STOCKHOLM SCHOOL OF ECONOMICS
Department of Economics
659 Degree project in economics
Spring 2021

The possibility of machine learning algorithms to explain long-run economic growth

Gabriel Ehrnst Grundin (24525) and Daniel Helldén (23959)

Abstract: The empirical investigation of economic growth has been one of the most researched topics in economics. Most recently, machine learning algorithms that can handle nonlinearities, discontinuities and other issues inherent with traditional linear approaches have been proposed to be able to more accurately describe the empirical determinants of economic growth. We investigate the determinants of economic growth using three state-of-the-art machine learning algorithms (Support Vector Regression Machines, Gaussian Process Boosting and Long Short-Term Memory Neural Networks) together with a more conventional regression algorithm (Generalized Additive Model), on 28 European countries between 1950-2019 and 56 explanatory variables. The machine learning algorithms diverge in their prediction patterns and estimated variable importance, and do not provide a compelling improvement over traditional methods.

Keywords: Economic Growth, Machine Learning, Empirical Econometrics

JEL: C33, C45, O11

Supervisor: Andreea Enache
Date submitted: 17 May 2021
Date examined: 27 May 2021
Discussants: Hannes Ludvigsson & Elias Jbari
Examiner: Johanna Wallenius

Table of Contents

1. Introduction 1

Purpose 1

Current state of knowledge 2

Economic growth since the 1950s – A snapshot 2

Empirical investigation of economic growth 2

Machine learning within empirical economics 4

2. Research questions 7

Limitations of scope 7

3. Methods 8

Data sample 8

Data sources and variables 9

Data preparation 9

Statistical analysis 10

Support Vector Regression Machines (SVRM) 10

Gaussian Process Boosting (GPBoost) 11

Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs) 13

Generalized Additive Linear Models (GAMs) 14

SHapley Additive exPlanations (SHAP) Values and Variable Importance 15

4. Results 17

Fit of the regression models 17

Variable importance 19

Predictive performance 26

5. Discussion 27

Comparison of algorithms 27

Results compared with previous literature 28

Limitations 30

Future studies 30

5. Conclusions 31

6. References 32

7. Appendices 37

Appendix A – Preparation of data (prescript.py) 37

Appendix B – Support Vector Regression Machines (SVM.py) 40

Appendix C – Gaussian Process Boosting (GPBoost) (GPBoost.py) 42

Appendix D – Long Short-Term Memory Recurrent Neural Network (imvlstm.py) 45

Appendix E – Generalized Additive Linear Model (GAM.py) 50

1. Introduction

Purpose

In macroeconomics, one of the most often asked and fundamental questions is: "what causes economic growth?" Why do average incomes vary so much between countries? How can we explain why some countries are rich, and some poor? What causes some economies to grow rapidly, and some not at all?

Among the most influential models explain growth in terms of a country's endowment of total factor productivity, capital and labor (Solow, 1956). Since its debut in the 1950s this model has been expanded and developed in a number of important directions (Swan, 1956). Other influential models distinguish between intangible ideas and tangible objects, and their relative importance for explaining growth (Romer, 1990).

The Solow-Swan and Romer models all proceed from macroeconomic assumptions about production. Other models make more explicit assumptions about the connection between growth and microeconomic behavior regarding consumption (Ramsey, 1928). These, as well, have seen important extensions made during the 20th century (Cass, 1965). General theories of economic growth have developed and are extensive, yet they often lack the broader empirical basis on which assumptions can be asserted (Jones, 2016). Further, they are often open-ended in the sense that one model does not logically deny the existence of other possible explanations of economic growth, making it nearly impossible to create complete empirical models based on the theories. Although there have been a multitude of empirical examinations on the determinants and drivers of economic growth, particularly with the advent of more sophisticated statistical methods and better quality of economic data in the late 20st century (Panhans and Singleton, 2015), the empirical approaches often fall short due to the lack of quality data and possibility to assess irregular, nonlinear or noisy real world data. In summary, the empirical exploration of the determinants of economic growth have not been as successful as its more theoretical counterpart.

Re-visiting one of the most essential topics in economics, we aim to contribute to the empirical approaches for economic growth through a data-driven approach, applying novel state-of-the-art machine learning algorithms on a broad set of societal indicators to investigate the possible causes and correlates of economic growth.

Current state of knowledge

Economic growth since the 1950s – A snapshot

For much of the history of the world, real GDP growth across the world was negligible by present-day standards. Real growth rates were not always positive, and when they were, seldom above 0.5% annually (Maddison, 2021). After industrialization took off in the 18th and 19th centuries, growth rates did as well. More than two centuries later, the countries of Europe continue growing at a brisk pace.

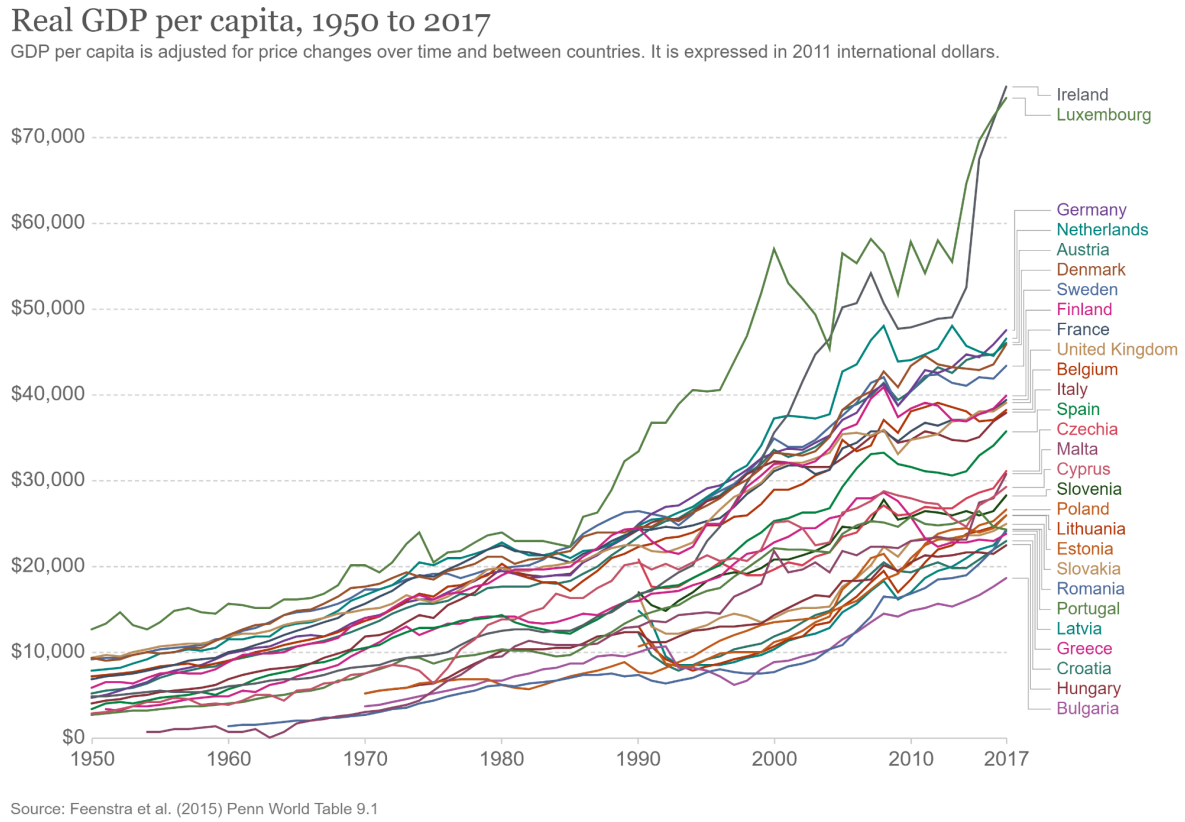

Figure 1. Real GDP per capita for the five most populous countries currently in the European Union, from 1950 until 2017. Layout from OurWorldInData.

Empirical investigation of economic growth

The literature on the empirical determinants of economic growth is vast, with cross-country comparisons yielding more than 140 indicators that could explain long run economic growth (Moral-Benito, 2012). Traditional neoliberal theories usually utilized single-country comparisons when trying to determine the drivers of economic growth. Rising to prominence with the seminal work of Islam in the early 1990s, panel data sets that distinguish the country-specific determinants

of economic growth became the standard mode within the area of empirical economics, and were used to evaluate various patterns and theories such as the convergence theory (Islam, 1995).

One of the most significant contributors to the field, Barro examined country growth rates using ordinary least squares linear regression based on panel data sets with a relatively small number of explanatory variables, including logarithmized per capita GDP, male upper-level schooling, the inverse of life expectancy at age one, logarithmized total fertility rate, government consumption ratio, change in terms of trade, investment ratio and the inflation rate. The linear model also included a number of subjective indices as variables in the form of rule of law, democracy, and openness. Based purely on data availability, 87 countries were sampled on the above variables averaged across 10-year intervals. Particularly interesting, Barro found evidence of a conditional convergence of economic growth rates between countries that were dependent on the starting state of the economy. Further, the panel regression showed that given a certain GDP per capita and human capital, economic growth seemed to depend positively on the rule of law and the investment ratio and negatively on the fertility rate, the ratio of government consumption to GDP, and the inflation rate (Barro, 2003).

In a similar fashion, Gallup et al. argued with the help of a cross country regression on a panel data set (combined with a new approach to incorporate transport costs into growth theory) with a broad range of explanatory variables that geography determines economic growth through various channels (Gallup, Sachs and Mellinger, 1998). Similarly, Acemoglu et al. used a panel data set and traditional regression methods to determine that democracy causes economic growth (Acemoglu et al., 2019). A different kind of topic is explanatory variables that have effect only after a certain time-lag. For example, increasing education expenditure can have an effect on growth, but only after a certain number of years (Chandra and Islamia, 2010).

The work of Barro, Gallup and Acemoglu and their colleagues follows a tradition within cross-country analysis that utilizes only a few selected variables, with the hope of establishing a significant relationship between economic growth and the variables of interest. Levine and Renelt showed already in 1992 that such an approach is highly sensitive to the explanatory variables chosen, and that there were essentially no combination of explanatory variables that held up to rigorous sensitivity investigation (Levine and Renelt, 1992). Further, the panel data sets are composed of often modeled underlying data, and changes to the methods of collecting, aggregating or modeling data have grave effects on the results of regressions (Ciccone and Jarociński, 2010).

Aiming to amend the model-specificity problem, a separate approach gained momentum at the beginning of the 2000s which tried to implement Bayesian models that incorporate the a-priori expected uncertainty. Specifically, Sala-i-Martin et al. used Bayesian Averaging of Classical Estimates approach that formulated an estimate of the value of the determinant based on both traditional linear regression and weighted Bayesian models that incorporate the uncertainty of variable selection explicitly. Running through 67 possible explanatory variables for economic growth for 88 countries between the 1960-1996 they found 18 variables that were robustly associated with economic growth. These included primary schooling enrolment, the relative price of investment goods, initial level of income, regional dummies as well as some measures of human capital as well as the public consumption and public investment shares (Sala-i-Martin, Doppelhofer and Miller, 2004).

More recently, Moral-Benito extended the Bayesian approach to formulate a Bayesian Averaging of Maximum Likelihood model which incorporate the uncertainty inherent in each possible configuration of explanatory variables as well as their underlying distributions, applying it on a set of 35 variables in 73 countries between 1960-2000. He concludes that the investment price, distance to major world cities and political rights hold up to be the most robust determinants of economic growth in the data set (Moral-Benito, 2012). Utilizing a similar Bayesian approach, Cuaresma et al. studied the possible determinants of economic growth in 255 European sub-national regions between 1995 and 2005 including 50 explanatory variables, finding presence of capital cities and a higher presence of educated workers in the labor force positively associated with economic growth (Cuaresma, Doppelhofer and Feldkircher, 2014).

In the late 1990s, Durlauf and Quah surveyed the landscape of empirical economics, noting that progress had been made, however statistical methods that could account for nonlinearities and discontinuities remained uninvestigated and traditional methods would be unsuccessful in fully capturing the determinants of economic growth. Bayesian models notwithstanding, it seemed that traditional statistical methods and modes of investigation did not move the understanding of the empirical determinants of economic growth forward (Durlauf and Quah, 1998).

Machine learning within empirical economics

The use of machine learning methods to alleviate some of the obstacles within economic growth research begun already in the 1990s (Lee, White and Granger, 1993; Kuan and White, 1994; Swanson and White, 1997) however it has become an increasing focus of research in more recent

years (Varian, 2014; Athey and Imbens, 2019). The most important issues with standard parametric, linear or Bayesian methods that machine learning offers a possible solution to is the difficulty of handling discontinuities, nonlinear relationships and highly correlated independent variables when trying to decipher the empirical determinants of economic growth.

Being based on methods that are not confined to the traditional rigid statistical assumptions, shifting the analysis from a pure hypothetical testing approach to a data driven or learning approach while utilizing a broader set of explanatory variables allows machine learning methods to discover new patterns and deliver new understandings of the determinants of economic growth (Mullainathan and Spiess, 2017; Coulombe et al., 2020). The problem with nonlinearity further seems to vary with the availability and quality of data, and is particularly prominent in developing countries, making the use of machine learning methods for understanding growth patterns in such countries especially interesting (Chuku, Simpasa and Oduor, 2019). The most obvious downsides to using machine learning is the difficulty of assessing statistical inference and providing sufficient interpretability for reproducibility, which have been crucial components of traditional econometrics.

A few studies have applied machine learning approaches to investigate econometric and economic growth patterns. Some have taken a broader perspective, with datasets including several countries, while others have focused more on a limited set with only a single country and more parameters. For example, Sokolov-Mladenović et al. used machine learning to predict GDP growth rates based on data from Eurostat (for the 28 countries then in the EU) and five parameters: trade in services, exports of goods and services, imports of goods and services, trade, and merchandise trade. The study used two different machine learning methods, an artificial neural network with extreme learning method and an artificial neural network with back-propagation. The first network could forecast GDP growth rate with a root-mean square error (RMSE) of 0.4289 and a determination coefficient (R^2) of 0.9884, while the second network produced a RMSE of 1.2883 and R^2 of 0.8949 (Sokolov-Mladenović et al., 2016). Similarly, Milačić et al. also applied the same neural network methods to predict growth rates in Portugal, based on four parameters: agriculture, manufacturing, industry (including mining, construction, electricity), and services. Each parameter was measured by adding all outputs from the sector and subtracting intermediate inputs. In this narrower dataset, the prediction accuracy was given by an RMSE of 1.97 and R^2 of 0.73 for the first network and RMSE 2.81 and R^2 of 0.45 for the second network (Milačić et al., 2017). Cicceri et al. have used machine learning to forecast Italian recessions specifically. Their variables included the inflation

rate, the unemployment rate, industrial production, gross debt, stock market index, average interest rate at issuing for short term bonds, interest rate for long term bonds, balance of payments, final consumption aggregates, state budget and government deficit/surplus. The time period studied consisted of the first quarter of 1995 to the second quarter of 2019. The study utilized seven different regression estimation techniques: a autoregressive model, ordinary least squares regression, nonlinear autoregression models, nonlinear autoregressive with exogenous variables model, support vector regression machines, k-nearest neighbors, and boosted trees. The two machine learning methods, boosted trees and support vector regression machines, achieved an overall predictive accuracy of around 80%, whereas their proposed nonlinear autoregressive model with exogenous variables had a predictive accuracy of 87%. The results constitute a statistically significant improvement over the standard ordinary least squares regression, measured in terms of their mean-squared errors (Cicceri, Inserra and Limosani, 2020).

Several studies have used machine learning methods to investigate various parameters that can have an indirect effect on economic growth. Using a support vector regression machines algorithm using a series of short-term rates (treasury bills) as well as long-term rates (bonds) from 1976 until 2011, together with the real GDP for the same period one study achieved a forecasting accuracy of 66.7% at predicting recessions (Gogas et al., 2015). One study forecasted inflation in the United States from 1984 to 2014 using a set of different methods, both traditional (e.g. autoregressive) as well as machine learning algorithms (e.g. artificial neural networks and support vector regression machines). It included four different inflation indicators as well as four different time horizons, for a total of 16 different conditions. The machine learning algorithms were better in seven of these conditions, and the traditional time series models were better in the other nine. The study concluded that support vector regression machines outperforms other models at predicting personal consumption expenditure inflation. The machine learning algorithms were particularly good the more volatile the underlying data series were (Ülke, Sahin and Subasi, 2018). Another study focused on tourism to China, a sector that accounts for 6.3% of the country's GDP. The study used a kernel extreme learning machine to forecast the number of tourist arrivals using queries at the Google and Baidu search engines as input. It concluded that machine learning methods were superior to traditional methods, such as autoregressive integrated moving average with exogenous variables, the outperformance was both in terms of forecasting accuracy and robustness (Sun et al., 2019). A similar study again used Baidu search engine data, but instead with a least squares support vector regression machine model with gravitational search algorithm in order to make the calculations less time consuming (Xie, Qian and Wang, 2021). The new method

was more accurate than the traditional methods, with a root-mean square error one-fifth as high as that of the autoregressive integrated moving average with exogenous variables model. Another study investigated natural gas use in Istanbul. It contrasted three methods: multiple linear regression, artificial neural network, and support vector regression machines. The support vector regression machines algorithm was best in terms of mean absolute percentage error, with a value of 8.14 compared to 9.89 for artificial neural network and 18.7 for the multiple linear regression (Beyca et al., 2019). Another set of studies have looked into the possibility of machine learning algorithms for estimating the price of crude oil, succeeding to various degrees (Gabralla, Jammazi and Abraham, 2013; Yu, Dai and Tang, 2016; Chen, He and Tso, 2017; An, Mikhaylov and Moiseev, 2019).

Together, these examples demonstrate the breadth of the applicability of machine learning algorithms within empirical economics studying economic growth. Although machine learning algorithms have become a more prominent tool in the toolbox of econometrics, historic economic growth and its determinants remain surprisingly understudied. To our knowledge, no study of such focus has been published.

2. Research questions

To determine the most important empirical factors for long run growth through of state-of-the-art machine learning algorithms and investigate the possibility to predict historic economic growth. Specifically, to

- Assess the applicability of support vector regression machines, gradient boosted decision trees and long-short neural networks for untangling complex non-linear associations within historic economic data.
- Study the relative importance of different economical and societal features for historic economic growth.
- Investigate the possibility of support vector regression machines, gradient boosted decision trees and long-short neural networks to predict historic economic growth.

Limitations of scope

In this analysis, we limit both the temporal and spatial aspects of the research question. First, we limit ourselves to a specific time period (1950-2019) primarily because this is the time period for which data from a range of explanatory empirical variables are available. Secondly, primarily due to

the data availability and quality aspects, we focus on European countries. Since the focus of the thesis is to investigate the applicability of machine learning algorithms and the empirical explanations of economic growth, we deem it beyond the scope to provide a more general in-depth mathematical or theoretical analysis of the feasibility of machine learning algorithms within empirical economics. Similarly, the traditional models and theories of economic growth are only touched upon lightly as we explore the empirical drivers of economic growth.

3. Methods

The investigation of the determinants of economic growth has a long history, however recent advancement in two areas have made it necessary to revisit the standard empirical explanation for economic growth. The first is the ever more detailed macroeconomic data available over a longer time horizon for many countries that were not available a decade ago. The second is the development of machine learning algorithms that can untangle complex non-linear relationships and function in non-parametric real-world settings. Overall, the methodological approach applied follows a data-driven machine learning approach, which tries to estimate relationships from raw data without a-priori assumptions or hypotheses that can hinder a broader analysis of a phenomenon.

Given the research question, we first assume the problem as a standard regression scenario whereby the independent or explanatory variables provide the input for a function that is able to predict the actual GDP growth rate of each country. Hence, the primary evaluation criteria for the applied algorithms are the fit of the proposed function and which variables that are the most important for providing such a function. As a secondary additional analysis, holding all other things the equal, the algorithms are evaluated on how well the function can predict GDP growth rate on a set of previously unobserved data, which indicates the ability of the algorithm to predict future GDP growth rates.

Data sample

The advantage of machine learning methods is particularly notable for large datasets. For this paper, we would preferably have as many data points as possible. However, of the circa 200 countries in the world, many do not have accurate data available, at least not when going more than a decade back in time. The countries for which complete data is available also tend to be comparatively rich and well-developed. Further, the availability of data has to be weighed against the

representativeness of the sample of countries. Balancing the pros and cons, we have chosen to make the following tradeoff. We choose to limit the analysis geographically to the EU countries and the United Kingdom and temporally to the time period of 1950-2019. Specifically, available data were collected for each year and country (Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden and the United Kingdom) during this period.

Data sources and variables

Combining the two most robust and rigorously compiled datasets, Penn World Tables version 10.0 which is provided by the University of Groningen and the most recent edition of the World Development Indicators compiled by the World Bank, the dependent variable is the annual GDP growth rate while all other available variables were considered independent possible explanatory variables (Groningen Growth and Development Centre, 2021; World Bank, 2021). Our data also includes the Freedom House Index categorization of the countries' respective status as "free", "partly free" or "unfree", (Freedom House, 2021) and the average number of years of schooling for the adult population (Barro, 2013; Lee, 2016).

Data preparation

In order to utilize the data for our purposes, the separate sources were combined into one single data file with a total of 1,490 independent variables. Independent variables similar to GDP growth rate and similarly redundant variables were excluded. Examples of redundant variables are the percentage of the population that is *male* and younger than five, which for the chosen countries is nearly identical to the percentage of the population that is *female* and younger than five. Missing values were prominent in the data set. To avoid bias, variables for which more than 50% of observations were missing were also excluded. This resulted in a final set of 56 independent variables (listed in the result section), onto which missing data was linearly interpolated and continuous variables were standardized to avoid independent variables with a large range to have a disproportionate large effect on the machine learning algorithms. For the primary evaluation criteria, algorithms were applied on the full dataset while for the evaluation of the projection possibility of the algorithms the dataset was divided into a training set containing 80% of the observations that are "seen" by the algorithms and a test set of 20% of the observations that are "unseen". The Python script for the full data preparation is included in appendix A.

Statistical analysis

An overview of three state-of-the-art used machine learning algorithms and their customization to the research question at hand are provided below, together with a description of a more traditional generalized linear model as well as a novel way to depict variable importance in the form of SHapley Additive exPlanations (SHAP) Values.

Support Vector Regression Machines (SVRM)

Originally developed for classification problems, Drucker et al. extended support vector machines to be able to perform regression (Drucker et al., 1997). In essence, Support Vector Regression Machines (SVRM) are a supervised form of machine learning algorithm that tries to find the function that minimizes the error (called epsilon) of the predicted regression solution. In the ordinary linear case, it is a convex optimization problem that tries to fit a function within the margin of a hyperplane. However, the algorithm is able to be applied in a nonlinear nonparametric setting through the use of different kernels that transform the support vectors into different dimensions and make it possible to analyze time series data (Müller et al., 1997). Concretely, a radial base kernel transforms each support vector into an indefinite number of dimensions to fit a function within a given epsilon/error.

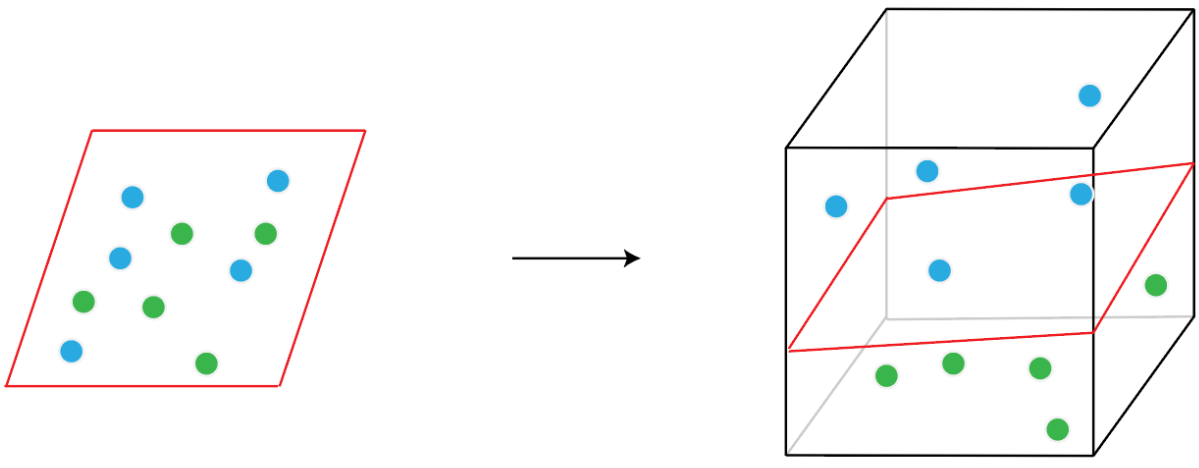

Figure 2. Illustration of finding an optimal hyperplane, from two to three dimensions.

Given training vectors $x_i \in R^p$, $i = 1,\ldots,n$ and a vector $y \in R^n$, the SVRM method solves the following primal optimization problem:

$$min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$$

subject to

$y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i,$

$w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*,$

$\zeta_i, \zeta_i^* \geq 0, i = 1,\ldots,n$

Here, the algorithm is penalizing samples whose prediction is at least ε away from their true target. These samples penalize the objective by $\zeta_i$or ζ_i^* depending on whether their predictions lie above or below the ε tube (Pedregosa et al., 2011).

For the radial base kernel used within the SVRM, the main parameter that needs to be taken into consideration is the designated epsilon, which is the tolerance for error in the function. A high epsilon results in an un-specific function, while too low epsilon results in over-fitting the function to the data and poor predictive performance. To promote reproducibility, the recommended standard value of epsilon = 0,1 was used. For other parameter specifications, a parameter search was conducted using a 4-fold cross validation to find the optimal parameter settings; these were thereafter used in the implementation of the SVRM. The code is available in appendix B.

Gaussian Process Boosting (GPBoost)

First illustrated and put forward by Breiman, in the original version of decision trees, each tree is built based on a bootstrap sample drawn randomly from the original dataset using a decreased gini impurity as the splitting criterion (Breiman, 2001). The gini impurity is in essence how often a variable is labeled wrong given a subset of the prediction. Within a regression setting, a decision tree takes a random set of the explanatory variables and tries to split the variables in the optimal space (minimizing the mean square error) in order to produce the most accurate suggestion for a predictive value. To reduce variance, a large number of decision trees are usually aggregated into a forest and the mean split criteria is used.

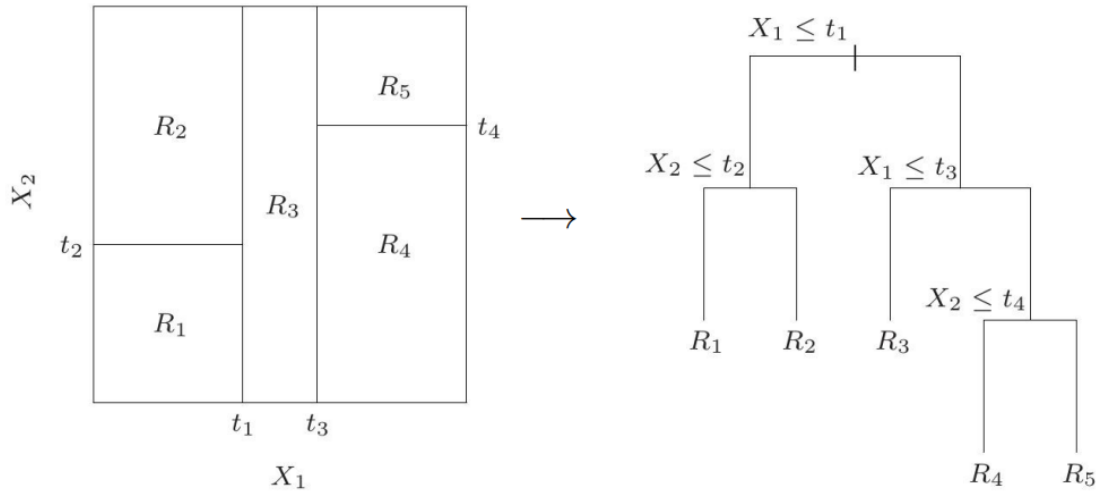

Figure 3. Simple example of a decision tree with two independent variables. In implementation, a random subset of variables at each split is considered and a large number of trees used.

However, in its original form, decision trees cannot account for temporal changes, making it unsuitable for regression across time series. This can be solved by combining a boosting algorithm with decision trees as base learners which can automatically handle non-linearities, discontinuities and other nonparametric aspects (Elith, Leathwick and Hastie, 2008; Johnson and Zhang, 2013) combined with a mixed-model setting that allows for a longitudinal or panel data sets.

Formally, Sigrist propose a mixed-effects model which we follow and can be described as

$$y = F(X) + Zb + \varepsilon,\ b \sim N(0, \Sigma),\ \varepsilon \sim N(0, \sigma^2 I_n)$$

where $y = (y_1,...,y_n)^T \in \mathbb{R}^n$ is the response variable, $F(X) \in \mathbb{R}^n$ are the so-called fixed effects, $b \in \mathbb{R}^m$ are the random effects with covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$, and $\varepsilon = (\varepsilon_1,...,\varepsilon_n)^T \in \mathbb{R}^n$ is an independent error term. Specifically, $F(X)$ is the row-wise evaluation of a function $F : \mathbb{R}^p \to \mathbb{R}$, i.e., $F(X) = (F(X_1),...,F(X_n))^T$, where $X_i = (X_{i1} ...,X_{ip})^T \in \mathbb{R}^p$ is the i-th row of X containing predictor variables for observation i, $i = 1, \ldots, n$. The matrices $X \in \mathbb{R}^{n \times p}$ and $Z \in \mathbb{R}^{n \times m}$ are the fixed and random effects predictor variable matrices. Further, *n* denotes the number of data points, *m* denotes the dimension of the random effects *b*, and *p* denotes the number of independent variables in X (Sigrist, 2020).

Developed by Sigrist the Gaussian Process Boosting algorithm (GPBoost) uses decision trees as a base learner for a nonlinear and nonparametric function F(X) together with a fixed effects component that are suitable for panel data. The algorithm learns at every time series step and

changes the designated function accordingly while being able to incorporate the fixed effects of panel data into the predicted regression value (Sigrist, 2020).

For implementation purposes, one needs to specifically consider the number of boosting steps as well as the parameters for the base learners (e.g. decision trees). A standardized parameter search for the number of boosting steps and parameters was conducted using a 4-fold cross validation to find the optimal parameter settings; these were thereafter used in the implementation of the GPBoost algorithm. The code is available in appendix C.

Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs)

Aimed to mirror what happens with information processing in human brains, a neural network consist of nodes that take input information, transform the signal into any type of non-linear function and then based on the weight of the information signal other nodes connected to the node until it reaches an output layer where the signal is transformed into a real value. It has become one of the most used machine learning algorithms due to its nonlinear and nonparametric nature, while having the major drawback of losing information across layers leading the algorithm to forget what happened early in a sequence (Albawi, Mohammed and Al-Zawi, 2018). This is referred to as the vanishing gradient problem with neural networks.

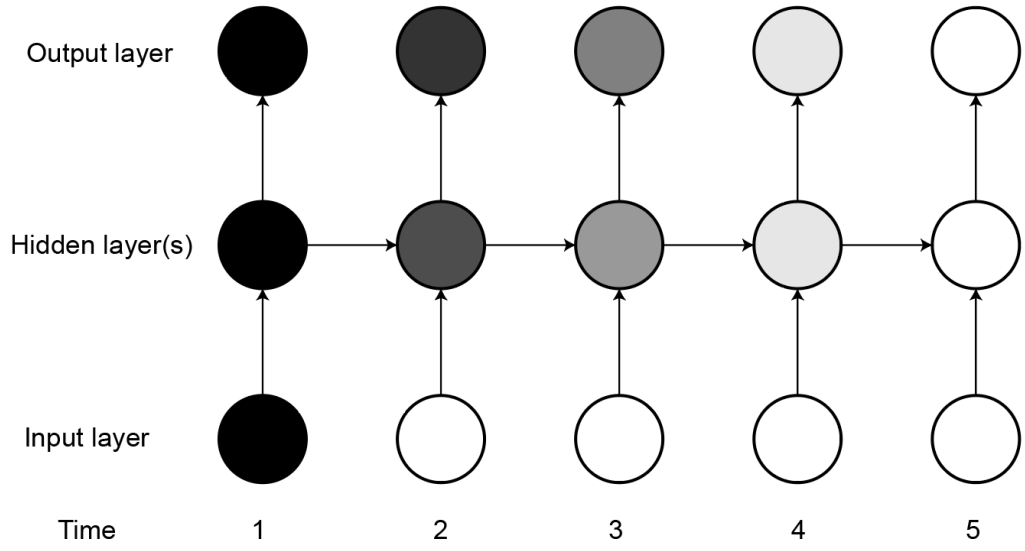

Figure 4. The vanishing gradient problem with neural network learning on time series and illustrated, with only one input considered.

This has been tried to be amended through the development of Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs), which include a function within each node that takes

into consideration the past weighted functions. These developments have led to the LSTM RNNs being able to perform optimization and regression solutions to time series data where events in the far past might affect a later value.

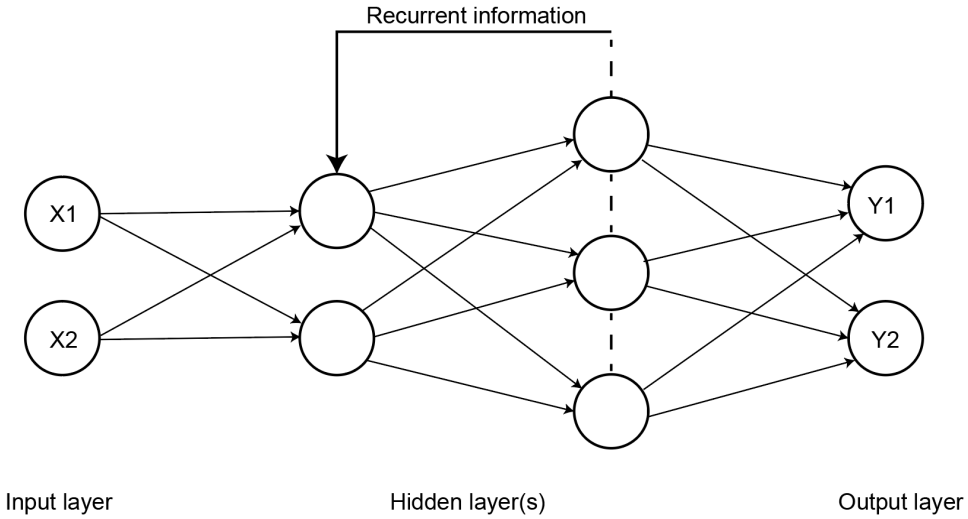

Figure 5. Schematic overview of a long-short term memory recurrent neural network

Another drawback of the LSTM RNNs has traditionally been the lack of interpretability, with many hidden layers of nodes it has been statistically difficult to find exactly how a specific input variable affects the model. Given these two problems, Guo et al. showcased how LSTM RNNs are superior to other machine learning algorithms in accurately learning complex nonlinear and nonparametric regression problems while also adding variable importance as a measure of the relative contribution of the variable to the prediction (Guo, Lin and Antulov-Fantulin, 2019). The mathematical foundations of LSTM RNNs are complex and deemed out of scope for this thesis, for a more rigorous description we refer the reader to (Graves, 2012).

For our implementation of the LSMT RNN, parameters were standardized through a 4-fold cross validation to find the optimal parameter settings to avoid additional complexity, and followed the practical application of the LSTM RNNs algorithm implemented by Guo et al. (Guo, Lin and Antulov-Fantulin, 2019). The code is available in appendix D.

Generalized Additive Linear Models (GAMs)

As the foremost generalization of the standard linear model approach, generalized additive linear models (GAMs) is a generalized linear model that relates a univariate response variable, Y, to predictor variables, x_i. An exponential family distribution is specified for Y (for example normal, binomial or Poisson distributions) along with a link function g (for example the identity or log

functions) relating the expected value of *Y* to the predictor variables via the following simple structure:

$$G(E(Y)) = \beta_0 + f_1(x_1) + f_2(x_2) + f_i(x_m)$$

The functions f_i may be functions with a specified parametric form (for example a polynomial, or an un-penalized regression spline of a variable) or may be specified non-parametrically, or semi-parametrically simply as smooth functions to be estimated by non-parametric means (Larsen, 2015).

The unique features of GAMs are that they are additive and relaxes the assumption of linearity between the dependent and independent variables makes for a much more flexible approach using splines. The splines are particularly useful as no a-priori transformation of variables are necessary for upholding traditional parametric assumptions, and they are formed by a set of base functions that does not interfere with the dependent variable. For a more detailed introduction to splines and their mathematical properties and GAMs we urge readers to consider (Hastie and Tibshirani, 1991).

In contrast to machine learning algorithms, GAMs provide a significance level (P-value) of the independent variable contribution in a regression setting which allows for more traditional interpretation of the importance of individual variables. However these must be interpreted with caution as the P-values calculated do not take into account the uncertainty from the splines and smoothing estimate and therefore often underestimate the P-value (Hastie and Tibshirani, 1985). Therefore, no specific significance level was deemed arbitrarily significant in the analysis.

For our purposes, a normal distribution of the underlying Y variable was assumed, and splines of continuous independent variables estimated by non-parametric means where the standard smoothing parameter of lambda = 0.6 used. Categorical independent variables were considered factor terms as per general uses. The code is available in appendix E.

SHapley Additive exPlanations (SHAP) Values and Variable Importance

Within machine learning there is usually a tradeoff between accuracy and interpretability. For instance, where a machine learning algorithm such as a SVRM might be more accurate in its prediction than a standard linear regression, it is almost impossible to untangle the individual effects of a variable in the neural network model. To overcome this difficulty or perceived tension, Lundberg and Lee have proposed a framework for interpreting predictions made by machine learning algorithms through SHapley Additive exPlanations (SHAP) Values (Lundberg and Lee, 2017). Using a game-theory approach, the SHAP values show for each respective feature the

change in the expected model prediction based on the value of the feature. One can think about SHAP values in a simple way: a random set of feature values of an individual independent variable that are in a room make a prediction (holding all other independent variables equal), the change in the prediction when one more random variable from the set of independent variables comes into the room is the SHAP value.

Specifically, the SHAP value is defined from a value function (*val*) of players in S, and is its contribution to the payout, weighted and summed over all possible feature value combinations

$$\varphi_j(val) = \sum_{S \subseteq \{x_1,\dots x_p\} \setminus \{x_j\}} \frac{|S|!\,(p - |S| - 1)!}{p!}\,(val(S \cup \{x_j\}) - val(S))$$

where *S* is a subset of the features used in the model, *x* is the vector of feature values of the instance to be explained and *p* the number of features.

s(val) is the prediction for feature values in set S that are marginalized over features that are not included in set S:

$$val_x(S) = \int \hat{f}(x_1, \dots, x_p)\; d\; P_{x \notin S} - E_X(\hat{f}(X))$$

The SHAP values are computed through approximations of the original model based on a set of iterations of the data and the model, leading to a set of SHAP values for each feature which can be used to illustrate the feature usage and importance in the model (Molnar, 2021).

Unfortunately, there are currently no practical applications for the computation of SHAP values for LSTM RNN and GAMs in any common statistical programming language, which is why SHAP values are only computed for the SVRM and GPBoost algorithms in our analysis. For the LSTM RNN algorithm standard variable importance in the form of decreased level of prediction when an independent variable is excluded from the input features is calculated, while the significant values for GAMs give an indication of which variables that significantly affect the regression solution as a more traditional statistical inference approach, complementing the variable importance measures.

4. Results

Fit of the regression models

The results for the first scenario (the traditional regression setting) are presented in table 1. This table gives the root-mean-square error (RMSE) and the mean absolute error (MAE) for the respective method, as output by the code. Since the underlying dataset is standard-normalized, the scale of the errors presented in the table is as well.

Table 1. Model fit on the dataset

Model	**RMSE**	**MAE**
SVRM	0.1090	0.01189
GPBoost	0.09516	0.009056
LSTM RNN	0.1517	0.1094
GAM	0.07625	0.005814

We find that the GAM method has the lowest errors and hence the closest fit, and the LSTM model the highest errors.

A chart visually illustrating the predictions from each respective model is in figure 6. One blip that can be clearly seen in our data comes from 2015. In that year, Ireland's economic output grew by a colossal 26.3% in one year. In this particular case, the most important and plausible explanation is a one-off change in accounting rules affecting how Ireland handles corporate taxes for large multinational companies, rather than any lasting change in Ireland's potential for long-term growth (OECD, 2016).

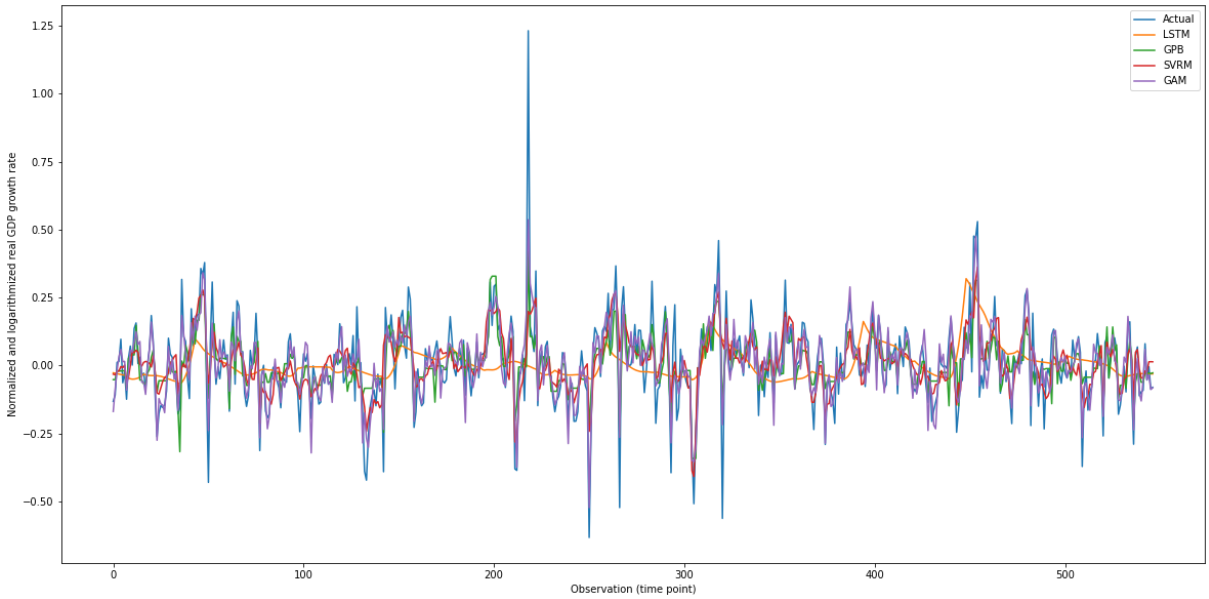

Figure 6a. The prediction of the SVRM, GPBoost and LSTM RNN model on the observed dataset versus the actual GDP growth. The vertical axis is the economic growth (logarithmized and normalized) and the horizontal axis is the time point. The blue line is the actual economic growth, the orange line is the predicted economic growth for LSTM RNN, red for SVRM, green for GPBoost and purple for GAM.

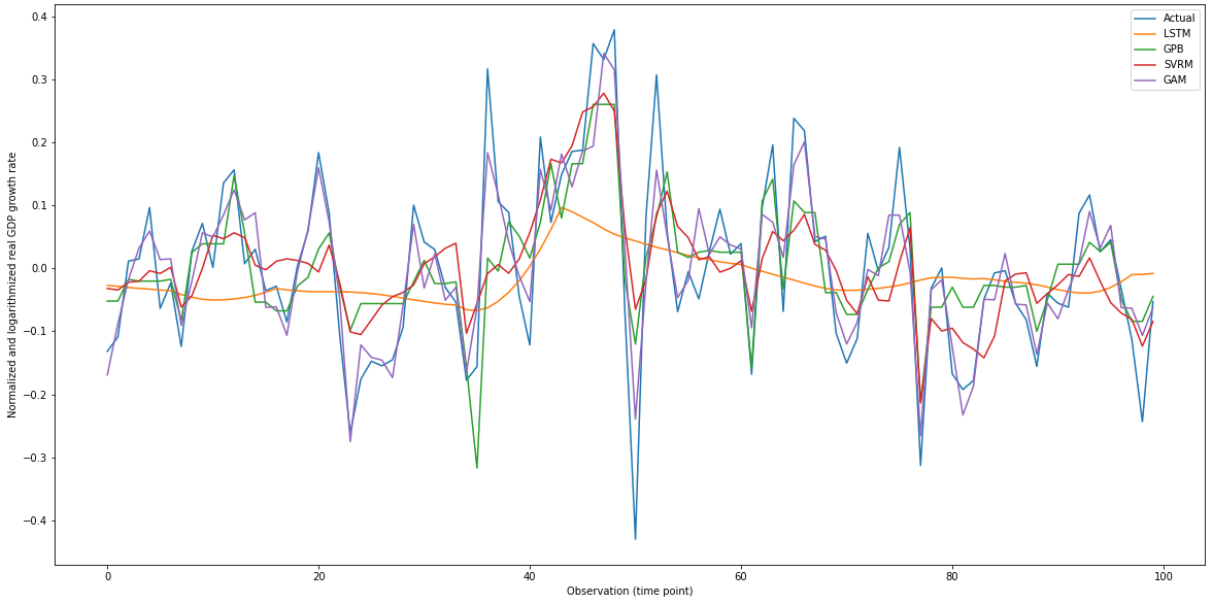

Figure 6b. In order to more clearly illustrate the differences between the various methods visually, the values in figure 6a are here given again, this time for the first 100 observations only.

Variable importance

The variable importance is our measure of the relative importance of the possible determinants of empirical historic economic growth. They are illustrated in figure 7, 8, 9 and 10 in two different ways. The first is mere the total variable importance, shown through the magnitude of impact on the algorithm by each variable and ranked from most important to least important by SHAP value estimation. The second is through the individual SHAP values and how each individual explanatory variable impact the model. For the LSTM RNN, figure 11 showcase the total variable importance and lastly, the significance of the variables in the GAM model is provided in figure 12 to indicate their significance in a more traditional linear sense.

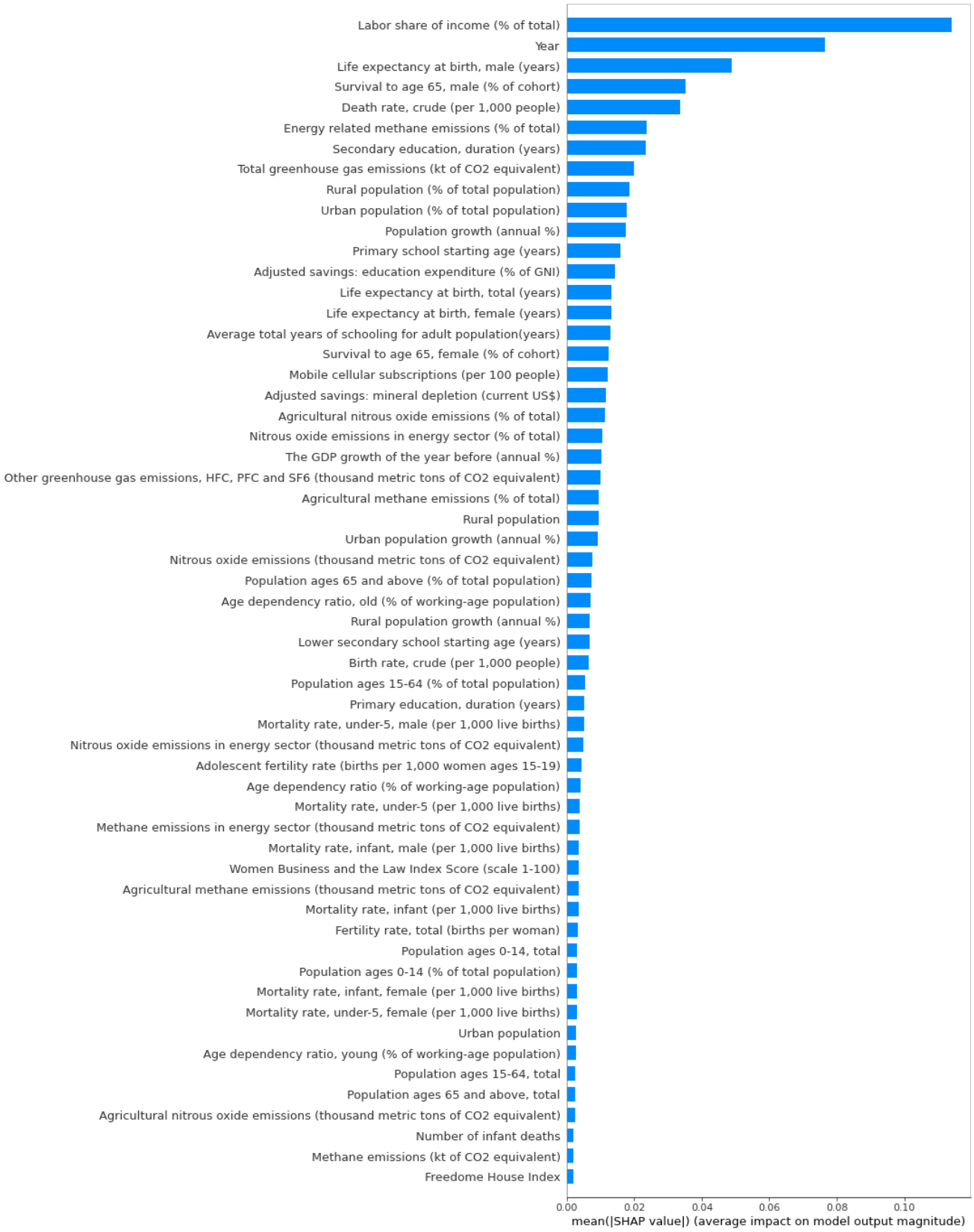

Figure 7. The variables are listed according to their variable importance in the SVRM model. The most important variable is labor share of income.

Figure 8. The variables are presented with their SHAP value. For the labor share of income variable, for example, a high number is associated with a negative impact on growth.

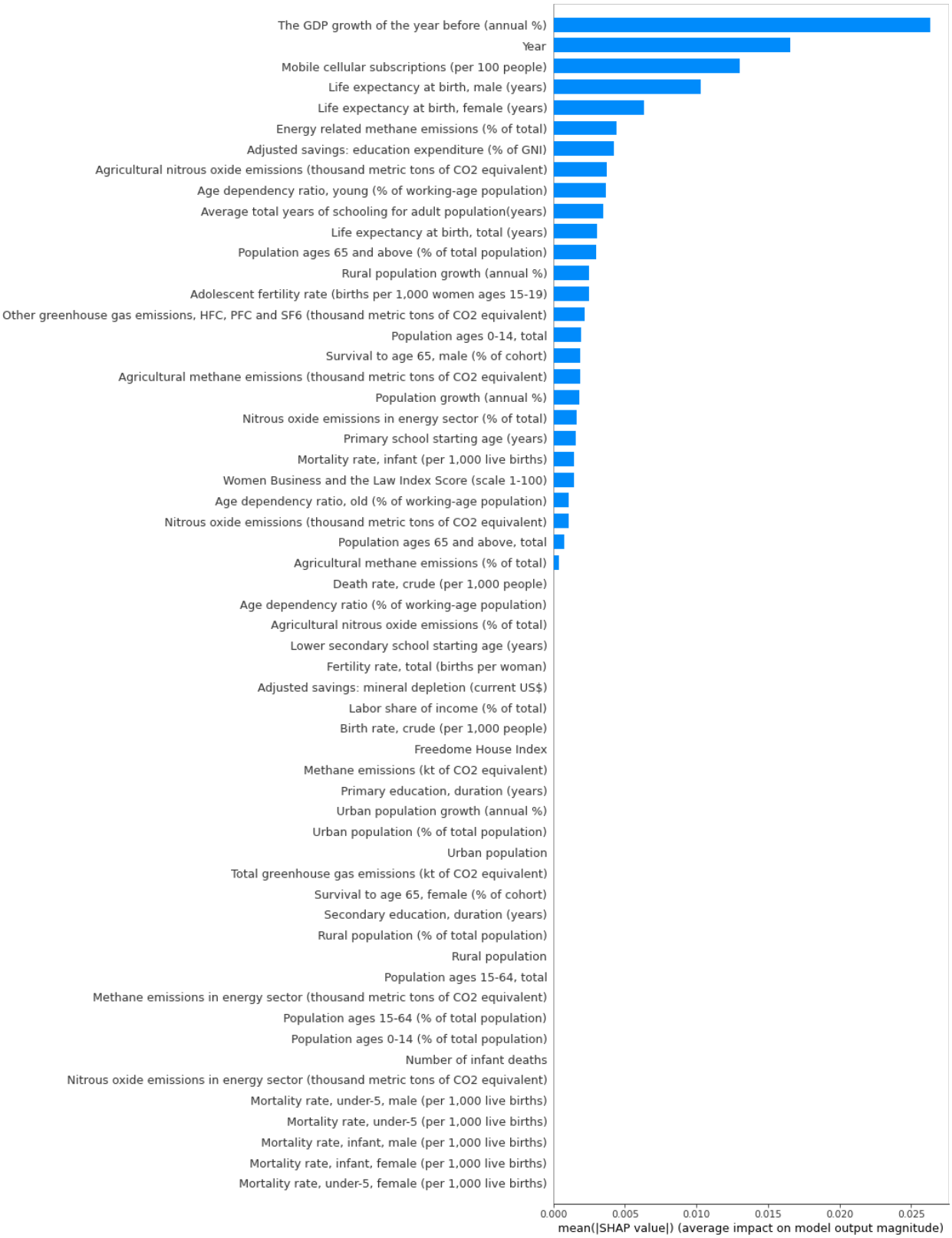

Figure 9. The variables are listed according to their variable importance in the GPBoost model. The most important variable is the growth rate in the previous year.

Figure 10. The variables are presented with their SHAP value in the GPBoost model. For the "previous growth" variable, for example, a high number is associated with a positive impact on growth.

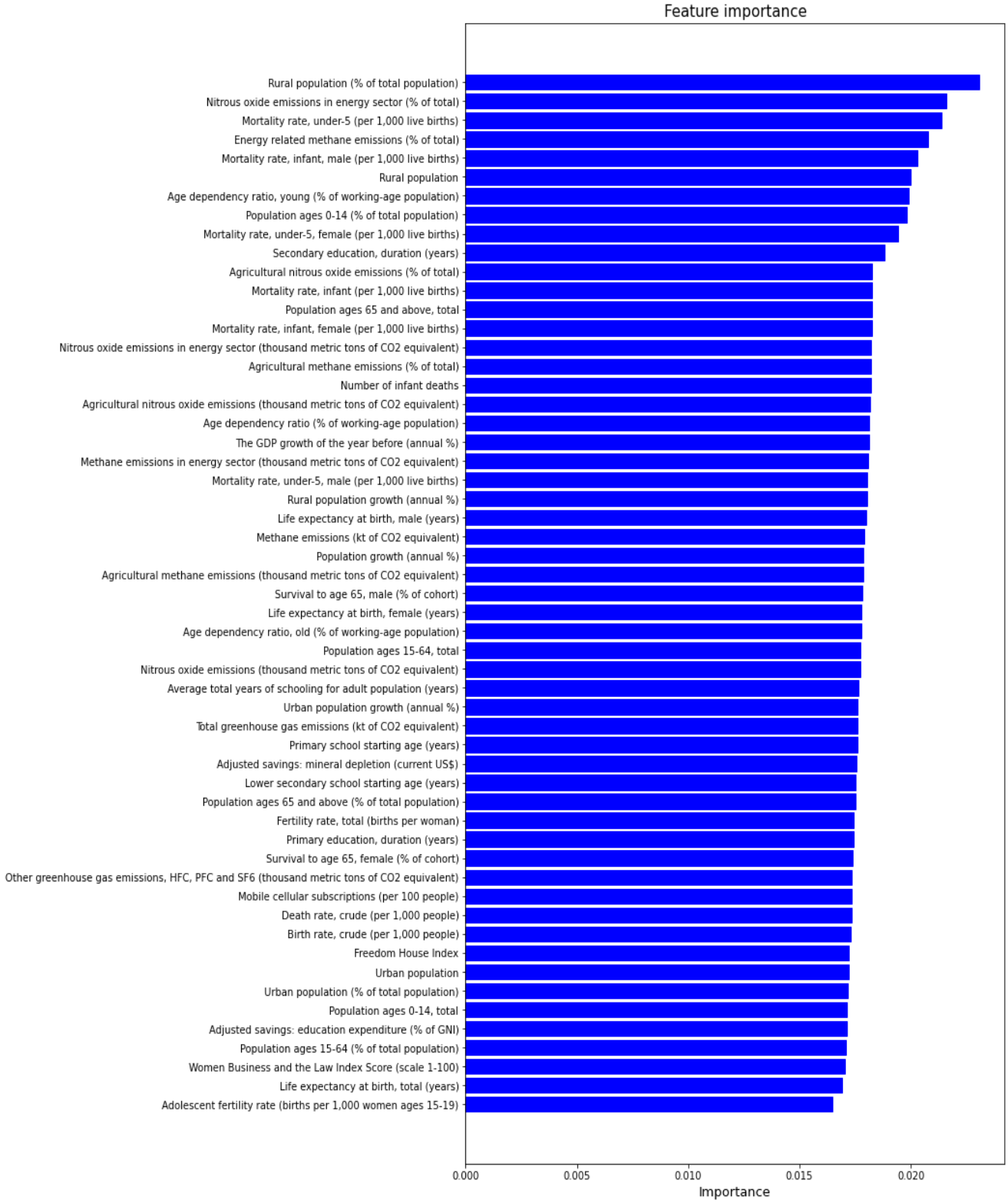

Figure 11. The variables are ranked according to their variable importance in the LSTM RNN model.

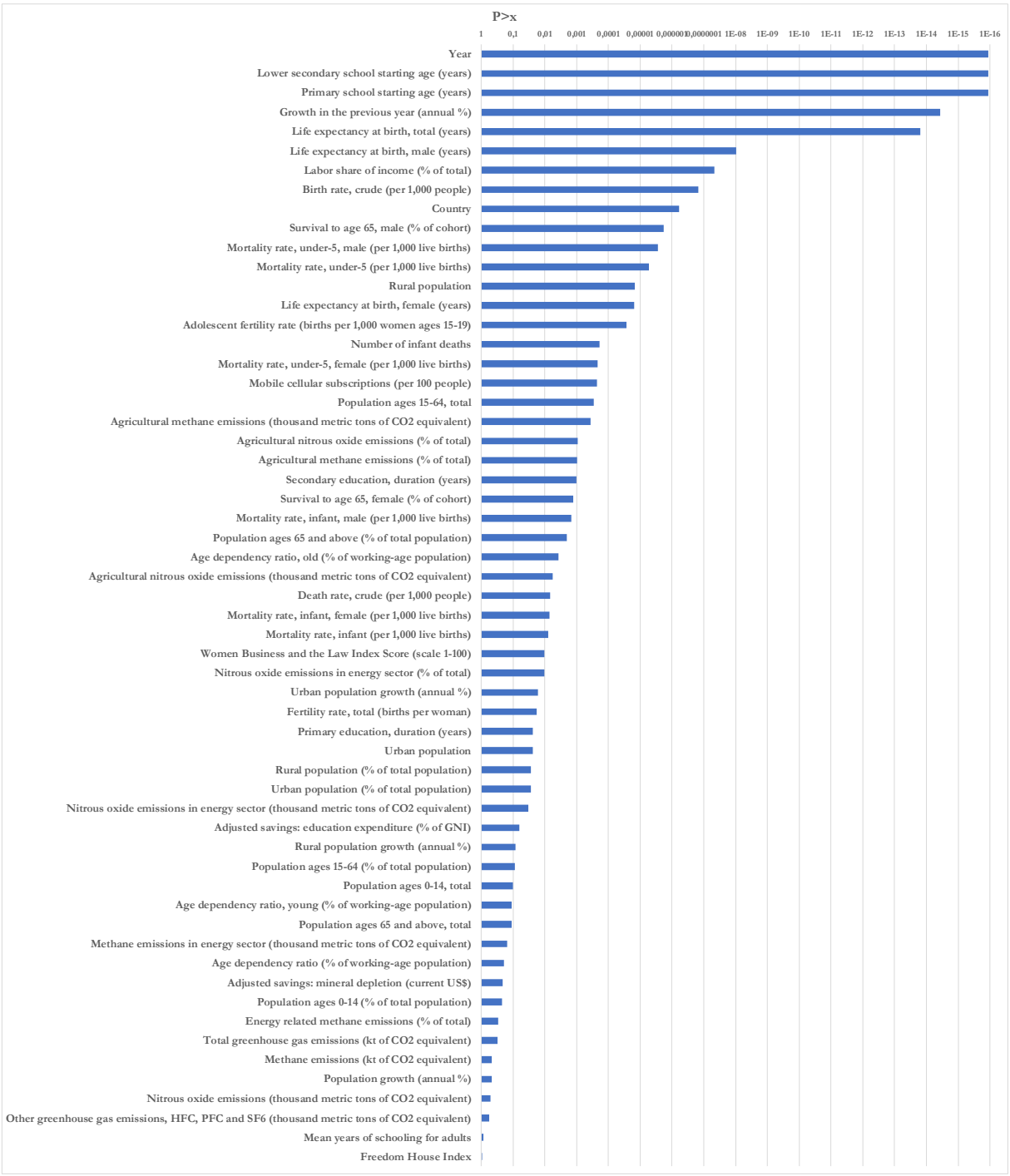

Figure 12. The significance probabilities of the independent variables in the GAM model.
The significance probabilities are given in E notation (for example, "1.11e-16" means 1.11×10^{-16}). Note that the bars show the inverse of the P-values (longer bars thus means lower p-values) and that the scale is logarithmic. Given the nature of the model care should be taken when interpreting these significance levels.

Predictive performance

The three machine learning algorithms were applied on the training set to fit an appropriate regression model, and then each model tried to predict the economic growth of the test or "unseen" dataset.

The results from our second scenario (models as a projection) are presented in table 3. It lists the RMSE and MAE for the respective method, showing that GPBoost algorithm has the lowest errors while SVRM has the highest. The predictions are illustrated in figure 12.

Table 3. Model performance on test dataset

Model	**RMSE**	**MAE**
SVRM	0.1789	0.1369
GPBoost	0.1465	0.02146
LSTM RNN	0.1578	0.09284
GAM	0.1648	0.09583

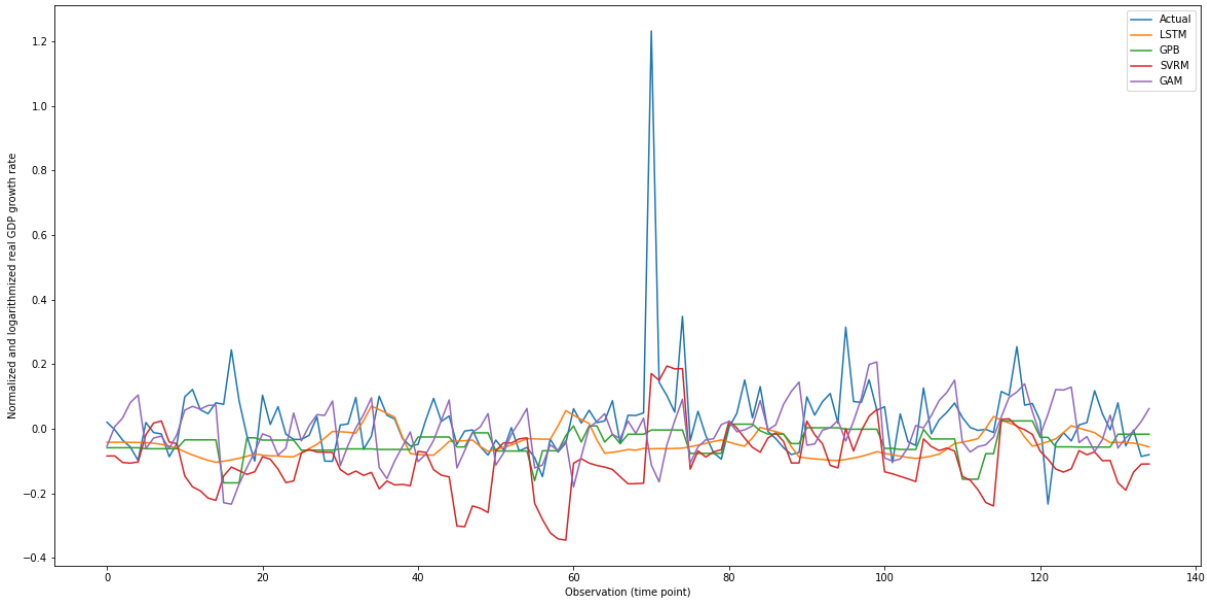

Figure 12. The prediction values for the evaluation subset of the data is given by the diagram. The horizontal axis is the time point and the vertical axis is the corresponding normalized growth rate. The blue line is the true values from the "unseen" test data, the orange line is the prediction from the LSTM method, the green from the GPBoost method, the red from the SVRM method, and the purple line is from the GAM method.

5. Discussion

In this thesis the possibility of using state-of-the-art machine learning algorithms to predict historic economic growth and determine the most important empirical factors for economic growth in such models are investigated. Given a limited dataset with only 28 European countries between the years of 1950-2019, we to various degrees successfully apply three machine learning algorithms, SVRM, GPBoost and LSTM RNN together with a generalized linear model to the dataset and develop predictive models. GPBoost produces a reasonable fit model and low RMSE that the SVRM and LSTM RNN cannot match. GAM seems to produce a more reliable fit of historical economic growth but is simultaneously underperforming when it comes to predicting growth. Intriguingly, the algorithms diverge quite heavily with regards to which variables are deemed most important to predict economic growth and in what way they inform the algorithms. Overall, the specific machine learning algorithms used in this analysis do not provide a convincing improvement over traditional methods.

Comparison of algorithms

The first, SVRM, had been among the best performing methods in several of the papers in our literature review. In our case, its performance was medium in the first scenario (with the entire dataset). In the second scenario, using "seen" data to predict "unseen" data, the performance of SVRM was the least accurate of all four methods. Due to the lack of possibility for intra-country comparison, the algorithm does not differentiate between the different countries but essentially treats all countries in the data set as one big country. This is a severe drawback which might explain the difference between its performance here and what we expected prior to running the algorithm and could possibly be amended through the use of a mixed model approach.

The second algorithm, GPBoost, is theoretically sound as it allows for nonparametric and nonlinear panel data to be analyzed in a mixed effects fashion, thus isolating the intra-country differences from the inter-country differences. In the first scenario, GPBoost was the second-best method, while in the second scenario, it was the best of all four. The primary reason for the lack of detail in the results is most likely due to the low number of observations. For each country there were only a few decades of data, which makes it hard for the algorithm to learn a function that might accurately predict the economic growth of the remaining years.

Similarly, the third method, LSTM RNN, has previously been shown to provide accurate predictions in time series, but in instances where the sample size was vastly larger (on the order of several thousand observations). One explanation for its comparatively low accuracy here can thus be an insufficiently large number of observations in our dataset. It is also likely that the algorithm needs more tuning so that it is customized to an environment with a smaller dataset. In our first scenario, LSTM was by far the least accurate of our four methods. In the second scenario, its performance was mediocre.

The fourth method, GAM, has the best performance of all four methods in the first scenario. In the second scenario, with predictions, its performance is mediocre and in par with the machine learning methods. In other words, it seems that the additive general linear buildup of the GAM is quite suitable for standard regressions or in scenarios when one would want to function to fit as closely as possible to a given data set, however this probably leads to a overfitting situation where the function perform bad on unobserved data. Overall however, it seems to be at least comparable with mathematically significantly more complex machine learning algorithms.

Results compared with previous literature

Ciccone & Jarociński had investigated determinants of growth by being agnostic about which variables matter for growth. They found that conclusions are highly sensitive to minor errors in measurement and to some degree vary across methods applied (Ciccone and Jarociński, 2010). The results from this analysis strengthens this caveat when trying to apply machine learning methods on these types of data sets.

In our case, using machine learning methods yields results which diverge dramatically from one another in terms of what variables are most important for economic growth. For the SVRM algorithm, the five most important factors were labor's share of income, life expectancy at birth, total greenhouse gas emissions, years of secondary education, and the death rate. For the GPBoost algorithm, the five most important variables were instead growth in the previous year, the year itself, mobile cellular subscriptions per 100 people, and life expectancy at birth. With the LSTM RNN algorithm, the most important variables are energy related methane emissions, the under-5 mortality rate, the rural population as percentage of total population, and the amount of nitrous oxide emissions in the energy sector. While there was some overlap between these three, their divergence is notable. In part, the divergence might simply be explained by the different ways in which the machine learning methods function and the applied parameter tuning. However, more

importantly, the divergence indicates that if one had solely relied on a single machine learning method, it might have led us to conclusions with a relatively low prediction error, but lacking in robustness. A general tendency of the data was similar for the GAM model, with a range of variables significantly contributing to the additive model. The top-ranking variables in terms of variable importance significantly contributed to the GAM model, adding some evidence that the variables do seem to be determinants of economic growth in the European region.

Comparing how the variables might be predicting economic growth through SHAP values for the SVRM and GPBoost models with the results from Barro, one can find the same signs of the effects (for example, a *lower* fertility rate is associated with *higher* growth), but not the same ranking in terms of which variables that might be most influential. As such, the way in which the algorithms use the explanatory variables to make predictions are in line with previous research even though the coefficient of rank differs.

Looking further into our SHAP values yields some interesting results. Some of them are, at first glance, counterintuitive. For example, in the SVRM algorithm the higher the country's life expectancy at birth, the lower its growth rate. In cases like this, it seems the most natural explanation is growth convergence. That is, poorer countries (with lower life expectancy) grow faster because they are poor, not because their life expectancy is low (Islam, 1995). The latter is not, by itself, a causal determinant of growth. Another seemingly counterintuitive example in the SVRM model is the variable "Adjusted savings: education expenditure (% of GNI)". The SHAP values indicate that whether this variable is high or low seems to have no unambiguous effect on growth in one clear direction. In this case it could be because the positive effect from increasing education expenditure only manifests itself after a certain time-lag (Chandra and Islamia, 2010) which we did not include as a variable in the dataset.

Lastly, predicting economic growth, even with historical data, is notoriously hard (Coulombe et al., 2020). The machine learning methods and variables included in this analysis does not provide a satisfactory prediction of the GDP growth, however it is hard to make a full judgment of the prediction since we did not include any type of benchmark prediction. It has been shown that machine learning methods can outperform standard GDP growth forecasts used by central banks (Yoon, 2021). In our case, it is possible that one would need more specified variable selection methods and adjusting parameters for the purpose of forecasting in order to get more reasonable predictions.

Limitations

There are several limitations to this analysis, some of which have been touched upon previously. The most prominent is the availability of data. Despite what one might assume, we actually have rather few variables with comparable and quality data before 1990. Given this shortage, machine learning methods are essentially working with a too small sample size to fully recognize patterns and form reliable variable importance measures. The second one is the interpretability of the methods, as there are currently no fully integrated mathematical and practical application of SHAP Values for LSTM RNNs and ordinary linear models. Although we are rather certain that the variable importance would be the same for the LSTM RNNs if measured through SHAP Values, we cannot derive the individual variable SHAP values and compare with SVRM and GPBoost, which severely limits the possibility for a fair comparison. Lastly, tuning the machine learning parameters can heavily affect the fit of the function, predictive performance and variable importance measures. Selecting a standard value for critical parameters (for example epsilon) and through automating the process of parameter value selection through a 4-fold cross validation parameter search similar for the three algorithms, we hoped to strike a balance between reproducibility and accuracy.

Future studies

There are several potential venues for future research and issues that would benefit from further inquiry. With regards to the empirical aspects of this paper, focusing on a smaller region of countries for a longer period could perhaps allow the machine learning algorithms to be used to full advantage and would most probably result in different findings. An extension of machine learning algorithms that can work for smaller data sets more prevalent within empirical economics should be a high priority, since most methods falter with less than 100 observations. It would also be beneficial to include as a comparison a more time sensitive yet parametric method, such as an autoregressive model. The context specificity inherent in following a data-driven and machine learning approach is an advantage, but to what extent the results can be generalized beyond the studied data set should be carefully examined. Since it seems that economic relationships in low income countries or high economic growth countries follow an even more nonlinear and almost chaotic pattern, an application of machine learning in these settings to determine the factors for economic growth could yield important understandings and policy options. It has also become clear that machine learning is not a silver bullet which can be used on any type of data and in any way and produce extraordinary results, within empirical economics the theoretical and practical limitations of machine learning (and the extension of it - AI) should be further researched.

5. Conclusions

The empirical investigation of the determinants of economic growth can be updated and revisited through the availability of state-of-the-art machine learning algorithms, however much work remains. In our analysis of 28 European countries between 1950-2019 they do not provide a compelling improvement over traditional methods. Specifically, we showcase that Support Vector Regression Machines, Gaussian Process Boosting and Long Short-Term Memory Neural Networks diverge in their prediction patterns and variable importance estimates, making it hard to make conclusions regarding the possible empirical determinants of economic growth. More data and research on a range of issues is needed before the full potential of machine learning within empirical economics can be harnessed.

6. References

Acemoglu, D. *et al.* (2019) 'Democracy Does Cause Growth', *Journal of Political Economy*, 127(1), pp. 47–100. doi: 10.1086/700936.

Albawi, S., Mohammed, T. A. and Al-Zawi, S. (2018) 'Understanding of a convolutional neural network', *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, 2018-Janua, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

An, J., Mikhaylov, A. and Moiseev, N. (2019) 'Oil price predictors: Machine learning approach', *International Journal of Energy Economics and Policy*, 9(5), pp. 1–6. doi: 10.32479/ijeep.7597.

Athey, S. and Imbens, G. W. (2019) 'Machine Learning Methods That Economists Should Know About', *Annual Review of Economics*, 11(1), pp. 685–725. doi: 10.1146/annurev-economics-080217-053433.

Barro, R. and J.-W. L. (2013) 'A New Data Set of Educational Attainment in the World, 1950-2010', *Journal of Development Economics*, 104, pp. 184–198. doi: 10.1016/j.jdeveco.2012.10.001.

Barro, R. J. (2003) *Determinants of Economic Growth in a Panel of Countries*, *Annals of Economics and Finance*. Available at: http://aeconf.com/Articles/Nov2003/aef040202.pdf (Accessed: 9 February 2021).

Beyca, O. F. *et al.* (2019) 'Using machine learning tools for forecasting natural gas consumption in the province of Istanbul', *Energy Economics*, 80, pp. 937–949. doi: 10.1016/j.eneco.2019.03.006.

Breiman, L. (2001) 'Random Forests', *Machine Learning*, 45(1), pp. 5–32. doi: 10.1023/A:1010933404324.

Cass, D. (1965) 'Optimum Growth in an Aggregative Model of Capital Accumulation', *The Review of Economic Studies*, 32(3), p. 233. doi: 10.2307/2295827.

Chandra, A. and Islamia, J. M. (2010) 'Does Government Expenditure on Education Promote Economic Growth? An Econometric Analysis', *MPRA Paper No. 25480*, (25480). Available at: http://mpra.ub.uni-muenchen.de/25480/1/MPRA_paper_25480.pdf.

Chen, Y., He, K. and Tso, G. K. F. (2017) 'Forecasting Crude Oil Prices: A Deep Learning based Model', in *Procedia Computer Science*. Elsevier B.V., pp. 300–307. doi: 10.1016/j.procs.2017.11.373.

Chuku, C., Simpasa, A. and Oduor, J. (2019) 'Intelligent forecasting of economic growth for developing economies', *International Economics*, 159, pp. 74–93. doi: 10.1016/j.inteco.2019.06.001.

Cicceri, G., Inserra, G. and Limosani, M. (2020) 'A Machine Learning Approach to Forecast Economic Recessions—An Italian Case Study', *Mathematics*, 8(2), p. 241. doi: 10.3390/math8020241.

Ciccone, A. and Jarociński, M. (2010) 'Determinants of economic growth: Will data tell?',

American Economic Journal: Macroeconomics, 2(4), pp. 222–246. doi: 10.1257/mac.2.4.222.
Coulombe, P. G. *et al.* (2020) 'How is Machine Learning Useful for Macroeconomic Forecasting?', *arXiv*. Available at: http://arxiv.org/abs/2008.12477 (Accessed: 9 February 2021).
Cuaresma, J. C., Doppelhofer, G. and Feldkircher, M. (2014) 'The Determinants of Economic Growth in European Regions', *Regional Studies*, 48(1), pp. 44–67. doi: 10.1080/00343404.2012.678824.
Drucker, H. *et al.* (1997) 'Support vector regression machines', *Advances in Neural Information Processing Systems*, 1, pp. 155–161. Available at: https://www.researchgate.net/profile/Harris-Drucker/publication/309185766_Support_vector_regression_machines/links/5b0053e5a6fdccf9e4f5689b/Support-vector-regression-machines.pdf.
Durlauf, S. N. and Quah, D. T. (1998) 'The new empirics of economic growth', *National Bureau of Economic Research*. Available at: http://www.nber.org/papers/w6422.
Elith, J., Leathwick, J. R. and Hastie, T. (2008) 'A working guide to boosted regression trees', *Journal of Animal Ecology and Economt*, 77(4), pp. 802–813. doi: 10.1111/j.1365-2656.2008.01390.x.
Freedom House (2021) *Freedom House Index*, *Country and Territory Ratings and Statuses, 1973-2021*. Available at: https://freedomhouse.org/reports/publication-archives (Accessed: 25 February 2021).
Gabralla, L. A., Jammazi, R. and Abraham, A. (2013) 'Oil price prediction using ensemble machine learning', in *Proceedings - 2013 International Conference on Computer, Electrical and Electronics Engineering: 'Research Makes a Difference', ICCEEE 2013*, pp. 674–679. doi: 10.1109/ICCEEE.2013.6634021.
Gallup, J. L., Sachs, J. D. and Mellinger, A. D. (1998) 'Geography and economic development', *National Bureau of Economic Research*. Available at: http://www.nber.org/papers/w6849.pdf.
Gogas, P. *et al.* (2015) 'Yield Curve and Recession Forecasting in a Machine Learning Framework', *Computational Economics*, 45(4), pp. 635–645. doi: 10.1007/s10614-014-9432-0.
Graves, A. (2012) 'Long Short-Term Memory', *Supervised Sequence Labelling with Reccurent Neural Networks*, pp. 37–45. doi: 10.1007/978-3-642-24797-2_4.
Groningen Growth and Development Centre (2021) *Penn World Table*. Available at: https://www.rug.nl/ggdc/productivity/pwt/ (Accessed: 15 February 2021).
Guo, T., Lin, T. and Antulov-Fantulin, N. (2019) 'Exploring interpretable LSTM neural networks over multi-variable data', *36th International Conference on Machine Learning, ICML 2019*, 2019-June, pp. 4424–4440. Available at: http://proceedings.mlr.press/v97/guo19b.html.
Hastie, T. and Tibshirani, R. (1985) 'Generalized Additive Models: Some Applications', *Journal of the American Statistical Association*, 82(398), pp. 371–386. doi: 10.1080/01621459.1987.10478440.

Hastie, T. and Tibshirani, R. (1991) 'Generalized Addative Models', *Statistical Science*, 6(1), pp. 15–51. Available at: http://www.jstor.org/stable/2245459.

Islam, N. (1995) 'Growth Empirics: a Panel Data Approach', *The Quarterly Journal of Economics,* (November), pp. 1127–1170. doi: 10.2307/2946651.

Johnson, R. and Zhang, T. (2013) 'Learning nonlinear functions using regularized greedy forest.', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5), pp. 942–954. doi: 10.1109/TPAMI.2013.159.

Jones, C. I. (2016) *Handbook of Macroeconomics*. 1st edn. Elsevier B.V. doi: 10.1016/bs.hesmac.2016.03.002.

Kuan, C. M. and White, H. (1994) 'Artificial neural networks: An econometric perspective', *Econometric Reviews*, 13(1), pp. 1–91. doi: 10.1080/07474939408800273.

Larsen, K. (2015) *GAM: The Predictive Modeling Silver Bullet*. Available at: https://multithreaded.stitchfix.com/assets/files/gam.pdf.

Lee, J.-W. and H. L. (2016) 'Human Capital in the Long Run', *Journal of Development Economics*, 122, pp. 147–169. doi: 10.1016/j.jdeveco.2016.05.006.

Lee, T. H., White, H. and Granger, C. W. J. (1993) 'Testing for neglected nonlinearity in time series models. A comparison of neural network methods and alternative tests', *Journal of Econometrics*, 56(3), pp. 269–290. doi: 10.1016/0304-4076(93)90122-L.

Levine, R. and Renelt, D. (1992) 'A sensitivity analysis of cross-country growth regressions', *American Economic Review*, 82(4), pp. 942–963. doi: 10.2307/2117352.

Lundberg, S. M. and Lee, S. I. (2017) 'A unified approach to interpreting model predictions', *arXiv*, (Section 2), pp. 1–10. Available at: https://arxiv.org/abs/1705.07874.

Maddison, A. (2021) *Maddison Historical Statistics*. Available at: https://www.rug.nl/ggdc/historicaldevelopment/maddison/.

Milačić, L. *et al.* (2017) 'Application of artificial neural network with extreme learning machine for economic growth estimation', *Physica A: Statistical Mechanics and its Applications*, 465, pp. 285–288. doi: 10.1016/j.physa.2016.08.040.

Molnar, C. (2021) *Interpretable Machine Learning*. Available at: https://christophm.github.io/interpretable-ml-book/ (Accessed: 28 February 2021).

Moral-Benito, E. (2012) 'Determinants of economic growth: A bayesian panel data approach', *Review of Economics and Statistics*, 94(2), pp. 566–579. doi: 10.1162/REST_a_00154.

Mullainathan, S. and Spiess, J. (2017) 'Machine learning: An applied econometric approach', in *Journal of Economic Perspectives*. American Economic Association, pp. 87–106. doi: 10.1257/jep.31.2.87.

Müller, K. R. *et al.* (1997) 'Predicting time series with support vector machines', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1327(x), pp. 999–1004. doi: 10.1007/bfb0020283.

OECD (2016) 'Irish GDP up by 26.3% in 2015?', *Oecd*, 2014(October), pp. 2014–2017. Available at: https://www.oecd.org/std/na/Irish-GDP-up-in-2015-OECD.pdf.

Panhans, M. T. and Singleton, J. D. (2015) 'The Empirical Economist's Toolkit: From Models to Methods', *SSRN Electronic Journal*, (October), pp. 1–27. doi: 10.2139/ssrn.2611236.

Pedregosa, F. *et al.* (2011) 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, 12, pp. 2825–2839. Available at: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

Ramsey, F. P. (1928) 'A Mathematical Theory of Saving', *The Economic Journal*, 38(152). doi: 10.2307/2224098.

Romer, P. M. (1990) 'Endogenous Technological Change', *Journal of Political Economy*, 98(5, Part 2). doi: 10.1086/261725.

Sala-i-Martin, X., Doppelhofer, G. and Miller, R. I. (2004) 'Determinants of long-term growth: a bayesian averaging of classical estimates (BACE) approach', *The American Economic Review*, 94(4), pp. 813–835. Available at: https://www.jstor.org/stable/3592794.

Sigrist, F. (2020) 'Gaussian process boosting', *arXiv*, pp. 1–40. Available at: https://arxiv.org/abs/2004.02653.

Sokolov-Mladenović, S. *et al.* (2016) 'Economic growth forecasting by artificial neural network with extreme learning machine based on trade, import and export parameters', *Computers in Human Behavior*, 65, pp. 43–45. doi: 10.1016/j.chb.2016.08.014.

Solow, R. M. (1956) 'A Contribution to the Theory of Economic Growth', *The Quarterly Journal of Economics*, 70(1). doi: 10.2307/1884513.

Sun, S. *et al.* (2019) 'Forecasting tourist arrivals with machine learning and internet search index', *Tourism Management*, 70, pp. 1–10. doi: 10.1016/j.tourman.2018.07.010.

Swan, T. W. (1956) 'Economic Growth and Capital Accumulation', *Economic Record*, 32(2), pp. 334–361. doi: 10.1111/j.1475-4932.1956.tb00434.x.

Swanson, N. R. and White, H. (1997) 'A model selection approach to real-time macroeconomic forecasting using linear models and artificial neural networks', *Review of Economics and Statistics*, 79(4), pp. 540–550. doi: 10.1162/003465397557123.

Ülke, V., Sahin, A. and Subasi, A. (2018) 'A comparison of time series and machine learning models for inflation forecasting: empirical evidence from the USA', *Neural Computing and Applications*, 30(5), pp. 1519–1527. doi: 10.1007/s00521-016-2766-x.

Varian, H. R. (2014) 'Big data: New tricks for econometrics', *Journal of Economic Perspectives*, 28(2), pp. 3–28. doi: 10.1257/jep.28.2.3.
World Bank (2021) *World Development Indicators*. Available at: https://databank.worldbank.org/source/world-development-indicators (Accessed: 20 February 2021).
Xie, G., Qian, Y. and Wang, S. (2021) 'Forecasting Chinese cruise tourism demand with big data: An optimized machine learning approach', *Tourism Management*, 82, p. 104208. doi: 10.1016/j.tourman.2020.104208.
Yoon, J. (2021) 'Forecasting of Real GDP Growth Using Machine Learning Models: Gradient Boosting and Random Forest Approach', *Computational Economics*, 57, pp. 247–265. doi: 10.1007/s10614-020-10054-w.
Yu, L., Dai, W. and Tang, L. (2016) 'A novel decomposition ensemble model with extended extreme learning machine for crude oil price forecasting', *Engineering Applications of Artificial Intelligence*, 47, pp. 110–121. doi: 10.1016/j.engappai.2015.04.016.

7. Appendices

In the following appendices the data preparation as well as the source code for the machine learning algorithms used are presented.

Appendix A – Preparation of data (prescript.py)

```
import csv
import pandas as pd
import numpy as np
import time

# add dummy variable that is 1 for those in training set and 0 for test set
def add_training_set_variable(df,lander,fraction):
    df['training_set']=0
    for land in lander:
        rader=list(df[df.countrycode==land].index)
        test_rader=rader[0:int(fraction*len(rader))+1]
        df.iloc[test_rader, df.columns.get_loc('training_set')]=1
    return df

for scenariotype in ['nordic','EU']:

    print("==== [Scenario: "+scenariotype+ "] ====")

    # set of countries to use

    if scenariotype=='EU':
        country_code = {
            "Austria": "AUT",
            "Belgium": "BEL",
            "Bulgaria": "BGR",
            "Cyprus": "CYP",
            "Czech Republic": "CZE",
            "Germany": "DEU",
            "Denmark": "DNK",
            "Spain": "ESP",
            "Estonia": "EST",
            "Finland": "FIN",
            "France": "FRA",
            "Greece": "GRC",
            "Croatia": "HRV",
            "Hungary": "HUN",
            "Ireland": "IRL",
            "Italy": "ITA",
            "Lithuania": "LTU",
            "Luxembourg": "LUX",
            "Latvia": "LVA",
            "Malta": "MLT",
            "Netherlands": "NLD",
            "Poland": "POL",
            "Portugal": "PRT",
            "Romania": "ROU",
            "Slovakia": "SVK",
            "Slovenia": "SVN",
            "Sweden": "SWE"
            }
    if scenariotype=='nordic':
        country_code = {
            "Sweden": "SWE",
            "Norway": "NOR",
            "Denmark": "DNK",
            "Finland": "FIN",
            "Iceland": "ISL"
            }

    countries = sorted(country_code.values())

    # set of years to do

    if scenariotype=='EU':
        firstyear=1950
        lastyear=2019
    if scenariotype=='nordic':
        firstyear=1950
        lastyear=2019
```

```
    # missing values threshold (0.8 = 80 percent)
    # remove columns from data whose share of values missing is higher than this

    missing_values_threshold=0.5

    # percentage training versus test

    training_percentage=0.8

    # create a main dataframe

    df=pd.DataFrame()

    numyears=len(range(firstyear,lastyear+1))

    for land in country_code.values():
        for year in range(firstyear,lastyear+1):
            df=pd.concat([df,pd.DataFrame({"countrycode":[land],"year":[year]})])

    # Penn-World Tables from Excel to csv

    print(time.ctime()+": Adding Penn-World Tables")
    pwt100=pd.read_excel('indata/pwt100.xlsx', sheet_name=2)

    # add gdp per capita, as well as growth

    pwt100['rgdpe_per_capita']=pwt100['rgdpe']/pwt100['pop']
    pwt100['growth']=pwt100['rgdpe_per_capita'].pct_change()
    pwt100['previous_growth']=pwt100['rgdpe_per_capita'].pct_change().shift(1)

    df=pd.merge(df,pwt100,how="left",on=['countrycode','year'])

    # WDI

    print(time.ctime()+": Starting to parse WDIEXCEL now")
    wdi_table=pd.read_excel('indata/WDIEXCEL.xlsx', sheet_name=0)
    print(time.ctime()+": Finished parsing WDIEXCEL now")
    wdi_indicators=wdi_table['Indicator Code'].unique()
    land_df=pd.DataFrame()
    list_df=[]
    for land in countries:
        print(time.ctime()+": [WDI] Adding country "+land)
        wdi_df=wdi_table[wdi_table['Country Code'].isin([land])]
        for year in range(firstyear,lastyear+1):
            dict_df={"countrycode":[land],"year":[year]}
            for indicator in wdi_indicators:
                try:
                    dict_df[indicator]=wdi_df.loc[wdi_df['Indicator Code'] == indicator][str(year)].values[0]
                except: # except KeyError?
                    dict_df[indicator]=np.nan
            list_df.append(pd.DataFrame(dict_df))
    land_df=pd.concat(list_df, axis=0)
    df=pd.merge(df,land_df,how="left",on=['countrycode','year'])

    # drop WDI indicators with too many missing values
    columns_to_be_kept = list(df.columns.values)
    print(time.ctime()+": Number of columns before dropping: "+str(len(df.columns)))
    for land in countries:
        temp_df=df[df['countrycode'].isin([land])]
        limitPer=missing_values_threshold*len(temp_df)
        temp_df=temp_df.dropna(thresh=limitPer, axis=1)
        columns_to_be_kept        =        sorted(set(columns_to_be_kept)        &        set(temp_df.columns.values),
key=columns_to_be_kept.index)
    df=df[columns_to_be_kept]
    print(time.ctime()+": Number of columns after dropping: "+str(len(df.columns)))

    # Average total years of schooling

    schooling=pd.read_csv('indata/mean-years-of-schooling-long-run.csv', dtype={'Year':'int'})
    land_df=pd.DataFrame()
    for land in country_code.values():
        school_df=schooling[schooling.Code.isin([land])]
        for year in range(firstyear,lastyear+1):
            try:
                school_value=school_df[school_df.Year.isin([year])].values[0][3]

land_df=pd.concat([land_df,pd.DataFrame({"countrycode":[land],"year":[year],"schooling":[school_value]})])
            except (IndexError, KeyError):
                land_df=pd.concat([land_df,pd.DataFrame({"countrycode":[land],"year":[year],"schooling":[np.nan]})])
    df=pd.merge(df,land_df,how="left",on=['countrycode','year'])

    # Freedom House

    freedomhouse=pd.read_excel('indata/Country_and_Territory_Ratings_and_Statuses_FIW1973-2021.xlsx', sheet_name=1)
```

```
    freedomvalue = {"NF": 1, "PF": 2, "F": 3}
    #land_df=pd.DataFrame()
    list_df=[]
    for land in country_code.keys():
        freedomhouse_df=freedomhouse.loc[freedomhouse['Survey Edition']==land]
        for year in range(firstyear,lastyear+1):
            dict_df={}
            try:
                if year<1982: # in order to handle the fact that both years 1981 and 1982 are in the same report
                    rowindex = (year-1971)*3
                else:
                    rowindex = (year-1972)*3
                freedomhouseindex=freedomvalue[freedomhouse.loc[freedomhouse['Survey
Edition']==land].values[0][rowindex]]

#land_df=pd.concat([land_df,pd.DataFrame({"countrycode":[country_code[land]],"year":[year],"freedomhouse":[freedomhou
seindex]})])
                dict_df={"countrycode":[country_code[land]],"year":[year],"freedomhouse":[freedomhouseindex]}
            except:

#land_df=pd.concat([land_df,pd.DataFrame({"countrycode":[country_code[land]],"year":[year],"freedomhouse":[np.nan]})]
)
                dict_df={"countrycode":[country_code[land]],"year":[year],"freedomhouse":[np.nan]}
            #print(dict_df)
            list_df.append(pd.DataFrame(dict_df))
    land_df=pd.concat(list_df, axis=0)
    df=pd.merge(df,land_df,how="left",on=['countrycode','year'])

    # drop unnecessary variables

    # first, remove specific list
    df=df.drop([val for val in ["currency_unit", "country", "i_cig", "i_xm", "i_xr", "i_outlier",  "i_irr"] if val in
list(df.columns)], axis=1)
    # second, remove all that are "GDP per capita"
    df=df.drop([s for s in list(df.columns) if "GDP.PCAP" in s], axis=1)
    # third, remove the "IT.CEL.SETS" variable as the "IT.CEL.SETS.P2" is sufficient
    df=df.drop([val for val in ["IT.CEL.SETS"] if val in list(df.columns)], axis=1)

    df=add_training_set_variable(df,countries,training_percentage) # add dummy

    for nopopulation in [False, True]:
        # output raw data

        pop_or_nopop = ""

        if nopopulation:
            pop_or_nopop = "_nopop"
            df=df.drop([s for s in list(df.columns) if "SP.POP.TOTL" in s], axis=1)
            df=df.drop([s for s in list(df.columns) if ("SP.POP" in s) and (".MA." in s)], axis=1)
            df=df.drop([s for s in list(df.columns) if ("SP.POP" in s) and (".FE." in s)], axis=1)
            print("=== [case: no population] ===")

        print(time.ctime()+": Outputting raw data")
        df.to_excel("output_"+scenariotype+"_raw"+pop_or_nopop+".xlsx")
        df.to_csv("output_"+scenariotype+"_raw"+pop_or_nopop+".csv")

        # output normalized raw data
        df_normalized=df.copy()
        # freedom house is often identical for all observations, thus standard deviation is zero, thus normalization
is division by zero
        no_variation_test = df.std()==0
        for kolonn in df.columns:
            try:
                if no_variation_test[kolonn]:
                    df_normalized=df_normalized.drop(columns=[kolonn]) # drop columns that show no variation
            except KeyError:
                continue

        #column_names_to_not_normalize = ['countrycode', 'year', 'country', 'currency_unit', 'i_cig', 'i_xm', 'i_xr',
'i_outlier', 'i_irr']
        column_names_to_not_normalize = ['countrycode', 'year', 'training_set']
        column_names_to_normalize = [x for x in list(df_normalized) if x not in column_names_to_not_normalize ]
        x = df_normalized[column_names_to_normalize]
        x_scaled = (x-x.mean())/x.std()
        df_temp = pd.DataFrame(x_scaled, columns=column_names_to_normalize, index = df_normalized.index)
        df_normalized[column_names_to_normalize] = df_temp

        df_normalized=add_training_set_variable(df_normalized,countries,training_percentage) # add dummy

        df_normalized.to_excel("output_"+scenariotype+"_raw_normalized"+pop_or_nopop+".xlsx")
        df_normalized.to_csv("output_"+scenariotype+"_raw_normalized"+pop_or_nopop+".csv")

        # output interpolated data

        print(time.ctime()+": Outputting interpolated data")
```

```
        with       pd.ExcelWriter("output_"+scenariotype+"_interpolated"+pop_or_nopop+".xlsx")       as       writer1,
pd.ExcelWriter("output_"+scenariotype+"_interpolated_normalized"+pop_or_nopop+".xlsx") as writer2:
            for interpol_method in ['linear']: # ['linear','quadratic','cubic','krogh']

                # output not yet normalized data
                df_interpol = pd.DataFrame()
                for land in country_code.values():

df_interpol=pd.concat([df_interpol,df.loc[df['countrycode'].isin([land])].interpolate(method=interpol_method)])
                df_interpol=df_interpol.reset_index(drop=True)
                df_interpol=add_training_set_variable(df_interpol,countries,training_percentage) # add dummy
                df_interpol.to_excel(writer1,sheet_name=interpol_method)
                df_interpol.to_csv("output_"+scenariotype+"_interpolated_"+interpol_method+pop_or_nopop+".csv")

                # output normalized imputed data
                df_normalized_interpol = pd.DataFrame()
                for land in country_code.values():

df_normalized_interpol=pd.concat([df_normalized_interpol,df_normalized.loc[df['countrycode'].isin([land])].interpolat
e(method=interpol_method)])
                df_normalized_interpol=df_normalized_interpol.reset_index(drop=True)

df_normalized_interpol=add_training_set_variable(df_normalized_interpol,countries,training_percentage) # add dummy
                df_normalized_interpol.to_excel(writer2,sheet_name=interpol_method)

df_normalized_interpol.to_csv("output_"+scenariotype+"_interpolated_"+interpol_method+"_normalized"+pop_or_nopop+".cs
v")

                # output without any N/A years
                missing_years = list(df_interpol[df_interpol.isna().any(axis=1)]['year'].values)
                for artal in missing_years:
                    df_interpol=df_interpol[df_interpol.year != artal]
                df_interpol=df_interpol.reset_index(drop=True)
                df_interpol=add_training_set_variable(df_interpol,countries,training_percentage) # add dummy
                df_interpol.to_excel(writer1,sheet_name=interpol_method+", no missing values")

df_interpol.to_csv("output_"+scenariotype+"_interpolated_"+interpol_method+"_no_missing_values"+pop_or_nopop+".csv")

                missing_years_normalized                                                                    =
list(df_normalized_interpol[df_normalized_interpol.isna().any(axis=1)]['year'].values)
                for artal in missing_years_normalized:
                    df_normalized_interpol=df_normalized_interpol[df_normalized_interpol.year != artal]
                df_normalized_interpol=df_normalized_interpol.reset_index(drop=True)

df_normalized_interpol=add_training_set_variable(df_normalized_interpol,countries,training_percentage) # add dummy
                df_normalized_interpol.to_excel(writer2,sheet_name=interpol_method+", no missing")

df_normalized_interpol.to_csv("output_"+scenariotype+"_interpolated_"+interpol_method+"_normalized_no_missing_values"
+pop_or_nopop+".csv")
```

Appendix B – Support Vector Regression Machines (SVM.py)

```
# Application of our dataset 80% model #


# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import copy
# %matplotlib inline



#X = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values.csv")
#X = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop.csv")
X = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop-2.csv")


X.head()

X_final = X.dropna()

print(X_final.head(10))

X = X_final.drop(['growth', "Unnamed: 0", "countrycode", "year"], axis = 1)
y = X_final[['growth', 'training_set']]

#X_shap = X.drop(['year'], axis = 1)

print(y)
```

```
X_train1 = X.loc[X['training_set'] == 1]
X_train = X_train1.drop(['training_set'], axis = 1)

X_test1 = X.loc[X['training_set'] == 0]
X_test = X_test1.drop(['training_set'], axis = 1)

y_train1 = y.loc[y['training_set'] == 1]
y_train = y_train1.drop(['training_set'], axis = 1)

y_test1 = y.loc[y['training_set'] == 0]
y_test = y_test1.drop(['training_set'], axis = 1)
X_train

# Create a SVM Regressor

from sklearn import svm

reg = svm.SVR(kernel='rbf', C=10, gamma=0.01, epsilon=0.1)

reg.fit(X_train, y_train)

y_pred = reg.predict(X_test)

# Model Evaluation
from sklearn import metrics

print('MAE:',metrics.mean_absolute_error(y_test, y_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

import csv

# exporting a list variable into the csv file
input_variable = y_pred

# Example.csv gets created in the current working directory
with open('SVRM_y_pred.csv', 'w', newline = '') as csvfile:
    my_writer = csv.writer(csvfile, delimiter = ' ')
    my_writer.writerow(input_variable)

# Variable importance
 %pip install shap
import shap as shap
#on SHAPLEY values look here: https://docs.seldon.io/projects/alibi/en/stable/index.html

explainer = shap.KernelExplainer(reg.predict, X_train)
shap_values = explainer.shap_values(X_test, nsamples=100)

shap.summary_plot(shap_values, max_display=56, features=X_test, feature_names=X.columns)

shap.summary_plot(shap_values, max_display=56, features=X_test, feature_names=X.columns, plot_type='bar')

"""# Application of our dataset all years model"""

# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import copy
# %matplotlib inline


# Load data
X = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop-2.csv")


X.head()

X_final = X.dropna()

print(X_final.head(10))

X = X_final.drop(['growth', "Unnamed: 0", "countrycode", "training_set"], axis = 1)
y = X_final[['growth']]
print(y)

from sklearn import svm

# Create a SVM Regressor
reg = svm.SVR(kernel='rbf', C=10, gamma=0.01, epsilon=0.1)

from sklearn.model_selection import GridSearchCV
svc = svm.SVR()
clf = GridSearchCV(svc, param_grid = [{'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ])
```

```
clf.fit(X, y)
clf.best_params_

reg.fit(X, y)

y_pred = reg.predict(X)

import csv

# exporting a list variable into the csv file
input_variable = y_pred

# Example.csv gets created in the current working directory
with open('SVRM_y_pred_all2.csv', 'w', newline = '') as csvfile:
    my_writer = csv.writer(csvfile, delimiter = ' ')
    my_writer.writerow(input_variable)

# Model Evaluation
from sklearn import metrics

print('R^2:',metrics.r2_score(y, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y, y_pred))*(len(y)-1)/(len(y)-X.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y, y_pred))
print('MSE:',metrics.mean_squared_error(y, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y, y_pred)))

# Variable importance


%pip install shap
import shap as shap
#on SHAPLEY values look here: https://docs.seldon.io/projects/alibi/en/stable/index.html

explainer = shap.KernelExplainer(reg.predict, X)
shap_values = explainer.shap_values(X, nsamples=100)

# Create vector of correct names

import pandas as pd
import time

wdi_table=pd.read_csv('namepairs.csv')
wdi_table_names=wdi_table[["Indicator Code", 'Indicator Name']]
wdi_table_names

data = [["previous_growth", 'The GDP growth of the year before (annual %)'],
        ["year", "Year"],
        ["labsh", "Labor share of income (% of total)"],
        ["schooling", "Average total years of schooling for adult population(years)"],
        ["freedomhouse", "Freedome House Index"]]


df = pd.DataFrame(data, columns=["Indicator Code", "Indicator Name"])

all_names=pd.concat([wdi_table_names,df])

wdi_dict=dict(all_names.values)

names= X.columns
new_names = []
for namn in names:
  try:
    new_names.append(wdi_dict[namn])
  except:
    new_names.append(namn)

# Make Shap plots

shap.summary_plot(shap_values, max_display=57, features=X, feature_names=new_names)

shap.summary_plot(shap_values, max_display=60, features=X, feature_names=new_names, plot_type='bar')
```

Appendix C – Gaussian Process Boosting (GPBoost) (GPBoost.py)

```
Application of our dataset 80% model


# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import copy
```

```
%matplotlib inline
%pip install gpboost
import gpboost as gpb
import sklearn.datasets as datasets
import time
import csv


X_final = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop-2.csv")

X_final.head()

X = X_final.drop(['growth', "Unnamed: 0", "countrycode"], axis = 1)
y = X_final[['growth', 'training_set']]

group = X_final[['countrycode', 'training_set']]
X_shap = X.drop(['year'], axis = 1)

print(group)

rows_in_table=185
proportion = 0.8
ntrain = int(proportion*rows_in_table)

n = int(ntrain/proportion) # combined number of training and test data
m = len(set(group))  # number of categories / levels for grouping variable

# split train and test data
y_train = y[0:ntrain]
y_test = y[ntrain:n]
X_train = X.iloc[0:ntrain,]
X_test = X.iloc[ntrain:n,]
group_train = group[0:ntrain]
group_test = group[ntrain:n]

X_train1 = X.loc[X['training_set'] == 1]
X_train = X_train1.drop(['training_set'], axis = 1)

X_test1 = X.loc[X['training_set'] == 0]
X_test = X_test1.drop(['training_set'], axis = 1)

y_train1 = y.loc[y['training_set'] == 1]
y_train = y_train1.drop(['training_set'], axis = 1)

y_test1 = y.loc[y['training_set'] == 0]
y_test = y_test1.drop(['training_set'], axis = 1)

group_train1 = group.loc[y['training_set'] == 1]
group_train = group_train1.drop(['training_set'], axis = 1)

group_test1 = group.loc[y['training_set'] == 0]
group_test = group_test1.drop(['training_set'], axis = 1)

# Define and train GPModel
gp_model = gpb.GPModel(group_data=group_train)
# create dataset for gpb.train function
data_train = gpb.Dataset(X_train, y_train)
# specify tree-boosting parameters as a dict
params = { 'objective': 'regression_l2', 'learning_rate': 0.01,
    'max_depth': 6, 'min_data_in_leaf': 5, 'verbose': 0 }
# train model
bst = gpb.train(params=params, train_set=data_train, gp_model=gp_model, num_boost_round=1)
gp_model.summary() # estimated covariance parameters

# Make predictions
pred = bst.predict(data=X_test, group_data_pred=group_test)
y_pred = pred['fixed_effect'] + pred['random_effect_mean'] # sum predictions of fixed effect and random effect
y_pred = y_pred.reshape(135,1) ## THIS IS SPECIFIC TO THE GIVEN NUMBER OF ROWS

mae=np.mean((y_test - y_pred) ** 2) # root mean square error (RMSE) on test data.
rmserr=np.sqrt(np.mean((y_test - y_pred) ** 2)) # root mean square error (RMSE) on test data.
print('root mean square error is '+str(rmserr))
print('mean abseolute error is '+str(mae))


# exporting a list variable into the csv file
import csv

input_variable = y_pred

# Example.csv gets created in the current working directory
with open('GPBoost_y_pred.csv', 'w', newline = '') as csvfile:
    my_writer = csv.writer(csvfile, delimiter = ' ')
    my_writer.writerow(input_variable)
```

```
gp_model = gpb.GPModel(group_data=group_train)
cvbst = gpb.cv(params=params, train_set=data_train,
               gp_model=gp_model, use_gp_model_for_validation=False,
               num_boost_round=100, early_stopping_rounds=5,
               nfold=4, verbose_eval=True, show_stdv=False, seed=1)
best_iter = np.argmin(cvbst['l2-mean'])
print("Best number of iterations: " + str(best_iter))
# Best number of iterations: 72

# SHAP Values

# Commented out IPython magic to ensure Python compatibility.
# %pip install shap
import shap as shap
#on SHAPLEY values look here: https://docs.seldon.io/projects/alibi/en/stable/index.html

explainer = shap.Explainer(bst)
shap_values = explainer(X_train)

#Shap plots

shap.summary_plot(shap_values, max_display=56, feature_names=X.columns)

shap.summary_plot(shap_values, max_display=56, feature_names=X.columns, plot_type="bar")

# Application of our dataset 100% model #

# Load dataset

# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import copy
%matplotlib inline
%pip install gpboost
import gpboost as gpb
import sklearn.datasets as datasets
import time
import csv

X_final = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop-2.csv")

X_final.head()

X = X_final.drop(['growth', "Unnamed: 0", "countrycode", 'training_set'], axis = 1)
y = X_final[['growth']]

group = X_final[['countrycode']]
print(group)

# Define and train GPModel
gp_model = gpb.GPModel(group_data=group)
# create dataset for gpb.train function
data_train = gpb.Dataset(X, y)
# specify tree-boosting parameters as a dict
params = { 'objective': 'regression_l2', 'learning_rate': 0.01,
    'max_depth': 6, 'min_data_in_leaf': 5, 'verbose': 0 }
# train model
bst = gpb.train(params=params, train_set=data_train, gp_model=gp_model, num_boost_round=2)
gp_model.summary() # estimated covariance parameters

# Make predictions
pred = bst.predict(data=X, group_data_pred=group)
y_pred = pred['fixed_effect'] + pred['random_effect_mean'] # sum predictions of fixed effect and random effect
y_pred = y_pred.reshape(729,1) ## THIS IS SPECIFIC TO THE GIVEN NUMBER OF ROWS

from sklearn import metrics

print('R^2:',metrics.r2_score(y, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y, y_pred))*(len(y)-1)/(len(y)-X.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y, y_pred))
print('MSE:',metrics.mean_squared_error(y, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y, y_pred)))

# exporting a list variable into the csv file
import csv
input_variable = y_pred

# Example.csv gets created in the current working directory
with open('GPBoost_y_pred.csv', 'w', newline = '') as csvfile:
    my_writer = csv.writer(csvfile, delimiter = ' ')
    my_writer.writerow(input_variable)

# Parmeter search
gp_model = gpb.GPModel(group_data=group)
```

```
cvbst = gpb.cv(params=params, train_set=data_train,
               gp_model=gp_model, use_gp_model_for_validation=False,
               num_boost_round=100, early_stopping_rounds=5,
               nfold=4, verbose_eval=True, show_stdv=False, seed=1)
best_iter = np.argmin(cvbst['l2-mean'])
print("Best number of iterations: " + str(best_iter))
# Best number of iterations: 0

# Shap values

# Commented out IPython magic to ensure Python compatibility.
# %pip install shap
import shap as shap
#on SHAPLEY values look here: https://docs.seldon.io/projects/alibi/en/stable/index.html

explainer = shap.Explainer(bst)
shap_values = explainer(X)

# Correct names for plots

import pandas as pd
import time

wdi_table=pd.read_csv('namepairs.csv')
wdi_table_names=wdi_table[["Indicator Code", 'Indicator Name']]
wdi_table_names

data = [["previous_growth", 'The GDP growth of the year before (annual %)'],
        ["year", "Year"],
        ["labsh", "Labor share of income (% of total)"],
        ["schooling", "Average total years of schooling for adult population(years)"],
        ["freedomhouse", "Freedome House Index"]]

df = pd.DataFrame(data, columns=["Indicator Code", "Indicator Name"])

all_names=pd.concat([wdi_table_names,df])

wdi_dict=dict(all_names.values)

names= X.columns
new_names = []
for namn in names:
  try:
    new_names.append(wdi_dict[namn])
  except:
    new_names.append(namn)

 # Shap plots
shap.summary_plot(shap_values, max_display=60, feature_names=new_names)

shap.summary_plot(shap_values, max_display=60, feature_names=new_names, plot_type="bar")
```

Appendix D – Long Short-Term Memory Recurrent Neural Network (imvlstm.py)

```
# too work on Google Colab, do following:
# Step 1) under menu "runtime", choose row "change runtime". Set "GPU" to "on"
# Step 2) upload file "networks.py"
# Step 3) upload file "output_EU_interpolated_linear_normalized_no_missing_values_nopop.csv"
# Step 4a) upload file "GPBoost_y_pred.csv"
# Step 4b) upload file "SVRM_y_pred2.csv"
# Step 4c) upload file "GAM_y_pred.csv"

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import csv

# Networks.py from https://github.com/KurochkinAlexey/IMV_LSTM/blob/master/networks.py

from networks import IMVTensorLSTM

import torch
from torch import nn
from torch.utils.data import TensorDataset, DataLoader

data = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop.csv")

gpb_preds_list=[]
with open('GPBoost_y_pred.csv', newline='') as csvfile:
  csv_reader = csv.reader(csvfile, delimiter=' ', quotechar='|')
```

```
  for row in csv_reader:
    for val in row:
      gpb_preds_list.append(float(val.strip("[]")))
gpb_preds=np.array(gpb_preds_list)

svrm_preds_list=[]
with open('SVRM_y_pred2.csv', newline='') as csvfile:
  csv_reader = csv.reader(csvfile, delimiter=' ', quotechar='|')
  for row in csv_reader:
    for val in row:
      svrm_preds_list.append(float(val))
svrm_preds=np.array(svrm_preds_list)

gam_preds_list=[]
with open('GAM_y_pred.csv', newline='') as csvfile:
  csv_reader = csv.reader(csvfile, delimiter=' ', quotechar='|')
  for row in csv_reader:
    for val in row:
      gam_preds_list.append(float(val))
gam_preds=np.array(gam_preds_list)

trainval_frac = 0.2 # decent: 0.75

# remove unwanted columns from our data file
#data.drop(['training_set'], axis=1, inplace=True)

#cbwd = pd.get_dummies(data['cbwd'])
#cbwd.columns = ["cbwd_{}".format(i) for i in range(cbwd.shape[1])]
#data = pd.concat([data, cbwd], axis=1)

#data.drop(['cbwd'], axis=1, inplace=True)

depth = 10

countryname = {
  "AUT": "Austria",
  "BEL": "Belgium",
  "BGR": "Bulgaria",
  "CYP": "Cyprus",
  "CZE": "Czech Republic",
  "DEU": "Germany",
  "DNK": "Denmark",
  "ESP": "Spain",
  "EST": "Estonia",
  "FIN": "Finland",
  "FRA": "France",
  "GRC": "Greece",
  "HRV": "Croatia",
  "HUN": "Hungary",
  "IRL": "Ireland",
  "ITA": "Italy",
  "LTU": "Lithuania",
  "LUX": "Luxembourg",
  "LVA": "Latvia",
  "MLT": "Malta",
  "NLD": "Netherlands",
  "POL": "Poland",
  "PRT": "Portugal",
  "ROU": "Romania",
  "SVK": "Slovakia",
  "SVN": "Slovenia",
  "SWE": "Sweden"
}
countrynumber = {
  "AUT": 1,
  "BEL": 2,
  "BGR": 3,
  "CYP": 4,
  "CZE": 5,
  "DEU": 6,
  "DNK": 7,
  "ESP": 8,
  "EST": 9,
  "FIN": 10,
  "FRA": 11,
  "GRC": 12,
  "HRV": 13,
  "HUN": 14,
  "IRL": 15,
  "ITA": 16,
  "LTU": 17,
  "LUX": 18,
  "LVA": 19,
  "MLT": 20,
  "NLD": 21,
```

```
  "POL": 22,
  "PRT": 23,
  "ROU": 24,
  "SVK": 25,
  "SVN": 26,
  "SWE": 27
}
countries = data.countrycode.unique()

### the line below creates dummy variables instead
#data=pd.get_dummies(data, columns=["countrycode"], prefix=["country"])
### the line below changes countrycodes to numbers
data.countrycode = [countrynumber[item] for item in data.countrycode]

# [new] training versus values
data_train = data.loc[data['training_set'] == 1]
data_train.drop(['training_set'], axis=1, inplace=True)
data_test = data.loc[data['training_set'] == 0]
data_test.drop(['training_set'], axis=1, inplace=True)
data.drop(['training_set'], axis=1, inplace=True)

y = data['growth'].fillna(method='ffill').values
data=data.drop(columns=['growth'])
cols = list(data.columns[4:])
#cols.insert(0, 'countrycode') # add 'countrycode' as a variable

X = np.zeros((len(data), depth, len(cols)))
for i, name in enumerate(cols):
    for j in range(depth):
        X[:, j, i] = data[name].shift(depth - j - 1).fillna(method='bfill')

train_bound = int(0.6*(len(data)))
val_bound = int(0.8*(len(data)))

X_train = X[:train_bound]
X_val = X[train_bound:val_bound]
X_test = X[val_bound:]
y_train = y[:train_bound]
y_val = y[train_bound:val_bound]
y_test = y[val_bound:]

# [new] X_test, X_train, and X_val
X_trainval = np.zeros((len(data_train), depth, len(cols)))
for i, name in enumerate(cols):
    for j in range(depth):
        X_trainval[:, j, i] = data_train[name].shift(depth - j - 1).fillna(method='bfill')
X_train = X_trainval[:int(trainval_frac*len(X_trainval))]
X_val = X_trainval[int(trainval_frac*len(X_trainval)):]
X_test = np.zeros((len(data_test), depth, len(cols)))
for i, name in enumerate(cols):
    for j in range(depth):
        X_test[:, j, i] = data_test[name].shift(depth - j - 1).fillna(method='bfill')

# [new] y_test, y_train, and y_val
y_trainval = data_train['growth'].fillna(method='ffill').values
y_train = y_trainval[:int(trainval_frac*len(y_trainval))]
y_val = y_trainval[int(trainval_frac*len(y_trainval)):]
y_test = data_test['growth'].fillna(method='ffill').values

X_train_min, X_train_max = X_train.min(axis=0), X_train.max(axis=0)
y_train_min, y_train_max = y_train.min(axis=0), y_train.max(axis=0)

X_train = (X_train - X_train_min)/(X_train_max - X_train_min + 1e-9)
X_val = (X_val - X_train_min)/(X_train_max - X_train_min + 1e-9)
X_test = (X_test - X_train_min)/(X_train_max - X_train_min + 1e-9)
y_train = (y_train - y_train_min)/(y_train_max - y_train_min +1e-9)
y_val = (y_val - y_train_min)/(y_train_max - y_train_min + 1e-9)
y_test = (y_test - y_train_min)/(y_train_max - y_train_min + 1e-9)

X_train_t = torch.Tensor(X_train)
X_val_t = torch.Tensor(X_val)
X_test_t = torch.Tensor(X_test)
y_train_t = torch.Tensor(y_train)
y_val_t = torch.Tensor(y_val)
y_test_t = torch.Tensor(y_test)

train_loader = DataLoader(TensorDataset(X_train_t, y_train_t), batch_size=64, shuffle=True)
val_loader = DataLoader(TensorDataset(X_val_t, y_val_t), batch_size=64, shuffle=False)
test_loader = DataLoader(TensorDataset(X_test_t, y_test_t), batch_size=64, shuffle=False)

model = IMVTensorLSTM(X_train_t.shape[2], 1, 128).cuda()

opt = torch.optim.Adam(model.parameters(), lr=0.001)

epoch_scheduler = torch.optim.lr_scheduler.StepLR(opt, 20, gamma=0.9)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

epochs = 1000
loss = nn.MSELoss()
patience = 35 # originally 35, proper value 1000
min_val_loss = 9999
counter = 0
for i in range(epochs):
    mse_train = 0
    for batch_x, batch_y in train_loader:
        batch_x = batch_x.cuda()
        batch_y = batch_y.cuda()
        opt.zero_grad()
        y_pred, alphas, betas = model(batch_x)
        y_pred = y_pred.squeeze(1)
        l = loss(y_pred, batch_y)
        l.backward()
        mse_train += l.item()*batch_x.shape[0]
        opt.step()
    epoch_scheduler.step()
    with torch.no_grad():
        mse_val = 0
        preds = []
        true = []
        for batch_x, batch_y in val_loader:
            batch_x = batch_x.cuda()
            batch_y = batch_y.cuda()
            output, alphas, betas = model(batch_x)
            output = output.squeeze(1)
            preds.append(output.detach().cpu().numpy())
            true.append(batch_y.detach().cpu().numpy())
            mse_val += loss(output, batch_y).item()*batch_x.shape[0]
    preds = np.concatenate(preds)
    true = np.concatenate(true)

    if min_val_loss > mse_val**0.5:
        min_val_loss = mse_val**0.5
        print("Saving...")
        torch.save(model.state_dict(), "imv_tensor_lstm_pm25.pt")
        counter = 0
    else:
        counter += 1

    if counter == patience:
        break
    print("Iter: ", i, "train: ", (mse_train/len(X_train_t))**0.5, "val: ", (mse_val/len(X_val_t))**0.5)
    if(i % 10 == 0):
        preds = preds*(y_train_max - y_train_min) + y_train_min
        true = true*(y_train_max - y_train_min) + y_train_min
        mse = mean_squared_error(true, preds)
        mae = mean_absolute_error(true, preds)
        print("lr: ", opt.param_groups[0]["lr"])
        print("mse: ", mse, "mae: ", mae)
        plt.figure(figsize=(20, 10))
        plt.plot(true)
        plt.plot(preds)
        plt.show()

model.load_state_dict(torch.load("imv_tensor_lstm_pm25.pt"))

with torch.no_grad():
    mse_val = 0
    preds = []
    true = []
    alphas = []
    betas = []
    for batch_x, batch_y in test_loader:
        batch_x = batch_x.cuda()
        batch_y = batch_y.cuda()
        output, a, b = model(batch_x)
        output = output.squeeze(1)
        preds.append(output.detach().cpu().numpy())
        true.append(batch_y.detach().cpu().numpy())
        alphas.append(a.detach().cpu().numpy())
        betas.append(b.detach().cpu().numpy())
        mse_val += loss(output, batch_y).item()*batch_x.shape[0]
preds = np.concatenate(preds)
true = np.concatenate(true)

preds = preds*(y_train_max - y_train_min) + y_train_min
true = true*(y_train_max - y_train_min) + y_train_min

mse = mean_squared_error(true, preds)
mae = mean_absolute_error(true, preds)
```

```
mse, mae

plt.figure(figsize=(20, 10))
plt.plot(true)
plt.plot(preds)
plt.plot(gpb_preds) # GPB
plt.plot(svrm_preds) # SVRM
plt.plot(gam_preds) # SVRM
plt.legend(['true', 'lstm_preds', 'gpb_preds', 'svrm_preds', 'gam_preds'])
plt.show()

alphas = np.concatenate(alphas)
betas = np.concatenate(betas)

alphas = alphas.mean(axis=0)
betas = betas.mean(axis=0)

alphas = alphas[..., 0]
betas = betas[..., 0]

alphas = alphas.transpose(1, 0)

fig, ax = plt.subplots(figsize=(20, 20))
im = ax.imshow(alphas)
ax.set_xticks(np.arange(X_train_t.shape[1]))
ax.set_yticks(np.arange(len(cols)))
ax.set_xticklabels(["t-"+str(i) for i in np.arange(X_train_t.shape[1], -1, -1)])
ax.set_yticklabels(list(cols))
for i in range(len(cols)):
    for j in range(X_train_t.shape[1]):
        text = ax.text(j, i, round(alphas[i, j], 3),
                       ha="center", va="center", color="w")
ax.set_title("Importance of features and timesteps")
#fig.tight_layout()
plt.show()

plt.figure(figsize=(20, 20))
plt.title("Feature importance")
plt.barh(range(len(cols)), betas)
plt.yticks(ticks=range(len(cols)), labels=cols, rotation=90)

plt.figure(figsize=(20, 10))
plt.plot(true)
plt.plot(preds)
plt.plot(gpb_preds) # GPB
plt.plot(svrm_preds) # SVRM
plt.plot(gam_preds) # SVRM
plt.legend(['true', 'lstm_preds', 'gpb_preds', 'svrm_preds', 'gam_preds'])
plt.text(0, 1.1, 'LSTM:\nRMSE = '+str(mse**0.5)+'\nMAE = '+str(mae))
plt.show()

plt.figure(figsize=(20, 10))
plt.plot(true)
plt.plot(preds)
plt.plot(gpb_preds) # GPB
plt.plot(svrm_preds) # SVRM
plt.plot(gam_preds) # SVRM
plt.legend(['true', 'lstm_preds', 'gpb_preds', 'svrm_preds', 'gam_preds'])
plt.xlabel('observation (time point)')
plt.ylabel('normalized logarithmized real GDP growth rate')
#plt.text(0, 1.1, 'LSTM:\nRMSE = '+str(mse**0.5)+'\nMAE = '+str(mae))
plt.show()

wdi_table=pd.read_csv('namepairs.csv')

wdi_dict=dict(wdi_table[['Indicator Code','Indicator Name']].values)

wdi_dict['schooling']='Number of years of schooling'
wdi_dict['labsh']='Labour share of income'
wdi_dict['freedomhouse']='Freedom House index'

unsorted_list = [(importance, feature) for feature, importance in
                 zip(cols, betas)]
sorted_list = sorted(unsorted_list)

features_sorted = []
importance_sorted = []

for i in sorted_list:
    try:
      features_sorted += [wdi_dict[i[1]]]
    except:
      features_sorted += [i[1]]
    importance_sorted += [i[0]]
```

```
plt.show()
plt.figure(figsize=(10, 20))

plt.title("Feature importance", fontsize=15)
plt.xlabel("Importance", fontsize=13)

plt.barh(range(len(importance_sorted)), importance_sorted, color="blue", edgecolor='blue')
plt.yticks(range(len(importance_sorted)), features_sorted);

for rad in sorted_list[::-1]:
  print(rad[1])

print(len(y_train))
print(len(y_val))
print(len(y_test))
print(str(len(y_train)+len(y_val)+len(y_test)))

for rad in sorted_list[::-1]:
  print(rad[1])
```

Appendix E – Generalized Additive Linear Model (GAM.py)

```
 # Application of dataset 100% #

# Commented out IPython magic to ensure Python compatibility.
# %pip install pygam
# %pip install pandas matplotlib

# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import copy

# %matplotlib inline

X_final = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop-2.csv")
X_final.head()
X = X_final.drop(['growth', "Unnamed: 0", "training_set"], axis = 1)
y = X_final[['growth']]
X

from sklearn.preprocessing import OrdinalEncoder
ord_enc = OrdinalEncoder()
X["countrycode"] = ord_enc.fit_transform(X[["countrycode"]])
X

X = X.to_numpy()
y = y.to_numpy()

from pygam import LinearGAM, s, f

gam = LinearGAM(f(0) + f(1) + s(2) + s(3) + s(4) + s(5) + s(6) + s(7) + s(8) + s(9) + s(10) + s(11) + s(12) + s(13) +
s(14)  + s(15) + s(16) + s(17) + s(18) + s(19) + s(20) + s(21) + s(22) + s(23) + s(24) + s(25) + s(26) + s(27) +
s(28) + s(29) + s(30) + s(31) + s(32) + s(33) + s(34) + s(35) + s(36) + s(37) + s(38) + s(39) + s(40) + s(41) + s(42)
+ s(43) + s(44) + s(45) + s(46) + s(47) + s(48) + s(49) + s(50) + s(51) + s(52) + s(53) + s(54) + s(55) + f(56) +
f(57)
).fit(X, y)

gam.summary()

y_pred = gam.predict(X)

from sklearn import metrics

print('R^2:',metrics.r2_score(y, y_pred))

print('Adjusted R^2:',1 - (1-metrics.r2_score(y, y_pred))*(len(y)-1)/(len(y)-X.shape[1]-1))

print('MAE:',metrics.mean_absolute_error(y, y_pred))

print('MSE:',metrics.mean_squared_error(y, y_pred))

print('RMSE:',np.sqrt(metrics.mean_squared_error(y, y_pred)))



import csv
# exporting a list variable into the csv file
input_variable = y_pred
```

```
# Example.csv gets created in the current working directory

with open('GAM_y_pred_all.csv', 'w', newline = '') as csvfile:

    my_writer = csv.writer(csvfile, delimiter = ' ')

    my_writer.writerow(input_variable)

# Application of dataset 80% #

%pip install pygam
%pip install pandas matplotlib

import pandas as pd
import numpy as np
import copy
%matplotlib inline

X_final = pd.read_csv("output_EU_interpolated_linear_normalized_no_missing_values_nopop-2.csv")
X_final.head()

X = X_final.drop(['growth', "Unnamed: 0"], axis = 1)
y = X_final[['growth', 'training_set']]

from sklearn.preprocessing import OrdinalEncoder

ord_enc = OrdinalEncoder()
X["countrycode"] = ord_enc.fit_transform(X[["countrycode"]])
X

X_train1 = X.loc[X['training_set'] == 1]
X_train = X_train1.drop(['training_set'], axis = 1)
X_test1 = X.loc[X['training_set'] == 0]
X_test = X_test1.drop(['training_set'], axis = 1)

y_train1 = y.loc[y['training_set'] == 1]
y_train = y_train1.drop(['training_set'], axis = 1)

y_test1 = y.loc[y['training_set'] == 0]
y_test = y_test1.drop(['training_set'], axis = 1)

X_train = X_train.to_numpy()
y_train = y_train.to_numpy()

from pygam import LinearGAM, s, f

gam = LinearGAM(f(0) + f(1) + s(2) + s(3) + s(4) + s(5) + s(6) + s(7) + s(8) + s(9) + s(10) + s(11) + s(12) + s(13) +
s(14) + s(15) + s(16) + s(17) + s(18) + s(19) + s(20) + s(21) + s(22) + s(23) + s(24) + s(25) + s(26) + s(27) + s(28)
+ s(29) + s(30) + s(31) + s(32) + s(33) + s(34) + s(35) + s(36) + s(37) + s(38) + s(39) + s(40) + s(41) + s(42) +
s(43) + s(44) + s(45) + s(46) + s(47) + s(48) + s(49) + s(50) + s(51) + s(52) + s(53) + s(54) + s(55) + f(56) + f(57)

).fit(X_train, y_train)

gam.summary()

y_pred = gam.predict(X_test)
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test, y_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

import csv
# exporting a list variable into the csv file
input_variable = y_pred
# Example.csv gets created in the current working directory

with open('GAM_y_pred.csv', 'w', newline = '') as csvfile:

    my_writer = csv.writer(csvfile, delimiter = ' ')

    my_writer.writerow(input_variable)
```