ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economcis

Bachelor Thesis Econometrics and Operations Research

Forecasts of the Tanzanian gross domestic product including factor models and the machine learning technique boosting

Author: Lise Lot Ridderbos (451248)

Supervisor: P.H.B.F. Franses

Second assessor: A.M. Schnucker

Date final version: 7 July 2019

Abstract

There has been extensive research on how 'big data' and machine learning techniques are useful for modeling low frequency macroeconomic variables. In this paper, it is analyzed if machine learning, variable selection, and shrinkage method boosting within the factor-augmented auto-regression are helpful in forecasting the gross domestic product (GDP) of Tanzania. In addition, it is tested whether this forecast method outperforms the 'simple' auto-regressive (AR) with the lag order selected by the Bayesian information criterion. The estimation is based on a combination of the lagged GDP growth of Tanzania and the dynamical factors. The latter are based on principal component analysis (PCA) and contain the GDP growth of 51 other African countries from 1963 up to 2016. The evaluation of the effectiveness of the forecasts methods is based on simulations that include a data generating process. It was found that this method proved to generate reliable results, given that the input values are accurately described. The five forecasts of the GDP of Tanzania result all in a lag order of 2 selected by the Bayesian information criterion. The MSFE of the boosting method that includes principal component analysis is lower than the MSFE of the benchmark AR(2) method. Therefore, it was found that the factor models and the machine learning technique boosting are indeed reliable means for modeling and forecasting the gross domestic product of Tanzania, and they outperform the 'simple' AR(2) model.

Contents

1 Introduction **2**

2 Literature Review: Using Machine Learning Techniques to Forecast the Gross Domestic Product Growth **3**

3 Data **4**

4 Methodology **5**

4.1 Factor and Forecasting Models 5

4.2 Factor Estimation Method: Principal Component Analysis 7

4.3 Machine Learning, Variable Selection, and Shrinkage Method: Boosting 9

4.4 Auto-egressive Models and Mean Square Forecast Error 10

4.5 Simulation With Data-generating Process 11

5 Results **13**

5.1 Results of the simulation 13

5.2 Results for the African Data 17

6 Conclusion **20**

7 Appendix **24**

7.1 Nomenclature 24

7.2 Results 25

7.3 Codes 27

7.3.1 Simulation 27

7.3.2 Codes with the African Data Set 39

1 Introduction

In the last five decades technological advances have resulted in an increase in the quantity of available macroeconomic data (Kim and Swanson (2014)). The amount of 'big data' and machine learning techniques that are available for modeling and computational methods is constantly increasing. These concepts are used in several studies in the economic literature to predict factor models (Forni et al. (2000), Stock and Watson (2002b), Stock and Watson (2006), Bai and Ng (2008), Dufour and Stevanovic (2010), Stock and Watson (2012), Kim and Swanson (2014) and Hassani and Silva (2015)).

In the process of forecasting, a large set of predictors needs to be examined. It can be a costly process to evaluate all possible combinations of the predictors if they have no specific ordering. Solutions to this costly process would be machine learning variable selection and shrinking methods. In the paper of Kim and Swanson (2018) it is concluded that 'big data' and machine learning techniques are useful for forecasting low frequency macroeconomic variables. There are various 'robust' shrinking techniques for forecasting, mentioned in Kim and Swanson (2018). In this paper the statistical learning algorithm boosting is used as a methodology of selecting the predictors in factor-augmented auto-regressions. The factors are constructed by principal component analysis (PCA). This is the best known and most frequently used technique in dimension-reduction (Jolliffe (2011)). PCA reduces the variables, while it retains much of the information of the original data set (Kim and Swanson (2018)).

The goal of this research is to forecast the low frequency macroeconomic variable gross domestic product (GDP) growth of Tanzania using the lagged GDP growth of Tanzania and the GDP growth of other African countries. The paper of Kim and Swanson (2018) is followed, by using one of their forecast approaches. This research extends the previous paper by using new data, namely the GDP growth of 52 countries in Africa from 1961 up to 2016. In addition, it is analyzed whether this forecast method outperforms a 'simple' auto regressive model with a lag order selected by the Bayesian information criterion. Hence the following main research question is formulated: *Are factor models and the machine learning technique boosting helpful for modeling and forecasting the gross domestic product of Tanzania, and does this forecast method outperform the 'simple' auto-regressive model?*

In order to answer the main research question, the forecast performances need to be adequately

evaluated. This is done by comparing the mean square forecast error (MSFE) of the forecast method with the MSFE of an auto-regressive forecast with lag order selected by the Bayesian information criterion. To evaluate the effectiveness of the factor model that includes boosting and PCA in a FAAR framework, data is simulated using a data-generating process. The input data from the data generating process is constructed in the same format as the African data set. Therefore, the problems in the simulation apply to the forecast of the GDP growth of Tanzania. As a consequence, the results can be compared to test the accuracy of the method. It was found by the simulation that the forecast method is specified correctly. The factor models and the machine learning technique boosting proved to be a reliable tool for modeling and forecasting the gross domestic product of Tanzania, and it outperform the 'simple' AR(2) model.

Answering this research question leads to valuable insights. It expands the knowledge of how 'big data', machine learning, variable selection, and shrinkage method boosting are used to predict factor models and, in specific, the GDP growth of Tanzania. Additionally, the methods used in this research can be applied to model other low frequency macroeconomic variables. Furthermore, the research validates the applicability of replacing 'simple' auto-regressive models with a boosting forecast method to increase the accuracy of the forecast.

The structure of the article is as follows: In the next section a brief summery of the relevant literature on the field of machine learning techniques to forecast gross domestic product growth is given. Consequently, section 3 elaborates on the sources of the required data. In the fourth section the research methodology is explained. In section 5, the methodology is implemented and the results are discussed. Finally, in the last section a conclusion is presented and recommendations for further research are proposed.

2 Literature Review: Using Machine Learning Techniques to Forecast the Gross Domestic Product Growth

There has been extensive research on the relationship between forecast low frequency macro economic variables like gross domestic product (GDP) growth and machine learning techniques. As mentioned in Hassani and Silva (2015) the potential of machine learning techniques that can handle big data will further increase in the future. In their research the authors examined the usefulness of these machine learning techniques for forecasting models. The authors concluded that factor models are

widely used to make these forecasts and the field of economics is most popular of exploiting big data by machine learning techniques. As mentioned in Plakandaras et al. (2015) machine learning techniques are becoming more important, because of their ability to forecast with a higher accuracy, compared to alternative methods such as 'simple' auto-regressive models.

One of the studies that uses machine learning techniques to forecast low frequency macro economic variables like GDP growth is Schumacher (2007). Forecasts of the German GDP growth is given. Dynamic and static principal components (PC) are used for these forecasts. The main conclusion of this article is that the factor models outperform the 'simple' auto-regressive (AR) models in these forecasts. Not only the forecasting of the GDP growth of Germany can be interesting, but also the forecasting the GDP growth of the United States can be of great importance. In the paper of Carriero et al. (2015), GDP growth predictions for the United States are given. More globally, Biau (2013) forecasts the European Union GDP growth with the machine learning technique Random Forest. This forecast method outperforms the 'simple' AR model. The GDP growth in the Euro zone can also be interesting to forecast. Factor models with maximum likelihood estimations are used for this forecast (Bańbura and Modugno (2014)).

Boosting can be used as a machine learning technique to forecast growth in GDP. In the paper of Kim and Swanson (2018), the GDP growth is estimated with a boosting algorithm. The success of the boosting algorithms have caused a lot of attention (Kim and Swanson (2018). The application of boosting to classification methodology is done by Hastie et al. (2009) and the authors conclude that in this classification boosting is one of the most important developments in this field. This research follows the footsteps of Kim and Swanson (2018), by using their approaches to forecast the GDP growth.

3 Data

The data set used in this paper concerns the gross domestic product (GDP) growth of 52 African countries from 1961 till 2016. The data is obtained from an unpublished paper (Franses and Vasilev (2019)). This paper presents a balanced panel data set for real GDP growth growth in Africa. The data of Ghana is moved by one row. This is corrected by starting the data of Ghana one year earlier. The data from 1961, 1962 and 2017 till 2020 are deleted because these data points do not conclude a GDP growth for a few African countries. So the data that is used in this research includes the

GDP growth of 52 African countries from 1963 till 2016.

The maximum GDP growth concerning all African Countries is the GDP growth of Eq. Guinea from 1997 till 1998 and is equal to 150. The minimum GDP growth concerning all African countries is the GDP growth of Libya from 2011 till 2012 and is equal to -62.1. The overall average GDP growth is equal to 3.985. In this research the GDP growth of Tanzania is represented in the dependent variable y, the GDP growth of the other 51 countries are represented by the explanatory variables X. The average and standard deviation of the GDP growth of Tanzania and the other African countries are given in figure 7 in the section 7.2 of the appendix.

When considering the whole data set, the GDP growth of the African countries from 1963 till 2016, the skeweness has a value of -0.0788 and the kurtosis has a value of 2.428. To test if the GDP growth of Tanzania from 1961 till 2016 has a normal distribution a Jarque Bera test is applied. The value of this Jarque Bera test is 0.79 and the *p-value* is 0.67, so the null hypothesis of a normal distribution is not rejected at a significance level of 5 %.

4 Methodology

This section contains five parts. The first subsection concerns how the factor and forecasting models are obtained and constructed. The next part discusses the principal component analysis as a factor estimation method. The third subsection examines boosting as a machine learning variable selection and shrinking method. Subsection 4 analyses the auto-regressive models and mean square forecast error. The last section elaborates on a simulation to test for the validity of the implemented factor and forecasting models.

4.1 Factor and Forecasting Models

In this research the factor and forecasting models follow the general framework described in Kim and Swanson (2018), Stock and Watson (2002a) and Bai and Ng (2009). The variance of the forecast dependent variable is described by structural components called factors. In specific, y is the time-variable to be constructed, and X are the N-dimensional multiple time series containing candidate predictors. The forecasts, y_{t+1} are created in two steps. Firstly, the factors F_t are identified using the data set X_t. Thereafter these factors are used to forecast the dependent variable y_t. The factor

model with the time series data set consisting of T observations (t=1 ... T), and N explanatory variables is defined as:

$$X_t = \Lambda' F_t + \epsilon_t, \tag{1}$$

where $X_t = (X_{t,1}, \ldots, X_{t,N})$, $F_t = (f_{t,1}, \ldots, f_{t,r})$ is a vector of $r < N$ common dynamic factors for time period t and is extracted from the factors F_t, $\Lambda = (\lambda_1, ..., \lambda_r)$ are factor loadings associated with these dynamic factors, and ϵ_t is the idiosyncratic components of X_t. The product $\Lambda' F_t$ is defined as the common product of X_t.

The general forecasting equation in a time series data set with recursive window consisting of T observations (t=1, ... T) with the factor augmented auto-regression (FAAR) formulation in (1) takes the form:

$$Y_{t+1} = W_t \beta_w + F_t \beta_F + \epsilon_{t+1}, \tag{2}$$

Y_{t+1} is a vector of $(t+1) \times 1$ consisting of the dependent variable and a forecast of this dependent variable, W_t is a $t \times (p_{max} + 1)$ matrix with additional explanatory variables and consists of a $t \times 1$ vector equal to ones for the constants and the remaining columns consists of the lags of y, β_W and β_F are the forecast parameters and ϵ_{t+h} is the disturbance error. It is important to note that if $\frac{\sqrt{T}}{N} \to 0$ the ordinary least squares estimates β_W and β_F are asymptotically normal and $\sqrt{T}$ is consistent, so that they do not cause regressor problems (Bai and Ng (2008)).

In this research, principal component analysis (PCA) is used to estimate (1), and is discussed in the subsection 4.2. Thereafter in section 4.3, the machine learning boosting method is described to make a forecast similarly as (2), and to estimate $\hat{\beta}_F$. In comparison to the factors included in F_t from Kim and Swanson (2018), these factors do not include lags from periods before t. The general framework for the dynamic factor and forecast models are still applicable, just by setting the lags equal to zero. The forecast method is the same as specification type one in Kim and Swanson (2018): first factors are constructed by PCA with the data set, secondly the machine learning shrinkage method boosting is implemented in a factor augmented auto-regressive (AR) framework to select functions of and weights for the factors. These functions and weights are used in the prediction method given in (2). So in the boosting algorihtm $beta_F$ is estimated, PCA estimate F_t and $W_t \beta_w$ is estimated with an AR model.

In the end the resulting forecasting model is compared by the mean squared prediction error with the use of a benchmark model. This benchmark model includes the AR model with the amount of lags (p_{bic}) selected by the Bayesian information criterion, explained in section 4.4. The amount

of lags to include in the forecast are also estimated using this Bayesian information criterion with the principal component regression. The amount of factors to include in the boosting method is estimated using a selection criteria from Bai and Ng (2002), and is explained in section 4.3. To evaluate the effectiveness of the factor estimation method, a simulation is considered. The data in this simulation is established by a data-generating process. The experimental Setup is shown below in figure 2.

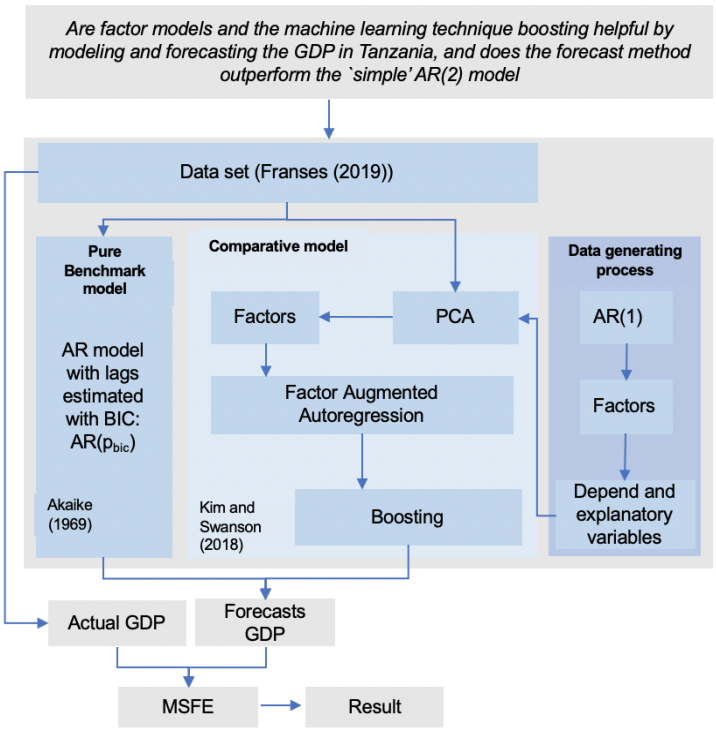


Figure 1: Visual representation of the experimental setup of this paper.

4.2 Factor Estimation Method: Principal Component Analysis

The main reason why principal component analysis (PCA) is chosen as the factor estimation method is, because in the paper of Kim and Swanson (2014) PCA is the mean square forecast error (MSFE) 'best' factor estimation for specification method one. PCA is a factor estimation method, and is used to solve (1). The number of underlying factors (r) are also estimated with PCA. In this research the latent principal components (PCs) are estimated on the same way as in Kim and Swanson (2014), with the statistical procedure PCA. The data projection is in the direction of the maximum variance, so the first principal component captures the maximum variance possible. The second

factor captures the maximum variance possible in the remaining orthogonal subspace, and is thus uncorrelated with the first component. This process is repeated to obtain the remaining factors.

Possibly correlated variables in a set of observations are orthogonally transformed into linearly uncorrelated principal components (Jolliffe (2011)). Furthermore, the first principal components related with higher variances, are believed to have more explanatory power than those later on with lower variances. Before using the method PCA the data is standardized, the mean of the data is equal to zero and there is an unit variance. Because of the normalization all variables are treated on the same "scale". The main advantage of PCs is that they are easily derived due to the singular value decomposition (Bai and Ng (2002), Stock and Watson (2002b)). This could be the reason that PCA is most frequently academically applied for factor estimation.

As discussed by Fan and Yao (2017), PCA makes use of (1) but the amount of factors r is unknown, so contrarily $F_t = (f_{t,1}, ..., f_{t,N})$ and $\Lambda = (\lambda_1, \ldots, \lambda_N)$. The variance of the j^{th} row of the explanatory variable X concerning a time series data set consisting of T observations (t=1, ... T) has the following explicit form:

$$\hat{V} = \frac{1}{T}\sum_{t=1}^{T}(X_t - \overline{X})(X_t - \overline{X})', \tag{3}$$

where $X_t = (x_{t,1}, ..., x_{t,N})$ and ϵ_t is the idiosyncratic component of X_t, $\overline{X} = \frac{1}{T}\sum_{t=1}^{T} X_t$ is the sample mean vector.

Furthermore, the j^{th} principal component is equal to the linear combinations $f_{j,t} = \lambda_j' x_{t,j}$ that solve the following maximization problem, for $j = 1, \ldots, N$, for $i = 1, \ldots, j-1$ and $t = 1, \ldots, T$:

$$\begin{aligned} \underset{\lambda_j}{\text{maximize}} \quad & var(f_{j,t}) = \lambda_j' \hat{V} \lambda_j \\ \text{subject to} \quad & \lambda_j' \lambda_j = 1, \\ & Cov(f_{j,t}, f_{i,t}) = 0 \end{aligned} \tag{4}$$

In this way the principal components are normalized and the factor's loading can be identified. When taking the Lagrangian form, where l is the Lagrange multiplier: $L_j = \lambda_j' \hat{V} \lambda - l(\lambda' \lambda - 1)$. Setting the derivative equal to zero results in: $\hat{V}_j \lambda_j = l\lambda_j$, which leads to an eigenvector λ_j of $\hat{V}$.

As discussed before the amount of factors is unknown when elaborating in PCA. The method estimates all N eigenvectors, but only the first r eigenvectors are used. The selection criteria for r are given in Bai and Ng (2002). In this research the same selection criteria Bayesian information

criterion 3 (BIC_3) is used as in the research of Kim and Swanson (2014), and is analyzed as follows:

$$
\begin{aligned}
V(r,\hat{F}) &= \min_{\Lambda} \frac{1}{NT}\sum_{j=1}^{N}\sum_{t=1}^{T}(X_{j,t}-\lambda_j'\hat{F}_t)^2 \\
BIC_3(r) &= V(r,\hat{F}) + r\hat{\sigma}^2\frac{((N+T-r)\ln(NT)}{NT} \\
\hat{r} &= \underset{0\leq r\leq r_{max}}{\operatorname{argmin}} BIC_3(r),
\end{aligned}
\tag{5}
$$

where the sum of squared residuals divided by NT is defined as $V(r,\hat{F})$, $\hat{\sigma}^2$ is the variance of residuals with r factors and r_{max} is the maximal amount of factors. Additionally, the shrinkage method boosting is also used to specify the amount of factors and the function of factors. The selection criterion above is used as a 'pre-selection' of factors before applying the boosting algorithm. This 'pre-selected' amount of estimated amount of factors ($\hat{r}$) is estimated consistently by Kim and Swanson (2014). In the following section the boosting algorithm is discussed.

4.3 Machine Learning, Variable Selection, and Shrinkage Method: Boosting

In this research the goal of a robust machine learning shrinking method is to select functions of and weights for the factors, so estimate parameters β_F in (2). The approach boosting is used in this research to make a forecast for y. When applying boosting in a factor augmented auto-regression (FAAR) framework the amount of predictors can be computed and the dependent variable y_t can be forecast similarly as in Kim and Swanson (2013) and Kim and Swanson (2018). Boosting is a method that merges the outputs of several 'weak-learners" (models) to create a forecast. The forecast is created by minimizing a quadratic loss function averaged over the training data. This forecast results in the lowest mean squared error (MSE) rates in the research of Bai and Ng (2009). When applying the boosting algorithm, boosting can be seen as a gradient descent technique. The boosting algorithm in this research handles time-series, on the same way as the 'Component-Wise L_2 Boosting' algorithm in Bai and Ng (2009). To prevent over-fitting with estimated predictors a stopping rule is included in this algorithm. This algorithm is convenient even when the number of potential predictors is large.

The boosting algorithm used in this research is given below as boosting algorithm 1. The estimation of $F_t'\beta_F$ from (2) is represented as μ^M in the boosting algorithm 1. When applying this algorithm the estimates of $\hat{\beta}_w$, the $T\times(p_{max}+1)$ matrix W_t containing additional explanatory

variables and residual vector E need to be established a priori. These parameter are estimated by the auto-regressive (AR) model, and this model is analyzed in the following section.

Algorithm 1 *Boosting*

Input Dependent variable y, dynamic factors F_t, stopping parameter M, step length v where $0 \leq v \leq 1$, amount of "pre-selected parameters" r, $\hat{\beta}_w$, Wt from (2) and $T \times 1$ residual vector E

Output Forecast of the dependent variable and the optimal beta ($\hat{\beta}^M$)

Initialize $\hat{\mu}^0 = \overline{E}$ and $B^0 = \frac{1'_T}{1_T}$, where 1_T is a T x 1 vectors of ones

for i=1, ..., M iterations **do**

the "current residual" is defined as: $\mu = E - \hat{\mu}^{i-1}$

for j=1, ..., r **do**

regress the "current Tx1 residual" μ on $\hat{F}_j$ to obtain the residuals $\hat{d}_j$

$SSR_j = \hat{d}'_j \hat{d}_j$

end for

$j_* = \min_{j \epsilon (1,...,r)} SSR_j$

obtain $\hat{\beta}_{j_*}$ by regressing μ on $\hat{F}_{j_*}$, and $\hat{\beta}^{\oplus}_{j_*}$ is non-zero only in the j_* position

$\hat{\mu}^i = \hat{\mu}^{i-1} + v\hat{F}_{j_*}\hat{\beta}_{j_*}$ ▷ Update residuals estimates using shrinkage parameters

$\hat{\beta}^i = \hat{\beta}^{i-1} + v\hat{\beta}^{\oplus}_{j_*}$ ▷ Update beta estimates using shrinkage parameters

$df^i = trace(B^{i-1} + v\hat{F}_{j_*}(\hat{F}'_{j_*}\hat{F}_{j_*})^{-1}\hat{F}'_{j_*}(I_T - B_{i-1}))$ ▷ Updating the degrees of freedom

$\hat{\sigma}^{i^2} = \sum_{t=1}^{T}(Y_t - \hat{\mu}^i)^2$ ▷ Estimating the variance of the boosting residuals

IC(i) = $\log(\hat{\sigma}^{i^2}) + \frac{log(T)df^i}{T}$

end for

M= argmin$_i$ $IC(i)$ ▷ Estimating the stopping parameter M

$\hat{y}^{Boosting}_{t+1} = W_t\hat{\beta}_w + \hat{\mu}^M$

4.4 Auto-egressive Models and Mean Square Forecast Error

The auto-regressive (AR) model with optimal lag order p_{bic} is used to estimate the additional explanatory variable W_T and parameter $\hat{\beta}_w$, and the $T \times 1$ residual vector E in the boosting algorithm 1. The AR model of lag order p_{max} has the following form:

$$y_t = \alpha + \sum_{i=1}^{p_{max}} \beta_i y_{t-i} + \epsilon_t, \tag{6}$$

where p_{max} is the maximum amount of lags, α is a constant, ϵ_t is white noise and $(\beta_1, \ldots, \beta_{p_{max}})$ are the parameters of the model. The optimal amount of lags p_{bic} is selected by minimizing the Bayesian information criterion (BIC) in the auto regressive model above (Heij et al. (2004) and Box et al. (2015)). The BIC value is defined as:

$$BIC = -2(logL) + N * log(T), \tag{7}$$

where logL is defined as the optimized log likelihood function, obtained by the estimation of the AR model. The AR model with optimal lag order p_{bic} is used as benchmark. This benchmark is used to evaluate the forecast performance of the factor model that includes boosting and PCA, by comparing this model with the 'simple' AR model with lag order P_{bic}. This comparison is evaluated by means of mean square forecast errors (MSFE). The definition of the MSFE is stated in (Kim and Swanson (2018)), and is defined as:

$$MSFE_h = \sum_{t=W-1}^{T} (Y_{t+1} - \hat{Y}_{t+1})^2, \tag{8}$$

where $\hat{Y}_{t+1}$ is the forecast of the dependent variable for time $t+1$, and W is the sample estimation period.

To test if structural breaks needs to be implemented in the factor and forecasting models, the GDP growth of Tanzania is fitted in a auto-regressive model with optimal lag order p_{bic}, for the five different forecast data sets in a recursive window. The residuals of this model are analyzed by means of kurtosis and skewness. If the skewness is smaller than one and if the kurtosis is around three, there is no need to include structural breaks in the models. It is also checked whether these residuals have a normal distribution by means of the Jarque-Bera test.

4.5 Simulation With Data-generating Process

To evaluate the effectiveness of the factor model that includes boosting and PCA in a factor augmented auto-regression (FAAR) framework, data is simulated using a data-generating process (DGP). First f_{ar} factors are made using the AR model with lag order one. The dependent and explanatory variables are created using those factors. The DGP will lead to estimates of the dependent variable y and to explanatory variables X. After the DGP, the dynamic factors $\hat{F}_t$ are estimated by principal component analysis (PCA). Subsequently, the boosting algorithm 1 is implemented. The algorithm of the DGP is given below as data-generating process algorithm 2, for $t = 1, \dots T$ observations.

When the data is generated with $f_{ar} = 3$, the correct amount of factors in the boosting algorithm 1 and in the 'pre-selection' needs to be equal to 3 (f_{ar}). The simulation is necessary to check if certain 'input values' play an important role in the simulation. These 'input values' include the step length v in the boosting algorithm 1, the maximum amount of lags (p_{max}) in (6), the maximum amount of factors (r_{max}) in (5). They also concern the amount of observations, the amount of variables and the

Algorithm 2 *data-generating process*

Output Dependent variable (y_{t+1}) and explanatory variables (X_t)

1. Construct f_{ar} factors with an AR model with lag order one:

$$F_{t,1} = \alpha + \beta_1 F_{t-1,1} + \epsilon_{t,1}$$
$$\vdots$$
$$F_{t,f_{ar}} = \alpha + \beta_r F_{t-1,f_{ar}} + \epsilon_{t,f_{ar}},$$

   where F_t are the constructed dynamic factors, α is a constant, β is a $f_{ar} \times 1$ vector with uniformly distributed random numbers and ϵ is the idiosyncratic component of F_t with variance equal to 1.

2. Construct N explanatory variables:

$$X_{t,1} = \sum_{i=1}^{f_{ar}} F_{t,i} \upsilon_{i,1} + w_{t,1}$$
$$\vdots$$
$$X_{t,N} = \sum_{i=1}^{f_{ar}} F_{t,i} \upsilon_{i,N} + w_{t,N},$$

   where X_t is the $T \times N$ explanatory variable, υ is a $f_{ar} \times N$ vector with normally distributed random numbers and w is $T \times N$ vector of normally distributed random numbers.

3. Construct the dependent variable:

$$y_t = \sum_{j=1}^{f_{ar}} F_{t,j} \lambda_j + \varepsilon_t,$$

   where y_t is the $T \times 1$ dependent variable, λ is a $f_{ar} \times 1$ vector with normally distributed random numbers and ε is $T \times 1$ vector of normally distributed random numbers.

amount of factors (f_{ar}) the data is generated with. This is done by analyzing if the 'pre-selection' of the amount of factors, and the amount of factors selected in the boosting method differs when changing the 'input values'. The examination is based on two hit rates: hit-rate$_{pr}$ and hit-rate$_{bo}$. The former represents the hit-rate when comparing the amount of factors selected in the 'pre-selection' with the amount of factors f_{ar}. The latter consists of the hit-rate when comparing the amount of factors selected in the boosting algorithm 1 with the amount of factors f_{ar}.

It is specifically important to test whether the optimal beta ($\hat{\beta}_{j*}$) in the boosting algorithm 1 is specified correctly. This is done by applying another DGP. The dynamic factors are set equal to the explanatory variables: $F_t = X_t$ in the boosting algorithm 1. The dependent variable y_t is

constructed as follows for $j = 1 \dots N$ variables, and $t = 1, \dots, T$ observations:

$$y_t = \sum_{j=1}^{N} \alpha_j X_{t,j} + \epsilon_t, \tag{9}$$

where X_t is a matrix of standardized normally distributed random numbers and ϵ_t is a $N \times 1$ vector with normally distributed random numbers. The $1 \times N$ vector α is chosen randomly but needs to remain constant when forecasting y_t multiple times. Because when the boosting algorithm 1 is constructed correctly, the optimal beta $\hat{\beta}^M = (\alpha_1, \dots, \alpha_N)'$. This can be verified by a one-sample t-test when implementing k times the boosting algorithm 1, for k sufficient large.

In the next section the results are presented, and the data of African countries is used to forecast the gross domestic product (GDP) growth of Tanzania. The data from the DGP is constructed into the same format as the African data set. Therefore the problems in the simulation apply to the forecasts of the GDP growth of Tanzania. Hence the valuation of the simulation is important for the accuracy of the implemented method for forecasting the GDP growth.

5 Results

This section consists of two parts. The first analyzes the results of the simulation. The second part discusses the forecast results of the gross domestic product (GDP) growth of Tanzania. The simulation is necessary for the second part because it test the correctness of the boosting method and principal component analysis implemented.

5.1 Results of the simulation

The data generating process (DGP) in algorithm 2 from the previous section 4.5 is applied for $T = 54$ observations, $N = 51$ explanatory variables and $alpha = 0.05$. The stopping parameter M has the value 100. The optimal amount of lags (p_{bic}), the value of $\hat{\beta}_w$, the $T \times (p_{max} + 1)$ matrix W_t containing additional explanatory variables and residual vector E are established as discussed in section 4.4. The optimal amount of 'pre-selected' factors are estimated as discussed in section 4.2.

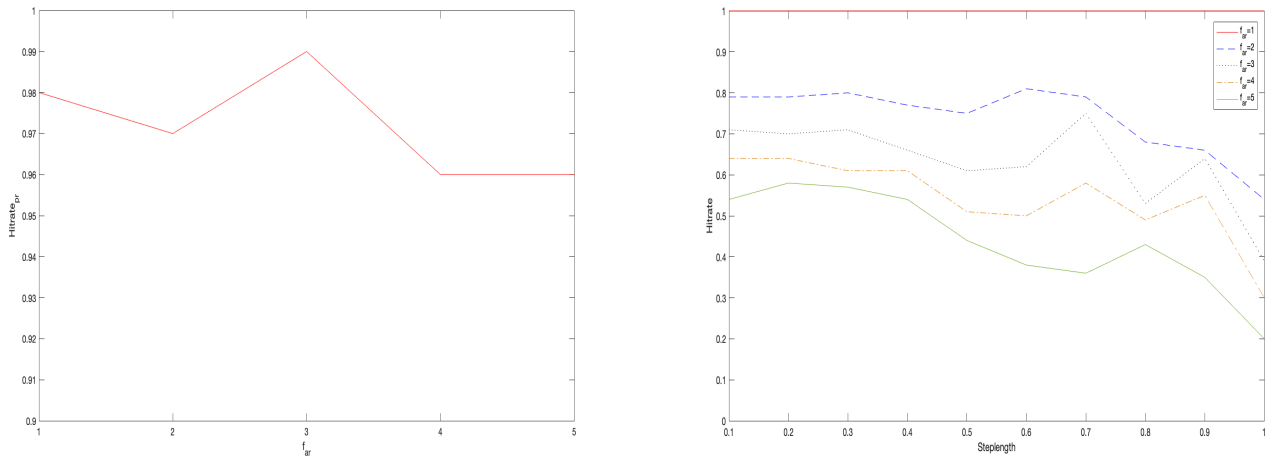
First, it was investigated if the amount of factors (f_{ar}) from which the data is generated, plays an important role in the estimation of the amount of 'pre-selected' factors. This analysis is based on the hit-rate of the 'pre-selected' amount of factors (hit-rate$_{pr}$). In figure 2(a), this hit-rate is

given for different values of f_{ar}. Note that the hit-rate does not show large variations. This means that different values of f_{ar} do not have a large impact on the hit-rate of the 'pre-selected' amount of factors. The hit-rate$_{pr}$ is around one. This indicates that the amount of factors in the 'pre-selection' is estimated correctly.

Secondly, the correct amount of factors of the boosting algorithm 1 is analyzed for using different values f_{ar} and step length v. This examination uses the hit-rate$_{bo}$, and is represented in figure 2(b) for different value of f_{ar} and v. This figure shows that hit-rates$_{bo}$ vary. When f_{ar} increases, the hit-rates$_{bo}$ decrease. When comparing these hit-rates$_{bo}$ with the hit-rates$_{pr}$ for different values of $f_{ar} \geq 1$ and step length v, the hit-rates$_{bo}$ are lower. This could be caused by the misspecification of the amount of factors in the boosting algorithm. The dependent variable y in (4), depends partly on the value of $F_t'\beta_F$ estimated in the boosting algorithm and partly on the value of $W_t\hat{\beta}_w$ estimated by the auto-regressive model. The data is generated from an auto-regressive model of lag order one. Therefore, a part of the estimations is already contained in $W_t\hat{\beta}_w$, while in the boosting algorithm $F_t'\beta_F$ is estimated. When increasing the amount of factors, fc, in the DGP, more information is included in $F_t'\beta_F$. As a result, less information needs to be added with $F_t'\beta_F$. Therefore, less factors in the boosting algorithm need to be selected. Hence, it can be true that the hit-rates$_{bo}$ are lower and decreases for increasing values of f_{ar} without having a misspecification of the amount of factors in the boosting algorithm.

In the following simulation $f_{ar} = 3$ and corresponding optimal step length $v = 0.2$. Figure 8 in the appendix section 7.2 represents the estimated amount of factors when the boosting algorithm 1 is implemented, for 100 iterations and different values of f_{ar}, with step length $v = 0.2$. In this figure it is shown that if the amount of factors is estimated incorrectly, the amount of estimated factors is too low. This is in line with the explanation of the lower hit-rate$_{bo}$, and the descending hit-rate$_{bo}$ for different values of f_{ar}. Thus, the amount of factors in the 'pre-selection' and the boosting algorithm can be correctly specified.

The amount of variables and observations could also affect the amount of 'pre-selected' factors and factors selected in the boosting algorithm. In figure 3(a) and 3(b) the hit-rates$_{pr}$ and hit-rates$_{bo}$ are represented for different values of observations and variables respectively. The hit-rates differ not substantially when the simulation is applied for a different amount of variables and observations. The African data set consists of $49-53$ observations and 51 explanatory variables. The same amount of variables and observations can be applied in the simulation.

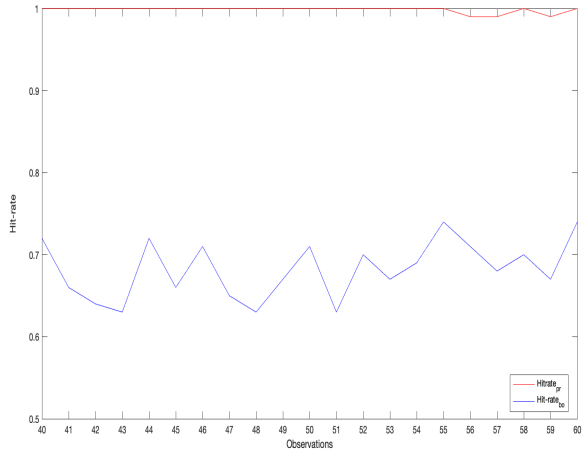


(a) Hit-rate$_{pr}$ (b) Hit-rate$_{bo}$

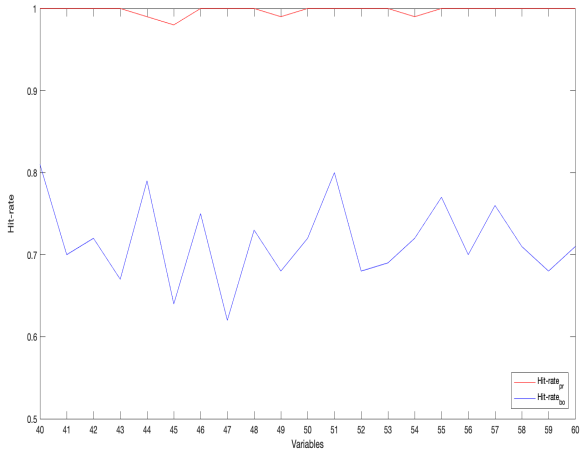
Figure 2: Hit-rate$_{pr}$ and Hit-rate$_{bo}$ for different values of f_{ar} and step length v

Afterwards, it was tested if the amount of maximum lags (p_{max}) in (6) influences the hit-rates$_{bo}$. Another important aspect is to analyze whether the amount of maximum factors (r_{max}) in (5) effects the hit-rates$_{pr}$. In figure 3(c) the hit-rates$_{bo}$ and hit-rates$_{pr}$ are represented for different values of p_{max} and r_{max} respectively. The amount of factors selected in the boosting algorithm 1 do not vary significantly, for different values of p_{max}. Therefore, the value of p_{max} has little effect on the hit-rates$_{bo}$. The optimal value of the maximum lags is 10, so $p_{max} = 10$ in the following simulation. The amount of r_{max} does effect the hit-rates$_{pr}$. The amount of r_{max} needs to be higher than f_{ar}, because otherwise the 'true' amount of factors can not be estimated. Furthermore, r_{max} needs to be smaller than 6 because otherwise the hit-rates$_{pr}$ will be lower. The factors in the 'pre-selection' is chosen 6 in the following simulation.

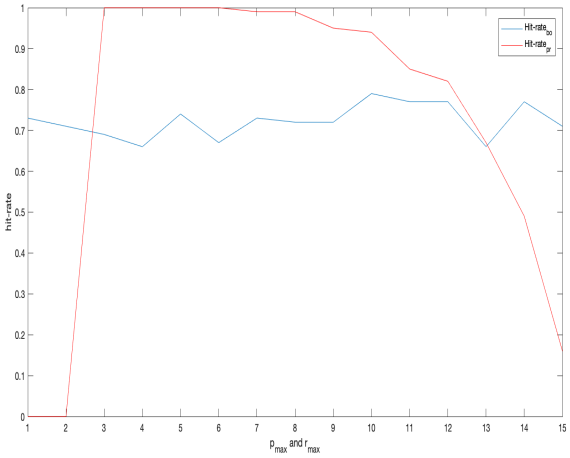
Lastly, the simulation tests if the optimal beta, $\hat{\beta}^M$ from algorithm 1 is chosen correctly. This is done by another DGP given in equation 9. The data is generated for $T = 200$ observations and $N = 4$ variables. The dependent variable in the same equation is predicted 100 times, by implementing the boosting algorithm. The vector $\alpha = (0.1, 0.2, 0.3, 0.4)$ in (9), so the 'true' values of the parameters $\hat{\beta}^M$ are (0.1, 0.2, 0.3, 0.4). The results of the estimate of the optimal beta when the dynamic factors are set equally to the explanatory variable X_t, are shown in figure 4. The blue lines are the estimated parameters, while the 'true' values of the parameter are given in red. As can be seen, the values of the optimal beta stays approximately constant. Thus, the hypothesis



(a) Different amount of observations



(b) Different amount of variables



(c) Different values of p_{max} and r_{max}

Figure 3: Hit-rates$_{bo}$ and Hit-rates$_{pr}$ or different values of p_{max}, r_{max} and for the data consisting of a different amount of observations and variables

is made that the parameter $\hat{\beta}^M$ is estimated correctly. To test this hypothesis a t-test is applied. The *p-value* of the estimated coefficients parameters are all larger than 0.05 as can be seen in figure 4. This leads to the conclusion that there is no significant evidence to reject the null hypothesis that $\hat{\beta}^M = (0.1, 0.2, 0.3, 0.4)$, at the 5% significance level. So the boosting algorithm 1 selects the optimal beta correctly.

In summary, the data generating method is implemented 100 times to estimate data consisting of 54 observations and 52 variables. The data is made of three factors, so $f_{ar} = 3$. The amount of

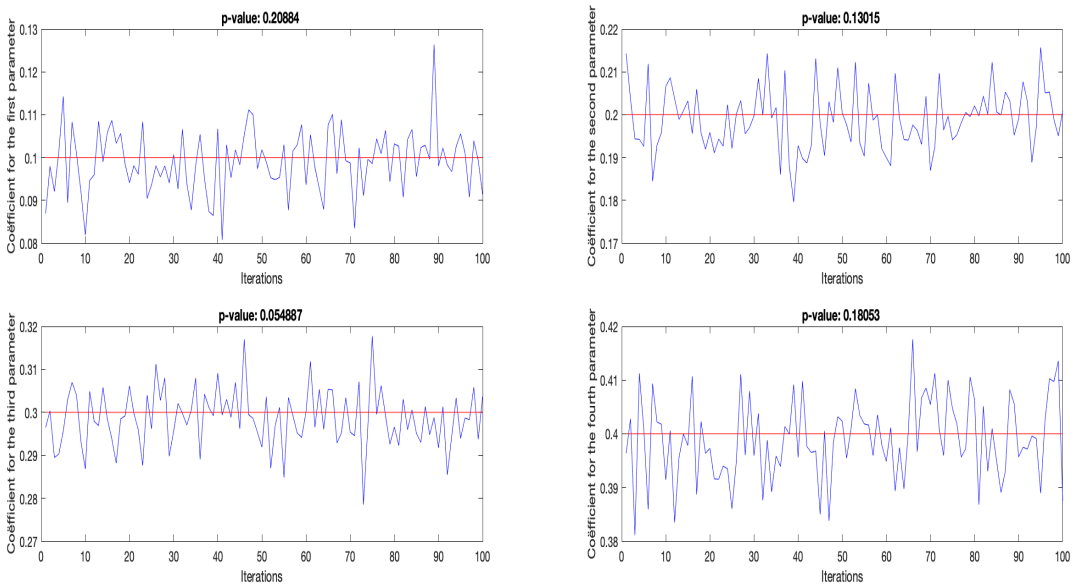


Figure 4: Test for the optimal beta in the boosting algorithm

'pre-selected' variables in (5) estimated with $r_max = 6$, the optimal amount of lags (p_{bic}) is estimated by minimizing the BIC value in (7) with $p_max = 10$. The boosting algorithm is implemented with stopping parameter $M = 100$ and step length $v = 0.2$. The hit-rate of the 'pre-selected' amount of factors with the 'input values' above is around 1 and the hit-rate of factors selected in the boosting algorithm is around 0.7. The hit-rate$_{bo}$ is lower than the hit-rate$_{pr}$, but as discussed before both the amount of factors in the 'pre-selection' and in the boosting algorithm 1 are still specified correctly. Furthermore it is concluded, based on the results of the t-test, that the optimal beta is specified correctly.

5.2 Results for the African Data

After analyzing the factor and forecasting models with the simulation, the models are used to make five forecasts of the gross domestic product (GDP) growth of Tanzania. The five forecasts are estimated using a recursive window. Prior to the forecast, it was tested whether it was necessary to implement structural breaks in the models. The kurtosis and skewness of the residuals of the five different forecasts are estimated for a data set corresponding to the forecast. To test if the GDP growth of Tanzania has a normal distribution a Jarque-Bera test was applied. The test and results are summarized below in table 1. The skewness is low and the kurtosis is nearby the value three,

and the Jarque-Bera test with 5% significance level concludes that the GDP growth of Tanzania has a normal distribution. Thus, there is no need to include structural breaks in the models.

Forecasts for year	Data-set	Kurtosis	Skewness	Jarque-Bera value (*p-value*)
2012	1963-2011	2.6281	-0.3479	1.4005 (0.3712)
2013	1963-2012	2.5860	-0.3183	1.2733 (0.4077)
2014	1963-2013	2.5452	-0.2873	1.1635 (0.4451)
2015	1963-2014	2.5139	-0.2563	1.0604 (0.4924)
2016	1963-2015	2.5913	-0.1779	0.6117 (0.500)

Table 1: Test for structural breaks and normal distribution of the GDP growth of Tanzania

The boosting algorithm 1 from section 4.3 is implemented, with step length $v = 0.2$, and stopping parameter $M = 100$. A recursive window is used for the forecasts and forecast 1 represents the forecast of year 2012, forecast 2 represents the forecast of year 2013, etc.. The auto-regressive model with optimal lag order p_{bic} is estimated by minimizing the Bayesian information criterion (BIC) in (7), with $p_{max} = 6$. This lag order is two in all five forecasts, so the benchmark model, concerns an auto-regressive model with lag order 2. The AR(2) model is also used to estimate the additional explanatory variable W_T, parameter $\hat{\beta}_w$, and the $T \times 1$ residual vector E.

The results of these forecasts are given in figure 5, together with the real GDP growth of Tanzania. It can be seen that the differences of the GDP growth between the benchmark forecasts and the real value of y are generally larger than when a combination of the boosting algorithm and PCA was used. The mean squared forecast error (MSFE) is calculated for five forecasts, with $T = 53$ and $W = 50$. The MSFE of the benchmark model is 7.7199 and is the MSFE of the forecast method that implement boosting and PCA is 7.4493. The forecasts that implement the factor and forecasting models, estimate the factors with PCA and implement the boosting algorithm has a lower MSFE than the AR(2) benchmark model, thus this model outperforms the benchmark model.

The amount of 'pre-selected' variables is estimated in (5) with $r_{max} = 6$. The first three factors are selected for all five forecasts during this 'pre-selection'. The boosting algorithm selects the first factor for all the five forecasts. In figure 6 below the factor loadings are represented for the African countries. The factors are estimated with PCA, so the factors are a linear combination of all variables and have all loadings. This leads to a difficult interpretation, but in this research there are only two countries with remarkably high factor loadings. This simplifies the interpretation substantially. Eq. Guinea (16) and Liberia (27) have a high factor loading. Therefore, these countries are used for the

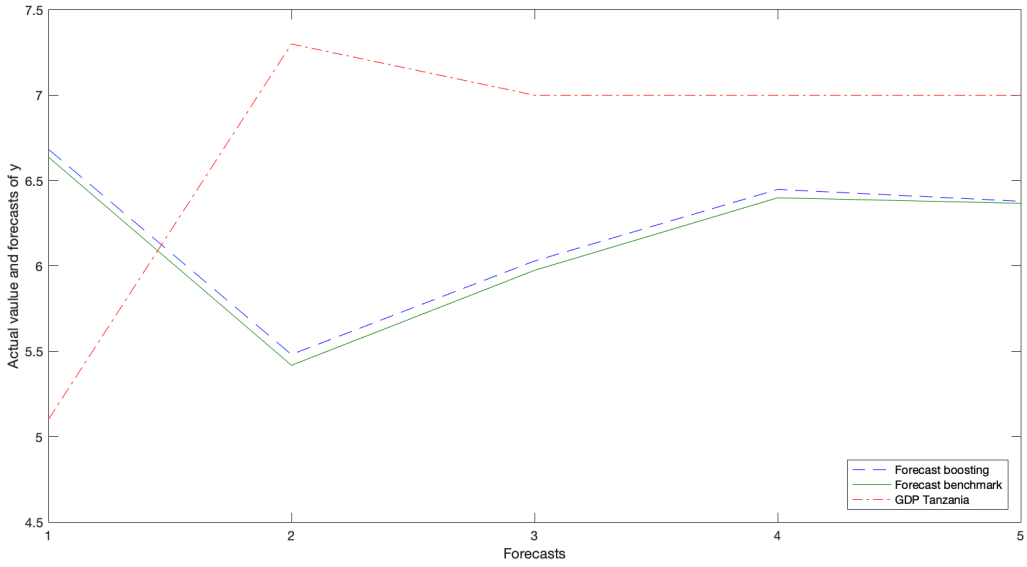


Figure 5: Forecasts of the model that implements the boosting algorithm 1 with PCA, the benchmark $AR(2)$ model and the GDP growth of Tanzania

forecast of the GDP growth of Tanzania in the boosting algorithm 1. In section 7.2 of the appendix figure 9 is given. This figure shows the map of Africa where Eq. Guinea, Liberia and Tanzania are coloured. As can be seen, the countries are located far away from Tanzania. Thus, there is not a clear explanation of the correlation between the GDP growth of these countries. The reason why these particular countries have such a high factor loading is left for future research.

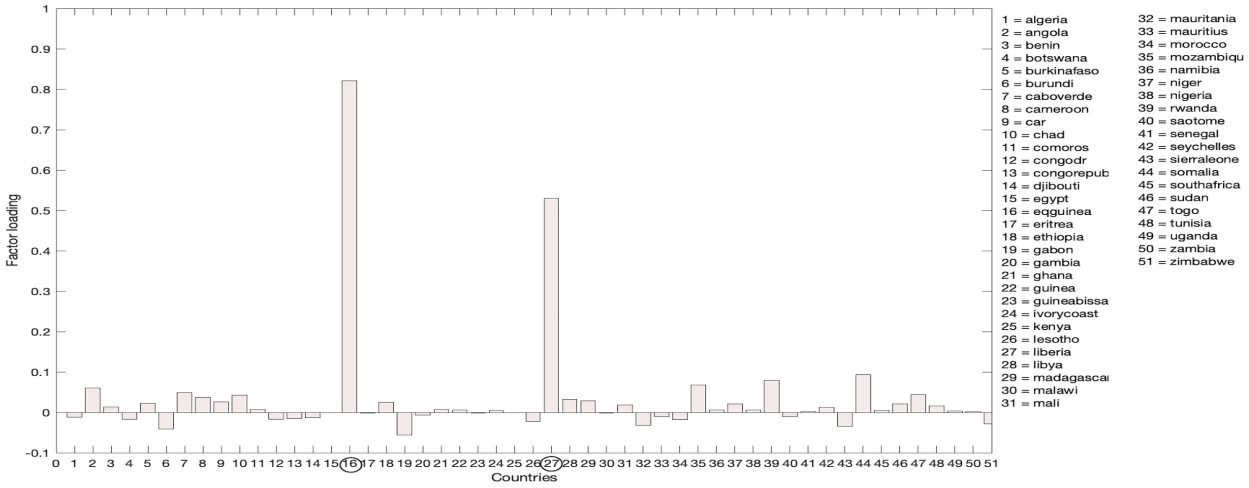


Figure 6: Factor loadings of the first factor

In conclusion, the factor and forecasting models that have factors estimated by PCA and implement the boosting algorithm 1 were analyzed. The GDP growth of Eq. Guinea and Liberia are used in these models to forecast the GDP growth of Tanzania in the implementation of the boosting algorithm. These models forecast the GDP growth of Tanzania better than the 'simple' $AR(2)$ model, as shown by the mean square forecast error.

6 Conclusion

Following in the footsteps of Kim and Swanson (2018), the factor and forecasting models in specification type one are used to make five forecasts of the gross domestic product (GDP) growth of Tanzania. The factors are first constructed by PCA with the data set, subsequently the boosting algorithm is implemented in a factor augmented auto-regressive (FAAR) framework to select functions of and weights for the factors. Lastly, these functions and weights are used in the prediction method to make a forecast of the GDP growth of Tanzania. This estimation is based on 51 GDP growths of other African countries, included in the dynamic factors. The lagged GDP growth of Tanzania is also used, and is included in the part estimated by the auto-regressive model. The forecasts are compared by means of the mean square forecast error and a benchmark model is considered including the auto-regressive AR model with two lags. To evaluate the effectiveness of these factor models that includes boosting and PCA in a FAAR framework, data was simulated using a data-generating process (DGP). The data was constructed with a AR model with lag order one. The simulation examines if the 'pre-selected' amount of factors and the amount of factors in the boosting algorithm 1 in section 4.3 were specified correctly. In the end, the optimal beta in the boosting algorithm 1 is also analyzed.

The results of the simulation is as follows, the amount of factors f_{ar} the data is made of does not have a large impact on the hit-rate of the 'pre-selected' amount of factors (hit-rate$_{pr}$). Contrarily, it does have an impact on the hit-rate of the factors selected in the boosting algorithm 1 (hit-rate$_{bo}$). The value of the step length (v) also influences these hit-rates$_{bo}$. In the end, the step length is chosen to be 0.2 and $f_{ar} = 3$. Furthermore, both hit-rates hit-rate$_{pr}$ and hit-rate$_{bo}$ differ not a lot when the simulation is applied for a different amount of variables and observations. So the amount of variables and observation can be chosen similar to the African data set. The optimal value of the maximum lags in (6) and (7) is 10, so $p_{max} = 10$. The amount of r_max in (5) does effect the

hit-rates$_{pr}$, and is chosen to be 6. Lastly, the boosting algorithm 1 selects the optimal beta ($\hat{\beta}_w$) correctly. The simulation returns a hit-rate$_{pr}$ of almost one and the hit-rate$_{bo}$ has a value around 0.7. The factors selected in the boosting algorithm 1 have a lower hit-rate but are still be correctly specified because of reasons discussed in the previous section. Thus, this forecast method proves to be accurate.

After the simulation, the GDP growth of Tanzania is forecast for the year 2012 up to 2016. The forecasts concern the factor and forecasting models explained in section 4. These forecasts outperforms the GDP growth forecast by the 'simple' AR(2) process. This is analyzed by the mean square forecast error. The GDP growths of Eq. Guinea and Liberia are used for forecasting the GDP growth of Tanzania for all five forecasts. In the end, factor models and the machine learning technique boosting are helpful by modeling and forecasting the low frequency macro economic variable gross domestic product in Tanzania.

For future research there are multiple areas that seem of interest. The first area has to do with the estimation of the factors. The factors are estimated with PCA, the disadvantage of this method is that the factors are linear combination of all variables, so they all have nonzero loadings. This leads to a more difficult interpretation, if there are no remarkably high factor loadings. Another disadvantage of PCA is that the most relevant components for predicting the target variable are the one with the 'largest' variance, but there is no particular reason to assume this. A solution to these disadvantages is Sparse principal component analysis (SPSE) as discussed in Zou et al. (2006). This method allows for 'sparsity', and can be interesting for future research. Another potential method is independent component analysis (Kim and Swanson (2018)). The second area concerns the machine learning variable, and shrinkage methods. As discussed in Kim and Swanson (2018) there are also other methods for example bagging, ridge regression the least angle regression, the 'eleastic net' and the 'non-negative garotte' or the Bayesian model averaging. These methods might result in forecasts with a lower MSFE and are therefore interesting to analyze.

References

Bai, J. and Ng, S. (2002). Determining the number of factors in approximate factor models. 70(1):191–221.

Bai, J. and Ng, S. (2008). Forecasting economic time series using targeted predictors. *Journal of Econometrics*, 146(2):304–317.

Bai, J. and Ng, S. (2009). Boosting diffusion indices. *Journal of Applied Econometrics*, 24(4):607–629.

Bańbura, M. and Modugno, M. (2014). Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data. *Journal of Applied Econometrics*, 29(1):133–160.

Biau, O. (2013). Euro area gdp forecasting using large survey datasets a random forest approach.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Carriero, A., Clark, T. E., and Marcellino, M. (2015). Realtime nowcasting with a bayesian mixed frequency model with stochastic volatility. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 178(4):837–862.

Christophe Hurlin (2013). Determining the Number of Factors in Approximate Factor Models. `http://www.runmycode.org/companion/view/69`. Accessed on 21/06/2019.

Dufour, J.-M. and Stevanovic, D. (2010). Factor-augmented varma models: Identification, estimation, forecasting and impulse responses. *Manuscript. Université du Québeca Montréal*.

Fan, J. and Yao, Q. (2017). *The elements of financial econometrics*. Cambridge University Press.

Forni, M., Hallin, M., Lippi, M., and Reichlin, L. (2000). The generalized dynamic-factor model: Identification and estimation. *Review of Economics and statistics*, 82(4):540–554.

Franses, P. H. and Vasilev, S. (2019). Realgdp growth in africa, 1963-2016. *Unpublished*.

Hassani, H. and Silva, E. S. (2015). Forecasting with big data: A review. *Annals of Data Science*, 2(1):5–19.

Hastie, T., Rosset, S., Zhu, J., and Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Heij, C., de Boer, P., Franses, P. H., Kloek, T., van Dijk, H. K., et al. (2004). *Econometric methods with applications in business and economics*. Oxford University Press.

Jolliffe, I. (2011). *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kim, H. H. and Swanson, N. R. (2013). Mining big data using parsimonious factor and shrinkage methods.

Kim, H. H. and Swanson, N. R. (2014). Forecasting financial and macroeconomic variables using data reduction methods: New empirical evidence. *Journal of Econometrics*, 178:352–367.

Kim, H. H. and Swanson, N. R. (2018). Mining big data using parsimonious factor, machine learning, variable selection and shrinkage methods. *International Journal of Forecasting*, 34(2):339–354.

Plakandaras, V., Papadimitriou, T., and Gogas, P. (2015). Forecasting daily and monthly exchange rates with machine learning techniques. *Journal of Forecasting*, 34(7):560–573.

Schumacher, C. (2007). Forecasting german gdp using alternative factor models based on large datasets. *Journal of Forecasting*, 26(4):271–302.

Stock, J. H. and Watson, M. W. (2002a). Forecasting using principal components from a large number of predictors. *Journal of the American statistical association*, 97(460):1167–1179.

Stock, J. H. and Watson, M. W. (2002b). Macroeconomic forecasting using diffusion indexes. *Journal of Business & Economic Statistics*, 20(2):147–162.

Stock, J. H. and Watson, M. W. (2006). Forecasting with many predictors. *Handbook of economic forecasting*, 1:515–554.

Stock, J. H. and Watson, M. W. (2012). Generalized shrinkage methods for forecasting using many predictors. *Journal of Business & Economic Statistics*, 30(4):481–493.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286.

7 Appendix

7.1 Nomenclature

Nomenclature

DGP Data generating process

$FAAR$ Factor augmented auto-regressive

GDP Gross domestic product

$MSFE$ Mean squared forecast error

PCA Principal component analysis

Hit-rate$_{bo}$ Amount of factors selected in the boosting algorithm 1 in section 4.3

Hit-rate$_{pr}$ Amount of factors selected in the 'pre-selection' in (5)

7.2 Results

Figure 7 represents the standard deviation and mean of dependent variable y and explanatory variable X. Figure 8 represents the estimated amount of factors when the boosting algorithm 1 is implemented, for 100 iterations and different values of f_{ar}, with step length $v = 0.2$.Figure 9 shows the map of Africa where Eq. Guinea, Liberia and Tanzania are coloured.

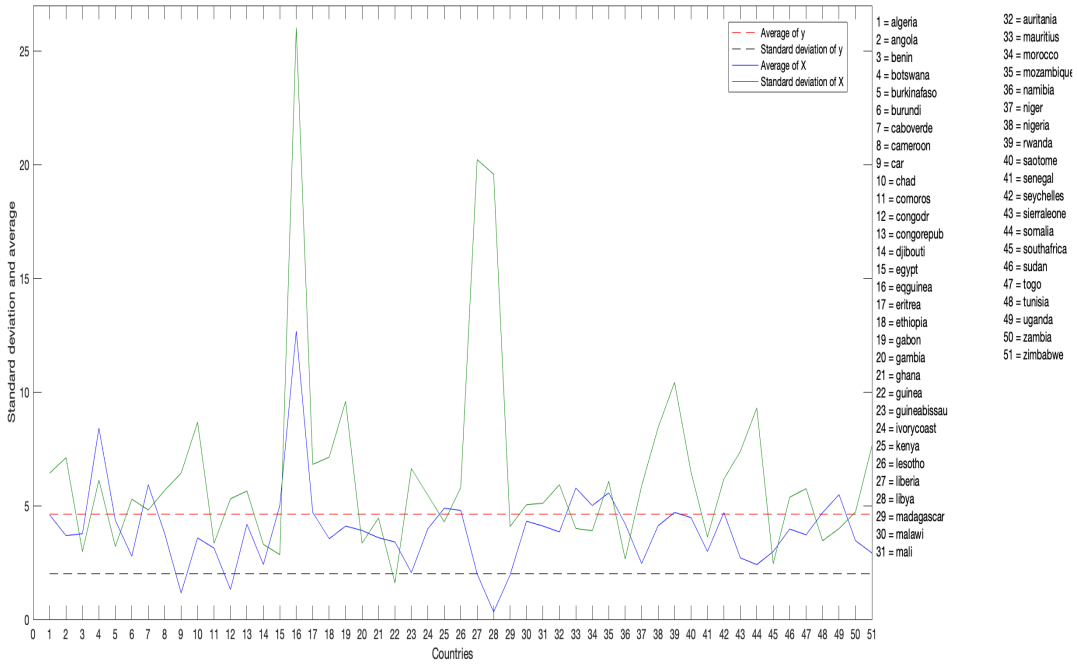


Figure 7: Standard deviation and mean of X and y

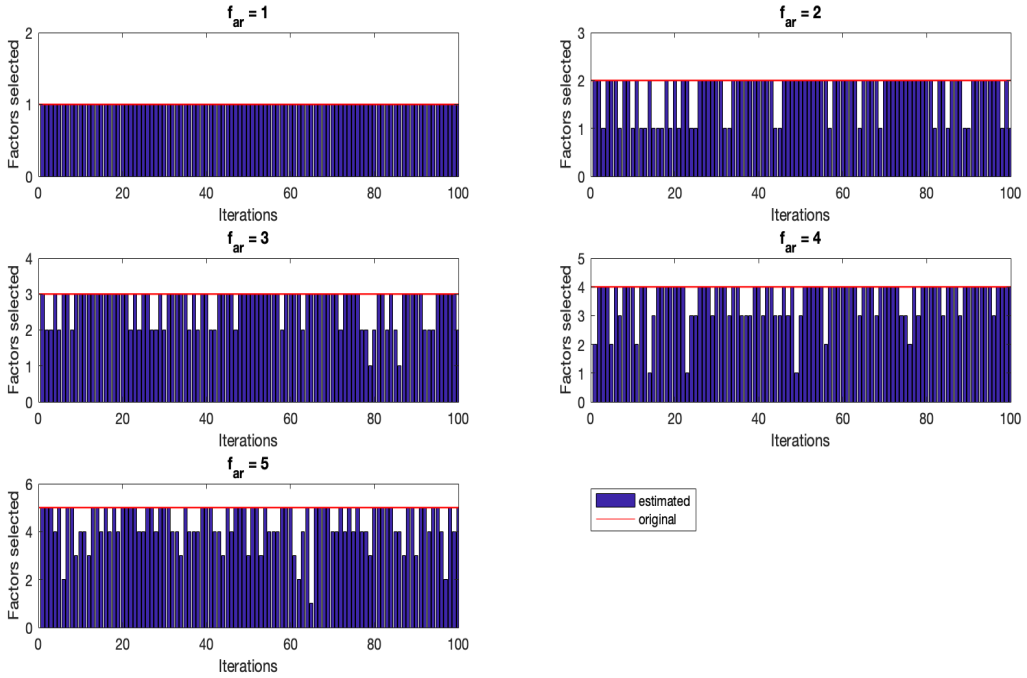


Figure 8: Selected factors for step length 0.2 and different values of f_{ar}

Figure 9: Map of Africa

7.3 Codes

7.3.1 Simulation

```
MATLAB read-me file for the Simulation

make_X_y_F.m
->  By running this program, you can get the explanatory variables, dependent variables and the dynamic factors which are
    obtained by the data generating process represented algorithm 2 of section 4.5

armodel_simulation.m
->  This function implements the auto-regressive (AR) model with optimal lag order p_bic. The estimates of the additional
    explanatory variable W_t and parameter
beta^_w, and the T x 1residual vector E in the boosting algorithm 1 of section 4.3 are estimated.

NbFactorsSimulation.m
-> This function gives you the result of the 'pre-selected' amount of factors. The code is implemented from Christophe
    Hurlin (2013)

l2boost_simulation.m
-> This function gives you the result of boosting algorithm 1 in section 4.3

Simulation_t.m
-> This function gives you the estimated hitrates_bo and hitrates_pr for different amount of observations. The results are
    given in figure 4(a) of section 5.1

Simulation_n.m
-> This function gives you the estimated hitrates_bo and hitrates_pr for different amount of variables. The results are
    given in figure 4(b) of section 5.1

simulation_factor.m
-> This function gives you the estimated hitrates_nb for different amount of factors r_ar in the DGP. The results are
    given in figure 3(a) of section 5.1

simulation_v.m
-> This function gives you the estimated hitrates_bo  for different amount of step length v and f_ar in the DGP. The
    results are given in figure 3(b) of section 5.1

Simulation_pmax.m
-> This function gives you the estimated hitrates_bo for different amount of maximum lags. The results are given in figure
    4(c) of section 5.1

simulation_rmax.m
-> This function gives you the estimated hitrates_pr for different amount of factors r_max. The results are given in
    figure 4(c) of section 5.1

test.m
-> This function gives you the tests results if the optimal beta from the boosting algorithm 1 in section 4.3 is chosen
    correctly. The results are given in figure 5 of section 5.1

simulation_result.m
-> This function gives the estimated hitrats_bo and  hitrate_pr with specific 'input values'
```

Figure 10: Read me file for the simulation

```
function[F_result, X_result, y_result] = make_X_y_F(n,T,r)
%Construct the data, from a data generating proces
par = rand(r,1);
F_result = [];
for i = 1:r %Construct r factors from AR model
    model = arima('Constant',0.05,'AR',{par(i,1)},'Variance',1);
    F_result = [F_result, simulate(model,T)];
end
lin_combinations = randn(r,n);
x = F_result*lin_combinations + randn(T,n);  %Construct dependent and explanatory variables
X= zscore(x);
y_result= X(:,1);
X_result= X(:,2:end);
end
```

Figure 11: Construct dependent variable y, explanatory variables X, and factors F

```
function [res_ar, lags, Wt_Bw, fcast]= armodel_simulation(y, p_max)
% Estimate the residuals, lags, the coefficient Wt_Bw and the value of the
% forecast with a AR model with optimal lag order. This lag order is
% selected by BIC with maximum amount of lags p_max
T=size(y, 1);
optimal_bic=0;
Wt_lags=[];
for i=1:p_max %Estimate the optimal amount of lags with the bic value
    model = ar(y,i);
    bic=model.Report.Fit.BIC;
    if (i==1)  || (bic<optimal_bic)
        optimal_bic=bic;
        lags=i;
    end
end

Mdl = arima(lags,0,0); % estimate ar model with optimal amount of lags
[ar_model,~,~,info] = estimate(Mdl,y, 'Display', 'off');
fcast = forecast(ar_model,1,'Y0',y);

[res_ar] = infer(ar_model,y); % AR(lags) residuals
information= info.X;
coef_ar=[];
for i=1:lags+1 %estimate coefficients of the AR(lags)
    [coef_ar]= [coef_ar; information(i)];
end
for i=1:lags %estimate the Wt value in the boosting algrithm
    Wt_lags=[Wt_lags, y(T-i+1)];
end
Wt= [1, Wt_lags]; %estimate the Wt value in the boosting algorithm
Wt_Bw= Wt*coef_ar; %estimate the WtBw value in the boosting algirhtm
end
```

Figure 12: Construct W_t, E, $\hat{\beta}_W$ from equation 2 by the auto-regressive model

```
function [khat]=NbFactorsSimulation(X,r_max)
if nargin==1, r_max=min(size(X)); end                          % Rule proposed by Bai and Ng to choose kmax
if isnan(r_max)==1, r_max=min(size(X)); end                    % Rule proposed by Bai and Ng to choose kmax
[T,N]=size(X)   ;                                              % Sample Sizes
if T<N                                                         % Choice of normalization according the computional cost
    [vectors,values] = eig(X*X');                              % Eigenvalues and eigenvectors of XX'
    factors=sqrt(T)*vectors(:,T-r_max+1:T);                    % Estimated Factors with kmax Factors
    loadings=X'*factors/T;                                     % Estimated Matrix of Factor Loadings
    betahat=loadings*chol(loadings'*loadings/N);               % Rescaled Estimator of the Factor Loading
else                                                           % Case T>N
    [vectors,values] = eig(X'*X);                              % Eigenvalues and eigenvectors of X'X
    loadings=sqrt(N)*vectors(:,N-r_max+1:N);                   % Estimated Matrix of Factor Loadings with kmax Factors
                                                ]
    betahat=(X'*X)*loadings/(N*T);                             % Rescaled Estimator of the Factor Loading
end
Z=X-X*betahat*inv(betahat'*betahat)*betahat';                  % Estimated Residuals
var_Z_kmax = sum(sum(Z.^2))/(N*T);                             % Estimated Variance of Residuals with kmax factors
V=zeros(r_max,1);                                              % Vector of V(k,Fk) for k=1,..,kmax
for k=1:r_max                                                  % Loop on the number of factor k
    if T<N                                                     % Choice of normalization according the computional cost
        [vectors,values] = eig(X*X');                          % Eigenvalues and eigenvectors of XX'
        factors=sqrt(T)*vectors(:,T-k+1:T);                    % Estimated Factors with kmax Factors
        loadings=X'*factors/T;                                 % Estimated Matrix of Factor Loadings
        betahat=loadings*chol(loadings'*loadings/N);           % Rescaled Estimator of the Factor Loading
    else                                                       % Case T>N
        [vectors,values] = eig(X'*X);                          % Eigenvalues and eigenvectors of X'X
        loadings=sqrt(N)*vectors(:,N-k+1:N);                   % Estimated Matrix of Factor Loadings with kmax Factors
        betahat=(X'*X)*loadings/(N*T);                         % Rescaled Estimator of the Factor Loading
    end
    Z=X-X*betahat*inv(betahat'*betahat)*betahat';              % Estimated Residuals
    V(k) = sum(sum(Z.^2))/(N*T);                               % V(k,Fk)
end
BIC3=repmat((0:1:r_max)',1,2);                                 % BIC information Criteria (BIC1, BIC2 and BIC3)
BIC3(1,2)=mean(sum(X.*X/T));                                   % PC information Criteria when r=0
CNT=min(N,T);                                                  % Function Cnt^2
Penalty=[(N+T)/(N*T)*log((N*T)/(N+T)) ...                      % Penalty Terms for IC and PC
    (N+T)/(N*T)*log(CNT) log(CNT)/CNT ];
Penalty=repmat(Penalty,r_max,1);                               % Penalty Terms for IC and PC
kk=repmat((1:1:r_max)',1,3);                                   % Matrix with increments
BIC3(2:end,2)=V+kk(:,1)*var_Z_kmax.*(N+T-kk(:,1))*log(N*T)/(N*T); % BIC3 criterium
[BIC3s,khat_BIC3] = min(BIC3(:,2));khat_BIC3=khat_BIC3-1;      % Estimated Numbers of Factor with BIC3
khat = khat_BIC3;                                              % Estimated Numbers of Factor with BIC3 criteria
                                                % BIC3 Information criterium for k=1,..,kmax
```

Figure 13: Select 'pre-selected' amount of factors, code from Christophe Hurlin (2013)

```
function [y_new, MSFE, BetaOpt,  F, mu_result] = l2boost_simulation(X, y, M_max, v, r, y_actual, Wt_Bw, E)
% Implemention of the boosting algorihtm
%input:
% X          explanatory variable
% y          dependent variable
% M_max      maximum number of boosting iterations
% v          steplength
% r          pre-selected amount of factors
% y_actual   real value of the GDP of Tanzania
% Wt_Bw      Estimation of WtBw in boosting algorithm
% E          E are the residuals of the ar model with optimal lag order
% lags       number of optimal lags
% outcome:
% y_new      forecasted GDP of Tanzania
% MSFE       Mean squared forecast error of the boosting algorihtm
% BetaOpt    Optimal beta estimated in the boosting alogirthm
% F          dynmically factors
% mu_result mu^M(Ft) in the boosting algorithm

T                = size(y,1);
ICm              = zeros(M_max,1);
MuHat            = zeros(T,M_max);
MuHat(:,1)       = mean(E, 'omitnan');
standardizedx    = zscore(X);
%F                = X; When applying the test for the optimal beta
F                = standardizedx*pca(standardizedx, 'VariableWeights', 'variance');
beta             = zeros(r, M_max);
Bm(:,:,1)        = ones(T,T,1)'*ones(T,T,1)'/(T);
SSR_min          = zeros(M_max,1);
for i = 1:M_max %boosting iterations
    u            = E - MuHat(:,i);
    SSR          = zeros(r,1);
    for j = 1:r
        [~,~,dhat] = mvregress(F(:,j), u);
        SSR(j)     = sum(dhat.^2);
    end
    [minSSR,jstar]  = min(SSR);
    bhat             = mvregress(F(:,jstar), u);
    gstarHat         = bhat'*F(:,jstar);
    MuHat(:,i+1)     = MuHat(:,i) + v .* gstarHat;

    %Updating Beta estimates using shrinkage parameters
    one_regressor           = zeros(r,1);
    one_regressor(jstar,1)  = 1;
    beta(:, i+1)            = beta(:,i) + v*bhat*one_regressor;

    % Updating degrees of freedom
    Pm          = F(:,jstar)*(F(:,jstar)'*F(:,jstar))^(-1)*F(:,jstar)';
    Bm(:,:,i+1) = Bm(:,:,i) + v*Pm*(eye(T) - Bm(:,:,i));
    dfm         = trace(Bm(:,:,i+1));
    % Estimating variance of boosting residuals
    sqrt_sigma_m_hat = (y-MuHat(:,i+1))' * (y-MuHat(:,i+1));
    % Computing information criteria
    ICm(i,1)     = log(sqrt_sigma_m_hat) + (log(T) * dfm) /T;
    SSR_min(i,1) = minSSR;
end
[~, Mopt]    = min(ICm);             % estimating the optimal amount of boosting iterations
BetaOpt      = beta(:,Mopt+1);       % selecting optimal beta
muopt        = MuHat(:,Mopt+1);      % selecting opitmal mu(F) for the forecast
y_new        = Wt_Bw + muopt(T);     % forecast
MSFE         = (y_actual-y_new)^2;   % calculate MSFE;
mu_result    =muopt(T);
end
```


Figure 14: Boosting algorithm

```
function []= simulation_t()
% Estimate hitrates_nb and hitrates_bo for different values of observations
M_max       =100; %amountboostings
p_max       =10; %maximum amount of lags
r_max       =6; %maximum amount of factors
T           =60; %amount of observations
n           =52; %amount of variables
iterations  =100; %amount of iterations
v           = 0.2; %step length
factor      = 3; %value of f_ar
hitrate_b=zeros(T-40,1);
hitrate_nb=zeros(T-40,1);
place=1;
for t=40:T %for different amount of observations
    hitrate_amount_nb =0;
    hitrate_amount_b=0;
    for i=1:iterations
    [~, X, y]                   = make_X_y_F(n,t,factor);
    r                           = NbFactorsSimulation(X,r_max);
    [E, ~, Wt_Bw]          = armodel_simulation(y, p_max);
    [~, ~, BetaOpt]             = l2boost_simulation(X, y, M_max, v, r, y(end), Wt_Bw, E);
    if sum(BetaOpt~=0)==factor
         hitrate_amount_b = hitrate_amount_b+1;
     end
    if r==factor
        hitrate_amount_nb          = hitrate_amount_nb+1;
    end
    end

    hitrate_nb(place)    = hitrate_amount_nb/iterations %estimte hitrate_nb
    hitrate_b(place)    = hitrate_amount_b/iterations %estimate hitrate_bo
    place=place+1
end
xplot=40:60;
plot(xplot, hitrate_nb)
hold on
plot(xplot, hitrate_b)
legend(["hit-rate_{nb}";"hit-rate_{bo}"])
end
```

Figure 15: Estimated hit-rates$_{bo}$ and hit-rates$_{pr}$ for different amount of observations

```
function []= simulation_n()
%estimate the hitrates_bo and hitrates_pr for different amount of variables
M_max       = 100; %amountboostings
p_max       = 10; %maximum amount of lags
r_max       = 6; %maximum amount of factors
t           = 54; %amount of observations
iterations  = 100; %amount of iterations
v           = 0.2; %step length
factor      = 3; %amount of f_ar
N           = 61;%amount of maximum variables
hitrate_b=zeros(N-41,1);
hitrate_nb=zeros(N-41,1);
place=1;
for n=41:N
    hitrate_amount_nb =0;
    hitrate_amount_b=0;
    for i=1:iterations
        [~, X, y]                   = make_X_y_F(n,t,factor);
        r                           = NbFactorsSimulation(X,r_max);
        [E, ~, Wt_Bw]          = armodel_simulation(y, p_max);
        [~, ~, BetaOpt]             = l2boost_simulation(X, y, M_max, v, r, y(end), Wt_Bw, E);
        if sum(BetaOpt~=0)==factor
            hitrate_amount_b = hitrate_amount_b+1;
        end
        if r==factor
            hitrate_amount_nb          = hitrate_amount_nb+1;
        end
    end
    hitrate_nb(place)    = hitrate_amount_nb/iterations; %estimate hitrate_nb
    hitrate_b(place)    = hitrate_amount_b/iterations; %estimate hitrate_bo
    place=place+1;
end
xplot=40:60;
plot(xplot, hitrate_nb)
hold on
plot(xplot, hitrate_b)
end
```

Figure 16: Estimated hit-rates$_{bo}$ and hit-rates$_{pr}$ for different amount of variables

```
function []= simulation_factor()
%estimate the hitrates_pr for different amount of f_ar
factor      =3; %amount factors data is made of
T           =54; %amount of observations
n           =52; %amount of variables
iterations  =100; %amount of iterations
hitrate_nb=zeros(10,1);
r_max       = 6; %maximum amount of factors

for factor=1:5
hitrate_amount_nb=0;
    for i=1:iterations
       [~, X]                   = make_X_y_F(n,T,factor);
        r                           = NbFactorsSimulation(X,r_max);
        if r==factor
            hitrate_amount_nb          = hitrate_amount_nb+1;
        end
    end
    hitrate_nb(r_max,1)    = hitrate_amount_nb/iterations ;
    r_max
end
f1      = figure;

figure(f1)
xplot=1:15;
plot(xplot, hitrate_nb);
title("Hitrate for 'preselected' amount of factors")
end
```

Figure 17: Estimated hit-rate$_{bo}$, with different amount of f_{ar} factors in the DGP

```
function [factors_estimated_b, hitrate_b]= simulation_v()
%Estimate hitrate_bo for different values of f_ar and v
M_max       =100; %amountboostings
p_max       =10; %maximum amount of lags
r_max       =6; %maximum amount of factors
T           =54; %amount of observations
n           =52; %amount of variables
iterations  =100; %amount of iterations
%v           = 0.2;
hitrate_b =zeros(10,1);
factors_estimated_b   =[];
%number=1;
for factor=1:5 %estimate for differnt values of f_ar
    number=1;
    for v=0.1:0.1:1 %estimate for differnt values of step length v
        hitrate_amount_b=0;
        for i=1:iterations
            [~, X, y]                   = make_X_y_F(n,T,factor);
            r                           = NbFactorsSimulation(X,r_max);
            [res_ar, ~, Wt_Bw]          = armodel_simulation(y, p_max);
            [~, ~, BetaOpt]             = l2boost_simulation(X, y, M_max, v, r, y(T), Wt_Bw, res_ar);
            if sum(BetaOpt~=0)==factor
                hitrate_amount_b = hitrate_amount_b+1;
            end
            factors_estimated_b(i,number)= sum(BetaOpt~=0);
        end
        hitrate_b(number,factor)    = hitrate_amount_b/iterations ;
        number               =number+1;
    end
end
f1=figure
figure(f1)
xplot=0.1:0.1:1
plot(xplot, hitrate_b);
title("Hitrate for amount of factors selected by boosting combined with PCA");
factors_estimated_b
%{
number2b=1;
for k=1:5
    subplot(3,2,k);
    xplot=1:1:iterations;
    bar(xplot, factors_estimated_b(:,k));
    hold on
    plot(0:iterations+1, k*ones(iterations+2), 'r');
    title("f_{ar} ="+ k);
    legend(["estimated";"original"])  ;
    number2b=number2b+1;
end
%}
```

Figure 18: Estimated hit-rates$_b o$ for different amount of f_{ar} and steplength v

```
function []= simulation_pmax()
%estimate the hitrates_bo for different amount of maximum lags
M_max       =100; %amount of boostings
factor      =3; %amount of f_ar
r_max       =6; %maximum amount of factors
T           =54; %amount of observations
n           =52; %amount of variables
iterations  =100; %amount of iterations
v           =0.2; %steplength
hitrate_b =zeros(10,1);
factors_estimated_b   =[];

for p_max=1:15 %for different amount of maximum lags
    hitrate_amount_b=0;
    for i=1:iterations
        [~, X, y]                   = make_X_y_F(n,T,factor);
        r                           = NbFactorsSimulation(X,r_max);
        [res_ar, ~, Wt_Bw]          = armodel_simulation(y, p_max);
        [~, ~, BetaOpt]             = l2boost_simulation(X, y, M_max, v, r, y(T), Wt_Bw, res_ar);
        if sum(BetaOpt~=0)==factor
            hitrate_amount_b = hitrate_amount_b+1;
        end
        factors_estimated_b(i,p_max)= sum(BetaOpt~=0);
    end
    hitrate_b(p_max)    = hitrate_amount_b/iterations ;
end
f1      = figure;
figure(f1)
xplot=1:15;
plot(xplot, hitrate_b);
end
```

Figure 19: Estimated hit-rates$_{bo}$ for different amount of maximum lags

```
function []= simulation_rmax()
%estimate the hitrates_pr for different amount of factors r_max
factor      =3; %amount factors data is made of
T           =54; %amount of observations
n           =52; %amount of variables
iterations  =100; %amount of iterations
hitrate_nb=zeros(10,1);

for r_max=1:15
hitrate_amount_nb=0;
    for i=1:iterations
       [~, X]                   = make_X_y_F(n,T,factor);
        r                           = NbFactorsSimulation(X,r_max);
        if r==factor
            hitrate_amount_nb          = hitrate_amount_nb+1;
        end
    end
    hitrate_nb(r_max,1)    = hitrate_amount_nb/iterations ;
    r_max
end
f1      = figure;

figure(f1)
xplot=1:15;
plot(xplot, hitrate_nb);
title("Hitrate for 'preselected' amount of factors")
end
```

Figure 20: Estimated hit-rates$_{bo}$ for different amount of maximum factors

```
function [OptimalBeta, results, p, stats] = test()
T=200;
n=4;
beta_wanted=[0.1 0.2 0.3 0.4];
r_max = 4; % maximum amount of factors when selecting 'pre-selected' factors
m_max=100; % maximum amount of boostings
p_max=10; % maximum amount of lags when estimating the optimal lags
v=0.2; %steplength
simulations=100; %amount of simulations
results=zeros(n,1);
p=zeros(n,1);
OptimalBeta=[];
result=[];

for i=1:simulations
    xtest= zscore(randn(T,n));
    ytest= 0.1*xtest(:,1)+ 0.2* xtest(:,2) + 0.3*xtest(:,3) + 0.4*xtest(:,4) + 0.1*randn(T,1);
    %now you want that your beta is equal to (0.1, 0.2, 0.3, 0)
    r= NbFactorsSimulation(xtest,r_max);
    [E, ~, Wt_Bw] = armodel_simulation(ytest, p_max);
    [~,~,OptimalBeta(i,:)] = l2boost_simulation(xtest, ytest, m_max, v, r, ytest(T), Wt_Bw, E);
    i
end
for i=1:n
[results(i,1), p(i,1),~, stats_between] = ttest(OptimalBeta(:,i), beta_wanted(i));
stats(i,1)= stats_between.tstat;
end

for j=1:n
    subplot(2,2,j)
    xplot=1:simulations;
    yplot= OptimalBeta(:,j);
    plot(xplot,yplot, 'b')
    hold on
    plot(xplot, beta_wanted(j)*ones(simulations), 'r');
end
end
```

Figure 21: Test for the optimal beta $\hat{\beta}^M$ from the boosting algorithm 1 in section 4.3

```
function []= simulation_result()
%Estimate the hitrates_bo and hitrates_nb for the optimal 'input-values'
M_max       =100; %amountboostings
p_max       =10; %maximum amount of lags
r_max       =6; %maximum amount of factors
n           =52; %amount of variables
iterations  =100; %amount of iterations
T           =54; %amount of observations
v           = 0.2; %step length
factor      = 3; %vamount of f_ar
hitrate_nb=[];
hitrate_b=[];
for j=1:10
hitrate_amount_nb =0;
hitrate_amount_b=0;
for i=1:iterations
    [~, X, y]                   = make_X_y_F(n,T,factor);
    r                           = NbFactorsSimulation(X,r_max);
    [E, ~, Wt_Bw]          = armodel_simulation(y, p_max);
    [~, ~, BetaOpt]             = l2boost_simulation(X, y, M_max, v, r, y(end), Wt_Bw, E);
    if sum(BetaOpt~=0)==factor
        hitrate_amount_b = hitrate_amount_b+1;
    end
    if r==factor
        hitrate_amount_nb          = hitrate_amount_nb+1;
    end
end
    hitrate_nb(j)    = hitrate_amount_nb/iterations %estimate hitrate_nb
    hitrate_b (j)    = hitrate_amount_b/iterations %estimate hitrate_b
end
nb_result = mean(hitrate_nb) %estimate the overall hitrate_nb
b_result = mean(hitrate_b) %estimate the overall hitrate_b
end
```

Figure 22: The estimated hit-rate$_b o$ and estimated hit-rate$_p r$ with specific 'input values'

7.3.2 Codes with the African Data Set

```
MATLAB read-me file for the African data set

armodel_Africa.m
-> This implement the auto-regressive (AR) model with optimal lag order p_bic. The estimates of the additional explanatory
     variable W_t and parameter  beta^_w, and the T x 1residual vector E in the boosting algorithm 1 of section 4.3 are
     estimated.

NbFactorsAfrica.m
-> This give you the result of the 'pre-selected' amount of factors. The code is implemented from Christophe Hurlin (2013).

l2boost_Africa.m
-> his gives you the result of boosting algorithm 1 in section 4.3.

testbreaks.m
-> This test if structural breaks need to be implemented in the models. The results are given in table 1 in section 5.2.

Africa.m
-> This makes forecast and gives the MSFE off the boosting algorithm and the AR model with optimal lag order. The results
     are given in figure 6 of section 5.2.
```

Figure 23: Read me file for the simulation

```
function [res_ar, lags, Wt_Bw, fcast]= armodel_Africa(y, pmax)
% Estimate the residuals, lags, the coefficient Wt_Bw and the value of the
% forecast with a AR model with optimal lag order. This lag order is
% selected by BIC with maximum amount of lags p_max
T=size(y, 1);
optimal_bic=0;
Wt_lags=[];
for i=1:p_max %Estimate the optimal amount of lags with the bic value
    model = ar(y,i);
    bic=model.Report.Fit.BIC;
    if (i==1)  || (bic<optimal_bic)
        optimal_bic=bic;
        lags=i;
    end
end

Mdl = arima(lags,0,0); % estimate ar model with optimal amount of lags
[ar_model,~,~,info] = estimate(Mdl,y, 'Display', 'off');
fcast = forecast(ar_model,1,'Y0',y);

[res_ar] = infer(ar_model,y); % AR(lags) residuals
information= info.X;
coef_ar=[];
for i=1:lags+1 %estimate coefficients of the AR(lags)
    [coef_ar]= [coef_ar; information(i)];
end
for i=1:lags %estimate the Wt value in the boosting algrithm
    Wt_lags=[Wt_lags, y(T-i+1)];
end
Wt= [1, Wt_lags]; %estimate the Wt value in the boosting algorithm
Wt_Bw= Wt*coef_ar; %estimate the WtBw value in the boosting algirhtm
end
```

Figure 24: Construct W_t, E, $\hat{\beta_W}$ from equation 2 by the auto-regressive model

```
function [khat]=NbFactorsAfrica(X,r_max)
if nargin==1, r_max=min(size(X)); end                          % Rule proposed by Bai and Ng to choose kmax
if isnan(r_max)==1, r_max=min(size(X)); end                    % Rule proposed by Bai and Ng to choose kmax
[T,N]=size(X)   ;                                              % Sample Sizes
if T<N                                                         % Choice of normalization according the computional cost
    [vectors,values] = eig(X*X');                              % Eigenvalues and eigenvectors of XX'
    factors=sqrt(T)*vectors(:,T-r_max+1:T);                    % Estimated Factors with kmax Factors
    loadings=X'*factors/T;                                     % Estimated Matrix of Factor Loadings
    betahat=loadings*chol(loadings'*loadings/N);               % Rescaled Estimator of the Factor Loading
else                                                           % Case T>N
    [vectors,values] = eig(X'*X);                              % Eigenvalues and eigenvectors of X'X
    loadings=sqrt(N)*vectors(:,N-r_max+1:N);                   % Estimated Matrix of Factor Loadings with kmax Factors
                                                ]
    betahat=(X'*X)*loadings/(N*T);                             % Rescaled Estimator of the Factor Loading
end
Z=X-X*betahat*inv(betahat'*betahat)*betahat';                  % Estimated Residuals
var_Z_kmax = sum(sum(Z.^2))/(N*T);                             % Estimated Variance of Residuals with kmax factors
V=zeros(r_max,1);                                              % Vector of V(k,Fk) for k=1,..,kmax
for k=1:r_max                                                  % Loop on the number of factor k
    if T<N                                                     % Choice of normalization according the computional cost
        [vectors,values] = eig(X*X');                          % Eigenvalues and eigenvectors of XX'
        factors=sqrt(T)*vectors(:,T-k+1:T);                    % Estimated Factors with kmax Factors
        loadings=X'*factors/T;                                 % Estimated Matrix of Factor Loadings
        betahat=loadings*chol(loadings'*loadings/N);           % Rescaled Estimator of the Factor Loading
    else                                                       % Case T>N
        [vectors,values] = eig(X'*X);                          % Eigenvalues and eigenvectors of X'X
        loadings=sqrt(N)*vectors(:,N-k+1:N);                   % Estimated Matrix of Factor Loadings with kmax Factors
        betahat=(X'*X)*loadings/(N*T);                         % Rescaled Estimator of the Factor Loading
    end
    Z=X-X*betahat*inv(betahat'*betahat)*betahat';              % Estimated Residuals
    V(k) = sum(sum(Z.^2))/(N*T);                               % V(k,Fk)
end
BIC3=repmat((0:1:r_max)',1,2);                                 % BIC information Criteria (BIC1, BIC2 and BIC3)
BIC3(1,2)=mean(sum(X.*X/T));                                   % PC information Criteria when r=0
CNT=min(N,T);                                                  % Function Cnt^2
Penalty=[(N+T)/(N*T)*log((N*T)/(N+T)) ...                      % Penalty Terms for IC and PC
    (N+T)/(N*T)*log(CNT) log(CNT)/CNT ];
Penalty=repmat(Penalty,r_max,1);                               % Penalty Terms for IC and PC
kk=repmat((1:1:r_max)',1,3);                                   % Matrix with increments
BIC3(2:end,2)=V+kk(:,1)*var_Z_kmax.*(N+T-kk(:,1))*log(N*T)/(N*T); % BIC3 criterium
[BIC3s,khat_BIC3] = min(BIC3(:,2));khat_BIC3=khat_BIC3-1;      % Estimated Numbers of Factor with BIC3
khat = khat_BIC3;                                              % Estimated Numbers of Factor with BIC3 criteria
                                                % BIC3 Information criterium for k=1,..,kmax
```

Figure 25: Select 'pre-selected' amount of factors, code from Christophe Hurlin (2013)

```
function [y_new, MSFE, BetaOpt,  F, mu_result] = l2boost_Africa(X, y, M_max, v, r, y_actual, Wt_Bw, E)
% Implemention of the boosting algorihtm
%input:
% X          explanatory variable
% y          dependent variable
% M_max      maximum number of boosting iterations
% v          steplength
% r          pre-selected amount of factors
% y_actual   real value of the GDP of Tanzania
% Wt_Bw      Estimation of WtBw in boosting algorithm
% E          E are the residuals of the ar model with optimal lag order
% lags       number of optimal lags
% outcome:
% y_new      forecasted GDP of Tanzania
% MSFE       Mean squared forecast error of the boosting algorihtm
% BetaOpt    Optimal beta estimated in the boosting alogirthm
% F          dynmically factors
% mu_result mu^M(Ft) in the boosting algorithm

T                = size(y,1);
ICm              = zeros(M_max,1);
MuHat            = zeros(T,M_max);
MuHat(:,1)       = mean(E, 'omitnan');
standardizedx    = zscore(X);
F                = standardizedx*pca(standardizedx, 'VariableWeights', 'variance');
beta             = zeros(r, M_max);
Bm(:,:,1)        = ones(T,T,1)'*ones(T,T,1)'/(T);
SSR_min          = zeros(M_max,1);
for i = 1:M_max %boosting iterations
    u            = E - MuHat(:,i);
    SSR          = zeros(r,1);
    for j = 1:r
        [~,~,dhat] = mvregress(F(:,j), u);
        SSR(j)     = sum(dhat.^2);
    end
    [minSSR,jstar]  = min(SSR);
    bhat             = mvregress(F(:,jstar), u);
    gstarHat         = bhat'*F(:,jstar);
    MuHat(:,i+1)     = MuHat(:,i) + v .* gstarHat;

    %Updating Beta estimates using shrinkage parameters
    one_regressor           = zeros(r,1);
    one_regressor(jstar,1)  = 1;
    beta(:, i+1)            = beta(:,i) + v*bhat*one_regressor;

    % Updating degrees of freedom
    Pm          = F(:,jstar)*(F(:,jstar)'*F(:,jstar))^(-1)*F(:,jstar)';
    Bm(:,:,i+1) = Bm(:,:,i) + v*Pm*(eye(T) - Bm(:,:,i));
    dfm         = trace(Bm(:,:,i+1));
    % Estimating variance of boosting residuals
    sqrt_sigma_m_hat = (y-MuHat(:,i+1))' * (y-MuHat(:,i+1));
    % Computing information criteria
    ICm(i,1)     = log(sqrt_sigma_m_hat) + (log(T) * dfm) /T;
    SSR_min(i,1) = minSSR;
end
[~, Mopt]    = min(ICm);             % estimating the optimal amount of boosting iterations
BetaOpt      = beta(:,Mopt+1);       % selecting optimal beta
muopt        = MuHat(:,Mopt+1);      % selecting opitmal mu(F) for the forecast
y_new        = Wt_Bw + muopt(T);     % forecast
MSFE         = (y_actual-y_new)^2;   % calculate MSFE;
mu_result    =muopt(T);
end
```


Figure 26: Boosting algorithm

```
function [res_ar, skewness_result, kurtosis_result] = testbreaks(y)
%test for structural breaks
skewness_result=[];
kurtosis_result=[];
test=[]; %Jarquebera test value
p=[]; %p-value
place =1;
for t=50:54
    [ar, ~,~, info_result]=estimate(arima(2,0,0 ), y(1:t), 'Display', 'off');
    information= info_result.X;
    [res_ar]= infer(ar,y(1:t));
    skewness_result(place,1)= skewness(res_ar)
    kurtosis_result(place,1) = kurtosis(res_ar)
    [~ ,p(:,place), test(:,place)]=jbtest(res_ar);
    place = place + 1
end
end
```

Figure 27: Test if structural breaks needs to be implemented in the models

```
ffunction [MSFE_Boost,MSFE_Benchmark] = Africa()
load data_Africa
M_max       =100; %amountboostings
p_max       =10; %maximum amount of lags
r_max       =6; %maximum amount of factors
v           =0.2; %step length
MSFE_boost=[];
MSFE_ar=[];
place=1;
BetaOpt_result=[];
GDP_Tanzania= zeros(5,1);
lag_order=[];
[T] =size(x,1);
for fc=5:-1:1 %forecast of the GDP of Tanzania from 2012 till 2016
r= NbFactorsAfrica(x,r_max) %estimate optimal amount of 'pre-selected' factors
[E, lags,Wt_Bw,fcast] = armodel_Africa(y(1:(T-fc)), p_max); %estimate benchmark model
MSFE_ar(place,1) = (y(T-fc+1)-fcast)^2; %MSFE benchmark model
MSFE_ar(place,2) = fcast;
lag_order(place,1) = lags;
[y_new, MSFE, BetaOpt] = l2boost_Africa(x(1:(T-fc),:), y(1:(T-fc)), M_max, v, r, y(T-fc+1), Wt_Bw, E); %estimate boosting
    algirthm
MSFE_boost(place,1)= MSFE; %MSFE boosting algorithm
MSFE_boost(place,2) = y_new;
GDP_Tanzania(place)= y(T-fc+1);
BetaOpt_result(:,place)=BetaOpt;
place=place+1;
end
MSFE_Boost= sum(MSFE_boost(:,1))
MSFE_Benchmark= sum(MSFE_ar(:,1))
xplot=1:5;
plot(xplot, MSFE_boost(:,2))
hold on
plot(xplot, MSFE_ar(:,2))
hold on
plot(xplot, GDP_Tanzania)
legend(["Forecast boosting";"Forecast benchmark"; "GDP Tanzania"])
end
```

Figure 28: Main code estimating the GDP of Tanzania, and he MSFE with the boosting algorithm that includes PCA and and the AR model with optimal lag order (p_{bic})