations for the second period through $m + 1$ and continues likewise. Figure 5.1 illustrates the partitions.

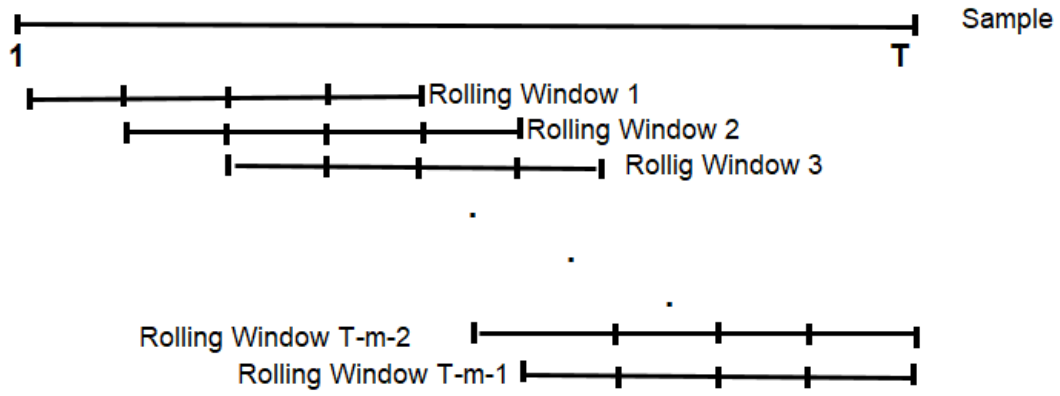


Figure 5.1: This figure illustrates how the test set is partitioned into rolling windows. The figure is adopted from MATLAB (2010) .

As the fourth step, for each rolling window subsamples, the following is done:

(a) the model is estimated;

(b) the one-step-ahead forecasts are estimated;

(c) the error for each forecast is computed,that is $e_{t+h,j} = y_{t+h,j} - \hat{y}_{t+h,j}$, where:

i. j indicates the number of each window;

ii. y is the response;

[6]The python code is given in Appendix B.

iii. $\hat{y}_{t+h,j}$ is the one-step-ahead forecast of rolling window subsample;

Finally, step fifth step involves the computation of the RMSE defined by equation 5.1 as follows:

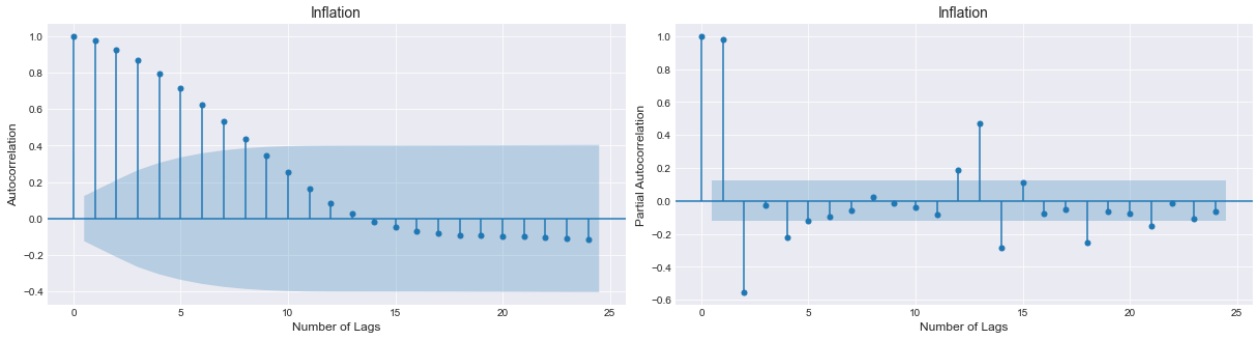
$$RMSE_j = \sqrt{\frac{1}{n}\sum_{n=1}^{N}\left(y_{t+h,j} - \hat{y}_{t+h,j}\right)^2} \tag{5.1}$$

The RMSE expresses the model forecasting error in the same units as inflation. Therefore, the metric will be reported in terms of basis point error in inflation, which gives more weight to large errors. The RMSE of the models will be compared, and the model with the lowest set of RMSEs is presumed to have the best predictive performance.

5.2 Benchmark forecasting models

5.2.1 ARMA model based on grid search

Diagnostic plots of the post-2000 time-series can be used along with empirical rules to determine the parameters of the ARIMA model. Figure 5.2 contains the plots. By looking at the series' ACF and PACF plots, the order of the AR and MA terms can be tentatively specified.



(a) The ACF (b) The PACF

Figure 5.2: The ACF and PACF plots

The ACF in figure 5.2(a) shows the correlation coefficients' bar chart between a series and lags itself. For example, the correlation at the first lag is measured as the correlation between values of the series at time t with all of the values measured at time $t-1$. The correlation at the second lag

measures the correlation between values at time t with all values measured at time $t-2$. The process continued until the maximum number of lags was specified. According to Brownlee (2020), the maximum number of lags can be set using $10\log 10(N)$, where N equals the length of the series and $\log 10$ is the base10 logarithm function. So for this study, the calculation results in 24 lags.

The PACF in figure 5.2(b) plots the partial correlation between the series and the lag itself, similar to the ACF. The only difference is that it controls any correlation between observations at shorter lag lengths. At the first lag, the ACF and PACF are the same. However, for the second lag, the PACF measures the correlation between observations at time t with those at time $t-2$ after controlling for the correlation between observations at time t with those at time $t-1$.

The patterns identified by the plots suggest that the series might have an AR(1) or AR(2) autoregressive process and an MA(1) or MA(2) moving average process. However, because the ARIMA model for forecasting can be tricky to configure, this research uses ML algorithms. In ML, the process of automating the process of training and evaluating the ARIMA model on a different combination of parameters is called grid search or model tuning (Brownlee, 2020). The process is divided into two parts; (1) evaluate an ARIMA model and (2) evaluate the set of ARIMA parameters.

In Python, the $evaluate\_arima\_model()$ function takes the series as input and a tuple with p, d, and q parameters to evaluate the model. The series is firstly split into training and test sets as specified in section 5.1. As the second step, each time step of the test set is iterated. Only one iteration gives the model that is used to make predictions on new data. This iterative approach allows a new ARIMA model to be trained each time step. The projections made for each iteration are stored in a list so that at the end of the test set, all predictions can be compared to the list of expected values and a root mean square error is calculated.

For this study, a suite of lag values,$p = [1,2,3]$ as well as difference iterations (d) and residual error lag values (q) ranging from 0 to 3 is specified. A grid of parameters to iterate is determined to evaluate the set of parameters. A model is created for each parameter, and its performance is eval-

uated by calling the *evaluate_arima_model*() function described above. The function must keep track of the lowest RMSE observed and the configuration that caused it. Moreover, the function is implemented in four loops, and floating-point values ensure that the ARIMA procedure does not fail, yielding the best configuration to be ARIMA(1,1,1) which contradicts the step-wise search results found in AppendixC. Section 6 will deliberate the implications of these contradictions and what it means for this study.

5.3 MLP forecasting architecture

A simple MLP is used for forecasting by taking "lag observations, using them as input features and predicting one-time steps from those observations" (Brownlee, 2020). This forecasting approach is precisely framing the supervised learning problem defined in section 5 above. The training set is a list of samples, where each sample has many observations from months before the time being forecasted, and the forecast is the next month in the sequence. The model generalises over the samples and makes predictions on unseen data.

In Python, the Keras deep learning library will implement the MLP model. The model will have a single input layer specified using the *input_dim*() argument and a single hidden layer with some number of nodes, and then a single output layer. The rectified linear activation function 'ReLu' will be used on the hidden layer because it has been proven to perform well, according to Brownlee (2020). In addition, a linear activation function will be used on the output layer because this study predicts a continuous value. The mean absolute error is set as the network's loss function[7], and finally, the efficient ADAM optimiser is set to train the model. According to Kingma and Ba (2017), the ADAM optimiser method is "computationally efficient, has little memory requirement and invariant to diagonal rescaling of gradients".

The $model\_fit()$ function adopted from (Brownlee, 2020) will fit the MLP model on the training set. The function expects the configuration to be a list with the following hyperparameters:

[7]The purpose of loss functions is to compute the quantity that a model should seek to minimise during training (Brownlee, 2020).

- n_input: The number of lag observations to use as input to the model;
- n_nodes: The number of nodes to use in the hidden layer;
- n_epochs: The number of times to expose the model to the whole training dataset;
- n_batch: The number of samples within an epoch after the weights are updated;
- n_diff: The difference order or 0 if series is not differenced.

"Neural network models are stochastic" (Brownlee, 2020). The stochastic nature means that the model will have different performance measures even if it is estimated using the same training set and configurations. Consequently, different internal sets of weight are produced each time the model is trained. So to overcome this, the model configurations will be evaluated multiple times via walk-forward validation. The repeat $repeat\_evaluates()$ function implements this, allowing the number of repetitions to be set to 10 as an optional parameter, and returns the mean RMSE score from all repeats.

Finally, the $grid\_search()$ function takes the dataset, a list of configurations to search, and the test set to perform the search. Once the root mean square error score is calculated for each configuration, the list is sorted in ascending order so that the best score is listed first. For this study, an optimal model configuration was found to be $[12, 100, 100, 1, 1]$, which can be interpreted as:

- n_input: 12;
- n_nodes: 100;
- n_epochs: 100;
- n_batch: 1;
- n_diff : 1

5.4 CNN-LSTM forecasting architecture

The gird search parameter setting of the CNN-LSTM for this study is shown in table 5.1

Parameters	Value
Convolution layer filters	3
Convolution layer kernel_size	64
Convolution layer activation function	*ReLu*
Number of hidden units in LSTM layer	100
LSTM layer activation function	*ReLu*
Time_step	12
Batch_size	100
Learning rate	0.001
Optimiser	ADAM
Loss function	RMSE
Epochs	100

Table 5.1: Parameter setting of CNN-LSTM.

According to grid search, the specific model is configured as follows: the input training set data is a one-dimensional data vector $(None, 12, 1)$, in which 12 is the size of the time_step with one feature of the input dimension. Firstly, the dataset enters the one-dimensional convolution layer for extract feature extraction and obtains a three-dimensional output vector $(None, 12, 3)$, in which 3 is the size of the convolution layer filters. Secondly, the vector enters the pooling layer, and a three-dimensional output vector $(None, 12, 3)$ is subsequently obtained. Finally, the output vector enters the LSTM layer for training. After training, the output data $(None, 100)$ enters another complete connection layer to derive the prediction, which is the output. The number of hidden units in the LSTM layer is 100. The configured CNN-LSTM model architecture is shown in figure 5.3

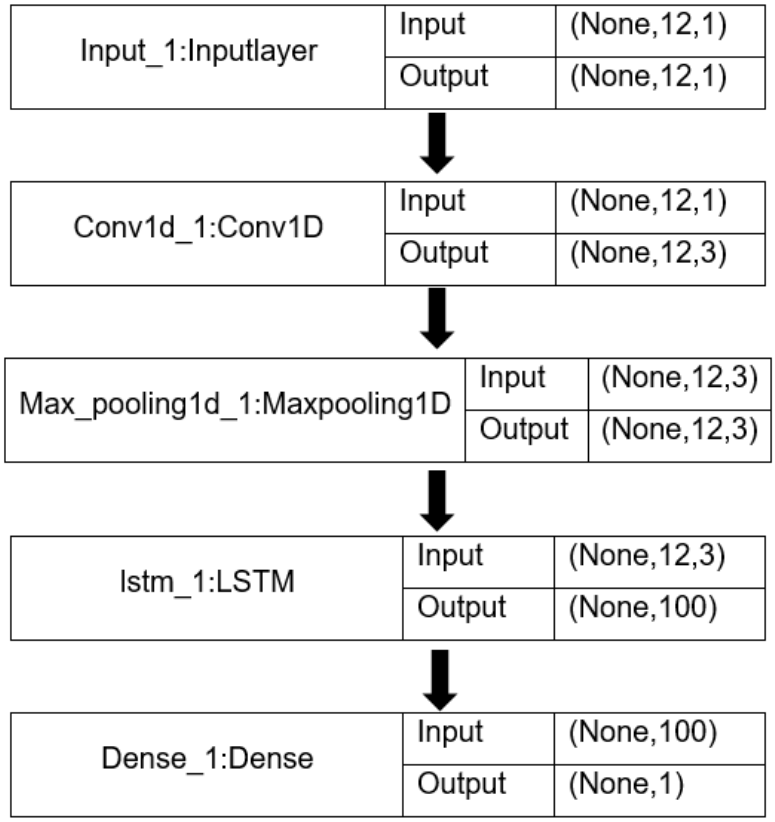


Figure 5.3: This model architecture of CNN-LSTM .

6 Results and Discussion

After configuration and hyperparameter tuning, the processed training dataset is used to train the ARIMA, MLP and CNN-LSTM, respectively. The trained models are then used to predict the test dataset, and the actual values are compared to the predicted value, as shown in Figures 6.1 to 6.4. The actual inflation values are expressed as y, and the number of observations is used because the time-series dataset was transformed into a supervised learning problem, as mentioned in section 6.

The grid search algorithm aims to optimize the ARIMA model using information criteria. Additionally, the RMSE is also calculated for comparability. The benchmark model shows the best fit at order $p = 1$, $d = 1$ and $q = 1$ when RMSE is used. Moreover, South Africa's inflation series had proven stationary when differenced once, implying that the order of integration is 1, which is less optimal because it can also be 0. In this case, the researcher would have to make a judgemental choice relying on their experience with time-series forecasting. Furthermore, the order of integrating the ARIMA model becomes more unclear and inconsistent when a step-wise search is used to minimize an information criterion such as AIC or BIC. The optimal model found, in this case, is AR=1, I=1 and MA=0 respectively, see Appendix C.

Noticeably, the major challenge identified when forecasting inflation using the ARIMA model is finding the correct order from a class of possible models that have proven challenging to understand. Although the RMSE for all the identified models is low, the predicted values are closer to the observed (see Figures 6.1 and 6.3). The model took longer to be trained (see Figure 6.4[8] below), rendering the ARIMA benchmark inconsistent and challenging to configure.

[8]The figure highlights the different approaches employed by the ARIMA benchmark in forecasting South Africa's inflation. The configuration time per second depends on the available central processing unit (CPU) and does not make a fair comparison. For purpose of this study, the configuration time shows how long each model was trained and was computed using an i7-core processor having speed between 2.59GHz and 2.60GHz.

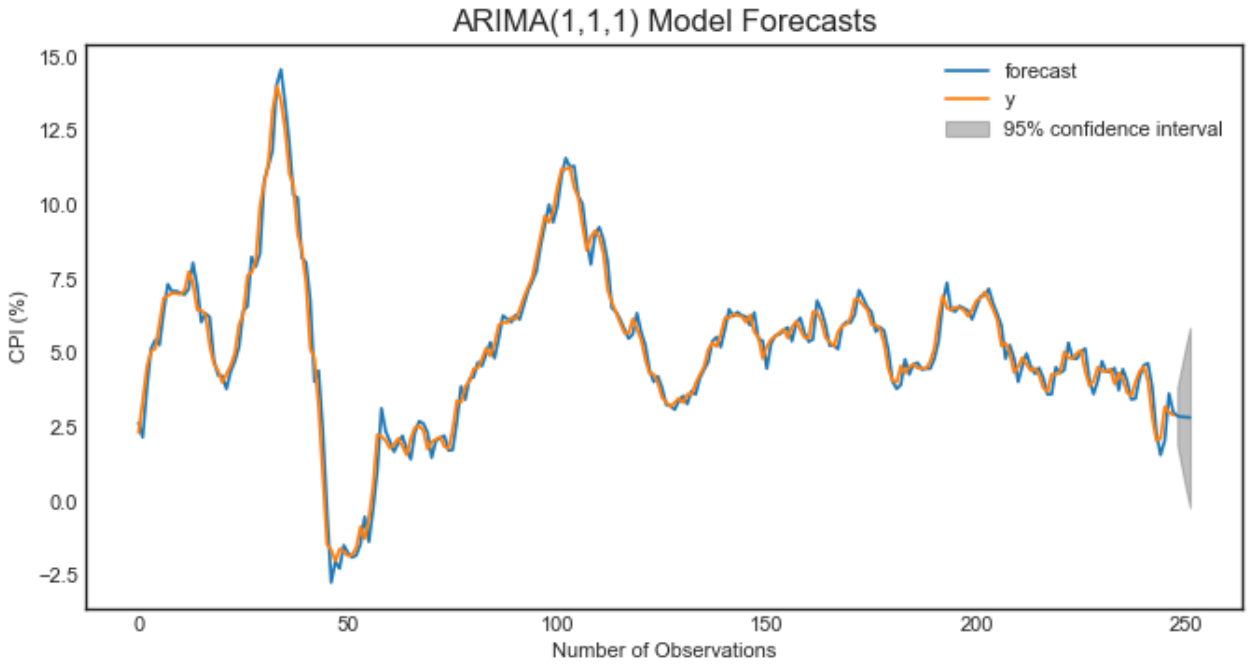


Figure 6.1: The comparison of the predicted value and the actual value

Figure 6.3 and 6.4 shows the DNN forecasting models results. The MLP is robust and performs well throughout the training dataset. The RMSE for the test dataset is 0.513. If the study considered other architectures say $[12, 100, 100, 1, 0]$, where the differencing factor of zero would be the number of nodes in the hidden layer, the RMSE would be 0.532, which indicates that the model remains robust to model specification and therefore consistent, unlike the ARIMA benchmark.

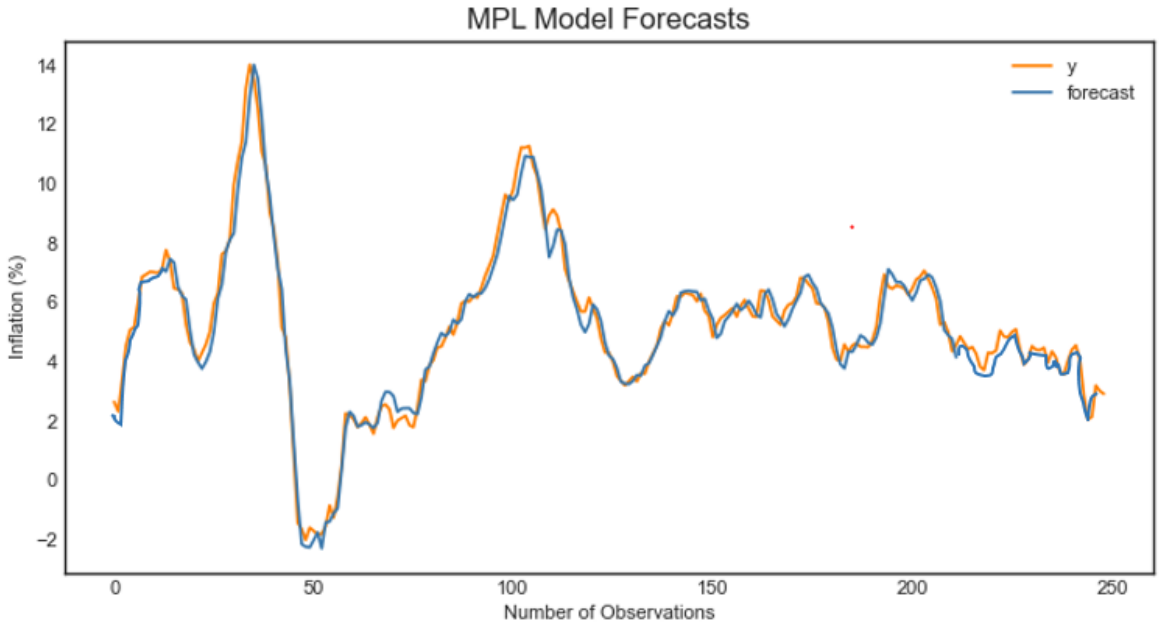


Figure 6.2: The comparison of the predicted value and the actual value

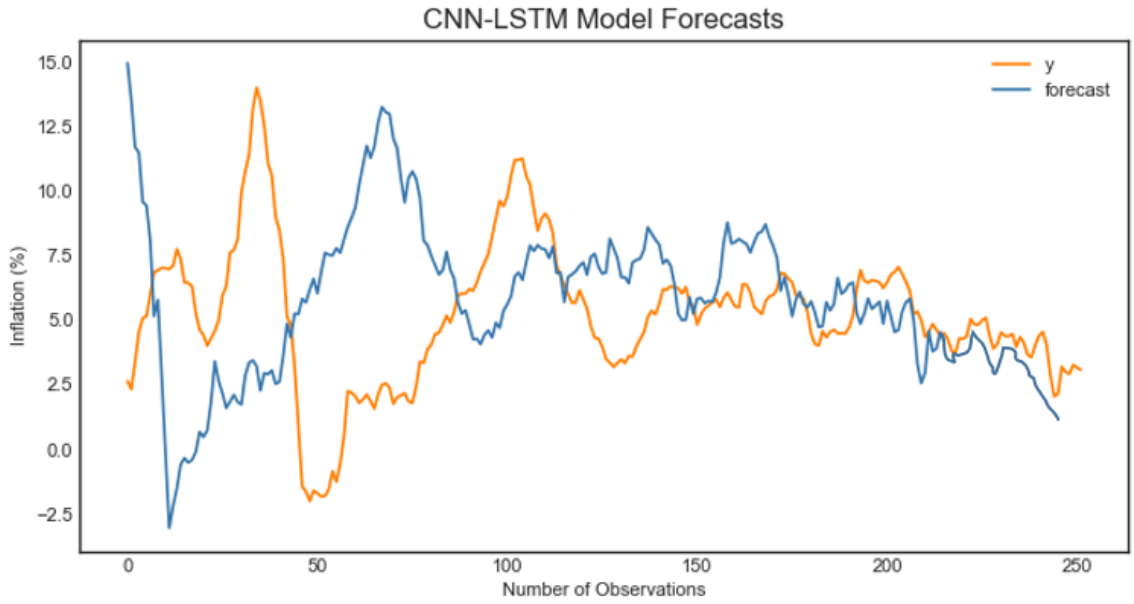


Figure 6.3: The comparison of the predicted value and the actual value

Conversely, the combined feedforward recurrent neural network that is the CNN-LSTM model is the worst model for forecasting South Africa's inflation. The model is unstable because it repeat-

edly over and under predicts throughout the training dataset; see Figure 6.3. What is more, it has the highest RMSE amounting to 0.869 compared to both the MLP and ARIMA and a high computational expense associated with the more extended configuration (see Figure 6.4) equivalent to the ARIMA benchmark.

Moreover, the linear layers of the LSTM layer require large amounts of memory bandwidth to be computed, which explains why the model took a long time to configure. Also, the LSTM part of the model suffered from overfitting. Finally, the study considered various architectures, like adding a 2DConvolutional input layer, which did not improve the model's accuracy because of the limited number of observations.

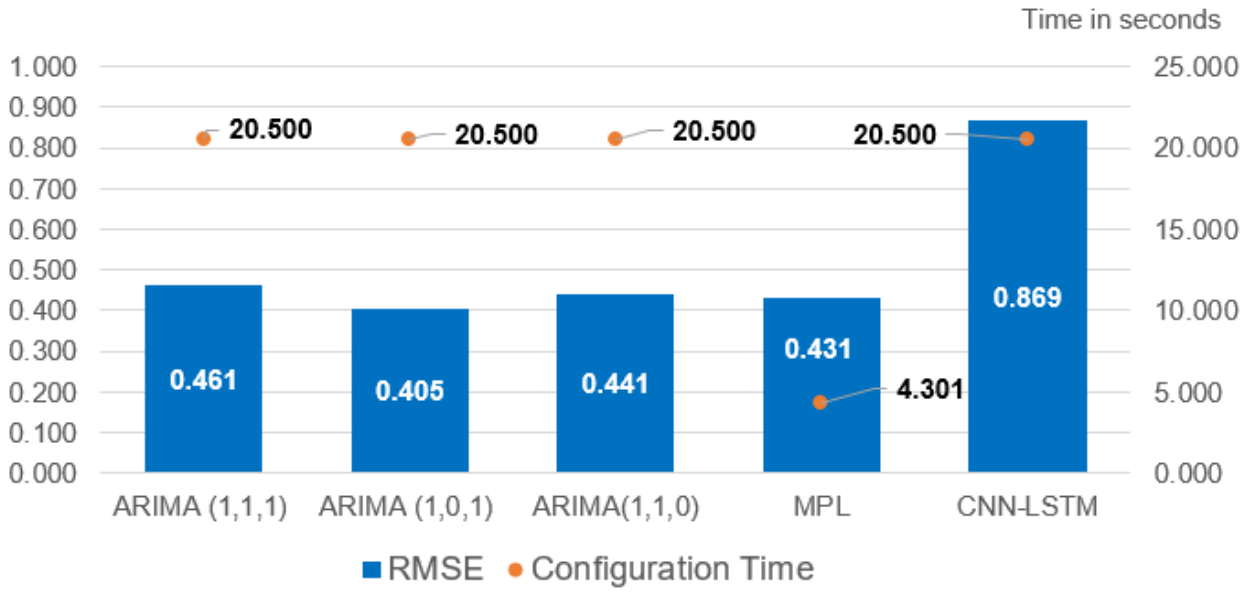


Figure 6.4: Comparison of three methods evaluation

6.1 Do deep learning algorithms make a feasible addition to the macro forecasting toolbox?

Several papers (Makridakis et al., 2018) and (Martin, 2019) insists that "neutral networks should be part of the South African macro forecasting toolbox". This study has proven that deep neural networks, specifically, the MLP generally outperforms different ARIMA models based on ease of

configuration and consistency. Cook and Hall (2017) has shown that ARIMA models are meaningful benchmarks to beat because of their popularity in macroeconomic forecasting. However, for inflation, Van Heerden et al. (2018) find evidence that neural networks are one of the best machine learning methods for forecasting this time-series.

As a reminder, deep learning is a subfield of machine learning, and the MLP is a neural network with more than three-node layers. The backbone of MLP is a neural network. It is not surprising that this study has found that deep neural networks are superior in forecasting inflation. Furthermore, this study confirms that deep neural networks, specifically the MLP, can forecast inflation consistently and accurately. Their accuracy can be improved through different architectural implementations that are easy to configure using the grid search algorithm without compromising the consistency of the model. Therefore, their inclusion in the macroeconomic toolbox is worth considering.

6.2 Further Consideration Avenues for Further Research

It is generally understood that macroeconomic forecasting aims to aid decision making. However, deep learning suffers from the black box issue, which relates to interpretability in forecasting models. According to Rudin (2019), "interpretability is the degree to which a human can understand the cause of a decision or the degree to which a human can consistently predict the model's result". For univariate forecasting problems, the interpretability issue is not prominent because the human understands what is predicted, why the variable is predicted, and how accurate the prediction is.

Choudhary and Haider (2008) found that deep neural networks can learn in the presence of structural breaks. The accuracy of the deep neural networks was compromised due to the low number of trainable observations when using the pre-2000 dataset. Therefore, the deep neural networks will be trained on the full inflation dataset from January 1960 to December 2020. The forecasting architecture is specified in section 5.4 except for batch size, which is now set at 150 due to the increased number of observation. In addition, the models will be evaluated using the 80%:20% train and test split. Therefore 585 observations will be used to train the model, while predictions

will be made and tested using 147 observations.

The MLP, being the basic structure of the deep neural network, still outperforms the 1D CNN-LSTM model achieving an RMSE of 0.486. Appendix D gives the graphical findings. Even after training the CNN-LSTM, the issue of overfitting, as previously stated in Section 6 above, was still persistent. To investigate the matter, it is always better to plot the model loss curve to see if it is learning. From figure 6.5, the presence of overfitting is clear. However, it can be noted that the model loss is decreasing with more training time which is a good indication that the model is learning.

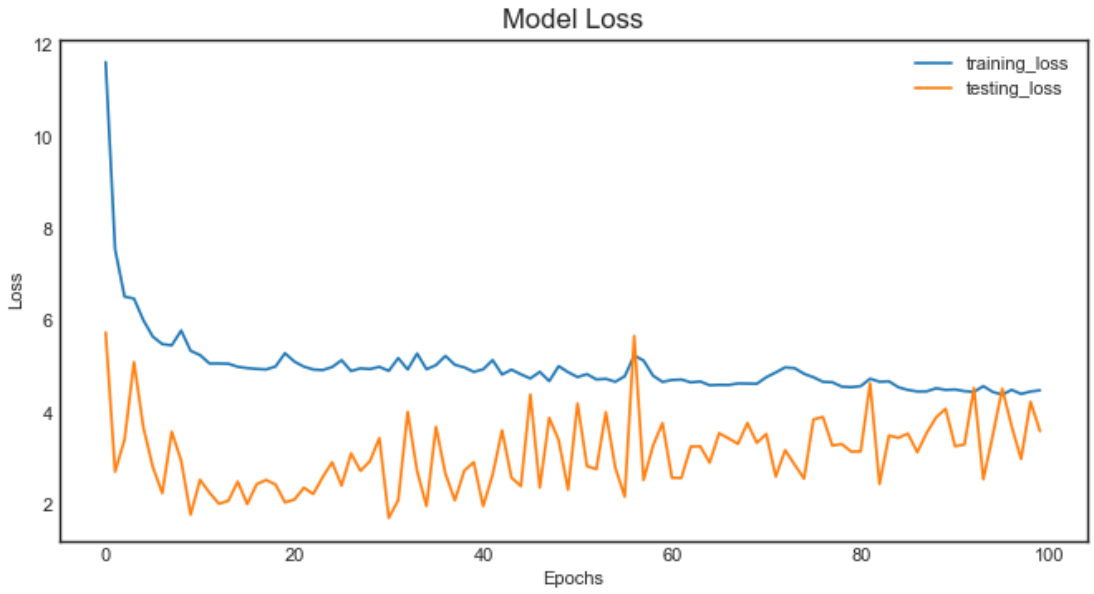


Figure 6.5: Model training and testing loss curves

Numerous avenues for further research can improve the CNN-LSTM model, given the replicability of the grid search algorithm in Python. According to Goodfellow et al. (2016b), the following can be done as more monthly inflation data (say more than 5000 observations) becomes available to improve the accuracy of this deep neural network. Firstly, the window size should be examined in conjunctions with model regularisation; for this study, regularisation was not used because the model would become more computationally intensive, requiring a more robust central processing unit (CPU). Secondly, using more training data, the rule of thumb is that 20% should be reserved for the validation set. An extensive collection of training data can resolve the over-fitting problem.

Thirdly, more node layers and more hidden units can be used, especially if the training error is high. Fourthly, use a different loss function and learning rate. For example, stochastic gradient descent (SGD) with momentum and a decaying learning rate would be an excellent start because they will help the model converge faster.

Lastly, as seen in figure 6.5, the model loss curves were not smooth because of the small batch size. So increasing the batch size will resolve this issue and improve the learning of the model; however, bear in mind that this may require a GPU that can be accessed via virtual machines. So the batch size can increase by the power of two starting from thirty-two until two hundred and fifty-six. Exploring the above can further ascertain if more deep neural networks besides the MPL can be added to the South African macroeconomic forecasting toolbox, particularly as more observations of the inflation time-series become available.

7 Conclusion

South Africa's inflation series has proven to be stationary when differenced once. As a result, this study has found that the ARIMA model tends to be inconsistent concerning the model specification. Specifically, eight different ARIMA model specifications with varying performance and forecasting outcomes yielded inconclusive results, ultimately making the ARIMA model unstable in forecasting South Africa's inflation.

Moreover, the benchmark was inconclusive because the results were not unique even when different approaches were investigated for choosing optimal model specifications and, therefore, not replicable. Often the researcher must rely on their experience with the theoretical framework to repeat the entire modelling procedure, especially the diagnostic checking stages, whenever new data becomes available. Although these can be automated in practice, the tests for optimal lag structures were inconclusive, making determining model specifications challenging.

Whereas deep neural networks are data-driven, they do not rely on the implicit data generating process that is investigated during the diagnostic checking stages. The MLP was found to outperform the benchmark and its peer, the CNN-LSTM model. Admittedly, the performance of the CNN-LSTM was sensitive to architectural design (for example, batch size and the number of hidden layers), especially when the model was trained using few observations.

Eventually, traditional approaches that do not consistently account for structural breaks will fail to predict inflation. The MLP becomes a viable addition to the macroeconomic forecasting toolbox in such a case. Finally, in light of CNN-LSTM's performance on inflation prediction, further analysis can be invested in a multivariate-forecasting setting which seems to be a promising avenue for future research.

References

Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. *CoRR* [online]. *abs/1302.6613* (2013). Available from: http://arxiv.org/abs/1302.6613

Almosova, A., & Andresen, N. (2019). Nonlinear inflation forecasting with recurrent neural networks latest version is available here (2019).

Azorin-Molina, C., Ali, Z., Hussain, I., Faisal, M., Nazir, H. M., Hussain, T., Shad, M. Y., Mohamd Shoukry, A., & Hussain Gani, S. (2017). Forecasting drought using multilayer perceptron artificial neural network model. *Advances in Meteorology* [online]. *2017* (2017), 5681308. https://doi.org/10.1155/2017/5681308

Backus, D., & Zin, S. (1993). Long-memory inflation uncertainty: Evidence from the term structure of interest rates. *Journal of Money, Credit and Banking* [online]. *25*.(3) (1993), 681–700. Available from: https://EconPapers.repec.org/RePEc:mcb:jmoncb:v:25:y:1993:i:3:p:681-700

Bernanke, B. S., & Woodford, M. (1997). Inflation forecasts and monetary policy. *Journal of Money, Credit and Banking* [online]. *29*.(4) (1997), 653–684. Available from: http://www.jstor.org/stable/2953656

Bontempi, G., Ben Taieb, S., & Le Borgne, Y.-A. (2013). Machine learning strategies for time series forecasting. In M.-A. Aufaure & E. Zimányi (Eds.), *Business intelligence: Second european summer school, ebiss 2012, brussels, belgium, july 15-21, 2012, tutorial lectures* (pp. 62–77). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-36318-4_3

Boot, T., & Pick, A. (2020). Does modeling a structural break improve forecast accuracy? *Journal of Econometrics* [online]. *215*.(1) (2020), 35–59. https://doi.org/10.1016/j.jeconom.2019.07

Box, G. E. P., & Tiao, G. C. (1975). Intervention analysis with applications to economic and environmental problems. *Journal of the American Statistical Association* [online]. *70*.(349) (1975), 70–79. Available from: http://www.jstor.org/stable/2285379

Brillinger, D. R. (2001). *Time series*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9780898719246

Brownlee, J. (2020). *Deep Learning for Time Series Forecasting*. Machine Learning Mastery. 1.9.

Burridge, P., Charemza, W., & Hristova, D. (2005). Is inflation stationary? *Applied Economics* [online]. *37* (2005, February), 901–903. https://doi.org/10.1080/00036840500076721

Chakraborty, C., & Joseph, A. (2017). *Machine learning at central banks* (Bank of England working papers No. 674). Bank of England. Available from: https://ideas.repec.org/p/boe/boeewp/0674.html

Chollet, F., & Allaire, J. J. (2018). Deep learning with r. Available from: https://proquest.safaribooksonline.com/9781617295546

Choudhary, A., & Haider, A. (2008). *Neural Network Models for Inflation Forecasting: An Appraisal* (School of Economics Discussion Papers No. 0808). School of Economics, University of Surrey. Available from: https://ideas.repec.org/p/sur/surrec/0808.html

Cook, T., & Hall, A. S. (2017). Macroeconomic indicator forecasting with deep neural networks (2017).

Creel, M. (2017). Neural nets for indirect inference. *Econometrics and Statistics* [online]. *2* (2017), 36–49. https://doi.org/https://doi.org/10.1016/j.ecosta.2016.11.008

Elhadad, M. (2021). *Why most machine learning applications use 10-fold cross-validation*.

Elliott, G., & Müller, U. K. (2014). Pre and post break parameter inference. *Journal of Econometrics* [online]. *180*.(2) (2014), 141–157. https://doi.org/10.1016/j.jeconom.2014.03

Gil-Alana, L. A. (2011). Inflation in South Africa. A long memory approach. *Economics Letters* [online]. *111*.(3) (2011, June), 207–209. Available from: https://ideas.repec.org/a/eee/ecolet/v111y2011i3p207-209.html

Gil-Alaña, L. A., Shittu, O. L., & Yaya, O. S. (2011). *Long memory, structural breaks and mean shifts in the inflation rates in Nigeria* (NCID Working Papers No. 04/2011). Navarra Center for International Development, University of Navarra. Available from: https://ideas.repec.org/p/nva/unnvaa/wp04-2011.html

Goodfellow, I., Bengio, Y., & Courville, A. (2016a). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016b). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Heerden, C. V., Heymans, A., & Seetharam, Y. (2018). Complementing south african inflation surveys: A suitable forecasting tool. *Journal of Economic and Financial Sciences* [online]. *11*.(1) (2018), 14. https://doi.org/10.4102/jef.v11i1.191

Hyndman, R., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd). OTexts.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in r*. Springer. Available from: https://faculty.marshall.usc.edu/gareth-james/ISL/

King, M., & Sriananthakumar, S. (2015). Point optimal testing: A survey of the post 1987 literature. *Model Assisted Statistics and Applications* [online]. *10* (2015, March). https://doi.org/10.3233/MAS-150323

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* [online]. *379*.(2194) (2021, February), 20200209. https://doi.org/10.1098/rsta.2020.0209

Liu, J., Wu, S., & Zidek, J. (1997). On segmented multivariate regression (1997).

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE* [online]. *13* (2018, March). https://doi.org/10.1371/journal.pone.0194889

Martin, L.-C. (2019). *Machine learning vs traditional forecasting methods: An application to south african gdp* (Working Papers No. 12/2019). Stellenbosch University, Department of Economics. Available from: https://EconPapers.repec.org/RePEc:sza:wpaper:wpapers326

MATLAB. (2010). *Version 7.10.0 (r2010a)*. The MathWorks Inc.

Mehtab, S., & Sen, J. (2020). Stock price prediction using cnn and lstm-based deep learning models.

Mills, T. C. (2013). Yule's lambdagram revisited and reclaimed. *Economics & Finance Research* [online]. *1*.(1) (2013), 1–12. https://doi.org/10.1080/21649480.2012.751518

Pesaran, M., Pick, A., & Timmermann, A. (2011). Variable selection, estimation and inference for multi-period forecasting problems. *Journal of Econometrics* [online]. *164*.(1) (2011), 173–187. Available from: https://EconPapers.repec.org/RePEc:eee:econom:v:164:y:2011:i:1:p:173-187

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* (1958), 65–386.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.

Stock, J. H., & Watson, M. W. (1998). *A Comparison of Linear and Nonlinear Univariate Models for Forecasting Macroeconomic Time Series* (NBER Working Papers No. 6607). National Bureau of Economic Research, Inc. Available from: https://ideas.repec.org/p/nbr/nberwo/6607.html

Yan, K., Wang, X., Du, Y., Jin, N., Huang, H., & Zhou, H. (2018). Multi-step short-term power consumption forecasting with a hybrid deep learning strategy. *Energies* [online]. *11* (2018, November), 3089. https://doi.org/10.3390/en11113089

Yao, Y.-C. (1988). Estimating the number of change-points via schwarz' criterion. *Statistics Probability Letters* [online]. *6*.(3) (1988), 181–189. Available from: https://EconPapers.repec.org/RePEc:eee:stapro:v:6:y:1988:i:3:p:181-189

Zhang, P., Patuwo, E., & Hu, M. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* [online]. *14* (1998, March), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7

A Appendix: Preliminary Analysis

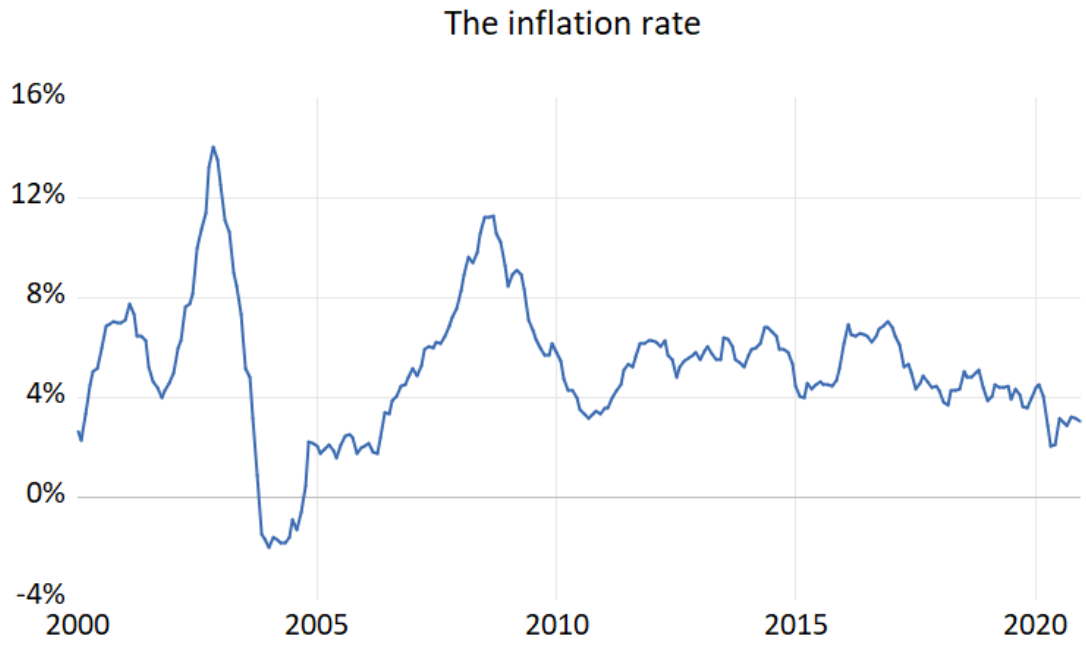


Figure A.1: Depicts South Africa's monthly inflation from 2000-2020 (post-2000). This series is used to train the ARIMA and deep learning models.

The removal of the exogenous structural breaks motivated this study to examine whether the post-2000 inflation series is stationary or follows a unit root process. The following hypothesis is formulated:

H_0: The post-2000 inflation series has a unit root.

H_1: The post-2000 inflation series does not have a unit root.

The Augmented Dickey-Fuller (ADF) test in the table below shows statistical results with an intercept included in the test regression. The test statistic of -2.255 is not smaller than the 5% critical value of -3.429, leading us to fail to reject the null of a unit root and to infer that the inflation series

is non-stationary.

ADF regression with an intercept				
	Test Statistic	5% Critical Value	10% Critical Value	P>|t|
Z(t)	-2.255	-3.429	-3.138	0.459

Table A.1: ADF Test at level

Therefore, the series can be made stationary by computing the difference of consecutive terms. This process is known as differencing, which is typically performed to get rid of the varying mean. Mathematically, differencing can be written as:

$$D[Inflation_t] = Inflation_t - Inflation_{t-1} \tag{A.1}$$

After repeating the ADF test on the differenced data, it is found that the series is now stationary. Expressly, the null hypothesis is rejected at all confidence levels of significance, which means the inflation series is stationary and does not follow the unit root at its first difference. Table A.2 below presents the results.

ADF regression with an intercept				
	Test Statistic	5% Critical Value	10% Critical Value	P>|t|
Z(t)	-6.354	-3.429	-3.138	0.000***

Table A.2: ADF test at first difference and the level of significance at ***<0.01, **<0.05 and *<0.10

B Appendix: Validation and Test Periods

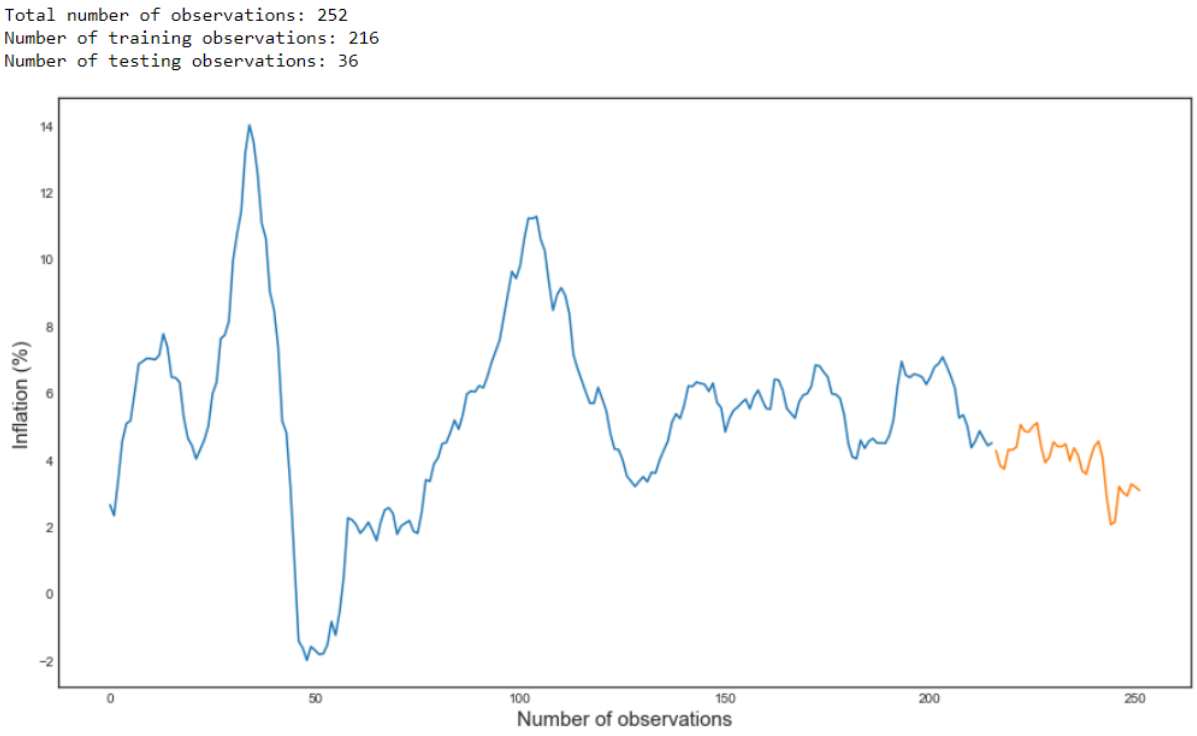


Figure B.1: The inflation series for the validation and test periods. The period from 2000-2017, coloured blue, is used for training the model. Furthermore, 2018 - 2020, coloured in orange, is used as a final testing set.

```python
# walk-forward validation for univariate data
def walk_forward_validation(data, n_test, cfg):
  predictions = list()
  # split dataset
  train, test = train_test_split(data, n_test)
  # seed history with training dataset
  history = [x for x in train]
  # step over each time step in the test set
  for i in range(len(test)):
    # fit model and make forecast for history
    yhat = simple_forecast(history, cfg)
    # store forecast in list of predictions
    predictions.append(yhat)
    # add actual observation to history for the next loop
    history.append(test[i])
  # estimate prediction error
  error = measure_rmse(test, predictions)
  return error
```

Figure B.2: Python code for implementing the walk-forward validation approach adopted from Brownlee (2020) .

C Appendix: Performing Step-wise Search

Model	AIC	BIC	RMSE
ARIMA(0,1,0)	433.549	437.075	0.435
ARIMA(2,1,0)	366.130	380.232	0.454
ARIMA(1,1,1)	365.185	379.287	0.461
ARIMA(1,1,0)*	362.818*	369.869*	0.441*
ARIMA(2,1,0)	364.130	374.706	0.452
ARIMA(1,1,1)	363.186	373.762	0.461
ARIMA(0,1,1)	377.065	384.116	0.409
ARIMA(2,1,1)	363.295	377.397	0.453

*Best Model

Table C.1: Step-wise search results

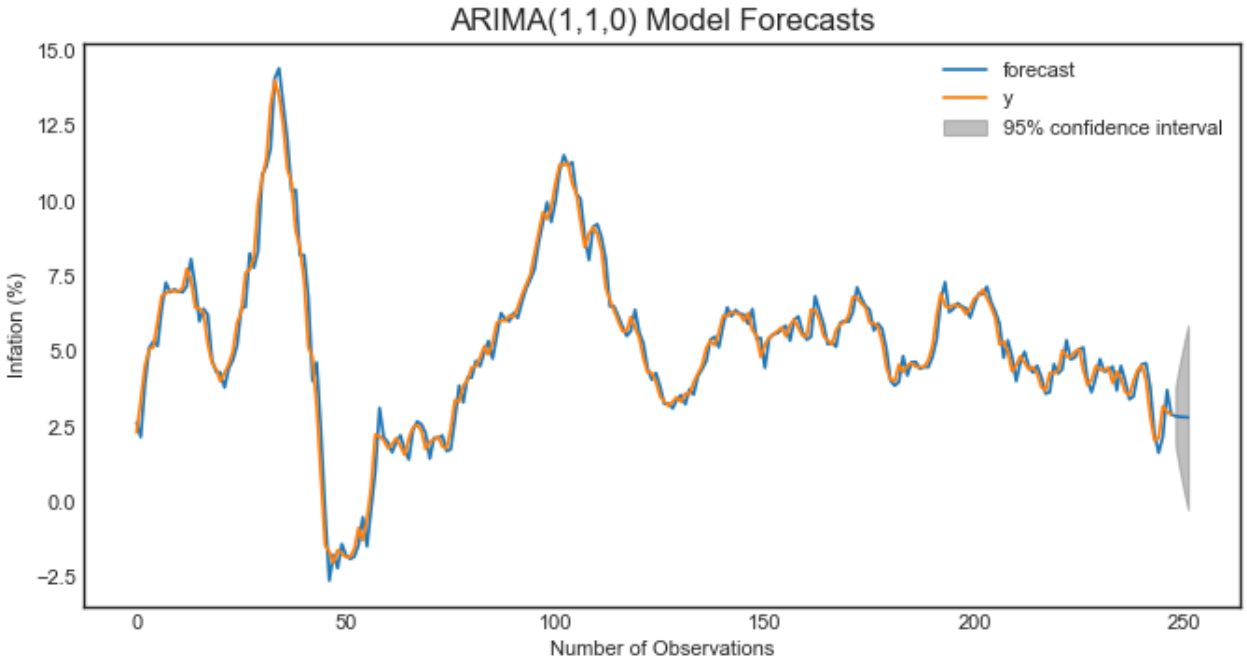


Figure C.1: The comparison of the predicted value and the actual value

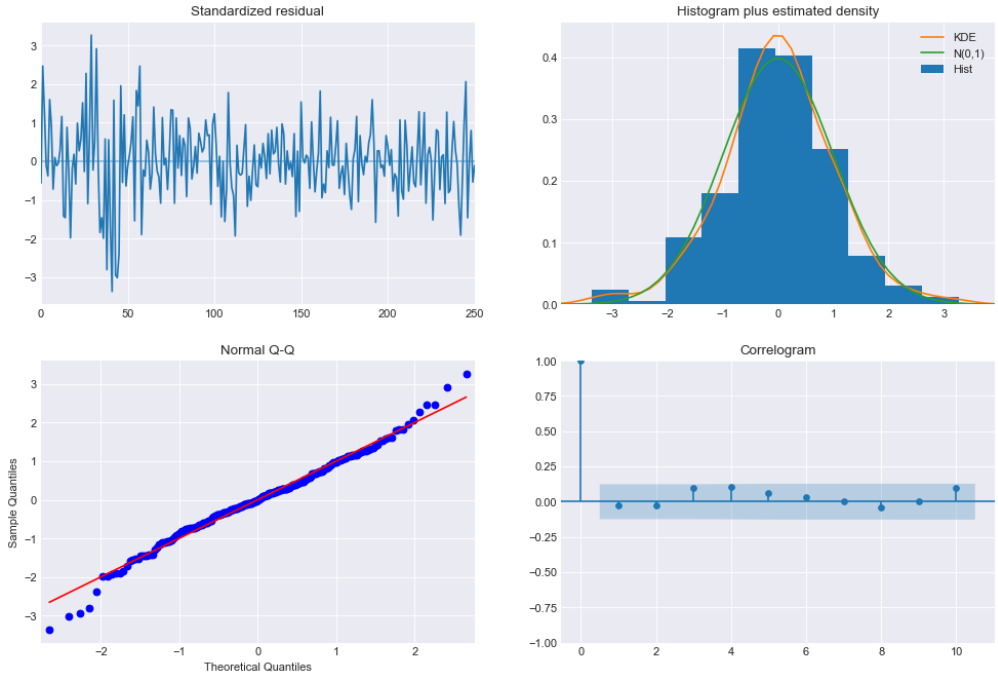


Figure C.2: Diagnostic plots of the ARIMA (1,1,0) hyperparameter configurations

D Appendix: Forecasting using deep neural networks

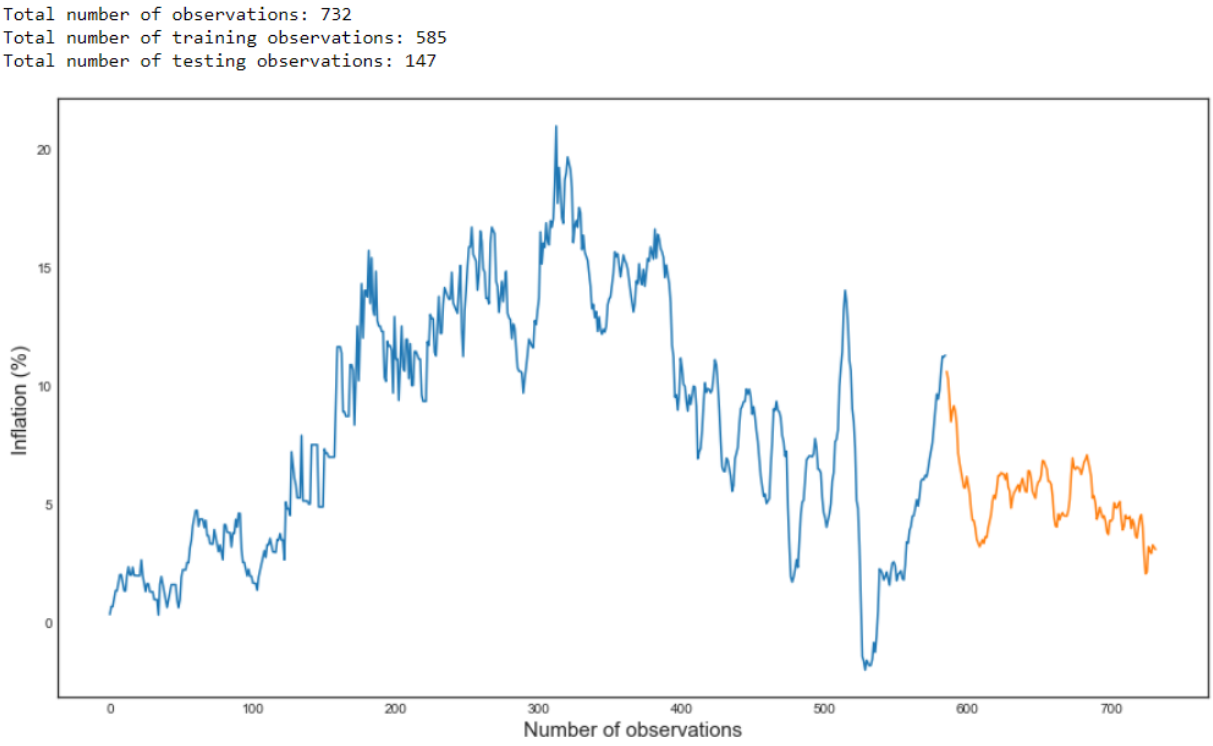


Figure D.1: The inflation series for the validation and test periods. The period from 1960 till October 2008 (observations = 585), coloured blue, is used to train the model. The period from November 2008 till December 2020 (observations = 147), coloured in orange, is used as a final testing set

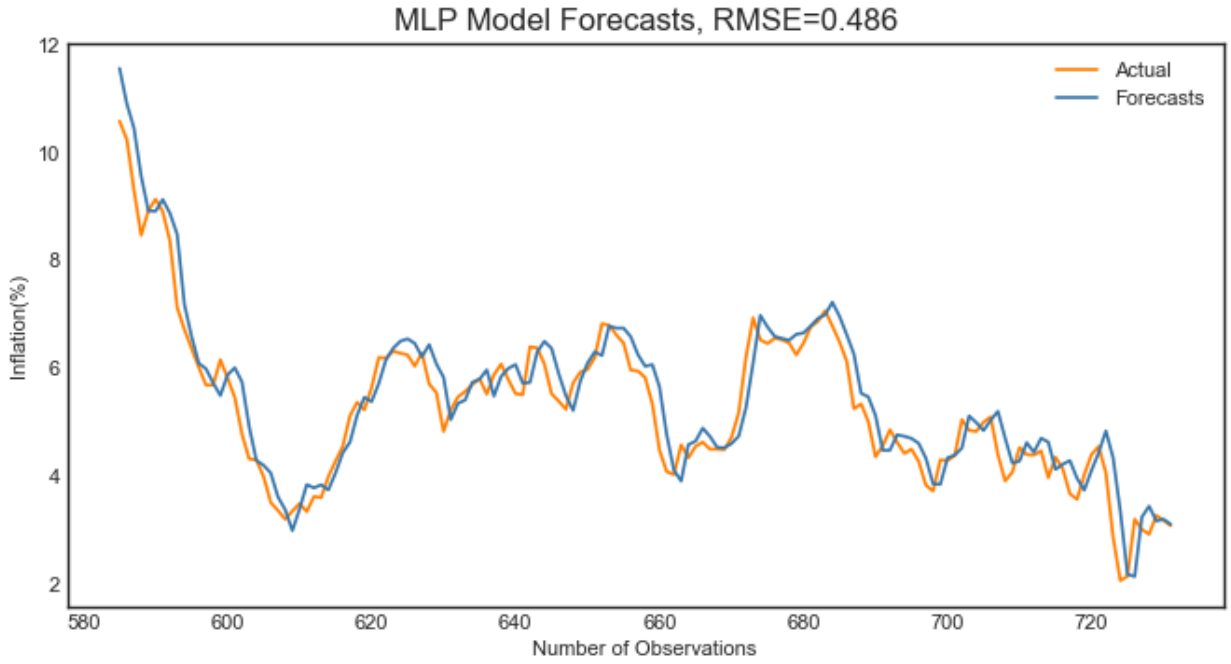


Figure D.2: The comparison of the predicted value and the actual value

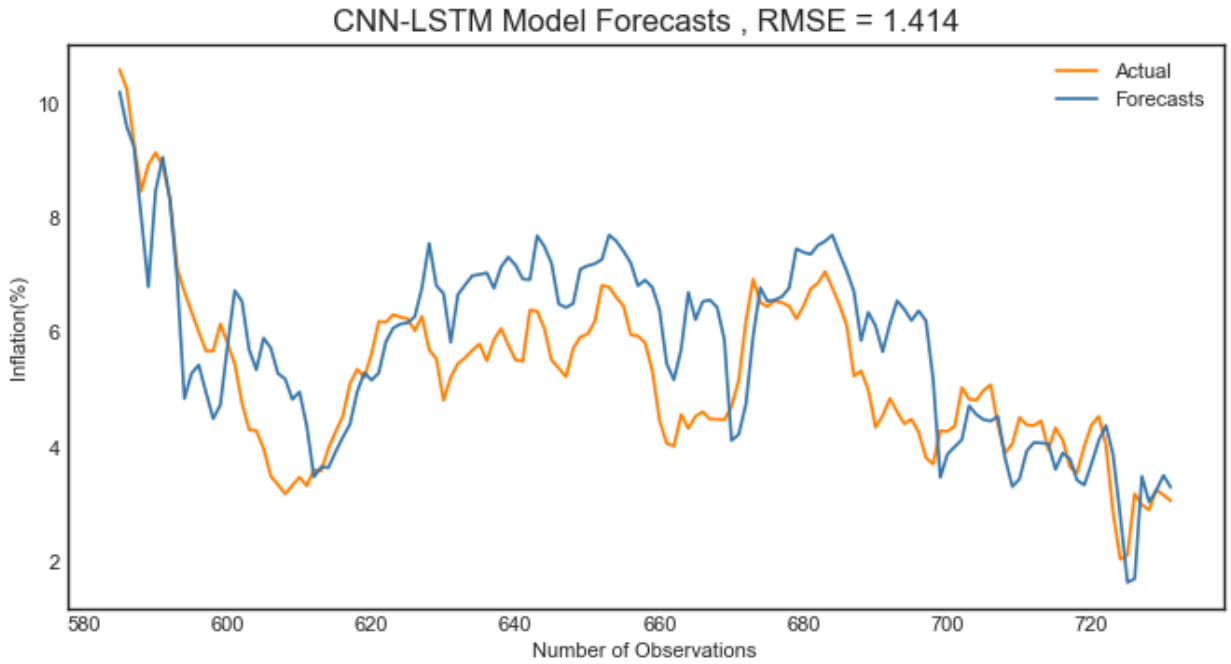


Figure D.3: The comparison of the predicted value and the actual value