J.P.Morgan

Economic Research
October 30, 2018

Special Report

Machine learning for macro: What you need to know

- **We cut through the hype and identify machine learning methods that macroeconomic forecasters should have in their toolbox**
- **We conduct an extensive horserace across a variety of econometric and machine learning techniques to forecast several important macroeconomic time series at multiple horizons**
- **Linear dimension reduction methods like principal components analysis and regularization methods like ridge regression perform the best in the small datasets available to macroeconomists**
- **Other machine learning and artificial intelligence methods like random forests, boosted trees, support vector machines, and neural nets do no better and sometimes much worse**
- **The more flexible ML and AI techniques would have performed particularly badly around the financial crisis when encountering conditions they had never seen before**
- **Linear methods were more successful in extrapolating from past experience in these unprecedented periods**
- **Linear variable selection methods like the lasso and elastic net also work well, but have the drawback of being unhelpful in interpreting excluded variables**
- **In truth, even the best-performing models perform poorly in absolute terms, highlighting the importance of understanding risks around the baseline forecast**
- **We conclude with practical recommendations for real-world forecasters. Tools are available to build forecasting models that:**
    - **Use ALL the data**
    - **Update in real time**
    - **Predict probabilities as well as averages**
- **These methods underlie the nowcasters, recession risk models, and other tools that we publish in the Real-time Quant Econon Monitor**
- **We show how to build models like this in Eviews and Python, and we discuss the pros and cons of these and related packages**

Table of Contents

New tools for the toolbox ……....1

A horserace: the track…..………3

A horserace: the field…………...5

Linear the winner ……..………...7

What's worth doing…………….11

What's next…………………..…14

Jesse Edgerton
(1-212) 834-9543
jesse.edgerton@jpmchase.com

Dan Weitzenfeld
(1-646) 343-4033
dan.weitzenfeld@jpmorgan.com

www.jpmorganmarkets.com

New tools for the toolbox

In the last several years, artificial intelligence has made headlines by learning to drive cars, answer phone calls, and beat humans in chess, go, and Jeopardy. Meanwhile, those of us who have spent years building forecasting models based on linear regressions have wondered if our jobs might be the next ones to be replaced by new technologies.

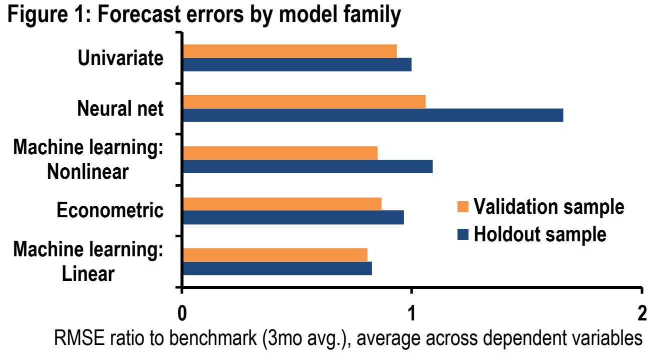
To be fair, macroeconomic forecasters have a difficult job. We make predictions about economy-wide variables like growth, unemployment, and inflation that are ultimately driven by the decisions of millions of unpredictable human beings. We are confronted with thousands of potential data sources that we could use to inform our forecasts. But we are cursed with a relatively small number of historical observations that we can use to estimate predictive models. For example, we are lucky if we have 30 years of monthly data—or 360 observations—for many important time series. For really important events like recessions or crises, there are even fewer observations. There have been only 9 recessions in the US during the period since 1955 where we have any significant amount of historical data that we might have used to predict them. If we want to predict the probability of a recession within 3 years, we have only 20 observations of non-overlapping 3-year periods that we can use to fit a model.

Furthermore, the data we have in hand are imperfect and incomplete. At any given point, we face a "jagged edge" of data, where we are typically still waiting for some data to be released about the previous month or quarter, even as we already have more timely data like sentiment surveys or unemployment claims for the current period. We receive a steady stream of data prints at quarterly, monthly, weekly, and daily frequencies, which may refer to the current period or to some previous period, and more new "alternative data" sources become available every day. What's more, many of the data that we have received can be revised—sometimes substantially—in future months and years.

Ideally, our forecasting models would work well in this context—we need to be able to estimate our models reliably in relatively short historical samples, but they should still be capable of ingesting a steady stream of many different time series and immediately producing updated predictions. Models like this provide forecasts that are always current and never need to wait for the next data print. Plus they are also useful to us humans as tools for interpreting the meaning of the data we receive each day. If our clients want to know what to make of a sharp decline in some measure of business confidence, for example, we can tell them how it affects our model forecasts for GDP growth or recession risk.

Macroeconomists have typically approached these tasks with econometric tools built upon the workhorse Ordinary Least Squares (OLS) linear regression. For example, vector autoregressions (VARs) allow joint forecasting of a number of time series, although they perform poorly if that number is more than a few. Other tools like factor-augmented VARs or Bayesian VARs can incorporate a larger number of time series, while dynamic factor models (DFMs) are built especially to handle the jagged edge of data in real time. Quantile regressions allow predicting different percentiles of the probability distribution of a variable, in addition to the mean forecast. And, of course, more recently we have begun to hear about tools from the machine learning (ML) and artificial intelligence (AI) toolboxes, like lasso and ridge regressions, random forests, boosted trees, support vector machines, and neural nets. It is these latter tools that are recognizing faces, translating foreign languages, and making us wonder whether the economics toolbox is out of date.

Figure 1: Forecast errors by model family



Source: J.P. Morgan

We first experimented with some of these newer tools in a horserace to predict monthly payroll growth in a note in 2016, but we came away unimpressed. The most flexible tools like random forests, boosted trees, and neural nets, which are capable of discovering hidden patterns in massive datasets, seemed to add little value in our context of short macroeconomic time series. In the subsequent years, however, our colleagues in quantitative equity, quantitative fixed income, technical, volatility, and cross-asset strategy seem to have had more success with these tools. It is true that their datasets of daily or weekly market returns can be orders of magnitude larger than ours, opening up the possibility that the more flexible tools can add more value by discovering patterns in these larger datasets. But we were still left wondering if we might have missed something.

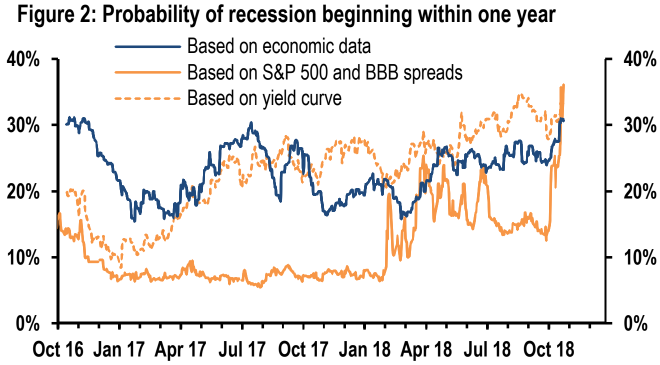
We return to the question in this report by conducting an extensive horserace across a broader set of econometric and machine learning methods to predict a variety of macroeconomic data. But, in fact, our results are similar. We still find

JPMorgan Chase Bank NA
Jesse Edgerton (1-212) 834-9543
jesse.edgerton@jpmchase.com
J.P. Morgan Securities LLC
Dan Weitzenfeld (1-646) 343-4033
dan.weitzenfeld@jpmorgan.com

that relatively simple, linear tools like OLS, principal components analysis (PCA), ridge regression, and the lasso perform best in our small datasets (Figure 1). The most flexible nonlinear ML and AI tools do no better, and sometimes much worse. Although these tools may be quite successful in learning how to recommend movies based on millions of ratings or detect credit card fraud from billions of transactions, they seem to add little value in our tiny datasets of a few hundred monthly observations. Their performance around the financial crisis was particularly bad in some cases, as these tools struggled to extrapolate outside the range of their prior experience.

In short, we doubt that macroeconomic forecasters have much to gain by adopting these most flexible new AI methods. But many of our readers are likely still unfamiliar with some of the linear tools like PCA, ridge regression, lasso, and elastic net, which we find most useful. So we show how these tools work in a simple example, and we demonstrate the techniques that we think are most worth adopting. We show with Eviews and Python code how to build forecasting models that can take signals from a long list of explanatory variables, update every day in real time, and quantify probabilities of other outcomes in addition to producing a mean forecast. These are some of the key methods that underlie our daily nowcasters, recession risk and other models that we publish in the US Real-time Quant Econ Monitor (Figure 2). We conclude with a discussion of techniques we think are worthy of further exploration.

Figure 2: Probability of recession beginning within one year



Source: Various government and non-government sources, J.P. Morgan

A horserace: the track

In this report, we conduct an extensive horserace across a variety of econometric and machine learning techniques to forecast several important macroeconomic time series at multiple horizons. This section lays out the details of how we conduct the race.

Left and right-hand-side variables

In our earlier work, we found that the most flexible ML and AI models brought few benefits in forecasting payroll growth in the upcoming month. But we are also interested in predicting a wide variety of other macroeconomic variables, and at horizons longer than one month. In this note, we choose three key macroeconomic time series to forecast, with representatives from the labor market, consumer, and business sectors: nonfarm payrolls, core retail sales (excluding autos, building supplies and gas stations), and core capital goods spending (shipments of nondefense capital goods excluding aircraft, plus imports minus exports). And we are also interested in making predictions at forecasting horizons beyond one month. For each of these series, we forecast changes in their logarithms over 1-month and 6-month horizons, which approximate their growth rates over these periods.

For the independent variables—the "predictor," "explanatory," or "right-hand-side" variables that we use to forecast the target—we used 44 other key macroeconomic series, ranging in frequency from weekly (e.g. unemployment claims) to monthly (e.g. the ISM nonmanufacturing index and industrial production) to quarterly (e.g. the investment-to-gdp ratio). These variables incorporate all of the key variables from our GDP growth "nowcasters" and our models of recession risk, and the variables are transformed in the same manner as in those models (e.g. we use the level of the ISM indexes and the growth rate of industrial production). We include both a 1-month version of each variable (like growth in industrial production over the past month) and a 6-month version (growth in industrial production over the last six months), as well as lags of these variables. Because the total list of variables including lags is in the hundreds and many are highly correlated with one another, we also allow our models the option of using a set of principal components computed from these variables in place of the full dataset, as we discuss further below.

Real-time data

As we noted up front, forecasters working with macroeconomic data face a "jagged edge" of data that are subject to future revisions. For example, as we write, we are attempting to forecast growth in payrolls for the month of October 2018. We have already received data for September payrolls, several October business surveys (like the Empire State and Philly Fed manufacturing surveys), and initial claims for unemployment insurance in the week ending October 20. But we do not yet have data for the October ISM nonmanufacturing survey or even *September* factory goods orders. Plus, many of these data will be subject to future revisions. Some, like the business surveys and initial claims, will be subject only to relatively minor revisions to seasonal adjustment factors, while others like factory goods and payrolls can see large revisions if, for example, a future census of businesses finds that the surveys underlying these reports draw from a larger population of businesses than is currently assumed.

For this report, we build a more sophisticated system for replicating as closely as possible the actual data that would have been available to us in real time at any day in the past. The St. Louis Fed archives real-time data like this in a service called ALFRED, and we download and use these data for all of the variables in our models where they are available. Only a limited subset of time series and vintages are available through ALFRED, however. For data series not covered by ALFRED, we construct a stylized version of the data release calendar and assume that the current vintage of the data became available on those historical release dates. We can thus nearly replicate the jagged edge of which data releases were available on any given date in the past, and populate them with the vintage data that were actually available on that date for many key series.

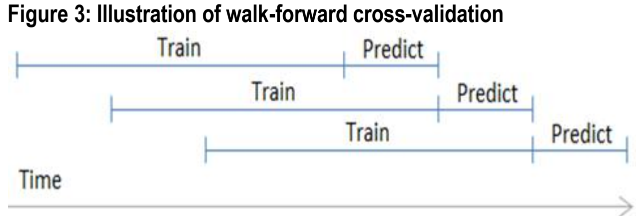
Cross Validation

If applied naively, machine learning models can easily overfit the training (or in-sample) dataset – they can learn patterns that will not be present in new, unseen data. For example, long-time forecasters are well aware that including more and more right-hand-side variables in a forecasting regression will always increase the R2, but eventually produce nonsensical coefficients and poor out-of-sample forecasting performance. Indeed, some of the more flexible ML and AI models are capable of memorizing every detail of the training set. To ensure that ML models learn from the training set only those patterns that can be expected to be present out of sample – to ensure that results generalize well – we use *regularization* and *cross-validation* to choose the best set of *hyperparameters* for each model.

Regularization means intentionally limiting the learning capacity of the model so that it can only fit less flexible functional forms that will have a better chance of performing well out of sample. And the degree of regularization for a given model can be governed by what we call a hyperparameter. For example, in building a classic linear regression model to forecast payrolls, we might use some kind of step-wise model selection procedure to select the best model that uses X different right-hand-side variables, where X could be 1, 2, 3, or more. X is a hyperparameter of this model selection procedure—it controls how complex the resulting model can be. Veteran forecasters might guess that the best X to choose in this case might be something between 1 and 5. If we use only the single best variable on the right-hand-side of our forecasting model, we would likely be leaving information on the table from other potential predictors. But if we use the 10 best variables for forecasting, our model would likely be overfit and perform poorly out-of-sample. *Hyperparameter optimization* is the process of choosing the value of these parameters in order to maximize out-of-sample forecasting performance.

In particular, we use *cross-validation* to perform the hyperparameter optimization and determine the optimal degree of regularization. Cross-validation means that we iteratively estimate (or "train") our model on some subset of our data (the "training sample"), make forecasts on data that the model has not yet seen (a "validation sample"), and then compare those forecasts with the actual results. By repeating this process many times on different subsets of the data, we can estimate how the model will perform out-of-sample as a function of the hyperparameters, and then choose the best set of hyperparameters to use when we make our actual forecasts. Cross-validation serves another purpose as well: it provides estimates of the magnitude of our out-of-sample errors. In other words, it gives us a sense of the size of the errors we can expect when making out-of-sample forecasts.

Figure 3: Illustration of walk-forward cross-validation



Source: J.P. Morgan

For this exercise, we use *grid-search walk-forward cross-validation.* Grid-search means that we define a space of possible hyperparameter combinations for each model, and simply try all of them. With walk-forward cross-validation, we train on the data observed as of a given point in time, make forecasts, then shift forward in time and repeat, as illustrated in Figure 2. For example, we use our real-time data setup to replicate the macroeconomic data that would have been available on January 1, 1998. Then, for every kind of model and for every possible combination of hyperparameters for that model, we estimate the model using those hyperparameters and make a forecast for December 1997 payroll growth, which had not yet been released on January 1, 1998. We make forecasts for payroll growth over the six months from November 1997 to May 1998, as well as growth in retail sales and core capital goods spending over the next one and six months. We store the forecasts from these thousands of different models, and then repeat the process.

That is, we step forward one month and replicate the macroeconomic data as it would have looked on February 1, 1998, and again estimate every model we would use to predict growth in payrolls, retail sales, and capital goods spending over the next 1 month and next 6 months.[1] Across all six dependent variables and every hyperparameter combination for every model, we make more than 10,000 forecasts in each month since 1998.

We walk forward, repeating this process up until the data we had available as of January 1, 2006. Then, using these forecasts generated from 1998 through 2006, we evaluate the performance of each of the 10,000 hyperparameter combinations over this period by computing the root-mean-squared error (RMSE) of the forecasts generated by each combination. We then pick the best set of hyperparameters for each type of model as the one with the lowest RMSE.

We can then compare these RMSEs across models to see which ones did the best in the one-step-ahead out-of-sample forecasting periods from 1998 into 2006 (we call these the "validation sample" results below). And then we subject these models to an even tougher test—comparing the performance of the winning hyperparameter combinations from 1998 to 2006 in the next period from 2007 onward, which included the financial crisis and recession (the "holdout" sample). The entire process involves making more than 2 million one-step-ahead forecasts and comparing them to realized outcomes.

A horserace: the field

Before getting to the results, we discuss the models we include in the horserace in more detail. We group the models into five families—univariate, linear, econometric, nonlinear machine learning, and neural networks—although this grouping is somewhat arbitrary. We attempt to give a flavor of how each type of model works while leaving out many details that can be found elsewhere.

Univariate Models

The "univariate" models rely only on the history of the variable being forecast and are particularly useful as benchmarks against which we can compare more complex models.

Trailing 3 month average: This simple estimator plays the role of a naïve benchmark.

Exponential Smoothing: A weighted average of lagged values, with weights decaying exponentially the longer the lag.

ARIMA: Short for Auto-Regressive Integrated Moving Average, ARIMA models essentially involve regressing the current value of the dependent variable on its lags and on the lags of the regression errors.

Linear Regression and Machine Learning Models

Linear Regression: Ordinary least squares (OLS) regression finds the linear relationship between the target variable and the independent variables that minimizes mean squared error. For this model (and most others below), we include as an additional hyperparameter in our estimation the choice of whether to use all of the right-hand-side variables directly in the regression, or instead to first reduce them down to 1, 3, 5, 10, or 20 principal components. So our "linear regression" results below include what we might also call "principal components regression," or regression on a set of principal components from the whole dataset.

Linear Regression – Small Data Set: We also include a linear regression on a reduced data set: just the lags of the target variables, the level of unemployment claims, and the Philly Fed Manufacturing Outlook survey, a classic set of variables that we would use in a simple forecasting model, particularly for payroll growth.

Ridge / Lasso / Elastic Net: This group of estimators is similar to OLS regression, but performs additional regularization by penalizing the total magnitudes of all of the coefficients in the regression. Where OLS coefficients are chosen by minimizing the sum of squared residuals (SSR) from the model, these models minimize the sum of the SSR plus a penalty term equal to a hyperparameter times the sum of the squared coefficients or of the absolute value of the coefficients. "Ridge regression" penalizes the sum of the squared coeffi-

[1] In some machine learning applications, researchers take care to create an "embargo" period between training and test observations, to prevent correlation between adjacent observations from "contaminating" model performance. In our application, however, this is exactly what we aim to do—we want to take advantage of any correlation between today's data and tomorrow's to help us forecast tomorrow. In macroeconomic forecasting, because the total amount of data is relatively small and new data are released at a low frequency—at most, we receive a few new data points each day—we re-estimate our models after we receive each new data point. This contrasts with other applications where complex models could be fit to billions of data points, with thousands more arriving each day. In these situations, it may not be feasible to re-estimate the parameters of the model often, and it might be important to focus on the performance of model predictions when using parameter estimates from some time in the past.

cients, the "lasso" (least absolute shrinkage and selection operator) penalizes the sum of the absolute values, and "elastic net" penalizes both. The key hyperparameters for this group of models determine exactly how much to penalize the coefficients. As these models turn out to be some of the most promising horses in our race, we demonstrate how they work in more detail in the following section.

Econometric Models

What we call the "econometric" models are somewhat more complex than the "linear" models above but still have their roots in the economics literature, in contrast to the "machine learning" models" below.

Bayesian VAR: Vector autoregressions (VARs) involve regressing a vector of indicators on its lags, such that each indicator in the vector is modeled as a function of both its own lags the lags of the other indicators in the system. In a Bayesian VAR, we use Bayesian inference to fit the model, with a prior distribution on the parameters that encourages sparsity of the coefficient matrix. This sparsity serves a similar function to the regularization in the ridge and LASSO models above, in that it discourages large and nonsensical coefficient estimates, even when estimating a large number of coefficients from a short time series sample.

Dynamic Factor Model: In a dynamic factor model, we model a potentially large number of macroeconomic series as being driven by a much smaller number of latent factors. In our implementation, we use "probabilistic principal components analysis" to model the latent factors, which allows us to handle the jagged edge elegantly: the estimated factors for "partially observed" periods are those which are statistically most likely given the data we have observed and the historical dynamics of the factors.[2] After estimating the factors, we regress the variable we are trying to predict on its lags and these factors. A related framework underlies the global GDP nowcasters that our global economics colleagues have been publishing since 2012.

Nonlinear Machine Learning Algorithms

By most definitions, "machine learning" means something like "using data to make predictions," so the term could encompass all of the models we consider here. But, in describing our results below, we use the term to refer to the following set of nonlinear models that originated outside of the economics literature.

Random Forest: A single "decision tree" is built by iteratively splitting observations in the training dataset such that the values of the target variable on each side of the split are the most similar. To an economist, a decision tree produces something like a "saturated model" of interacted dummy variables that can essentially find the average value of the dependent variable in narrow bins of observations. A random forest is an average of many decision trees, with the average of the individual trees' predictions becoming the final prediction of the forest. The "random" part of the random forest comes from fitting each tree on a random subset of the full training set, and only considering a random sample of variables at each tree split-point. The hyperparameters for the random forests govern things like how many of the right-hand-side variables can be considered when splitting the tree, and the number of splits that will be allowed in each tree.

Gradient Boosted Decision Trees: Like random forests, boosted trees are ensembles (or combinations) of decision trees. With boosted trees, however, each tree is trained on the residuals still left after fitting a sequence of prior trees.

K-Nearest Neighbors: The KNN algorithm finds the K months in the historical training sample that are "nearest" to what the data look like now, and its forecast is the average of these neighbors' values for the dependent variable. This algorithm thus mimics the intuitive idea of finding the past historical periods that looks most similar to today, and choosing a forecast as the average outcome from those past periods.

Support Vector Regression: Support vector machines are based around the idea of choosing optimal separating hyperplanes to divide the training dataset into regions that best predict the dependent variable. Support vector regressions extend this idea to allow the boundaries of the regions to be nonlinear in various ways and to allow continuous as well as categorical dependent variables.

Neural Networks

Loosely inspired by a simplified model of the brain, neural networks consist of layers of "neurons" that each take a weighted function of their inputs, run it through a non-linear activation function, and pass the results on to the next layer of neurons. With complex, often large networks of neurons,

[2] This handling of the jagged edge contrasts with the principal components approach described in the linear regression section. There we simply compute the principal components of the set of the most recent available observation from each right-hand-side variable, regardless of whether they refer to the same time period. Note that this is the only model in our horse race that is able to learn from both principal components and the lags of the target variable; all of the other multivariate models are able to choose between using principal components or the full dataset including lags.

JPMorgan Chase Bank NA
Jesse Edgerton (1-212) 834-9543
jesse.edgerton@jpmchase.com
J.P. Morgan Securities LLC
Dan Weitzenfeld (1-646) 343-4033
dan.weitzenfeld@jpmorgan.com

and non-linearity introduced by the activation functions, neural networks can learn extremely complicated relationships if given enough data.

Dense: The most basic neural network architecture, a dense or fully-connected neural network consists of layers of neurons, each of which receives inputs from every neuron in the preceding layer.

LSTM: A Long-Short-Term Memory network is a recurrent network – a class of neural networks specifically designed to handle sequences and time-series. The LSTM essentially walks along the sequence or time series, storing and (purposefully) forgetting information in its hidden state as it goes. When it reaches the end of the time series (the present), the hidden state can be used to predict the next value.

Linear the winner

Cutting to the chase, Table 1 lists the single model that performed best in predicting each of our six dependent variables during the one-step-ahead test periods through 2006, after choosing the best hyperparameters. In predicting core capital goods spending in the upcoming month, a random forest model was the winner by a narrow margin. For the other 5 out of 6 dependent variables, the best-performing models were all some combination of principal components analysis and a linear model like the ridge regression, elastic net, or the dynamic factor model. These results are similar to the ones we obtained in 2016 for forecasting one-month payrolls growth, as well as recent results from our Australian colleagues.

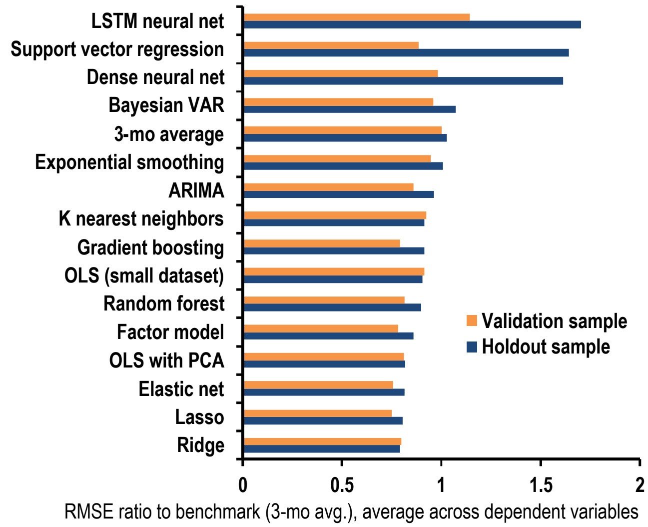
Table 1: Best performing models in validation and holdout samples

Target Variable		Validation sample	Holdout sample
Non-farm Payrolls	1-mo delta	Ridge with PCA	Elastic Net with PCA
	6-mo delta	Elastic Net	OLS with PCA
Retail Sales	1-mo delta	Factor Model	Random Forest
	6-mo delta	Factor Model	Factor Model
Core Capital Goods	1-mo delta	Random Forest	Elastic Net
	6-mo delta	Factor Model	OLS with PCA

Source: J.P. Morgan

The pattern of linear model outperformance continues in our holdout period. Figure 1 places the RMSEs for all models across the six dependent variables on the same scale by dividing each model's RMSE by the RMSE of the 3-month average for that dependent variable in the holdout sample period. For each model family, we take the average RMSE across the individual models, where each model is represented by its single best hyperparameter combination. On average, the models in the linear family have the lowest error in both the training and holdout period.

Figure 4: Forecast errors by model type



Source: J.P. Morgan

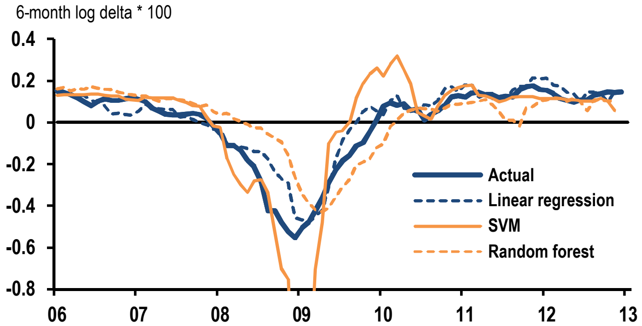
Figure 4 breaks out Figure 1 by showing the ratio of the RMSEs to the benchmark for each individual model, again averaged across our six dependent variables. The top five models in the holdout sample are all linear. To be fair, the best of the nonlinear machine learning models are not far behind, with the random forest and gradient boosting machine in sixth and eighth place. The differences between the RMSEs of these models are minor in practical terms, but there is certainly no sign of the nonlinear models doing better than the linear ones.

Nonlinear models make a few big mistakes

Some of the other, most flexible nonlinear models do very badly in the holdout period, however, with the support vector regression and both kinds of neural nets producing holdout errors that are twice as large as the best performing linear models. Figure 5 further illustrates this poor performance by showing the 1-step-ahead forecasts for 6-month growth in nonfarm payrolls, using the best hyperparameters chosen from the validation sample for the linear regression on principal components, the support vector machine, and the random forest. The linear regression does not too badly compared to the actual data—by no means does it foresee the crisis, but once the recession began in early 2008, the linear model began to predict further job losses. The random forest also eventually began to predict job losses, but it was slower to come around to this realization. The support vector regression, on the other hand, began making predictions in late

2008 that turned out to be wildly too pessimistic, before swinging around and making overly optimistic predictions early in the recovery.

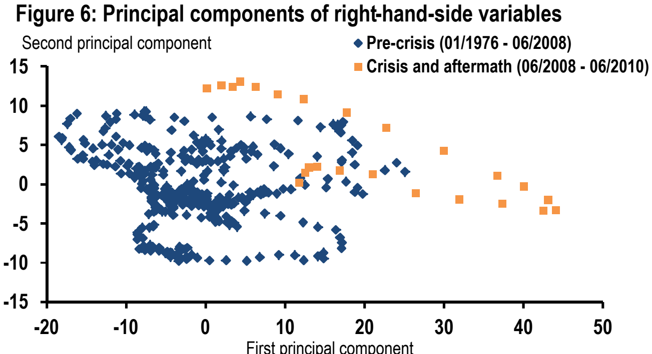
Figure 5: Nonfarm payrolls: 1-step ahead forecasts of 6-month growth



Source: J.P. Morgan

So why might some of the machine learning models perform so poorly during the crisis and its aftermath? This period was unprecedented, to be sure--at least compared to data from recent decades. Figure 6 shows just how unusual this period was by charting the first two principal components of our matrix of independent variables from June 2008 to June 2010. Both were beyond the extremes of the range experienced since 1976, when our earliest estimation samples begin. Our forecasting models thus had to tackle a regime they had not seen before in the training data.

Figure 6: Principal components of right-hand-side variables



Source: J.P. Morgan

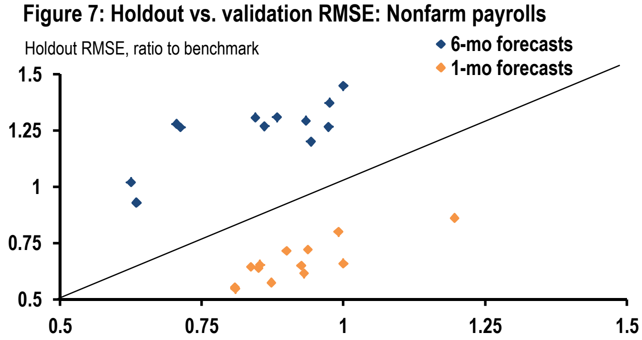
But why did some models do so much worse than others in forecasting during this period? We believe it is because the linear models were better able to make sensible extrapolations based on past experience, while the nonlinear models struggled with this task. Random forests and other tree-based methods, for example, are fundamentally ill-equipped to extrapolate: because their predictions are essentially an average over the outcome of the most similar observations in the training set, they responded only slowly as the crisis deepened in 2008. In contrast, both the linear regression and the support vector regression are able to extrapolate beyond the observations they have seen in the training data. But the SVR is essentially extrapolating nonlinear functions outside the range over which they were estimated, which in this case results in wildly overshooting both the trough in 2009 and the recovery in 2010.

Cross validation helps, but it isn't magic

Recall that our cross-validation procedure has two functions: to help us choose the best model and hyperparameters, and to predict the magnitude of out-of-sample errors. Figures 4 and 5 show that our cross-validation procedure struggled with the second task—in general, the predicted forecast errors based on the validation sample were lower than the actual errors seen during the holdout sample, in some cases by a dramatic margin. This example illustrates the fundamental shortcomings of the cross-validation approach--even a careful cross-validation will struggle to foresee the implications of events that are well outside the range of experience covered by the data available for training.

But all is not lost. For the most part, our cross-validation procedure did reasonably well in its other primary task—choosing the best models. Across all 3 of our target variables and both forecasting horizons, the model that performed best in the validation sample was either the best or close to the best in the holdout sample. Figure 7 provides an illustrative example by plotting holdout error against validation error for growth in nonfarm payrolls, at both the 1-month and 6-month forecast horizons. Each point represents one model type (like a ridge regression or a random forest), with the hyperparameters for each type chosen to minimize the validation sample RMSE. We can see that at both the 1-month and 6-month horizons, the model type that did the best in the validation sample was also very close to being the best in the holdout sample. That is, the blue dot that is furthest to the left (best in the validation sample) is also close to being furthest to the bottom (best in the holdout sample).

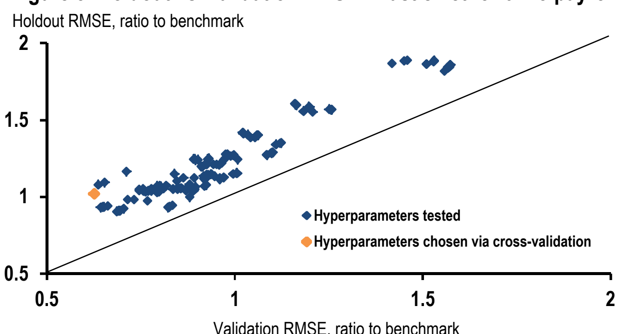
Figure 7: Holdout vs. validation RMSE: Nonfarm payrolls



Source: J.P. Morgan

In many cases, the cross-validation procedure also performed well in choosing the best hyperparameter combination within each model type. Figure 8 is similar to Figure 7, except rather than each point being a model type, each point is now a different hyperparameter combination for the elastic net in forecasting payroll growth over 6 months. As with our model selection process, the hyperparameter combination we chose via cross-validation is still one of best performers out-of-sample in the holdout period. That is, the left-most point is also one of the lowest. For most of our model types, the hyperparameters chosen by the cross-validation procedure were still some of the best ones in the holdout sample, as they were for the elastic net. The exceptions were the support vector and neural net models, which did so poorly in forecasting during the crisis.

Figure 8: Holdout vs. validation RMSE: Elastic net for 6-mo payrolls

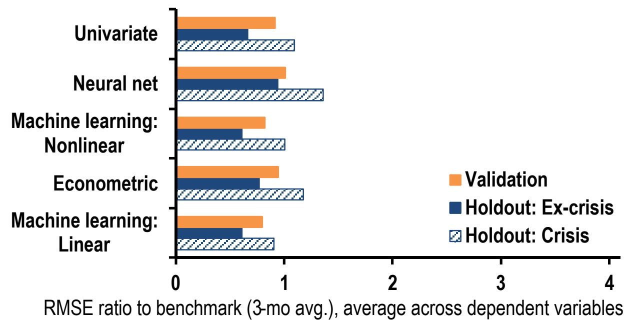


Source: J.P. Morgan

Sharp-eyed readers might notice one surprising feature of Figure 7—the orange dots are all below the diagonal line, indicating that that holdout error was below the validation error for the 1-month horizon forecasts. This result seems to be driven by two features of the holdout sample. First, it includes a long stretch from about 2012 to 2018 when payroll growth was relatively stable. In fact, Figures 9 and 10 show that RMSEs during the non-crisis holdout period were often lower than training RMSEs at both 1 and 6 month horizons. And second, the crisis was simply more extreme when viewed through the lens of the 6-month deltas than the 1-month deltas. A single large monthly decline is less of a rarity than a string of large consecutive monthly declines.

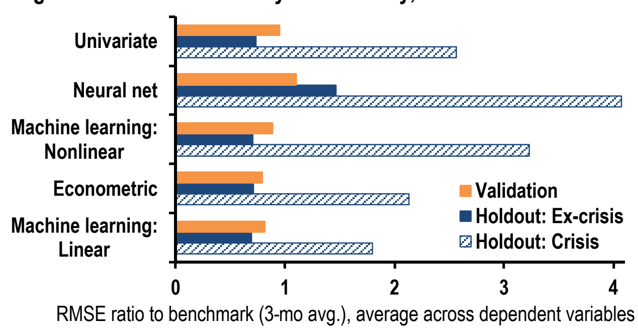
Indeed, we see in Figure 9 that all model families struggled mightily to forecast at a six-month horizon during the crisis and its aftermath. But the humble linear models did better than most alternatives, and saw forecast errors less than half the size of the worst-performing nonlinear models. The overall outperformance of the linear models seen in Figures 1 and 3 is thus driven largely by their relative performance in forecasting at the 6-month horizon during the crisis and its aftermath.

Figure 9: Forecast errors by model family, 1-month forecast horizon



Source: J.P. Morgan

Figure 10: Forecast errors by model family, 6-month forecast horizon



Source: J.P. Morgan

We thus conclude that the linear models are likely to be the best practical tools for macro forecasters. There is no sign that the ML and AI tools that get so much publicity are able to peer into the future by recognizing patterns that simpler models cannot see. Our datasets of monthly and quarterly observations are simply too small to take advantage of these tools' biggest strengths. And in fact, many of these models have the serious flaw of performing poorly when encountering conditions unlike anything they have seen before, which could prove outright dangerous in some situations. The linear models cannot see the future with precision either, of course, but at least they can be counted on to make reasonable predictions even in unfamiliar territory.

Understanding the linear models

Although these various linear techniques are on the simpler end of the range of models we consider, we suspect many of our readers are still less familiar with these tools than with workhorse Ordinary Least Squares (OLS) regressions. We thus provide a brief example to demonstrate how they work.

Table 2 shows results from five different models that predict monthly growth in core capital goods orders based on ten different surveys of businesses: the Institute for Supply Management (ISM) surveys for the manufacturing and nonmanufacturing sectors, plus eight different surveys from regional

Federal Reserve Banks. Before diving into the results, think for a moment what we should expect a "good" forecasting model would look like here. All of the business surveys are individually positively correlated with capital goods orders, and all are likely to capture a somewhat different selection of firms, so each could capture some unique bit of information not included in the others. Thus, while some of the surveys might be more useful than others, a sensible model would likely place a positive coefficient on each of them.

Table 2: Coefficients in predicting capital goods orders

RHS variable	OLS	PCA	Ridge	LASSO	Elastic-Net
ISM Manufacturing	1.495	0.091	0.119	0.708	0.572
ISM Nonman.	-0.587	0.083	0.010	0	0
Philly Fed Mfg.	0.386	0.088	0.091	0.020	0.162
Empire State Mfg.	-0.398	0.080	0.064	0	0
Kansas City Mfg.	0.194	0.079	0.095	0.053	0.131
Richmond Mfg.	-0.300	0.087	0.058	0	0
Dallas Mfg.	-0.559	0.091	0.072	0	0
NY Fed Nonman.	-0.236	0.079	0.034	0	0
Richmond Nonman.	-0.283	0.075	-0.026	-0.076	-0.265
Dallas Nonman.	0.931	0.089	0.079	0	0.131
Constant	0.084	0.084	0.084	0.084	0.084

Source: Census Bureau, ISM, Federal Reserve, J.P. Morgan. Figures are coefficients in model predicting monthly growth in core capital goods orders using normalized versions of the listed series. The estimation sample is 2007-2018, and hyperparameters for the models are set to illustrative values.

The first column of Table 2, however, shows that if we include all ten surveys in a simple Ordinary Least Squares regression in the data since 2007, they produce erratic and implausible coefficients. The coefficients range from -0.587 on the ISM nonmanufacturing index to 1.495 on the ISM manufacturing index, and six of the ten coefficients have the "wrong" (negative) sign. This is the classic "multicolinearity" problem with OLS—including multiple highly correlated variables on the right-hand-side produces erratic results. Small changes in the values of the survey data could produce large swings in the coefficient estimates and in the forecast from the model. Experienced forecasters are well aware that predictions from a model like this would be largely garbage, a result that we have confirmed more rigorously in prior work.

Table 2 also shows how the four linear machine learning models each handle this problem a bit differently. For the principal components model in the table, we calculate the first component of the surveys and include the component in an OLS regression to predict capital goods orders. We then compute the implied coefficient on each survey in the regression by taking the loading that each survey receives in the component and multiplying this by the coefficient on the component in the regression. This procedure shows that each survey effectively receives a small, positive coefficient in the regression, ranging from 0.075 for the Richmond Fed nonmanufacturing survey to 0.091 for the ISM manufacturing index. More variation in the coefficients across the surveys could be allowed by including additional principal components in the regression model. For example, our primary GDP nowcaster is based on three principal components. We also see that the ridge regression produces somewhat similar results, with most of the surveys receiving small positive coefficients, but the Richmond Fed nonmanufacturing survey a small negative. Adjusting the hyperparameter to vary the amount of penalization for large squared coefficient magnitudes would alter these results. Increasing the penalty parameter would tend to reduce the variation across coefficients and could eliminate the negative coefficient, while decreasing the penalty parameter would move the ridge coefficients towards the OLS results.

Because the lasso penalizes the absolute values of the coefficients instead of their squares, it has the effect of setting some of the coefficients to zero, while still allowing a broad range of values across the nonzero coefficients. For example, the ISM manufacturing index receives a coefficient of 0.708 and the Richmond nonmanufacturing index -0.076, while six of the surveys are zeroed out. The elastic net looks like a combination of the ridge and lasso results in that it still sets some coefficients to zero, but spreads magnitudes more evenly across the nonzero coefficients. For example, where the ISM and Philly Fed manufacturing indexes receive coefficients of 0.708 and 0.020 with the lasso, they receive .572 and 0.162 with the elastic net.

Our results from the horserace show that each of these methods tends to produce similar out-of-sample forecasting performance. So which should we actually use in practice? One additional consideration is transparency and interpretability, which is a necessary requirement for any model that we would actually take seriously in forming our outlook--that is, we need to be able to understand which variables are driving the model. In fact, all of these models do well by that criterion. It is easy to calculate and report the contribution that each variable makes to the forecast. Or in contexts with larger sets of explanatory variables, it is easy to communicate the forecast contributions from related groups of variables.

One final consideration, however, is how useful the models will be in interpreting the data we receive each day as we fill in the jagged edge. Financial markets react to the implications of each data point we receive in real time, and they do not have the luxury of waiting until all data for the month are available before considering their implications. The lasso model in Table 1 would thus be at a disadvantage. It would entirely ignore the Empire State manufacturing survey and the New York Fed's services survey, which are among the very first of the business surveys we receive each month, and it would only react subtly to other early data points like the Philly and Kansas City Fed manufacturing surveys. When our clients ask us the meaning of a move in the Empire State index, the lasso would have no answer for them. It would essentially have to wait to provide a useful forecast for the release of the ISM manufacturing survey a few weeks later. Thus, in general, we tend to prefer models like the PCA and ridge regressions in the table, which we can use to help us interpret every data release in real time.[3] The models in our Quant Econ Monitor, which update every day with new data, are based primarily on the principal components approach.

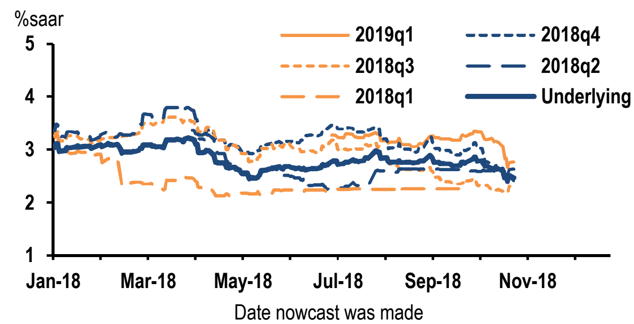
What's worth doing

With these horserace results in hand (and more years of practical forecasting experience than we like to admit), we are ready to draw conclusions about which of these methods are actually worth implementing. In our view, it is relatively easy for forecasters predicting monthly or quarterly data to create forecasting models that do as well or better than sophisticated machine learning techniques, even when working with limited technical knowledge and traditional economics tools like Eviews or Stata. In particular, it is relatively easy to build forecasting models that:

Use ALL the data. Forecasters have access to dozens (or even hundreds or thousands) of time series that are potentially useful in forecasting any given variable of interest, and there is no need to restrict forecasting models to a handful of relevant predictors due to multi-colinearity. Methods like principal components or ridge regression can extract signals from a large number of explanatory variables, while still being robust and transparent.

Update in real time. Although data for a given period trickle out over time, forecasting models never need to "wait" for the last data point of the month to produce a forecast. All that is required is a sensible method of forecasting any explanatory variables that are still missing. As more data are released, forecasts of the explanatory variables are replaced with actual values, and the overall forecast can update each day. Figure 11 shows results from our US nowcaster, which incorporates signals from more than 100 time series to provide daily estimates on the pace of GDP growth. Tools like the dynamic factor models that underlie our nowcasters are built for this purpose, but there are also simpler methods for doing this, as we show below.

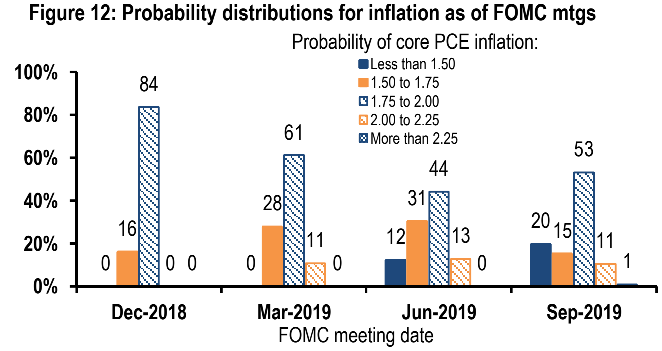
Figure 11: JP Morgan nowcaster of US GDP growth



Source: Various government and non-government sources, J.P. Morgan

Predict probabilities as well as averages. Although we have not focused on it thus far, one result from the horserace above is that even the best forecasting models are actually pretty bad at their jobs. Our ability to predict the future with precision is simply quite limited, even when using the best possible forecasting techniques. Thus we think it worthwhile to spend as much or more time and energy quantifying risks as we do producing the mean or modal "baseline" forecast. Our models of recession risk are a key example, and we have also developed models for quantifying the probability distribution of GDP growth, unemployment, and inflation (Figure 12). Although it has not been our focus in this note, in prior work we have also found that relatively simple tools do no worse and often better than complex models at this task as well. None of our results here would suggest any reason to think otherwise.

Figure 12: Probability distributions for inflation as of FOMC mtgs



Source: Various government and non-government sources, J.P. Morgan

[3] Averaging forecasts across multiple models would alleviate this concern to some extent, though including the LASSO in the average would still tend to overweight the indicators selected by the LASSO. Another possibility would be to update the LASSO model forecasts for capital goods by first updating an auxiliary model forecast for the ISM in response to the Empire State. This setup would still allow the Empire State to affect the capital goods forecast, although we find the PCA or dynamic factor model approach to be a more straightforward method of achieving this linkage.

And how to do it

We demonstrate these points through an implementation of these methods in Eviews and Python code, for the same example of predicting capital goods orders using the business surveys that we discussed above. In the code in Box 1 below, the Eviews group Surveys contains the data on all of the business surveys, and the date forecastMonth will be the next monthly release for capital goods orders that we want to forecast. The code loops through all of the surveys in the group, and if a given survey has not yet been released for forecastMonth, it forecasts the survey based on a regression on two of its own lags. The code then calculates the first two principal components of the surveys (including any forecasted values), and it uses these components to forecast growth in capital goods orders (along with two lags of the dependent variable). The code uses the same right-hand-side variables to estimate a quantile regression and predict the 10^{th} percentile of the distribution as a method of monitoring whether downside risks are building. It also runs a linear regression to predict the probability of growth above 1%, which could be useful, for example, if an analyst thought that was a critical threshold that could induce a move in machinery stocks. Python code performing the same procedure is in Box 2.

This code can run in a few seconds and produce a forecast every day, even if only a subset of surveys has been released. For example, immediately after the capital goods data were released for September, the code would have begun forecasting any missing surveys for October based on their own lags, and then using those forecasts to construct the capital goods forecast for October. As the October surveys are released, the survey forecasts will be replaced with actuals and the forecast for capital goods orders will also update.

And, in fact, there is no need to limit this model to the ten business surveys discussed above. The underlying ingredients of principal components analysis and ordinary least squares regression will work robustly in datasets of any size, with no dependence on particular solution algorithms, parameter starting values, or other considerations that can complicate estimation of more complex models. Thus this same code could easily accommodate dozens, hundreds, or thousands of explanatory variables that it might be useful to include. Our horserace results above show no evidence that more complex econometric, machine learning, or artificial intelligence methods would make better predictions than this kind of model, although some could bring an increase in computation time, a loss of transparency and interpretability, and poor performance in unfamiliar environments.

In terms of the statistical software tools, there are pros and cons to both. The Eviews code is shorter, as we often find to be the case. Eviews is a high-level package that is custom-built to perform exactly these kinds of tasks with time series data, and thus it features built-in handling of things like sample selection and missing values that require extra lines of Python code. It is thus no surprise that the Eviews code can be shorter. We also find that these features can make the Eviews code easier to read and write in some cases, so working in Eviews often requires somewhat less programmer time to accomplish the same task. But, by requiring the code to detail every step of constructing the dataset to be used in fitting a model, Python can force the revelation of logical errors that can slip by undetected in Eviews code.

And, of course, Eviews is also a less powerful and flexible tool. It currently lacks the capability to estimate even the simpler machine learning models like ridge regression and lasso, which we think that even forecasters working with short time series may find useful (although to be fair, Eviews also offers some more traditional econometric methods that do not seem to be available in any python packages). And, in general, python will more easily interact with a wider range of other tasks that we might want our code to perform. For example, if we wanted to use sophisticated machine learning tools to turn satellite images into time series data, we would likely do this in a package like python, and it would be somewhat easier to then run time series forecasting regressions using the data as part of the same python code. Similarly on the other end, Python has more powerful capabilities to take the output from forecasting models and turn it into chartpacks, web pages, trading signals, or other output than Eviews. For example, our Quant Econ Monitor is generated by Python code each day. Python also has some advantages here over other packages like R, which also offers the full suite of machine learning estimators. So we will likely work primarily in Python in the future, though, in truth, a great deal of what we do can be done well in Eviews.

What's next

Of course, there remains ample room for creative forecasters to add additional art, science, technology, theory, or other improvements to the framework in this code. A key piece of the art of forecasting is selecting and constructing the explanatory variables to include on the right-hand-side, ideally informed by economic theory. And there would be many ways to improve the guts of the forecasting procedure presented here. As just one example, the code shown above has no means for the release of the Empire State survey to affect the forecasts of the other surveys to be released later in the month, although this mechanism is a key component built

JPMorgan Chase Bank NA
Jesse Edgerton (1-212) 834-9543
jesse.edgerton@jpmchase.com
J.P. Morgan Securities LLC
Dan Weitzenfeld (1-646) 343-4033
dan.weitzenfeld@jpmorgan.com

into the dynamic factor model framework behind all of the tools in our daily Quant Econ Monitor. We might also want to combine various aspects of the many models we have discussed—for example, by including both ARIMA terms and principal components of other variables in a single model, or by creating averages or ensembles of multiple models.

Although we have made useful progress in developing practical methods for predicting probabilities, we suspect there may still be more value to add from experimenting with alternative approaches to this problem, as well as simply identifying additional factors in the economic data that signal the presence of risks or vulnerabilities. And there remain whole swaths of forecasting questions that we have not touched on here. For example, more "structural" approaches to forecasting, like the Fed's FRB/US model or estimated Dynamic Stochastic General Equilibrium (DSGE) models rely more heavily on economic theory and facilitate counterfactual simulations of alternative policy paths like faster rate hikes or larger tax cuts. Another set of models focuses on handling regime changes or parameters that shift over time. We think this latter set of topics also merits further exploration, particularly as it relates to understanding shifts in the distribution of risks facing the economy.

We have also focused here on using new statistical techniques to make forecasts based on the same old economic data that we have always used. But there are also new sources of alternative data becoming available all the time. This is one area where the nonlinear machine learning tools that have added little value in our investigation here might still have something to offer to macroeconomists. The nonlinear tools are likely necessary, for example, to extract data from satellite images, even if we end up using linear models to incorporate the new data into our economic forecasts. Of course, with the wide range of macroeconomic time series already available to us, it remains to be seen how much value these new data sources can add to our forecasts, and we will return to this question in future work.

Box 1: Eviews code for PCA-based forecast

```
pagecreate(wf=rec,page=wk) m 2000 2020                                          '' Set up workfile and fetch data
fetch(d=usecon) nmocnx nmfc napmc
fetch(d=surveys) bocgx kcmca rimcx emgi dbacts blain risrx dsacts
series ccgo = nmocnx
string surveyList = "napmc nmfc bocgx kcmca rimcx emgi dbacts blain risrx dsacts" '' String of business survey variable names
group Surveys {surveyList}                                                      '' Create Eviews group of survey data
string ccgoLast = ccgo.@last                                                    '' Month of last observation of core capital goods
string forecastMonth = @datestr(@dateadd(@dateval(ccgoLast),1,"MM"),"yyyyfmm")   '' First month to be forecast = ccgoLast+1
for %survey {surveyList}                                                        '' Loop through surveys in the list
	if @elem({%survey},forecastMonth) = na then                                 '' If the survey is missing in forecastMonth
		string surveyLast = {%survey}.@last
		smpl 2007 {surveyLast}
		equation eq_{%survey}.ls {%survey} c {%survey}(-1) {%survey}(-2)    '' Forecast it based on its own lags
		smpl {surveyLast}+1 {forecastMonth}
		eq_{%survey}.forecast(f=na) {%survey}_forecast
		{%survey} = {%survey}_forecast
	endif
next
string pcList = "Surveys_pc1 Surveys_pc2"    '' String list of first two principal components to calculate
smpl 2007 {forecastMonth}
Surveys.makepcomp {pcList}                                                      '' Calculate the principal components
equation eq_ccgo_mean.ls 100*dlog(ccgo) c dlog(ccgo(-1)) {pcList}               '' Regression to predict mean growth
equation eq_ccgo_gt1.ls (100*dlog(ccgo)>1) c dlog(ccgo(-1)) {pcList}            '' Regression to predict probability of growth > 1%
equation eq_ccgo_p10.qreg(quant=0.1) 100*dlog(ccgo) c dlog(ccgo(-1)) {pcList}   '' Quantile regression to predict 10th pctile growth
smpl {forecastMonth} {forecastMonth}
eq_ccgo_mean.fit(d,f=na) ccgo_forecast                                          '' Calculate mean forecast
eq_ccgo_gt1.fit(f=na) ccgo_probgt1                                              '' Calculate probability of growth > 1%
eq_ccgo_p10.fit(d,f=na) ccgo_p10                                                '' Calculate 10th percentile growth
```

Source: JP Morgan

Box 2: Python code for PCA-based forecast

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
import statsmodels.formula.api as smf
import Haver
y_var = Haver.data('USECON:NMOCNX', startdate='2006-01-01')
df = Haver.data(['USECON:NAPMC', 'USECON:NMFC', 'SURVEYS:BOCGX ', 'SURVEYS:EMGI', 'SURVEYS:KCMCA',
                 'SURVEYS:RIMCX', 'SURVEYS:DBACTS', 'SURVEYS:BLAIN', 'SURVEYS:RISRX', 'SURVEYS:DSACTS'],
                frequency='m', startdate="2006-01-01")
# Log difference in CCG to be used as dependent variable
ccgo = (np.log(y_var['nmocnx']) - np.log(y_var['nmocnx'].shift(1))) * 100
ccgo_last = ccgo.last_valid_index()  # Month of last observation of Core capital goods
forecast_month = ccgo_last + 1  # First month to be forecast
# Define three ranges: through the last obs of ccgo; through the forecast month; and just the forecast month
reg_samp = pd.period_range(pd.Period('2007-01-01'), ccgo_last, freq='M')
fore_samp = pd.period_range(pd.Period('2007-01-01'), forecast_month, freq='M')
fore_mo_samp = pd.period_range(forecast_month, forecast_month, freq='M')
# Reindex the survey series so that it goes through forecast_month populating nan where no data is available
df = df.reindex(fore_samp, fill_value=np.nan)
for survey in df.columns:  # Loop through the surveys dataframe
    last_observed_period = df[survey].last_valid_index()
    if last_observed_period < forecast_month:  # If the survey is missing in forecast_month
        # Create a sample which goes from January 2007 to the last month of the survey
        lag_samp = pd.period_range(pd.Period('2007-01-01', freq='M'), last_observed_period, freq='M')
        # Create a sample which starts from the last date of the series+1 and goes through to forecast_month
        forecast_samp = pd.period_range(last_observed_period + 1, forecast_month, freq='M')
        # Create dataframe which has the first 2 lags of the dependent variable and drop rows with any NA values
        dfX = pd.concat([df[survey].shift(1), df[survey].shift(2)], axis=1)
        dfX_train = dfX.reindex(lag_samp).dropna(axis=0, how='any')
        dfX_predict = dfX.reindex(forecast_samp)
        eq_survey = LinearRegression().fit(dfX_train, df[survey].loc[dfX_train.index])  # Run the regression
        # Forecast the series through to forecast_month and update the Surveys dataframe
        df.update(pd.Series(eq_survey.predict(dfX_predict), name=survey, index=dfX_predict.index))

# Create a pipeline to normalize and then run PCA
p = Pipeline([('normalize', StandardScaler()), ('pca', PCA(n_components=2))])
df_pc = pd.DataFrame(p.fit_transform(df), index=df.index, columns=['pca_{0}'.format(i) for i in range(2)])
# Create an dataframe with X vars: our principal components, and the lag of ccgo
dfX = df_pc.join(ccgo, how='outer')
dfX['ccgo_lag1'] = dfX.nmocnx.shift(1)
dfX = dfX.drop('nmocnx', axis=1)
ols_mean = LinearRegression().fit(dfX.loc[reg_samp], ccgo.loc[reg_samp]) # OLS Regression to predict mean growth
ccgo_gt1 = (ccgo > 1).astype(int)                                       # Logistic Regression to predict probability of growth > 1%
ols_gt1 = LogisticRegression().fit(dfX.loc[reg_samp], ccgo_gt1.loc[reg_samp])
qreg_data = dfX.join(ccgo)                                              # Quantile regression to predict 10th quantile growth
qreg_p10 = smf.quantreg('nmocnx ~ ccgo_lag1 + pca_0 + pca_1 ', qreg_data.loc[reg_samp]).fit(q=0.1)
ccgo_mean_forecast = ols_mean.predict(dfX.loc[fore_mo_samp])  # Make predictions for the forecast month
ccgo_prob_gt1 = ols_gt1.predict_proba(dfX.loc[fore_mo_samp])
ccgo_p10 = qreg_p10.predict(qreg_data.loc[fore_mo_samp])
```

Source: JP Morgan

Analysts' Compensation: The research analysts responsible for the preparation of this report receive compensation based upon various factors, including the quality and accuracy of research, client feedback, competitive factors, and overall firm revenues.

Other Disclosures

J.P. Morgan ("JPM") is the global brand name for J.P. Morgan Securities LLC ("JPMS") and its affiliates worldwide. J.P. Morgan Cazenove is a marketing name for the U.K. investment banking businesses and EMEA cash equities and equity research businesses of JPMorgan Chase & Co. and its subsidiaries.

Principal Trading: J.P. Morgan and/or its affiliates normally make a market and trade as principal in fixed income securities discussed in this report.

Private Bank Clients: Where you are a client of the private banking businesses offered by JPMorgan Chase & Co. and its subsidiaries ("J.P. Morgan Private Bank"), research is issued to you by J.P. Morgan Private Bank and not by any other division of J.P. Morgan, including but not limited to the J.P. Morgan corporate and investment bank and its research division.

Legal Entities Disclosures
U.S.: JPMS is a member of NYSE, FINRA, SIPC and the NFA. JPMorgan Chase Bank, N.A. is a member of FDIC. **Canada**: J.P. Morgan Securities Canada Inc. is a registered investment dealer, regulated by the Investment Industry Regulatory Organization of Canada and the Ontario Securities Commission and is the participating member on Canadian exchanges. **U.K.**: JPMorgan Chase N.A., London Branch, is authorised by the Prudential Regulation Authority and is subject to regulation by the Financial Conduct Authority and to limited regulation by the Prudential Regulation Authority. Details about the extent of our regulation by the Prudential Regulation Authority are available from J.P. Morgan on request. J.P. Morgan Securities plc (JPMS plc) is a member of the London Stock Exchange and is authorised by the Prudential Regulation Authority and regulated by the Financial Conduct Authority and the Prudential Regulation Authority. Registered in England & Wales No. 2711006. Registered Office 25 Bank Street, London, E14 5JP. **Germany**: This material is distributed in Germany by J.P. Morgan Securities plc, Frankfurt Branch which is regulated by the Bundesanstalt für Finanzdienstleistungsaufsich and also by J.P. Morgan AG (JPM AG) which is a member of the Frankfurt stock exchange and is regulated by the Federal Financial Supervisory Authority (BaFin), JPM AG is a company incorporated in the Federal Republic of Germany with registered office at Taunustor 1, 60310 Frankfurt am Main, the Federal Republic of Germany. **South Africa**: J.P. Morgan Equities South Africa Proprietary Limited is a member of the Johannesburg Securities Exchange and is regulated by the Financial Services Board. **Hong Kong**: J.P. Morgan Securities (Asia Pacific) Limited (CE number AAJ321) is regulated by the Hong Kong Monetary Authority and the Securities and Futures Commission in Hong Kong and/or J.P. Morgan Broking (Hong Kong) Limited (CE number AAB027) is regulated by the Securities and Futures Commission in Hong Kong. **Korea**: This material is issued and distributed in Korea by or through J.P. Morgan Securities (Far East) Limited, Seoul Branch, which is a member of the Korea Exchange(KRX) and is regulated by the Financial Services Commission (FSC) and the Financial Supervisory Service (FSS). **Australia**: J.P. Morgan Securities Australia Limited (JPMSAL) (ABN 61 003 245 234/AFS Licence No: 238066) is regulated by ASIC and is a Market, Clearing and Settlement Participant of ASX Limited and CHI-X. **Taiwan**: J.P. Morgan Securities (Taiwan) Limited is a participant of the Taiwan Stock Exchange (company-type) and regulated by the Taiwan Securities and Futures Bureau. **India:** J.P. Morgan India Private Limited (Corporate Identity Number - U67120MH1992FTC068724), having its registered office at J.P. Morgan Tower, Off. C.S.T. Road, Kalina, Santacruz - East, Mumbai – 400098, is registered with Securities and Exchange Board of India (SEBI) as a 'Research Analyst' having registration number INH000001873. J.P. Morgan India Private Limited is also registered with SEBI as a member of the National Stock Exchange of India Limited (SEBI Registration Number - INB 230675231/INF 230675231/INE 230675231), the Bombay Stock Exchange Limited (SEBI Registration Number - INB 010675237/INF 010675237) and as a Merchant Banker (SEBI Registration Number - MB/INM000002970). Telephone: 91-22-6157 3000, Facsimile: 91-22-6157 3990 and Website: www.jpmipl.com. For non local research reports, this material is not distributed in India by J.P. Morgan India Private Limited. **Thailand**: This material is issued and distributed in Thailand by JPMorgan Securities (Thailand) Ltd., which is a member of the Stock Exchange of Thailand and is regulated by the Ministry of Finance and the Securities and Exchange Commission and its registered address is 3rd Floor, 20 North Sathorn Road, Silom, Bangrak, Bangkok 10500. **Indonesia**: PT J.P. Morgan Sekuritas Indonesia is a member of the Indonesia Stock Exchange and is regulated by the OJK a.k.a. BAPEPAM LK. **Philippines**: J.P. Morgan Securities Philippines Inc. is a Trading Participant of the Philippine Stock Exchange and a member of the Securities Clearing Corporation of the Philippines and the Securities Investor Protection Fund. It is regulated by the Securities and Exchange Commission. **Brazil**: Banco J.P. Morgan S.A. is regulated by the Comissao de Valores Mobiliarios (CVM) and by the Central Bank of Brazil. **Mexico**: J.P. Morgan Casa de Bolsa, S.A. de C.V., J.P. Morgan Grupo Financiero is a member of the Mexican Stock Exchange and authorized to act as a broker dealer by the National Banking and Securities Exchange Commission. **Singapore:** This material is issued and distributed in Singapore by or through J.P. Morgan Securities Singapore Private Limited (JPMSS) [MCI (P) 099/04/2018 and Co. Reg. No.: 199405335R], which is a member of the Singapore Exchange Securities Trading Limited and/or JPMorgan Chase Bank, N.A., Singapore branch (JPMCB Singapore) [MCI (P) 046/09/2018], both of which are regulated by the Monetary Authority of Singapore. This material is issued and distributed in Singapore only to accredited investors, expert investors and institutional investors, as defined in Section 4A of the Securities and Futures Act, Cap. 289 (SFA). This material is not intended to be issued or distributed to any retail investors or any other investors that do not fall into the classes of "accredited investors," "expert investors" or "institutional investors," as defined under Section 4A of the SFA. Recipients of this document are to contact JPMSS or JPMCB Singapore in respect of any matters arising from, or in connection with, the document. **Japan**: JPMorgan Securities Japan Co., Ltd. and JPMorgan Chase Bank, N.A., Tokyo Branch are regulated by the Financial Services Agency in Japan. **Malaysia**: This material is issued and distributed in Malaysia by JPMorgan Securities (Malaysia) Sdn Bhd (18146-X) which is a Participating Organization of Bursa Malaysia Berhad and a holder of Capital Markets Services License issued by the Securities Commission in Malaysia. **Pakistan**: J. P. Morgan Pakistan Broking (Pvt.) Ltd is a member of the Karachi Stock Exchange and regulated by the Securities and Exchange Commission of Pakistan. **Saudi Arabia**: J.P. Morgan Saudi Arabia Ltd. is authorized by the Capital Market Authority of the Kingdom of Saudi Arabia (CMA) to carry out dealing as an agent, arranging, advising and custody, with respect to securities business under licence number 35-07079 and its registered address is at 8th Floor, Al-Faisaliyah Tower, King Fahad Road, P.O. Box 51907, Riyadh 11553, Kingdom of Saudi Arabia. **Dubai**: JPMorgan Chase Bank, N.A., Dubai Branch is regulated by the Dubai Financial Services Authority (DFSA) and its registered address is Dubai International Financial Centre - Building 3, Level 7, PO Box 506551, Dubai, UAE.

Country and Region Specific Disclosures
U.K. and European Economic Area (EEA): Unless specified to the contrary, issued and approved for distribution in the U.K. and the EEA by JPMS plc. Investment research issued by JPMS plc has been prepared in accordance with JPMS plc's policies for managing conflicts of interest arising as a result of publication and distribution of investment research. Many European regulators require a firm to establish, implement and maintain such a policy. Further information about J.P. Morgan's conflict of interest policy and a description of the effective internal organisations and administrative arrangements set up for the prevention and avoidance of conflicts of interest is set out at the following link https://www.jpmorgan.com/jpmpdf/1320742677360.pdf. This report has