An Interpretable Machine Learning Workflow with an Application to Economic Forecasting*

Marcus Buckmann and Andreas Joseph
Bank of England

We propose a generic workflow for the use of machine learning models to inform decisionmaking and to communicate modeling results with stakeholders. It involves three steps: (i) a comparative model evaluation, (ii) a feature importance analysis, and (iii) statistical inference based on Shapley value decompositions. We discuss the different steps of the workflow in detail and demonstrate each by forecasting changes in U.S. unemployment one year ahead using the well-established FRED-MD data set. We find that universal function approximators from the machine learning literature, including gradient boosting and artificial neural networks, outperform more conventional linear models. This better performance is associated with greater flexibility, allowing the machine learning models to account for time-varying and non-linear relationships in the data-generating process. The Shapley value decomposition identifies economically meaningful non-linearities learned by the models. Shapley regressions for statistical inference on machine learning models enable us to assess and communicate variable importance akin to conventional econometric approaches. While we also explore high-dimensional models, our findings suggest that the best trade-off between interpretability and performance of the models is achieved when a small set of variables is selected by domain experts.

JEL Codes: C14, C45, C53, E27.

*The content and views presented in this article do not represent the views of the Bank of England (BoE). Many thanks to David Bholat, Philippe Bracke, and Helena Robertson for support of the project and helpful comments on the manuscript. The code for implementing and replicating the paper's results can be found at https://github.com/bank-of-england/InterpretableMLWorkflow.

1. Introduction

Predictive machine learning models are increasingly being used at decisionmaking institutions, such as central banks, governments, and international institutions (Doerr, Gambacorta, and Garralda 2021). Major appeals of these models are that they often give more accurate predictions than conventional approaches and can handle high-dimensional data (Haldane 2018).

On the downside, many machine learning methods suffer from the black box critique. It is not straightforward to assess the factors driving predictions and therefore to understand the relations between the inputs and output of the model. However, this understanding of a model is crucial, especially for decisionmaking processes, for several reasons. First, both decisionmakers and their audiences naturally have a desire to understand the inputs leading to decisions and legitimize them. Second, decisionmaking processes often involve multiple models.[1] The information derived from different models should be compatible, leading to a coherent picture. The understanding of all models involved is needed for this. Third, models can "misfire" for several reasons—for example, by picking up spurious relations in the data. This often can only be detected and prevented if one has a good understanding of a model.

Prediction models whose accuracy is a key motivation behind their deployment—which often holds for machine learning methods—should also help to inform the narrative approach behind any economic policy decision rather than providing mere black box predictions (George 1999; Burgess et al. 2013; Independent Evaluation Office 2015). Machine learning models also can provide a richer set of information compared with more conventional statistical models, like linear regression models. In particular, they can implicitly learn non-linear functional forms and interaction from the data without the need to specify them a priori.

In this paper, we lay out a multi-step workflow for the use of machine learning models, which we deem suitable to inform

[1]Without any stringent assumption on the data-generating process, machine learning models can be labeled "non-structural," describing correlations between inputs and targets. Other "structural" models make richer assumptions about the data-generating process. Comparing the two requires analyses of the assumptions on the data-generating process, estimation techniques, and results.

decisionmaking processes. It consists of three steps which can be directly applied to other contexts as well than those presented in the accompanying case study. First, a model comparison is conducted between conventional statistical methods and machine learning models to provide prima facie evidence of whether a machine learning approach is likely to deliver benefits. If the primary objective is model accuracy, e.g., for forecasting, this would be a model horse race to minimize the forecasting error. Second, the machine learning predictions are decomposed into the contributions of the individual model variables. This allows us to uncover the relative importance of variables and understand the functional forms learned by the different machine learning models. By a comparison across models, one can gauge how robust feature decompositions are to the choice of the algorithm. Third, statistical inference is conducted to understand which variables make a statistically significant contribution to the accuracy of a model, providing a level of confidence for our interpretations and any narrative attached to them. This inference uses a parametric regression analysis, allowing for a standardized communication of statistical model results. A rich set of robustness checks provides guidance for frequently encountered challenges, especially when using machine learning. These include variable selection in a high-dimensional setting, model stability, and computational requirements.

Throughout the paper, we apply the proposed procedures to a macroeconomic case study, where we forecast changes in unemployment—an important input for fiscal and monetary policy decisions (Burgess et al. 2013). Along the steps of the workflow, we contrast the use of machine learning models with a simpler but less flexible linear model.

Most of the presented techniques generalize to other settings in a straightforward manner, such that this paper provides a one-stop reference for practitioners for the use of machine learning models in situations where the understanding and communication of model results is crucial. This is especially the case at policy institutions. Inevitably this ties to many technicalities. We discuss the most relevant ones in intuitive terms and provide references to the related literature for further guidance.

There are several levels of communication involved in a data-driven decision process, from technical modeling experts performing the analysis, over communications to management, up to the decisionmaking bodies. The ability to communicate modeling results with varying levels of complexity is crucial in this setting. However, effective communication is highly contextual. The technical knowledge and experience can vary greatly within decisionmaking bodies and their audiences. Thus, there is no one-size-fits-all mapping between workflow outputs and target audiences. Instead we provide some broad guidance and suggestions on matching individual outputs with target audiences as follows. We layer target audiences by how close they are to the technical details of the analysis, going from analysts, who perform the analysis, to management, who aggregate and filter information from different sources (e.g., different teams of analysts) and distill information for decisionmaking bodies, and finally decisionmakers and their audiences. This gives three levels, where guidance is to be understood as a "smaller or equal," meaning that if the target audience is the management, it also includes analysts, and if it is decisionmakers, it may serve for all. Labels for the target audience are mostly attached to table and figure captions which summarize the outputs of our workflow.

The present paper connects different fields, ranging from machine learning and model interpretability to statistical inference and economic forecasting. There is a growing literature that suggests that machine learning methods can outperform more conventional models in economic prediction problems including forecasting. For example, machine learning methods have been shown to be better at predicting bond risk premia (Bianchi, Büchner, and Tamoni 2019), forecasting macroeconomic variables such as unemployment and inflation (Sermpinis et al. 2014; Chen et al. 2022), recessions (Döpke, Fritsche, and Pierdzioch 2017), and financial crises (Bluwstein et al. 2020).[2] However, other papers do not observe consistently improved performance by using machine learning, instead finding that it is state or horizon dependent (Kock and Teräsvirta 2014). This mixed

[2]In these problems, several variables are used to forecast the outcome variable. In the univariate case, when only the lagged outcome is used for prediction, evidence suggests that statistical methods or hybrid models combining statistical and machine learning approaches outperform pure machine learning methods, on average (Makridakis, Spiliotis, and Assimkopoulos 2018a, 2018b; Parmezan et al. 2019).

evidence validates our horse race as an important first step for the workflow.

Predicting macroeconomic dynamics is challenging. Relationships between variables may not hold over time, and shocks such as recessions or financial crises might lead to a breakdown of previously observed relationships (Elliott and Timmermann 2008; Ng and Wright 2013). In line with the literature, we suggest that it is the inherent non-linearity of non-parametric models that allows them to learn and exploit complex relationships for prediction (Wang and Manning 2013). Coulombe et al. (2020) show that this advantage of machine learning models to exploit non-linearities in macroforecasting is enhanced at longer horizons. However, the non-linear relationships learned are not directly observable, which has led to the aforementioned black box critique of these models as a major challenge to their applicability to inform decisions.

Approaches to interpretable machine learning come from different directions: epistemic discussions about what it means for a model to be interpretable (Miller 2019), technical approaches in machine learning research (Doshi-Velez and Kim 2017), and methodology in econometrics and statistics (Chernozhukov et al. 2018).

Miller (2019) analyzes the psychology of explanations and suggests that humans expect explanations that are based on a limited number of causes rather than an exhaustive account of all factors—acknowledging that the simplification of the problem risks introducing bias. Relatedly, Lipton (2016) argues that a high-dimensional linear model is not necessarily more interpretable than a compact artificial neural network that learns from only few features. Also, if the linear model is trained on abstract features, for instance, obtained by principal component analysis, its parameters may not provide an obvious economic interpretation.

In the machine learning literature, approaches to interpretability usually focus on measuring how important input variables are for prediction. *Variable attributions* can be either global, by assessing the variable importance across the whole data set, or local, by measuring the importance of the variables at the level of individual observations in the form of a decomposition. Such local attributions can always be summarized in a global variable importance measure by averaging local attributions across all observations. Popular global methods are permutation importance or Gini importance for tree-based models (Breiman 2001a). Popular local decomposition methods are LIME (Ribeiro, Singh, and Guestrin 2016), DeepLIFT (Shrikumar, Greenside, and Anshul 2017), and Shapley values (Štrumbelj and Kononenko 2010; Lundberg and Lee 2017). Lundberg and Lee (2017) demonstrate that Shapley values offer a unified framework of LIME (local interpretable model-agnostic explanations) and DeepLIFT with appealing properties. Most importantly, Shapley values guarantee *consistency*, where a consistent measure of variable importance preserves the relative importance between variables across situations where such a ranking is imposed. We therefore focus on Shapley values when describing the workflow and presenting the case study. For illustrative purposes, we contrast the use of Shapley values with permutation importance.

These global and local attribution methods are only descriptive—they explain the drivers of model predictions and performance, but they do not assess the predictors' statistical significance, i.e., how certain one can be that a variable is actually important to describe a specific outcome. We extend our interpretation of machine learning models for forecasting by statistically testing the predictors in a Shapley regression framework (Joseph 2019). Shapley values and inference based on them is arguably the most general and rigorous approach to address the issues of machine learning interpretability and model communication. In this way, we close the gap between two traditional modeling approaches, the maximization of predictive performance using "black box" machine learning methods and the application of statistical techniques to make inferences about the data-generating process (Breiman 2001b).

The remainder of this paper is structured as follows. Section 2 describes the proposed workflow. The data and the methodology used for the macroeconomic forecasting study used throughout this paper is introduced in Section 3. Section 4 presents the outputs of the workflow for our baseline scenario. This includes model performances, the analysis of feature importances and learned functional forms, and statistical inference. Section 5 discusses a rich set of robustness checks and how they relate to different aspects of the proposed workflow. We conclude with a short discussion in Section 6. The technical appendix (Appendix B) discusses the computation of model interpretability measures used in the case study.

2. A Machine Learning Workflow

Our proposed workflow for the use of machine learning models is geared towards situations where model interpretability and the communication of modeling results is important. It consists of three steps: a model comparison, the assessment of variable importances, and statistical inference on model components. The latter two steps are required due to the opaque nature of machine learning models.

We keep the notation deliberately simple and general, with a more detailed and specific description used in the next section and Appendix B. We say that a model $f(x;\theta) = \hat{y}$ takes inputs x, consisting of N variables indexed k, and has fitted parameters θ. The model predicts the target variable $y = \hat{y} + \epsilon$, with ϵ being the error term of which some form is minimized during model training (fitting), e.g., the squared error ϵ^2. The Greek letter ϕ_k denotes variable components of f, i.e., $f(x) = \sum_{k=0}^{N} \phi_k(x)$, with ϕ_0 being an intercept. Additionally, let us denote $\mathcal{P}$ as a performance metric to evaluate the goodness of a model. For example, we may define $\mathcal{P}$ as the prediction error, which we aim to minimize.

2.1 Step 1: Model Comparison

Machine learning methods require additional effort from the modeler compared with conventionally used econometric models (steps 2 and 3). Thus, the first step is to decide whether to proceed with any further analysis of the machine learning models by comparing their performance with that of a benchmark model. The performance metric $\mathcal{P}$ can, for example, be the absolute forecasting error when predicting a continuous variable in a time series. However, $\mathcal{P}$ can also have a more complex forms. For example, it may be a function describing the trade-off between type 1 and 2 errors when predicting a binary variable, or when describing treatment heterogeneity in an experimental setting. Crucial questions regarding this horse race are what models (not) to compare, their stability, and what or how much data to use.

2.1.1 Model Selection

The choice of models can be both specific and general—specific in the sense that there are established models for certain tasks, which can serve as benchmarks, and general in the sense that it is usually not possible to know for machine learning models which models will predict best in a given situation (Fernández-Delgado et al. 2014). A reasonable start are popular general-purpose machine learning models, such as random forests, gradient boosting, support vector machines, and artificial neural networks (see Friedman, Hastie, and Tibshirani 2009 for an introduction to different models). Some authors include penalized regressions in the machine learning toolbox. These models arguably lie at the boundary between traditional econometric and machine learning techniques. We do not include them in the latter, as they do not have the universal approximator property (see, for example, Cybenko 1989). Universal function appoximation means that a model will, under the right circumstances, learn to approximate any functional form given enough training data. A further difference between the two classes of models is that the parameter vector θ is clearly defined for (penalized) regression models, while it is generally undefined in terms of its shape and values for machine learning models. These are set during the cross-validation and training process, respectively.

Some machine learning models may not be suited for a certain prediction task. For example, support vector machines have substantial computational costs when the data set is large. Random forests can be memory intensive, especially when allowed to grow many large trees on a large data set. Further, random forests are not suited for extrapolation beyond the training set, i.e., they cannot make predictions that exceed the observed values in the training set. On the other hand, random forests can deal well with high-dimensional data and a limited number of observations. Compared with other methods, they also deal well with extreme values and correlated variables.

Artificial neural networks are both computation and data intensive. They have a wide range of architectures, some adapted to certain data types (see Goodfellow, Bengio, and Courville 2016 for an overview), but finding the appropriate architecture and other hyperparameters can be a challenging task. In contrast, support vector machines only have a few relevant hyperparameters, and the random forest often performs well without tuning the hyperparameters at all.

2.1.2 Model Stability

Another aspect to consider is model stability. A linear or support vector regression (SVR) will always produce a deterministic optimal solution, while the training process of a random forest or artificial neural network is not deterministic, leading to different solutions when trained repeatedly on the same data with different random seeds. D'Amour et al. (2020) showed for complex neural networks that these different solutions can produce substantially different predictions on new data that differs from the distribution on which the model was trained. We may encounter this situation in economic forecasting when there is some unknown drift in the data-generating process. Further, the hyperparameter search can introduce randomness into the training of any machine learning method. For complex models, such as gradient boosting or artificial neural networks, an exhaustive search for hyperparameters often is infeasible. In practice one only tests a few values for selected key hyperparameters. Alternatively, one employs a random search, testing only a subset of all possible combinations in a larger hyperparameter space. Ideally, a machine learning model is insensitive to changes in its hyperparameters, which makes a model comparison more robust and increases the replicability of the modeling.

A remedy for low model stability is averaging several models trained based on different random seeds or slightly different training samples. However, this comes at the cost of increased computational requirements—for training the ensemble of models, storing them, and explaining their predictions.

2.1.3 Data and Variables Selection

We assume that the data set is structured, i.e., it can be represented well in a table.[3] A modeler might be tempted to use all available variables as predictors. However, including more variables increases the likelihood that a model exploits spurious correlations that might

[3]The complement to this is unstructured data such as text, images, video, etc. On these kind of data, artificial neural networks generally perform better than other machine learning approaches, while data always need to be brought into some potentially very high-dimensional tabular representation.

not hold outside the training sample, increasing model variance and lowering performance.

Accordingly, one strand of the forecasting literature recommends to hand-pick a few predictors based on prior causal knowledge (Einhorn and Hogarth 1985; Armstrong, Green, and Graefe 2015). Another important consideration is the number of observations per variable. For example, a standard least squares regression model cannot find a model if there are more variables than observations in the data. Penalized regression methods, like lasso and ridge, can produce a solution in that case (Chetverikov, Liao, and Chernozhukov 2020). However, the rate of convergence of a non-parametric estimator depends on the dimension of the input space (Stone 1982). Convergence rates for machine learning models can be low and the theory-based estimates of convergence rates is mostly not practical in real-world situations.[4] The potential problem with slow convergence, especially in high-dimensional settings, is that a model may show high variance and thus perform poorly.

On the other hand, some studies have reported successes in forecasting with large sets of variables (Medeiros et al. 2021; Chen et al. 2022)—but at the cost of interpretability: The more features are used in a model, the more difficult it is to understand and communicate the model independent of the type of model used.

A common approach to increase the predictive performance and simplify the prediction model is to calibrate it on common factors that provide a lower-rank representation of the large set input variables (Stock and Watson 2002; Kim and Swanson 2018). However, this approach also makes the interpretation of the resulting model challenging, as the data-driven factors do not necessarily have a clear interpretation. In contrast, using a small hand-picked set of diverse predictors allows us to interpret their relationship with the response variable as learned by the prediction models. But this might lead to a decrease in performance in some data sets. Giannone, Lenza, and Primiceri (2021) use Bayesian modeling on a handful of data sets to show that selecting a small set of predictors from a large set of variables is often not feasible without trading off the performance of a linear model.

[4]They may, nevertheless, be estimated empirically if enough observations are available.

Two other aspects that need to be considered are data revisions and a reporting lag. Macroeconomic data are often substantially revised (Runkle 1998). Using the most recent vintage in pseudo out-of-sample forecasting removes the data uncertainty, but revised data cannot be used when the forecasting model is used in real time to make predictions about the future. Furthermore, data are often only reported with a delay, e.g., GDP growth for this month might only be published the following month. In this case, a real-time forecast 12 months ahead on this variable is actually a 13-month-ahead forecast.

Finally, in a forecasting setting, the modeler needs to determine how many observations should be used to train the model. There is a trade-off between using all past data, which improves the convergence of the model, or only recent data, which avoids that the model is trained on observations that do not reflect the present and future due to structural shifts over time.

2.2 Step 2: Variable Importance

Variable importance measures usually answer one of two questions. How important is a variable for a model's performance $\mathcal{P}$? Or, how important is a variable to generate a predicted value $\hat{y}$? The two questions are related: the more accurate a model is, the closer $\hat{y}$ is to y and thus the more similar the two metrics of importance will be.

Measures related to $\mathcal{P}$ often are *global*, which means they provide a single number for each variable and model across the test set.[5] This is practical for communication, as one obtains a simple variable ranking. Global measures, however, can obscure many nuances of a model. For instance, machine learning models are non-linear (often non-monotonic) and, as such, a global measure risks oversimplification or producing inconclusive results when evaluated across differing domains of the input space.

[5]Variable importance can be evaluated on any fraction of the test set. Evaluation of the training set has to be interpreted with caution because of overfitting. However, comparisons across training and test sets can help to identify problems of model generalization, such as overfitting.

Local importance measures decompose individual predictions $f(x_t)$ of observation t into attributions of the individual features:

$$f(x_t) = \sum_{k=0}^{N} \phi_k(x_t)\,, \tag{1}$$

with ϕ_0 being a model baseline value (intercept). Equation (1) defines an additive feature attribution. The advantage of feature importance measures of this form is that it provides more detailed information. For instance, comparing inputs x_k with attributions ϕ_k provides the functional form of feature k learned by this model. Furthermore, any local measure can provide global information via aggregation.

We employ two feature importance measures that are model agnostic, unlike other approaches, such as Gini impurity (Friedman, Hastie, and Tibshirani 2009; Kazemitabar et al. 2017), that are only compatible with specific machine learning methods. We argue for the use of *Shapley values* (Shapley 1953; Štrumbelj and Kononenko 2010; Lundberg and Lee 2017), which are local and of the form (1), and contrast them with *permutation importance* (Breiman 2001a; Fisher, Rudin, and Dominici 2019), a simple global measure. A concise technical description of both measures is provided in Appendix B.

The idea of these and other feature importance measures is to either remove the information of the variables of interest or that of the other variables in the model, and then to observe how model outputs change. Permutation importance does this by randomly shuffling the values of a variable to observe how much the performance of the model deteriorates over the test set. Shapley values, on the other hand, explain individual predictions by measuring the contribution that a variable makes on top of others in the model. Shapley values have the advantage that they come with a set of appealing mathematical properties inherited from their game-theoretic origins (Young 1985; Lundberg and Lee 2017). In particular, Shapley values are the only variable attribution scheme which provides accurate local, linear attributions (Equation (1)), respects null contributions, and is consistent. Consistency is a monotonicity property, that is, if a variable is more important in a model compared with another model, then it should also have a larger importance attributed to it. Most popular feature attribution metrics do violate consistency (Lundberg et al. 2020), which makes Shapley values the preferred importance measure, especially for local attribution. Computing Shapley values is computationally more demanding than computing permutation importance. However, there exist accurate approximations that substantially reduce the computation time of Shapley values which we will investigate as well.

While the primary goal of a model comparison (Step 1 of the workflow) is to identify the most accurate model, it is also informative for the modeler to compare different machine learning methods in their variable importance. Strong disagreements in the importance or functional forms learned by the models can be an indication that the modeling needs to be refined. The better aligned the models are in how they use the predictive variables, the more confident the modeler can be that the models generalize well to the data-generating process.

The information derived from global and local feature importance measures is descriptive. They do not by themselves provide measures of certainty, i.e., an estimate on how certain one can be that a variable is actually important to describe or predict the outcome. This is the realm of statistical testing, e.g., in the form of hypothesis testing.

2.3 Step 3: Shapley Regressions

Linear regression-based models are the workhorse in many applied settings, as they allow for standardized and well-established communication. They achieve that by means of regression coefficients and statistical tests that show whether coefficients are different from zero. The canonical test against the null that there is no effect means that we test if there is a significant alignment between a variable of interest and the target (reject the null). We can ask the same about local attributions coming from the variables components ϕ_k in Equation (1) within a linear regression setting (Joseph 2019),[6]

$$y_t = \phi_0^S + \sum_{k=1}^{N} \phi_k^S(x_t)\beta_k^S + \epsilon^S \quad \text{with} \quad \mathcal{H}_k^0 : \{\beta_k^S \leq 0 \,|\, x_t \in \Omega\}. \tag{2}$$

[6]This approach differs from the previous use of Shapley values in econometrics to analyze multicollinearity (Lipovetsky and Conklin 2001).

Equation (2) is almost identical to a standard linear regression, with two differences.[7] First, the null hypothesis $\mathcal{H}_k^0$ includes negative values. This is because Shapley values absorb the sign of a contribution, such that only significant positive values for β^S mean alignment. Second, inference from Equation (2) is only valid within a region Ω, usually the test set. This is because non-linear machine learning models may show alignment with the target only in bounded regions of the input space. Non-linearity also means that we cannot summarize a variable's importance by a single coefficient universally. We can, however, define something akin to a linear regression coefficient within a region Ω. Let *sign* be the sign of the coefficients when regressing y on x, and let $\psi_k^t = \frac{|\phi_k^S(x_t))|}{\sum_{l=1}^N |\phi_l^S(x_t)|}$ be the share of absolute Shapley values of observation t attributed to variable k. The average share of Shapley values across all observations in Ω is denoted by $\bar{\psi}_k = \frac{1}{|\Omega|}\sum_{x_t \in \Omega} \psi_k^t$.[8] Further, let $(*)$ indicate the confidence level with which we can reject $\mathcal{H}_k^0$ according to Equation (2); then we define the following:

$$\textit{Shapley share coefficient:} \qquad \Gamma_k \equiv sign\, \bar{\psi}_k^{(*)} \quad \in \quad [-1, 1]\,. \tag{3}$$

Shapley share coefficients can be communicated as commonly used regression coefficients in a well-known table form. The interpretation of Γ_k is also similar to that of a regression coefficient, as it measures strength and confidence in alignment with the target variable. However, it cannot be interpreted as a marginal effect, unless the model is linear. In this case, Shapley share coefficients are aligned with the actual linear regression coefficients.

3. Data and Experimental Setup

We describe the notation, data, and experimental procedure for the macroeconomic forecasting exercise which we use to demonstrate the proposed machine learning workflow.

[7]Equation (2) is based on generated regressors (Pagan 1984). The validity of inference and asymptotic properties of estimating the β^S are discussed in detail in Joseph (2019).

[8]Shapley values do not have a natural scale on which to represent them and they can change alongside the region Ω being considered. This motivates the normalizing denominator in the definition of ψ_k^t.

We first introduce the necessary notation. Let y and $\hat{y} \in \mathbb{R}^T$ be the observed and predicted outcome, respectively, where T is the number of observations in the time series. The feature matrix[9] is denoted by $x \in \mathbb{R}^{T \times N}$, where N is the number of features in the data set. The feature vector of observation t is denoted by x_t. Generally, we use t to index the point in time of the observation and k to index features. The forecasting horizon in months is denoted by h. The forecasting horizon is a crucial aspect regarding the purpose of a forecast. One does not necessarily expect models to perform equally well or to pick up the same information across horizons. The default discussed in this paper is the one-year forecast ($h = 12$), sitting between short- and medium-term projections.

3.1 Data

We use the FRED-MD macroeconomic database (McCracken and Ng 2016), which contains monthly macroeconomic indicators for the United States. Our vintage of the data goes from 1959 to 2019. Our forecast target is changes in unemployment, and we hand-pick nine variables as predictors in our baseline approach, each capturing a different macroeconomic channel. We use the stationarity transformations suggested by the authors of the data set that include first differences ($\Delta^l(x) = x_t - x_{t-l}$), log differences ($\Delta^l log(x)$), and second-order log differences ($\Delta^l log(x_t) - \Delta^l log(x_{t-l})$). Given that we predict the yearly change of unemployment, we set transformation span l to 12 for the outcome and lagged outcome (predictor) variables. For the remaining predictors, we set $l = 3$ in our baseline setup. Table 1 shows the variables, with the respective transformations and the series names in the original database. The augmented Dickey-Fuller test confirms that all transformed series are stationary ($p < 0.01$). Different choices for handling the data, like choosing l as well as the aspects discussed in Section 2.1, are investigated in detail in Section 5.

[9]Features or predictors in the machine learning literature correspond to independent variables, or just variables. The observed response, outcome, or dependent variable is often referred to as the target.

Table 1. Series Used in Our Baseline Forecasting Experiment

Variable	Transformation	Name in the FRED-MD Database
Unemployment	Changes	UNRATE
Three-Month Treasury Bill	Changes	TB3MS
Real Personal Income	Log Changes	RPI
Industrial Production	Log Changes	INDPRO
Consumption	Log Changes	DPCERA3M086SBEA
S&P 500	Log Changes	S&P 500
Business Loans	Second-Order Log Changes	BUSLOANS
CPI	Second-Order Log Changes	CPIAUCSL
Oil Price	Second-Order Log Changes	OILPRICEx
M2 Money	Second-Order Log Changes	M2SL

Note: The middle column shows the transformations suggested by the authors of the FRED-MD database, the right column shows how the series are named in that database. Target audience: analysts.

3.2 Models

We test two types of models, a simple autoregressive model and several full-information models containing lags of the response variable and the features. The latter group is further split into linear regression models, with and without penalization, and non-linear machine learning models.

- The *autoregressive model (AR)* uses lagged values of the response variable as predictors: $\hat{y}_t = \alpha + \sum_{l=1}^{L} \theta_l y_{t-l}$. We test AR models of lag lengths $1 \leq L \leq 12$, where we chose L using the Akaike information criterion in the training set. We also test a simple AR1 model by setting $L = 1$. The model's fitted coefficients are given by $\theta \in \mathbb{R}^L$. Forecasts over a horizon h are obtained iteratively from $\hat{y}_{t+h} = \alpha + \sum_{l=1}^{L} \theta_l \hat{y}_{t+h-l}$.
- The *full-information models* use the h-month lag of the outcome variable and the other features as independent variables: $\hat{y}_t = f(y_{t-h}, x_{t-h}; \theta)$, where f is any given predictive model. For example, if f is a linear model, a horizon-h projection takes the form $\hat{y}_t = \alpha + \theta_0 y_{t-h} + \sum_{k=1}^{N} \theta_k x_{t-h,k} + \epsilon_t$, with ϵ_t being the error term. To simplify notation in what follows, we include the lagged outcome in the feature matrix x. We test seven full-information models: ordinary least squares (OLS) regression, regularized regression with ridge and lasso penality, and four machine learning models: random forests (Breiman 2001a), gradient boosting (Friedman 2001), support vector regression (Drucker et al. 1997), and artificial neural networks (Goodfellow, Bengio, and Courville 2016). Table B.1 in Appendix B provides details on the implementation of the models.

3.3 Experimental Procedure

We evaluate how all models predict $l = 12$ month changes in unemployment $h = 12$ months ahead in a pseudo out-of-sample setting with an expanding horizon. All methods are evaluated on the 359 data points of the forecasts between January 1990 and November 2019 using an expanding window approach. We choose the absolute error as our performance metric. It is easy to interpret and less sensitive to outliers than the squared error. Accordingly, we pick the hyperparameter values that minimize absolute error.[10]

We fit, i.e., train, the *AR* models every month. The full-information models are trained every 12 months such that each model makes 12 predictions before it is updated. Compared with updating the models every month, this reduces the computational cost considerably, while only minimally affecting model performance in normal times. However, one may refit a particular model more frequently during operation. Especially machine learning models can be quick in picking up different or new (economic) regimes, as we will see below.

As the models predict changes h months ahead, we have to create an initial gap between training and test set when making predictions to avoid a look-ahead bias. For a model trained on observations $1 \dots t$, the earliest observation in the test set that provides a pseudo real-time h-month forecast is $t + h$. For observations $t + 1, \dots, t + h - 1$, the time difference from the last observation in the training set t is less than one year.

Most of the models we use can be affected by outliers. We therefore test how winsorization of the features at the 1st and 99th percentile affects the predictive performance. We do not winsorize the response variable and the lagged response that is used as a feature.

All machine learning models that we test have hyperparameters which need calibration. We use two types of cross-validation for the hyperparameter tuning. First, we employ ordinary fivefold cross-validation (see Chakraborty and Joseph 2017), which does not consider temporal dependencies in the data, but randomly assigns the observations in the training set to five folds. Second, we use fivefold block cross-validation (Snijders 1988; Bergmeir and Benítez 2012), where the folds are assigned to five consecutive blocks. This approach respects the temporal dependency of the training and test data. More concretely, we use *hv-block cross-validation* (Racine 2000), which additionally introduces a gap of 12 months between blocks of the training and test set. We employ a random search across 100 hyperparameter combinations and pick the hyperparameters that minimize the mean absolute error.

[10]Note however, that different models have different loss functions. Minimizing these is not necessarily equivalent to minimizing the absolute error.

As the hyperparameter optimization is computationally expensive, we conduct it only every 36 months. Even during operation, it is unlikely that hyperparameters need to be updated with a higher frequency unless one expects dramatic model changes, e.g., due to a large change in the data-generating process. Smaller changes will be reflected in the changes in the model parameters (e.g., the weights of the neural network) rather than hyperparameters (e.g., the architecture of the neural network).

To increase the stability of the full-information models, we train each model 30 times on different bootstrapped samples of the training set and average their predictions. This bootstrap aggregation approach is also referred to as *bagging* in the literature (Breiman 1996). Each of the 30 models uses the same hyperparameters, which are calibrated on the full training set.

To estimate how stable our models are, we repeat each experiment—including the training of the 30 bootstrapped models every 12 months and the hyperparmaeter search every 36 months—10 times for all methods, each time with a different random seed. Sources of randomness that can lead to performance differences between these 10 iterations are the chosen hyperparameters,[11] the random bootstrap samples, and random initializations of the random forest, gradient boosting, and neural networks.

4. Workflow Output

This section presents the results of our workflow when applying it to the baseline setting for forecasting changes in the U.S. unemployment rate on a one-year horizon as described in the previous two sections. Not all results presented here are meant to be communicated during operation. Rather, we also present additional analyses that are only relevant for the technical expert that is developing a forecasting model or that are shown for illustration purposes to help the reader better understand the technicalities of the workflow.

[11]Both the randomly selected hyperparameter combinations and the random assignments to folds when using k-fold cross-validation (folds in hv-block cross-validation are not randomly assigned) can induce randomness.

4.1 Step 1: Model Performance

Table 2 summarizes the empirical performance of the different forecasting models. For this table and the following analyses, we applied winsorization to all models and used hv-block cross-validation for the hyperparameter search. Further, to obtain more stable predictions, we average the predictions of 10 models, each trained with a different random seed. In the table, the models are ordered by decreasing mean absolute error over the whole test period between 1990 and 2019.

The table also breaks down the performance in three periods: the 1990s and the periods before and after the global financial crisis (GFC, September 2008). The best model in the individual periods is highlighted in bold. We statistically compare the error of the best model in each period against all other models using a Diebold-Mariano test.[12]

All machine learning models outperform the linear models on the whole sample. In the 1990s and the periods before the global financial crisis, the difference in performance between the models is rather small compared with the period after the crisis. This is indicative that machine learning models may be particularly suited for detecting regime shifts or the modeling of non-linearities, both aspects we will investigate in more detail.

The simple AR_1 model performs better than the AR_{12} model and the linear full-information models. Ridge and lasso regression perform very similarly, both outperforming the OLS regression. In the following analyses we will only consider one AR model, the AR_1, and will focus on one regression model, the ridge regression.

Apart from the aggregated performance across the test period, it is informative to look at the models' individual predictions. Figure 1 (top panel) shows the observed response variable and the predictions of gradient boosting, ridge regression, and the AR_1 model. The bottom panel shows the prediction error. The vertical lines indicate

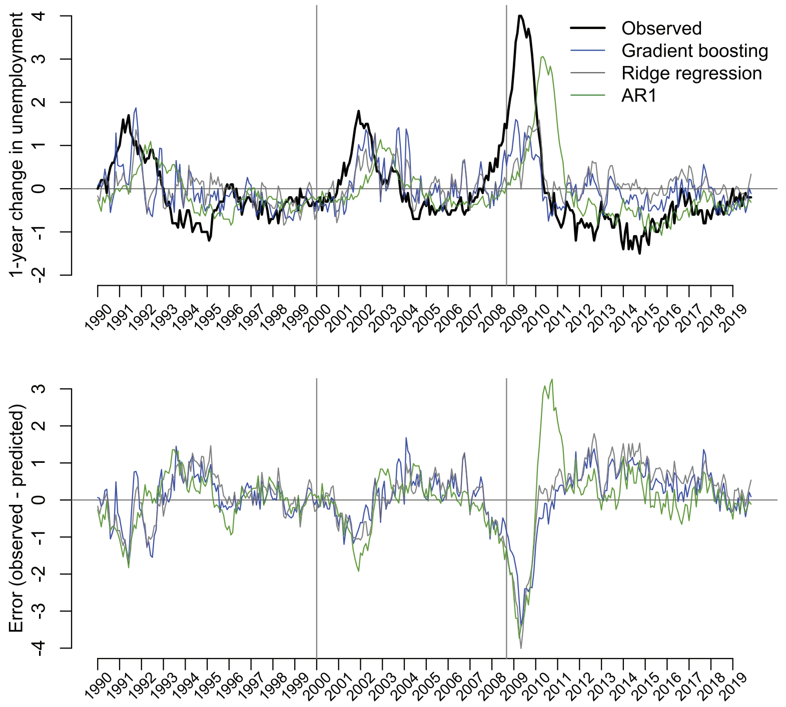
[12]The horizon of the Diebold-Mariano test is set to 1 for all tests. Note however, that the horizon of the AR models is 12 so that the p-values for this comparison are biased. Setting the horizon of the Diebold-Mariano test to 12, we do not observe significant differences between the absolute error of the best model and the AR.

Table 2. Forecasting Performance for the Different Prediction Models in the Baseline Setup

Time Period	**01/1990–11/2019**	**01/1990–12/1999**	**01/2000–08/2008**	**09/2008–11/2019**
Gradient Boosting	**0.559** —	**0.460** —	**0.466** —	0.718 (0.353)
SVR	0.565 (0.323)	0.470 (0.328)	0.489 (0.219)	**0.709** —
Forest	0.581 (0.018)	0.472 (0.240)	0.471 (0.413)	0.762 (0.005)
Neural Network	0.589 (0.009)	0.468 (0.336)	0.503 (0.070)	0.762 (0.001)
AR_1	0.608 (0.063)	0.472 (0.382)	0.503 (0.216)	0.811 (0.064)
AR_{12}	0.626 (0.001)	0.543 (0.011)	0.482 (0.356)	0.810 (0.001)
Lasso Regression	0.637 (0.000)	0.498 (0.061)	0.474 (0.378)	0.886 (0.000)
Ridge Regression	0.639 (0.000)	0.497 (0.065)	0.481 (0.272)	0.886 (0.000)
OLS Regressions	0.648 (0.000)	0.516 (0.016)	0.508 (0.053)	0.872 (0.000)

Note: The models are ordered by decreasing MAE on the whole sample. The best-performing model in each time period is highlighted in bold. The p-values in parentheses indicate the statistical significance (one-sided) of the Diebold-Mariano test comparing the best model in each column with the other models. Target audience: management to decisionmakers.

Figure 1. Comparison of Observed and Predicted Outcome



Note: The top panel shows the observed one-year change in unemployment and the predictions by the gradient boosting model, ridge regression, and the AR_1. The bottom panel shows the error of these two methods. Target audience: management or decisionmakers.

the different time periods distinguished in Table 2. All three models underestimate unemployment growth during the global financial crisis and overestimate it during the recovery. However, the gradient boosting model is least biased in those periods and forecasts the increase in unemployment earlier during the crisis. A similar observation can be made after the burst of the dot-com bubble in the early 2000s. Such a chart can be presented to the policymaker to convey the model's performance in a clear and detailed way.

4.2 Step 2: Feature Importance

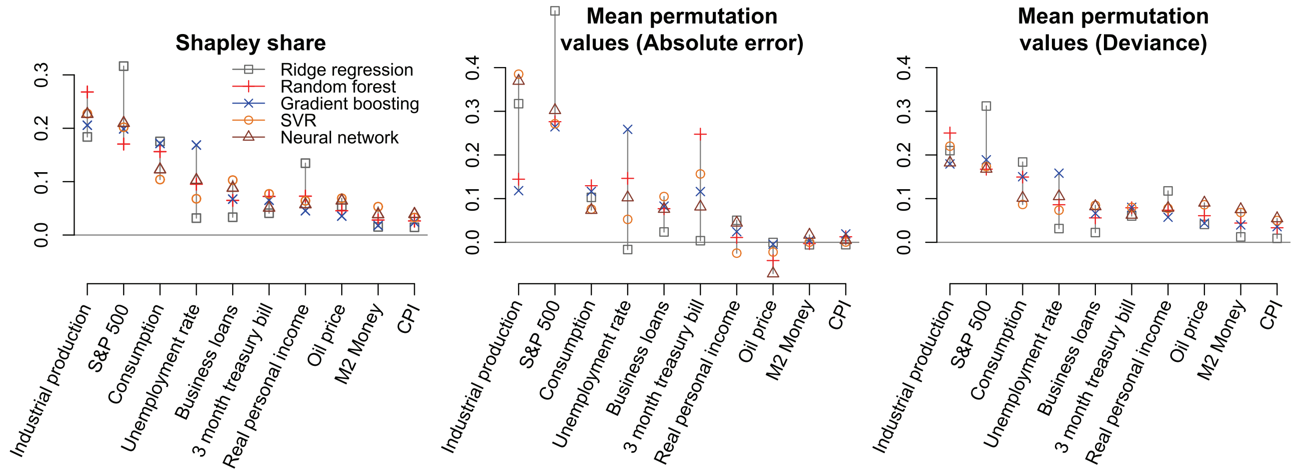
We explain the predictions of the machine learning models and the linear regression as calibrated in our baseline setup. Our focus is largely on explaining forecast predictions in a pseudo real-time setting. However, in some cases it can be instructive to explain the predictions of a model that was trained on observations across the whole time period. For that, we exploit the fact that we trained the models on 30 different bootstrapped samples across the whole time series. Each of these models can make predictions on those observations not in the bootstrapped training sample. In this way we obtain several predictions for each observation in the time series, which are then averaged. This *out-of-bag* analysis is subject to look-ahead bias, as we use future data to predict the past.

We first analyze our two methods of model interpretation at a global level. Figure 2 compares Shapley shares $|\Gamma^S|$ (left panel) with permutation importance (middle panel). The variables are sorted by average Shapley shares of the four machine learning models. Vertical lines connect the lowest and highest share across models for each feature to highlight the disagreement between models.

Shapley values and permutation importance do not agree in their ranking of feature importance. For instance, using a random forest model, the three-month Treasury bill seems to be a more important indicator according to permutation importance than according to Shapley calculations.

The permutation importance is a measure of a feature's influence on the accuracy of the model and is affected by how the relationship between outcome and features changes over time. In contrast, Shapley values reflect a variable's influence on the predicted value, independent of that value's accuracy. Arguably this

Figure 2. Variable Importance According to Different Measures



Note: The left panel shows the importance according to the Shapley shares $|\Gamma^S|$ and the middle panel shows the variable importance according to permutation importance. The right panel shows an altered metric of permutation importance that measures the effect of permutation on the predicted value rather than prediction error. Target audience for the left panel: decisionmakers. Permutation importance is shown for illustrative purposes.

measure of importance is more useful in a forecasting setting when the variable importance should be computed for data points for which the true outcome has not been observed yet, which means that permutation importance is not computable. The right panel of Figure 2 shows an altered measure of permutation importance. Instead of measuring the change in the error due to permutations, we measure the change in the predicted value.[13] We see that this importance measure is more closely aligned with Shapley values. Further, when we evaluate the error-based permutation importance metric using predictions based on the out-of-bag analysis, we find a strong alignment with Shapley values (not shown), as the relationship between variables is not affected by the changes between the training and test set.[14]

Overall, the different prediction models have a similar importance ranking of the features according to the Shapley share. There are, however some notable differences—especially the ridge regression model often differs substantially from the other models in the Shapley shares. Even the different machine learning models do not completely agree on the relative importance of features. For example, gradient boosting gives more importance to the lagged unemployment indicator than the other methods.

While the computation of Shapley values is technically rather complex and difficult to communicate to a non-technical audience, we believe that the intuition behind Shapley values as the contribution to the model's predictions is easy to understand. Thus, a chart such as the left panel of Figure 2—but only showing the best-performing model—can be communicated to decisionmakers.

This global analysis only conveys which variables are important across all observations in the test set. Local attributions will often be more useful in a pratical setting, as they allow to assess individual, e.g., the latest, predictions.

[13]This metric computes the mean absolute difference between the observed predicted values and the predicted values after permuting feature k m times: $\frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - \hat{y}_{i(k)}^{perm}|$. The higher this difference, the higher the importance of the feature k (see Lemaire, Féraud, and Voisine 2008 and Robnik-Šikonja and Kononenko 2008 for similar approaches to measure variable importance).

[14]Comparing out-of-sample and out-of-bag measures allows to evaluate model drift and look-ahead bias more generally.

Local attributions also reveal the functional form learned by a model. To illustrate this, we consider the out-of-bag predictions, abstracting away from model drift which we discuss in a moment. Here, the most accurate models are the gradient boosting model (absolute error of 0.431) followed by the random forest (0.450), the SVR (0.452), the neural network (0.452), and ridge regression (0.584).[15]

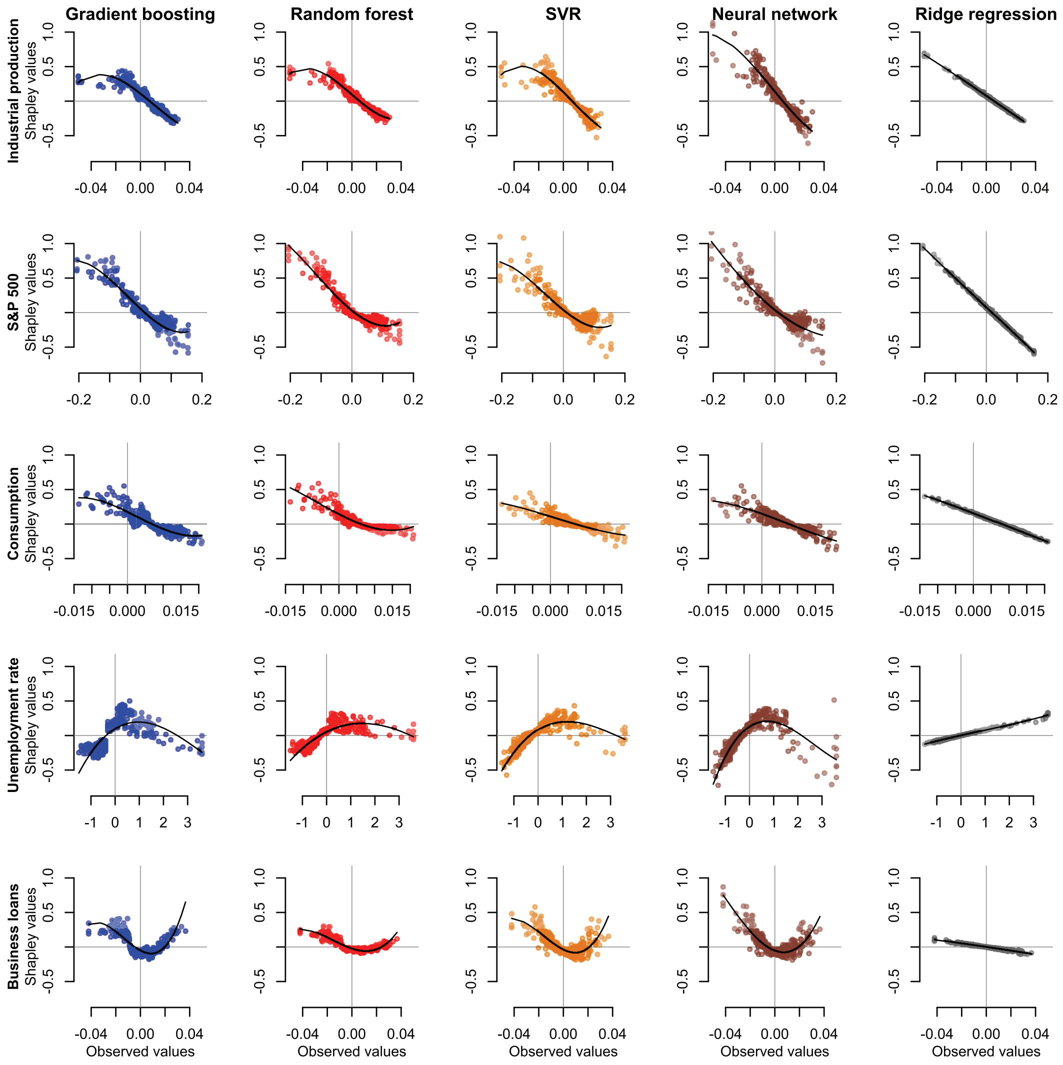
Figure 3 shows the functional forms that the machine learning models have learned from the most important features according to Shapley shares shown in Figure 2 (left panel). It depicts local Shapley values against the observed input values (horizontal axis), with rows showing the variables and columns the different models. The approximate functional form learned by each model for each feature is traced out by a best-fit third-degree polynomial. Although the four machine learning methods use very different learning mechanisms and even do not agree perfectly on the global importance of features, the functional forms learned by all of them are highly consistent for all variables shown. This gives us confidence that the functions learned are meaningful and robust.

For example, consider the S&P 500. The ridge regression learns a steep negative slope with higher stock market values being associated with lower unemployment one year ahead. This makes economic sense. However, we can make more nuanced statements when looking at the other models. There is an asymmetry between market increases and decreases. While large decreases suggest large increases in unemployment down the line, there is a saturation effect for high market valuations with only a small expected decrease in unemployment.

For unemployment, all machine learning models learn a quadratic function. A high increase in unemployment makes future increases in unemployment less likely compared with a medium increase. For business loans we also observe a quadratic function, where very low and high loans lead to a positive predicted change in unemployment. In contrast, the linear model cannot model quadratic trends, so it is

[15]As in the forecasting setting, winsorization is applied, as it helps the performance of the SVR (see Section 5), while it does not substantially affect the other models' performance. In the out-of-bag analysis, we use k-fold cross-validation rather than blocked cross-validation, as this generally improves the performance.

Figure 3. Functional Forms Learned by Different Models for Five Features with the Highest Average Shapley Share



Note: The lines show a third-degree polynomial fitted to the data. The Shapley values are computed on the out-of-bag predictions and are therefore subject to look-ahead bias. Target audience: analysts (comparison); decisionmakers (single model if robust).

not surprising that the Shapley share of these two variables (Figure 2, left panel) are substantially smaller according to the linear model compared with the machine learning models.

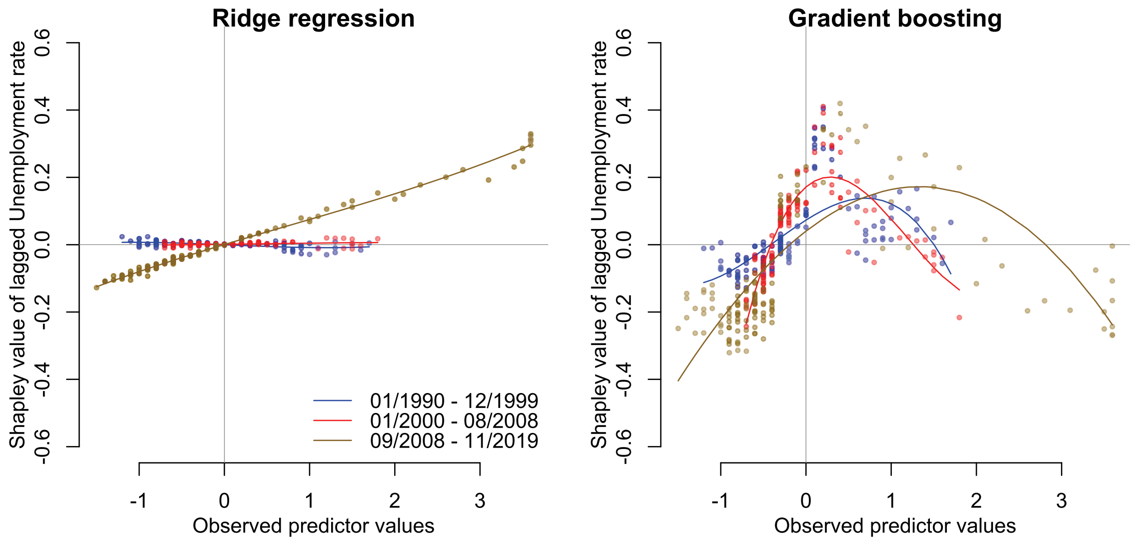
Figure 3 serves several purposes. The functional forms learned from the data, although not causal, might provide new economic insights about the underlying processes. These can then be further investigated by, for example, using structural models. Further, by providing information about the inner workings of different models, these charts can be used as a diagnostic tool for the technical expert training and tuning the model. For instance, if the functional forms learned by an SVR are mostly linear whereas those of the other machine learning models are not, this might suggest a problem constraining the flexibility of the SVR. Finally, by evaluating the functional forms learned at different points in time, model drift or structural breaks can be detected.

For example, we consider the out-of-bag predictions of the models trained up to three different points in time. Figure 4 shows the functional form for the lagged unemployment change variable. The ridge regression model (left panel) trained up to the periods 2000 and 2008 finds no predictive power for lagged unemployment. It is only after the onset of the GFC that the regression learns a positive relationship—an increase of unemployment increases the predicted unemployment change one year ahead. However, this is simply reflective of the trend—the one-year unemployment change was high for a prolonged period following the financial crisis: it was persistently greater than or equal to 1 percentage point for 23 consecutive months (May 2008–March 2010). In contrast, the functional form of the gradient boosting model (right panel) is rather stable. Across the three time periods it learns a non-monotonic relationship where high absolute values in the unemployment make future increases in unemployment less likely compared with small changes. The scale of this learned functional form increased after the GFC in line with larger movements in the unemployment rate during this time.

To better understand the non-monotonic function of lagged unemployment change learned by the gradient boosting model, we look into the role of recessions within the model.[16] Figure 5 (left)

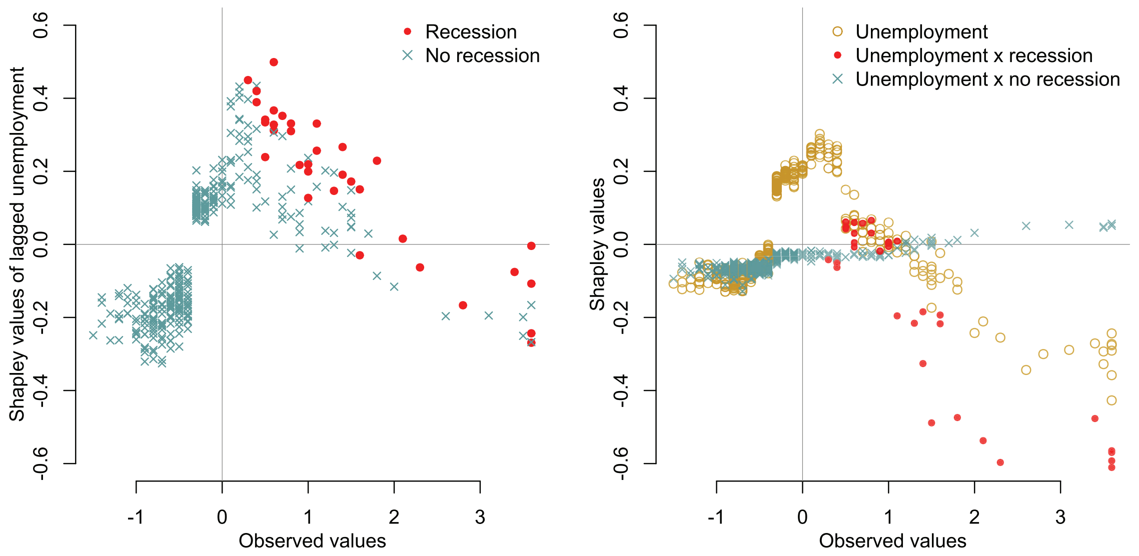
[16]We use the definition of recessions provided by the Federal Reserve Bank of St. Louis (Federal Reserve Bank of St. Louis 2020).

Figure 4. Functional Form of Lagged Unemployment Change Learned by Ridge Regression (left panel) and Gradient Boosting (right panel) for Three Models Trained Up to Different Points in Time



Note: The lines show third-degree polynomials fitted to the data. Target audience: analysts.

Figure 5. Interaction between Unemployment Changes and Recessions as Learned by Gradient Boosting Model



Note: The left panel shows the functional form of lagged unemployment changes when the model is trained on the baseline features without a recession dummy (as in Figure 4). The right panel includes the Shapley values of the interactions when the model was trained with a recession indicator. Target audience: analysts (comparison); decisionmakers (recession effect only, right-hand side).

again shows the functional form of lagged unemployment as learned by the gradient boosting model in the out-of-bag setup. But now recession observations (also lagged by a year) in the input space are highlighted in red. Even though we did not include recessions explicitly as an indicator the model could learn from, these periods account for a large share of the downward-sloping part on the right-hand side. This makes economic sense as larger current increases of unemployment during recessions make future decreases of unemployment during the recovery period more likely.

We further elaborate on this observation by including a lagged recession dummy in our models and compute the Shapley-Taylor index (Agarwal, Dhamdhere, and Sundararajan 2019) to decompose the predictions into the main effects from past unemployment, the recession dummy, and their interaction.[17]

The Shapley values of this interaction as well as the main effect of lagged unemployment are shown in the right panel of Figure 5. The main effect of lagged unemployment still shows the inverted U-shaped form—even after controlling for interactions with all other variables. The Shapley values of the interaction show that during a recession thereis an additional strong negative effect of lagged unemployment on the prediction for larger input values (red), which is in line with the above "reversal-to-the-mean" explanation.

While including the recession indicator improves the interpretation of the results and the interaction with unemployment has high Shapley values that contribute substantially to the prediction, the predictive accuracy of the gradient boosting model does not increase meaningfully. Adding the recession indicator, the forecast error only slightly decreases from 0.559 to 0.554. This suggests that the model learned the role of recession periods implicitly, incorporating two different regimes, normal times and recessions.

4.3 Step 3: Statistical Inference with Shapley Regressions

Shapley value-based inference (Equation (2)) allows us to communicate machine learning models analogously to a linear regression analysis. In Table 3, we present the Shapley regression for the full

[17]We follow Joseph (2019) in his empirical approach and group all remaining variables into a single "other" variable to reduce computation time. We compute the Shapley Taylor expansion to the third order (see Section 6) such that two-way interaction terms are unbiased.

Table 3. Shapley Regression of Gradient Boosting Mode (left) and the Ridge Regression (right) for the Forecasting Predictions between 1990 and 2019

	Gradient Boosting			Ridge Regression		
	β^S	p-value	Γ^S	β^S	p-value	Γ^S
Industrial Production	1.132	0.000	–0.217***	2.280	0.000	–0.185***
S&P 500	0.942	0.000	–0.191***	0.907	0.000	–0.317***
Consumption	1.103	0.000	–0.177***	0.966	0.012	–0.173**
Unemployment	1.443	0.000	+0.175***	9.789	0.000	+0.031***
Business Loans	3.086	0.000	–0.066***	5.615	0.006	–0.035***
Three-Month Treasury Bill	4.273	0.000	–0.062***	–6.816	1.000	–0.042
Personal Income	–0.394	0.682	+0.04	–0.658	0.870	+0.138
Oil Price	0.298	0.387	–0.035	–2.256	0.973	–0.055
CPI	0.272	0.438	+0.021	–4.294	0.875	+0.014
M2 Money	–8.468	1.000	–0.016	–18.545	0.994	–0.009

Note: Significance levels: *$p < 0.1$; **$p < 0.05$; ***$p < 0.01$. Target audience: analysts (comparison); decisionmaker (left-hand side).

out-of-sample forecasting period between 1990 and 2019 based on the predictions of the gradient boosting model. For illustrative purposes, the table also shows the Shapley regression for the ridge regression.

As mentioned above, the coefficients β^S measure the alignment of a variable with the target. Values close to one indicate perfect alignment and convergence of the learning process. Values larger than one indicate that a model underestimates the effect of a variable on the outcome. And the opposite is the case for values smaller than one. This can intuitively be understood as the model hyperplane of the Shapley regression either tilting more towards a Shapley component from a variable (underestimation, $\beta_k^S > 1$) or away from it (overestimation, $\beta_k^S < 1$). Significance decreases as the β_k^S approaches zero.

Variables with higher Shapley shares $\left|\Gamma^S\right|$ (same as in Figure 2) tend to have lower p-values. This is intuitive, demonstrating that the model learns to rely more on features that are important for predicting the target variable. However, this does not hold by construction, especially not in a forecasting setting where the relationships between variables change over time. Any statistical significance may disappear in the test set—even for features with high Shapley shares.

One more variable is statistically significant for the gradient boosting method than for the linear model. This is expected given the greater flexibility of machine learning models. It also provides further evidence of how non-parametric methods, like gradient boosting forests, exploit non-linear relationships that linear regression cannot account for (as in Figure 3).

A Shapley regression table can provide meaningful insights for decisionmakers that are acquainted with standard statistical inference for regression. Further, it can help the technical expert to refine the model—for example, by removing variables with negative coefficients—or adjust the period of analysis until the coefficients align better with the target.

5. Robustness

We consider a wide array of alternative choices made during our baseline analysis and how these affect the outputs from the first two steps of our workflow, model performance and Shapley feature importance.[18] We first propose and run a set of analyses that vary key parameters of our experimental setup to test whether our results that machine learning models outperform linear models is robust. Next, we replicate our results using real-time data and show how our workflow can be applied to substantially larger sets of predictors. Finally, we investigate the robustness of the estimation of Shapley values, discussing computationally cheap approximations.

5.1 Experimental Setup

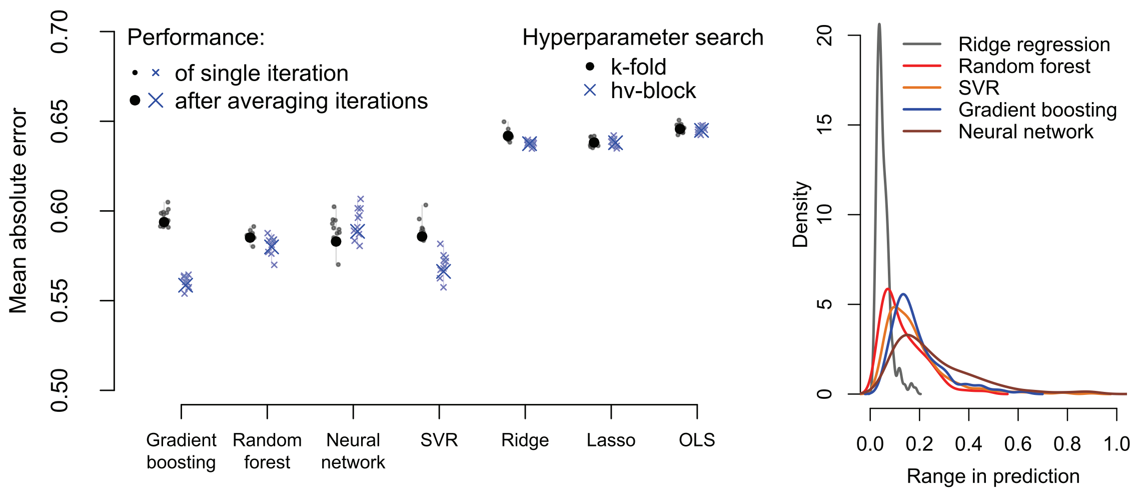
In our main analysis we used a bagging ensemble of 30 models for each of the full-information methods. We show in Figure A.1 (Appendix A) that only the gradient boosting model and the neural network improve using bagging. The figure shows the mean absolute error across the 10 iterations (based on different seeds) as a function of the bagging ensemble size. The confidence intervals ($\pm$ 1.96 standard errors of the mean) of the gradient boosting and the neural network decrease visibly when increasing the size of the bagging ensembles, suggesting that bagging makes these models more stable and thus less sensitive to the random seed. In the following, we use bagging only for these two methods to save computation time.

Further, we used blocked cross-validation for the hyperparameter search and have averaged the predictions of 10 models, each trained on a different random seed. Figure 6 (left panel) investigates the impact of these choices on model performance. It shows the mean absolute error (MAE) across the whole test period between 1990 and 2019 and conveys several findings.

First, the machine learning models, especially the neural network and the SVR, show a substantial variance in performance (smaller transparent dots) for the 10 different iterations based on different random seeds. Averaging the predictions across these iterations (bigger non-transparent points) tends to produce more accurate predictions than the average individual model. This variance in the error across the 10 iterations reflects substantial differences in the predictions on individual data points.

[18]Investigating changes in statistical alignment of feature components (step 3 of the workflow) can be interesting during practical applications, but does not add much value to the discussion here, we believe.

Figure 6. Robustness of Predictions



Note: The left panel shows the MAE of our main prediction models. The small markers show the performance of the individual interations, each based on a different random seed. The larger markers show the average performance across these 10 iterations. For each model we test two types of hyperparameter searches (k-fold versus hv-block cross-validation). The OLS does not have hyperparameters and is not affected by this test. The right panel shows the variation in the predicted values across the 10 iterations. For each observation in the test period (1990–2019), we measure the range of predicted values and show the distribution of this measure in the chart. Target audience: analysts.

To investigate this further we measure, for each observation in the test set, the range of predicted values across the 10 iterations. The right panel of Figure 6 plots the distribution of this range across observations. The 10 models are very similar for the ridge regression, with a mean range of 0.05 (90 percent percentile: $P_{90} = 0.08$). The random forest is less stable, with a mean range of 0.14 ($P_{90} = 0.26$) but a factor of two more stable than the neural network with a mean range of 0.27 ($P_{90} = 0.5$). This is—given a mean absolute error of less than 0.6—a substantial variation in the prediction of the models.

In a practical forecasting setting, the modeler might decide to slightly trade off predictive performance against model stability and choose, for example, the random forest over the SVR. We believe that the repeated training of the same model with different random seeds is crucial to get a sense of the stability of their performance. To stabilize the models and make them less susceptible to the random seeds, we suggest averaging them.

Second, the type of cross-validation employed in the hyperparameter search matters for the performance of some of the methods. The linear models, the random forest, and the neural network do not differ markedly in their performance for the two types of cross-validation. However, for the gradient boosting model and the support vector regression, we observe a substantially better performance when using the blocked cross-validation approach.

Even rather small design factors such as the type of cross-validation can change the conclusion about which model performs best. The fact that this and other factors (see below) affect the performance of the models in different ways suggests that the modeler should conduct a extensive set of experiments before identifying the best prediction model, and also assure its stability.

We next alter several parameters with respect to our baseline setup. The results are shown in Table 4, with the best model in each row highlighted in bold.

- *Prediction Horizon:* In the baseline setup, we have predicted unemployment changes $h = 12$ months ahead. Here, we alter the prediction horizon between 1 and 36 months. We observe that the AR_1 models competes well with the full-information models at prediction horizons 1, 3, and 6 months but falls

Table 4. Performance for Different Parameter Specifications

	Gradient Boosting	SVR	Random Forest	Neural Network	Ridge Regression	AR$_1$
Prediction Horizon h (lag between response and predictors in months)						
1	0.20	0.19	**0.17**	0.18	0.18	**0.17**
3	0.28	0.28	**0.27**	**0.27**	**0.27**	**0.27**
6	0.41	0.41	**0.39**	0.42	0.43	0.41
12 (Baseline)	**0.56**	0.57	0.58	0.59	0.64	0.61
24	0.68	0.67	**0.62**	0.69	0.73	0.79
36	0.64	0.63	**0.61**	0.72	0.72	0.80
Training Set Size (in months)						
60	0.83	0.87	**0.79**	0.84	0.87	0.95
120	0.63	0.67	**0.57**	0.66	0.66	0.71
240	0.58	**0.56**	0.57	0.58	0.61	0.67
360	**0.57**	0.58	0.58	0.60	0.61	0.64
480	**0.56**	0.57	0.57	0.57	0.63	0.62
Max (Baseline)	**0.56**	0.57	0.58	0.59	0.64	0.61
Transformation Span l (in months)						
1	0.57	0.60	**0.55**	0.59	0.64	—
3 (Baseline)	**0.56**	0.57	0.58	0.59	0.64	—
6	**0.60**	**0.60**	**0.60**	0.67	0.66	—
9	**0.65**	0.68	0.67	0.70	0.70	—
12	0.68	0.74	0.70	0.71	0.74	**0.61**
Winsorization at 1 Percent and 99 Percent						
Yes (Baseline	**0.56**	0.57	0.58	0.59	0.64	0.61
No	**0.56**	0.59	0.58	0.60	0.64	0.61

Note: The shown metric is mean absolute error. The best model(s) in each row are highlighted in bold. Target audience: management.

behind when increasing the horizon. This is not surprising, as the autocorrelation in the response variables decreases with increasing h. The table shows that the machine learning models can provide meaningful signals for the unemployment changes at longer horizons, even three years ahead. The good performance of the random forest is notable: For all horizons different from 12, it performs as well as or better than the other models.

- *Window Size:* In the baseline setup, the training set grows over time (expanding window). This can potentially improve the performance, as more observations may facilitate a better approximation of the data-generating process. On the other hand, it may make the model sluggish and prevents quick adoption to structural changes. To differentiate between these two cases, we test sliding windows of 60 to 480 months. All methods perform worst on the smallest horizons of 60, and only the random forest performs well on a sample of just 120 months. Gradient boosting consistently improves its performance with a growing sample size. This is not surprising for machine learning models, as they can learn different regimes for different time periods due to their flexibility and exploit them for prediction. For instance, different paths down a tree model, or different trees in a forest, are all different submodels. By contrast, the ridge regression, like all linear models, cannot adjust in this way and needs to fit the best hyperplane to the current situation. This can explain why its performance declines for sample sizes larger than 360.
- *Transformation Span:* We use $l = 3$ months in the baseline, when calculating first differences, log differences, and second-order log differences of the predictors (see Table 1). Testing lag lengths of 1, 6, 9, and 12 months, we find that shorter horizons of 1 or 3 months generally lead to better performance than longer ones. This is useful from a practical point of view, as quarterly changes are commonly used for short-term economic projections.
- *Winsorization:* Winsorization only helps the SVR and the neural network. It does not have a visible impact on the performance of the other models. As the response variable is not

winsorized, there is, by design, no effect on the performance of the AR_1 model.

Testing different training set sizes, transformation horizons, and winsorization of predictors are crucial to refine and improve the prediction models. The choice of the prediction horizons will be informed by the needs of the decisionmakers. But testing different horizons can help to assess the change in predictability of the response and by explaining predictions (see Section 4.2), one can detect differential signals provided by the predictors at different horizons.

5.2 Real-Time Data

Our pseudo out-of-sample forecasting approach does not reflect how forecasts are made in the real world. When training and testing our models in Section 4.1, we used revised, macroeconomic data. In a practical setting, we have to rely on early vintages that are likely to be revised. We investigate whether the results change substantially when replicating the horse race using real-time data.

There exist monthly vintages of the FRED-MD database[19] starting from August 1999, each providing estimates for the indicators lagging one month behind.[20]

As before, we predict the change in unemployment one year ahead, this time for the period for which we can produce real-time forecasts (August 2000–November 2019). As the real-time data are delayed by a month, an actual one-year-ahead forecast requires lagging the variables by 13 months. More formally, we use the data (features and response) of the vintage at time t, which contains the measurements up to date $t-1$. We train the models with response

[19]https://research.stlouisfed.org/econ/mccracken/fred-databases/.

[20]The *consumption* variable is not included in the vintages before 2004. When using these vintages for training, we use the revised time series from our baseline data set. Further, the variables *business loans* and *real personal income* have missing values in some of the vintages. Again, we replace these missing values with the revised series. Some variables (e.g., real personal income and industrial production), have been re-indexed for the different vintages. This does not affect our modeling, as we use variable transformations such that level differences do not matter.

Table 5. Comparison of the Forecasting Performance When Using Real-Time vs. Revised Data

	Real Time (13 Months)	Real Time (12 Months)	Revised Data (12 Months)
Gradient Boosting	0.63 —	0.62 —	0.62 —
SVR	0.64 (0.33)	0.62 (0.48)	0.63 (0.37)
Forest	0.66 (0.04)	0.64 (0.02)	0.65 (0.02)
Neural Network	0.67 (0.01)	0.64 (0.05)	0.66 (0.01)
AR_1	0.72 (0.04)	0.69 (0.05)	0.69 (0.06)
Lasso Regression	0.73 (0.00)	0.71 (0.00)	0.72 (0.00)
Ridge Regression	0.75 (0.00)	0.72 (0.00)	0.73 (0.00)
Linear Regressions	0.75 (0.00)	0.72 (0.00)	0.73 (0.00)

Note: The performance metric is mean absolute error. The models are tested in the period between August 2000 and November 2019. Target audience: analysts.

$y_{t'}$ and lagged predictors $x_{t'-13}$, where $t' \leq t-1$. To make a prediction one year ahead ($\hat{y}_{t+12}$), we use the latest feature values of the same vintage (x_{t-1}) and compare the prediction against the revised response variable y^r_{t+12}. As in the previous experiments, we update the machine learning models every 12 months, winsorize the features, and use hv-block cross-validation to calibrate the hyperparameters.

Table 5 compares the model performances using real-time (left two columns) and revised data (right column). As in our main empirical analysis (see Table 2), the machine learning methods outperform the linear models, with gradient boosting being the best model. The p-values in parentheses indicate the statistical significance (one sided) of the Diebold-Mariano test, estimating whether the gradient boosting model significantly outperforms the other models.

The predictions based on the revised data are slightly more accurate than those based on the real-time data. This is driven by the fact that the real-time prediction is a 13-month forecast rather than a one-year-ahead forecast because of the reporting lag of one month. The middle column of the table shows the performance when this reporting lag would not exist. Here the real-time data is used to make predictions 12 months ahead of the latest available data, which effectively is a forecast 11 months ahead. The performance differences between these real-time predictions and the predictions based on revised data are small and do not suggest that the models improve when using revised data. This is not surprising, given that the real-time and revised series most often only differ by a small degree, as shown in Figure A.2 in Appendix A. We therefore do not investigate real-time data in detail but focus on the revised data in this study, which allows us to investigate the models over a longer time period.

5.3 Extending the Set of Features

So far, we have used nine hand-picked key features (see Table 1) to predict unemployment changes. However, the FRED-MD database (McCracken and Ng 2016) offers a much richer set of variables—97 of which do not have any missing values between 1959 and 2019. Can we improve the forecasting performance by exploiting all of these? We make the variables stationary by applying the transformations suggested by the authors of the database using a transformation span of $l = 3$ for all variables.

Table 6 compares the performance when using the key features (first column) versus all features (second column). Using all features, the performance of the best models, gradient boosting, as well as the OLS regression, declines, whereas the performance of the other models improves or does not change. The random forest based on all features performs even slightly better than gradient boosting based on the key features. However, the Diebold-Mariano test shows that the difference is not significant (p = 0.72, two sided).

An analysis of the Shapley values shows that the machine learning models do not learn a sparse model when trained on all features. For the three best models—random forest, SVR, and neural network—the Shapley share of the top 10 features, respectively, only accounts for 41 percent, 32 percent, and 34 percent of the variance in the predictions. To account for at least 80 percent, we need to consider at least the top 39, 53, and 47 features, respectively. The large number of variables also increases the disagreement between models. While the agreement in the Shapley share is high between the SVR and the neural network (correlation of 0.93), it is lower between the forest and the other two methods (0.69, 0.70) (see Figure A.3 in Appendix A). Further, unlike the models trained on the key features

Table 6. Comparison of the Forecasting Performance When Using Different Input Data

	Key Features	All Features	PCA$_1$	PCA$_2$	PCA$_3$	PCA$_5$	PCA$_7$
Gradient Boosting	0.56	0.58	0.67	0.53	**0.52**	0.54	0.57
SVR	0.57	0.57	0.61	**0.52**	**0.52**	0.55	0.59
Random Forest	0.58	0.55	0.62	**0.52**	0.53	0.55	0.61
Neural Network	0.59	0.57	0.69	**0.52**	0.53	0.55	0.55
Lasso	0.64	0.63	0.65	0.56	**0.54**	0.56	0.59
Ridge	0.64	0.58	0.65	0.56	**0.54**	0.56	0.58
OLS	0.65	0.80	0.65	0.56	**0.54**	0.56	0.59

Note: The models are trained on the 10 key features, all 97 features, or the k top components of a principal component analysis (PCA$_k$), which was calibrated on all features. The performance metric is the mean absolute error. The best input data for each model (rows) is highlighted in bold. Target audience: analysts.

only (Figure 3), the functional forms do not align well between methods when trained on all features. Figure A.4 in Appendix A shows this for some of the key features.

This is not surprising given the rather small number of observations in our data and the fact that non-parametric convergence often is slow when the number of features is high (Stone 1982).

5.3.1 Dimensionality Reduction

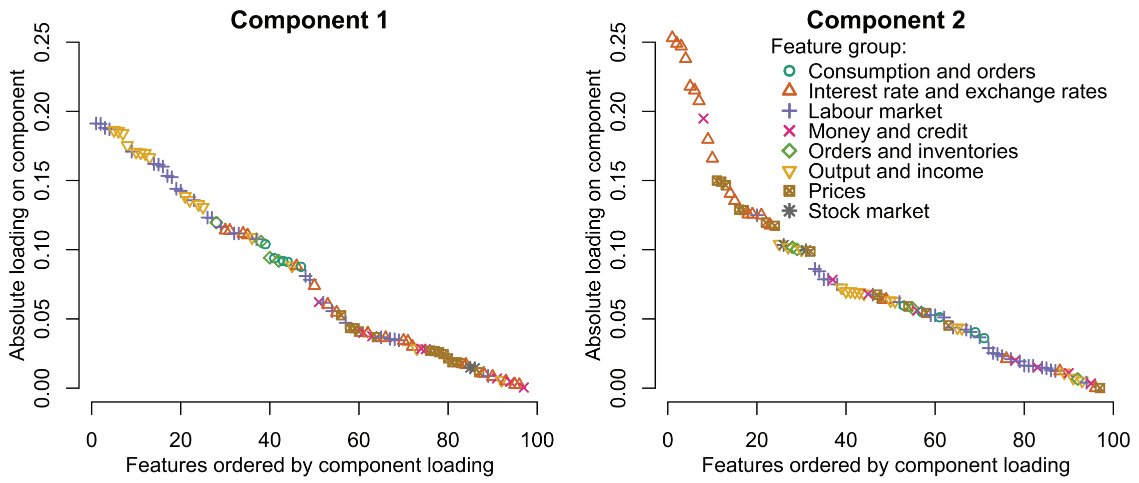
In the literature on economic forecasting, a standard approach to exploit the predictive power of many features is to calibrate a dimensionality reduction model (e.g., principal components analysis, or PCA) and train the prediction models on the most important components (Stock and Watson 2002; Kim and Swanson 2018). Aggregating redundant variables in the same component allows models to learn more effectively from a lower-dimensional feature representation.

We follow this approach and use a PCA to summarize all 97 features.[21] Table 6 shows the performance for the forecast error when the machine learning models are trained on one, two, three, five, and seven components of the PCA. The best performance is achieved when using only two components, with the SVR, neural network, and random forest all performing equally well. Comparing these three models with the gradient boosting model trained on the key features, the Diebold-Mariano test estimates the following p-values (two sided), respectively: 0.054, 0.028, and 0.064. This suggests that using the PCA leads to a superior performance.

A model based on just two components may seem easy to interpret at first sight. However, as shown in Figure 7, the loadings of the 97 variables on these components are not sparse. We show to which group a variable belongs, where the groups have been defined by the authors of the data set (see also Ludvigson and Ng 2009). Most variables with high loadings on the first component belong to the labor market and output and income variable groups, but other variables have substantial loadings as well. Similarly, on the second component, the variables with the highest loadings belong to the

[21]We calibrate the PCA model on the training set only and update it every year as we do for the machine learning models.

Figure 7. Absolute Loadings of the 97 Features on the First (left panel) and Second (right panel) PCA Component



Note: The loadings shown are averages based on 30 PCA models trained on bootstrapped samples of the complete time series. Target audience: analysts.

interest rate and exchange rates group, but other variables also contribute substantially.[22] This suggests that the components do not have a simple economic interpretation.

At the same time, using the PCA components also limits the insights we can draw from the analysis of Shapley values. The first two components only explain 24 percent and 9 percent of the total variance in the data, respectively. Thus, most of the variation in the variables is not accounted for by the first components of the PCA. Further, making a machine learning model learn from only a few components will confine its ability to learn idiosyncratic functions of the individual features underlying that components. Rather, we expect that all functional forms of the variables loading on the same component will be similar.

Figure A.5 in Appendix A supports this conjecture. It shows the Shapley values based on the random forest when trained on the top two PCA components. The features in the top row have a high loading on the first component and low loadings on the second. The opposite holds for the features in the second row. In each row, the features show highly similar functional forms. The functional form of the features lagged unemployment and S&P 500—both included in the set of key features—differ from those shown in Figure 3. We do not observe a quadratic functional form for any of the 97 features when training the models on the PCA loadings, whereas two of the five most important features in our baseline experiment have such a functional form.

While we observe a small performance improvement when using a PCA instead of the hand-picked key features, this comes at the cost of a more complex model that arguably provides less economic insights. Our results partially support the idea of an "illusion of sparsity" (Giannone, Lenza, and Primiceri 2021). The authors used linear models to show that making a model sparse by picking a small set of predictors from the larger set comes with the cost of an inferior predictive performance. We observe the same for our ridge regression for which the absolute error falls by 0.06 and 0.1, respectively,

[22]It is important to note that the variables within the same group are not redundant. For example, the median absolute correlation (after transformation) of all variables within the labor market group and within the output and income group only is 0.29 and 0.43, respectively.

when using all features directly or training the model on three PCA components.

However, the performance gains from exploiting all variables are smaller for the machine learning models. Further, our set of key features was selected based on economic considerations rather than empirical selection and is thus probably not the best possible subset. This suggest that the trade-off between sparsity and accuracy might be less pronounced when using non-linear models because these are able to extract more information from sparse models.

5.3.2 A Richer Lag Structure

Finally, we extend the number of features by adding more lags of the key variables. The minimum lag of 12 months constitutes the prediction horizon. We add additional yearly lags from 24 to 72 months.[23] Table 7 shows the results of that experiment. While most models improve when adding more lags, the performance of the SVR and the neural network does not.

The best performance is achieved by the gradient boosting model when trained on annual lags of the last four years. We take a closer look at this model. Figure A.6 in Appendix A shows the functional forms for the different lags of the top features. The 12-month lags of the variables contribute most to the predictions. The other lags mostly make only small contributions to the predictions. It is interesting to observe that the functional forms differ not only in their size between the lags of the same variables but also in their shape. For example, comparing the lags of 12 months and 24 months, we observe contrary directions of the functional form for both industrial production and S&P 500. Larger lags thus provide a form of correction to the main effects (first lag), explaining the somewhat better model performance. Whether this improvement in performance warrants the more complex interpretation of the resulting models depends on the practical situation at hand.

5.4 Robustness of Shapley Values

We have shown in Figure 6 that the performance of the prediction models can be quite sensitive to random seeds. Here, we investigate

[23]We also experimented with adding monthly lags (e.g., $12, 13, \ldots, 23, 24$) but this richer set of features produced inferior results.

Table 7. Performance Comparison When Using Different Lag Structures

	Lags *h* (in months and steps of 12)					
	12	**12–24**	**12–36**	**12–48**	**12–60**	**12–72**
Gradient Boosting	0.56	0.59	0.54	**0.52**	0.53	0.54
SVR	**0.57**	0.58	0.60	0.61	0.63	0.64
Forest	0.58	0.58	0.56	0.57	**0.56**	**0.54**
Neural Network	**0.59**	0.62	0.61	0.60	**0.59**	**0.59**
Lasso	0.64	0.63	0.62	0.62	**0.61**	**0.61**
Ridge	0.64	0.64	0.63	0.60	0.59	0.60
OLS	**0.65**	**0.65**	0.68	0.70	0.75	0.78

Note: Lags are shown in months and are incremented in steps of 12. For example, the lag structure 12–48 contains lags 12, 24, 36, 48 of our key features. The lag of 12 corresponds to the baseline experiment. The error metric is mean absolute error. The lag structure that leads to the lowest error for each model (row) is highlighted in bold. Target audience: analysts.

whether random seeds also affect the global and local feature attributions and with that the economic interpretation.

Figure A.7 in Appendix A presents the Shapley shares of 10 different gradient boosting models, each based on a different random seed. For each variable, there is little variance in the Shapley share between the models. The functional forms learned by the models are also rather robust. Figure A.8 in Appendix A shows the Shapley values of the four most important predictors based on the 10 gradient boosting models with different random initializations. There are only minor differences between the fitted third-degree polynomials. However, the Shapley values of single data points can differ substantially between the different model realizations. This is indicated by the vertical lines which show, for each observation, the range of Shapley values across the 10 iterations. This shows the benefits of model averaging, which leads to more stable estimates.

Computing exact Shapley values is computationally expensive. It requires testing the predictions of all possible coalitions of features (see Appendix B). The number of coalitions grows exponentially with the number of features so that, in practice and as implemented by the *kernel* approach in the SHAP Python library (Lundberg and Lee 2017), coalitions are sub-sampled to approximate the true Shapley values. When a coalition does not include a feature k, it is integrated out by using its values within a background data set (see again Appendix B). Ideally, the background set is big and represents the data set well. However, the bigger the background sample, the more expensive the computation of the Shapley values becomes. In practice, the background sample is often summarized by using a random sub-sample of the training set, or by approximating the training set with a few representative centroids using k-means clustering.

In all experiments above, we have used the kernel method with 2,000 coalitions, and 25 centroids when estimating Shapley values for all machine learning models. Here, we investigate the robustness of the Shapley values when altering these two parameters. Figure A.9 in Appendix A shows the Shapley values of industrial production,[24] the most important predictor, for our most accurate models, gradient boosting and the SVR. We order all observations by increasing

[24]The results are qualitatively similar for the other features.

Shapley values. When varying the number of coalitions (top row), the gradient boosting model is insensitive to this parameter. Sampling only 50 coalitions suffices for an accurate estimation of Shapley values. In contrast, we see some variation in Shapley values for the SVR if only 50 coalitions are sampled. There is almost no variation anymore if 100 coalitions are used.

The middle row of Figure A.9 shows the effect of varying the size of the background sample. Here for both the gradient boosting model and the SVR, we only observe some errors in the estimates for a small background sample size of five.

An alternative to the kernel-based computation of Shapley values is Tree SHAP (Lundberg et al. 2020). It is not model agnostic and can only be used on tree-based models such as gradient boosting and random forests. It does not estimate Shapley values by enumerating all possible coalitions of features but by only considering those that actually are observed in the tree models, which makes this approach computationally much cheaper by construction.[25] The bottom panel of Figure A.9 compares the Shapley values of industrial production based on the kernel method with those based on Tree SHAP for both tree-based models. Both methods produce very similar estimates of Shapley values for gradient boosting and the random forest.

Table 8 shows the computation time required to obtain Shapley values for the 359 data points of the whole test period between 1990 and 2019.[26] With the baseline parameters of 2,000 coalitions, and a background sample of 25, the computation time is about one minute for the gradient boosting model and eight minutes for the SVR. But by reducing the coalitions to 100, the computation time for both methods drops substantially. Using Tree SHAP, Shapley values are estimated within two seconds.

This analysis suggests that, while the exact estimation of Shapley values can be computationally prohibitive, they can be approximated accurately and efficiently. For instance, in our case study,

[25]We set the parameter *feature_perturbation* to *interventional.* In this way, Tree SHAP, like the kernel method, ignores dependencies between features (see Appendix B). As another robustness check, we set this parameter to *tree_path_dependent* and thus consider correlations between features. Doing this, the Shapley values of our tree models do not change markedly.

[26]We train a single model every year and do not use bagging. The computation was conducted on a single kernel (not parallelized) of an AMD Ryzen 7 3700X.

Table 8. Computation Time (in seconds) When Estimating the Shapley Values of the SVR and Gradient Boosting for the Whole Test Period (1990–2019) Containing 359 Observations

			Computation Time (Seconds)	
Method	**Background Sample**	**Coalitions**	**Gradient Boosting**	**SVR**
Kernel	25	100	16.35	76.76
Kernel (Baseline Parameters)	25	2,000	69.74	451.73
Kernel	100	2,000	292.06	1,772.17
Tree SHAP	—	—	1.70	—

Note: Target audience: analysts.

the Shapley value computation for the full test period can be reduced to the order of one minute without noteworthy differences in attribution.

6. Discussion

This paper presents a workflow for using machine learning to inform decision making in situations where transparency and ease of communicating results are key. The three steps of the workflow are a model comparison, a decomposition of predictions into feature contributions, and statistical inference on those contributions.

We applied the workflow to an economic forecasting exercise, predicting the change of the U.S. unemployment rate one year ahead for the past 30 years.

In the first step of this case study, we found a significantly better performance of machine learning models compared with linear benchmarks. In the second step, we observed pronounced nonlinearities learned by the machine learning models that have clear economic interpretations. In the third step of the workflow, we use the Shapley regression framework to find that a larger number of variables is statistically significant when using machine learning models than for the linear benchmark, which demonstrates that non-linear machine learning models can extract and uncover a richer set of robust predictive relationships in the data.

Machine learning methods are increasingly used in economic and social science research. However, most studies using machine learning focus on maximizing predictive accuracy and accept the black box nature of the models. Research that does attempt statistical inference on machine learning models often uses controlled data—for example, from randomized controls trials. Our study shows that the use of machine learning models and statistical inference can be combined to learn from real-world prediction problems. But our study also revealed challenges of machine learning modeling on rather small data. First, we showed that the performance of some of the machine learning models is rather sensitive to random seeds. Second, the machine learning models differ in how they are affected by experimental parameters such as the type of hyperparameter search (e.g., k-fold cross-validation versus blocked cross-validation) or winsorization. These challenges can be addressed by model averaging to increase robustness, and by rigorous robustness checks, respectively. At the same time, our diverse set of machine learning methods, which differ significantly in their predictive performance, all learned highly similar functional forms from the data.

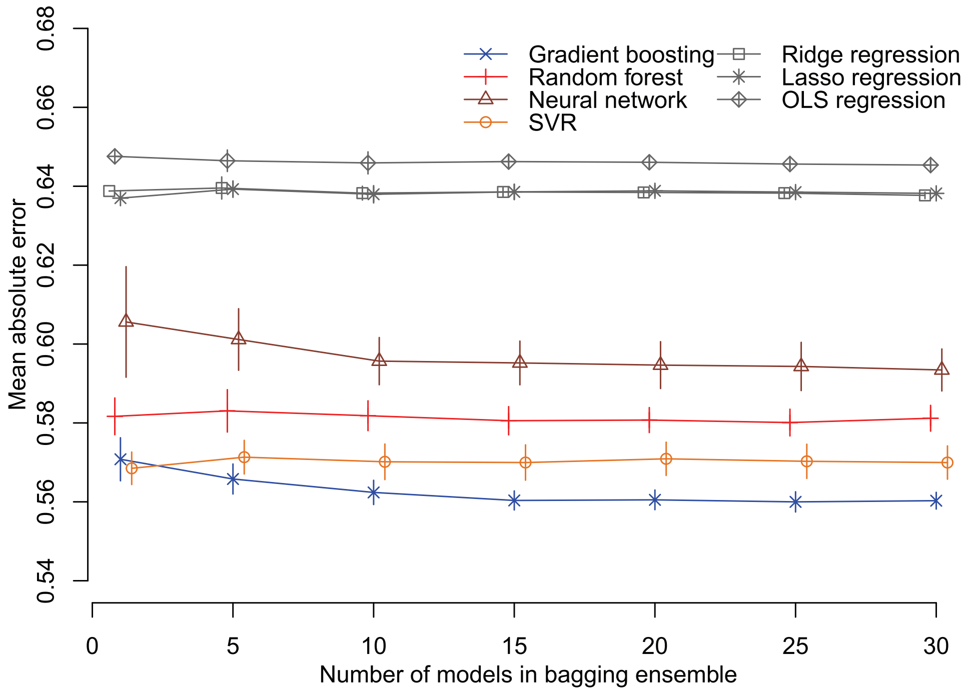
When using a broad set of predictive variables instead of a small hand-picked selection, the predictive performance increased slightly. But as the non-linear machine learning models do not learn a sparse model but use most features for prediction, interpreting their predictions becomes considerably more challenging. Further, we showed that the functional forms learned were less consistent across model families. When we trained the models on PCA components, the interpretability was reduced as well, because the components do not have a clear interpretation. At the same time, being trained on just a few components limits the differential functional forms that the models can learn from each underlying feature. Finally, the Shapley regression will suffer from a reduction of statistical power when being calibrated on a large number of features. Collectively, these considerations suggest that, while our workflow is general and works in principle for problems with a large number of features, it will deliver more robust and easier to interpret results when based on a smaller set of features.

More generally, our case study is reminiscent of many real-world settings where one has a considerable number of features to learn from but a limited number of observations. Our results suggest that a combination of expert domain knowledge to select key indicators and robust model properties may lead to the best trade-off between model performance and the interpretability of results—if such a trade-off exists in the first place.

Machine learning methods may be unfamiliar to many decision-makers, but their predictive accuracy and ability to detect richer, more nuanced signals in data justify their use to inform decisions. While the Shapley regression framework cannot *fully* communicate complex and nonlinear models with a single statistic, like the Shapley share coefficient, it allows statistical inference on nonlinear outputs and can be communicated similarly to well-understood regression results.

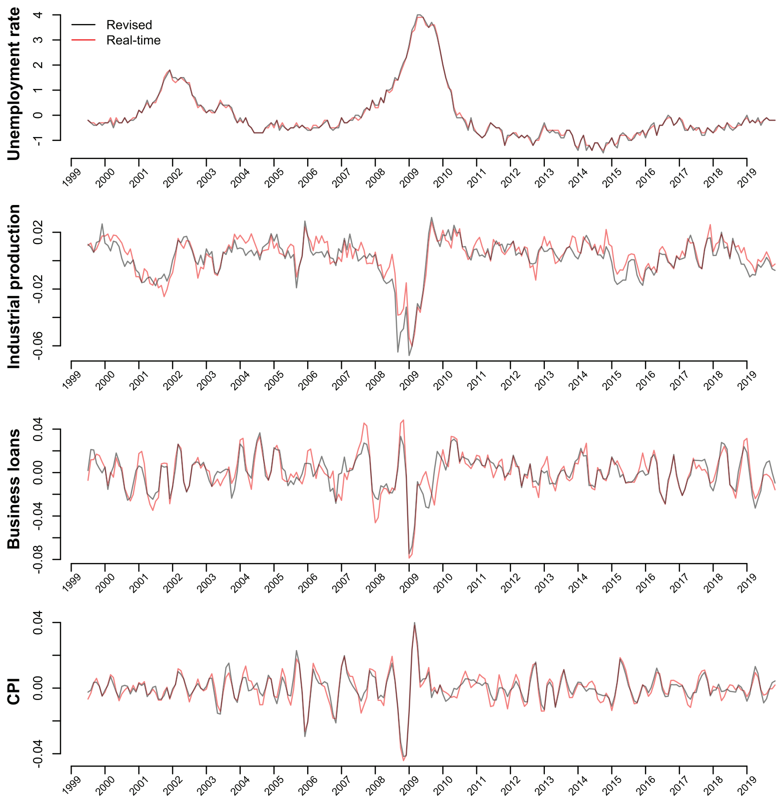
Appendix A. Results

Figure A.1. Forecasting Performance across 10 Iterations as a Function of the Number of Models in the Bagging Ensenble



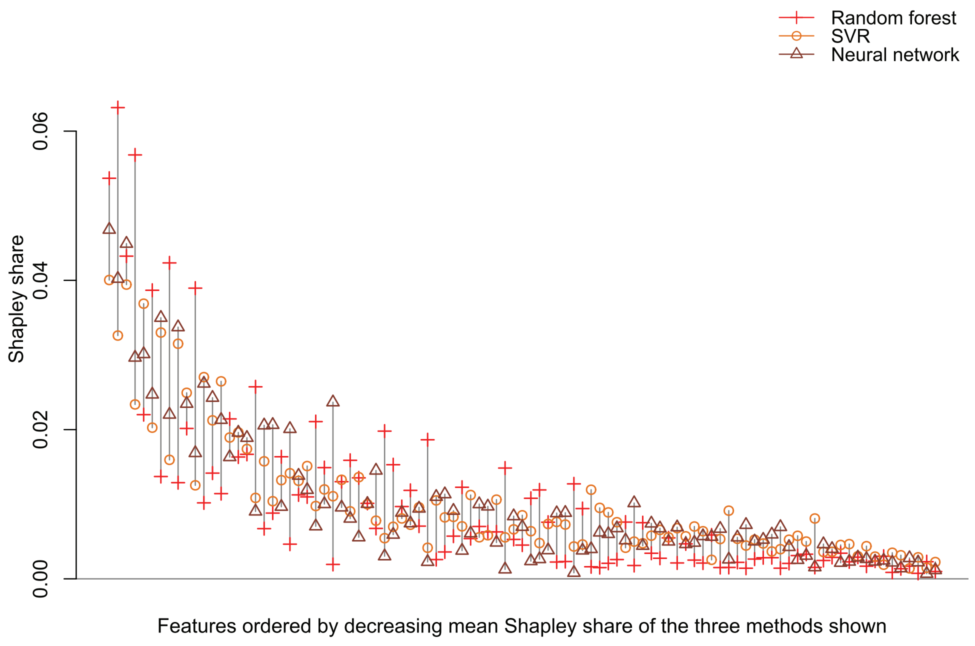
Note: The horizontal lines show the mean performance across all 10 iterations; the vertical bars show ± two standard errors around that mean. Target audience: analysts.

Figure A.2. Comparison of Real-Time and Revised Series after Transformations that Make Them Stationary (see Table 1)



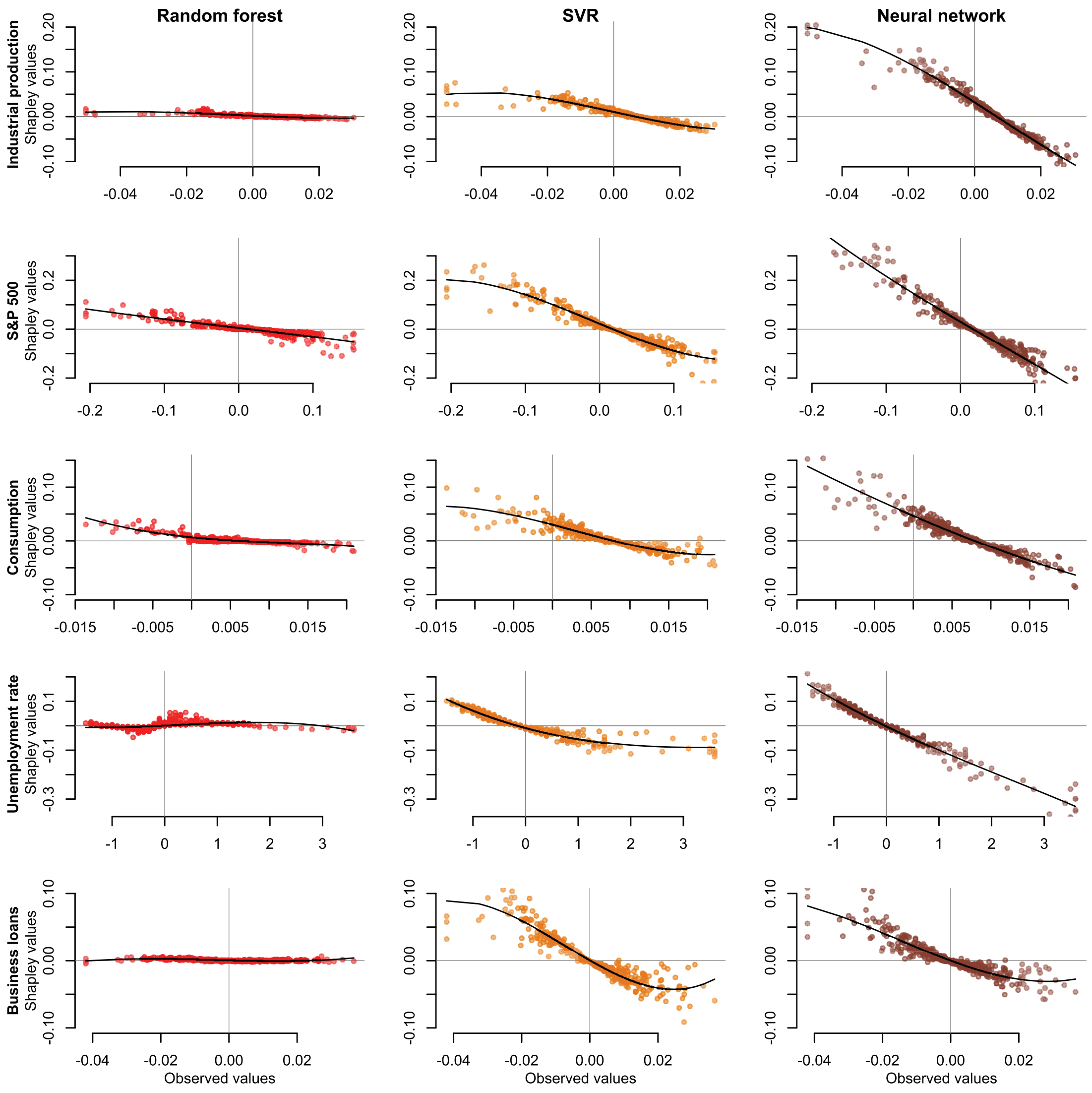
Note: Target audience: analysts.

Figure A.3. Shapley Share When Machine Learning Models Are Calibrated on All 97 Features of the Data Set



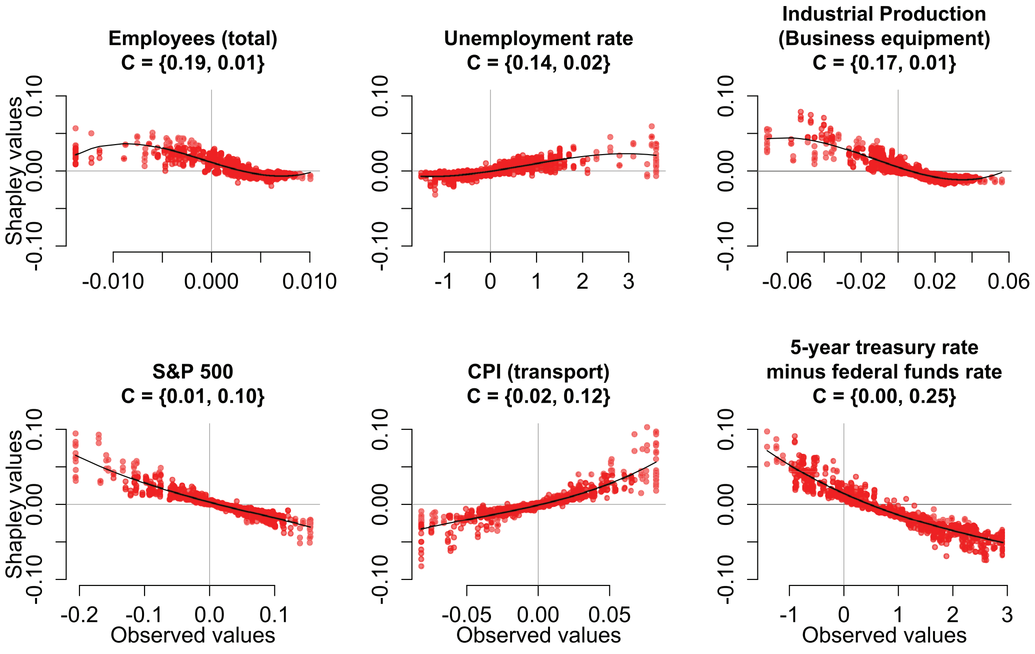
Note: The three shown models perform best in prediction when calibrated on all features. Target audience: analysts.

Figure A.4. Selected Learned Functional Forms for the Three Best-Performing Models When Using All 97 Features



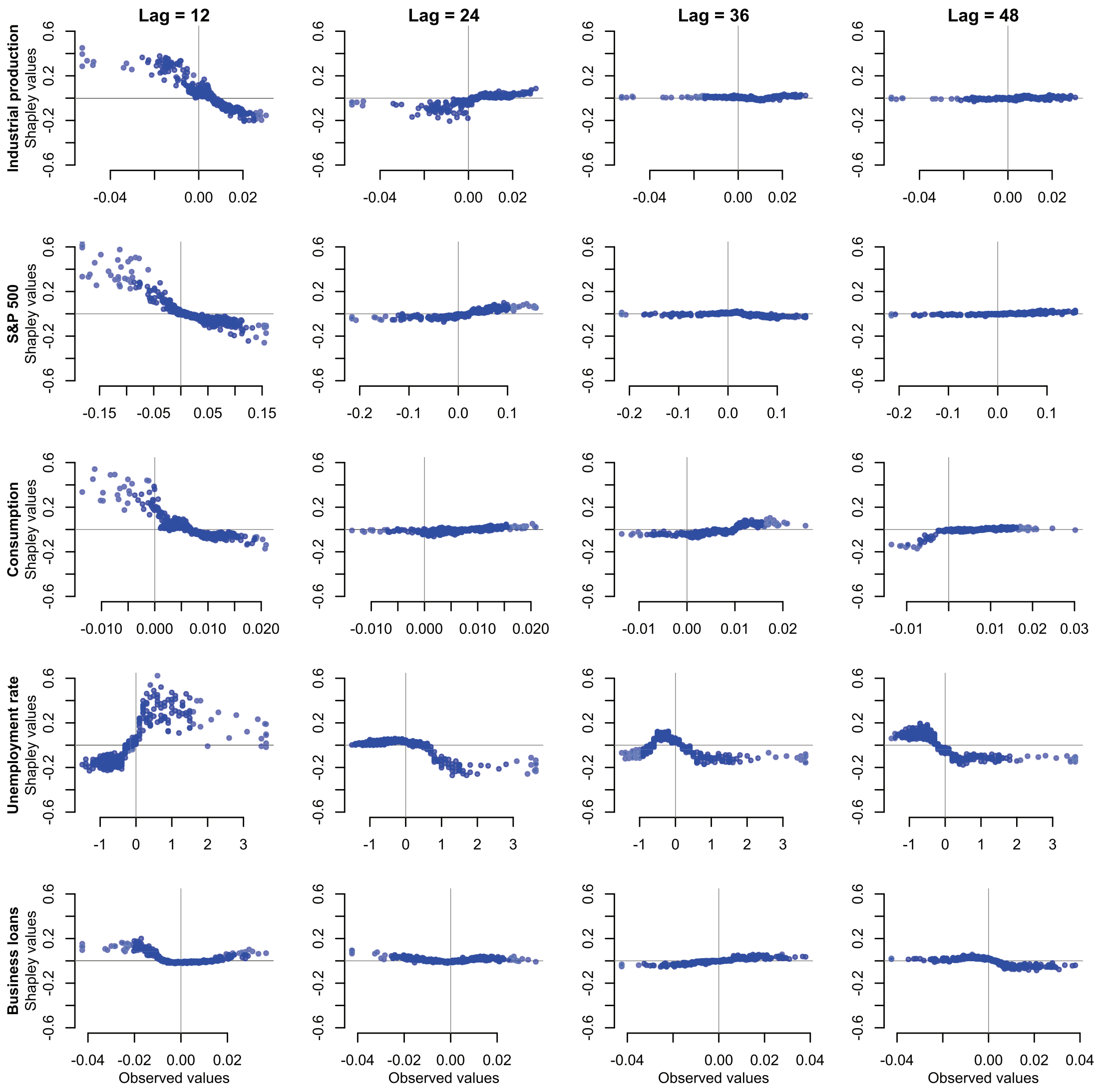
Note: The lines show best-fit third-degree polynomials. Note that the scale of the vertical axis differs between rows. Target audience: analysts.

Figure A.5. Learned Functional Forms of Selected Features Based on the Predictions of a Random Forest Trained on the Two First Components of a PCA



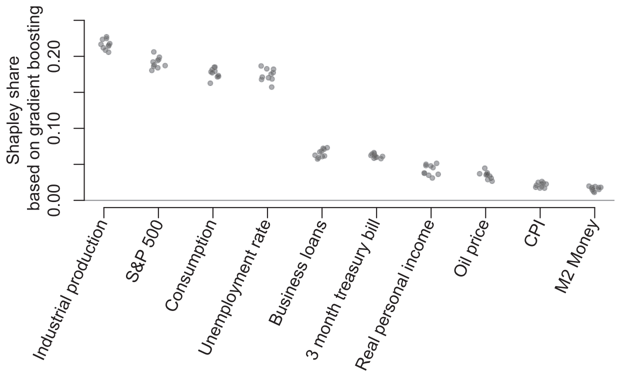
Note: The term C shows the loadings of the features on the first and second principal component. The first row shows features with a high loading on the first principal component and low loadings on the second component. The second row shows features with a high loading on the second principal component and low loadings on the first component. The lines show best-fit third-degree polynomials. Target audience: analysts.

Figure A.6. Learned Functional Forms of Key Predictors at Different Lags as Learned by the Gradient Boosting Model



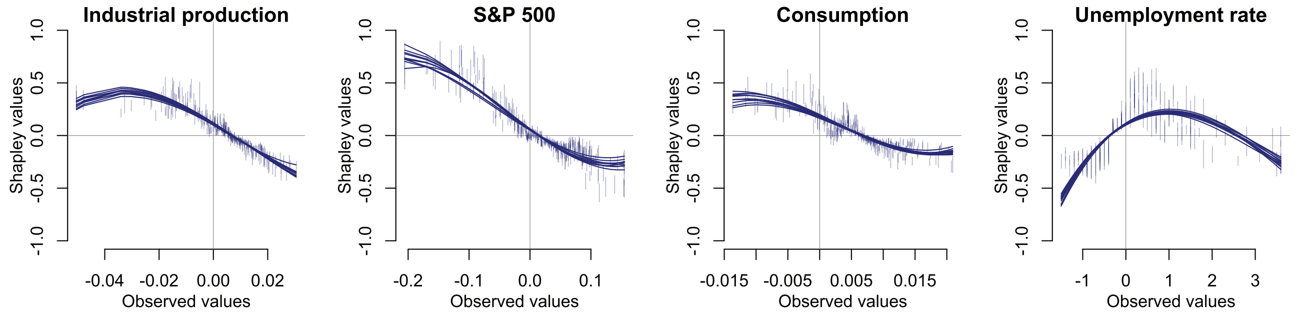
Note: The results shown are based on a model trained on 40 features: four lags (12, 24, 36, 48 months) for each of the 10 key features. Target audience: analysts.

Figure A.7. Robustness of the Shapley Share



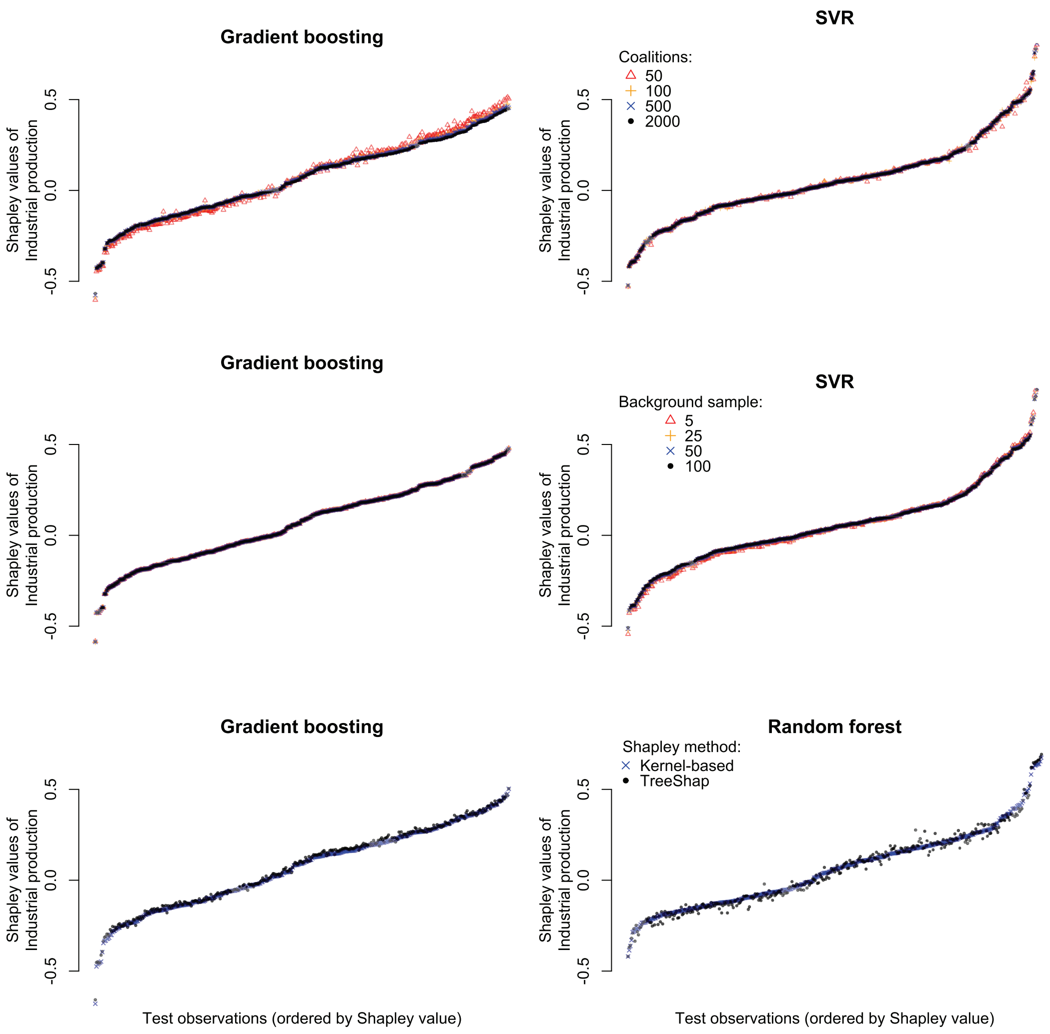
Note: The figure shows the Shapley shares according to 10 different gradient boosting models, each trained with a different random seed. Target audience: analysts.

Figure A.8. Robustness of Individual Shapley Values



Note: Each line shows the functional form learned by one of 10 gradient boosting models, each trained with a different random seed. The vertical lines show the maximum range in Shapley values across the 10 iterations for each observation. Target audience: analysts.

Figure A.9. Accuracy of Shapley Value Computations



Note: The top row compares Shapley values estimated by the kernel method for different coalition sizes. The middle row shows the Shapley values for different background sample sizes. The bottom row compares Shapley values estimated by the kernel-based method and the Tree SHAP method for the two tree-based methods. From the top to bottom row, observations are ordered by increasing Shapley values of the largest number of coalitions, the largest background, sample, and the kernel-based method, respectively. Target audience: analysts.

Appendix B. Feature Importance Measures

We present a concise description of the computation of the two feature importance measure used in this paper, permutation importance and Shapley values. We also discuss computational and conceptual challenges.

B.1 Permutation Importance

The permutation importance of a variable measures the change of model performance when the values of that variables are randomly scrambled, i.e., permuted. If a model has learnt a strong dependency between the model outcome and a given variable, scrambling the value of the variable leads to very different model predictions and thus affects performance. A variable k is said to be important in a model if the error $\mathcal{P}$ after scrambling feature k is substantially higher than when using the original values for k, i.e., $\mathcal{P}_k^{perm} >> \mathcal{P}_k^{baseline}$. The value of the permutation error $\mathcal{P}_k^{perm}$ depends on the realization of the permutation, and variation in its value can be large, particularly in small data sets. Therefore, it is recommended to average $\mathcal{P}_k^{perm}$ over several random draws for more accurate estimation and to assess sampling variability. Note that it is intractable in most applications to evaluate all $M!$ permutations in a test set of size M. However, the average of multiple realizations of $\mathcal{P}_k^{perm}$ will mostly converge quickly with the number of permutation, making permutation importance a computationally cheap way to assess feature importance.[27]

The following procedure estimates the permutation importance.

(i) For each feature x_k:
   a. Generate a permutation sample x_k^{perm} with the values of x_k permuted across observations.

   b. Re-evaluate the test performance for x_k^{perm}, resulting in $\mathcal{P}_k^{perm}$.

[27]Given a large test set, bootstrap sub-samples may suffice.

c. The permutation importance of x_k is given by $I(x_k) = \mathcal{P}_k^{perm}/\mathcal{P}^k_{baseline}$. Alternatively, the difference $\mathcal{P}_k^{perm} - \mathcal{P}_k^{baseline}$ can be considered.

d. Repeat and average over Q iterations and average $\bar{I}_k = 1/Q \sum_q I(x_k)$.

(ii) If I_k is based on the ratio of errors $\mathcal{P}_k^{perm}/\mathcal{P}_k^{baseline}$, consider the normalized quantity $\bar{I}_k = (I_k - 1)/\sum_k (I_k - 1) \in (0, 1)$.[28]

This ease of use comes at some cost. For example, if two features contain similar information, permuting either of them will not reflect the actual importance of this feature relative to all other features. Only permuting both or excluding one would do so. This motivates our comparison with Shapley values because they identify the individual marginal effect of a feature, accounting for its interaction with all other features. More generally, permutation importance does not come with the same analytical guarantees as Shapley values. Additionally, the computation of permutation importance requires access to true outcome target values to evaluate performance. In many situations, e.g., when working with models trained on sensitive or confidential data, these may not be available.

B.2 Shapley Values

Shapley values originate from game theory as a general solution to the problem of attributing a joint pay-off obtained in a cooperative game to the individual players of a coalition based on their contribution to the game (Shapley 1953). Štrumbelj and Kononenko (2010) introduced the analogy between players in a cooperative game and variables in a general supervised model. In the latter, variables jointly generate a prediction, the pay-off.

The Shapley values of a model offer a local decomposition of each model prediction[29] x_i of the form given in Equation (1), $f(x_i) = \sum_{k=0}^{N} \phi_k(x_i)$. Here $\phi_k^S(x_i)$ is the Shapley value associated

[28]Note, $I_k \geq 1$ in general. If not, there may be problems with model optimization.

[29]We label observations by $i \in \{, \ldots, M\}$ here, which is more general than the time-series notation used in Section 3.

with predictor k and ϕ_0^S an intercept, usually the mean prediction of the model. Shapley values come with a host of appealing analytical properties which are inherited from their game-theoretic origins. Moreover, the decomposition in Equation (1) does not need to refer to single variables but can also include interactions or even higher-order terms of interest as introduced by Agarwal, Dhamdhere, and Sundararajan (2019).

The Shapley attribution $\phi_k^S(x_i; f)$ for variable k in observation x_i and model f in Equation (1) is given by

$$\phi_k^S(x_i; f) = \sum_{x' \subseteq \mathcal{C}(x) \setminus \{k\}} \frac{|x'|!(N - |x'| - 1)!}{N!} \big[f(x_i | x' \cup \{k\}) - f(x_i | x') \big], \tag{B.1}$$

where $\mathcal{C}(x) \setminus \{k\}$ is the set of all possible variable combinations (coalitions) when excluding variable k and $|x'|$ denotes the number of variables included in that coalition. Equation (B.1) is the weighted sum of marginal contributions of variable k to all possible variable coalitions not including k itself. For example, assuming we have three variables (players) $\{A, B, C\}$, the Shapley value of player C would be $\phi_C^S(f) = 1/3[f(\{A, B, C\}) - f(\{A, B\})] + 1/6[f(\{A, C\}) - f(\{A\})] + 1/6[f(\{B, C\}) - f(\{B\})] + 1/3[f(\{C\}) - f(\{\emptyset\})]$.

There are challenges in evaluating Equation (B.1), which may be called the no-free-lunch theorem of Shapley values. First, the number of coalitions x' to evaluate grows exponentially with the number of variables. This means that an exhaustive enumeration is not feasible for already moderate variable sets. Second, we need to evaluate conditional model predictions of the form $f(x_i | x')$, i.e., models where only variables in x' are "active," and information from the other variables is excluded.[30] Under the assumption of feature independence, $f(x_i | x')$ can be evaluated by conditional expectations over an informative background data set b. That is, non-active variables are integrating out numerically using b. Let $\omega_{x'} \equiv |x'|!(N - |x'| - 1)!/N!$

[30]This does not mean a different model which only uses variables in x'. Such a model would need to be retrained and would generate different predictions, i.e., not be the model we want to evaluate.

be the combinatorial weighting factor and $\bar{x'}$ the set of variables among all not included in x'; then Equation (B.1) can be written as

$$\phi_k^S(x_i; f) = \sum_{x' \subseteq \mathcal{C}(x) \setminus \{k\}} \omega_{x'} \left[\mathbb{E}_b[f(x_i)|x' \cup \{k\}] - \mathbb{E}_b[f(x_i)|x'] \right], \tag{B.2}$$

$$\text{with} \qquad \mathbb{E}_b[f(x_i)|x'] \equiv \int f(x_i)\, \mathrm{d}b(\bar{x'}) = \frac{1}{|b|} \sum_b f(x_i|\bar{x'}). \tag{B.3}$$

The intercept ϕ_0^S is determined by the background data set b motivating its name,

$$\phi_0^S = \mathbb{E}_b[f(\emptyset)] = \mathbb{E}_b[f(\bar{x'} = b)]. \tag{B.4}$$

This means that the components ϕ_k^S measure variable contributions relative to the mean model prediction in b and that ϕ_0^S is a reference point. The choice of b will also affect the interpretation of Shapley components. That is why b should be informative, e.g., as being representative of the whole data-generating process, or to represent a sub-group of a population, like the group of untreated in an experimental setting.

We have not yet discussed the problem of computational complexity and the case when features are not independent. These are briefly discussed with further references to the literature.

- *Computational Complexity:* The sum over variable coalitions becomes impractical or even infeasible for already relatively small sets of variables of about 8–10 depending on the application. For larger sets of variables some form of coalition sampling or variable selection is needed. The Kernel SHAP algorithm Lundberg and Lee (2017) provides an efficient sampling and evaluation of Shapley contributions in the form of a weighted least square calculation. We have shown in Section 5.4 that only 100 coalitions suffice to accurately estimate Shapley values.

  A variable selection method which provides an exact computation for a subset of features is presented in Joseph (2019).

Often one is not interested in the contributions of all variables of a model, but only a subset. In this case variables *not* of interest can be grouped into a single *other* component, or sub-groups may serve as *super-variables* until an exhaustive evaluation of Equation (B.1) is possible. We have used this approach in Section 4.2, when computing the Shapley values of an interaction of features.

Additionally, the computation of Shapley components can be reduced by limiting the size of the background b. A default option is the whole training data set representing all information the model has learned from. However, this can be impractical for large data sets. Here either sub-samples or summary points, e.g., cluster centroids, can be used. We have shown in Section 5.4 that even small background samples of 25 observations suffice to accurately estimate Shapley values.

- *Feature Dependence:* The evaluation of conditional expectations (Equation (B.3)) does not consider dependencies between features, which can lead to feature value combinations that are non-sensical and would not occur in the real world. We discuss three ways to address this. First, one can estimate Shapley values of tree-based models for which there exists an efficient algorithm that accounts for feature dependence (Lundberg et al. 2020). By comparing Shapley values when respecting or ignoring feature dependence, one can gauge the importance of feature dependencies. However, caution is warranted when transferring the findings to other model families, e.g., artificial neural networks. These may have learned different feature attributions for which the comparison of Shapley components evaluated under the independence assumption is indicative.

  Second, one can net out the effect of higher-order feature interactions using the Shapley-Taylor index (Agarwal, Dhamdhere, and Sundararajan 2019) to understand dependencies between features. This is an expansion of a function over sets of active features including interactions of any order of interest analogous to the Taylor expansion of differentiable functions. This means that the importance of a feature is either its main effect or the main effect in addition to different interaction terms. Interaction terms also provide additional information, as shown in the analysis presented in the main text.

Third, the dependence structure within a variable set can be estimated (Aas, Jullerm, and Lølund 2021). While adding an extra potentially computationally costly step, this has the advantage of providing a single attribution for each feature which accounts for the dependence of the data at least approximately. It may, however, be argued that it is not necessarily clear what a single component in a non-linear model with feature dependence captures. This is more intuitive for Shapley contributions of feature interactions, which capture the effect of co-movements of two or more variables.

- *Expectation Consistency:* As shown by Sundararajan and Najmi (2020), attribution consistency which, casually put, avoids logical contradictions in feature attribution, can be violated when using conditional expectations for the computation of Equation (B.3),[31] and a single reference value is advocated for. However, this discards much of the potentially rich information a model has learned, such as non-linearities. A solution to this is provided in Joseph (2019) by an additional condition when comparing different models against a common background. The models' intercepts ϕ_0^S over the background data b need to coincide. This is fulfilled in many practical situations where models optimize the same objective functions, like the mean squared error. Variations in ϕ_0^S are of concern as soon as they reach the order of magnitude of the Shapley components ϕ_k^S.

None of the above challenges is fatal for the application of Shapley values for model interpretability, as we have shown in detail. However, one has to be aware of the possible pitfalls and consequences of approximations for model interpretations and any decisions based on them.

[31]See also Janzing, Minorics, and Blöbaum (2020).

Table B.1. Implementation Details on the Prediction Models

Method	**Implementation**	**Fixed Parameters**	**Hyperparameter Space**
Random Forest	`scikit-learn` RandomForestRegressor	n_estimators: 500 criterion: mae	max_depth: [2, 3, 4, 5, 6, 8, 10, 20, 30]* max_features: [1, 3, 5, 7, 9, 11, 15, 30, 50]*
Gradient Boosting	`lightgbm` LGBMRegressor		subsample: [0.05, .1, .2, .3, .4, .6, .6, .7, .8, .9, 1] reg_lambda: [0, 0.1, 1, 10, 20, 50, 100] reg_alpha: [0, 0.1, 1, 2, 7, 10, 50, 100] num_leaves: [2, 3, 4, 5, 10, 20, 40, 70, 100] n_estimators: [1, 3, 5, 10, 20, 30, 40, 50, 75, 100]* max_depth: [1, 2, 3, 5, 8, 15]* colsample_bytree: [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 1]
SVR	`scikit-learn` SVR		C: $[2^1, 2^{1+1\frac{4}{9}}, 2^{1+2\frac{4}{9}}, \dots, 2^{1+9\frac{4}{9}}]$ gamma: $[2^{-7}, 2^{-7+1\frac{6}{9}}, 2^{-7+2\frac{6}{9}}, \dots, 2^{-7+9\frac{6}{9}}]$ epsilon: [0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5]*
Neural Network	`scikit-learn` MLPRegressor	solver: lbfgs max_iter: 2000	hidden_layer_sizes: [5, (5, 5), (5, 5, 5), 10, (10, 10), ..., (10, 10, 10), 15, (15, 15), ..., 25, (25, 25), (25, 25, 25)] activation: [relu, tanh] alpha: $[10^{-5}, 10^{-4}, \dots, 10^{3}]$
Lasso Regression	`scikit-learn` Lasso		alpha: $[10^{-5}, 10^{-5+1\frac{9}{99}}, 10^{-5+2\frac{9}{99}}, \dots, 10^{-5+99\frac{9}{99}}]$
Ridge Regression	`scikit-learn` Ridge		alpha: $[10^{-5}, 10^{-5+1\frac{9}{99}}, 10^{-5+2\frac{9}{99}}, \dots, 10^{-5+99\frac{9}{99}}]$
OLS Regression	`scikit-learn` LinearRegression		

*We initially used a large parameter space but have refined it to these values without sacrificing performance.
Note: The second column shows the Python package and the respective name of the machine learning method. The third column shows parameters that we set to a different value than the default. The fourth column shows the hyperparameter space.

References

Aas, K., M. Jullum, and A. Løland. 2021. “Explaining Individual Predictions when Features are Dependent: More Accurate Approximations to Shapley Values.” *Artificial Intelligence* 298 (September): Article 103502.

Agarwal, A., K. Dhamdhere, and M. Sundararajan. 2019. “A New Interaction Index Inspired by the Taylor Series.” arXiv: 1902.05622.

Armstrong, S., K. Green, and A. Graefe. 2015. “Golden Rule of Forecasting: Be Conservative.” *Journal of Business Research* 68 (8): 1717–31.

Bergmeir, C., and J. Benítez. 2012. “On the Use of Cross-validation for Time Series Predictor Evaluation.” *Information Sciences* 191 (May 15): 192–213.

Bianchi, D., M. Büchner, and A. Tamoni. 2019. “Bond Risk Premia with Machine Learning.” USC-INET Research Paper No. 19-11.

Bluwstein, K., M. Buckmann, A. Joseph, M. Kang, S. Kapadia, and O. Simsek. 2020. “Credit Growth, the Yield Curve and Financial Crisis Prediction: Evidence from a Machine Learning Approach.” Staff Working Paper No. 848, Bank of England.

Breiman, L. 1996. “Bagging Predictors.” *Machine Learning* 24 (2): 123–40.

———. 2001a. “Random Forests.” *Machine Learning* 45 (1): 5–32.

———. 2001b. “Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author).” *Statistical Science* 16 (3): 199–231.

Burgess, S., E. Fernandez-Corugedo, C. Groth, R. Harrison, F. Monti, K. Theodoridis, and M. Waldron. 2013. “The Bank of England’s Forecasting Platform: COMPASS, MAPS, EASE and the Suite of Models.” Staff Working Paper No. 471, Bank of England.

Chakraborty, C., and A. Joseph. 2017. “Machine Learning at Central Banks.” Staff Working Paper No. 674, Bank of England.

Chen, J., A. Dunn, K. Hood, A. Driessen, and A. Batch. 2022. “Off to the Races: A Comparison of Machine Learning and Alternative Data for Predicting Economic Indicators.” In *Big Data for 21st Century Economic Statistics,* ed. K. G. Abraham, R. S. Jarmin, B. Moyer, and M. D. Shapiro. University of Chicago Press.

Chernozhukov, V., M. Demirer, E. Duo, and I. Fernandez-Val. 2018. "Generic Machine Learning Inference on Heterogenous Treatment Effects in Randomized Experiments." NBER Working Paper No. 24678.

Chetverikov, D., Z. Liao, and V. Chernozhukov. 2020. "On Cross-validated Lasso in High Dimensions." arXiv: 1605.02214.

Coulombe, P. G., M. Leroux, D. Stevanovic, and S. Surprenant. 2020. "How is Machine Learning Useful for Macroeconomic Forecasting?" arXiv: 2008.12477.

Cybenko, G. 1989. "Approximation by Superpositions of a Sigmoidal Function." *Mathematics of Control, Signals, and Systems* 2: 303–14.

D'Amour, A., K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen et al. 2020. "Underspecification Presents Challenges for Credibility in Modern Machine Learning." arXiv: 2011.03395.

Doerr, S., L. Gambacorta, and J. M. S. G. Garralda. 2021. "Big Data and Machine Learning in Central Banking." BIS Working Paper No. 930.

Döpke, J., U. Fritsche, and C. Pierdzioch. 2017. "Predicting Recessions with Boosted Regression Trees." *International Journal of Forecasting* 33 (4): 745–59.

Doshi-Velez, F., and B. Kim. 2017. "Towards a Rigorous Science of Interpretable Machine Learning." arXiv: 1702.08608.

Drucker, H., C. Burges, L. Kaufman, A. Smola, and V. Vapnik. 1997. "Support Vector Regression Machines." In *Advances in Neural Information Processing Systems*, Vol. 9, ed. M. C. Mozer, M. Jordan, and T. Petsche, 155–61. Cambridge, MA: MIT Press.

Einhorn, H. J., and R. M. Hogarth. 1985. "Prediction, Diagnosis, and Causal Thinking in Forecasting." In *Behavioral Decision Making*, ed. G. Wright, 311–28. Springer.

Elliott, G., and A. Timmermann. 2008. "Economic Forecasting." *Journal of Economic Literature* 46 (1): 3–56.

Federal Reserve Bank of St. Louis. 2020. "NBER Based Recession Indicators for the United States from the Period Following the Peak through the Trough [USREC]."

Fernández-Delgado, M., E. Cernadas, S. Barro, and D. Amorim. 2014. "Do We Need Hundreds of Classifiers to Solve Real World

Classification Problems?" *Journal of Machine Learning Research* 15 (1): 3133–81.

Fisher, A., C. Rudin, and F. Dominici. 2019. "All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously." *Journal of Machine Learning Research* 20 (177): 1–81.

Friedman, J. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics* 29 (5): 1189–1232.

Friedman, J., T. Hastie, and R. Tibshirani. 2009. *The Elements of Statistical Learning.* Springer Series in Statistics. Berlin: Springer.

George, E. 1999. "Economic Models at the Bank of England." Technical Report, Bank of England.

Giannone, D., M. Lenza, and G. E. Primiceri. 2021. "Economic Predictions with Big Data: The Illusion of Sparsity." *Econometrica* 89 (5): 2409–37.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning.* MIT Press.

Haldane, A. 2018. "Will Big Data Keep Its Promise?" Speech at the Bank of England Data Analytics for Finance and Macro Research Centre, King's Business School, London, April 19.

Independent Evaluation Office. 2015. "Evaluating Forecast Performance Report." Technical Report, Bank of England.

Janzing, D., L. Minorics, and P. Blöbaum. 2020. "Feature Relevance Quantification in Explainable AI: A Causal Problem." In *Proceedings of Machine Learning Research*, Vol. 108, ed. S. Chiappa and R. Calandra, 2907–16. Proceedings of the International Conference on Artificial Intelligence and Statistics, August 26–28.

Joseph, A. 2019. "Parametric Inference with Universal Function Approximators." arXiv: 1903.04209.

Kazemitabar, J., A. Amini, A. Bloniarz, and A. S. Talwalkar. 2017. "Variable Importance Using Decision Trees." In *Advances in Neural Information Processing Systems,* Vol. 30, ed. I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 426–35. Red Hook, NY: Curran Associates, Inc.

Kim, H. H., and N. R. Swanson. 2018. "Mining Big Data Using Parsimonious Factor, Machine Learning, Variable Selection and

Shrinkage Methods." *International Journal of Forecasting* 34 (2): 339–54.

Kock, A. B., and T. Teräsvirta. 2014. "Forecasting Performances of Three Automated Modelling Techniques during the Economic Crisis 2007–2009." *International Journal of Forecasting* 30 (3): 616–31.

Lemaire, V., R. Féraud, and N. Voisine. 2008. "Contact Personalization Using a Score Understanding Method." In *2008 IEEE International Joint Conference on Neural Networks*, 649–54.

Lipovetsky, S., and M. Conklin. 2001. "Analysis of Regression in Game Theory Approach." *Applied Stochastic Models in Business and Industry* 17 (4): 319–30.

Lipton, Z. C. 2016. "The Mythos of Model Interpretability." arXiv e-prints, Vol. 1606.03490.

Ludvigson, S., and S. Ng. 2009. "A Factor Analysis of Bond Risk Premia." NBER Working Paper No. 15188.

Lundberg, S., G. Erion, H. Chen, A. DeGrave, J. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. 2020. "From Local Explanations to Global Understanding with Explainable AI for Trees." *Nature Machine Intelligence* 2 (1): 56–67.

Lundberg, S., and S.-I. Lee. 2017. "A Unified Approach to Interpreting Model Predictions." In *Advances in Neural Information Processing Systems,* Vol. 30, ed. I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 4765–74. Red Hook, NY: Curran Associates, Inc.

Makridakis, S., E. Spiliotis, and V. Assimakopoulos. 2018a. "The m4 Competition: Results, Findings, Conclusion and Way Forward." *International Journal of Forecasting* 34 (4): 802–8.

———. 2018b. "Statistical and Machine Learning Forecasting Methods: Concerns and Ways Forward." *PLOS ONE* 13 (3).

McCracken, M. W., and S. Ng. 2016. "FRED-MD: A Monthly Database for Macroeconomic Research." *Journal of Business and Economic Statistics* 34 (4): 574–89.

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

Miller, T. 2019. "Explanation in Artificial Intelligence: Insights from the Social Sciences." *Artificial Intelligence* 267 (February): 1–38.

Ng, S., and J. Wright. 2013. "Facts and Challenges from the Great Recession for Forecasting and Macroeconomic Modeling." *Journal of Economic Literature* 51 (4): 1120–54.

Pagan, A. 1984. "Econometric Issues in the Analysis of Regressions with Generated Regressors." *International Economic Review* 25 (1): 221–47.

Parmezan, A. R. Sabino, V. Souza, and G. Batista. 2019. "Evaluation of Statistical and Machine Learning Models for Time Series Prediction: Identifying the State-of-the-Art and the Best Conditions for the Use of Each Model." *Information Sciences* 484 (May): 302–37.

Racine, J. 2000. "Consistent Cross-validatory Model-selection for Dependent Data: *hv*-block Cross-validation." *Journal of Econometrics* 99 (1): 39-61.

Ribeiro, M., S. Singh, and C. Guestrin. 2016. "Why Should I Trust You?: Explaining the Predictions of Any Classifier." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Vol. 22, 1135–44. New York, NY: Association for Computing Machinery.

Robnik-Šikonja, M., and I. Kononenko. 2008. "Explaining Classifications for Individual Instances." *IEEE Transactions on Knowledge and Data Engineering* 20 (5): 589–600.

Runkle, D. 1998. "Revisionist History: How Data Revisions Distort Economic Policy Research." *Quarterly Review* (Federal Reserve Bank of Minneapolis) 22 (4): 3–12.

Sermpinis, G., C. Stasinakis, K. Theofilatos, and A. Karathanasopoulos. 2014. "Inflation and Unemployment Forecasting with Genetic Support Vector Regression." *Journal of Forecasting* 33 (6): 471–87.

Shapley, L. 1953. "A Value for n-person Games." *Contributions to the Theory of Games*, Vol. 2, ed. H. W. Kuhn and A. W. Tucker, 307–17. Princeton University Press.

Shrikumar, A., P. Greenside, and K. Anshul. 2017. "Learning Important Features through Propagating Activation Differences." arXiv: 1704.02685.

Snijders, T. 1988. "On Cross-validation for Predictor Evaluation in Time Series." In *On Model Uncertainty and Its Statistical Implications*, ed. T. Dijkstra, 56–69. Springer.

Stock, J., and M. Watson. 2002. "Forecasting Using Principal Components from a Large Number of Predictors." *Journal of the American Statistical Association* 97 (460): 1167–79.

Stone, C. 1982. "Optimal Global Rates of Convergence for Nonparametric Regression." *Annals of Statistics* 10 (4): 1040–53.

Štrumbelj, E., and I. Kononenko. 2010. "An Efficient Explanation of Individual Classifications Using Game Theory." *Journal of Machine Learning Research* 11 (March): 1–18.

Sundararajan, M., and A. Najmi. 2020. "The Many Shapley Values for Model Explanation." In *Proceedings of Machine Learning Research*, Vol. 119, ed. H. Daumé and A. Singh, 9269–68. Proceedings of the International Conference on Artificial Intelligence and Statistics, July 13–18.

Wang, M., and C. Manning. 2013. "Effect of Non-linear Deep Architecture in Sequence Labeling." In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 1285–91. Asian Federation of Natural Language Processing.

Young, P. 1985. "Monotonic Solutions of Cooperative Games." *International Journal of Game Theory* 14 (2): 65–72.