Inflation Forecasting in Emerging Markets: A Machine Learning Approach

Kriti Mahajan & Anand Srinivasan*

Abstract

In developing and emerging economies, the accuracy of macroeconomic forecasts is often constrained by the limited availability of data both in time series and in cross-section. Given this constraint, this paper uses a suite of machine learning methods to explore if they can offer any improvements in forecast accuracy for headline CPI inflation (y-o-y) in 3 emerging market economies: India, China and South Africa. For each forecast horizon for each country, we use a host of machine learning models and compare the accuracy of each method to 2 benchmark models (namely, a moving average forecast and SARIMA). For India, we find that the deep neural networks out-perform the benchmark forecast for all horizons except the 1 month ahead forecast. The reduction in forecasting error ranges from 44% to 63%. For South Africa, the neural network model provides a reduction in forecasting error between 42% and 57% for the 1 year forecast. For China, the reduction in forecasting error is much more modest ranging from 5% to 33%. An average forecast using different neural net methods performs much better than any individual forecast.

JEL codes: C45, C52, C53, C54, E00

---

*Ms. Kriti Mahajan is a Research Associate at CAFRAL & Dr. Anand Srinivasan is Additional Director of Research at CAFRAL & Associate Professor of Finance at NUS Business School, National University of Singapore. The views expressed in this article are those of the authors, and do not necessarily reflect the views or position of CAFRAL or National University of Singapore. The authors can be contacted at: kriti.mahajan@cafral.org.in and bizas@nus.edu.sg or anand.srinivasan@cafral.org.in.

1. Introduction

One of the most important goals of central banking is to maintain a stable inflation rate. In an analysis of central bank objectives for 47 developed and developing economies, BIS finds that price stability is the most important objective for every single one of these economies[1]. This suggests that developing a better inflation forecast is probably the most central issue in central banking research.

Over the last few decades, most traditional inflation forecasting methods assume that there exists an underlying stochastic data generating process which can be determined by a pre-specified model. However, such pre-specified models suffer from two key short comings. Firstly, a pre-specified model "can only be as good as its specification, regardless of what the data might suggest" (Jung, Patnam and Ter-Martirosyan, 2018). Secondly, if the underlying data generating process changes, the prevailing pre-specified model is invalidated. For instance, Stock and Watson (2007) estimate an integrated moving average (time varying trend cycle) model for inflation in the USA and find that the coefficients for this model changed in the beginning of the 1970s and then again in the mid-1980s, leading them to conclude that "…if the inflation process has changed in the past, it could change again".

Additionally, the extant forecasting approaches "bring a variety of undesirable properties, ranging from high sensitivity to model specification to high data requirements" (Smalter Hall and Cook, 2017). This is particularly relevant for forecasting inflation (and other macroeconomic variables) because it is not a high frequency variable, being available only at the monthly, quarterly or annual level. This problem is compounded for emerging and developing economies where "data availability is even poorer and sometimes close to not existent" (Jung, Patnam and Ter-Martirosyan, 2018).

An alternative approach (Breiman, 2001) advocates for models which do not make any assumptions regarding a) the underlying data generating process and thus are invariant to changes in the same and b) do not make any assumptions regarding the underlying relationship between the independent variables and thus are not sensitive to model mis-specifications. Models belonging to the second school of thought focus on finding a function that best represents the relationship between the dependent and independent variables. Machine learning methods fall in this category of statistical modelling. Most machine learning models estimate non-linear relationships, which helps overcome a key disadvantage of linear models, primarily that linear models "fail to identify many macroeconomic phenomenon namely asymmetric business cycles, volatility of stock exchange, inherent regime switching and many others" (Tong, 1990).

Thus, machine learning models provide an opportunity to improve accuracy in a limited data environment and as such are extremely relevant for developing and emerging markets. As a first step towards evaluating the usefulness of machine learning methods for developing markets, this paper forecasts the headline CPI inflation (y-o-y) for 3 emerging market economies: India, China and South Africa. For India, we forecast the 1 month ahead to 12 month while for China and South Africa we forecast the 12 month ahead forecast. We use three

[1] Issues in Central Banking, Chapter 2, BIS Publications

different classes of supervised machine learning methods, namely: penalized linear regression methods (Elastic Net regression), tree based methods (random forests and XG-Boost) and neural networks (CNN, CNN-LSTM and Encoder Decoder).

For each forecast horizon for each country, we use a host of machine learning models and compare the accuracy of each method to 2 benchmark models (namely, a moving average forecast and SARIMA). We find that the machine learning models – in particular the deep neural networks – out-perform the best benchmark forecast for all horizons except the 1 month ahead forecast. For the 3-month forecast of India's inflation, neural network methods provide between 39% to 55% reduction in forecasting error when compared to the benchmark model, where forecast error is measured as the mean absolute deviation of the forecast from the realized inflation. For a 1 year forecast if Indian inflation, neural network models provide a 27% to 44% reduction in forecast error.

The superior performance of the most non-linear methods suggests that there exists a non-linear relationship between CPI (y-o-y) and its determinants in the three emerging market economies (baring the 1 month ahead forecast). Notably, deep neural networks are able to forecast both the peaks and troughs in CPI inflation despite having been trained on small samples. Thus, there are gains to be made from adopting machine learning methods to inform policy decisions in India specifically and all emerging market economies generally.

We find that a combination of the three neural net methods provide an improvement of each method individually for all 3 countries. At the one year horizon, the average neural net forecast results in a reduction in forecasting error (measured by the Mean Absolute Deviation) of 51% for India, 30% for China and 67% for South Africa.

While neural network methods provide limited scope for determining causal relationships, they do provide some avenues for determining which independent variables contribute most significantly to the forecast accuracy for each model. For each machine learning model, we determine which independent variables contribute most significantly to the accuracy of the forecast (variable importance). For CPI inflation in India, we find CPI and its sub-components, food, oil and bank related variables improve the forecast accuracy most significantly, which reinforces the findings of the literature examining the determinants of CPI inflation in India.

The rest of the paper is organized as follows: Section 2 presents the literature review. Section 3 describes the benchmark models and the machine learning methods. Section 4 explains the methods used for interpreting each machine learning method. Section 6 describes the sample and variable construction. Section 7 presents the results of the forecasting exercise while Section 8 presents the interpretation of the forecasts. Section 9 concludes.

2. Literature Review

The literature on the use of machine learning methods for the forecasting of macroeconomic variables is limited but is expanding rapidly. Jung, Patnam and Ter-Martirosyan (2018) use elastic net, SuperLearner and Recurrent Neural Networks to forecast

the macroeconomic data of 7 advanced and emerging economies and find that the machine learning methods outperform the benchmark WEO forecasts. Smalter Hall and Cook (2017) forecast civilian unemployment in the US using 4 different deep neural networks, each of which outperforms the benchmark directed autoregressive model over short horizons. Biau and D'Elia (2010) find that Random Forests combined with a linear model out-performs an AR model for forecasting the GDP growth in the euro area. Tkacz and Hu (1999) forecast GDP growth using artificial Neural Networks (ANN) which are 15% to 19% accurate than the linear benchmark models considered.

In contrast, Chuku, Oduor, and Simpasa (2017) use artificial neural networks to forecast macroeconomic variables in African countries and find that they only marginally outperform ARIMA and traditional structural econometric models**.**

The literature on the use of machine learning methods for forecasting inflation is even sparser. Medeiros, Vasconcelos, Veiga, and Zilberman (2018) forecast in inflation over multiple horizons in the 1990s and 2000s for the US and find that machine learning models (random forests in particular) dominate the benchmark models. Chakraborty and Joseph (2017) present three case studies illustrating the potential utility of machine learning at central banks, of which one is forecasting CPI inflation (UK) on a medium term horizon (two years). They find that the machine learning model beat the benchmark AR and VAR models by at least 29%. McNelis and McAdam (2005) estimate linear and neural network-based models "for forecasting inflation based on Phillips–curve formulations in the USA, Japan and the euro area". They find that the neural network based models outperform the linear models for forecasting the euro area service price indices but have variable performance across consumer and producer indices. Nakarumra (2005) finds that on average neural networks dominate univariate AR models on for one and two quarter ahead inflation forecasts for the US.

In India, the literature on the use of machine learning methods - in particular, for inflation forecasting using multivariate data - does not exist. A recent paper by Pratap and Sengupta (2019) estimate CPI inflation using a suit of machine learning models using univariate data but find that none of the models can out-perform an SARIMA model. A paper by Sanyal and Roy (2014) compares linear, non-linear and consensus forecasting for IIP and GDP in India. Sanyal and Roy (2014) find that combination forecasts dominate linear and non-linear methods for forecasting both IIP and GDP in the short horizon (1-6months). For long term forecasts (7-12months), non-linear methods are best for IIP while consensus forecasts are best for GDP. However, the paper observed improvement in forecast accuracy by using "combination forecast for series with long memory property/ less volatile series."

Our paper is the very first to examine neural network prediction to examine inflation prediction in three of the BRICS countries – India, China and South Africa. Our use of multiple countries mitigates concerns that this method is not generalizable across several countries.

3. Model Description

This section describes each machine learning method used CPI headline inflation for three emerging market economies: India, South Africa and China along with the benchmark

models (those that we compare the given machine learning method with in terms of prediction error). We choose a multivariate approach for forecasting because we assume that "the inclusion of additional information as model inputs will improve model considerably" (Cook and Hall, 2017). In the remainder of the paper the notation we is throughout the paper is as follows: y_i is i^{th} observation of the dependent variable y (for $i = 1, 2, .., N$), $\hat{y}_{i,a}$ is the predicted value of y_i, $X_{i,p}$ is the ith observation of independent variable

3.1 Benchmark Models

We consider two benchmark models: moving average and seasonal ARIMA. For each forecast horizon *h*, the moving average forecast (given my MA(*h*)) is computed as follows

$$\hat{y}_{t+h} = \frac{1}{H}\sum_{h=1}^{H} \hat{y}_{t-h} \quad (1)$$

i.e. the predicted value of y_i at time $t+h$ is the average value y_i over the preceding relevant forecast horizon.

The seasonal ARIMA (SARIMA) is an extension of ARIMA that is capable of modelling the seasonal components in a univariate time series in addition to the autoregressive, moving average and trend components typically modelled by ARIMA. SARIMA is given by the following notation

$$ARIMA(p,d,q)(P,D,Q)m \quad (2)$$

where (p,d,q) are the traditional ARIMA parameters namely, p is the trend autoregressive order, d is the trend difference order and q is the trend moving average order. $(P,D,Q)\ m$ are the additional seasonal parameters : P is the number of seasonal autoregressive terms, D is the number of seasonal difference terms, Q is the number of seasonal moving average terms and m is the number of time steps for a seasonal period. *(p,d,q)* are determined using Autocorrelation Function (ACF) , Partial Autocorrelation Functions (PACF) and tests for stationary. m is traditionally set to 12 for monthly data and suggests a yearly seasonal cycle.

The accuracy of each model is gauged using the Mean Squared Error (MSE). We choose the MSE as the metric to be minimized by each model because as compared to the mean absolute error (MAE), the MSE penalizes large deviations more. However, after having fit each model to achieve the lowest MSE possible using grid search to find the best hyper parameters, for ease in interpretation, to compare the forecasting accuracy across models in the testing sample, we report the Root Mean Squared Error (RMSE). We also report the mean absolute error (MAE) for each model. We report two accuracy improvement metrics – one for RMSE and one for MAE. The percentage accuracy increase is as compared to the best performing benchmark model i.e. for performance metric i ,

$$Accuracy\ Improvement_i = \left(\frac{best\ benchmark\ by\ metric_i - machine\ learning\ model\ metric_i}{best\ benchmark\ by\ metric_i}\right).100$$

We also report the over-estimate percentage which is the number of times the forecast exceeds the actual as a proportion of the total testing observations.

3.2 A Brief Introduction to Machine Learning Methods

Next, we describe the machine learning methods that we use, focusing on the tuning parameters used in our estimation. While in traditional econometric model, we choose the independent variables X_p and the estimation strategy to obtain the parameter ($\hat{\beta}$) for a pre-specified function, in machine learning models we find the optimal hyperparameters to arrive at the optimal parameters for a function forecasting $\hat{y}_{i,a}$. We provide some level of details here drawn from Tibshirani and Friedman (2017), Hastie, James , Tibshirani and Witten (n.d.) and Goodfellow , Bengio and Courville (2016), as the typical readers in economics or finance may be less familiar with some of these methods.

To understand the theoretical underpinnings of the machine learning methods used, we refer to Hastie, Tibshirani and Friedman (2017), Hastie, James , Tibshirani and Witten (n.d.), Varian (2012), Goodfellow , Bengio and Courville (2016) , Mullainathan and Spiess (2017) and Athey and Imbens (2019). In the machine learning literature, input arguments which define the structure (architecture) of the model are known as hyperparameters. The parameters learnt from a model so structured are known as model parameters. Model parameters define the function used for forecasting y_i. Illustratively, in traditional econometrics, we can think of the *chosen* independent variables as the hyperparameters and the $\hat{\beta}$ as the parameter. For each class of machine learning models, there is a different set of hyper parameters. Different values of a hyperparameter result in different model architectures which ultimately results in different predictions of varying accuracy. The process of searching for the hyperprameters that result in 'ideal' model architecture i.e. the model architecture that results in the highest predictive accuracy is known as hyperparameter tuning.

Hyperparameter tuning can be done a) based on previous literature i.e. hyperparameter values and/or rules of thumb discovered in past applications of the relevant machine learning model to the subject at hand ; b) manually : changing the hyperparameters until a satisfactorily high accuracy is reached; c) automatic search (Grid Search and Random Search): Grid Search is the process of specifing a set of values for each hyperparameter. The total number of model architectures is the Cartesian product of each set of each hyperparameter. Random Search is Grid Search combined with subsampling. In random search, instead of specifying a set of values, we specify a distribution for each hyper-parameter. The joint distribution of the hyperparameters gives all the possible model architectures under the given distributional assumptions. R random samples i.e model architectures are chosen from the joint distribution. For both Grid Search and Random Search, the ideal model architecture is the architecture resulting in the highest predictive accuracy.

Thus, the objective is to find hyperparameters which work no matter what the underlying data. Finding such a generalizable set of hyper parameters requires careful specification of the training sample and out-of-sample testing period. To measure the predictive accuracy of a model, it is important that the forecast accuracy be measured out-of sample as the training accuracy can be made arbitrarily high through overfitting. However, if we use the entire out-of-sample data for testing, we may overfit to the out-of-sample data (a phenomenon known as 'data leakage'), resulting poor true generalizability. To protect against 'data leakage', we split the out-of-sample data into two parts: validation data and testing data. The validation

set allows the evaluation of the model on unseen data to select the best model architecture, while still holding out a subset of data for final evaluation after finding the best model.

The training, validation and testing data can be organized in many ways, namely, a) cross validation (bootstrap sampling for cross sectional methods), b) fixed window (training, validation and testing periods demarcated by dates), c) rolling window (shifting a window of fixed size ahead by one observation successively) and expanding window[2] (increasing the window size by 1 successively).

Each machine learning model needs a stopping/penalization/regularization criterion to reduce complexity and over fitting and they differ across machine learning classes.

In the following sections, we move from simple linear models to increasingly complex models. We start with penalized linear regression methods (Elastic Net regression), tree based methods (random forests and XG-Boost) and deep neural networks (CNN, CNN-LSTM and Encoder Decoder).

3.3 Penalized Regression/ Shrinkage Methods

The β_{ols} estimates are BLUE (i.e. Best Linear Unbiased Estimates) i.e. they have zero bias but may have high variance. The variance of β_{ols} increases when a) P is almost as large as, equal to or greater than N and b) there is multicollinearity. In fact, when N<P (also known as "fat data"), OLS estimates don't have a unique solution. In the presence of high variance, β_{ols} become unstable[3] and have a tendency to over-fit on the training data resulting in poor out of sample predictions.

To reduce the variance of β_{ols} estimates –albeit at the cost of having a positive bias - and thereby improve the accuracy of the out of sample predictions, the machine learning literature proposes many penalized regression/shrinkage methods. Each shrinkage method is a linear function which aims to reduce the variance of the β estimates. Penalized Linear Regression Methods/ Shrinkage Methods are named so because they shrink the β estimates towards 0 by adding a penalty - known as the regularization term - to the regression sum of squares (RSS_{ols}) equation. The regularization term penalizes model complexity to avoid over fitting on the training data.

Depending on the type of type of penalty used, there are different kinds of shrinkage methods. Here we consider one shrinkage method, Elastic Net regression, which with suitable hyper parameters, encompasses a broader class of shrinkage models. We use this method because the regularization term for Elastic Net (originally proposed by Zou and Hastie (2005)) is a weighted , convex combination of two different types of penalties– the ridge penalty and least absolute shrinkage selector operator (LASSO) penalty - as follows:

[2] Popularly known as 'online learning' in machine learning literature

[3] 'A small change in the training data can cause a large change in the least squares coefficient estimates.'(Hastie, Friedman, and Tibshirani, 2017).

$$\widehat{\beta_{elastic\ net}} = \underset{\beta}{\operatorname{argmin}} \left\{ \underbrace{\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{p=1}^{p} \beta_p X_{p,i} \right)^2}_{RSS_{ols}} + \underbrace{\lambda \sum_{p=1}^{P} (\alpha \overbrace{|\beta_p|}^{\text{LASSO penalty}} + (1-\alpha) \overbrace{\beta_p{}^2}^{\text{Ridge penalty}})}_{\text{Elastic net penalty}} \right\} \tag{3}$$

Minimization of the above objective function implies minimizing: a) the RSS_{ols} (which results in coefficients which fit the data well) and b) the shrinkage penalty (which amounts to shrinking β_p towards 0). Within the shrinkage penalty, the use of the LASSO penalty allows elastic net to perform variable selection by setting irrelevant β_p to 0 and while the ridge penalty shrinks the coefficients of (highly) correlated independent variables in a similar fashion, resulting in stable coefficients . Thus, by combining the LASSO and ridge regression penalty terms, elastic net gives stable coefficients even in the presence of 'fat data' and high multicolinearity while also performing variable selection. Furthermore, Smeekes and Wijler (2018) also find that "penalized regression methods are more robust to misspecification than" a " dynamic factor approach".

There are two hyperparamters in the elastic net method as in equation (3) above. The parameter λ is the shrinkage penalty: the smaller the value of λ (i.e. the closer λ is to 0) the closer $\widehat{\beta_{elastic\ net}}$ is to $\widehat{\beta_{ols}}$, while the greater the value of λ, the more β_p is shrunk towards 0 reducing their variance. The domain of λ ranges from 0 to ∞.

The parameter α controls which penalty has more weight, whose domain ranges from 0 to 1. When $\alpha = 1$, (3) reduces to LASSO regression while $\alpha = 0$ reduces (3) to ridge regression. If grid search finds 1 to be the optimal value of α, it suggests that the dependent variable is given by a sparse function (and vise versa for $\alpha = 0$).

3.4 Tree based Models

While penalized regression methods are capable of discerning linear relationships in the data, they cannot find a) interactions among the independent variables and b) non-linear relationships, unless the same are explicitly modelled. Modelling all pairwise interactions and/or non-linearites explicitly to solve this issue is "infeasible as it produces more regressors than data points" (Mullainathan and Spiess, 2017). Non-linear methods in machine learning remedy this short coming of linear methods. First, we consider tree based methods (random forests and XG-Boosted trees) followed by deep neural networks (Convolutional Neural Networks (CNNs), Long Short Term Memory (LSTM) networks and a combination of CNN and LSTM neural networks (CNN-LSTM)).

The building block of all tree based machine learning methods is a decision tree. Decision trees can be of two types: classification trees and regression trees. Given that we want to forecast a continuous variable, we focus on regression decision trees. A regression tree is a non-parametric method which splits the entire $X_{p,i}$ space into R rectangular and non-overlapping sub-samples called leafs (given by L_1, L_2 , …, L_R) such the RSS is minimized across all R leaves as follows

$$RSS_{tree} = \sum_{r=1}^{R} \sum_{i \in L_r} (y_i - \widehat{y_{L_R}})^2 \tag{4}$$

$\widehat{y_{L_R}}$ equals $\bar{y}_{L_r}$ i.e. the predicted value for y_i in each leaf is the average value of y_i in leaf L_R. Thus, the relationship between $X_{p,i}$ and y_i is "approximated by a piecewise constant model where each leaf (terminal node) represents a distinct regime" (Medeiros, Vasconcelos, Veiga, and Zilberman, 2018). Note that much like penalized regression methods a regression decision tree also aims at reducing variance.[4]

How does a regression tree find the leaves which minimize (4)? It sequentially divides $X_{p,i}$ into two successively smaller sub-regions based on threshold values for each split. A threshold value is the observation *i* of independent variable X_p that splits entire region into two regions such that the MSE is minimized across sub-regions. To find the threshold value *t*, the regression tree splits the region under consideration into two regions based on each observation *i* for each independent variable X_p and chooses the (i, X_p) pair which gives the lowest MSE. This process of sequential, binary splitting continues till a stopping criterion is reached to prevent over fitting. The last layer of sub-samples form the leaves L_R where $\widehat{y_{L_R}}$ equals $\bar{y}_{L_r}$.

Among non-linear methods, the primary appeal of regression trees is that they are highly interpretable but may suffer from omitted variable bias in the presence of multicolinearity[5]. Furthermore, standalone decision trees are associated with high variance and the estimated regression tree is often "discontinuous with substantial jumps" (Athey and Imbens, 2019) which reduces the accuracy of the predictions made by a single decision tree substantially and making them uncompetitive. However, when regression trees are used in ensemble methods (like random forests and boosting), the accuracy of the predictions improves drastically.

Ensemble learning methods forecast y_i by aggregating the predictions of many weaker models (called base learners) into a single prediction.[6] Two popular ensemble learners are bagging and boosting which when used in conjunction with decision trees result in random

[4] However, Athey and Imbens (2019) caution against interpreting trees on the grounds of omitted variable bias, especially in the presence of correlated variables as "covariates that have strong associations with the outcome may not show up in splits because the tree splits on covariates highly correlated with those covariates."

[6] Base learners can be from the same learning algorithm (as in the case with Random Forests) or different learning algorithms (eg: SuperNets).

forests and XG-Boost respectively. Each of the two has a different objective: while random forests aim at reducing variance XG-Boost reduces bias.

Random Forests average over a large number of de-correlated trees to improve the accuracy of its predictions vis-à-vis a single regression tree. A random forest generates multiple decision trees simultaneously. The predictions of each tree are uncorrelated from those of the other because: a) each tree is built on a boot-strapped sample and b) at each new split in each tree, a new random subset of the independent variables of m predictors is chosen for determining the threshold value. After all the trees have been built, $\hat{y}_{rf}$ is obtained by averaging the predictions of each tree. [7]

Athey and Imbens (2019) state that random forests are very effective when a relatively small number of the independent variables are related to the dependent variable. However, they also state that random forests a) "are not efficient at capturing linear or quadratic effects", b) are not efficient at "exploiting smoothness of the underlying data generating process" , c) tend to "have bias, particularly near boundaries" and d) "in small data sets will have more of a step function shape."

In a random forest, each tree is independent of the other because all the trees are built simultaneously. Boosting grows the trees sequentially rather than simultaneously to allow each subsequent tree to achieve a smaller forecast error than the preceding tree by learning from the residuals of the preceding tree. This is achieved by fitting each subsequent tree on the residuals of the preceding tree. Boosting updates each tree in a sequentially additive manner: The weighted output of the current tree is added to the preceding tree to update the boosted tree. $\hat{y}_{boost}$ is the weighted average of these additive models.

Currently, the most popular boosting algorithm is Extreme Gradient Boosting (XG-Boost). XG-Boost uses boosting in conjunction with a gradient descent algorithm to minimize the loss function when adding a new model. This means that instead of fitting the each subsequent tree on the residuals of the previous tree, XG-Boost fits each subsequent tree on the gradient of the loss function of the previous tree. This to makes the boosting algorithm more generalizable to any differentiable loss function.

3.4.1 Hyper-parameters for Tree Based Methods

Using grid search, we determine the optimal hyperparameters listed in Table 1, which are used to determine the best parameters for building trees i.e. the best splitting variable and its associated threshold value for each node of each tree.

Each tree based method requires a regularization criterion because a tree can over fit on the training data by growing till each node is a 100% pure node (i.e. a node where all the observation in the training data belongs to one regime). However, such a tree gives poor out of sample predictions. One popular regularization criterion is tree pruning once the whole tree is

[7] m<p. Usually, m = $\sqrt{P}$ i.e. the number of independent variables considered at each split (m) approximately equals the square root of the total number of independent variables.

built. Alternatively, determining the optimal hyper parameters in listed Table 1 also helps avoid over fitting.

Maximum tree depth is the maximum possible number of levels a tree can have. If the tree is too short, it will be unable to find the relevant patterns in the data. However, if it is too deep, it will overfit. The minimum observations for node splitting, minimum observations for leaf formation and maximum number of leaf nodes help prevent the tree from overfitting by preventing the formation of regions that are too niche. Total number of trees defines the number of de-correlated trees grown. In general, deeper trees reduce bias while a larger number of trees reduces variance. The number of independent variables to consider in each node while determining the best splitting variable also reduces variance.

XG-Boost has additional parameters to reduce the variance of the trees: Columns Sample by Tree, Columns Sample by Level and Columns Sample by Nodes. 'Columns Sample by Tree' is the percentage of independent variables to consider while building each tree, level and node respectively in XG-Boost. Additionally, much like elastic net, XG-Boost regularizes the leaf weights using L1 and L2 penalty to encourage scarcity and reduce complexity. The learning rate slows down the weight updating process thereby reducing the possibility of overfitting.

3.5 Deep Neural Network

The machine learning methods considered till now are not capable of discerning any information from the sequential and temporal structure of time series data. If fact, tree based methods treat data as cross sectional. This is especially an issue if the data is not stationary. The deep neural networks considered by us - Convolutional Neural Networks (CNNs), a combination of CNN and LSTM neural networks (CNN-LSTM) and an Encoder Decoder network - remedy this short coming as they can discern information from the temporal and spatial structure of time series data.

In general, neural networks are composite functions which are universal function approximators, i.e., they can approximate any arbitrarily complex function after being specified in the appropriate manner. A neural network is a linear/non-linear transformation of the weighted linear combinations of the data $X_{i,p}$. Every neural network is broadly composed of three types of layers: the input layer, the hidden layer/s and an output layer. A neural network with only 1 hidden layer is known as a single-layer neural network.[8] A neural network with more than 1 hidden layer is known as a deep neural network. Goodfellow, Bengio, and Courville (2016) state that in out-of-sample testing, on average, deep neural networks generalize better than single-layer neural networks and thus we use the former for forecasting inflation.

Each of the 3 layers is comprised of multiple nodes and is connected to the subsequent layer through weights. The structure of a neural network "follows the structure of a GLM model" but instead of using maximum likelihood estimation), it uses the feed forward mechanism and back propagation (a non-parametric algorithm) to determine the weights that

[8] Lippmann (1987) finds that a multi-layer perceptron (MLP) with the appropriate number of hidden nodes is sufficient for estimating convex regions/regions of any shape thus over-coming the limitation of linearly separable regions.

result in the function for forecasting y_i (Smalter Hall and Cook, 2017).The weights so determined, "identify which features and parameters (i.e. computational nodes) are relevant for prediction" (Smalter Hall and Cook, 2017). These processes are best illustrated by considering a fully connected feed forward neural network.

A fully connected feed forward neural network is called fully connected because each node is globally connected i.e. each node in each layer is connected to each node in each subsequent layer. As a consequence, all the data is fed simultaneously to the hidden layers from the preceding layer.

Fully connected neural networks rely on two processes for training: the feed forward mechanism, followed by back propagation. The feed forward mechanism is the process through which data goes from the input layer to the hidden layers and then to the output layer to produce the predicted value $\hat{y}_{nn}$. The input layer is connected to computational nodes in the subsequent hidden layer where the weighted linear transformation of the data is computed. The computational node is connected to further computational nodes where the same operations take place. This continues till the last layer i.e. the output layer where the last transformation of the weighted linear combination is computed. This completes one pass of the feed forward mechanism.

After completing one forward pass of the feed forward mechanism, the loss function $L(y_i, \hat{y}_{i,nn})$ (which is a penalized version of MSE) is computed to determine the accuracy of the $\hat{y}_{i,nn}$ .The feed forward mechanism is repeated for E iterations/epochs (for e going from 1 to… E), such that in each subsequent epoch $(e + 1)$, the accuracy of $\hat{y}_{i,nn}$ is improved by minimizing the loss function through back propagation.

Back Propagation is the process through which a neural network "learns" i.e. it's the algorithm through which a neural network determines the weights and biases required for minimizing $L(y_i, \hat{y}_{i,nn})$. These optimal weights and biases for minimizing $L(y_i, \hat{y}_{i,nn})$ are determined from the gradient of the loss function. To determine the gradient of the loss function a popular method is stochastic gradient descent (SGD)[9]. The method is stochastic because it partitions the entire training sample into b random sub-samples randomly. The use of random sub-samples increases the chances of finding the global (vis-a-vis the local minima) of the loss function in each epoch e. The weights and biases are updated for E epochs or till a stopping criterion is reached whichever comes first.

As stated previously, the input layer being a column vector means that the data is revealed to each hidden node simultaneously. As a result the data is treated as being cross sectional because the spatial and sequential nature of the data is not exploited. Thus, we don't estimate inflation using a fully connected feed forward neural network because it is dominated by more complex deep neural networks that can infer information from time series data.

[9] An alternative to SGD is an extension of SGD called Adaptive Moment (ADAM) optimization.

3.5.1 Convolutional Neural Networks (CNN)

The temporal structure of time series data means that values closer in time to each other have more in common than values separated by larger periods of time. Illustratively, time series data exhibits autocorrelation. Convolutional neural networks (CNNs) can extract information from the temporal structure of the data by a) preserving the spatial/ temporal structure of the data and b) using filters which look for patterns in spatially adjacent data. For instance, one filter could finding peaks, another could find troughs while another could find a linear trend. They achieve this by using convolutional layer which are not fully connected layers.

A CNN consists of the following layers in the following order: one or more convolutional layer/s, subsampling layer/s, optionally followed by fully connected feed forward neural network/s and finally the output layer (see Figure 2).[10] The first layer in a CNN is always a convolutional layer which is comprised of the input layer, filters and feature maps. The input layer preserves the temporal structure of the data by accepting data in a 3 dimensional format - $width^1$x $height^1$x $depth^1$. $Width^1$ is the number of the independent variables (X_p), $height^1$ is the number of observations we assume to be related across time and $depth^1$ equals 1 in a 1D – CNN. Instructively, refer to the input layer in Figure 2. Each column of nodes in the input layer represents a single independent variable. For P independent variables, there will be P columns of in the input layer. The height of each column (T) is the number of time units for which we think the data is related. If we think that every 6 monthly set of data is related, T equals 6.

The input nodes are not fully connected to the computational nodes. This is local connectivity is achieved through the process of convolution[11]. For a neural network, convolution is the matrix dot product of the input layer and the filter/s computed in the locally connected computational nodes[12]. The input layer is associated with one or more filters $\boldsymbol{w}_f$ (for $f = 1, 2, \dots, F$) . $\boldsymbol{w}_f$ is a matrix of weights with dimensions n x n x $depth^1$ which looks for patterns in spatially adjacent subsets of data (x_c for c = 1,2, ... C) [13]. This is achieved by the filter $\boldsymbol{w}_f$ convolving over the input layer. That is the filter connects each subset x_c to one computational node in the hidden layer where the dot product is computed and then transformed. This process is repeated sequentially for every subset x_c. This process achieves local connectivity. A hidden layer constructed with such locally connected nodes is called a feature map, an activation map or a convolved feature. Each additional filter ($w_{f+1}, w_{f+2\dots.}$) gives a new convolved feature.

As illustrated in Figure 2, the convolution layer is often followed by a subsampling layer P_f of size mxl to further condense and amplify the feature maps[14]. The sub-sampling layers may be followed by a fully connected layer which is followed by an output layer

[10] In cross sectional methods, it does not matter how the data is sorted but for CNN, LSTMs and Encoder Decoders it does.

[11] In mathematical terms, a convolution is an integral which measures the degree of overlap between two functions as one function passes over the other.

[12] This is equivalent to the weighted linear combination in the feed forward fully connected network

[13] n< $width^1$ and n< $height^1$

[14] The total number of subsampling layers equals the total number convolved features which in turn equals the total number of filters.

following which back propagation takes place as described in (albeit with some modifications) till a stopping criterion is met to produce $\hat{y}_{i,cnn}$.

3.5.2 CNN -LSTM Neural Network

In the feed forward networks considered hitherto (i.e. fully connected neural networks and CNNs) the information has a unidirectional flow: it goes from the input layer to the hidden layers to the output layer. A short coming of unidirectional networks is that they cannot store the history of a variable/s because they lack memory. Consequently, they are unable to extract any information from the sequential nature of time series data. Recurrent neural networks (RNNs) remedy this shortcoming.

Like CNNs, the RNNs draw information from the temporal structure of the input data. However, unlike CNNs, they also draw information from the sequential nature of the data $X_{p,i}$ because they have memory. This memory is used to inform the predictions made by the RNN. There are many RNN specifications to choose from and we chose the Long Short Term Memory (LSTM) RNN over a vanilla RNN as the former can learn from long sequences while the former may not.[15] Thus, we consider a hybrid model which is a combination of a CNN and LSTM neural network. The combined method offers improvements over each individual method as it extracts information from both the spatial and sequential nature of time series data.

The structure of a CNN+LSTM neural networks is as follows: the first layer is a convolutional layer/s, followed by an LSTM layer/s, followed by fully connected layer/s which (as always) gives $\hat{y}_{i,cnn+lstm}$ as a non-linear transformation of a weighted sum, followed by back propagation to optimize the model weights. The structure of the CNN network is as described in the previous section. Here we focus on the architecture of the LSTM network.

Consider Figure 3. As the name suggests, LSTM networks have both long term and short term memory. Intuitively, one can think of LSTM nodes as more complex computational nodes in the hidden layer in a neural network. In part, the complexity arises from the differing manner in which data is revealed to the computational nodes. In a fully connected neural network, all the data is seen by the computational node at once because the data is stored without structure. However, data is reveled to an LSTM node in a sequential manner allowing it to learn from a sequence of values in an iterated and incremental fashion. This iterated and incremental learning is due to the looped/rolled structure of an LSTM node.

Each input node in an LSTM network as a vector $\boldsymbol{x_p}$ which is composed T observations from independent variable X_p. T is the length of the sequence (i.e. past data) we want the LSTM network to remember and learn from (this T is similar to the T used in CNNs). Each observation t in $\boldsymbol{x_p}$ is revealed to each LSTM node sequentially which allows the LSTM node to compute both the long term memory and the working memory. Based on the updated long term memory, the LSTM cell updates the working memory. The updated working memory at the last element

[15] For more on this topic see the vanishing gradient problem for RNNs.

of the sequence is the output of an LSTM node. The LSTM layer is may be followed by a fully connected layer, which is then followed by the output layer.

3.5.3 Encoder Decoder

We consider an encoder decoder model belonging to a broader class of models called sequence to sequence (Seq2seq) models which translate sequences from one domain (such as a sentence in French) to a sequences in another domain (such the same sentence translated to Hindi).

An encoder decoder is composed of two sub-models: one is the encoder that reads input sequences and converts it to an internal representation which is the neural networks understanding of the data. The decoder is an output mode which takes the encoded representation (i.e. the understanding of the network) and its own predictions to previous portions of the encoded sequences to predict the output. That is "the decoder allows for the model to make predictions that fit with the context established in its earlier predictions" (Smalter Hall and Cook, 2017).

In time series forecasting, using an Encoder Decoder model amounts to translating the past into the future. The available history of the concerned time series is encoded, allowing the encoding of patterns like seasonality and trend, conditional on which predictions are made. In the Encoder Decoder model used by us, the encoder is a CNN which has proven very effective in learning the features of our data while the decoder is an LSTM model. Both the CNN and LSTM work as described in the preceding two sections.

3.5.4 Hyper-parameters for DNNs

Using grid search we determine the hyper-parameters in Table 2 to determine the neural network parameters i.e. the weights. Every neural network requires a stopping criterion for the training process to prevent the neural network from over fitting on the training data as a neural network can achieve an arbitrarily low MSE in-sample. A popular stopping criterion is the total number of epochs i.e. the number of times a network under goes back propagation. Too few epochs could result in the optimal function not being reached while too many epochs might result in over training. Given that we don't know the critical values at which under-estimation transitions into over-fitting, we use an alternate popular stopping criterion is to stop training the neural network once the forecasting error for the validation sample stops decreasing for a certain number of epochs called 'patience'.

Batch size is the number of observations in each random sub-sample for SGD to update the weights. The smaller the batch size, the larger the number of random samples, the greater the generalizability of the estimated function. The learning rate controls the amount by which the weights are updated during back propagation. Usually, smaller batch sizes are paired with larger number of epochs.A very high learning rate may result in the back propagation algorithm not finding the minima of the function. A very low learning rate guarantees finding the minima but slows down the algorithm significantly. A good rule of thumb is to start from smaller learning rates and batch size and the progress to higher values.

Currently, “there is no universally accepted analytical way to determine the optimal number of neurons and layers for a given classification or regression application, adding large "degrees of freedom" to the estimation of neural networks” (Jung, Patnam and Ter-Martirosyan ,2018). Some rules of thumb suggested in the academic literature are "somewhere between the input layer size and the output layer size" (Blum, 1992) and "as many hidden nodes as dimensions needed to capture 70-90% of the variance [in] the input data" (Boger and Guterman, 1997). However, in practice, the optimal number of layers, nodes and filters is determined though out-of-sample testing (Tkacz and Hu, 1999).

Note that having multiple filters in the convolutional layer results in a hierarchical structure – the first filter helps discern the simplest features from the data (like a linear trend) while each subsequent filter discerns increasingly complex features from the data (such as pro/anti-cyclical activity). We also use Batch normalization between the convolutional layer and the LSTM layer. Batch normalization transforms the activations (i.e. output) of the previous layer such that the mean activation is close to 0 and the activation standard deviation reaches 1. Batch normalization accelerates the training process of the neural network and may improve model performance by penalizing complexity marginally i.e. by having a regularization effect.

Given that there is no previous literature for the estimation of multivariate machine learning models for India, we choose Grid Search to find and select our optimal hyperparameters. We choose grid search instead of random search because we do not have any priors regarding which hyperparameter is more important and thus presume all hyperparameters are equally important. In the presence of a prior regarding relative importance of hyperparameters, random search would be a better choice as it would allow greater exploration of the possible values of that parameter[16].

We use a rolling window approach to train and test our model (for the neural network as well as all other models) as this will allow for structural change in the parameters. Figure 4 presents a graphical depiction of this approach. Given that machine learning models benefit from longer series of data, we set the window size to total number of observations minus the total number of observations to be forecasted. First we specify the model, the Grid Search parameters and then check its accuracy across all the windows. For each window, the machine learning model is fit as follows (see figure 4):

Each window is divided into two parts: the training period and the out-of-sample testing period. 10% of the total observations in the window form the out-of-sample data while the remaining comprise the training data. The out-of-sample set is further divided into two parts: the validation data and the testing data. Of the out-of-sample set, the last observation is the test data and the remaining observations form the validation data. Each machine learning model is fit on the training data and is validated for accuracy in the validation sample. Thus, for W windows, average validation MSE is

[16] A “Gaussian process analysis of the function from hyperparameters to validation set performance reveals that for most data sets only a few of the hyper-parameters really matter, but that different hyperparameters are important on different data sets”. (Bergstra and Bengio, 2012)

$$Average\ Validation\ MSE = \frac{1}{W}\sum_{i=1}^{W} Validation\ MSE_i$$

The model with the lowest average validation MSE is chosen as the best model. The best model is then used for forecasting the testing observation in each window. Given that the last observation is of every window is the testing window, the number of windows equals the number of testing observations.

Additional Concerns while Training a Neural Network

The "training process for a neural network is subject to stochasticity" (Smalter Hall and Cook, 2017). First, the initial weights for each neural network are small random weights not equaling to zero.[17] Second, the use of random sub samples in the optimization process. Third, we use 'dropout' which is a regularization techinque whereby the neural network ignores the output of a randomly selected subset of nodes to limit the "over-dependence of the model on any one node" and thus reduce the potential for over-fitting (Smalter Hall and Cook, 2017) . As a consequence of this stochasticity, training the same model repeatedly results in different weights and thus different forecasts in each run. To accommodate this variance, we train 30 instances of each model, allowing the computation of expected model accuracy across multiple runs of the same model. Thus, to select the best neural network architecture, we choose the architecture which gives the lowest $Average\ Validation\ MSE$ across 30 runs of each rolling window. Furthermore, given that we use a rolling window approach to estimation, the weights of the model are updated by each new window, allowing the weights to be updated iteratively.

4. Empirical Analysis

In this section, we present the results for forecasting error that result under the various methods considered in the previous section. Our principal variable for forecast is the headline CPI inflation (y-o-y) at varying horizons for 3 emerging market economies: India, China and South Africa. We forecast y-o-y inflation because it has lower seasonality and volatility than month on month (m-o-m) inflation. We choose the lowest available frequency for both the dependent variables which is the monthly level. The data for India is obtained from CEIC while the data for China and South Africa is obtained from the FRED database.[18] Table 3 presented the data period used for each country as well. The unit of observation used for analysis is data at the monthly level.

[17] Importantly, "if the weights are near zero, then the operative part of the sigmoid is roughly linear, and hence the neural network collapses into an approximately linear model. Hence the model starts out nearly linear, and becomes nonlinear as the weights increase" (Hastie, Tibshirani and Friedman, 2017).

[18] To calculated the CPI inflation (y-o-y) we use the CPI (Combined - 2012 base year) series as calculated by the Central Statistical Organization (CSO), India and rebase it using the IMF CPI series.

4.1 Data Preparation

For each dependent variable, we choose independent variables based on the following two criterions:

1) is the variable available for the entirety of the training and testing period?

2) if the answer to (1) is yes, is the variable available at the monthly level?

If a variable meets these two criterions, we use it for forecasting the dependent variable under consideration. Beyond these two criterions, we do not use any other filter to choose the independent variables.

Broadly, the suit of independent variables considered fall into the following categories : WPI and its Subcomponents, CPI and its Subcomponents, food related indicators, oil related indicators, automobile industry indicators, electricity generation related indicators, monetary policy and finance related variables and trade related variables. Additionally, for China and South Africa, the OECD Composite Leading Indicators (CLI) are also included as independent variables ((See Appendix A: Table 1 - 3 for more details).

Given that our dependent variables measure the y-o-y change, we transform each independent variable into its y-o-y change equivalent. Next given that we are using a rolling window approach, in each window we normalize the data in the training sample because machine learning methods are not scale invariant, especially deep neural networks. In fact, for the deep neural networks, after normalization, the data is rescaled to a suitable range (-1 to 1 in our case) as DNNs are not invariant to the magnitude of the data. The normalizing constants (mean and standard deviation) and the rescaling parameters from the training sample are used to normalize and rescale the data in the validation and testing samples to ensure that the out-of-sample data does not have a look forward bias.

For each forecast horizon h, we start from the h^{th} lag and include a maximum of 12 lags of the dependent variable and each independent variable for each forecast horizon for each country. Thus, the 1 month ahead forecast uses the 1^{st} to 12^{th} lags while the 12 month ahead forecast only uses the 12^{th} lag. This allows us to investigate the effectiveness of machine learning models both in scenarios where N<P and N>P.

4.2 Results for India

We start with a comparison of the RMSE and the MAE for all the ML models at various horizons (1 month to 12 months) in Table 4. At the one month horizon, we find that the naïve/MA model performs the best when either RMSE or MAE are considered. In particular, the MAE for the naïve forecast is 0.46% whereas the best neural network method has a forecasting error of 0.57%. Thus, given that our average inflation rate in the testing period is 3.58%, this corresponds to an error of 12.85% (15.92% relative to this average value).

However, the key power of the neural network approach comes about for longer forecasting horizons. In particular, for the 3 month ahead forecast, the naïve forecast error increases to 1.13%, which is a large error of almost 31.56%. Most of the other linear models

(shrinkage and tree bases models) have similar or worse performance relative to the naïve forecast. In contrast, the neural net models have a forecast error between 0.5% to 0.68% which is a huge increase in forecasting ability relative to the benchmark model as well as all other methods. In terms of forecasting error, the reduction in forecasting error of the best neural net model (CNN+LSTM) relative to the benchmark model is 48.15% if one considered the RMSE, and 55.54% if one considered the MAE. Another interesting feature that comes out here is that all the other non-neural network methods used (shrinkage and tree based models) have worse performance relative to the naïve model for this horizon of forecast.

Next, we examine the 6 month forecast performance. Here again, the best neural network model (CNN) has a lower forecast error using the MAE relative to the naïve forecast by 56.3%. It is also important to note that other neural net methods – encoder decoder and CNN+LSTM also perform much better, reducing the forecast error by 47%-48%. In contrast, the best possible non-neural net method – the elastic net – has a forecast error improvement of 27.27% relative to the benchmark model.

Note also that the absolute error of the forecast (regardless of the model) goes up for the longer forecasting horizon. Here again, the increase in forecast error of the 3 month relative to the 1 month is around 145% (1.13% relative to 0.46% using the MAE). In contrast, for the best neural network model, the increase in forecast error measured by the MAE is much more modest 19.26% (0.571 to 0.682) for the encoder-decoder. This pattern continues even for the 12 month forecast where the naïve model has a forecast error of 1.16%, while the encoder decoder has a forecast of 0.84%. In contrast to these two methods, almost all the other methods, whether they be SARIMA or shrinkage models, show huge increases in forecast error which essentially make them irrelevant tools for prediction at the one year horizon, as they have much larger prediction errors relative to the naïve model.

It is clear from the above table that shrinkage or tree based models are not particularly useful for inflation prediction in India. This is not surprising in the light of the earlier discussion that these methods are not well suited to the time series setting, especially in the presence of structural change and non-stationarity. India has undergone a lot of structural changes in the last two decades, and in particular, has moved to an explicit inflation target in the last 5 years. One counter argument to the above is to use parameters estimated with a limited time series focusing on the recent past. However, we already incorporate this approach using the rolling window estimation for all the methods. Thus, any structural change that can be captured is already done so at least in the sense of ignoring all observations prior to the rolling window period. Thus, this result suggests that shrinkage and tree based models are fundamentally unsuited to inflation forecasting in the Indian context.

The above analysis implies that neural network methods dominate other method learning methods, linear methods and naïve forecasts. However, an alternate ‘popular’ metric used, as evidenced from the above press articles, for evaluating accuracy of a given method is the number of times it provides an overestimate relative to an underestimate. It is feasible that non-linear methods may have lower absolute and square error, but may still be subject to average bias. To evaluate this, we tabulate the number of times each method provides an overestimate relative to the actual inflation. Recall that the training period was in the pre-demonetization period, prior to the large crash in food and vegetable prices post-demonetization, while the testing period does include the large crash in food prices.

This is also motivated by a large degree of interest in the press and policy circles that the Reserve Bank of India was consistently overestimating inflation rates. In an article titled "Has RBI consistently overestimated inflation forecasts," published in LiveMint, 2017, the author Tadit Kundu argues that RBI has been consistently overestimating inflation in 2015-2016. At the same time, this article argues that this is true for other central banks as well and that this is true for professional forecasters as well.[19] On the other hand, a Mint Street Memo authored by Raj et al (2019) argues that the large forecast errors were attributable to large unanticipated food price shocks, and that countries with high share of food prices in their CPI baskets tended to have higher forecast errors. The authors further argue that RBI forecasts did not have any bias if one excluded the demonetization period and compared favorably to other central banks, especially if one considered the fact that inflation in India is much more volatile.

Thus, one might expect that most methods have a fraction of observations with overestimate to be greater than 50%, although unbiasedness in the long run implies that this fraction should be close to 50%. Table 5 presents the results of this analysis. For the 1 month forecast, the naïve forecast has a fraction of overestimates of 56.67%, which is the closest to the 50%. On the other hand, shrinkage and tree based models have a large positive bias with overestimate fractions ranging from 70%-90%, while neural network methods have overestimate fractions ranging from 30% to 64%. If one took an average of the neural networks, this works out around 51.3%, which is quite close to the expected fraction of 50%.

For the 3 month forecast, the naïve forecast overestimates inflation 70% of the time. Shrinkage and tree based models also perform poorly with overestimate fractions ranging from 66.67% to 90%. In contrast, neural networks have overestimate fractions ranging from 46.67% to 76.67%, which implies an average overestimate fraction of 57.45% which is much closer than the other methods. This continues for the other horizons.

An observation that is pertinent to mention here is that the average of the 3 neural network method has an overestimate fraction that is much closer to 50% relative to each individually. This suggests that an average of the forecasts of these 3 methods may perform much better than each of them, a topic to which we will return towards the end of this section.

Interestingly, the Wilcoxson statistics that tests for the difference in distribution of the predicted and actual values implies that the distribution of neural net forecasts does not differ significantly from the distribution of the actual inflation for almost all horizons, with the exception of the exception of the CNN+LSTM forecast for the 3 and 9 month horizon. On the other hand, almost all other method forecast differ significantly from the realized values, suggesting that neural network methods are likely to be significantly better even at forecasting the distribution of the realized values. We leave an evaluation of the distribution of the quantiles of the inflation for a future revision of this paper.

4.3 Time Series Examination

The previous sub-section examined the average performance of various methods in forecasting inflation. To enhance ease of interpretation, we focus only three variables for the time series analysis – the actual inflation, the best benchmark method (the naïve forecast in all

[19] "Inflation targeting: Did India sleepwalk into a disaster," Economic Times, Dec 21, 2018

cases) and the best machine learning (always one of the three methods of estimation for neural network for all forecast horizons above 1 month). In each case, the best method is defined by the one with the lowest RMSE.

Figure 5A shows the time series performance of 1 month forecasts for naïve and best machine learning forecast relative to actual. It is clear that both the naïve and the best machine learning overestimate inflation in the period when inflation was reducing – from Oct 2016 to June 2017. In the period from July 2017 to April 2018, both methods underestimate the actual inflation. Similarly, from June 2018 to the end our sample period, both methods overestimate the true inflation. One pattern that emerges for the 1 month horizon is that both methods lag the true inflation and underestimate inflation in periods of increasing inflation and overestimate inflation in periods of reducing inflation.

Next, in figure 5B, we present a similar analysis for the 3 month forecast. One important fact that emerges is that the tracking error for both methods is much larger than that in the 1 month forecast. At the same time, it is clear that the naïve forecast is much worse relative to the actual inflation. Interesting, the machine learning model over predicts the dip in inflation during the demonetization and overpredicts the bounce-back as well. However, from May 2018, the model tracks actual inflation quite well and in fact, predicts the turning points quite well.

Figures 5C, 5D & 5E provide these comparison for the 6 month, 9 month and 12 month horizons. At the 6 month horizon (Figure 5C), both methods have a significant deterioration in performance. The demonetization episode registers for the machine learning methods while the naïve forecast completely misses it. From June 2017 to July 2018, the best ML method shows a marginal increase, while the naïve forecast has the wrong direction of the trend as well as a very large forecasting error. In the post June 2018 period, the best ML method tracks the direction as well as the magnitude of the actual inflation very well.

In contrast, at the 9 month and 12 month horizon, the performance of the best ML methods improves substantially. With the exception of the large inflation dip in June 2017, ML captures both the direction as well as magnitude of the actual inflation very well. It also performs well in the pre-demonetization period. In both cases, the naïve forecast varies little as anticipated.

4.4 Results for other emerging markets

Next, we examine the extent to which our results are generalizable using two other emerging markets – China and South Africa. We only estimate the neural network models as the results for the other methods were significantly poorer for all the other methods. Also, we focus only on the 12 month forecast, as this had the largest prediction error. A first difference of the results relative to India is that SARIMA performs much somewhat better for South Africa relative to the MA or naïve forecast. Hence, the benchmark model for South Africa we uses is the SARIMA model, while for China, the naïve or MA forecast performs much better relative to SARIMA, as was the case with India.

Table 6 presents the results of this estimation. For China, the improvement in performance using neural networks is much more modest, ranging from 5.8% for the encoder

decoder technique and 33.42% for the CNN-LSTM model for the MAE measure. The increase in accuracy using RMSE are even smaller. On the other hand, for South Africa, using neural networks enhances the forecasting ability significantly, with decreases in MAE ranging from 42% to 57%.

In Table 7, we present the results of the fraction of forecasts that were overestimates. For China, the naïve forecast does very well, overestimating the realized inflation around 46% of the time. In contrast, South Africa appears to be much more similar to India in that both the naïve forecasts are overestimated for a large fraction of the testing sample (between 86% and 90%). For South Africa, using neural network methods improves the fraction of times that overestimation happens.

Figure 6 presents the time series of 12 month forecasts using the best machine learning method and the actual inflation rates along with the benchmark model for China. This figure suggests quite a different picture relative to the comparisons in Table 6. In particular, the benchmark model is quite smooth and does not reflect a lot of the dynamics of the inflation rate. In contrast, the machine learning model has much more variable predictions, which, at least in a visual sense, vary more with the data. There are notable exceptions – the peak in July 2018 and the trough in Feb 2019, both of which are completely missed by the machine learning algorithm. In contrast, for both of these episodes, the benchmark model, by virtue of its relative flatness happens to be much closer to the realized inflation.

Figure 7 presents the time series results for the 12 month ahead forecast for South Africa. Quite clearly, the results are in line with the increase in forecasting ability based on the MAE and RMSE results. In addition, the neural network method captures the overall decrease in inflation from June 2017 to March 2018 and the reversal. The benchmark model's overall trends are completely reversed – it predicts an increase in this period, and then predicts a decrease from Feb 2018 onwards, when actual inflation was increases. Nevertheless, even for the neural network methods, there are sub-periods when the prediction diverges significantly from the actual. In May 2017, actual inflation decreased significantly while the neural net predicted an increase till July 2017. In June 2018, ML predicted a sharp decrease in inflation while actual inflation increased.

4.5 Combined neural net forecast

One of the issues that came up in earlier sections is that the bias in the different neural network forecasts seemed to be in opposite directions in several cases. As we had alluded to earlier, this suggested that a combined forecast using the different neural network methods may be better than each of these individually. To test this, we combine the three forecasts by averaging them and recompute the accuracy of the resulting forecast. Tables 8A and 8B give the results of this approach. We find large increases in forecast accuracy using the combined forecast especially for China where the increase in forecast accuracy using MAE increases to 30%. The direction fraction of overestimate also comes much close to 50%.

4.5 Variable importance

Recall we had mentioned in the introduction that one of the disadvantages of the neural net methods is that it is not feasible to estimate the partial effects of each individual variable. One approach that partially alleviates this 'black-box' nature of this disadvantage is an analysis called 'variable importance', which describes "how much a prediction model's accuracy depends on the information in each covariate" (Fisher, Rudin, and Dominici, 2018). For each machine learning method under consideration, we attempt to understand which independent variables contribute most significantly to the forecast accuracy i.e. which variables contribute most significantly to a reduction in the MSE. For each forecasting horizon, we determine the variable importance for the best performing machine learning method.

Given that we are considering three disparate classes of machine learning models, we use a different measure of variable importance for each class. For each shrinkage (linear) method, we compare the absolute size of the coefficients of the independent variables. The larger the absolute value of the coefficient, the more important the variable is to the accuracy of the forecast (and vice-versa).

For tree based methods, the importance of each independent variable is gauged by examining decline in the RSS achieved by splitting the sample using a given independent variable, averaged over all the bootstrapped trees. The larger the reduction in the RSS, the more important the independent variable is (and vice versa).

The deep neural networks are the most difficult to interpret. We use a simple but effective approach known as model reliance (MR) as proposed by Fisher, Rudin, and Dominici (2018). MR "measures the importance of a feature by calculating the increase in the model's prediction error after permuting the feature" but leaving all the other independent variables and dependent unchanged (Molnar, 2019)[20]. Permuting/shuffling an independent variable breaks the relationship between the independent variable and the dependent variable. The shuffling approach is especially appropriate for deep neural networks as random shuffling invalidates the spatial and temporal information in time series data. Effectively, this creates an unconditional counterfactual for X_p. An independent variable is important if shuffling its values increases the MSE as this indicates that the model relied on the actual realization of the independent variable for forecasting $\hat{y}_i$. The permutation variable importance (PIV_p) is calculated as follows: After having trained the model we arrive at the final MSE for the deep neural network (MSE_{real}). Then, for each variable X_p , the following is repeated :

Step 1: X_p is randomly shuffled leaving all the other independent variables and y_i unchanged

Step 2: Using the dataset with the shuffled X_p , $\hat{y}_i$ is forecasted again to arrive at the new MSE ($MSE_{shuffled,p}$).

[20] In machine learning literature, features refer to independent variables

Step 3: Given that the permutation process is inherently random, we repeat Steps 1 and 2 a 100 times and calculate the mean of $MSE_{shuffled,p}$ across the 100 iterations ($MMSE_{shuffled,p}$).

Step 4: PIV_p is calculated as follows

$$PIV_p = \left(\frac{MMSE_{p_{shuffled}} - MSE_{real}}{MSE_{real}}\right) 100$$

Variable Importance Results: India

We determine the 10 most important variables for each of the best performing machine learning models for each horizon for India. For the 1 month ahead forecast - where the best performing machine learning model was the linear elastic net model- we find that the only non-zero coefficient and thus the most important variable is the first lag of CPI (Y-o-Y). It is found that for the 3 months ahead, 6 months ahead and 12 months ahead forecasts, the sub-components of CPI and WPI, food, fuel and banking related variables contribute most significantly to the accuracy of the relevant forecast, which is broadly in line with the literature on the determinants of CPI in India. The results for the 9 months ahead forecast find lags of rainfall and Net FII and FPI to be the most importance variables for forecasting CPI in India.

Conclusion

We conducted an analysis for three emerging markets – India, China and South Africa using a variety of machine learning methods. Out of the approaches used, neural networks were most effective in reducing forecast errors relative to SARIMA or naïve forecasts. This suggests that neural networks are a good potential approach for forecasting inflation for other emerging economies where there are less number of cross-sectional and time series of data available. Future research would compare on how the forecasts in this compare to professional forecasters and central banks.

References

Athey, S. and Imbens, G. (2019). Machine Learning Methods That Economists Should Know About. *Annual Review of Economics*, 11(1).

Atkeson, A. (2001). Are Phillips curves useful for forecasting inflation?.

Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*.

Biau, O. and D'Elia, A. (2010). Euro area GDP forecasting using large survey datasets A random forest approach.

Breiman, L. (2001). Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3), pp.199-231.

Chakraborty, C. and Joseph, A. (2017). Machine Learning at Central Banks. *SSRN Electronic Journal*.

Chuku, C., Oduor, J. and Simpasa, A. (2017). Intelligent forecasting of economic growth for African economies: Artificial neural networks versus time series and structural econometric models.

Cunha Medeiros, M., Vasconcelos, G., Veiga, A. and Zilberman, E. (2018). Forecasting Inflation in a Data-Rich Environment: The Benefits of Machine Learning Methods. *SSRN Electronic Journal*.

Fisher, A., Rudin, C. and Dominici, F. (2018). All Models are Wrong but Many are Useful: Variable Importance for Black-Box, Proprietary, or Misspecified Prediction Models, using Model Class Reliance.

Friedman, J. (1999). Greedy Function Approximation : A Gradient Boosting Machine.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep learning*.

Hastie, T., Friedman, J. and Tibshirani, R. (2017). *The elements of statistical learning*. New York: Springer.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (n.d.). *An introduction to statistical learning*.

Jung, J., Patnam, M. and Ter-Martirosyan, A. (2018). An Algorithmic Crystal Ball: Forecasts-based on Machine Learning. *IMF Working Papers*, 18(230), p.1.

McAdam, P. and McNelis, P. (2005). Forecasting inflation with thick models and neural networks. *Economic Modelling*, 22(5), pp.848-867.

Molnar, C. (2019). *Interpretable Machine Learning:A Guide for Making Black Box Models Explainable*.

Mullainathan, S. and Spiess, J. (2017). Machine Learning: An Applied Econometric Approach. *Journal of Economic Perspectives*, 31(2), pp.87-106.

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

Pratap, B. and Sengupta, S. (2019). Macroeconomic Forecasting in India: Does Machine Learning Hold the Key to Better Forecasts? *RBI Working Paper*.

Raj, J. and Dhal, S. (2008). *The Inflation Rate in India: Some Applied Issues*.

Raj, J., Kapur, M., Das, P., George, A., Wahi, G. and Kumar, P. (2019). Inflation Forecasts: Recent Experience in India and a Cross-country Assessment.

Sanyal, A. and Roy, I. (2014). Forecasting Major Macroeconomic Variables in India – Performance Comparison of Linear, Non-linear Models and Forecast Combinations.

Smalter Hall, A. and Cook, T. (2017). Macroeconomic Indicator Forecasting with Deep Neural Networks. *SSRN Electronic Journal*.

STOCK, J. and WATSON, M. (2007). Why Has U.S. Inflation Become Harder to Forecast?. *Journal of Money, Credit and Banking*, 39, pp.3-33.

Tiffin, A. (2016). Seeing in the Dark: A Machine-Learning Approach to Nowcasting in Lebanon. *IMF Working Papers*, 16(56), p.1.

Tkacz, G. and Hu, S. (1999). Forecasting GDP Growth Using Artificial Neural Networks.

Tong, H. (1995). *Non-linear time series*. Oxford: Clarendon.

Varian, H. (2014). Big Data: New Tricks for Econometrics. *Journal of Economic Perspectives*, 28(2), pp.3-28.

Tables

Table 1: Hyper parameters for Tree Based Methods

Hyper-parameters	Hyper-parameters Domain	Random Forests	XG Boosted Trees
Max Tree Depth	1,depth till only pure leaves	x	x
Min. Samples for Splitting a Node	2,No. of Obs.	x	x
Min. Samples to form a Leaf	1,No. of Obs.	x	x
Max. no. of Leaf Nodes in a Tree	1, No. of pure leaves	x	x
No. of Independent Variables to Consider to Find Best Split for a Node	1, No. Independent Variables	x	x
No. of Trees	1, ∞	x	x
Columns Sample by Tree (%)	ε,1		x
Columns Sample by Level (%)	ε,1		x
Columns Sample by Node (%)	ε,1		x
L1 Regularization	1, ∞		x
L2 Regularization	1, ∞		x
Learning Rate	0,1		x
Parameters		Threshold variables and values	

*the x denotes that the hyper parameter is needed for the model under consideration

Table 2: Hyperparameters for Neural Networks

Hyperparameter	Hyperparameter Domain	CNN	CNN+ LSTM	Encoder Decoder
Patience	0, ∞	x	x	x
Learning Rate	0,∞	x	x	x
No. of Epochs	0,∞	x	x	x
Optimizer Type	SGD/ADAM	x	x	X
Batch Size	1, No. of Obs.	x	x	x
No. of Steps In	1, No. of Obs. in Test Set	x	x	x
No. of Conv. Layers	1, ∞	x	x	x
Conv. Activation Layer Type	Linear/ Tanh /Logistic/ReLU	x	x	x
No. of Filters per Conv. Layer	1, ∞	x	x	x
Filter Size	1, No. of Steps In*No. Independent Variables	x	x	x
Stride Size	1, ∞	x	x	x
Sub-sampling Layer type	Max Pooling / Average Pooling	x	x	x
Sub-sampling Layer Size	1, No. of Steps In – Filter Size	x	x	x
No. of Dropout layers	0,No. of Hidden Layers	x	x	x
Dropout Percentage	0,1	x	x	x
No. of Full Connected Hidden Layers	0, ∞	x	x	x
No. of Hidden Nodes	0, ∞	x	x	x
Batch Normalization	Yes/No	x	x	x
Output Layer Activation Type	Linear, Tanh, Logistic, ReLU	x	x	x
No. of LSTM layers	1, ∞		x	x
No. of LSTM nodes	1, ∞		x	x
No. Repeat Vector	1, ∞			x
Parameters				Network weights

*the x denotes that the hyper parameter is needed for the model under consideration

Table 3: Sample Construction

Country	Full Period	Window Size	Training Obs. per Window	Validation Obs. per Window	Test Obs. per Window	Consolidated Testing Period	No. of Independent Variables
India	1st Jan 03 – 1st Feb 19 (195 obs.)	153	138	14	1	1st Sep 16 – 1st Feb 19 (30 obs.)	48
China	1st Jan 03 – 1st July 19 (200 obs.)	158	142	15	1	1st Feb 17 – 1st July 19 (30 obs.)	30
South Africa	1st Jan 03 – 1st July 19 (200 obs.)	158	142	15	1	1st Feb 17 – 1st July 19 (30 obs.)	60

Table 4 : India: 1 Month to 12 Month Ahead Forecasts

Horizon	Category	Method	RMSE	MAE	RMSE Accuracy Increase (%)	MAE Accuracy Increase (%)
1 Month Ahead	Benchmark Model	SARIMA	2.388	1.932	-318.500	-316.24
		MA/Naive	0.571	0.464		
	Shrinkage Models	Elastic Net	0.622	0.510	-8.917	-9.97
	Tree Based Models	Random Forests	1.414	1.237	-147.752	-166.57
		XG Boost	1.191	0.877	-108.798	-88.84
	Neural Networks	CNN	0.962	0.773	-68.526	-66.43
		CNN+LSTM	0.773	0.592	-35.487	-27.55
		Encoder Decoder	0.722	0.571	-26.449	-22.93
3 Months Ahead	Benchmark Model	SARIMA	2.388	1.932	-79.713	-70.89
		MA/Naive	1.329	1.130		
	Shrinkage Models	Elastic Net	2.419	2.099	-82.040	-85.66
	Tree Based Models	Random Forests	2.211	1.928	-66.398	-70.59
		XG Boost	1.585	1.151	-19.313	-1.85
	Neural Networks	CNN	0.800	0.606	39.774	46.37
		CNN+LSTM	0.689	0.503	48.152	55.54
		Encoder Decoder	0.854	0.682	35.700	39.65
6 Months Ahead	Benchmark Model	SARIMA	2.388	1.932	-32.137	-18.13
		MA/Naive	1.807	1.635		
	Shrinkage Models	Elastic Net	1.521	1.189	15.826	27.27
	Tree Based Models	Random Forests	1.735	1.391	3.973	14.92
		XG Boost	1.903	1.564	-5.282	4.38
	Neural Networks	CNN	0.922	0.715	48.987	56.30
		CNN+LSTM	1.065	0.839	41.063	48.69
		Encoder Decoder	1.074	0.866	40.542	47.04
9 Months Ahead	Benchmark Model	SARIMA	2.388	1.932	-36.12	-25.71
		MA/Naive	1.754	1.537		
	Shrinkage Models	Elastic Net	2.208	1.790	-25.86	-16.50
	Tree Based Models	Random Forests	3.184	2.848	-81.50	-85.30
		XG Boost	1.474	1.163	15.99	24.33
	Neural Networks	CNN	1.032	0.830	41.18	46.02
		CNN+LSTM	1.106	0.785	36.98	48.91
		Encoder Decoder	0.730	0.558	58.37	63.67
12 Months Ahead	Benchmark Model	SARIMA	2.388	1.9317	-63.52	-66.06
		MA/Naive	1.460	1.1633	0.00	0.00
	Shrinkage Models	Elastic Net	2.610	2.3122	-78.75	-98.76
	Tree Based Models	Random Forests	2.995	2.7957	-105.10	-140.32
		XG Boost	2.203	1.8550	-50.86	-59.46
	Neural Networks	CNN	0.888	0.6858	39.21	41.05
		CNN+LSTM	0.846	0.6446	42.08	44.59
		Encoder Decoder	1.088	0.8483	25.47	27.08

Table 5: Forecast Bias - India: 1 Month to 12 Months Ahead Forecasts

Horizon	Category	Method	Over Estimate (%)	Wilcox Statistic: Forecast
1 Month Ahead	Benchmark Models	SARIMA	90.00	10***
		MA/Naive	56.67	194
	Shrinkage Model	Elastic Net	70.00	63***
	Tree Based Models	Random Forests	90.00	10***
		XG Boost	80.00	57***
	Neural Networks	CNN	30.00	161.13
		CNN+LSTM	64.19	129.06
		Encoder Decoder	60.00	192.03
3 Months Ahead	Benchmark Model	SARIMA	90.00	10***
		MA/Naive	70.00	137*
	Shrinkage Model	Elastic Net	86.67	24***
	Tree Based Models	Random Forests	90.00	7***
		XG Boost	66.67	126*
	Neural Networks	CNN	46.67	218
		CNN+LSTM	76.67	97**
		Encoder Decoder	49.03	210.84
6 Months Ahead	Benchmark Model	SARIMA	90.00	10***
		MA/Naive	70.00	135*
	Shrinkage Model	Elastic Net	70.00	59***
	Tree Based Models	Random Forests	76.67	46***
		XG Boost	83.33	41***
	Neural Networks	CNN	43.01	168.06
		CNN+LSTM	47.31	201.03
		Encoder Decoder	47.31	203.29

Level of significance: * 5% , **1% , ***0.1%

Table 5 (continued): Forecast Bias - India: 1 Month to 12 Months Ahead Forecasts

9 Months Ahead	Benchmark Model	SARIMA	90.00	10***
		MA/Naive	63.33	128*
	Shrinkage Model	Elastic Net	73.33	85***
	Tree Based Models	Random Forests	100.00	0***
		XG Boost	73.33	59***
	Neural Networks	CNN	43.33	195
		CNN+LSTM	82.58	45.16***
		Encoder Decoder	64.52	132.23
12 Months Ahead	Benchmark Model	SARIMA	90	10***
		MA/Naive	70.00	87***
	Shrinkage Model	Elastic Net	86.67	19***
	Tree Based Models	Random Forests	100.00	0***
		XG Boost	66.67	89***
	Neural Networks	CNN	36.67	169
		CNN+LSTM	65.59	147.06
		Encoder Decoder	53.44	208.94

Level of significance: * 5% , **1% , ***0.1%

Table 6: Emerging Market Forecasts - 12 Months Ahead Forecast

Country	Category	Method	RMSE	MAE	RMSE Accuracy Increase (%)	MAE Accuracy Increase (%)
China	Benchmark	SARIMA	1.012	0.911	-87.48	-104.08
		MA/Naive	0.540	0.447	0.00	0.00
	Neural Networks	CNN	0.527	0.365	2.27	18.23
		CNN LSTM	0.407	0.297	24.63	33.42
		Encoder Decoder	0.526	0.420	2.49	5.85
South Africa	Benchmark	SARIMA	1.217	1.112		
		MA/Naive	1.418	1.145	-16.52	-2.963
	Neural Networks	CNN	0.635	0.518	47.85	53.400
		CNN LSTM	0.590	0.469	51.52	57.849
		Encoder Decoder	0.809	0.643	33.48	42.176

Table 7: Emerging Market Forecasts - 12 Months Ahead Forecast – Forecast Bias

Country	Category	Method	Over Estimate (%)	Wilcox Statistic: Forecast
China	Benchmark	SARIMA	100	0***
		MA/Naive	46.66	163
	Neural Networks	CNN	60.00	168.03
		CNN LSTM	37.42	158.29
		Encoder Decoder	63.33	202.00
South Africa	Benchmark	SARIMA	86.67	45***
		MA/Naive	90.00	42***
	Neural Networks	CNN	53.33	195
		CNN LSTM	70.00	106***
		Encoder Decoder	62.69	154.0323

Table 8A: Neural Network Consensus Model – Forecast Accuracy

Country	Horizon	Method	RMSE	MAE	RMSE Accuracy Increase (%)	MAE Accuracy Increase (%)
India	1 Month	Average NN	0.613	0.517	-7.46	-11.35
	3 Month	Average NN	0.604	0.458	54.55	59.52
	6 Month	Average NN	0.877	0.725	51.45	55.66
	9 Month	Average NN	0.699	0.582	60.17	62.13
	12 Month	Average NN	0.710	0.559	51.35	51.93
China	12 Month	Average NN	0.395	0.312	26.82	30.02
South Africa	12 Month	Average NN	0.429	0.364	64.72	67.22

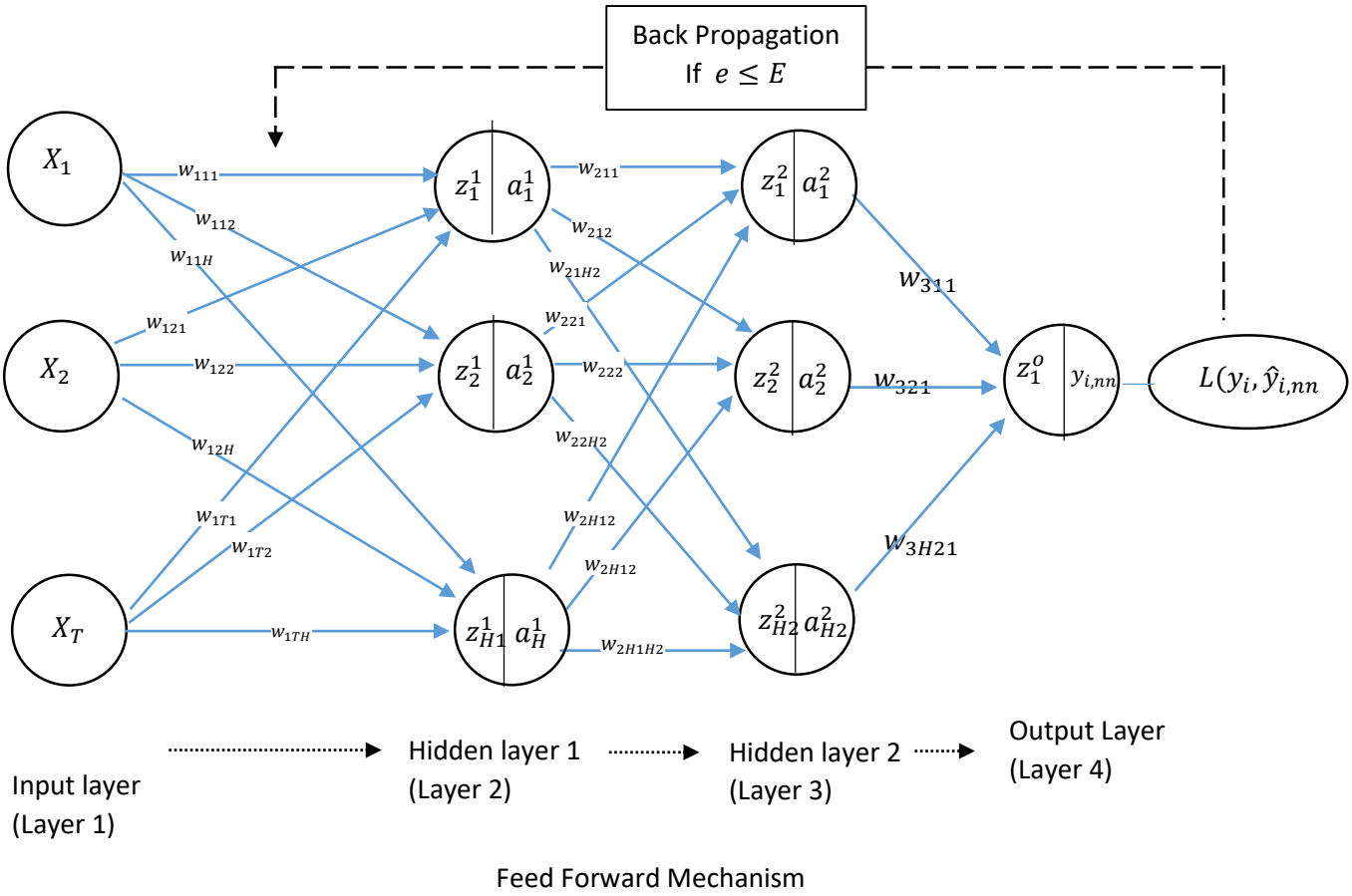
Table 8B: Neural Network Consensus Model – Forecast Bias

Country	Horizon	Method	Over Estimate (%)	Wilcox Statistic: Forecast
India	1 Month	Average NN	56.67	201
	3 Month	Average NN	50.00	213
	6 Month	Average NN	36.67	184
	9 Month	Average NN	56.67	134*
	12 Month	Average NN	53.33	220
China	12 Month	Average NN	53.33	195
South Africa	12 Month	Average NN	66.67	149

Level of significance: * 5% , **1% , ***0.1%

Graphs

Figure 1: Fully Connected Feed Forward Neural Network



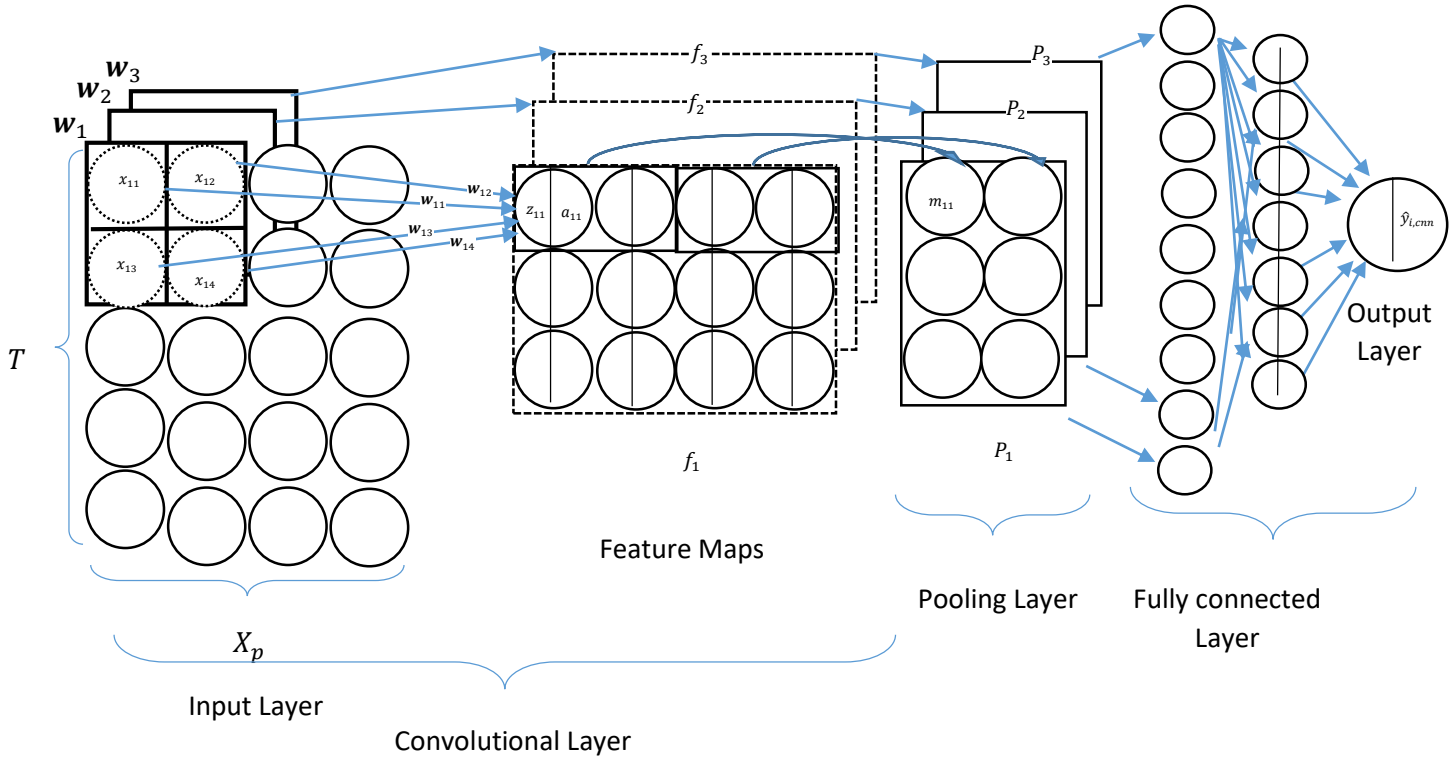
Legend

= input node (holds true for all subsequent figures)

= hidden/computational node (holds true for all subsequent figures)

T is the total number of nodes in the input layer. H is the total number of hidden nodes in the 1st hidden layer (Layer 2) and $H2$ is the total number of hidden nodes in hidden layer two (Layer 3). z_1^o is the weighted sum for the output layer and $\hat{y}_{nn}$ i.e. the predicted value of y_i the activation of the output layer.

Figure 2: Convolutional Neural Network



Legend

= input chunk x_c

= Filter w_f

= Disjoint area *d*

Figure 3: Long Short Term Memory Network

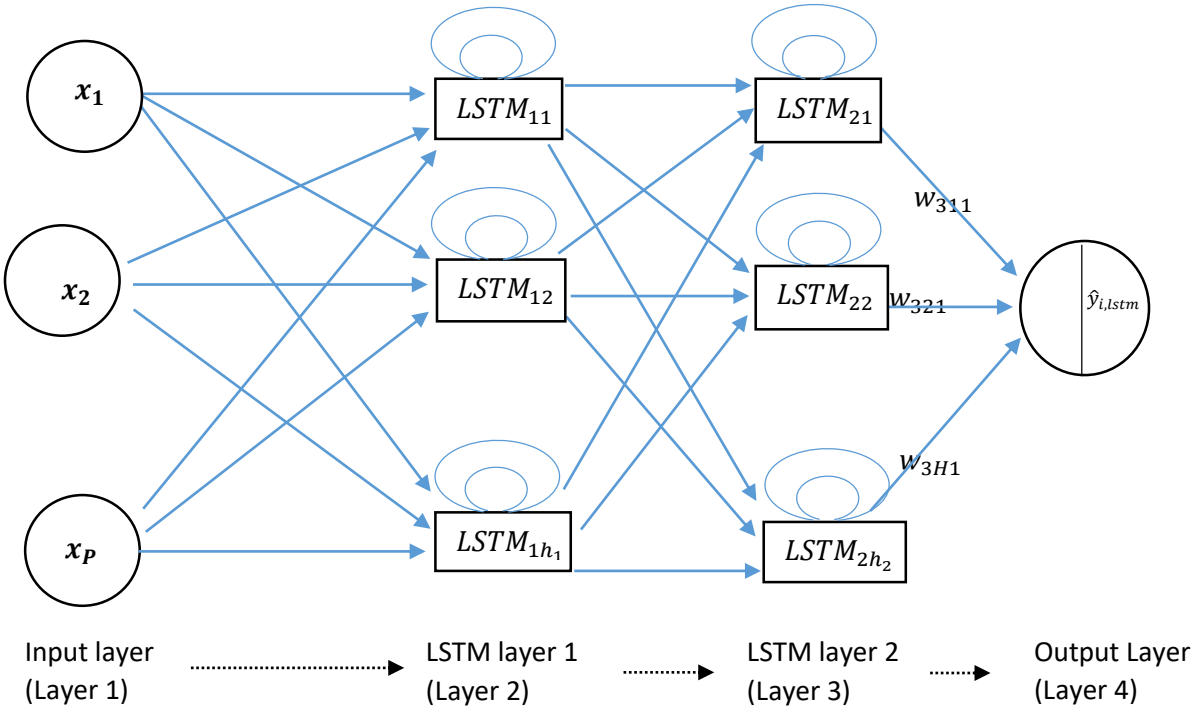
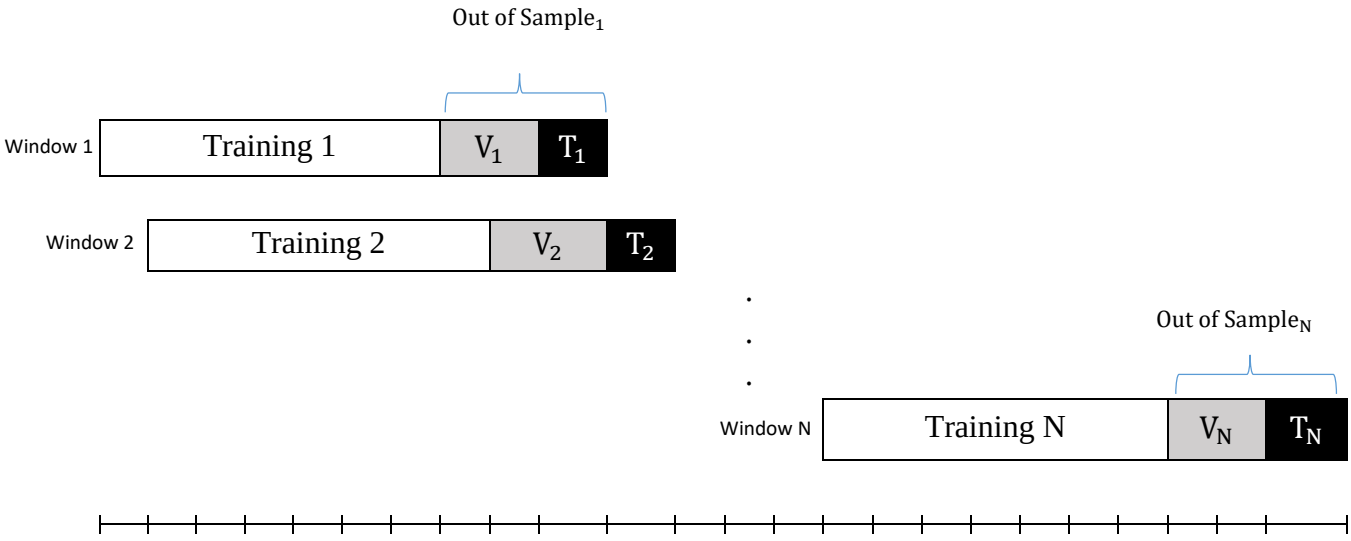


Figure 4: Rolling Window Training and Testing Split



For N windows, there are a total of N training, testing and out-of-sample (i.e. validation and testing) data sets. The last observation of every window (i.e. the test set) is the final prediction in each window such that the N final predictions are temporally sequential and non-overlapping.

Figure 5A

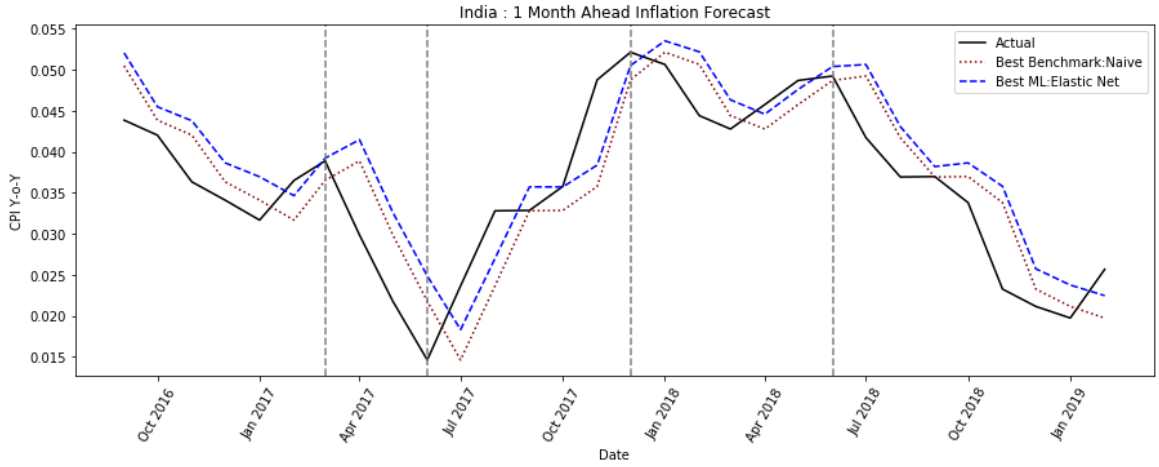


Figure 5B

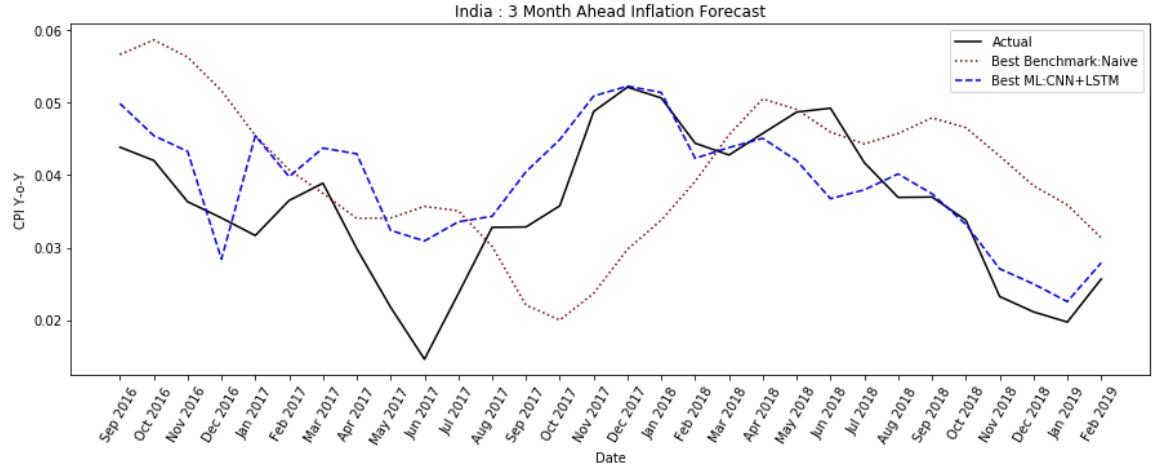


Figure 5C

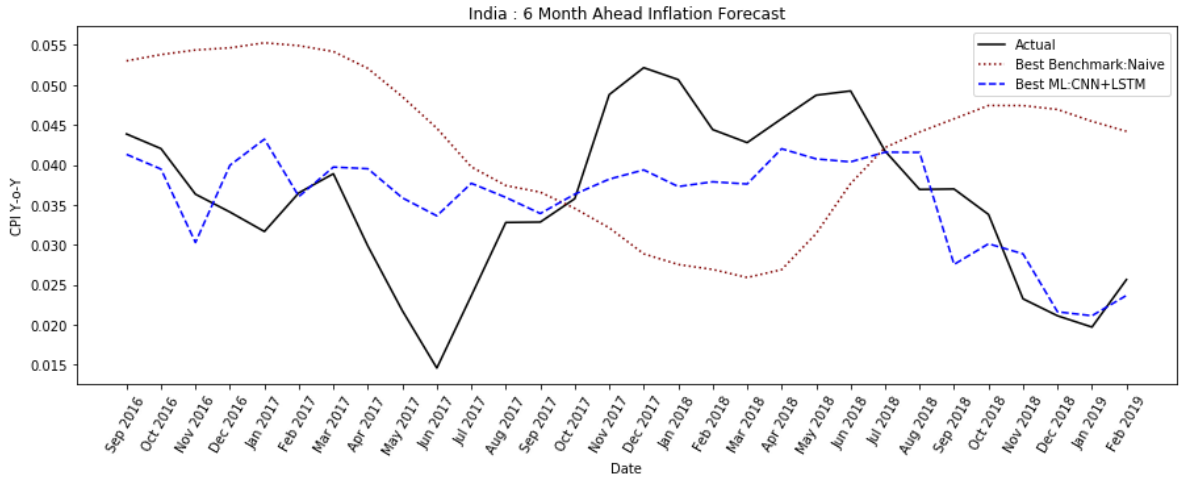


Figure 5D

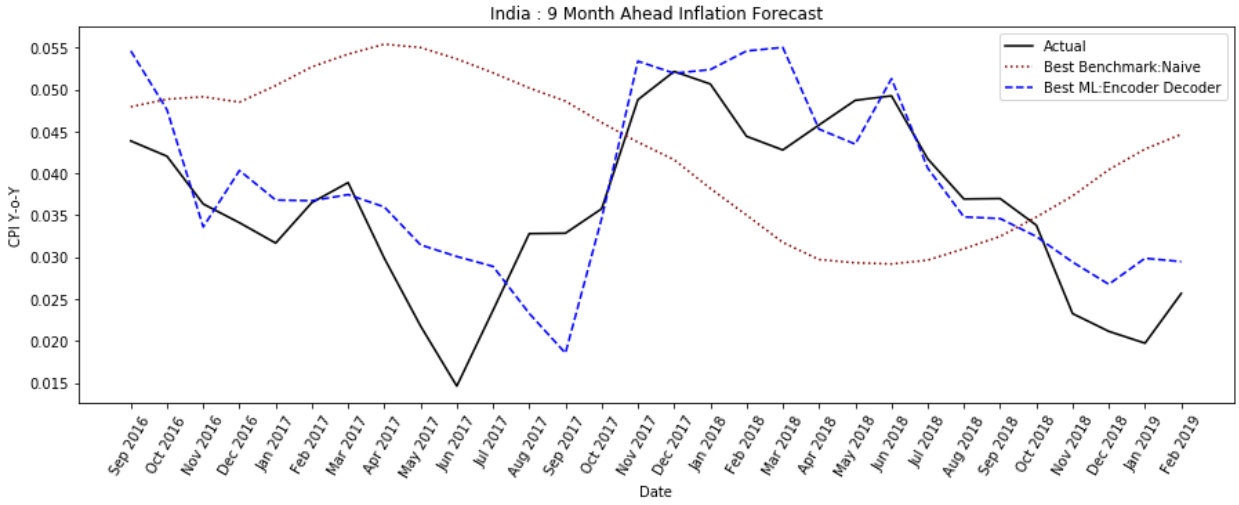


Figure 5E

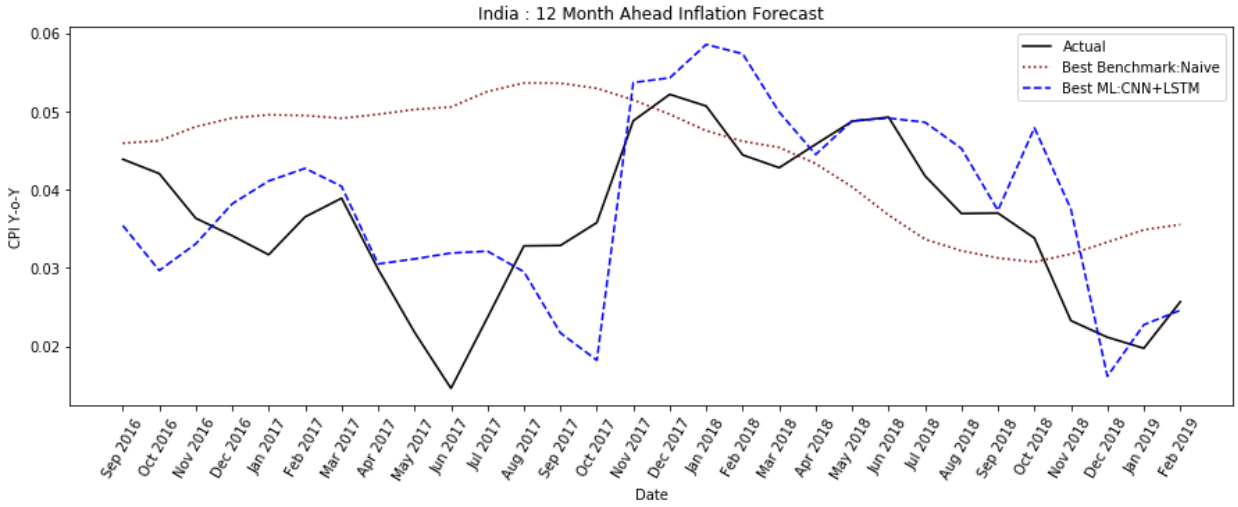


Figure 6

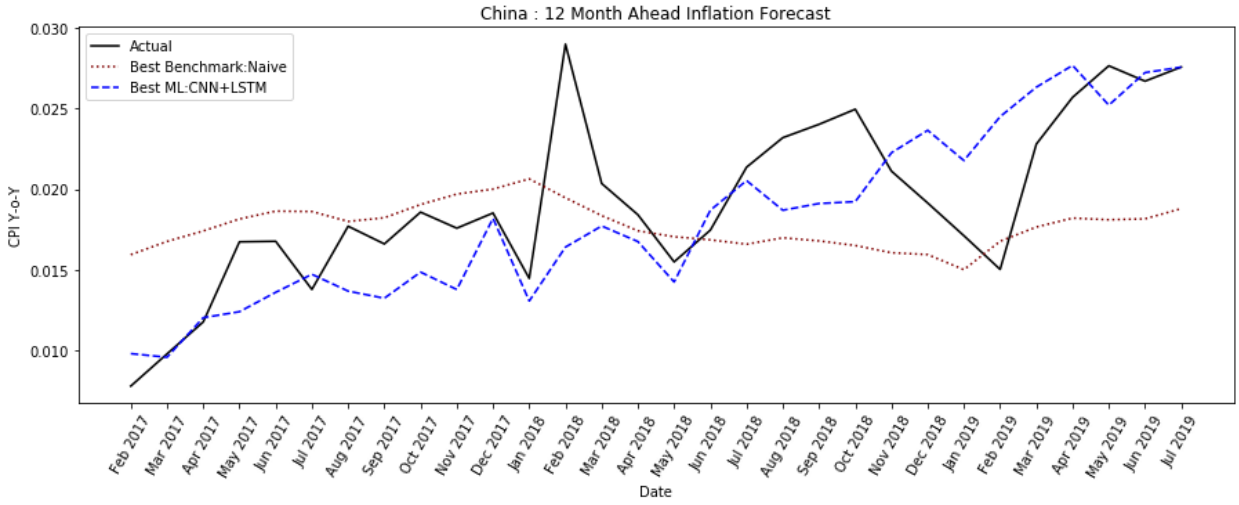
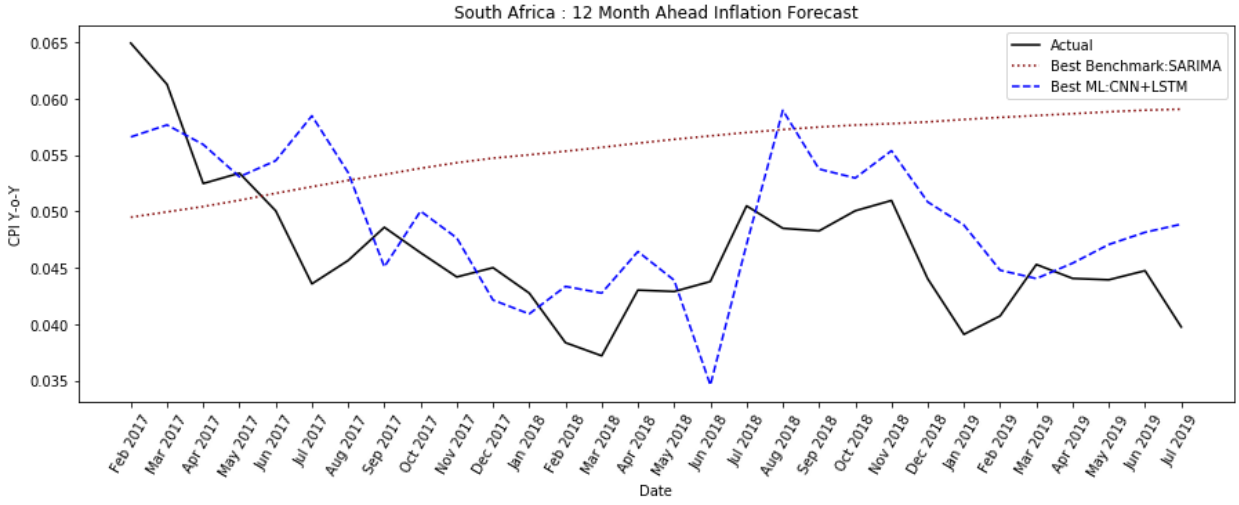


Figure 7



Appendix A

Table 1: Variable Description - India

Category	Variable Name
WPI and Sub Components	Wholesale Price Index (WPI)
	WPI Primary Articles
	WPIPA
	WPIPA Non Food
	WPI Manufacturing
	WPIMfg Food
CPI and Sub Components	Consumer Price Index (CPI)
	CPI Agricultural Labour
	CPIAL Food
	CPIAL Fuel and Light
	CPIAL Clothing
	CPIAL Miscellaneous
Oil Related Indicators	West Texas Intermediate (USD)
	Crude Petroleum Production
	Petrol Price
	Diesel Price
Food Related Indicators	Food Grain Stock
	Wheat Stock
	Rice Stock
	Rainfall
Auto Industry Related Indicators	Monthly vehicle sales
	Total vehicle sales
	Total Automobile Production
	Automobile Production (Passenger Vehicle)
	Automobile Production (Commercial Vehicle)
	Automobile Production (Commercial Vehicle -2 Wheeler)
	Automobile Production (Commercial Vehicle -3 Wheeler)
	Total Automobile Sales
	Automobile Sales (Passenger Vehicles)
	Automobile Sales (Commercial Vehicle)
	Automobile Sales (2 Wheeler)
	Automobile Sales (3 Wheeler)
Electricity Related Indicators	Electricity Generation
Monetary Policy and Finance Related Indicators	Repo Rate
	Reverse Repo Rate
	Cash Reserve Ratio
	Statuary Liquidity Ratio
	Bank Rate
	M1 Money Supply

	M2 Money Supply
	Commercial Bank Deposits
	Total Liabilities RBI
	Reserve Money
Trade Related Indicators	India USD Forex
	Net Foreign Institutional Investment (FII)/ Foreign Portfolio Investment (FPI)
	Net Foreign Institutional Investment (FII)/ Foreign Portfolio Investment (FPI) in Equity
	Net Foreign Institutional Investment (FII)/ Foreign Portfolio Investment (FPI) in Debt
	Trade Balance
Miscellaneous	Cement Production

Table 2: Variable Description – China

Category	Variable Name
Monetary Policy and Finance Related Indicators	M0 for China
	M2 for China
	Total Reserves excluding Gold for China
	Total Reserves excluding Gold for Province of China Taiwan
	Share Prices: All shares/broad: Total: Total for China
	Total Share Prices for All Shares for China
CPI and Sub Components	Consumer Price Index: All Items for China
	Consumer Opinion Surveys: Confidence Indicators: Composite Indicators
	Consumer opinion surveys: Economic Situation: Future tendency
Trade Related Indicators	Net Trade: Value Goods for China
	Ratio of Exports to Imports for China
	Broad Effective Exchange Rate for China
	Exports: Value Goods for China
	Imports: Value Goods for China
	International Trade: Exports: Value (goods): Total for China
	International Trade: Imports: Value (goods): Total for China
	U.S. Exports of Goods by F.A.S. Basis to China Mainland
	U.S. Imports of Goods by Customs Basis from China
	National Currency to US Dollar Exchange Rate: Average of Daily Rates
	National Currency to US Dollar Spot Exchange Rate for China
	Real Broad Effective Exchange Rate for China
	Real Effective Exchange Rates Based on Manufacturing Consumer Price
	China / U.S. Foreign Exchange Rate
OECD Composite Leading Indicators	Leading Indicators OECD: Leading indicators: CLI: Amplitude adjusted
	Leading Indicators OECD: Leading indicators: CLI: Normalised
	Leading Indicators OECD: Leading indicators: CLI: Trend restored
	Leading Indicators OECD: Reference series: Gross Domestic Product
	Economic Policy Uncertainty Index: Mainland Papers for China
Production Related Indicators	Producer Prices Index: Economic activities: Industrial activities:
	Production: Construction: Total construction: Total for China
	Sales: Retail trade: Total retail trade: Value for China
	Business Tendency Surveys for Manufacturing: Confidence Indicators

Table 3: Variable Description – South Africa

Category	Variable Name
CPI and Sub Components	Consumer Price Index: All Items for South Africa
	Consumer Opinion Surveys: Confidence Indicators: Composite Indicators:
	Consumer Price Index: All Items Excluding Food and Energy for South
	Consumer Price Index: Energy for South Africa
	Consumer Price Index: Food and non-Alcoholic beverages (COICOP 01):
	Consumer Price Index: Housing water electricity gas and other fuels
Trade Related Variables	Broad Effective Exchange Rate for South Africa
	National Currency to US Dollar Exchange Rate: Average of Daily Rates
	National Currency to US Dollar Spot Exchange Rate for South Africa
	Real Broad Effective Exchange Rate for South Africa
	Real Effective Exchange Rates Based on Manufacturing Consumer Price
	South Africa / U.S. Foreign Exchange Rate
	Total Retail Trade in South Africa
	U.S. Exports of Goods by F.A.S. Basis to South Africa
	U.S. Imports of Goods by Customs Basis from South Africa
	Ratio of Exports to Imports for South Africa
	Exports: Value Goods for South Africa
	Imports: Value Goods for South Africa
Monetary Policy and Finance Related Indicators	3-Month or 90-day Rates and Yields: Interbank Rates for South Africa
	3-Month or 90-day Rates and Yields: Treasury Securities for South
	Immediate Rates: Less than 24 Hours: Call Money/Interbank Rate for
	Immediate Rates: Less than 24 Hours: Central Bank Rates for South
	Interest Rates Government Securities Government Bonds for South
	Interest Rates Government Securities Treasury Bills for South Africa
	Long-Term Government Bond Yields: 10-year: Main (Including Benchmark)
	Long-Term Government Bond Yields: Combined Terms for South Africa
	M3 for South Africa
	Monetary aggregates and their components: Broad money and components:
	Share Prices: All shares/broad: Total: Total for South Africa
	Total Share Prices for All Shares for South Africa
	Total Reserves excluding Gold for South Africa
Production Related Indicators	Domestic Producer Prices Index: Manufacturing for South Africa
	Orders: Construction: Permits issued: Dwellings / Residential
	Producer Prices Index: Economic Activities: Domestic Manufacturing for
	Producer Prices Index: Economic Activities: Domestic Mining and
	Producer Prices Index: Economic activities: Industrial activities:
	Producer Prices Index: Economic activities: Manufacturing: Domestic
	Producer Prices Index: Economic activities: Mining and quarrying
	Production in Total Manufacturing for South Africa
	Production of Total Construction in South Africa

	Production: Construction: Total construction: Total for South Africa
	Production: Energy: Electricity: Total for South Africa
	Production: Manufacturing: Total manufacturing: Total manufacturing
	Production: Mining: Total mining: Total for South Africa
	Business Tendency Surveys for Manufacturing: Confidence Indicators
	Sales: Manufacturing: Total manufacturing: Value for South Africa
	Sales: Retail trade: Car registration: Passenger cars for South Africa
	Sales: Retail trade: Total retail trade: Value for South Africa
	Sales: Retail trade: Total retail trade: Volume for South Africa
	Sales: Wholesale trade: Total wholesale trade: Value for South Africa
	Sales: Wholesale trade: Total wholesale trade: Volume for South Africa
Automobile and Housing Related Indicators	Passenger Car Registrations in South Africa
	Permits Issued for Dwelling in South Africa
OECD Composite Leading Indicators	Leading Indicators OECD: Component series: BTS - Business situation
	Leading Indicators OECD: Component series: BTS - Demand or orders
	Leading Indicators OECD: Component series: Car registration - sales
	Leading Indicators OECD: Component series: Construction: Original
	Leading Indicators OECD: Component series: Interest rate spread
	Leading Indicators OECD: Component series: Share prices: Original
	Leading Indicators OECD: Leading indicators: CLI: Amplitude adjusted
	Leading Indicators OECD: Leading indicators: CLI: Normalized