Erasmus University Rotterdam
Erasmus School of Economics
Econometrics and Operational Research

Bachelor Thesis Quantitative Finance

July 4, 2021

Investigating Non-Linear Machine Learning Methods for Inflation Forecasting in a Data-Rich Environment

Realised by:

Rogier Nagelkerken *497436*

Supervised by:

Prof. E.P. O'Neill

Second Assessor:

A.A. Naghi

Abstract

This paper extends the work by Medeiros et al. (2021) by comparing the performance of Random Forest (RF) and Long Short-Term Memory (LSTM), two non-linear Machine Learning (ML) models, in forecasting CPI inflation. Their performances are also compared to those of two univariate benchmarks, the Autoregressive (AR) and Random Walk (RW) models. It uses a US and a Canadian high-dimensional macroeconomic dataset. Forecasts are computed for different horizons (1 to 12-months) using a rolling window framework. This paper finds that both ML models outperform the RW model consistently for both datasets. And, even though the ML models consistently outperform the AR model for the US dataset, they fail to outperform the AR model consistently for the Canadian dataset.

Keywords: Inflation Forecasting, Long Short-Term Memory, Random Forest, Machine Learning

Contents

1 Introduction

Inflation forecasting is of great importance in many sectors. Long-term nominal commitments such as mortgages and labor contracts are influenced by the inflation forecasts. Inflation forecasting thus has some effect on almost everyone (e.g. households, businesses, policymakers).

Although inflation forecasting is a long-studied subject, it was only in recent years that researchers found models that perform better than simple univariate models. The most prominent univariate models in inflation forecasting are: the Random Walk (RW), Autoregressive (AR) and Unobserved Components with Stochastic Volatility (UCSV) models. Stock & Watson (2007) point out that inflation in the US has recently become both easier and harder to predict. On the one hand, since the mid-80s the volatility of inflation has decreased, making it easier to predict. On the other hand, outperforming the naive univariate RW model became harder since then. Stock & Watson (2007) and other researches argue that this may have to do with the changing behaviour of inflation over time. This observation leads to an increase in popularity of non-linear forecasting models.

This paper extends the work of Medeiros et al. (2021). In Medeiros et al. (2021), different models for forecasting US inflation in a data-rich environment are compared for different horizons (1 to 12-months). These models are of varying origins, from the univariate Autoregressive (AR) model, to Machine Learning (ML) models like Random Forest (RF) as well as shrinkage models like LASSO. For all models, a rolling window framework is used. They found that, for the US data, RF consistently gives better forecasts than all other models. They argue that the RF performs well, mostly because of its ability to capture non-linearities in the data. Our paper extends the aforementioned research by adding a relatively novel non-linear ML model to the comparison: Long Short-Term Memory (LSTM). We choose to use LSTM because it is a type of Neural Network (NN) which is specialized in capturing long-term (non-linear) dependencies in high-dimensional time series data. Therefore, it may perform well in forecasting inflation. We compare LSTM against the RW and AR models (benchmark models) to check if it can consistently outperform these univariate models. We also compare LSTM against RF to find out how the accuracy of LSTM relates to that of the best model found by Medeiros et al. (2021). We compare these models for two different datasets, a US dataset, as used by Medeiros et al. (2021), and a Canadian dataset. We use the latter dataset to check if the results found by Medeiros et al. (2021), as well as our own results, can be generalised to forecasting inflation in different countries, using different datasets.

For obtaining forecasting results using LSTM, we first need to set the hyperparameter values. We use

Bayesian Hyperparameter Optimisation to find the optimal number of neurons in the hidden layers. The other hyperparameter values are set manually because the computation time would increase too much if all hyperparameter values are optimised. We compare the results for RF and LSTM to those obtained by the benchmark models using the Harvey-Leybourne-Newbold (HLN) test. The HLN test checks whether two models have equal accuracy for the entire sample. For comparing the results for LSTM to those obtained by RF, we us the Fluctuation Test. The Fluctuation Test checks whether the models produce forecasts of equal accuracy over the entire sample, or whether one model temporarily performs better than the other model.

We find that, for both datasets, LSTM and RF have similar accuracy. Also, both ML models outperform the RW model consistently for (almost) all horizons for both datasets. For the US dataset, both LSTM and RF also outperform the AR model consistently. However, for the Canadian dataset, both LSTM and RF fail to outperform the AR model consistently. This implies that the results found for comparing the ML models to the AR model for the US dataset can not be generalised to different datasets.

The structure of the remainder of this paper is as follows. Section 2 serves as a literature overview, and Section 3 describes the data. Next, Section 4 presents the methodology, followed by the results in Section 5. Section 6 contains the conclusion.

2 Theory

Inflation forecasting has been around for a long time. Traditional literature on inflation forecasting usually uses Phillips curve-based models. The Phillips curve relates unemployment or some other measure of aggregate economic activity to inflation. Using this relation, Phillips curve-based models set the forecasting function to be a function of an aggregate economic activity variable and autoregressive terms. Although well-established, the Phillips curve-based models have varying performances over time. Moreover, they can be outperformed by univariate models, like the unobserved components stochastic volatility (UCSV) model (Stock & Watson, 2007), the random walk model (Atkeson et al., 2001), or autoregressive models. Other benchmarks used for inflation forecasting are BVARs (Giannone et al., 2015) and dynamic factor models (Stock & Watson, 2002; Ludvigson & Ng, 2007). Dynamic factor models especially performed well in forecasting for short horizons.

In the recent decades, we have seen many advances in computational power and the availability of large datasets. For those reasons, machine learning (ML) techniques have become more popular

in the forecasting environment. This is because ML models are generally better at handling high-dimensional data. There are multiple researches claiming that there exist no models, including ML models, that can consistently outperform the aforementioned univariate benchmark models (Stock & Watson, 1999; Atkeson et al., 2001; Stock & Watson, 2007). In the early days of using ML models, predominantly linear ML models were used, these models were often outperformed by the univatiate benchmark models (Teräsvirta et al., 2005). However, there are multiple researches that prove to consistently obtain better results by using non-linear ML models like Random Forests (RFs) (Medeiros et al., 2021), Support Vector Regressions (SVRs) (Sermpinis et al., 2014; Sermpinis et al., 2014), or Neural Networks (NNs) (Nakamura, 2005; Choudhary & Haider, 2012).

Our paper adds to current literature firstly by checking if the results obtained by Medeiros et al. (2021), namely that RF consistently outperforms the AR and RW models in forecasting inflation, can be generalised to other high-dimensional datasets. Secondly, we use Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and compare the forecasting results of LSTM to those of RF and the AR and RW models for both of the high-dimensional datasets. LSTM is a type of NN with increasing popularity in the recent years for forecasting inflation (Almosova & Andresen, 2019; Masini et al., 2020; Paranhos, 2021), but it is still unclear whether it performs consistently better or worse than competing ML models or whether the difference is insignificant. Although there has been a substantial number of researches addressing how well NNs perform in the inflation forecasting context, most of them use the more basic NNs, like Feed Forward NNs (Nakamura, 2005; Chakraborty & Joseph, 2017). The main distinction between LSTM and the basic NNs is that LSTM models the input explicitly as a sequence of observations, whereas basic NNs are not able to handle the time-series aspect of the data. Moreover, LSTM assumes that there exists a dependence across time steps. Because of these aspects, we choose to investigate the performance of LSTM in forecasting inflation rather than that of other NNs.

3 Data

This paper aims to extend Medeiros et al. (2021), therefore the process of collecting and transforming the data is similar. Following Medeiros et al. (2021), we use the FRED-MD database[1] by McCracken & Ng (2016) to compare the performance of LSTM to RF and the benchmark AR and RW models in forecasting inflation using high-dimensional data. The FRED-MD database is a high-dimensional

[1]Available at https://research.stlouisfed.org/econ/mccracken/fred-databases/, date accessed: May 14, 2021.

monthly macroeconomic dataset containing US macroeconomic variables. Moreover, we use the LCDMA database[2] by Fortin-Gagnon et al. (2018) to check if results of the model comparisons can be generalised to other datasets. The LCDMA database is, similar to FRED-MD, a high-dimensional monthly macroeconomic dataset containing Canadian macroeconomic variables. For performing LSTM, all variables (including CPI) in both datasets are standardized.

3.1 FRED-MD

As described above, the FRED-MD database contains US macroeconomic data. The FRED-MD database is updated in real time. We have chosen not to use the more recent data since January 2016 because this research is an extension of Medeiros et al. (2021), and it is more practical to use the exact same dataset as is used by them. This means that we use the vintage as of January 2016, with the sample extending from January 1960 to December 2015 (672 observations). Following

Figure 1: The monthly US inflation during the sample period.

Medeiros et al. (2021), we only use variables for which there are no missing values within the sample period (122 variables). On top of that, we use the four principal component factors computed from this set of variables as potential predictors. All variables are transformed such that stationarity is guaranteed. The transformations for all variables can be found in McCracken & Ng (2016). Inflation is calculated as the price difference in consecutive months: $\pi_t = \log(P_t) - \log(P_{t-1})$, where P_t is the price index at period t. In this research, we use the CPI as the price index indicator. Figure 1 shows the movement of the inflation within the sample period. We compare the performances across models in two different subsample periods, namely, January 1990 to December 2000 (132

[2]Available at http://www.stevanovic.uqam.ca/DS_LCMD.html, date accessed: May 14, 2021.

observations) and January 2001 to December 2015 (180 observations). The first subsample has substantially lower inflation volatiliy ($\sigma = 0.17\%$) than the second subsample ($\sigma = 0.32\%$). For the first subsample, we use a rolling window of 360 observations for training, for the second subsample the rolling window is of size 492. Also, for training the RF model, a dummy variable is added such that the outlier in November 2008 does affect the training process.

3.2 LCDMA

The LCDMA database is the Canadian equivalent of the FRED-MD database. Even though it does not contain the exact same variables, it does contain many Canadian macroeconomic variables which are transformed to guarantee stationarity as well. The transformations for all variables can be found in Fortin-Gagnon et al. (2018). They have collected data starting from 1914, but only the observations starting from January 1981 are usable. This is because most variables only have observations starting from January 1981 and the data transformations are not applied on the data before 1981. We use the vintage as of June 2021. The sample we use extends from January 1981 to December 2015 (420 observations) because we aim to make forecasts on the same period as for the second subsample of the US data, that is, from January 2001 to December 2015 (180 observations). We use a rolling window of 240 observations for training the models. Similar to the US data, we use the CPI as a price index indicator for inflation. We do not use principal components for the Canadian data because there are more explanatory variables than observations. Figure 2 shows the inflation over the sample period. There is a high peak in January 1991, which coincides with the

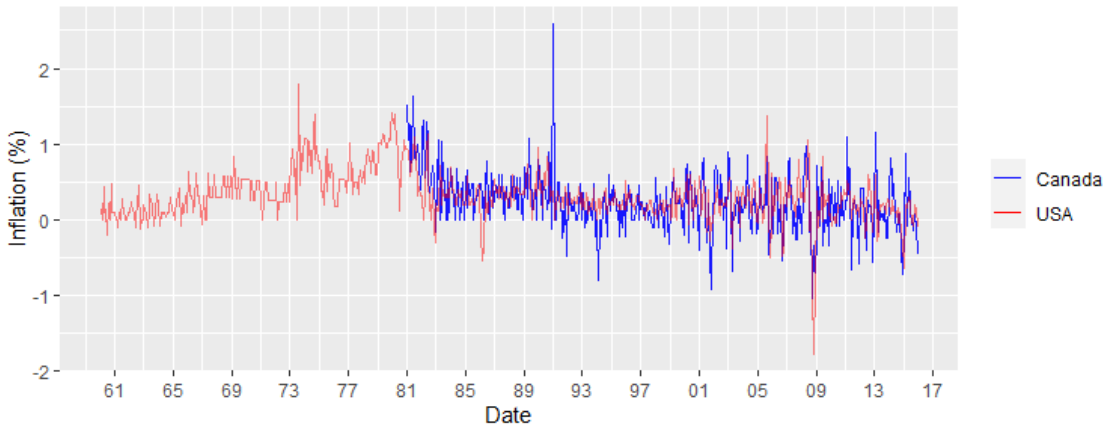


Figure 2: The monthly Canadian inflation during the sample period.

introduction of the federal Goods and Service Tax (GST), which led to an increase in prices of almost

all products. Similar to the US CPI, the Canadian CPI shows a negative spike corresponding to November 2008. However, the spike for Canadian CPI has an almost twice as small magnitude. The volatility of the CPI from January 1984 to January 2001 is equal to 0.31% (0.27% when excluding January 1991), from January 2001 to December 2015, the volatility is equal to 0.39%. Similar to the US, there is a higher volatility in the period from January 2001 onward, the difference becomes more significant when excluding January 1991 from the sample.

4 Methodology

The models we discuss are: the Random Walk (RW) and Autoregressive (AR) models, Random Forest (RF), and Long Short-Term Memory (LSTM). The first two methods are traditional, univariate models that we use as a benchmark for the latter two. RF and LSTM both are machine learning (ML) methods. Both methods have the ability to capture nonlinear dependencies between the variables. RF can be used in a wide range of applications, whereas LSTM is predominantly used in forecasting time series. LSTM is a type of Neural Networks (NN) and Recurrent Neural Networks (RNN). Therefore, before discussing the LSTM method, we will cover what NNs and RNNs are.

4.1 Benchmark Methods

The two benchmark methods are, Random Walk (RW) and the Autoregressive (AR) model.
RW is a simple univariate forecasting model and is therefore often used as a benchmark model. For any horizon h, the predicted inflation is equal to the last known inflation value, that is: $\hat{\pi}_{t+h|t} = \pi_t$. For the h-period ahead accumulated forecast, the prediction consists of adding up the h last known inflation values, that is: $\hat{\pi}_{(t+1:t+h)|t} = \sum_{i=0}^{h-1} \pi_{t-i}$.
The AR model is, similar to the RW model, a univariate forecasting method which only makes use of autoregressive terms. As discussed in Section 2, even though the AR model is relatively simple, it has been one of the better performing methods for a long time in forecasting inflation. An AR(p) model makes h-period ahead forecasts as follows:

$$\hat{\pi}_{t+h|t} = \hat{\beta}_{0,h} + \sum_{i=0}^{p-1} \hat{\beta}_{i+1,h}, \pi_{t-i} \tag{1}$$

where the β's are parameters which are estimated using Ordinary Least Squares (OLS). Hyperparameter p determines how many lags are used in the forecast, p is determined by the Bayesian Information Criterion (BIC) which prevents the model from overfitting by penalising higher model

complexity. The accumulated h-period ahead forecast is calculated by aggregating the individual forecasts for each horizon, that is: $\hat{\pi}_{(t+1:t+h)|t} = \sum_{i=1}^{h} \hat{\pi}_{t+i|t}$. The accumulated forecasts for RF and LSTM are calculated in the same manner.

4.2 Random Forest

The Random Forest (RF) model was first introduced by Breiman (2001), with the aim to reduce variance from regression trees. It uses bootstrap aggregation (bagging) of regression trees to achieve a lower variance (Breiman, 1996).

A detailed explanation on how RFs works, can be found in Appendix A. We will not go into much detail here because the method was discussed in Medeiros et al. (2021).

In short, a RF constructs B regression trees. Each regression tree is trained using a random subset of covariates. Moreover, assuming we have T training observations, the training data for each regression tree consists of T observations which are sampled with replacement from the original training observations. The forecasts of a RF are produced by averaging the forecasts of all B regression trees.

Appendix D explains which hyperparameter values need to be chosen and how this choice is made.

4.3 Neural Networks

Neural Networks (NNs) are one of the most traditional non-linear models. A NN consists of an input layer, one or multiple hidden layers, and an output layer. Each layer consists of one or multiple neurons. The input of neurons (except for the input layer) is a linear combination of the output of all the neurons in the previous layer plus a bias. The input of the neurons in the input layer are the values of the covariates of the model ($\mathbf{X}$). The most common NN is a Feed Forward NN. Appendix B describes the inner workings of a Feed Forward NN.

4.4 Recurrent Neural Network

A Recurrent Neural Networks (RNN) is a type of NN which is able to process data sequentially rather than simultaneously as done for Feed Forward NNs. This is especially useful when you are dealing with data for which the order is important. Therefore, RNNs are often used in e.g. linguistic and economic time series research. To understand RNNs better, we first discuss its architecture. Then we discuss the math behind RNNs. Finally, we discuss some of the shortcomings of 'normal' RNNs.

An RNN treats the training data as a sequence and tries to find patterns by memorising past observations. To go in more detail on how RNNs store past information, we use the example of an RNN with one hidden layer. Figure 3 shows the architecture of a 1-layer RNN. On the left, it shows an architecture which is similar to a Feed Forward NN architecture. $\mathbf{x}$ denotes the values of the input layer, s those of the hidden layer, and o denotes the output. Moreover, matrices U, V and W are used to form linear combinations of neuron outputs to pass to the next layer. The architecture differs from a Feed Forward NN because it contains an arrow with matrix W which is a feedback loop within the hidden layer s. Matrices U, V and W are referred to as *weight matrices*. On the right, Figure 3 shows how this feedback loop unfolds, namely, the output of hidden layer s at step t is used as input for s at step $t+1$, together with the original input $\mathbf{x}_{t+1}$.

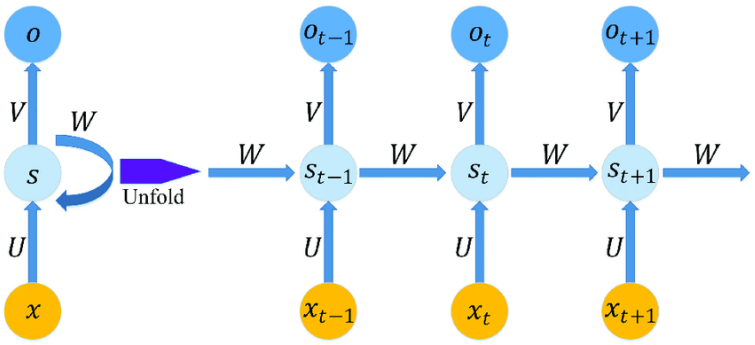


Figure 3: 1-layer Recurrent Neural Network architecture.[3]

The values of the weight matrices are estimated by using Backpropagation Through Time (BPTT). Before explaining in more detail what BPTT does, we will first introduce some notation. The value we try to forecast at time t, or the ground truth, is denoted as y_t, such that:

$$\begin{aligned} s_t &= S_s(W s_{t-1} + U\mathbf{x}_t + b_s) \\ o_t &= S_o(V s_t + b_o) \\ Loss_t &= L(o_t - y_t) \\ Loss &= \sum_{t=1}^{T} Loss_t \end{aligned} \tag{2}$$

where T is the number of observations. In the above equation, function $L(\cdot)$ determines how

[3]From https://journals.plos.org/plosone/article/figures?id=10.1371/journal.pone.0180944, date accessed: June 30, 2021

the *Loss* depends on the error in the output. Oftentimes, $L(\cdot)$ is the least squares loss function: $L(error) = \frac{1}{2}error^2$. $S_s(\cdot)$ and $S_o(\cdot)$ are the activation functions, which are discussed in more detail in Appendix B. Moreover, b_o and b_s are the biases.

BPTT aims to minimise the *Loss*. It does so by calculating the gradient of the *Loss* with respect to the weight matrices. Based on the gradient, the weight matrices are adjusted towards a minimisation of the *Loss*. Appendix C explains the problem of vanishing and exploding gradients which may occur for 'normal' RNN structures.

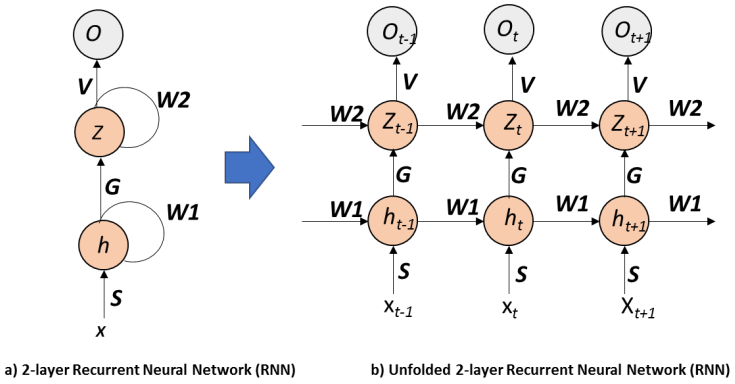


Figure 4: 2-layer Recurrent Neural Network architecture.[4]

Until now, we have only discussed the case of a single hidden layer RNN. RNNs with multiple layers are a straightforward extension of this. Figure 4 shows a RNN with two hidden layers. As we can see, the only difference is that there are more weight matrices to calculate and the input goes through more activation functions (need not be the same). Similar to Feed Forward NNs, RNNs with more layers give opportunity for more different non-linear dependencies to be captured, but they are also computationally more costly with a bigger chance of overfitting. The exploding and vanishing gradients problem also holds for a 'normal' RNN with multiple layers. Therefore, long-term dependencies can not be captured consistently. To capture long-term dependencies, specific RNN structures are introduced. The two most prominent RNNs that are specialised in capturing long-term dependencies are: Gated Recurrent Unit (GRU) (Chung et al., 2014) and Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997).

[4]From https://www.oreilly.com/library/view/r-deep-learning/9781787121089/19380309-271e-49b4-9ef4-4c6ee331edcf.xhtml, date accessed: June 30, 2021

4.5 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of RNN which is able to consistently capture long-term dependencies, without having the problem of exploding or vanishing gradients. Figure 5 shows the architecture of a typical LSTM layer. A NN can consist of multiple LSTM layers, possibly combined with non-LSTM (e.g. Feed Forward) layers. In the figure, σ represents a logistic activation function and *tanh* represents a hyperbolic tangent activation function (see Equation (7)). The "×"" and "+" denote element-wise multiplication and sum operations respectively. An LSTM layer consists

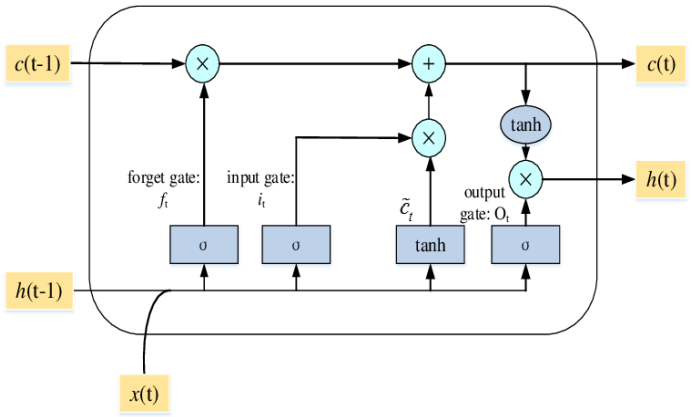


Figure 5: Architecture of a Long Short-Term Memory (LSTM) layer.[5]

of a cell state, forget gate, input gate and an output gate. The 'input' of an LSTM layer at period t consists of both the covariate values at time t ($\mathbf{x}_t$) and the output of the layer at period $t-1$ (h_{t-1}). The cell state (upper horizontal line) is able to memorise the past information, and is thus the 'memory' of the layer. The cell state (memory) is also transferred from t to $t-1$, but that does not fall under the 'input'. Since not all past information stays relevant, there is a forget gate which helps distinguish what information is relevant. As we can see, the forget gate consists of a logistic activation function, which maps the inputs between 0 and 1. Then, it performs an element-wise multiplication with the cell state. For elements in the forget gate close to zero, the corresponding information in the cell state is 'forgotten'. On the other hand, for elements in the forget gate which are close to 1, the corresponding information in the cell state is transferred through the layer and stays in the memory.

Besides forgetting irrelevant information, the cell state also needs to be updated for every new input, this happens through the input gate. The first step in the input gate is putting the input

[5]From https://www.researchgate.net/figure/The-structure-of-the-LSTM-unit_fig2_331421650, date accessed, July 1, 2021

data into two different activation functions, the logistic and *tanh* functions. Then, the outputs of both activation functions are combined using an element-wise multiplication. Since the logistic activation function maps inputs between 0 and 1, and *tanh* maps inputs between -1 and 1, the element-wise multiplication will result in values between -1 and 1. The input gate is then element-wise added to the cell state (memory).

The values of the cell state after this addition are used for two purposes. They are passed to the next time step, and, after going through the *tanh* function, they are element-wise multiplied with the output gate to construct an output. The output gate consists of putting the input data into a logistic activation function. Similar to the input gate, this element-wise multiplication will lead to values between -1 and 1. The values of the output gate are used both as the current output of the layer, and as the input of the layer 1-period ahead. Note that, similar to the 'normal' RNN, weight matrices are used to form linear combinations of inputs. Algorithm 1 describes analytically

Algorithm 1: Inner workings of a 1-layer LSTM model

Initiate with $h_0 = 0$ and $c(0) = 0$;

Given the input $\mathbf{x}_t$, **for** $t \in \{1, ..., T\}$ **do**

$$f_t = \sigma(W_f \mathbf{x}_t + U_f h_{t-1} + b_f);$$
$$i_t = \sigma(W_i \mathbf{x}_t + U_i h_{t-1} + b_i);$$
$$\tilde{c}_t = tanh(W_c \mathbf{x}_t + U_c h_{t-1} + b_c);$$
$$o_t = \sigma(W_o \mathbf{x}_t + U_o h_{t-1} + b_o);$$
$$c(t) = [f_t \odot c(t-1)] + [i_t \odot \tilde{c}_t];$$
$$h_t = o_t \odot tanh[c(t)];$$
$$\hat{y}_t = W_y h_t + b_y;$$

end

Where $W_f, W_i, W_c, W_o, W_y, U_f, U_i, U_c$, U_o, b_f, b_i, b_c, b_o and b_y are parameters to be estimated.

how a 1-layer LSTM layer model works. The difference between LSTM and a 'normal' RNN is the cell state (memory) which makes LSTM able to have a long memory, note that removing the upper half of Figure 5 results in a 'normal' RNN architecture.

There are multiple variations for input and output data structures when using LSTM. We use a structure for which the input data consists of 12 lags, that is, $\mathbf{X}_{t-1} = (\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-12})$. Using $\mathbf{X}_{t-1}$, the model predicts 1 to 12-months ahead inflation, that is, $\hat{\mathbf{y}}_{t|t-1} = (\hat{y}_{t|t-1}, \hat{y}_{t+1|t-1}, ..., \hat{y}_{t+11|t-1})$. Appendix D explains which hyperparameter values need to be chosen and how this choice is made.

4.6 Model Comparison

After obtaining forecasting results, it is useful to know if the quality of the results differ significantly. One of the first widely accepted methods to perform forecast comparison is that of Diebold & Mariano (1995). The so-called Diebold-Mariano (DM) test tests the null hypothesis of no difference in accuracy of two competing forecasting models against either a one or two-sided alternative. Although McCracken (2020) has shown that the DM test statistic may diverge in some cases, Giacomini & White (2006) have shown that the test is valid if the model parameters are estimated using a rolling window scheme. Since our research uses a rolling window for parameter estimation, the DM test statistic will not diverge.

Another criticism on the DM test is that for the DM test, the distribution of the test statistic under the null hypothesis is determined asymptotically. This asymptotic distribution may not always generalise well under finite samples. Therefore, Harvey et al. (1997) propose a variation of the DM test for which a correction for small samples is used, the Harvey-Leybourne-Newbold (HLN) test. We use the HLN test for checking whether the Machine Learning models (RF and LSTM) consistently perform better than the benchmark models (RW and AR). Moreover, we test whether the AR model performs better than the RW model.

The disadvantage of the DM/HLN test statistic is that it does not tell us anything about how the models perform over time, it gives a single value for the entire forecasting period. It may be interesting to see how the relative model performances behave within the forecasting sample. Therefore, we consider another method for model comparison, namely the Fluctuation Test (Giacomini & Rossi, 2010; Rossi, 2013). The Fluctuation Test analyses whether the ranking of two forecasts is stable over time. The test is very similar to the DM test procedure, except that it is performed over the forecasts within a rolling window of fixed size m. This test is especially relevant for the forecasting results of the US inflation, because there seems to be a structural break in the variance of the inflation around 2001. Therefore, there may also be a structural break in the predictive accuracy of the models. The HLN test could capture this structural break if it happens from one month to another. But it seems like the increase in variance happens more gradually, making the Fluctuation test more suitable in this case. We use the Fluctuation test on models with similar performance, to check if one model may temporarily perform better than the other, and if these fluctuations can be explained.

5 Results

This section presents the results of the research by first describing the results for the US data. Next, we discuss the results for the Canadian data. Finally, we compare the results for the US and Canadian data and attempt to give possible explanations for similarities and differences in the outcomes. All ML forecasts are obtained in R. For estimating the RFs, we use the *RandomForest* package. For estimating the LSTMs, we use the *Keras* package, together with *Tensorflow*[6].

5.1 US Data (FRED-MD)

For the US data, we use the same data and benchmark models as Medeiros et al. (2021). Therefore, our results for RW, AR and RF are the same as theirs. However, we present new results for the LSTM model using US data. After applying Bayesian Hyperparameter Optimisation (Appendix D) on the US dataset, we find the optimal number of neurons in the first and second hidden layer to be 119 and 7 respectively. Panel A of Table 1 presents the results for the four models for the entire forecasting sample. Appendix E shows results for two subsamples because there appears to be a structural break near 2001, which means that the results may differ significantly. We can see that LSTM performs relatively badly for a small horizon (1 or 2-months ahead). For the other horizons, both RF and LSTM consistently outperform RW in all samples. In the first subsample, the ML models (RF and LSTM) seem to also outperform the AR model consistently, but these differences are rarely significant following a 5% significance level for the HLN test. Both for the second subsample and the complete sample, the differences between the AR model and the ML models are consistently significant. These differences between the results for the first sample and the second sample could potentially have (a combination of) two explanations. Either the ML models perform better on the second sample, or the benchmark models (RW, AR) perform worse on the second sample. The latter explanation seems to be the more plausible. This has to do with the variance of the CPI for each period. As discussed in Section 3, the variance of the CPI increases from the first to the second sample. Since the benchmark models are relatively simple models that only use autoregressive terms, the expected squared error of the models (especially for RW) is positively related to the variance of the dependent variable. ML models use a big set of explanatory variables, these variables could be able to deal with the higher variance better, which explains why the ML models perform relatively better.

[6]The packages are available at https://cran.r-project.org/web/packages/.

The figures in Appendix F show the 1, 3, 6 and 12-month ahead forecasts for the different models together with the actual CPI. The RW model is excluded because it has the same form as the actual inflation, shifted to the right. Including forecasts of the RW model reduces the clarity of the figure substantially, without adding much information. For most of the sample, the forecasts of the three models behave somewhat similar. The ML models are in some cases able to capture immediate shocks (e.g. near the end of 2006), whereas the AR model seems to almost never capture those kinds of shocks.

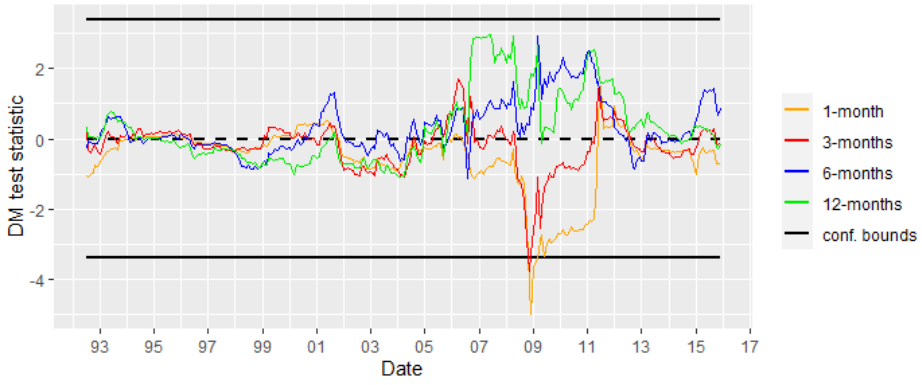


Figure 6: Fluctuation test for RF and LSTM with a 30-month window size for different horizons using US data, confidence bounds are of 5%.

For comparing the RF forecasts to the LSTM forecasts, we use the Fluctuation Test for 1, 3, 6 and 12-month ahead forecasts. Figure 6 shows the results of this test using squared errors. Values of the DM test statistic of 0 imply equal accuracy. For values above the upper confidence bound, LSTM produces significantly more accurate forecasts in the past 30 months. For values below the lower confidence bound, RF produces significantly more accurate forecasts in the past 30 months. From Figure 6, we can see that in the first sample (1990-2000), for all forecast horizons LSTM and RF perform almost equally well. After 2005, bigger differences between forecast horizons begin to emerge. For the shorter horizons (1 and 3-months), RF seems to be more accurate, performing significantly better for a few periods near the end of 2008. On the other hand, for the longer horizons (6 and 12-months), LSTM seems to be the more accurate model, but for none of the periods LSTM performs significantly better than RF.

5.2 Canadian Data (LCDMA)

Applying Bayesian Hyperparameter Optimisation to the Canadian data results in an optimal number of neurons of 31 and 128 for the first and second hidden layer respectively. Panel B of Table 1 presents the results of the four models for the entire forecasting sample. We can see that, for all forecasting horizons, the RW model gets consistently significantly outperformed by the AR, RF and LSTM models. This could have to do with the fact that the variance of the Canadian CPI is even higher than the variance of the second subsample of the US CPI. As mentioned before, the squared errors of the benchmark models (especially the RW model) are positively related to the variance of the dependent variable. It is remarkable that the AR model performs so well under these circumstances. The performance of the AR model is similar to that of both ML models. Although the AR model does not achieve the lowest RMSE for any of the horizons, its RMSE is lower than that of LSTM for some horizons. Also, both RF and LSTM fail to significantly perform better than the AR model consistently, with LSTM performing significantly better only once.

The figures in Appendix G show the 1, 3, 6 and 12-month ahead forecasts for the different models together with the actual CPI. Again, the RW model is excluded because it has the exact same form as the actual CPI. From these figures, it seems like the forecasts of the LSTM model are less conservative than those of the AR and RF models. Where the AR and RF models both form predictions with relatively low magnitude, the predictions of the LSTM model seem to have spikes with magnitudes comparable to the actual inflation. Indeed, the volatility (σ) of the 12-month ahead forecasts using the LSTM model is more than 1.5 times larger than the σ for the AR model forecasts and more than 2 times larger than the σ for the RF model forecasts.

For comparing the RF forecasts to the LSTM forecasts, we use the Fluctuation Test for 1, 3, 6 and 12-month ahead forecasts. We also compare both models to the AR model because the results of the ML models and the AR model were relatively similar. Figure 7 shows the results of this test for RF and LSTM using squared errors. The figures in Appendix H show the results of this test for the ML models and the AR model using squared errors. For the RF and LSTM models, for all horizons, the differences are insignificant. Even though the differences are insignificant, it seems like there is an upward trend for all horizons. This means that the LSTM model produces better forecasts towards the end of the sample relative to the forecasts of the RF model. Between 2009 and 2012, the forecast horizon seems to have a bigger influence on the relative model accuracy. LSTM favors shorter horizons, whereas RF favors longer horizons. This could have to do with the relatively high variance in the LSTM forecasts. LSTM produces forecasts with magnitudes closer

to the real behaviour of the CPI, while RF forecasts are consistently close to the mean of the CPI without deviating much. These 'more risky' LSTM forecasts often have a greater probability of succeeding for shorter horizons because it is easier to predict fluctuations in the near future. For longer horizons, these fluctuations are harder to predict. However, predicting the long-term mean is of approximately equal difficulty for all horizons. That could be the reason why RF predictions are relatively more accurate for longer horizons.

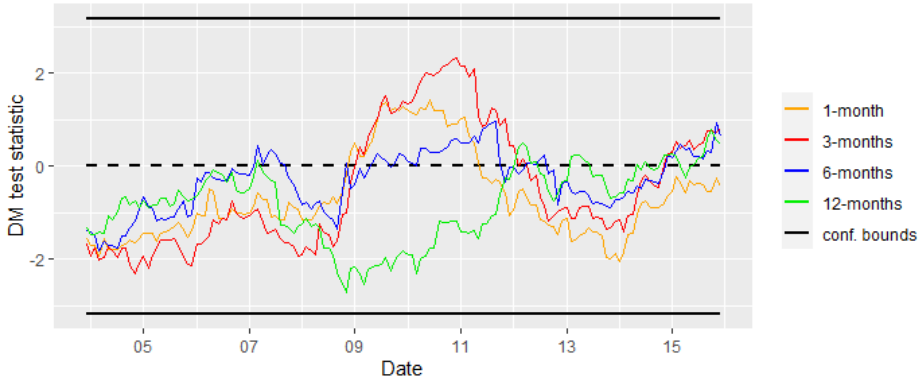


Figure 7: Fluctuation test for RF and LSTM with a 35-month window size for different horizons using Canadese data, confidence bounds are of 5%.

From the figures in Appendix H we can see that the relation between LSTM and AR forecasts is somewhat similar to the relation between LSTM and RF forecasts. This further strengthens the aforementioned possible explanation of the behaviour observed in the Fluctuation Test results for RF and LSTM forecasts. The AR model produces forecasts which behave similar to those of RF, they seem to represent the long-term mean of the real CPI without deviating much from it. Therefore, getting similar results for the Fluctuation Test decreases the probability that the results are solely based on randomness.

The relation between RF and AR forecasts seems more stable. Although there are no significant differences, the RF model produces more accurate outcomes consistently for most horizons.

5.3 Comparison of the Results

We will now discuss the similarities and differences of the results for the different datasets.

The first similarity is that for both the US data and the Canadian data, the ML models consistently perform significantly better than the RW model. Secondly, for both datasets, the performances of both ML models are relatively similar. Moreover, the Fluctuation Test returns (almost) no

significant values, implying that the performances are similar over the entire sample for the different horizons.

The first difference is that for the US data, the ML models consistently performed significantly better than the AR model, whereas the ML models were unable to obtain these results for the Canadian data. This could have (a combination of) multiple reasons. Firstly, ML models usually need relatively much training data to perform well out of sample. For the Canadian data, there is less training data available which could be the reason that ML fails to consistently perform better than the AR model. Secondly, it could be that the Canadian data contains too many covariates. Although both ML models generally perform well with many covariates, a combination of an insufficient number of observations and too many covariates can lead the models to overfit and perform worse out of sample. It could also be that the results obtained by Medeiros et al. (2021) and our results for LSTM using the US dataset do not generalise well to other datasets, without the number of observations/covariates being of importance. The second difference is that for the US data, LSTM seems to perform relatively better for larger horizons, whereas for Canadian data, LSTM seems to perform relatively better for smaller horizons. This could have to do with the variance of the forecasts. For the Canadian data, the forecasts seem to fluctuate more than for the US data. These fluctuations are relatively more accurate for shorter horizons, whereas the performance of more steady forecasts is less dependent on the horizon, which makes steady forecasts perform relatively better for longer horizons.

Table 1: Results for forecasting CPI using different models, horizons and datasets.

Panel A: US CPI 1990-2015 RMSE ratio

	Forecast horizon (months)														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
RW	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00$^{\bullet}$
AR	0.90*	0.81*	0.79*	0.81*	0.79*	0.79*	0.78*	0.76*	0.78*	0.82*	0.84*	0.75*	0.86*	0.97	1.22
RF	**0.85**$^{*\bullet}$	**0.73**$^{*\bullet}$	**0.71**$^{*\bullet}$	0.74$^{*\bullet}$	0.71$^{*\bullet}$	0.71$^{*\bullet}$	0.72$^{*\bullet}$	0.71*	0.72$^{*\bullet}$	0.76$^{*\bullet}$	0.78$^{*\bullet}$	0.68$^{*\bullet}$	**0.71**$^{*\bullet}$	**0.71**$^{*\bullet}$	**0.77**$^{*\bullet}$
LSTM	0.93	0.74$^{*\bullet}$	0.71$^{*\bullet}$	**0.73**$^{*\bullet}$	**0.69**$^{*\bullet}$	**0.70**$^{*\bullet}$	**0.71**$^{*\bullet}$	**0.69***	**0.71***	**0.74***	**0.77**$^{*\bullet}$	**0.67**$^{*\bullet}$	0.78$^{*\bullet}$	0.76$^{\bullet}$	0.78$^{\bullet}$

Panel B: Canadian CPI 2001-2015 RMSE ratio

	Forecast horizon (months)														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
RW	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
AR	0.82*	0.72*	0.70*	0.67*	0.66*	0.68*	0.64*	0.65*	0.66*	0.72*	0.78*	0.82*	0.69*	0.59*	0.76*
RF	**0.78**$^{*\bullet}$	**0.69**$^{*\bullet}$	**0.67**$^{*\bullet}$	**0.66***	**0.65***	**0.68***	0.63*	**0.63**$^{*\bullet}$	**0.66***	**0.71**$^{*\bullet}$	0.76$^{*\bullet}$	**0.81**$^{*\bullet}$	**0.64**$^{*\bullet}$	**0.56***	**0.69***
LSTM	0.81*	0.70*	0.68*	0.68*	0.65*	0.69*	**0.62***	0.64*	0.67*	0.75*	**0.76**$^{*\bullet}$	0.83*	0.67*	0.64*	0.83*

[1] 3m, 6m and 12m denote the accumulated forecasts for 3, 6, and 12-months ahead respectively.

[2] *: Better performance than RW model on a 5% significance level using the one-sided HLN test with squared errors.

[3] $^{\bullet}$: Better performance than the AR model on a 5% significance level using the one-sided HLN test with squared errors.

6 Conclusion

This paper investigates the performance of Random Forest (RF) and Long Short-Term Memory (LSTM) models in forecasting inflation. It compares the models to Random Walk (RW) and Autoregressive (AR) models and it compares the models to each other.

We analyze these questions using the following methods. First, we use Bayesian Hyperparameter Optimisation to find the optimal number of neurons for the LSTM model. Then, using a rolling window, we make 1 to 12-month ahead forecasts of the inflation. The forecasts are compared using the Root Mean Squared Error (RMSE). We determine significance of differences in forecast accuracy using the Harvey-Leybourne-Newbold (HLN) test, a variation of the Diebold-Mariano (DM) test with corrections for small samples. Finally, we investigate whether the differences in forecast accuracy change over time using the Fluctuation Test, a moving-window variation of the DM test.

The methodology as described above is applied on two high-dimensional datasets: the FRED-MD dataset containing US macroeconomic data, and the LCDMA dataset containing Canadian macroeconomic data. Both of these datasets also contain the inflation as measured by the Consumer Price Index (CPI).

For the US dataset, we find results indicating that both ML models consistently perform significantly better than both benchmark models (RW and AR model). The LSTM model has a worse performance for short horizons (1 and 2-months), but performs well for the other horizons, especially for longer horizons (12-months). The RF model performs well for all horizons. Apart from the short horizons, the performances of RF and LSTM are relatively similar.

For the Canadian dataset, we find results indicating that both of the ML models and the AR model consistently perform significantly better than the RW model. The results differ from the results for the US dataset in multiple ways. Firstly, both ML models fail to consistently perform significantly better than the AR model. This could have to do with the lack of training data or an overload of covariates. Secondly, LSTM forecasts seem to perform better for short horizons, whereas, for the US data, this was the other way around. The reason for this could be that forecasts using LSTM fluctuate more for the Canadian data. Generally, fluctuations can be predicted more accurately for short horizons. The performance of forecasts with relatively little fluctuations are generally less dependent on the horizon.

All in all, the ML models seem to perform better than the RW model almost always. It seems like

this observation can be generalised to other datasets as well, because the performance of the RW model has not been able to compete with the ML models for almost all horizons in both of the datasets. The relative performance of the ML models and the AR model is more complicated. It seems that we can not make a conclusion that can be generalised about whether the ML models perform better than the AR model. It highly depends on the dataset whether the ML models are able to consistently perform better than the AR model. Lastly, the performances of RF and LSTM are relatively similar.

There are several interesting avenues for extensions to this research in the future. Firstly, if there are less time constraints, more of the hyperparameter values could be optimized for both LSTM and RF. Secondly, for the LSTM model, (multiple) non-LSTM hidden layers (e.g. Feed Forward) could be added to the model. This may increase the accuracy because more complicated non-linearities in the data can be captured when adding additional layers. Instead of using the *Dropout* method for training the LSTM model, the *MC Dropout* method (Gal & Ghahramani, 2016) could be used. Some researches have shown that for using Dropout, there is still a chance of overfitting the model. Therefore, MC Dropout was introduced as an improvement on the original Dropout.

Bibliography

Almosova, A. & Andresen, N. (2019). Nonlinear inflation forecasting with recurrent neural networks. *May 2019*.

Atkeson, A., Ohanian, L. E., et al. (2001). Are phillips curves useful for forecasting inflation? *Federal Reserve bank of Minneapolis quarterly review*, 25(1), 2–11.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5–32.

Broomhead, D. S. & Lowe, D. (1988). *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

Chakraborty, C. & Joseph, A. (2017). Machine learning at central banks.

Choudhary, M. A. & Haider, A. (2012). Neural network models for inflation forecasting: an appraisal. *Applied Economics*, 44(20), 2631–2635.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Diebold, F. X. & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13(3), 253–263.

Fortin-Gagnon, O., Leroux, M., Stevanovic, D., & Surprenant, S. (2018). *A large canadian database for macroeconomic analysis*. Centre interuniversitaire de recherche en analyse des organisations.

Gal, Y. & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).: PMLR.

Giacomini, R. & Rossi, B. (2010). Forecast comparisons in unstable environments. *Journal of Applied Econometrics*, 25(4), 595–620.

Giacomini, R. & White, H. (2006). Tests of conditional predictive ability. *Econometrica*, 74(6), 1545–1578.

Giannone, D., Lenza, M., & Primiceri, G. E. (2015). Prior selection for vector autoregressions. *Review of Economics and Statistics*, 97(2), 436–451.

Harvey, D., Leybourne, S., & Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of forecasting*, 13(2), 281–291.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.

Hongwei, G., Zhuang, X., Liang, D., & Rabczuk, T. (2020). Stochastic groundwater flow analysis in heterogeneous aquifer with modified neural architecture search (nas) based physics-informed neural networks using transfer learning.

Ludvigson, S. C. & Ng, S. (2007). The empirical risk–return relation: A factor analysis approach. *Journal of Financial Economics*, 83(1), 171–222.

Masini, R. P., Medeiros, M. C., & Mendes, E. F. (2020). Machine learning advances for time series forecasting. *arXiv preprint arXiv:2012.12802*.

McCracken, M. W. (2020). Diverging tests of equal predictive ability. *Econometrica*, 88(4), 1753–1754.

McCracken, M. W. & Ng, S. (2016). Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4), 574–589.

Medeiros, M. C., Vasconcelos, G. F. R., Álvaro Veiga, & Zilberman, E. (2021). Forecasting inflation in a data-rich environment: The benefits of machine learning methods. *Journal of Business & Economic Statistics*, 39(1), 98–119.

Nakamura, E. (2005). Inflation forecasting using a neural network. *Economics Letters*, 86(3), 373–378.

Paranhos, L. (2021). Predicting inflation with neural networks. *arXiv preprint arXiv:2104.03757*.

Rossi, B. (2013). Advances in forecasting under instability. In *Handbook of economic forecasting*, volume 2 (pp. 1203–1324). Elsevier.

Sermpinis, G., Stasinakis, C., Theofilatos, K., & Karathanasopoulos, A. (2014). Inflation and unemployment forecasting with genetic support vector regression. *Journal of Forecasting*, 33(6), 471–487.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929–1958.

Stock, J. H. & Watson, M. W. (1999). Forecasting inflation. *Journal of Monetary Economics*, 44(2), 293–335.

Stock, J. H. & Watson, M. W. (2002). Macroeconomic forecasting using diffusion indexes. *Journal*

of Business & Economic Statistics, 20(2), 147–162.

Stock, J. H. & Watson, M. W. (2007). Why has us inflation become harder to forecast? *Journal of Money, Credit and banking*, 39, 3–33.

Teräsvirta, T., Van Dijk, D., & Medeiros, M. C. (2005). Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A re-examination. *International Journal of Forecasting*, 21(4), 755–774.

Yarotsky, D. (2017). Error bounds for approximations with deep relu networks. *Neural Networks*, 94, 103–114.

Appendix A Description of Random Forests

To understand the RF method better, it is important to first understand how regression trees work. A regression tree splits the explanatory variables (x) in intervals and creates a prediction for the dependent variable (y) for each interval. Each interval is represented by a node in the regression tree. The nodes at which no more splits are performed are called terminal nodes.

Given a number of terminal nodes K and a forecast horizon h, the splits are determined by minimizing the sum of squared errors of the following regression model:

$$y_{t+h} = \sum_{k=1}^{K} c_k I_k(\mathbf{x}_t; \boldsymbol{\theta}_k), \tag{3}$$

where I_k denotes an indicator function such that:

$$I_k(\mathbf{x}_t; \boldsymbol{\theta}_k) = \begin{cases} 1, & \text{if } \mathbf{x}_t \in R_k(\boldsymbol{\theta}_k), \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\boldsymbol{\theta}_k$ is the set of parameters that define the k-th region R_k. The parameter c_k determines what the forecast for y_{t+h} is if $\mathbf{x}_t$ falls in region k.

To prevent overfitting, different heuristics are introduced that stop the tree from splitting any further. This is necessary because variance increases sharply if trees grow too big while the bias decreases very slowly, which means that the SSE of test data would increase (even though the SSE of training data still decreases). An example of such an heuristic is that each node/interval should contain a minimum number of observations in the training data.

We will use a 2-dimensional example to give a visual representation of regression trees. In this example, y is the dependent variable and x the explanatory variable. Suppose the real data generating process (DGP) of y is a sigmoid function with normally distributed errors (see Figure 8). Figure 9a shows in which way the data is split using a regression tree and, at the bottom, which y-value belongs to each interval for x. Figure 9b shows for each x-value what the regression tree predicts the y-value to be. This figure shows that regression trees can capture non-linear relationships between x and y, although in a simplistic manner. Note that Figures 9a and 9b can be linked to each other, each horizontal line/interval in Figure 9b corresponds to one of the leaf nodes in Figure 9a.

The disadvantage of regression trees is that they can only capture a very simplified version of functions without having to deal with a sharp increase of the variance. Random Forest (RF) was introduced to overcome this problem. As discussed, RF is a bagging method. It uses bootstrapping

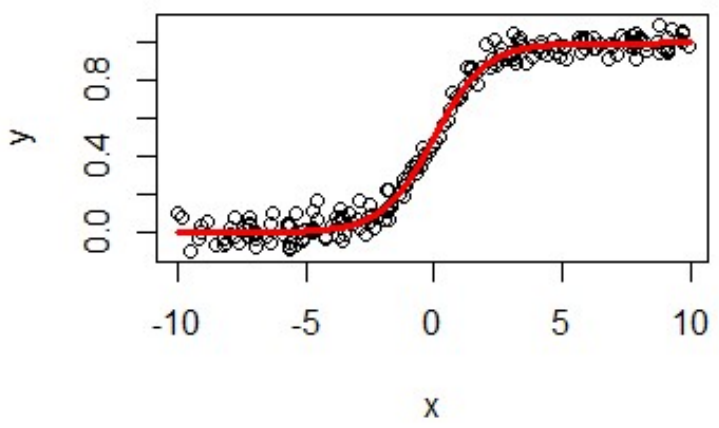
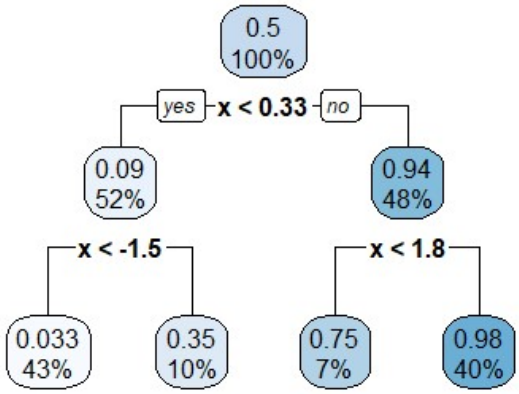
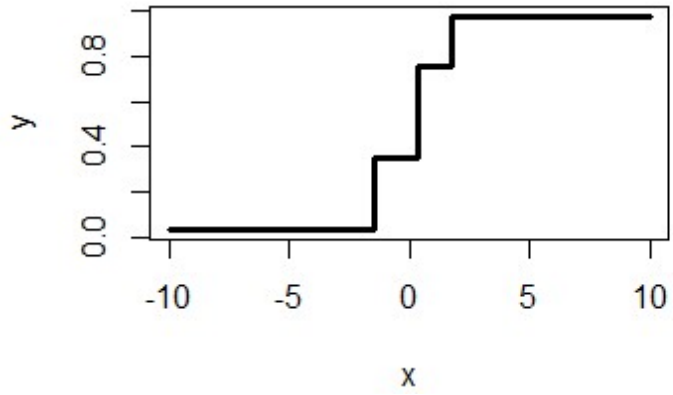


Figure 8: The data points used in the example, the red line denotes the sigmoid function.



(a) Splitting points in the tree.



(b) Predicted y for all x-values using the regression tree.

Figure 9: Regression tree example.

to create B regression trees, and aggregates predictions of these regression trees to form a prediction. Each regression tree is trained using a random subset of covariates. Moreover, assuming we have T training observations, the training data for each regression tree consists of T observations which are sampled with replacement from the original training observations. The forecasts of a RF are produced by averaging the forecasts of all B regression trees, that is:

$$\hat{y}_{t+h} = \frac{1}{B}\sum_{b=1}^{B}\left\{\sum_{k=1}^{K_b} c_k I_k(\mathbf{x}_t;\boldsymbol{\theta}_k)\right\}, \tag{5}$$

where K_b is the number of terminal nodes of the regression tree constructed in the b-th bootstrap step.

For the previous example (Figure 8), Figure 10 shows for each x-value what the predicted y-value

would be for a RF ($B = 500$). We can see that the RF is better able to capture the original DGP than the regression tree, which shows the power of using bagging on multiple regression trees (RF) rather than only using one regression tree.

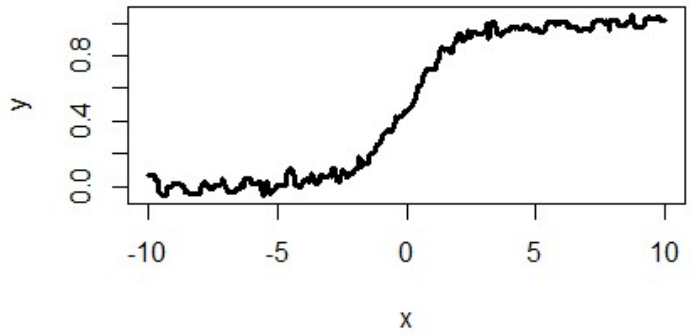


Figure 10: Predicted y for all x-values using the random forest.

Appendix B Description of Feed Forward Neural Network

For a Feed Forward NN, the values that are passed to the first hidden layer, are a linear combination of the covariates: $\gamma_{0,j} + \boldsymbol{\gamma}_j' \mathbf{X}_t$, $j = 1, ..., m_1$. Where m_1 is the amount of neurons in the hidden layer and $\gamma_{0,j}$ denotes the bias. Inside the neurons of the hidden layer, the linear combinations are put in an activation function $S(\cdot)$. For a single hidden layer Feed Forward NN (Figure 11a), the neuron(s) in the output layer then receive a linear combination of the output from the neurons in the hidden layer plus a bias as input. This leads to the following approximation function for the k-th neuron in the output layer:

$$g_{D_k}(\mathbf{X}) := g_{D_k}(\mathbf{X}, \boldsymbol{\theta}_k) = \beta_{0,k} + \sum_{j=1}^{m_1} \beta_{j,k} S(\gamma_{0,j} + \boldsymbol{\gamma}_j' \mathbf{X}). \qquad k = 1, ..., m_2 \tag{6}$$

Where m_2 is the amount of neurons in the output layer and
$\boldsymbol{\theta}_k = (\beta_{0,k}, ..., \beta_{m_1,k}, \boldsymbol{\gamma}_1', ..., \boldsymbol{\gamma}_{m_1}', \gamma_{0,1}, ..., \gamma_{0,m_1})$ denotes the set of parameters. The parameter values, $\hat{\boldsymbol{\theta}}_k$, are estimated by minimizing the squared error. Since $\boldsymbol{\theta}_k$ is unrestricted and can become very high-dimensional (especially for multiple hidden layers), this minimization problem may cause over-fitting to take place. To prevent this, Srivastava et al. (2014) introduced a method named *Dropout*. Dropout randomly drops neurons (including their connections) from the network in the parameter estimation process. There are 2^{m_1} different ways to do this. Dropout first samples from these 2^{m_1} available 'thinned' NNs. After that, it trains the sampled thinned NNs. Finally, combining the trained thinned NNs, it trains an unthinned NN. The unthinned NN will function as the forecasting model.

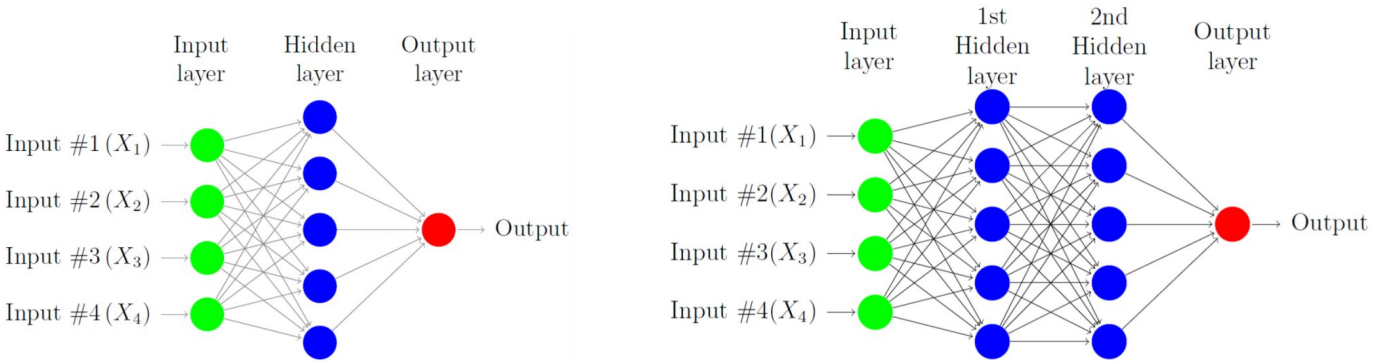
Often, squashing functions are used as activation functions. Squashing functions are non-decreasing functions that will not pass some finite upper and lower bound, regardless of the input. The two historically most popular choices are the logistic and hyperbolic tangent functions, such that:

$$\begin{aligned} \text{Logistic: } S(x) = \sigma(x) &= \frac{1}{1 + \exp(-x)} \\ \text{Hyperbolic Tangent: } S(x) = tanh(x) &= \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \end{aligned} \tag{7}$$

Other activation functions that gained popularity in recent years are the Radial-Basis Function (RBF) (Broomhead & Lowe, 1988) and the Rectified Linear Units (ReLU) (Yarotsky, 2017).

For a NN with multiple hidden layers, the process of passing linear combinations of neuron values (plus biases) and putting them in an activation function is repeated multiple times. The Dropout method can also be applied to a NN with multiple hidden layers. Figure 11b shows the architecture

of a NN with 2 hidden layers, each consisting of 5 neurons, note that the amount of neurons across hidden layers does not need to be the same.



(a) Feed Forward NN with one hidden layer.

(b) Feed Forward NN with two hidden layers.

Figure 11: Feed Forward Neural Network architecture. From Masini et al. (2020).

Appendix C Vanishing/Exploding Gradients for Backpropagation Through Time

A problem with the gradient in the 'normal' RNN structure appears for the gradient of $Loss$ with respect to W (and U). This is because these gradients have a recursive structure:

$$\begin{aligned}\frac{\partial s_t}{\partial W} &= \frac{\partial s_t}{\partial (W s_{t-1} + U x_t + b_s)} \frac{\partial (W s_{t-1} + U x_t + b_s)}{\partial W} \\ &= S_s'(W s_{t-1} + U x_t + b_s)\,(s_{t-1} + W \frac{\partial s_{t-1}}{\partial W}).\end{aligned} \tag{8}$$

This leads to the following expression for the gradient of $Loss$ with respect to W:

$$\begin{aligned}\frac{\partial Loss_t}{\partial W} &= \frac{\partial Loss_t}{\partial o_t} \frac{\partial o_t}{\partial s_t} \frac{\partial s_t}{\partial W} \\ &= L'(o_t - y_t)\, S_o'(V s_t + b_o)\, V\, S_s'(W s_{t-1} + U x_t + b_s)\,(s_{t-1} + W \frac{\partial s_{t-1}}{\partial W}).\end{aligned} \tag{9}$$

The problem that may occur because of this recursion is that of exploding or vanishing gradients. The gradient has a recursive structure, and the recursive part is multiplied by W in every recursion step. Therefore, the gradient of $Loss_t$ with respect to W will eventually, among others, depend on W^t. This leads to the gradient eventually depending on some λ^t, which is related to W. How the λ is calculated and how it is related to W is outside the scope of this paper. What is important is that for $|\lambda| > 1$, the gradient will explode, and for $|\lambda| < 1$, the gradient will vanish. An exploding gradient causes the BPTT to fail because the gradient can take on values that are too large for a computer to handle. Also, the estimation process becomes unstable. A vanishing gradient also causes BPTT to fail. This is because BPTT changes values in the weight matrices according to the gradient. Therefore, vanishing gradients will cause the algorithm to stop learning.

Appendix D Hyperparameter Value Selection

For both RF and LSTM, some hyperparameter values need to be set before starting the training process. We will briefly discuss how the hyperparameter values are chosen for RF, after that, we will discuss how we chose the hyperparameter values for LSTM.

For RFs, we use the same hyperparameter values as Medeiros et al. (2021), because we are extending their work. Therefore, each tree in the RF only performs a split if the resulting leafs contain at least five observations in the training data. The proportion of variables selected in each bootstrapping step is $\frac{1}{3}$ and the number of bootstrap steps, B, is set to 500. They investigated using different hyperparameter values and came to the conclusion that the results are reasonably stable.

LSTM has, like most NNs, relatively many hyperparameters that need to be set before training the model. These hyperparameters are: *batch size*, *nepochs*, *lr* *nlayers*, *units*. The *batch size* determines how the training data is supplied to the LSTM algorithm. LSTM can not handle all observations at once, so the observations are supplied in batches of size *batch size*. A higher *batch size* is computationally faster, but leads to suboptimal outcomes and vice versa. The hyperparameter *nepochs* determines how often the training data is passed through the LSTM algorithm. It is necessary to pass the data through LSTM multiple times because LSTM does not yet obtain its most optimal parameter values after passing the data through it only once. A higher *nepochs* leads to more optimal outcomes, but is computationally slower and vice versa. Then, *lr* (learning rate) is a hyperparameter that controls how much to change the weight matrices in response to the estimated error and gradients each time the model updates its weights. Setting *lr* too high may lead to an unstable training process because the algorithm takes too big steps for each iteration, it may also lead to a sub-optimal set of weights. On the other hand, setting *lr* too low may result in a very slow training process that could get stuck on a sub-optimal set of weights. *nlayers* determines the number of hidden layers in the network, setting this lower gives opportunity to less non-linear functions to be captured, but also a lower chance of overfitting, and vice versa. Finally, *units* is a vector of hyperparameters that determines, for each hidden layer, how many neurons to use. The number of neurons in a layer determines the dimensions of the weight matrices in that layer. Thus, setting this higher, results in more parameters to be estimated. Setting *units* too high may lead to overfitting of the model. On the other hand, setting *units* too low may result in the model not being able to train sufficiently.

For none of these hyperparameters there exists a method to determine the optimal value. For *units*

we use Bayesian Optimisation to find the optimal values. This optimization algorithm is computationally very costly, therefore, with the limited amount of time we have, we decided to only use this for *units* and set the other hyperparameters manually. We use a *batch size* of 1, because increasing it leads to disproportionately worse results, even though it decreases the computation time. For *nepochs*, it is computationally not feasible to use a value higher than 50, given the *batch size* which is equal to 1. We set *nepochs* equal to 20 after tweaking it between 1 and 50 for the training sets of both datasets. For the *lr* (learning rate), we use the standard value of 0.001. Then, we set *nlayers* to 2 because we want to capture non-linearities in the inflation without overfitting the model. Using 2 hidden layers enables the model to capture sufficient non-linearities. For both hidden layers, the *Dropout* (Appendix B) method is used to prevent overfitting. Finally, for the *units* hyperparameter, which is a vector of 2 values because we set *nlayers* equal to 2, we use Bayesian Optimisation.

Bayesian Optimisation is an adaptive hyper-parametric search method that predicts the combination of values for the hyperparameters that is likely to give the most benefit based on the combination of hyperparameter values that have currently been tested Snoek et al. (2012). It assumes that the function of hyperparameter optimization $f(x)$ follows a Gaussian process, where x denotes the hyperparameter values. This means that $p(f(x)|x)$ follows a Normal distribution. Following the notation of Hongwei et al. (2020), The Bayesian Optimisation process is modelled as a Gaussian process, which is based on the results of N existing experiments. Each of the inputs for these experiments (x_n) led to a deviation between output and ground truth. Since the Bayesian Optimisation algorithm uses a maximisation function, we use the negative of the Root Mean Squared Error (RMSE) as $f(x)$. Then the experiments and their outcomes are grouped: $H = \{x_n, y_n\}_{n=1}^{N}$. Based on H, the posterior distribution of $f(x)$ is calculated: $p\left(f(x)|x, H\right)$. After the posterior distribution of the objective function is obtained, an acquisition function $a(x, H)$ needs to be defined to determine the next sample point. The next sample point is then defined as $x' = argmax_x\, a(x, H)$. For our research, we use an acquisition function as follows: $a(x, H) = E\left[f(x)|x, H\right] + \sigma\left[f(x)|x, H\right]$. Thus, x' is chosen at points with either a high uncertainty or a high expected accuracy $(f(x'))$, or a combination of both.

Before the process starts, we produce forecasts for 15 random points on the grid as initialisation. These points (x^*), together with the negative of their respective RMSE (y^*), are used as initialisation (H^*). Then we use $M = 35$ Bayesian Optimisation steps to find the optimal value for *units*. Algorithm 2 shows the Bayesian Optimisation process.

Algorithm 2: Bayesian Hyperparameter Optimisation

input : $f(x)$, M, $a(x, H)$, H^*;

$H = H^*$;

Model Gaussian process according to H, calculate $p\left(f(x)|x, H\right)$;

for $m \in \{1, ..., M\}$ **do**

  $x' = argmax_x\, a(x, H)$;

  evaluate $y' = f(x')$;

  $H = H \cup (x', y')$;

  Remodeling Gaussian process according to H, calculate $p\left(f(x)|x, H\right)$;

end

Output H

We apply the Bayesian Optimisation algorithm on the *units* hyperparameter. Both for US data and Canadian data, we use the training sample (part of the sample for which no forecasts will be calculated) to calculate the RMSE of the models. RNNs, including LSTM, are only able to predict 1 period ahead because of its recurrent structure. Therefore, the training and predicting structure in Figure 12 is used. We separately predict the last 5 periods in the training sample using all the observations up until the point of the predictions. For each prediction, a new model is estimated because the training data is slightly different. Since LSTM returns 12 values for each prediction (Section 4.5), we have a set of 60 predictions. For these predictions, the RMSE is calculated and the negative of this is returned as $f(x)$-value in each step of the Bayesian Optimisation process.

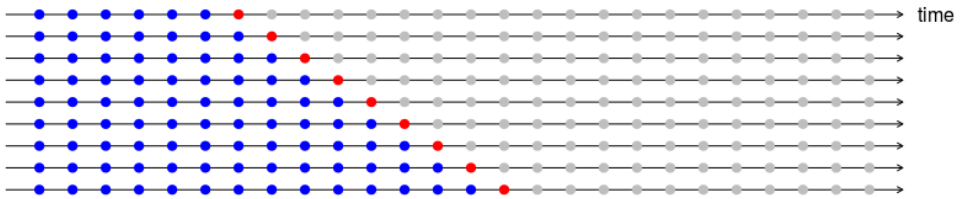


Figure 12: The training and testing process in Bayesian Optimisation, the blue dots are used for training, the red dot is predicted.[7]

[7]From https://rstudio-pubs-static.s3.amazonaws.com/351073_677a795d25d9418a843640940a2dacf5.html0, date accessed, July 1, 2021

Appendix E Forecasting Results US CPI Split in Two Samples

Because of the size of the table, it is presented on the next page.

Table 2: Results for forecasting US CPI using different models, windows and two subsamples.

Panel A: US CPI 1990-2000 RMSE ratio

	Forecast horizon (months)														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
RW	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00$^{\bullet}$	1.00$^{\bullet}$
AR	0.84*	0.82*	0.88*	0.82*	0.78*	0.79*	0.79*	0.8*	0.87*	0.89	0.95	0.85*	1.00	1.19	1.24
RF	**0.79**$^{*\bullet}$	**0.77**$^{*\bullet}$	0.85*	0.78*	0.73$^{*\bullet}$	**0.76***	**0.76***	**0.77***	**0.81**$^{*\bullet}$	0.83$^{*\bullet}$	**0.86**$^{*\bullet}$	**0.71**$^{*\bullet}$	**0.87**$^{\bullet}$	**0.98**$^{\bullet}$	**0.87**$^{\bullet}$
LSTM	0.85*	0.79*	**0.84***	**0.77***	**0.72***	0.77*	0.79*	0.81*	0.88	**0.82***	0.87	0.74$^{*\bullet}$	1.01	1.07	0.98$^{\bullet}$

Panel B: US CPI 2001-2015 RMSE ratio

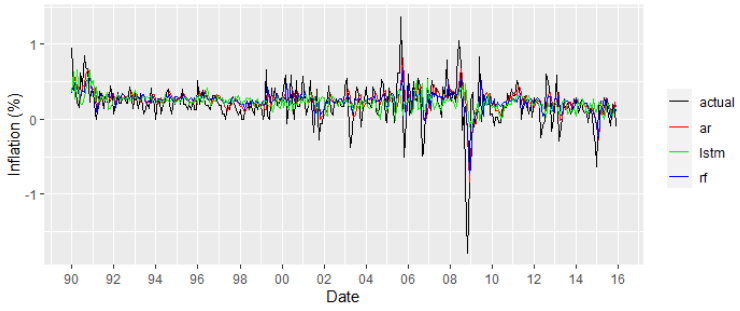
	Forecast horizon (months)														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
RW	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00$^{\bullet}$
AR	0.92	0.81*	0.78*	0.80*	0.79*	0.79*	0.78*	0.76*	0.77*	0.81*	0.82*	0.73*	0.85*	0.95	1.21
RF	**0.87**$^{*\bullet}$	**0.72**$^{*\bullet}$	**0.68**$^{*\bullet}$	0.73$^{*\bullet}$	0.71$^{*\bullet}$	0.71$^{*\bullet}$	0.71$^{*\bullet}$	0.70*	0.71$^{*\bullet}$	0.75$^{*\bullet}$	0.77$^{*\bullet}$	0.68$^{*\bullet}$	**0.69**$^{*\bullet}$	**0.68**$^{*\bullet}$	0.75$^{*\bullet}$
LSTM	0.95	0.74$^{*\bullet}$	0.69$^{*\bullet}$	**0.72**$^{*\bullet}$	**0.69**$^{*\bullet}$	**0.68**$^{*\bullet}$	**0.70**$^{*\bullet}$	**0.67**$^{*\bullet}$	**0.68**$^{*\bullet}$	**0.73***	**0.75**$^{*\bullet}$	**0.65**$^{*\bullet}$	0.75$^{*\bullet}$	0.73$^{*\bullet}$	**0.74**$^{*\bullet}$

[1] 3m, 6m and 12m denote the accumulated forecasts for 3, 6, and 12-months ahead respectively.

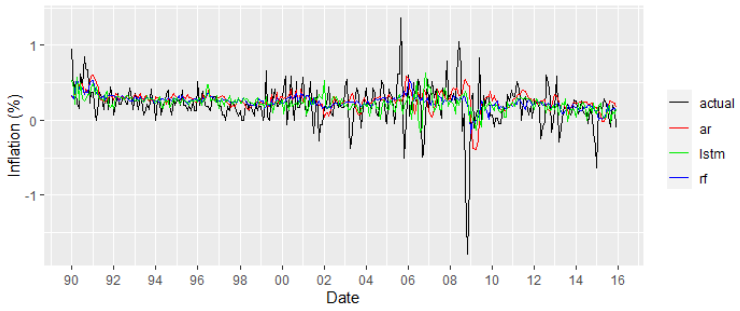
[2] *: Better performance than RW model on a 5% significance level using the one-sided HLN test with squared errors.

[3] $^{\bullet}$: Better performance than the AR model on a 5% significance level using the one-sided HLN test with squared errors.

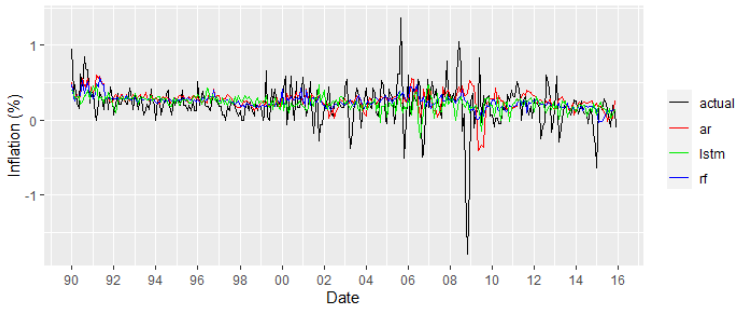
Appendix F CPI Forecasts for US data



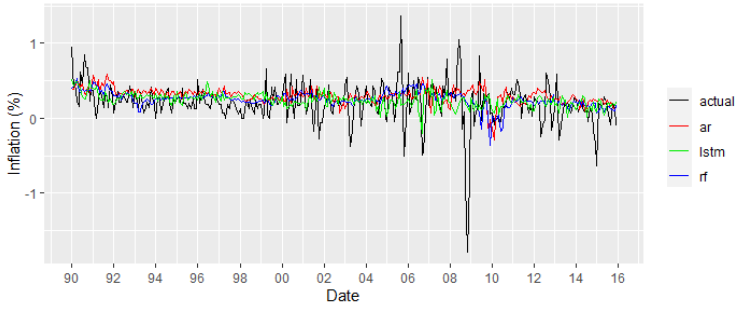
(a) 1-month ahead forecasts.



(b) 3-months ahead forecasts.



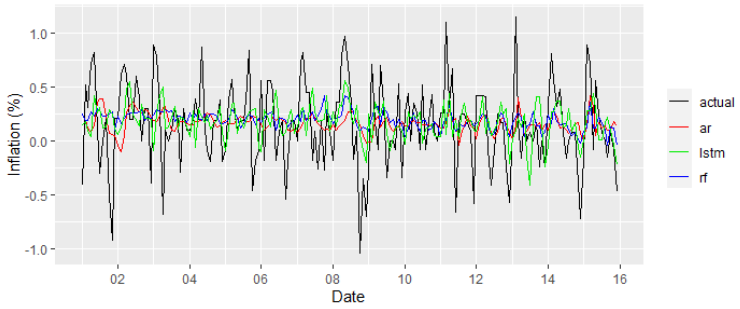
(c) 6-months ahead forecasts.



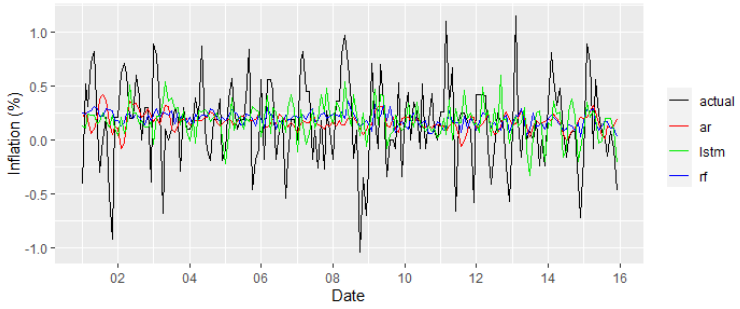
(d) 12-months ahead forecasts.

Figure 13: US CPI forecasts over the sample period for the different models.

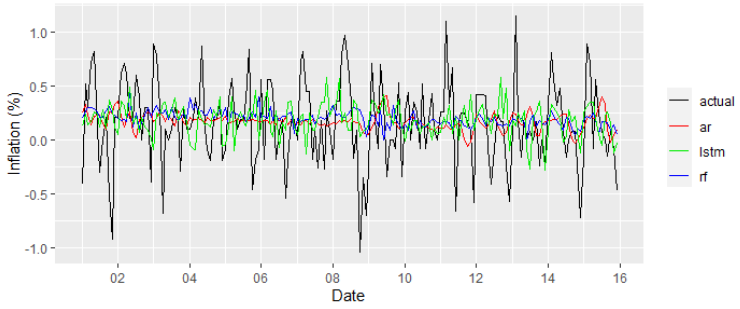
Appendix G CPI Forecasts for Canadian data



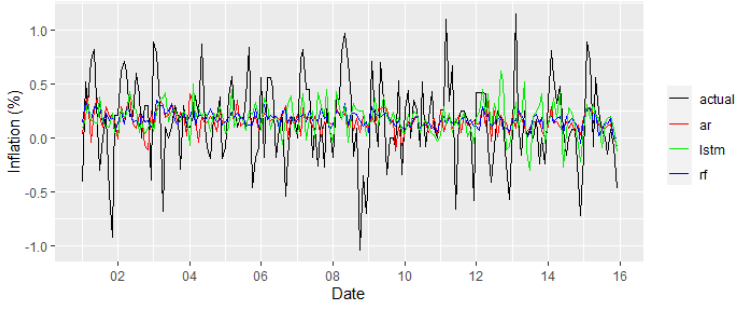
(a) 1-month ahead forecasts.



(b) 3-months ahead forecasts.



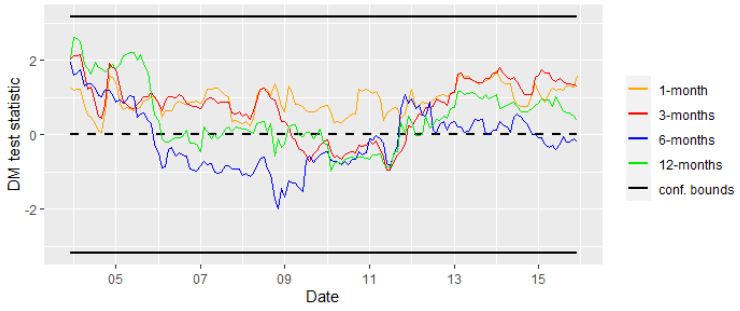
(c) 6-months ahead forecasts.



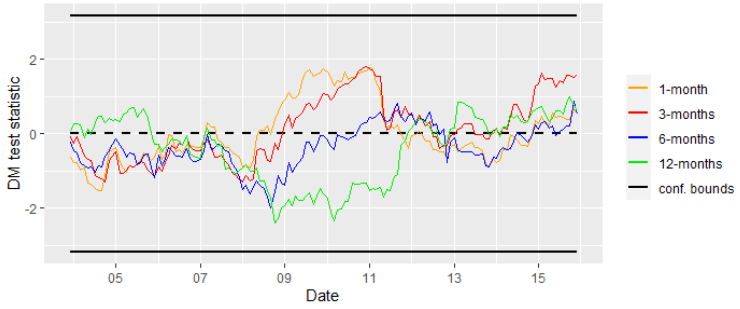
(d) 12-months ahead forecasts.

Figure 14: Canadian CPI forecasts over the sample period for the different models.

Appendix H Fluctuation Tests ML Models with AR Model



(a) RF against AR.



(b) LSTM against AR.

Figure 15: Fluctuation test for RF and LSTM against AR with window size 35 for different horizons using Canadese data, confidence bounds are of 5%.

Appendix I Replication

I decided to replicate Table 5 of the original paper because it has all the information which is present in Table 1. The tests and mean/median methods can not be replicated since I did not include all models because of the long running time. The RF results differ slightly because the authors did not give a seed for the second sample. Also, the results for the shrinkage models differ, the cause of this is unknown.

Table 3: Forecasting errors for the CPI from 1990 to 2015 (RMSE).

	Forecast horizon (months)														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
RW	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
AR	0.90	0.81	0.79	0.81	0.79	0.79	0.78	0.76	0.78	0.82	0.84	0.75	0.86	0.97	1.22
Factor	0.87	0.78	0.78	0.79	0.78	0.78	0.80	0.81	0.82	0.84	0.84	0.78	0.82	0.90	1.17
T.Factor	0.88	0.79	0.78	0.80	0.77	0.79	0.79	0.80	0.80	0.82	0.83	0.78	0.82	0.91	1.17
LASSO	0.90	0.80	0.72	0.81	0.77	0.81	0.87	0.95	0.94	1.05	1.05	0.94	0.75	0.82	1.31
adaLASSO	0.90	0.79	0.72	0.80	0.75	0.81	0.87	0.94	0.95	1.04	1.03	0.93	0.74	0.80	1.30
ElNet	0.92	0.77	0.74	0.77	0.76	0.81	0.84	0.95	0.95	1.00	1.03	0.88	0.76	0.82	1.29
adaElNet	0.92	0.76	0.73	0.77	0.75	0.80	0.83	0.94	0.95	0.99	1.01	0.89	0.75	0.79	1.26
Bagging	0.83	0.76	0.76	0.80	0.79	0.80	0.83	0.82	0.78	0.81	0.82	0.74	0.74	0.77	0.83
CSR	0.85	0.77	0.76	0.79	0.77	0.79	0.79	0.77	0.79	0.83	0.84	0.76	0.79	0.87	1.13
RF	0.85	0.73	0.71	0.74	0.71	0.71	0.72	0.71	0.72	0.76	0.78	0.68	0.71	0.71	0.77
adaLASSO/RF	0.85	0.75	0.72	0.73	0.74	0.72	0.72	0.71	0.72	0.79	0.82	0.70	0.73	0.73	0.80

Table 4: Forecasting errors for the CPI from 1990 to 2015 (MAE).

	Forecast horizon (months)														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
RW	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
AR	0.87	0.79	0.78	0.81	0.80	0.81	0.78	0.76	0.81	0.85	0.86	0.76	0.89	1.04	1.22
Factor	0.88	0.80	0.80	0.82	0.82	0.80	0.78	0.80	0.87	0.87	0.87	0.82	0.89	1.02	1.21
T.Factor	0.87	0.82	0.81	0.84	0.83	0.84	0.80	0.80	0.84	0.87	0.86	0.80	0.90	1.06	1.23
LASSO	0.87	0.78	0.74	0.85	0.80	0.85	0.89	1.00	1.05	1.15	1.15	1.00	0.79	0.95	1.42
adaLASSO	0.87	0.77	0.72	0.81	0.76	0.84	0.87	0.99	1.03	1.12	1.12	0.97	0.76	0.89	1.37
ElNet	0.89	0.78	0.74	0.81	0.80	0.84	0.87	1.01	1.06	1.09	1.14	0.96	0.79	0.94	1.41
adaElNet	0.89	0.77	0.72	0.79	0.77	0.82	0.84	0.99	1.03	1.06	1.09	0.94	0.78	0.87	1.33
Bagging	0.84	0.78	0.79	0.86	0.88	0.87	0.84	0.81	0.81	0.83	0.86	0.77	0.79	0.90	0.90
CSR	0.84	0.76	0.75	0.79	0.79	0.79	0.76	0.74	0.79	0.83	0.84	0.77	0.82	0.94	1.11
RF	0.82	0.72	0.71	0.75	0.73	0.73	0.70	0.68	0.71	0.75	0.78	0.66	0.74	0.77	0.78
adaLASSO/RF	0.82	0.73	0.72	0.74	0.74	0.73	0.71	0.68	0.72	0.79	0.81	0.68	0.76	0.80	0.82