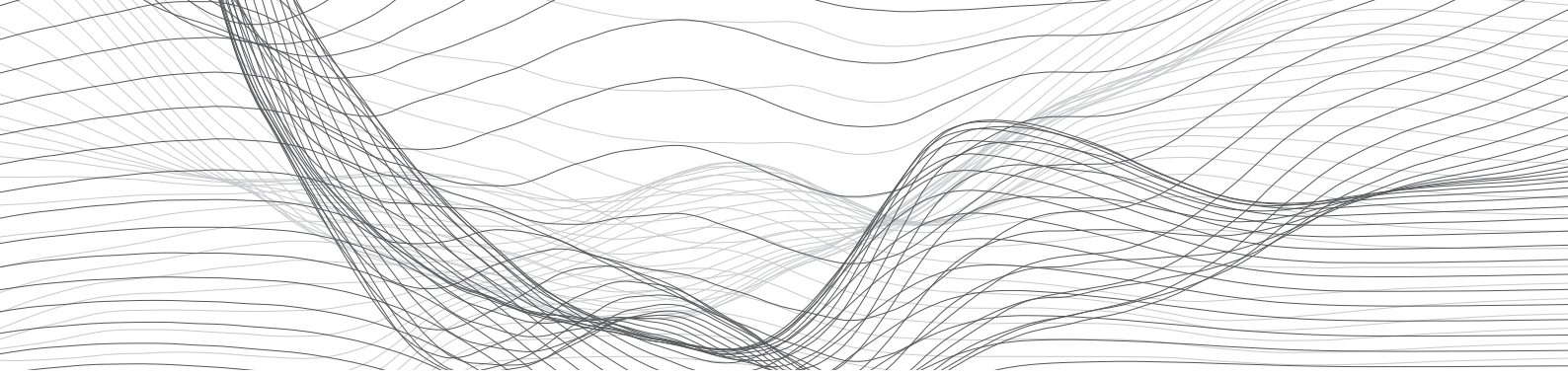


Nowcasting GDP: what are the gains from machine learning algorithms?

Milen Arro-Cannarsa, Rolf Scheufele

SNB Working Papers
6/2024

SCHWEIZERISCHE NATIONALBANK
BANQUE NATIONALE SUISSE
BANCA NAZIONALE SVIZZERA
BANCA NAZIUNALA SVIZRA
SWISS NATIONAL BANK

ISSN 1660-7716 (printed version)
ISSN 1660-7724 (online version)

Nowcasting GDP: What are the gains from machine learning algorithms?*

Milen Arro-Cannarsa† and Rolf Scheufele‡

June 4, 2024

Abstract

We compare several machine learning methods for nowcasting GDP. A large mixed-frequency data set is used to investigate different algorithms such as regression based methods (LASSO, ridge, elastic net), regression trees (bagging, random forest, gradient boosting), and SVR. As benchmarks, we use univariate models, a simple forward selection algorithm, and a principal components regression. The analysis accounts for publication lags and treats monthly indicators as quarterly variables combined via blocking. Our data set consists of more than 1,100 time series. For the period after the Great Recession, which is particularly challenging in terms of nowcasting, we find that all considered machine learning techniques beat the univariate benchmark up to 28 % in terms of out-of-sample RMSE. Ridge, elastic net, and SVR are the most promising algorithms in our analysis, significantly outperforming principal components regression.

JEL classification: C53, C55, E32, E37.

Keywords: Nowcasting, forecasting, machine learning, ridge, LASSO, elastic net, random forest, bagging, boosting, SVM, SVR, large data sets.

*We especially thank Pierpaolo Benigno, Martin Brown, Michele Lenza, Dirk Niepelt, and the participants of the ICMAIF 2024 Conference in Crete, Alumni 2023 Conference in Gerzensee, the COMPSTAT 2023 Conference in London, seminars at the University of Bern, and Brown Bag series at the Swiss National Bank (SNB) for helpful discussions and insightful comments. The views, opinions, findings, and conclusions or recommendations expressed in this paper are strictly those of the authors. They do not necessarily reflect the views of the SNB. The SNB takes no responsibility for any errors or omissions in, or for the correctness of, the information contained in this paper.

†Study Center Gerzensee & University of Bern, e-mail: milen.arro@szgerzensee.ch

‡Swiss National Bank, Economic Analysis, e-mail: rolf.scheufele@snb.ch

1 Introduction

Timely and accurate information on economic conditions, such as Gross Domestic Product (GDP), is critical for effective policy-making. However, official statistics often suffer from significant time delays, hindering policymakers' ability to promptly respond to evolving economic dynamics. To address this challenge, *nowcasting* procedures have been developed to provide real-time estimates of economic indicators that draw upon a wide array of data sources (see Bańbura et al., 2013, Evans, 2005, Giannone et al., 2008).

Traditionally, nowcasting methods have relied on statistical models, such as factor models (see e.g. Bańbura et al., 2013, Stock and Watson, 2002b) and single-indicator pooling techniques (see e.g. Heinisch and Scheufele, 2018, Kuzin et al., 2013), to extract information from diverse economic indicators[1]. While these methods have proven effective in handling large datasets with different frequencies and release delays, recent advancements in machine learning (ML) present new opportunities for enhancing nowcasting accuracy.

We ask ourselves the following question: Can machine learning (ML) algorithms be used to enhance the accuracy of nowcasting gross domestic product (GDP)? The literature on various machine learning techniques has grown in recent decades. Books, such as Hastie et al. (2009) and James et al. (2013), analyze the benefits and techniques of various machine learning algorithms in detail. Machine learning algorithms build statistical models to explain data and then optimize the fit based on past data (supervised learning). The algorithms continuously and autonomously update the parameters until a threshold of accuracy has been met. It is important for policy makers, particularly for central banks, to understand if these techniques can be leveraged for GDP nowcasting. In the case of nowcasting, machine learning algorithms have the advantage that they can efficiently handle large datasets with lots of potential regressors. Thus, ML algorithms can solve the curse of dimensionality when there are many potential predictors and are a promising alternative to the usual time-series regression based methods used by central banks for nowcasting.

Galli et al. (2019) showed that a large data set is key to obtaining accurate nowcasts

[1]For Switzerland, these methods have been successfully applied by Galli et al. (2019) and Kronenberg et al. (2023).

for Swiss GDP. Instead of relying on standard factor models, we investigate whether using machine learning algorithms can improve standard nowcasting methods. We use a dataset with over 1,100 indicators and just 76 observations to nowcast Swiss GDP growth. Therefore, we have a large data set with relatively small number of time periods. To address the issue of overfitting and reduced accuracy in forecasting due to a limited number of observations, we explore the potential benefits of using machine learning algorithms. By leveraging advanced techniques in machine learning, we hope to improve the accuracy of our nowcasting predictions despite limited available data.

In our analysis, we apply regression based methods: ridge (Hoerl and Kennard (1970)), LASSO (Tibshirani (1996)), elastic net (Zou and Hastie (2005)), tree based ensemble methods: bagging (Breiman (1996)), random forest (Breiman (2001)), boosting (Friedman (2001)) and support vector regression (SVR) with various kernels. To evaluate the nowcasting performances of these machine learning models, we use root mean squared forecast error (RMSE), a statistical measure of accuracy.

We add to the growing literature by central banks and other authors, such as Chinn et al. (2023), Richardson et al. (2021), Döpke et al. (2017), Chakraborty and Joseph (2017), Smalter Hall and Cook (2017), and Sermpinis et al. (2014), that have applied machine learning for nowcasting economic activity. Our contribution is threefold. First, we show how to deal with many indicators in the context of mixed frequencies and ragged edges. Similarly to Bec and Mogliani (2015), we use blocking to handle mixed frequency in our data set. We address the different publications dates of indicators, the "ragged-edge" phenomena, with realignment proposed by Altissimo et al. (2010). The combination of these methods allows the policymaker to run nowcasting models straightforwardly in real time. Second, we illustrate how to optimally tune various machine learning algorithms in terms of cross-validation and shrinkage parameters. Third, our analysis suggests models that work well for nowcasting GDP in Switzerland.

Our findings indicate that ML algorithms offer significant improvements in GDP nowcasting accuracy compared to traditional methods, with certain algorithms achieving up to 28% reduction in out-of-sample RMSE. Importantly, the superiority of ML methods is consistent across different time periods, suggesting robust performance across varying economic conditions.

Contrary to expectations, our results suggest that linear ML algorithms, such as ridge regression, elastic net, and SVR with a linear kernel, outperform non-linear methods in GDP nowcasting. This underscores the importance of algorithm selection and parameter tuning to achieve optimal forecasting outcomes when using a large macroeconomic indicator set.

Our results indicate that in terms of GDP nowcasting, non-linearity is not the driver of the superiority of ML methods over traditional methods. Non-linear methods such as tree-based procedures or support vector regression with higher order kernels are not able to improve linear methods such as elastic net, ridge regression, or support vector regression with a linear kernel. Linear procedures in combination with the use of cross-validation seem to give the best nowcasting results.

In summary, this paper contributes to the ongoing discourse on integrating ML techniques in economic forecasting, offering insights into their effectiveness in enhancing GDP nowcasting accuracy. The subsequent sections detail the methodology, data sources, empirical results, and concluding remarks.

2 Methodology

In this section, we provide an overview of the various methods that we use for nowcasting GDP. We estimate three relatively standard models that will serve as benchmarks and various machine learning algorithms that we compare to the performance of benchmark models in terms of RMSE.

2.1 Benchmark models

We estimate three benchmark models: the Univariate Autoregressive (AR) model, Principal Component Analysis (PCA), and Forward Subset Selection (FSS). The AR model is our simple first benchmark. PCA is our hard-to-beat benchmark, as it is the foundation of most nowcasting models and has been shown to be very successful for GDP forecasting (see e.g. Giannone et al., 2008, Stock and Watson, 2002b). The FSS serves as a way to think about variable selection. Estimating these benchmarks will involve making choices about certain model specific parameters, such as lag order in the AR model or the number of principal components in the PCA model. Traditionally information criteria, such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC), can be used to determine such critical choices. Those measures balance the fit of the model to the data against its complexity. In general, the BIC places a larger penalty on models with many variables and therefore, is expected to perform better than the AIC in a set up with many predictors, such as ours[2]. As a first approach, we use the BIC to determine parameter choices in benchmark models. Later, in section 4.2, we make the same choices with cross-validation to see if incorporating learning from the data in itself improves the benchmark models.

The univariate autoregressive (AR) model is a natural first benchmark. We estimate an AR model of lag order q that can be expressed as:

$$y_t = \sum_{i=1}^{q} \phi_i y_{t-i} + \varepsilon_t, \tag{1}$$

[2]The AIC penalizes models for the number of parameters less harshly compared to the BIC.

where y_t is the dependent variable at time t, $\phi_1, \phi_2, \ldots, \phi_q$ are the coefficients of the lagged dependent variables, $y_{t-1}, y_{t-2}, \ldots, y_{t-q}$ and ε_t is the error term at time t.

One important consideration when working with AR models is the choice of lag order q. We use the BIC to determine the lag order q.

A factor model is used as a second, more sophisticated multivariate benchmark model. The basic idea of factor models is that a small number of latent variables (i.e., factors) can well approximate the fluctuations of many macroeconomic variables. Stock and Watson (2002b) and Stock and Watson (2002a) have suggested PCA, which can well approximate a *Dynamic Factor Model* when the number of variables is sufficiently large. The factor model allows one to characterize the interactions of a large number of variables within a relatively parsimonious framework and it has been found to be very successful in terms of nowcasting GDP. Given the factors, the forecast equation for GDP is simply

$$y_t = c + \sum_{i=1}^{q} \delta_i f_{it} + u_t, \tag{2}$$

where y_t is the dependent variable at time t, δ_i is the coefficient for the ith factor, f_{it} is the ith factor at time t, and u_t is the error term at time t.

The factors are extracted by PCA. The principal components, f_{it}, are defined as linear combinations of the original p variables, x_{jt}, such that:

$$f_{it} = \sum_{j=1}^{p} \lambda_{ij} x_{jt}, \tag{3}$$

where λ_{ij} is the jth loading for the ith principal component. PCA aims to find a linear combination of the original variables that captures the maximum amount of variation in the data, so that the resulting q principal components can be ordered by the amount of variation that they are able to explain in the total data set ($j = 1, ..., q, ..., p$). The individual PCs are uncorrelated with each other.

Since $q \ll p$, the procedure reduces the dimension of the data used for nowcasting from p indicators to a much smaller number of q principal components that explain GDP. This enables us to work with a dataset, where the number of indicators p is much larger than observations T. The key is to choose the correct number of principal components q

to include in our model. In line with Stock and Watson (2002b) we use the BIC to assess the optimal q.

A forward subset selection (FSS) is used as a third benchmark. FSS is a method used in regression analysis to select a subset of predictor variables that are most useful to predict a response variable. Let Q be the total number of useful predictors. The forward subset selection model can be expressed as:

$$y_t = \beta_0 + \sum_{q=1}^{Q} \beta_q x_{qt} + \varepsilon_t, \tag{4}$$

where y_t is the dependent variable at time t, β_0 is the intercept term, β_q is the coefficient for the qth predictor, x_{qt} is the value of the qth predictor at time t, ε_t is the error term at time t, and Q is the number of predictors included in the final model.

The method begins by fitting a simple linear regression model separately for each of Q predictor variable, and then selecting the variable with the highest correlation or lowest p-value as the initial predictor variable. Then, additional predictor variables are added to the model one by one, in order of decreasing correlation or increasing significance, until a stopping criterion is met. Clearly, the stopping criterion is crucial in this algorithm. We use the BIC to find the optimal model and set Q to minimize the BIC.

2.2 Machine learning methods

We use various models to learn from the data and then use them to make an out of sample nowcast. The exact procedure is described in section 3 and results are reported in section 4.

2.2.1 Regression based methods

Models used in this subsection are all linear and follow a standard regression model which is

$$y_t = \beta_0 + \sum_{j=1}^{p} \beta_j x_{jt} + \varepsilon_t, \tag{5}$$

where y_t is the dependent variable at time t, β_0 is the intercept term, β_j is the coefficient for the jth predictor, x_{jt} is the value of the jth predictor at time t, ε_t is the error term at

time t, and p is the number of predictors in the model.

The approaches that we use are known as *shrinkage* or *regularization* methods. The intention is to fit a model that takes all p predictors into consideration. The estimated coefficients are shrunken towards zero relative to the OLS estimate, which has the effect of reducing the variance (James et al., 2013). The advantage of shrinkage methods over OLS can be explained by the bias-variance trade-off. Particularly, in cases where the number of coefficients is larger than the number of observations (where OLS has no unique solution), shrinkage works well because a small increase in bias can be traded-off for a large increase in variance.

Ridge regression The ridge regression model, which was first proposed by Hoerl and Kennard (1970), is a regularized linear regression technique that adds a penalty term to the sum of squared errors to shrink the regression coefficients towards zero. The ridge regression model's objective function can be expressed as

$$\sum_{t=1}^{T}(y_t - \beta_0 - \sum_{j=1}^{p}\beta_j x_{jt})^2 + \lambda \sum_{j=1}^{p}\beta_j^2, \tag{6}$$

where y_t is the dependent variable at time t, β_0 is the intercept term, β_j is the coefficient for the jth predictor, x_{jt} is the value of the jth predictor at time t, ε_t is the error term at time t, and p is the number of predictors in the model.

The ridge estimator of the regression coefficients, β, is given by the following closed-form solution

$$\hat{\beta} = (X'X + \lambda I)^{-1}X'y,$$

where $\hat{\beta}_{ridge}$ represents the ridge estimator of the regression coefficients, X is an $T \times p$ matrix of predictor variables, y is an $T \times 1$ vector of response variables, λ is the tuning parameter (ridge penalty) that controls the amount of shrinkage applied to the coefficients, and I is the $p \times p$ identity matrix. Note that the ridge estimator is a biased estimator, as it introduces a bias towards zero in the coefficient estimates. However, this bias is often considered a reasonable trade-off for the reduction in variance, particularly when dealing with high-dimensional data.

The lambda value controls the strength of the penalty and determines the amount of shrinkage applied to the coefficients. A larger value of λ results in greater shrinkage with many coefficients near zero, while a smaller value of λ results in less shrinkage and a coefficients that are closer to their OLS counterparts.

Ridge regression can handle high-dimensional data sets with many predictors and can effectively handle multicollinearity between the predictors by shrinking their coefficients towards zero, but without forcing them to be exactly zero as in LASSO regression that will be described below. Ridge regression tends to perform better for models where y_t is truly a function of many predictors without any predictor being especially influential.

Least Absolute Shrinkage and Selection Operator (LASSO) LASSO regression is a regularization method used in linear regression to select a subset of relevant predictors and to reduce the impact of irrelevant predictors by imposing a penalty on their coefficients. Proposed by Tibshirani (1996), the LASSO regression adds a penalty term to the sum of squared errors to shrink the regression coefficients towards zero and then performs variable selection. The objective function is the sum of squared errors plus the LASSO penalty term:

$$\sum_{t=1}^{T}(y_t - \beta_0 - \sum_{j=1}^{p}\beta_j x_{jt})^2 + \lambda \sum_{j=1}^{p}|\beta_j|, \tag{7}$$

where λ is the LASSO hyperparameter, which controls the strength of the penalty, and the amount of shrinkage applied to the regression coefficients. A larger value of λ results in greater shrinkage and more coefficients set to zero, while a smaller value of λ results in less shrinkage and more coefficients retained. Unlike for ridge regression, there is no closed-form solution for the estimator of LASSO regression coefficients.

LASSO works well if there are some very influential predictors and some that have little to no effect, as it delivers a sparse model that involves only a subset of the original p predictors.

Elastic Net is a regularization method introduced by Zou and Hastie (2005) that combines both LASSO and ridge regularization techniques to achieve a balance between variable selection and coefficient shrinkage. The elastic net objective function can be repre-

sented mathematically as follows:

$$\sum_{t=1}^{T}(y_t - \beta_0 - \sum_{j=1}^{p}\beta_j x_{jt})^2 + \lambda_1 \sum_{j=1}^{p}|\beta_j| + \lambda_2 \sum_{j=1}^{p}\beta_j^2, \tag{8}$$

where y_t is the dependent variable at time t, β_0 is the intercept term, β_j is the coefficient for the jth predictor, x_{jt} is the value of the jth predictor at time t, ε_t is the error term at time t, and p is the number of predictors in the model. Furthermore, λ_1 and λ_2 are the elastic net parameters, which control the amount of shrinkage applied to the regression coefficients. The parameter λ_1 controls the amount of LASSO-like penalty, while λ_2 controls the amount of ridge-like penalty.

Elastic net combines the strengths of ridge and LASSO regression. It can select a subset of relevant predictors by setting some coefficients to exactly zero, while also allowing for correlated predictors to be selected together.

2.2.2 Tree based methods

Tree-based methods generally involve splitting the predictor space into a number of regions and make predictions using the mean of observations in the region to which it belongs. The regression tree model is a non-parametric regression technique that recursively partitions the predictor space into subregions and fits a separate constant value in each subregion. This model can be expressed as:

$$y_t = \sum_{m=1}^{M} c_m I(x_t \in R_m),$$

where y_t is the dependent variable at time t, x_t is the vector of predictor values at time t, R_m is the mth subregion of the predictor space, c_m is the mean value for the mth subregion, and M is the number of subregions in the tree. The predictor space X consisting of $x_1, x_2, ..., x_p$ is divided into J distinct and non-overlapping regions, $R_1, ..., R_J$. For every observation that falls into R_j, we make the same prediction, which is the mean of the outcome variable for the training observations in R_j.

To find the regions, we need to apply a top-down strategy which is called *recursive splitting*. The splitting rule sees to maximize the reduction in the sum of squared errors (SSE) between the predicted values and the actual values so that the resulting tree has

the lowest SSE. Namely, we choose $R_1(j,s)$ and $R_2(j,s)$ to minimize:

$$\sum_{x \in R_1(j,s)} (y_i - \hat{y}_{R1})^2 + \sum_{x \in R_2(j,s)} (y_i - \hat{y}_{R2})^2,$$

where j is the index of the predictor variable, s is the splitting threshold for the predictor variable, $\hat{y}_{R1}$ is the mean responds in R_1 and $\hat{y}_{R2}$ is the mean responds in R_2. Therefore, we need to find values for j and s to minimize this expression, which can be accomplished rather quickly when p is not too large. This process is repeated for the next best predictor and cutoff point and will be repeated until a stopping criterion is reached.

More precisely, in our analysis, we use ensemble learning methods that involve combining many trees and that are described below.

Bagging is the short version of bootstrap aggregation, which is a general statistical method for reducing the variance of a learning procedure. The bagging regression tree model is an learning method that takes advantage of averaging over multiple regression trees to improve the predictive accuracy and to reduce overfitting. The bagging regression tree model can be expressed as:

$$y_t = \frac{1}{B} \sum_{b=1}^{B} f_b(x_t), \tag{9}$$

where y_t is the dependent variable at time t, x_t is the vector of predictor values at time t, B is the number of trees in the ensemble, and $f_b(x_t)$ is the prediction of the bth regression tree for the predictor vector x_t.

Given the original training set, B separate training sets are calculated by bootstrapping and computing a regression tree $f_b(x_t)$ is computed for each bootstrapped data set. Finally, all predictions $f_1(x_t), ..., f_B(x_t)$ are averaged and this gives the bagged predictor for y_t. The intuition is that each individual tree has high variance, but low bias. Averaging over B dramatically reduces the variance.

The bagging regression tree model is a powerful and flexible machine learning method that can be applied to a wide range of predictive modeling tasks. However, it can be computationally intensive and difficult to interpret, especially when the number of trees and predictor variables is large.

Random Forest is an ensemble learning method first proposed by Breiman (2001) that is very similar to bagging. Similar to bagging, it combines multiple decision trees to improve the predictive accuracy and to reduce overfitting, however it also tries to increase robustness by introducing randomness in the predictors considered at each split. The random forest model can be expressed as:

$$y_t = \frac{1}{B}\sum_{b=1}^{B} f_b(x_t),$$

where y_t is the dependent variable at time t, x_t is the vector of predictor values at time t, B is the number of trees in the forest, and $f_b(x_t)$ is the prediction of the bth tree for the predictor vector x_t.

Similar to bagging, we built a number of decision trees on bootstrapped samples. However, for each split of the tree, only a random sample of m predictors are used as candidates instead of the full sets of predictors. This randomness helps reduce the correlation among the trees and to increase the diversity of the forest. We choose $m = \sqrt{p}$ (bagging implies $m = p$).

Gradient boosting is our third ensemble learning method that combines multiple weak regression trees to improve the predictive accuracy. The boosting regression tree model can be expressed as:

$$y_t = \sum_{b=1}^{B} \alpha_b f_b(x_t), \tag{10}$$

where y_t is the dependent variable at time t, x_t is the vector of predictor values at time t, B is the number of trees in the ensemble, $f_b(x_t)$ is the prediction of the bth regression tree for the predictor vector x_t, and α_b is the learning rate that is a form of shrinkage for a value less than 1.

The boosting regression tree model sequentially builds trees to correct the errors of the previous trees. At each iteration, the tree is built using the residual errors of the previous trees as the dependent variable and the learning rate scales the update of the model. One of the critical hyperparameters is the choice of number of learning cycles, i.e., the number of trees, to include in the model.

2.2.3 Support Vector Regression

Support Vector Machines (SVM) have been proposed as an approach for classification in the 1990s (Cortes and Vapnik (1995),Boser et al. (1992), Vapnik (1999)) and have gained some popularity. The approach can be also adapted for regression with a quantitative response, Support Vector Regression (SVR), that inherits some of the properties of the SVM classifier (see, e.g., Hastie et al., 2009). In the case of SVR with a linear kernel, the model aims to fit a linear function

$$f(x_t) = x_t^{'}\beta$$

to the data. To estimate the weight matrix β, we want to minimize

$$H(\beta) = C\sum_{t=1}^{T} V(y_t - x_t^{'}\beta) + \frac{1}{2}||\beta||^2, \tag{11}$$

whereas

$$V(y_t - x_t^{'}\beta) = \begin{cases} 0 & \text{if } |y_t - x_t^{'}\beta| < \epsilon \\ |y_t - x_t^{'}\beta| - \epsilon, & \text{otherwise.} \end{cases}$$

The first term in equation (11) represents the loss function for the t-th data point and quantifies the difference between the predicted and the actual target value. The second term represents a regularization term that prevents overfitting. This method requires the parameters ϵ and C to be chosen, whereas ϵ is the threshold parameter of the loss function and C is a regularization parameter. Note that the error measure ignores residuals of size less than ϵ. Thus, SVR can handle outliers and is relatively robust to overfitting.

3 Data and forecast setup

3.1 Data

To nowcast GDP growth in Switzerland we use a mixed-frequency data set of over 500 domestic and foreign indicators, covering the period from 2001 Q1 to 2019 Q4. The Swiss GDP is published quarterly by the State Secretariat for Economic Affairs (SECO) and annually by the Swiss Federal Statistical Office (FSO) with a considerable lag, making it important for policy makers to nowcast the current state of the economy in real-time. Our data set is comprised of both quarterly and monthly variables, with approximately 300 monthly and 200 quarterly variables used. These variables have been transformed to be stationary where necessary. This data set has been previously utilized in studies by Galli (2018) and Galli et al. (2019). Galli (2018) provides a comprehensive overview of the data set, categorizing the indicators into four main categories: hard data, soft data, foreign indicators, and financial indicators. The indicators within the data set cover a wide range of economic activity, including labor market, consumption, investment, foreign trade, foreign activity, financial markets, prices, as well as construction, retail trade, wholesale trade, accommodation, manufacturing, project engineering, banking, and insurance sectors. This diverse set of indicators allows for a comprehensive analysis of the Swiss economy and provides valuable insights into the factors that impact GDP growth.

3.2 Mixed-frequencies and ragged-edges

To address mixed-frequencies of our indicators, we use the blocking method that treats higher-frequency data as multiple lower-frequency series. In our case, this implies splitting the monthly data into three quarterly series, as illustrated by figure 1. The first quarterly series proxies quarterly observations with first months of each quarter (January, April, July, and October), the second series uses observations from the second month (February, May, August, and November), and the last collects observations from the third month (March, June, September, and December).

The blocking solution to mixed-frequency and ragged edge data problems has many advantages highlighted by Bec and Mogliani (2015). First, it is very flexible, as it can

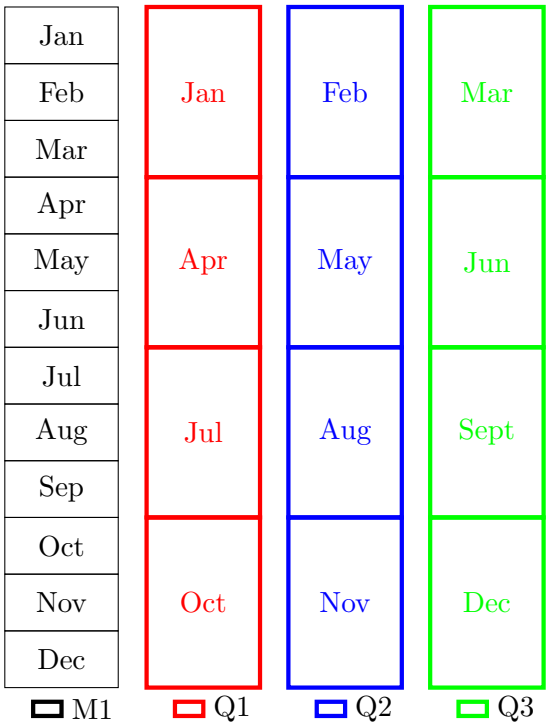


Figure 1: The blocking method treats higher-frequency data as multiple lower-frequency data. In our case, one monthly indicator becomes three quarterly series.

be handled in many possible modelling frameworks. Second, this approach allows the forecaster to exploit the partially available data directly at any time when nowcasting, without a need to extrapolate the missing information. Third, the blocking approach readily allows the forecaster to evaluate and interpret the signals provided by changes in the current economic activity in terms of GDP growth.

After having transformed the indicators into quarterly series using blocking, we realign the data set to adjust for missing values. Realignment is a technique proposed by Altissimo et al. (2010). Most of the indicators have different publication dates and therefore, not all of the observations are readily available in real-time.

In our setup, we focus on a specific information set within the quarter, namely, the beginning of the third month. While multiple states of data could be considered, we select this particular point as it reflects the most relevant nowcast situation for Switzerland. Institutions like the Swiss National Bank typically update their forecasts quarterly upon receiving new GDP figures. Hence, employing a realistic unbalanced dataset is essential to capture these publication lags effectively.

To deal with this so called "ragged-edge" phenomena, we apply a vertical realignment

technique in which the old series with a missing value is replaced by a realigned series. The process is illustrated by figure 2. We want to nowcast GDP growth (y) using various p indicators (x). For example, series x_2 is available with one lag and therefore, we replace the series x_2 with realigned series $\tilde{x}_2 = lag(x_2)$.

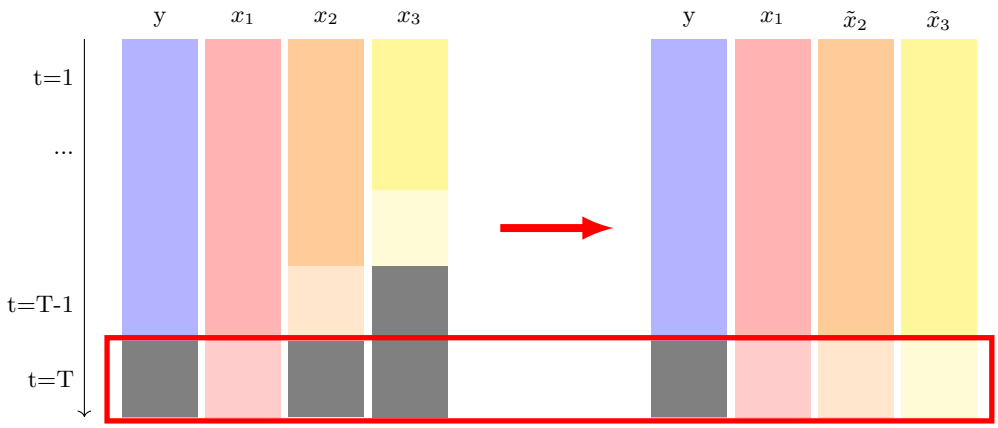


Figure 2: Colored columns represent available data, with each color representing one time series. Grey cells represent missing values and lighter colored cells represent last available observation. We want to make a forecast of y for time t=T and therefore, we realign all the indicators x to T. The lagged series replace the old ones.

3.3 Recursive nowcasting

The blocking and realignment yields a balanced quarterly panel of $p = 1139$ candidate indicators and $t = 76$ observations. We use this data set to perform recursive nowcasts for Swiss GDP growth from 2009 Q3 to 2019 Q4, with the last nowcast made for 2019 Q4 using data up to December 3rd, 2019. For most methods, including PCA, shrinkage methods, and SVR, we standardize all indicators in each nowcasting round, ensuring each indicator has a mean of zero and unit variance in our estimation sample. Figure 3 illustrates the recursive nowcasting process. For example, the first nowcast is done for 2009 Q3. We use data from 2001Q1 to 2009Q2 to train and test via cross-validation for optimal hyperparameters in our machine learning models, which we then use to nowcast out-of-sample 2009Q3. In the next period, we use data from 2001Q1 to 2009Q3 to train and test for the optimal model and subsequently, use it to nowcast 2009 Q4.

More precisely, data is divided into portion reserved for cross-validation and the last period reserved for an out-of-sample nowcast. Data part reserved for cross-validation consists of training and testing data. We use k-fold cross-validation to make decisions

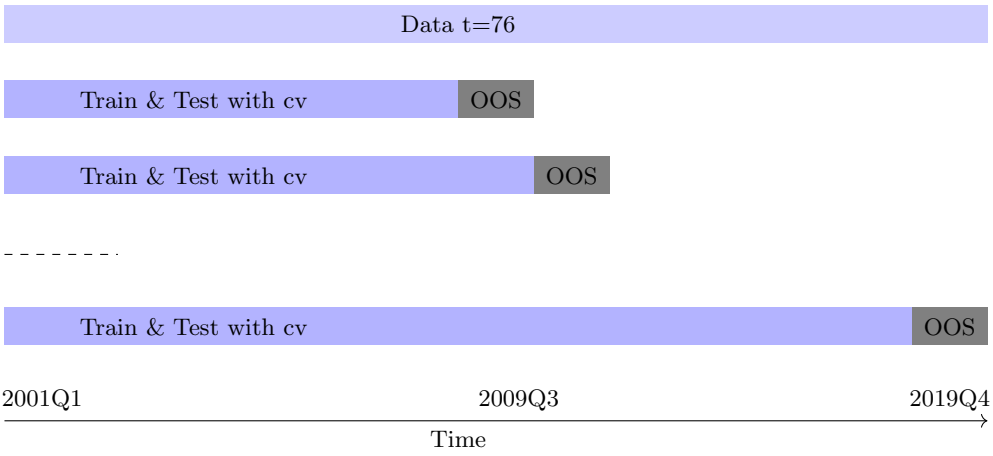


Figure 3: Graph illustrates recursive nowcasting. OOS stands for Out-Of-Sample nowcasting. The first OOS nowcast is made for 2009 Q3, whereas the last one is done for 2019 Q4. The blue area called "Train & Test" is used to select the optimal tuning of model used for OOS nowcasting. The model is chosen via cross-validation (CV).

about hyperparameters in our models. We will display the results for five-fold cross-validation. This involves dividing the cross-validation portion of the dataset into random five equal-sized subsets, or folds. The model is trained on four folds, while the remaining one fold is held back to serve as the validation set. This process is repeated five times, with each fold serving as the validation set once. By rotating the validation set across different folds, we obtain a robust estimate of the model's performance. We identify the optimal hyperparameters by minimizing forecast errors within the validation set. Table 1 provides an overview of the model specific hyperparameters that are described in section 2. The table summarizes the hyperparameters that we are tuning and their corresponding ranges from which we are searching for the optimal value via cross-validation[3]. Subsequently, we use the OOS period to evaluate the predictive accuracy of the fitted model with the optimal hyperparameters. More details on cross-validation are provided in Appendix A.4. All models are implemented in $\mathbb{R}$ with Appendix A.3 providing an exact overview of packages being used for estimation. For not specified parameters we use default value from respective $\mathbb{R}$ package.

[3]Please note that the equation (8) for the elastic net can be rewritten as $\sum_{t=1}^{T}(y_t - \beta_0 - \sum_{j=1}^{p}\beta_j x_{jt})^2 + \lambda(\alpha\sum_{j=1}^{p}|\beta_j| + (1-\alpha)\sum_{j=1}^{p}\beta_j^2)$, whereas the hyperparameters in equation (8) are defined as $\lambda_1 \equiv \lambda\alpha$ and $\lambda_2 \equiv \lambda(1-\alpha)$. For $\alpha = 0$, we have the ridge regression and for $\alpha = 1$ we have the LASSO regression.

Table 1: Hyperparameters Overview

Model	Hyperparameters	Range
LASSO	λ	0.0005 – 1
Ridge	λ	0.01 – 10
Elastic net	λ	0.01 – 10
	α	0.01 – 0.99
Bagging	No. of trees	1000
Random Forest	No. of trees	1000
	No. of predictors at each split	100 – 500
Boosting	No. of trees	500 – 1500
	Learning rate	0.001 – 0.1
	Tree depth	1 – 5
SVR	C	0.01 – 2
	ϵ	0.01 – 0.3

Notes: The table reports the hyperparameters (column 2) of various models that are estimated with five-fold cross-validation from ranges reported in column 3.

3.4 Forecast evaluation

We compare the nowcasting accuracy of machine learning models to benchmark models. Our primary evaluation metric is the RMSE, a common measure of forecasting accuracy, calculated as:

$$RMSE = \sqrt{\frac{\sum_t^T (y_t - \hat{y}_t)^2}{T}} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} \hat{e}_t^2}, \tag{12}$$

whereas y_t is the realized GDP growth, $\hat{y}_t$ is the nowcast by the respective model, T is the number of out-of-sample nowcasts, and $\hat{e}_t$ denotes the forecast error of the model. To gain an overview of average forecast properties of the different ML models, we look at relative RMSEs and compute the average forecast gains in percentage terms of each ML method compared to the AR and PCA models. For robustness, we also consider A.1.3 another loss function, mean absolute error (MAE), to compare the nowcasting accuracy of our considered models in the appendix. The MAE is given by

$$MAE = \frac{1}{T}\sum_{i=1}^{T}|y_i - \hat{y}_i|. \tag{13}$$

Additionally, we evaluate whether the ML model forecasts are systematically better than the benchmarks. This evaluation must be judged by statistical tests of equal predictive ability. We will use the Diebold and Mariano (1995) test to compare the significance of performance of each model to the AR and PCA benchmarks.

Moreover, we look at the cumulative differences in squared forecast errors (CDSFE) between the best performing ML models and the PCA. This allows us to investigate the predictive gains over time in our evaluation sample. The CDSFE of model i at time t is defined as

$$CDSFE_t = \sum_{s=1}^{t}\hat{e}_{s,i}^2 - \sum_{s=1}^{t}\hat{e}_{s,PC}^2. \tag{14}$$

A negative value of $CDSFE_t$ indicates that the forecasting model under examination outperforms the PCA model by reporting a smaller value of cumulative squared errors.

Lastly, we check whether the forecasts of the PCA and each of the ML model encompass each other. If the PCA forecast model encompasses each ML forecast model, then it implies that the PCA provides all the necessary information and that a combination with the ML model does not lead to any further improvement. Then we would conclude that the PCA model is superior and the ML model can be classified as redundant. On the contrary, if the ML forecast model encompasses the PCA model, then it implies that the ML model is clearly superior. Encompassing in a forecasting concept can be illustrated by the following regression:

$$y_t = \lambda\hat{y}_{t,i} + (1-\lambda)\hat{y}_{t,PC} + u_t, \tag{15}$$

where $\hat{y}_{t,PC}$ denotes the forecast from the PCA model, and $\hat{y}_{t,i}$ is the forecast from ML model i. $\lambda = 0$ then implies that the PC model encompasses the ML model i and $\lambda = 1$ implies that the ML model i encompasses the PC model (Chong and Hendry, 1986). Generally, λ is the optimal weight of the combined forecast of model i and model PC. If λ is somewhere between zero and one, then a combined forecast is better than one of

the individual forecast. To implement this test, we employ Harvey et al. (1998)'s forecast encompassing test. It is based on a simple transformation of equation (15) and equals

$$\hat{e}_{t,PC} = \lambda(\hat{e}_{t,PC} - \hat{e}_{t,i}) + v_t, \tag{16}$$

which is expressed in terms of forecast errors.

4 Results

We compare the nowcasting performance of various models described in section 2 using forecast evaluation methods described in section 3.

4.1 Baseline specification

Table 2 provides a first overview of results. In the first column, the RMSEs from the recursive nowcasting exercise are displayed, whereas the second and third columns report the relative gains/losses in terms of RMSE to benchmark models AR and PCA. We use the Diebold and Mariano (1995) test to assess the significance of difference in nowcasting accuracy. The stars in columns two and three represent the significance levels of one-sided Diebold-Mariano test, whereas the exact corresponding p-values can be found in appendix A.2.

Table 2: Results Overview

Model	RMSE	Benchmark comparison	
		AR	PCA
AR	0.421		25%
PCA	0.337	-20%**	
FSS	0.486	15%	44%
LASSO	0.315	-25%**	-6%
Ridge	0.310	-26%**	-8%*
Elastic Net	0.303	-28%***	-10%**
Bagging	0.354	-16%**	5%
Random Forest	0.353	-16%**	5%
Boosting	0.339	-19%**	1%
SVR	0.308	-27%***	-8%*

Notes: The table reports in column 2 the RMSE of models described in section 2. The models were estimated via five-fold cross-validation. Columns 3 and 4 show relative gains/losses in terms of RMSE compared to benchmark models AR and PCA, respectively. The stars represent the significance levels of one-sided Diebold-Mariano test: *** significant at 1%, ** significant at 5%, * significant at 10%.

All machine learning models considered reduce the RMSE compared to the AR benchmark with all of them significantly outperforming the AR benchmark. Especially, SVR with a linear kernel and elastic net do so by improving RMSE by 27% and 28%, respec-

tively. When it comes to our strong benchmark PCA, only the linear methods (LASSO, ridge, elastic net, and SVR) improve the nowcasting accuracy relative to it. However, it is important to note that ridge, elastic net, and SVR do so on a significant level, meaning that there are nowcasting gains to be had from these methods. The elastic net improves the PCA benchmark by 10%, whereas ridge and SVR with a linear kernel both improve the accuracy by 8%.

We see that regression based methods (LASSO, ridge, elastic net, and SVR) with a linear kernel perform better than the non-linear tree methods, which suggests a linear relationship between indicators and GDP. This argument is strengthened by the result that the SVR with a linear kernel outperforms the SVR with other kernels such as polynomial and radial. More detailed results considering the SVR with other kernels are shown in appendix A.1.2.

Furthermore, ridge regression performing better than LASSO in both baseline version shown in table 2 as well as lagged version displayed in appendix A.1.1 seems to suggest that many of the predictors considered matter roughly in equal size and there are not any substantially more important predictors. This is in line with Galli et al. (2019), who showed that a large data set is essential for obtaining precise nowcasts of Swiss GDP. This suggests that for Switzerland, as a small open economy, many very different indicators matter, unlike for an economy such as the US.

4.2 Additional results

Stability over time

To investigate the sensitivity of our baseline results, we look at the forecast performance of the most promising models over time. The cumulative difference in squared forecast errors (CDSFE) is computed between each of the three best performing ML models and the PCA benchmark and plotted against the forecasting horizon to visualize the error accumulation over time. Figure 4 illustrates the difference in cumulative errors over time. It plots the CDSFEs of ridge, elastic net, and SVR with a linear kernel relative to the PCA against the nowcasting horizon. These relative cumulative errors remain below zero over the entire forecast period and they decline uniformly. This indicates that the ML

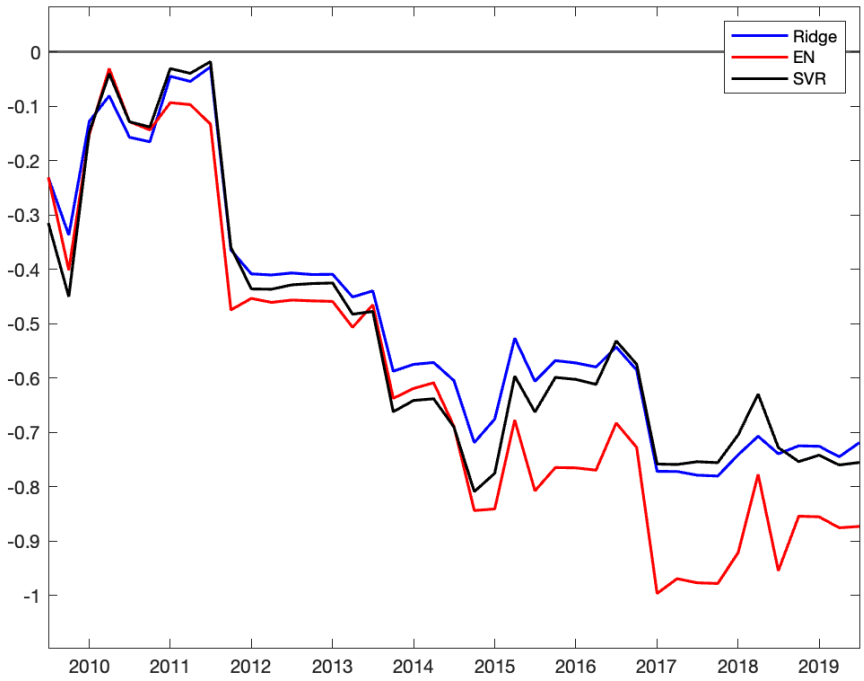


Figure 4: Cumulative difference in squared forecast errors (CDSFE) between the three best performing ML models and the PCA benchmark.

models show a stable improvement in nowcasting accuracy over the PCA method.

Do ML methods encompass traditional nowcast methods?

Next, we investigate whether some of the ML models outperform the benchmark PCA in terms of encompassing. Table 3 shows the corresponding results. Interestingly, ridge, SVR, and elastic net seem to encompass the PCA model. This is shown by a significant λ that is close to 1. It implies that those models already contain all information for the GDP nowcast relative to the PCA. This indicates that a combination of those models with PCA will not lead to further improvements in terms of forecast ability. On the other hand, tree methods (e.g., bagging and random forest) are not able to improve the PCA. This implies that those methods do not add any information to the PCA benchmark. LASSO and boosting are the two methods, where a combination between ML and traditional PCA models seems to improve the GDP nowcast relative to using a single method.

Table 3: Encompassing Test

Test regression: $y_t = \lambda \hat{y}_{t,i} + (1-\lambda)\hat{y}_{t,PC} + u_t$			
Model	λ	p-values	
		$H_0 : \lambda = 0$	$H_0 : \lambda = 1$
Ridge	0.918	0.001	0.755
LASSO	0.613	0.000	0.002
Elastic Net	0.748	0.000	0.099
Bagging	0.324	0.250	0.015
Random Forest	0.330	0.209	0.010
Boosting	0.473	0.060	0.031
SVR	0.791	0.001	0.364

Notes: The table reports the results of encompassing test. We test whether a specific ML model encompasses the PCA. λ is the optimal weight of the combined forecast of the ML and PCA models.

Can the traditional methods be improved by cross validation?

In our benchmark models (PCA and FSS), described in section 2, we decided the critical parameters using the BIC. The BIC is based on the available data and may not guarantee optimal performance on out-of-sample data. Of course, we could use cross-validation that directly estimates model's performance on unseen data to choose the number of principal components q included in the PCA and the number of predictors Q included in the FSS[4]. Therefore, we apply the same five-fold cross-validation procedure as we did with ML methods.

Table 4: Benchmark models with cv

Model	RMSE		gains
	BIC	cv	in %
PCA	0.337	0.317	-5.9
FSS	0.486	0.408	-16.0

Notes: The table compares the RMSE of the benchmark models, PCA and FSS, for different choice methods of parameters. Crucial parameters were selected with either the BIC (column 2) or five-fold cross-validation (column 3). Column 4 reports the relative gain/loss.

[4]Please note that five-fold cross-validation was purposefully not used for the AR benchmark. For AR model, the time series dimension is of critical importance and the five-fold cross-validation applied here does not respect the temporal structure of the data.

Results for this exercise can be seen in table 4. We see that determining these crucial parameters by learning from the data improves the nowcasting accuracy. However, the gain is relatively modest for the factor model approach (i.e., PCA), where the RMSEs decline by around 6% and not fully converge to the best ML methods. The FSS with cross-validation improves by 16% compared to the BIC based version, which indicates that the standard forward selection with an information criterion based stopping rule suffers from overfitting. The version with cross-validation is now better than the univariate benchmark but still worse than all other ML procedures. Generally, cross-validation seems to be a good and more robust alternative to information criteria in finding an appropriate factor model or to determine the optimal stopping rule in forward selection.

5 Conclusion

In this paper, we explored the potential of machine learning algorithms to enhance the accuracy of GDP nowcasting using a comprehensive set of economic indicators for Switzerland. We addressed the challenges posed by mixed frequencies and ragged edges through the use of blocking and realignment techniques. By optimizing various shrinkage parameters through cross-validation, we ensured robust model performance.

Our empirical analysis demonstrated that machine learning methods significantly outperform traditional autoregressive benchmarks. Specifically, the elastic net model achieved a 28% improvement in nowcasting accuracy, as measured by RMSE. Ridge regression, elastic net, and support vector regression with a linear kernel emerged as the best-performing models, consistently surpassing the principal component analysis benchmark.

Contrary to our initial expectations, linear machine learning algorithms outperformed their non-linear counterparts in the context of GDP nowcasting. This finding underscores the critical role of algorithm selection and parameter tuning when leveraging large macroeconomic indicator sets.

The cumulative differences in squared forecast errors between our best-performing models and the PCA benchmark highlight the stability and reliability of machine learning approaches in enhancing nowcasting accuracy. These results suggest that non-linearity is not the primary driver of machine learning methods' superiority over traditional techniques in this domain.

Overall, our study contributes to the growing body of literature on the application of machine learning in economic forecasting. We provide evidence that machine learning algorithms can offer substantial improvements in the accuracy of GDP nowcasts, providing policymakers with more timely and reliable economic insights. Future research could further explore the integration of additional data sources and the application of other advanced machine learning techniques to continue improving the precision of economic forecasts.

References

Altissimo, F., R. Cristadoro, M. Forni, M. Lippi, and G. Veronese (2010): "New Eurocoin: Tracking economic growth in real time," *The review of economics and statistics*, 92, 1024–1034.

Bańbura, M., D. Giannone, M. Modugno, and L. Reichlin (2013): "Nowcasting and the real-time data flow," in *Handbook of economic forecasting*, Elsevier, vol. 2, 195–237.

Bec, F. and M. Mogliani (2015): "Nowcasting French GDP in real-time with surveys and "blocked" regressions: Combining forecasts or pooling information?" *International Journal of Forecasting*, 31, 1021–1042.

Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992): "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152.

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

——— (2001): "Random forests," *Machine learning*, 45, 5–32.

Chakraborty, C. and A. Joseph (2017): "Machine learning at central banks," .

Chinn, M. D., B. Meunier, and S. Stumpner (2023): "Nowcasting World Trade with Machine Learning: a Three-Step Approach," Tech. rep., National Bureau of Economic Research.

Chong, Y. Y. and D. F. Hendry (1986): "Econometric evaluation of linear macroeconomic models," *The Review of Economic Studies*, 53, 671–690.

Cortes, C. and V. Vapnik (1995): "Support-vector networks," *Machine learning*, 20, 273–297.

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

Döpke, J., U. Fritsche, and C. Pierdzioch (2017): "Predicting recessions with boosted regression trees," *International Journal of Forecasting*, 33, 745–759.

Evans, M. D. (2005): "Where Are We Now? Real-TIme Estimates of the Macroeconomy," *International Journal of Central Banking*.

Friedman, J., R. Tibshirani, and T. Hastie (2010): "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, 33, 1–22.

Friedman, J. H. (2001): "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, 1189–1232.

Galli, A. (2018): "Which indicators matter? Analyzing the Swiss business cycle using a large-scale mixed-frequency dynamic factor model," *Journal of Business Cycle Research*, 14, 179–218.

Galli, A., C. Hepenstrick, and R. Scheufele (2019): "Mixed-frequency models for tracking short-term economic developments in Switzerland," *International Journal of Central Banking*, 15, 151–178.

Giannone, D., L. Reichlin, and D. Small (2008): "Nowcasting: The real-time informational content of macroeconomic data," *Journal of monetary economics*, 55, 665–676.

Harvey, D. I., S. J. Leybourne, and P. Newbold (1998): "Tests for forecast encompassing," *Journal of Business & Economic Statistics*, 16, 254–259.

Hastie, T., R. Tibshirani, and J. H. Friedman (2009): *The elements of statistical learning: data mining, inference, and prediction*, vol. 2, Springer.

Heinisch, K. and R. Scheufele (2018): "Bottom-up or direct? Forecasting German GDP in a data-rich environment," *Empirical Economics*, 54, 705–745.

Hoerl, A. E. and R. W. Kennard (1970): "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, 12, 55–67.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2013): *An introduction to statistical learning*, vol. 112, Springer.

Kronenberg, P., H. Mikosch, S. Neuwirth, M. Bannert, and S. Thöni (2023): "The Nowcasting Lab: Live Out-of-Sample Forecasting and Model Testing," *Available at SSRN 4353052*.

Kuhn and Max (2008): "Building Predictive Models in R Using the caret Package," *Journal of Statistical Software*, 28, 1–26.

Kuzin, V., M. Marcellino, and C. Schumacher (2013): "Pooling versus model selection for nowcasting GDP with many predictors: Empirical evidence for six industrialized countries," *Journal of Applied Econometrics*, 28, 392–411.

Liaw, A. and M. Wiener (2002): "Classification and Regression by randomForest," *R News*, 2, 18–22.

Richardson, A., T. van Florenstein Mulder, and T. Vehbi (2021): "Nowcasting GDP using machine-learning algorithms: A real-time assessment," *International Journal of Forecasting*, 37, 941–948.

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

Smalter Hall, A. and T. R. Cook (2017): "Macroeconomic indicator forecasting with deep neural networks," Working Paper 17-11, Federal Reserve Bank of Kansas City.

Stock, J. H. and M. W. Watson (2002a): "Forecasting Using Principal Components from a Large Number of Predictors," *Journal of the American Statistical Association*, 97, 1167–1179.

——— (2002b): "Macreconomic forecasting using diffusion indexes," *Journal of Business and Economic Statistics*, 20, 147–162.

Tibshirani, R. (1996): "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58, 267–288.

Vapnik, V. (1999): *The nature of statistical learning theory*, Springer science & business media.

Zou, H. and T. Hastie (2005): "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67, 301–320.

A Appendix

A.1 Additional Results

In this section, we report the results of various additional modifications of the baseline version displayed in section 4.

A.1.1 Additional Results: Including lagged predictors

We allow for a lag of each predictor p to be included in ML models. We re-estimate the machine learning models with setup described above, but now also include a lag of order one from all of the predictors p. This means that we have a panel of $t = 75$ observations and $\tilde{p} = 2278$ potential predictors. Table 5 reports the overview of results.

Table 5: Additional Results Overview: Including a lag of each p predictor.

Model	RMSE
Ridge	0.326
LASSO	0.376
Elastic Net	0.317
Bagging	0.360
Random Forest	0.364
Boosting	0.345
SVR	0.308

Notes: The table reports in column 2 the RMSE of ML models including lagged predictors. The models were estimated via five-fold cross-validation.

We see that all results are worse compared to the baseline version. However, we notice that, again, the best performing models are the SVR with a linear kernel, elastic net, and ridge. The relative performance of the models compared to each other is consistent with our baseline results that do not include lagged predictors.

A.1.2 Additional Results: SVR other kernels

For the SVR, it is important to choose the correct kernel. We consider an SVR with a linear, polynomial, and radial kernel. Using a polynomial kernel involves choosing the degree of the polynomial. Firstly, we consider a polynomial of degree 2 and secondly, we

let cross-validation choose the degree of the polynomial[5]. Table 6 provides an overview of results for these different kernels[6]. The SVR with linear kernel outperforms all others. This is in line with our results in section 4, further suggesting a linear relationship between predictors and GDP.

Table 6: Additional Results Overview: SVR kernels

Model	RMSE
SVR linear kernel	0.308
SVR polynomial kernel (d=2)	0.402
SVR polynomial kernel (d with cv)	0.402
SVR radial kernel	0.340

Notes: The table reports in column 2 the RMSE of various SVR models with different kernels estimated via five-fold cross-validation.

A.1.3 Additional Results: Mean Absolute Error

To check robustness, we also want to consider another loss function to quantify the accuracy of our forecasts than the squared errors loss function discussed in section 4.1. Table

Table 7: Results Overview

Model	MAE	Benchmark comparison	
		AR	PCA
AR	0.313		33%
PCA	0.236	-25%***	
FSS	0.370	18%	57%
Lasso	0.249	-20%*	5%
Ridge	0.228	-27%***	-3%
Elastic Net	0.223	-29%***	-5%
Bagging	0.268	-14%**	13%
Random Forest	0.266	-15%**	13%
Gradient boosting	0.246	-21%***	4%
SVR	0.232	-26%***	-2%

Notes: The table reports in column 2 the MAE of models described in section 2. The models were estimated via five-fold cross-validation. Columns 3 and 4 show relative gains/losses in terms of MAE compared to the AR and PCA benchmark models, respectively. The stars represent the significance levels of one-sided Diebold-Mariano test: *** significant at 1%, ** significant at 5%, * significant at 10%.

[5]We have an additional hyperparameter: degree (d). We search for the optimal value of d from a range $2-4$ using cross-validation.

[6]Please note that the optimal choice for the degree of the polynomial using cross-validation is $d = 2$ in each period and therefore, the results are identical to SVR with polynomial of degree 2.

7 provides an overview of the MAEs. In the first column, the MAEs from the recursive nowcasting exercise are displayed, whereas the second and third columns report the relative gain/loss in terms of MAE to the AR and PCA benchmark models. We see that the overall pattern of results stays largely intact. The elastic net is still the best performing model, closely followed by the SVR and ridge models. Again, all ML methods reduce the MAE compared to AR benchmark on a significant level. On the other hand, elastic net, SVR, and Ridge slightly improve the nowcasting accuracy in terms of MAE compared to the strong PCA benchmark, but the effect is not statistically significant.

A.2 Diebold and Mariano test

Table 8: Diebold and Mariano (1995) one-sided test p-values

	AR	PCA
AR		0.96
PCA	0.04	
FSS	0.87	0.99
LASSO	0.04	0.20
Ridge	0.02	0.06
ElasticNet	0.01	0.05
Bagging	0.05	0.74
RF	0.03	0.75
Boosting	0.02	0.54
SVR	0.01	0.10

Notes: The table reports the p-values of one sided Diebold and Mariano test with squared errors loss function.

Table 8 reports the p-values of one sided Diebold and Mariano test corresponding to the results shown in table 2 in section 4 that reports the RMSEs. The Diebold and Mariano test is a statistical test used to evaluate the accuracy and statistical significance of forecast differences between two competing forecasting models. The loss function applied to estimate the significance of forecast differences is squared errors. More precisely, we compare the forecast accuracy of each method versus that of the AR benchmark first and then versus that of the PCA benchmark second.

Table 9: Diebold and Mariano (1995) one-sided test p-values

	AR	PCA
AR		0.99
PCA	0.01	
FSS	0.85	1.00
Lasso	0.08	0.68
Ridge	0.01	0.33
ElasticNet	0.01	0.29
Bagging	0.05	0.93
RF	0.03	0.93
Boosting	0.01	0.71
SVR	0.01	0.43

Notes: The table reports the p-values of one sided Diebold and Mariano test with absolute error loss function.

Whereas table 8 applies the squared error loss function to calculate the significance of forecast differences, table 9 reports the p-values of one sided Diebold and Mariano test corresponding to the absolute error loss function shown in table 7 in section A.1.3. Again, we compare the forecast accuracy of each method versus that of the AR benchmark first and then versus that of the PCA benchmark second.

A.3 Packages used in $\mathbb{R}$

All our models are estimated in $\mathbb{R}$ using following packages.

We use the package **caret** by Kuhn and Max (2008) to perform cross-validation. This package provides a unified interface for training and evaluating a wide range of machine learning models.

Package **glmnet** by Friedman et al. (2010) is used to estimate ridge, LASSO, and elastic net. It provides functions for fitting generalized linear models with regularization including ridge, LASSO, and elastic net regression.

Both bagging and random forest are estimated using package **randomForest** by Liaw and Wiener (2002). For bagging the parameter *mtry* equals p, which means that it considers all predictors at each split of the tree, whereas for random forest this parameter is set to $m < p$, which means that only a random set m of p predictors is considered at each split.

Boosting method is implemented using package **gbm** from https://CRAN.R-project.org/package=gbm.

SVR is estimated via package **e1071** from https://CRAN.R-project.org/package=e1071

A.4 Cross-validation

For the benchmark models above, we used the BIC to make critical decisions such as the lag order in AR model or the number of principal components in PCA analyses. The BIC is based on the likelihood function of the model and penalizes models with more parameters. It is useful because it provides a quantitative measure of the trade-off between model complexity and goodness of fit, which can help to avoid overfitting. However, the BIC, like other model selection criteria, is based on the available data and does not take into account the model's performance on new, unseen data. Therefore, selecting a model based solely on the BIC may not guarantee optimal performance on out-of-sample data.

We want to minimize the MSE, which is a commonly used metric to evaluate the performance of a machine learning model. It measures the average squared difference between the predicted and true values across all instances in the dataset. The MSE can be expressed as:

$$\text{MSE} = \mathbb{E}[(\hat{\theta} - \theta)^2] = \text{Var}(\hat{\theta}) + \text{Bias}(\hat{\theta}, \theta)^2, \tag{17}$$

where $\hat{\theta}$ is an estimator of the true parameter θ, $\mathbb{E}$ denotes the expected value, Var is the variance, and Bias is the bias of the estimator.

The equation (17) captures the bias-variance trade-off, a fundamental concept in machine learning that involves balancing two sources of error: bias and variance. Bias refers to the tendency of a model to consistently make erroneous predictions in the same direction. It can be thought of as the difference between the expected prediction of the model and the true value. A high bias model is typically oversimplified and cannot capture the complexity of the data, leading to underfitting. Variance, on the other hand, refers to the tendency of a model to make erratic predictions based on small fluctuations in the training data. It can be thought of as the variability of model predictions for different training sets. A high variance model is typically overfitted and too complex, capturing the noise in the data and unable to generalize well to new data. A model with high bias will have a high MSE due to its underfitting, whereas a model with high variance will also have a high MSE due to its overfitting. The ideal model strikes a balance between bias and variance, leading to a low MSE.

We care about something called the test MSE that measures the model's performance on data that was not used to fit the model and is a more accurate assessment of its ability to generalize than the training MSE (in-sample MSE). While the training MSE is a useful metric for evaluating how well a model fits the training data, it is not necessarily a good indicator of how well the model will perform out-of-sample. This is because a model can potentially be overfit to the training data by memorizing the noise and idiosyncrasies of the training set, resulting in a low training MSE but a high test MSE. In practice, the test MSE is often estimated by splitting the available data into training and testing sets, fitting the model on the training set, and evaluating its performance on the test set. This allows us to get a sense of how well the model is likely to perform on new data before deploying it in the real world. A large gap between the training error and the test error indicates overfitting, while a high test error indicates underfitting or poor model choice.

Cross-validation is a resampling technique that directly estimates the model's performance on unseen data and allows us to compare different models based on their performance on testing data. The basic idea of cross-validation is to split the available data into two sets: a training set and a validation set. The model is trained on the training set and then evaluated on the validation set to estimate its performance. This process is repeated multiple times, with different random splits of the data, to obtain an average estimate of the model's performance. Therefore, we will use cross-validation to make decisions about hyperparameters in our models.

Recent SNB Working Papers

2024-06 Milen Arro-Cannarsa, Rolf Scheufele:
Nowcasting GDP: what are the gains from machine learning algorithms?

2024-05 Jessica Gentner:
The role of hedge funds in the Swiss franc foreign exchange market

2024-04 Tobias Cwik, Christoph Winter:
FX interventions as a form of unconventional monetary policy

2024-03 Lukas Voellmy:
Decomposing liquidity risk in banking models

2024-02 Elizabeth Steiner:
The impact of exchange rate fluctuations on markups – firm-level evidence for Switzerland

2024-01 Matthias Burgert, Johannes Eugster, Victoria Otten:
The interest rate sensitivity of house prices: international evidence on its state dependence

2023-08 Martin Brown, Laura Felber, Christoph Meyer:
Consumer adoption and use of financial technology: "tap and go" payments

2023-07 Marie-Catherine Bieri:
Assessing economic sentiment with newspaper text indices: evidence from Switzerland

2023-06 Martin Brown, Yves Nacht, Thomas Nellen, Helmut Stix:
Cashless payments and consumer spending

2023-05 Romain Baeriswyl, Alex Oktay, Marc-Antoine Ramelet:
Exchange rate shocks and equity prices: the role of currency denomination

2023-04 Jonas M. Bruhin, Rolf Scheufele, Yannic Stucki:
The economic impact of Russia's invasion of Ukraine on European countries – a SVAR approach

2023-03 Dirk Niepelt:
Payments and prices

2023-02 Andreas M. Fischer, Pınar Yeşin:
The kindness of strangers: Brexit and bilateral financial linkages

2023-01 Laura Felber, Simon Beyeler:
Nowcasting economic activity using transaction payments data

2022-14 Johannes Eugster, Giovanni Donato:
The exchange rate elasticity of the Swiss current account

2022-13 Richard Schmidt, Pınar Yeşin:
The growing importance of investment funds in capital flows

2022-12 Barbara Rudolf, Pascal Seiler:
Price setting before and during the pandemic: evidence from Swiss consumer prices

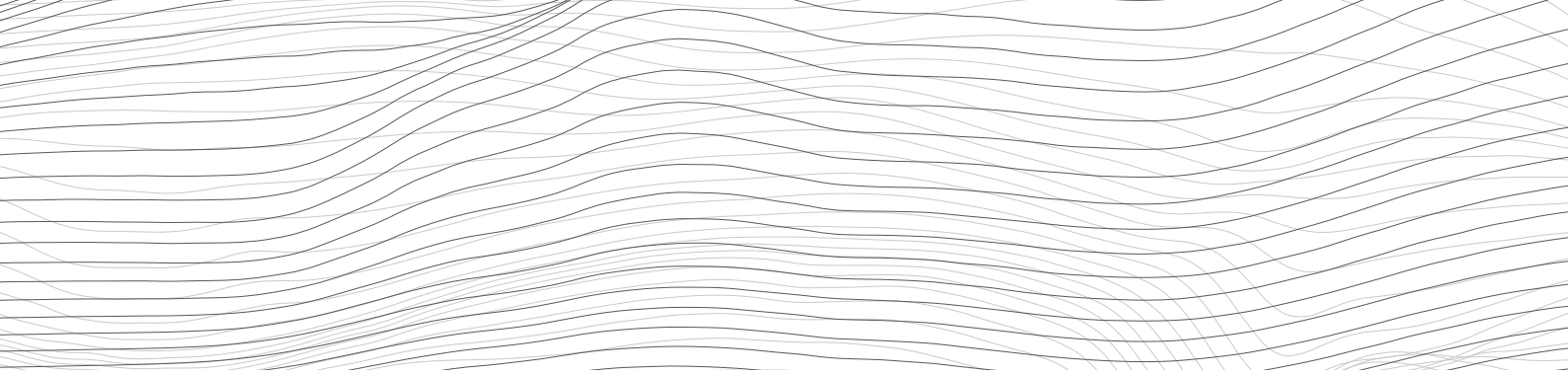
2022-11 Yannic Stucki:
Measuring Swiss employment growth: a measurement-error approach

2022-10 Lukas Altermatt, Hugo van Buggenum, Lukas Voellmy:
Systemic bank runs without aggregate risk: how a misallocation of liquidity may trigger a solvency crisis

2022-09 Andrada Bilan, Yalin Gündüz:
CDS market structure and bond spreads

2022-08 Diego M. Hager, Thomas Nitschka:
Responses of Swiss bond yields and stock prices to ECB policy surprises

2022-07 Gregor Bäurle, Sarah M. Lein, Elizabeth Steiner:
Firm net worth, external finance premia and monitoring cost – estimates based on firm-level data



SCHWEIZERISCHE NATIONALBANK
BANQUE NATIONALE SUISSE
BANCA NAZIONALE SVIZZERA
BANCA NAZIUNALA SVIZRA
SWISS NATIONAL BANK