Working Paper

No. 157 March 2025

Forecasting Quarterly GDP Growth Rates Using Machine Learning Methods

By Kristian Jönsson

National Institute of Economic Research

Sammanfattning

Indikatorer från Konjunkturinstitutets Konjunkturbarometer används ofta i olika indikatormodeller för att ta fram modellbaserade kortsiktsprognoser för den ekonomiska utvecklingen. Barometerdata kan också användas för att ta fram prognoser med hjälp av maskininlärningsmodeller. I denna studie undersöker vi prognosförmågan hos olika maskininlärningsmodeller som använder barometerdata för kortsiktsprognoser av den kvartalsvisa BNP-tillväxten. Vi jämför resultaten med prognosförmågan hos linjära indikatormodeller. Jämförelsen visar att vissa trädbaserade modeller (Random Forest och Gradient-Boosted Regression Trees) samt ett enkelt neuralt nätverk (Multi-Layer Perceptron) producerar prognoser som står sig väl i relation till andra modellers prognoser.

Forecasting Quarterly GDP Growth Rates Using Machine Learning Methods

Kristian Jönsson*

March 6, 2025

Abstract

Predicting current and near-term macroeconomic developments using linear indicator models and Economic Tendency Survey data is standard nowcasting practice. In the current paper, we investigate whether machine learning methods, when used together with survey data, can improve the forecasts of Swedish quarterly GDP compared to baseline linear models. The results indicate that machine learning methods generally perform well compared to linear baseline models. In particular, Gradient-Boosted Regression Trees, Random Forests, and Multi-Layer Perceptron models are identified as some of the best-performing models.

1 Introduction

Forecasting economic developments is important for a number of reasons, not least for planning and policy purposes. When producing economic forecasts that span longer time horizons, dynamic relationships and interactions in the economy are often utilized. However, since statistics on many important economic quantities are time-consuming to produce and are often published with a considerable delay, the outcomes that constitute the starting point of the longer-term forecasts are frequently uncertain well beyond the beginning of the forecasting period. In order for the longer-term forecasts to depart from the best position possible, extra care is therefore often taken to predict the current stance and the shorter-term developments of the economy. Since there are various indications of the current stance of the economy, taking these

*This article was written while the author was leading the project and workshop series "Machine Learning Methods for Macroeconomic Forecasting" at the National Institute of Economic Research (NIER). The author is grateful for discussions with and comments from workshop and seminar participants at the NIER. The responsibility for the remaining shortcomings rests solely with the author. Also, the opinions expressed in the article are the sole responsibility of the author and should not be interpreted as reflecting the views of Sveriges Riksbank.

data into account can provide a better understanding of both the current economic environment and how the economy is going to develop in the short term. Employing well-performing models and methods for economic nowcasting and short-term forecasting can thus improve the quality of longer-term predictions by providing them with a more robust point of departure. Since the indications on the current state of the economy are often volatile and continuously change as new shocks impact the economy, they seldom contain information that is useful for longer-term economic forecasts and are typically used only for making nowcasts and short-term forecasts covering the current, and perhaps the upcoming, quarter.

One useful source of information when it comes to gaining timely insights into economic developments is Economic Tendency Surveys (ETS). In such surveys, businesses and households are asked a set of questions. By processing the answers in such a way that the information they contain becomes available for nowcasting and short-term forecasting prior to the publication of statistics for macroeconomic aggregates, it is possible to gain early insights into the economic situation. The tendency survey series are often useful when formalizing predictions through the use of models (see e.g. Kaufmann and Scheufele, 2017, and the references therein for a discussion). In Sweden, the National Institute of Economic Research publishes the results of a comprehensive tendency survey, and the indices and indicators from this survey have been frequently used in various forecasting models, see for example Hansson et al. (2005), Österholm (2014), Billstam et al. (2017) and den Reijer and Johansson (2019).

When the tendency survey data series are used for model-based nowcasting or short-term forecasting, it is often the case that the series are included in basic linear models, or models that are closely related to the basic linear models such as vector autoregressive models, mixed-frequency models, or dynamic factor models. Recently, however, more and more forecasting endeavors have resorted to machine learning (ML) methods to produce nowcasts and short-term forecasts of macroeconomic quantities. In order to forecast production in the economy, Richardson et al. (2018) and Richardson et al. (2021) use a large set of predictors and a set of different machine learning models to predict GDP for New Zealand and find that machine learning methods can be used to improve forecasts compared to commonly used benchmark models. A similar approach is taken by Kant et al. (2024), who find that the so-called Random Forest method can be useful for forecasting GDP. Arro-Cannarsa and Scheufele (2024) also investigate an array of machine learning models for nowcasting GDP and find that linear machine learning models work well. Specific machine learning methods have been investigated, for example, by Yoon (2021) and Soybilgen and Yazgan (2021), who find that tree-based methods, such as gradient-boosted trees and random forests, work well when forecasting GDP. The K Nearest Neighbor method has been investigated for GDP forecasting by Jönsson (2020), Jönsson (2021) and Jönsson (2024). Investigations of the ability of machine learning models to forecast other macroeconomic quantities, such as labor market variables and inflation, have been carried out by, for example, Smalter Hall (2018) and Medeiros et al. (2021).

The current paper contributes further to the literature on GDP nowcasting using machine learning (ML) methods by investigating a wide set of ML models and studying their usefulness for nowcasting Swedish GDP growth using the Economic Tendency Survey (ETS) of the National Institute of Economic Research (NIER). Seven different confidence indicators are employed, and a set of baseline linear models is also included in the study and compared to the set of ML models in terms of nowcasting performance.

The results obtained in this paper show that the average performance of the machine learning models is better than the average performance of the baseline linear models. This applies to all the forecast performance metrics considered, (more specifically, the mean forecast error, the mean absolute forecast error, and the mean squared forecast error) and across four different nowcast evaluation settings investigated for robustness purposes. Among the machine learning models, the Gradient-Boosted Regression Tree, the Random Forest, and the Multi-Layer Perceptron produce comparatively good forecasting results.

The remainder of this paper is organized as follows. Section 2 presents the baseline linear models and machine learning models that are considered when producing nowcasts of Swedish GDP. The same section also discusses the methods for forecast evaluation and the hyperparameter tuning for the machine learning models. The data series used in the forecast evaluation are presented in Section 3, while Section 4 presents the main results of the forecast evaluation. Concluding remarks are provided in Section 5.

2 Forecasting models

When evaluating the machine learning models in terms of forecast performance, the workhorse models of nowcasting and short-term forecasting, namely linear indicator models, serve as baseline models used for comparison in the current paper.

2.1 Linear indicator models

Three different variants of the linear indicator models are included as baseline models for comparison. First, all single-indicator models that can be specified are evaluated. In general, if there are M indicators available, this will render a total of M different single-indicator models. For the model containing the i:th indicator, the specification is given by (1) below.

$$y_{t,i} = \beta_{i,0} + \beta_{i,1} x_{t,i} + \varepsilon_t \tag{1}$$

The parameters of this model are estimated by ordinary least squares (OLS) on a sample where $t = 1, ..., T$. With the estimated parameters $\hat{\beta}_{i,0}$ and $\hat{\beta}_{i,1}$, the forecasts based on the i:th ETS indicator can be calculated as in (2) for $t = T + 1, ..., T + H$.

$$\hat{y}_{t,i} = \hat{\beta}_{i,0} + \hat{\beta}_{i,1} x_{t,i} \tag{2}$$

In (2), $\hat{y}_{t,i}$ will be the forecast obtained from the single-indicator model using indicator i as an explanatory variable. Since having M different indicators available implies that a set of forecasts will be derived from the single-indicator models, it is reasonable to also use the average of these forecasts as a baseline forecast. This baseline forecast is calculated as in (3).

$$\hat{y}_t^{avg} = \frac{1}{M} \sum_{i=1}^{M} \hat{y}_{t,i} \tag{3}$$

As a final baseline model, a multiple linear model will be used. This model includes all the ETS indicators simultaneously, and the specification is given by (4).

$$y_t = \beta_0 + \sum_{i=1}^{M} \beta_i x_{t,i} + \varepsilon_t \tag{4}$$

As with the single-equation models, the OLS estimator is used to obtain estimates of β_i for $i = 1, ..., M$ in (4). The forecasts from the model will be given by (5).

$$\hat{y}_t = \beta_0 + \sum_{i=1}^{M} \hat{\beta}_i x_{t,i} \tag{5}$$

The multiple linear regression model includes several estimated parameters. Since some machine learning models are built around the idea of regularizing the size of the parameter estimates from multiple linear models to obtain a more sparse model with better forecasting ability, this regression model is interesting to include as a baseline model in the forecast evaluation.

2.2 Machine learning algorithms

2.2.1 Regularized linear models

Since it is often the case that there is a trade-off between forecast error bias and forecast error variance when identifying forecast models that perform well in terms of mean-squared error (MSE), it may be of interest to examine whether the forecast error variance can be decreased by reducing the influence of the explanatory variables in the model, although this may potentially introduce a forecast error bias. In the context of the regularized multiple linear regression model, this amounts to penalizing the size of the regression parameters in (5) and potentially even setting them to zero.

Examples of machine learning methods that perform parameter shrinkage are the regularized linear models Lasso, Ridge, and Elastic Net (see e.g. Tibshirani, 1996; Hoerl and Kennard, 1970; Zou and Hastie, 2005). The main idea behind these models is to augment the sum of squared errors that occurs in the OLS loss function with an additional penalty term that depends on the size of the parameters. This penalty term

is intended to shrink the influence of variables that do not contribute sufficiently to the model fit. In this way, the predictive ability of the model is supposed to increase. The loss functions of the Lasso, Ridge, and Elastic Net models, which differ in how the regression parameters are factored into them, are shown in (6)-(8).

$$L_{Lasso} = \sum_{t=1}^{T} \left(y_t - \hat{\beta}_0 - \hat{\beta}_1 x_{t,1} - \sum_{i=1}^{M} \hat{\beta}_i x_{t,i} \right)^2 + \alpha \sum_{i=1}^{M} |\hat{\beta}_i| \tag{6}$$

$$L_{Ridge} = \sum_{t=1}^{T} \left(y_t - \hat{\beta}_0 - \sum_{i=1}^{M} \hat{\beta}_i x_{t,i} \right)^2 + \alpha \sum_{i=1}^{M} \hat{\beta}_i^2 \tag{7}$$

$$L_{ElaNet} = \sum_{t=1}^{T} \left(y_t - \hat{\beta}_0 - \sum_{i=1}^{M} \hat{\beta}_i x_{t,i} \right)^2 + \alpha \left(\lambda \sum_{i=1}^{M} |\hat{\beta}_i| + (1-\lambda) \sum_{i=1}^{M} \hat{\beta}_i^2 \right) \tag{8}$$

In (6)-(8), α is a hyperparameter that determines how much weight is to be given to the regression parameters in the loss function. A larger value for α implies, everything else being equal, that the loss function assumes higher values for larger parameter estimates, which forces the estimates closer to zero. In addition to α, the Elastic Net loss function in (8) also includes a hyperparameter λ that determines how much relative weight is to be placed on the sum of absolute values and the sum of squared values of the parameters, respectively.

In order to implement the regularized linear models, the hyperparameter α must be set for the Lasso and Ridge models, while the parameters α and λ need to be set for the Elastic Net model. This is accomplished by applying cross-validation on the estimation/training part of the dataset, as described in Section 2.4.

It can be noted that, while the predictions of the multiple linear regression model are invariant to the positioning and scaling of the explanatory variables, this is not the case for the Lasso, Ridge, and Elastic Net models. Instead, when these models are employed, each explanatory variable is rescaled to have a mean value of zero and a unit standard deviation. The same transformation of the explanatory variables is used for all the ML models employed in the paper.

2.2.2 The K Nearest Neighbor algorithm

The K Nearest Neighbor (KNN) algorithm produces nowcasts and forecasts by averaging the target variable values for the K periods in the historical record where the explanatory data exhibit the most similarity to the explanatory data for the period being predicted. If the target variable and the explanatory variables in the historical record (which is equal to the estimation/training sample) are denoted $y_{est} = (y_1, ..., y_T)$, with dimensions $T \times 1$, and $x_{est} = (x_1', ..., x_T')'$, with dimensions $T \times M$, while the $1 \times M$ vector with the explanatory variables for the period being predicted is x_{T+i}, then a $T \times 1$ vector with distances, $d = d(x_{est}, x_{T+i}) = (d_1, ..., d_T)$, can be calculated

following (9) below.

$$d_t = ||x_t - x_{T+i}||_2^2 = \sum_{j=1}^{M} (x_{t,j} - x_{T+i,j})^2 \tag{9}$$

As seen in (9), d_t in is the square of the Euclidean distance between the vectors x_t and x_{T+i}. Once calculated, the distances in d are ordered in ascending order into the vector d^*. A set of K target variable values is then formed into the set y^{KNN} as shown in (10).

$$y^{KNN} = \{y_i | d_j^* = d_i \text{ for } j = 1, ..., K\} \tag{10}$$

Finally, the forecast value for the target variable in period $T + i$, denoted $\hat{y}_{T+i}$, is given by the average over the values in y^{KNN}, calculated as in (11).

$$\hat{y}_{T+i} = \frac{1}{K} \sum_{k=1}^{K} y_k^{KNN} \tag{11}$$

The number of neighbors, K, to consider in the nearest-neighbor algorithm is a hyperparameter that will be tuned based on the estimation/training sample.

2.2.3 Support Vector Regression

In the traditional regression-based methods described above, all the squared deviations from the fitted values enter the loss function when fitting the model. As an alternative, the so-called ϵ-intensive Support Vector Regression can be applied. For this model, the loss function consists of terms that are associated with the observations where the distance between the outcome and the fitted value exceeds a certain threshold, ϵ. Hence, all target values that are within a distance of ϵ from the fitted value are regarded as being close enough to the outcome not to be included in the loss function. For the other observations, the absolute value of the difference between the fitted value and the outcome, decreased by ϵ, enters the loss function.

The general model applied in the Support Vector Regression is given by (12) below.[1]

$$y_t = \beta_0 + \phi(x_t)w \tag{12}$$

In (12), β_0 is an intercept term, while $\phi(\cdot)$ is a vector with basis functions that imply a certain kernel, and $\phi(x_t)$ is the vector of values obtained when the basis functions are applied to observation x_t. The parameters associated with the values in $\phi(x_t)$ are denoted w.

[1]See e.g. Bishop (2006), pp. 340-341, for a more detailed description of the Support Vector Regression.

Given the model in (12) , the terms in (13) will enter the loss function of the support vector regression.

$$\begin{cases} y_t - \beta_0 - \phi(x_t)w - \epsilon = \xi_t & \text{if } y_t - \beta_0 - \phi(x_t)w \geq \epsilon \\ -(y_t - \beta_0 - \phi(x_t)w) - \epsilon = \xi_t^* & \text{if } -(y_t - \beta_0 - \phi(x_t)w) \geq \epsilon \\ 0 = \xi_t = \xi_t^* & \text{if } |y_t - \beta_0 - \phi(x_t)w| < \epsilon \end{cases} \tag{13}$$

The parameters of the Support Vector Regression are obtained from a loss function that includes the terms described in (13). But besides the summation over the errors, a regularization term is included in the loss function. The generic form of the Support Vector Regression loss function can then be written as in (14).

$$\underset{\beta_0, w, \xi, \xi^*}{min} \; C \sum_{t=1}^{T} (\xi_t + \xi_t^*) + ||w||_2^2 \tag{14}$$

In (14), C is a regularization parameter, which, along with ϵ and the choice of kernel (implied by the specific basis functions applied above) are hyperparameters of the Support Vector Regression model.

2.2.4 Regression tree

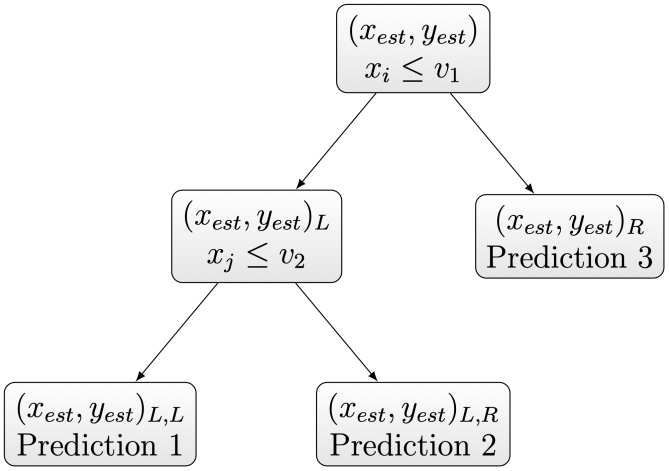
When producing predictions using a regression tree, the first step is to partition the estimation/training data into non-empty, non-overlapping subsets by applying recursive binary splitting of the data. The splitting is based on a condition that has the general form $x_i \leq v_{i,j}$, where x_i is a vector with T observations of the explanatory variable i and $v_{i,j}$ is a value of that variable that allows for a split to be made without any of the resulting subsets being empty. The partitions that are created are such that the condition holds for all the observations in one of the partitions, while the condition does not hold for the observations in the other partition. The tree is built by, in each split, finding the explanatory variable x_i and the value $v_{i,j}$ that minimize the target-variable deviation around the mean in each of the two partitions that result from the split. The minimization problem that is solved when splitting an aggregate set with target variable y into two subsets/partitions, y_L and y_R, is given by (15).

$$\begin{aligned} \underset{x_i, v_{i,j}}{min} \; SSR &= \sum_{t \in T_L} (y_t - \bar{y}_L)^2 + \sum_{t \in T_R} (y_t - \bar{y}_R)^2 \\ \bar{y}_L &= \frac{1}{\#y_L} \sum_{t \in T_L} y_t \\ \bar{y}_R &= \frac{1}{\#y_T} \sum_{t \in T_R} y_t \end{aligned} \tag{15}$$

In (15), T_L and T_R are index sets with the time periods that are partitioned into y_L and y_R, respectively. The recursive splitting of the partitions continues until some

exit criterion is met. Then the splitting stops, and the data partitions that are not split any further will be used for producing predictions. More specifically, the average value of the target variable in the final partitions will be the predictions produced by the tree. A schematic illustration of a regression tree is given in Figure 1.

Figure 1: Basic structure of a regression tree



In the schematic example in Figure 1, the full training dataset, (x_{est}, y_{est}), is first split based on whether or not variable x_i takes on values greater than $v_{i,1}$. The observations for which this condition holds constitute one of the partitions, $(x_{est}, y_{est})_L$, and are sent down the left branch below the top node. The other partition, $(x_{est}, y_{est})_R$, where the condition does not hold, is sent to the right. The first of the partitions, the one sent to the left, will be split again based on whether x_j takes on values greater than $v_{j,2}$, rendering two new partitions, $(x_{est}, y_{est})_{L,L}$ and $(x_{est}, y_{est})_{L,R}$. The second partition of the top node, $(x_{est}, y_{est})_R$, will not be split any further. In total, this renders three different predictions from the example tree. For every new observation of the explanatory variables, it is possible to navigate down the tree until a prediction partition is reached. When arriving in such a partition, the forecast is set to the average training-data value of the target variable in that partition.

In the current application, the stopping criteria for the recursive binary splitting will be the hyperparameters for the regression tree. More specifically, the stopping criteria considered here are the maximum number of successive splits allowed in the tree, the minimum number of observations required in a partition to permit further splitting, and the minimum number of observations that a resulting partition must have to allow for a split of the previous partition.

2.2.5 Bootstrap Aggregation of Regression Trees

One of the drawbacks of using regression trees is that the predictions produced can be sensitive to changes in the data on which the tree is trained. This implies that

a small alteration in the training data can result in relatively large changes in the predictions made by the regression tree.

To reduce this sensitivity, a so-called bootstrap aggregation approach can be taken (see e.g. Breiman, 1996; Lee et al., 2019). Instead of building a single tree based on the training data, the bootstrap aggregation approach entails that a set of B bootstrapped training datasets are constructed. A new tree is fitted to each of these datasets. When a prediction is constructed based on an observation of the explanatory variables, one prediction is produced from each of the trees that are fitted to the bootstrapped datasets. The average over the B different predictions is then used as the bootstrap-aggregated tree forecast.

In the current application, 100 bootstrap replications are used. The hyperparameters of the Bootstrap-Aggregated Regression Tree are the same as those of the Regression Tree, that is, the maximum number of successive splits in a tree, the minimum number of observations in a partition to allow for further splitting, and the minimum number of observations in a resulting partition for a previous partition to be split.

2.2.6 Random Forest

An extension of the Bootstrap-Aggregated Regression Tree is the Random Forest (see e.g. Ho, 1998; Breiman, 2001; Lee et al., 2019). When using this algorithm, it is not only the training data that is randomly resampled. Instead of considering all the explanatory variables when partitioning the data in a tree node, only a random subset of the variables is considered. Once a set of trees has been fitted using the two different forms of randomness, one prediction from each tree can be calculated from an observation of the explanatory data, just as in the case where a simple regression tree has been fitted. The aggregate forecast is obtained as the average of the forecasts in the bootstrapped samples.

Three of the hyperparameters of the Random Forest are the same as those for the Regression Tree and the Bootstrap-Aggregated Regression Tree. Additionally, the fraction of explanatory variables randomly selected for splitting each node is a hyperparameter.

2.2.7 Gradient-Boosted Regression Tree

Instead of creating a set of trees in parallel, as is the case with Bootstrap-Aggregated Regression Trees and Random Forests, the Gradient-Boosted Regression Tree involves fitting a series of trees to the part of the target variable that has not yet been predicted.

The training of the Gradient-Boosted Regression Tree begins with an initial set of predictions for each observation of the target variable in the training sample. The deviations between these predictions and the observed target variable values are then calculated. A regression tree is fitted to the deviations, and the previous predictions

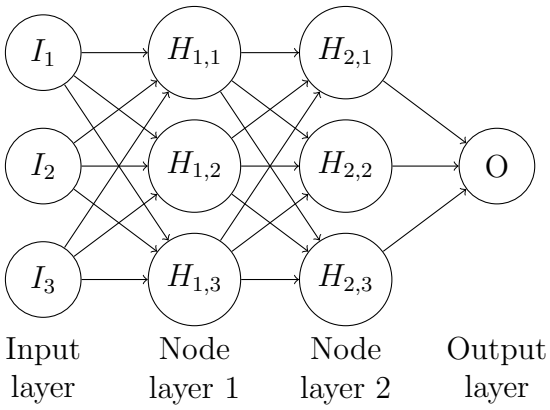
are updated with a fraction of the predictions from this tree. This fraction is called the learning rate. The updated predictions give rise to a new set of deviations. A new tree is fitted to these, the aggregate predictions are updated once again, and so on. When the iterations stop, it is possible to create a prediction from the sequentially fitted trees based on an observation of the explanatory data (see e.g. Kuhn and Johnson, 2013, Section 8.6, or Hastie et al., 2009, Section 10.10, for a general description of boosting).

In addition to the hyperparameters of the Random Forest, the learning rate is a hyperparameter of the Gradient-Boosted Regression Tree.

2.2.8 Multi-Layer Perceptron

The final machine learning algorithm evaluated in the current paper is the Multi-Layer Perceptron. This is a simple form of a neural network in which the value of each explanatory variable is weighted together by adjustable weights and sent to the nodes or neurons in the first layer of the network. Within these nodes, the input data, which is the weighted sum of the input variables, is transformed by a specific function, and the transformed values from the neurons of the first layer are sent to the nodes in the second layer. The output values of the nodes in the first layer are weighted together, again with adjustable weights, and the weighted sums are taken as input values for the nodes in the second layer. This procedure of weighting together input values, transforming them, and sending on the transformed values in the network continues until an output node is reached. In the final step, when reaching the output node, an output value is calculated by weighting together the outputs from the neurons in the preceding layer of the network. A schematic illustration of a MLP neural network is given in Figure 2.

Figure 2: The Multi-Layer Perceptron



As indicated in Figure 2, the layers with nodes that fall between the layer with

input data and the layer with output data are called hidden layers. The specific architecture of the neural network, along with the functions in the nodes, enables the Multi-Layer Perceptron to capture complex and non-linear relationships in the data. When fitting the Multi-Layer Perceptron to the training data, the parameters that are adjusted are the weights used to combine the values sent from the nodes in the previous layer of the network.

In the nodes of the neural network, various types of functions can be used to transform the input values to output values. These functions will be one of the hyperparameters of the neural network. Additionally, the number of node layers considered and the number of nodes in each layer will also be hyperparameters. Finally, to keep the parameterization of the model tractable, there is a regularization parameter associated with the size of the parameters of the neural network model that will be a hyperparameter.

2.3 Hyperparameter space of the ML models

As evident from the discussion on the various machine learning models above, there are several hyperparameters that need to be set before a model can be used to produce a forecast. A summary of all the hyperparameters mentioned above is given in Table 1.

Looking at the hyperparameters in Table 1, it is clear that there is a wide array of different implementations of the machine learning models. If a forecast evaluation is to be performed by dividing the available data sample into an estimation/training part and an evaluation/test part, and then estimating all models on the former part and evaluating the forecast properties on the latter part, it is not unlikely that problems could arise. When there are many different variants of models evaluated on an evaluation sample of limited size, it is not unlikely that one of the models could turn out, just by chance, to fit the data in the evaluation sample really well. Since the model fits the data by chance, it may not perform as well in a forecast production process going forward. Hence, care should be taken not to evaluate too many models on the evaluation data set. Instead, a pre-selection of models is performed based on the estimation set, and only a smaller number of models is brought to the evaluation dataset. More specifically, only one model implementation from each machine learning model class is evaluated on the test part of the sample. This model is selected by tuning the hyperparameters of the machine learning models using a cross-validation approach.

2.4 Tuning the hyperparameters

Instead of evaluating all variants of the machine learning models on the evaluation sample, one hyperparameter setting is selected for each machine learning algorithm. The machine learning model chosen from each model class is then evaluated on the

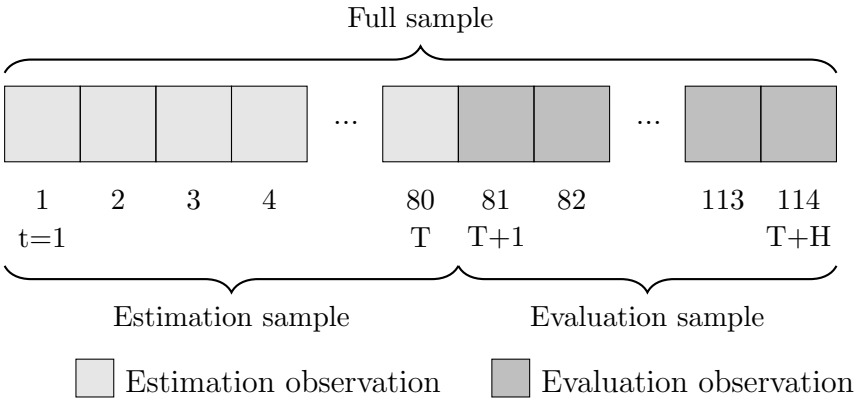
Table 1: Hyperparameter space of the ML models.

Hyperparameter	Explanation
	Lasso and Ridge
$\alpha \in \{0.01, 0.1, 1, 10, 100, 1000\}$	Weighting of the regression parameters in the loss function.
	Elastic Net
$\alpha \in \{0.01, 0.1, 1, 10, 100, 1000\}$	Overall regularization in the loss function.
$\lambda \in \{0.25, 0.50, 0.75\}$	Relative weight assigned to the sum of absolute values of parameters in loss function.
	K Nearest Neighbors
$K \in \{2, 3, 4, 5, 6, 7, 8\}$	Number of neighbors to consider.
	Support Vector Regression
Kernels: Linear, Radial Basis Function	Kernel to use in estimation.
$C \in \{0.1, 1, 10, 100\}$	Regularization parameter in the loss function.
$\epsilon \in \{0.1, 1, 10\}$	Value that defines the width of the tube within which the errors do not enter the loss function.
	Regression Tree and Bootstrap Aggregated Tree
Depth: 2, 3, 4	Maximum number of consecutive splits in the tree.
Min. to split: 4, 8, 12	Minimum number of observations required to split.
Min. in partition: 4, 8, 12	Minimum number of observations in a resulting partition.
	Random Forest
Depth: 2, 3, 4	Maximum number of consecutive splits in the tree.
Min. to split: 4, 8, 12	Minimum number of observations required to split.
Min. in partition: 4, 8, 12	Minimum number of observations in a resulting partition.
Fraction expl. vars: 0.25, 0.50, 0.75	Fraction of explanatory variables in split.
	Gradient-Boosted Regression Tree
Depth: 2, 3	Maximum number of consecutive splits in the tree.
Min. to split: 8, 12	Minimum number of observations required to split.
Min. in partition: 8, 12	Minimum number of observations in a resulting partition.
Fraction expl. vars: 0.25, 0.50, 0.75	Fraction of explanatory variables in split.
Learning rate: 0.05, 0.10, 0.15	Learning rate when generating aggregate predictions.
	Multi-Layer Perceptron
Nodes and layers: (3,3,3), (5,5), (10)	The number of nodes in each layer and the number of layers. For example, (3,3,3) means three nodes in the first, second and third layer, respectively.
Node function: Identity, Logistic, Hyperbolic tangent and Rectified Linear Unit	Functions applied to weighted inputs of a node.
$\alpha \in \{0.0001, 0.001, 0.01, 0.1, 1\}$	Weighting of the parameters in the loss function.

evaluation sample and compared to the baseline models and models from the other machine learning model classes.

To describe how hyperparameter tuning and the estimation of the chosen models are done, first consider the entire sample as described in Figure 3.

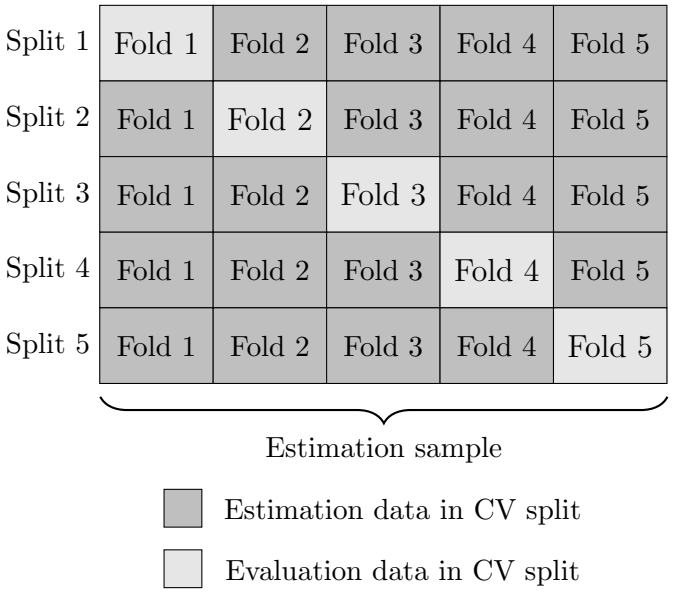
Figure 3: Estimation and evaluation samples



As illustrated in Figure 3, the entire data sample will consist of T+H=114 observations (see Section 3 for a further description of the data). Out of these, T=80 observations will be used for estimation/training, while H=34 will be used for evaluation/testing.

All the baseline models and one model from each machine learning model class are estimated on the 80 observations before the forecasts are calculated and evaluated. To select which machine learning model from each model class will be brought forward to the evaluation data, a choice has to be made regarding which hyperparameters to use. This is done by employing a 5-fold cross-validation search within the estimation sample. By splitting the estimation sample into five different parts, so-called folds, each model can be estimated on four out of the five folds and evaluated on the remaining fold. The cross-validation setup is described in Figure 4.

Figure 4: Cross-validation for hyperparameter tuning.



By excluding one of the folds from the estimation sample in each of the cross-validation splits, an out-of-sample-like prediction can be made for the data in the excluded fold. By calculating a score for the models on each of the excluded folds and then averaging over the five different scores that are obtained, it is possible to identify the best hyperparameter setting for each machine learning model in the estimation sample.

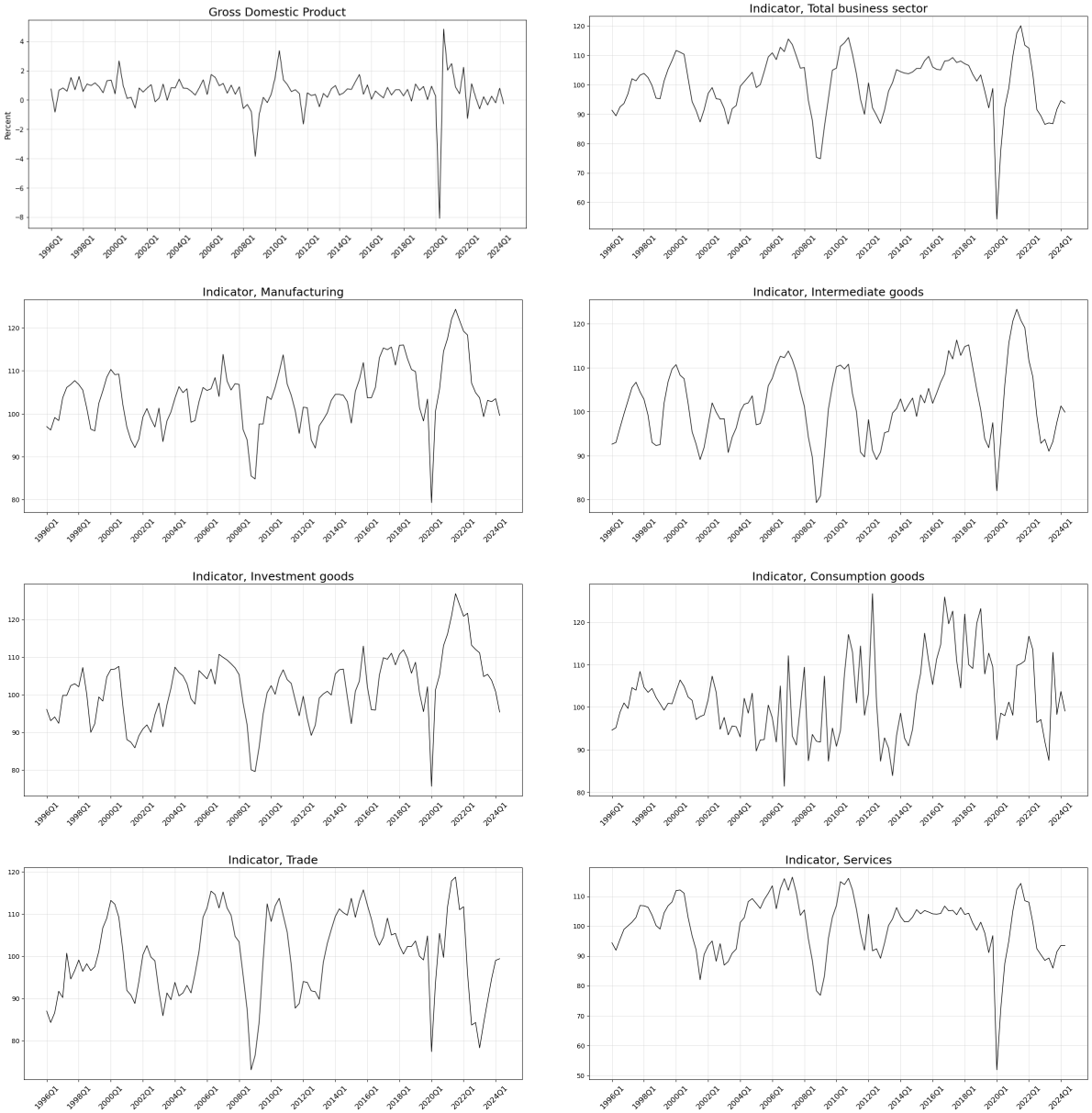
By carrying out cross-validation within the estimation sample to tune the hyperparameters of each machine learning model, and then selecting one model from each class for evaluation, the intention is to reduce the risk of identifying a model that performs well in the evaluation sample by mere chance.

3 Data

The data series used in the current paper are similar to those used in Jönsson (2020, 2021, 2024). The variable for which nowcasts are produced is the quarter-on-quarter growth in seasonally adjusted real GDP. As explanatory variables, seven sentiment indicators are used. These include the confidence indicators for the total business sector, the aggregate manufacturing sector, and three subsectors within the manufacturing sector (intermediate, investment, and consumer goods), as well as the trade sector and the services sector. The data series, covering the period from 1996Q1 to 2024Q2, are plotted in Figure 5.

Looking at the data in Figure 5, two things stand out. The first is that there are large variations in GDP growth around the outbreak of the Covid-19 pandemic.

Figure 5: GDP growth and ETS indicators for 1996Q1-2024Q2.



In 2020Q2, there is a significant decrease in growth, followed by a large rebound in 2020Q3. To ensure that the results from the forecast evaluations are not critically dependent on these outcomes, forecast evaluations are performed both by including and excluding 2020Q2 and 2020Q3 from the evaluation.

Second, the ETS indicators are dated such that the April, July, October, and January surveys correspond to Q1, Q2, Q3, and Q4 indicators, respectively. Reviewing the developments in early 2020, it is evident that the indicators declined during 2020Q1, while GDP growth decreased in 2020Q2. This raises the general question of whether GDP is better predicted by lagging the indicators, such that the Q1 values of the indicators are utilized to predict Q2 values of GDP growth. Both of these scenarios are taken into account in the forecast evaluations.

Taken together, this results in four different forecasting evaluations being made. These are defined by including or excluding 2020Q2 and 2020Q3, and by using contemporaneous or lagged indicators when predicting GDP growth.

4 Forecast evaluations

In Table 2, the results from the forecast evaluation exercise are presented. Three different forecast error metrics are considered: the mean error (ME), the mean absolute error (MAE) and the mean squared error (MSE). Furthermore, results are presented in four columns. The first column contains the results for the full dataset without lagging the ETS indicators. The second column contains the results obtained when 2020Q2 and 2020Q3 are excluded from the evaluation sample, as discussed above. In the third and fourth columns, there are results corresponding to those of the first two columns, but with the ETS indicators lagged by one quarter.

When looking at the results in Table 2, a few things, not specifically related to the baseline or machine learning models, stand out. First, it seems that the GDP developments during the outbreak of the Covid-19 pandemic were hard to predict based on the ETS indicators alone. This becomes evident by comparing the absolute and squared forecast errors for the cases where these observations are included and excluded, respectively. When the quarters are included in the evaluation, the MAE and MSE metrics are larger.

Second, with some exceptions, the results indicate that lagging the indicators, so that the current quarter's GDP growth is explained by the indicator values of the previous quarter, improves forecasting ability compared to not lagging the indicators. This aspect is particularly emphasized when the baseline models are considered, which could indicate that the information obtained from the ETS has a forward-looking component when employed in linear forecasting models.

In terms of the values of the mean error of the model forecasts, there are some differences between the baseline models and the machine learning models. The single-indicator models tend to produce a positive mean error, while the multiple linear model and the average forecast over the single-indicator models exhibit a negative

Table 2: Forecast evaluation results.[a]

Model	Full dataset	Excl. 2020Q1-Q2	Lagged indic.	Excl. 2020Q1-Q2 and lagged indic.
		Mean Error		
		Univariate models		
Const.	-0.223	-0.101	-0.223	-0.101
AR(1)	-0.153	-0.133	-0.153	-0.133
		Baseline models		
Total business sector	-0.189	-0.121	-0.191	-0.149
Manufacturing	-0.857	-0.764	-0.571	-0.503
Intermediate goods	-0.609	-0.507	-0.495	-0.426
Investment goods	-0.748	-0.639	-0.504	-0.416
Consumer goods	-0.307	-0.254	-0.174	-0.040
Trade	-0.277	-0.155	-0.246	-0.151
Services	-0.007	-0.054	-0.064	-0.025
Multiple reg. model	-0.368	-0.280	-0.369	-0.226
Forecast avg.	-0.437	-0.341	-0.321	-0.244
		Machine learning models		
Lasso	-0.357	-0.267	-0.207	-0.077
Ridge	-0.422	-0.329	-0.240	-0.164
Elastic Net	-0.310	-0.204	-0.250	-0.137
KNN	-0.420	-0.318	-0.178	-0.098
Support Vector regression	-0.285	-0.162	-0.024	0.075
Regression Tree	-0.268	-0.156	-0.330	-0.241
Bootstr. Agg. Tree	-0.393	-0.288	-0.288	-0.213
Random Forest	-0.319	-0.209	-0.220	-0.131
Gradient-Boosted Reg. Tree	-0.274	-0.168	-0.010	0.089
Multi-Layer Perceptron	-0.368	-0.269	-0.243	-0.140
		Mean Absolute Error		
		Univariate models		
Const.	0.916	0.569	0.916	0.569
AR(1)	1.049	0.607	1.049	0.607
		Baseline models		
Total business sector	1.019	0.708	0.898	0.569
Manufacturing	1.262	0.951	1.051	0.737
Intermediate goods	1.040	0.730	1.026	0.696
Investment goods	1.167	0.843	0.996	0.672
Consumer goods	0.949	0.603	0.935	0.577
Trade	0.943	0.616	0.874	0.533
Services	0.957	0.640	0.878	0.543
Multiple reg. model	1.035	0.723	0.958	0.596
Forecast avg.	0.994	0.670	0.918	0.583
		Machine learning models		
Lasso	0.986	0.668	0.926	0.568
Ridge	0.992	0.670	0.895	0.556
Elastic Net	0.918	0.584	0.903	0.551
KNN	0.965	0.650	0.916	0.573
Support Vector regression	0.920	0.577	0.908	0.550
Regression Tree	1.046	0.755	1.056	0.727
Bootstr. Agg. Tree	0.939	0.603	0.898	0.576
Random Forest	0.900	0.563	0.887	0.552
Gradient-Boosted Reg. Tree	0.919	0.584	0.846	0.516
Multi-Layer Perceptron	0.923	0.598	0.904	0.548
		Mean squared Error		
		Univariate models		
Const.	3.307	0.583	3.307	0.583
AR(1)	4.485	0.633	4.485	0.633
		Baseline models		
Total business sector	3.037	0.876	2.809	0.630
Manufacturing	3.946	1.415	3.093	0.855
Intermediate goods	3.259	0.905	3.203	0.818
Investment goods	3.818	1.151	3.160	0.797
Consumer goods	3.367	0.649	3.408	0.591
Trade	3.131	0.628	2.984	0.533
Services	2.937	0.797	2.838	0.625
Multiple reg. model	3.099	0.832	3.585	0.598
Forecast avg.	3.155	0.724	2.980	0.615
		Machine learning models		
Lasso	3.008	0.695	3.393	0.590
Ridge	3.128	0.735	2.953	0.574
Elastic Net	3.067	0.554	3.231	0.563
KNN	2.965	0.637	3.107	0.663
Support Vector regression	3.269	0.588	3.192	0.558
Regression Tree	2.937	0.822	3.267	0.851
Bootstr. Agg. Tree	3.171	0.603	2.771	0.574
Random Forest	3.103	0.549	2.932	0.568
Gradient-Boosted Reg. Tree	3.094	0.586	2.769	0.510
Multi-Layer Perceptron	2.954	0.585	3.199	0.559

[a] Const. and AR(1) refer to models where only an intercept and an intercept and the lagged dependent variable are included as explanatory variables, respectively.

mean error. The machine learning models also have a negative mean error. Looking at the magnitudes of the mean errors, the machine learning models produce magnitudes of mean error that are on average approximately 20-40 percent smaller than those of the baseline models. However, within the group of baseline models, there is a larger variation in the magnitude of mean error than what is the case in the group of machine learning models. Some of the baseline models perform the worst among all the models considered, whereas the better models, in terms of the absolute value of the mean error, are also often found among the baseline models when examining the four different evaluations in Table 2.

Turning to the MAE criterion, the average machine learning model performs 3-13 percent better than the average baseline model, depending on which of the columns in Table 2 is considered. From the results, it can be seen that the Gradient-Boosted Regression Tree generally, across the four cases considered, performs well compared to the other models in terms of MAE. In all four cases, it displays better performance than any of the baseline models. Comparing the results among the different machine learning models, it turns out that the Random Forest and Multi-Layer Perceptron models also fare relatively well. All of these models have the ability to capture nonlinearities in the data, which could be one explanation for their relatively good results.

In terms of MSE, a similar pattern to that of ME and MAE can be observed. When comparing the different evaluations, the average machine learning model performs 1-28 percent better, in terms of MSE, than the average baseline model. Again, the Gradient-Boosted Regression Tree, the Random Forest, and the Multi-Layer Perceptron often seem to perform well in comparison to the other models.

Across all the four evaluation cases in Table 2, there are two more general results that stand out. First, across all three evaluation metrics (i.e. across ME, MAE, and MSE), the variation in forecast performance within the machine learning models is always smaller than the variation in performance within the baseline models. As a model class, the machine learning models hence deliver more stable results than the baseline models. Second, in terms of MAE and MSE, and across all four of the evaluation cases, there is always a machine learning model with at least as good forecasting performance as the best baseline model.

Taken together, the results in Table 2 indicate that it can be fruitful, in terms of forecast precision, to not only consider linear indicator models but also machine learning models when producing nowcasts and short-term forecasts of Swedish GDP growth using Economic Tendency Survey data. Gradient-Boosted Regression Trees, Random Forest, and Multi-Layer Perceptron models seem to perform well in terms of forecasting accuracy. That these models perform well is in line with what has been previously found by, for example, Yoon (2021); Soybilgen and Yazgan (2021); Kant et al. (2024), who also find that various forms of ensemble-based tree methods seem promising.

5 Concluding remarks

Having access to accurate nowcasts and short-term forecasts can be important for various reasons. In addition to providing insights into the current economic situation and near-term developments, accurate nowcasts can enhance the quality of longer-term forecasts by offering a better point of departure.

One important source of information regarding the current stance of the economy is Economic Tendency Surveys. Together with linear indicator models, they constitute a backbone in the production of macroeconomic nowcasts. However, by utilizing a wider array of models, additional insights into economic developments can be gained. In the current paper, the forecasting performance of baseline linear models for nowcasting Swedish GDP growth is compared to the performance of machine learning models.

The nowcast performance results indicate that machine learning models can complement the linear baseline models. In terms of forecasting accuracy, the machine learning models, as a class, deliver, on average, better forecasts than the baseline models, regardless of whether mean error, mean absolute error, or mean squared error is considered. Furthermore, the variation in nowcasting performance within the machine learning models is smaller compared to that of the baseline models. Additionally, in terms of MAE and MSE, it is often observed that machine learning models, more specifically either the Gradient-Boosted Regression Tree, the Random Forest, or the Multi-Layer Perceptron, exhibit the best forecasting performance. This applies across all the models, regardless of whether machine learning models or baseline models are considered.

References

Arro-Cannarsa, M. and Scheufele, R. (2024). Nowcasting GDP: What are the Gains from Machine Learning Algorithms? SNB Working Paper 6/2024, Swiss National Bank.

Billstam, M., Frändén, K., Samuelsson, J., and Österholm, P. (2017). Quasi-Real-Time Data of the Economic Tendency Survey. *Journal of Business Cycle Research*, 13:105–138.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24:123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45:5–32.

den Reijer, A. and Johansson, A. (2019). Nowcasting Swedish GDP with a Large and Unbalanced Data Set. *Empirical Economics*, 57(4):1351–1373.

Hansson, J., Jansson, P., and Löf, M. (2005). Business Survey Data: Do They Help in Forecasting GDP growth? *International Journal of Forecasting*, 21:377–389.

Hastie, T., Tibshirani, R., and Fiedman, J. (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer.

Ho, T. K. (1998). The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67.

Jönsson, K. (2020). Machine Learning and Nowcasts of Swedish GDP. *Journal of Business Cycle Research*, 16:123–134.

Jönsson, K. (2021). Bootstrap Aggregation Accuracy Gains in Nearest-Neighbor Nowcasting of Swedish Gross Domestic Product. *Applied AI Letters*, 2(2).

Jönsson, K. (2024). Neighbor Weighting and Distance Metrics in Nearest Neighbor Nowcasting of Swedish GDP. *Journal of Quantitative Economics*, 24:1077–1089.

Kant, D., Pick, A., and de Winter, J. (2024). Nowcasting GDP Using Machine Learning Methods. *AStA Advances in Statistical Analysis*. forthcoming.

Kaufmann, D. and Scheufele, R. (2017). Business Tendency Surveys and Macroeconomic Fluctuations. *International Journal of Forecasting*, 33:878–893.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Lee, T.-H., Ullah, A., and Wang, R. (2019). Bootstrap Aggregating and Random Forest. Working Paper 201918, University of California at Riverside, Department of Economics.

Medeiros, M. C., Vasconcelos, G. F. R., Veiga, A., and Zilberman, E. (2021). Forecasting Inflation in a Data-Rich Environment: The Benefits of Machine Learning Methods. *Journal of Business & Economic Statistics*, 39(1):98–119.

Österholm, P. (2014). Survey Data and Short-Term Forecasts of Swedish GDP Growth. *Applied Economics Letters*, 21:135–139.

Richardson, A., van Florenstein Mulder, T., and Vehbi, T. (2018). Nowcasting New Zealand GDP Using Machine Learning Algorithms. CAMA Working Paper 47/2018, Centre for Applied Macroeconomic Analysis, Australian National University.

Richardson, A., van Florenstein Mulder, T., and Vehbi, T. (2021). Nowcasting GDP Using Machine-Learning Algorithms: A Real-Time Assessment. *International Journal of Forecasting*, 37(2):941–948.

Smalter Hall, A. (2018). Machine Learning Approaches to Macroeconomic Forecasting. Federal Reserve Bank of Kansas City Economic Review, Federal Reserve Bank of Kansas City.

Soybilgen, B. and Yazgan, E. (2021). Nowcasting US GDP Using Tree-Based Ensemble Models and Dynamic Factors. *Computational Economics*, 57:387–417.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Yoon, J. (2021). Forecasting of Real GDP Growth Using Machine Learning Models: Gradient Boosting and Random Forest Approach. *Computational Economics*, 57(1):1–19.

Zou, H. and Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320.