Neural Networks, the Treasury Yield Curve, and Recession Forecasting

Michael Puglia and Adam Tucker

KEY FINDINGS

- It is difficult to make the case that neural network classifiers greatly outperform traditional econometric methods such as probit regression when forecasting US recessions if performance is measured on the basis of forecast accuracy alone.

- That said, neural network classifiers identify important features of the joint distribution of recession over term spreads and other macro-financial time series that probit regression and other traditional methods cannot.

- The authors propose a three-step econometric process for conducting statistical inference on machine learning classifiers, with the goal of better explaining recession forecasts and enabling model outputs to be linked back to the instruments of monetary policy.

ABSTRACT

The authors use neural networks to examine the power of Treasury term spreads and other macro-financial variables to forecast US recessions and compare them with probit regression. They propose a novel three-step econometric method for cross-validating and conducting statistical inference on machine learning classifiers and explaining forecasts. They find that probit regression does not underperform a neural network classifier in the present application, which stands in contrast to a growing body of literature demonstrating that machine learning methods outperform alternative classification algorithms. That said, neural network classifiers do identify important features of the joint distribution of recession over term spreads and other macro-financial variables that probit regression cannot. The authors discuss some possible reasons for their results and use their procedure to study US recessions over the post-Volcker period, analyzing feature importance across business cycles.

TOPICS

Fixed income and structured finance, big data/machine learning, financial crises and financial market history*

It is well documented that an inverted Treasury yield curve is a strong signal of recession in the United States. The predictive power of term spreads to forecast recessions during the Volcker era was particularly strong, and the results of a probit regression of Treasury term spreads against an indicator of recession over a
four-quarter horizon will bear that out. Studies dating to the early 1990s were first to make this case formally.

Virtually every recession forecasting study after those first published in the 1990s has used some version of probit regression to make its conclusions. Many have attempted to extend the method to test and control for the time-series properties of term spreads and other inputs over recession to improve forecast performance. Few extensions have been satisfactory, however, and the fact that simple probit regression remains the workhorse method in this field is evidence of this.

Econometric research on this topic over the past 30 years has failed to improve on simple probit regression because all extensions have had to contend with one simple fact: the probit framework is rigid. This rigidity can be credited to the normal link function embedded in its parametric assumptions. To classify data, the probit method attempts to draw a hyperplane through the feature space by minimizing a cost function. In doing so, the optimization trades off goodness of fit in some areas of the feature space (by fitting the hyperplane separator well in those regions) against poorness of fit in other areas.

Machine learning methods for classifying data are an attractive alternative to probit regression in the present application. Whereas probit methods must typically introduce additional parameters to create model flexibility, flexibility is an innate feature of machine learning methods. That said, machine learning methods tend to overfit data, unless controlled sufficiently via their hyperparameters, which is not a problem that arises in probit regression. Thus, the flexibility of machine learning methods also presents new difficulties, which are managed via a bias-variance trade-off.

In this article, we investigate the performance of neural network classifiers vis-à-vis probit regression when forecasting US recessions using term spreads and other macro-financial data. We propose a three-step econometric method for cross-validating and conducting statistical inference on machine learning classifiers and explaining forecasts. The method is composed of (1) a nested time-series (NTS) cross-validation strategy that addresses issues posed by sparse economic data when conducting analysis using machine learning methods, (2) pairwise post hoc McNemar’s tests for selecting models and algorithms from many possible candidates, and (3) Shapley value decomposition of forecasts to aid in the economic interpretation of results.

In a preview of our results, we find that probit regression does not underperform a neural network classifier in the present application, which stands in contrast to a growing body of literature demonstrating that machine learning methods outperform alternative classification algorithms. That said, neural network classifiers identify important features of the joint distribution of recession over term spreads and other macro-financial variables that probit regression cannot, such as skewness and fat tails. We discuss some possible reasons for our results and use our procedure to study US recessions over the post-Volcker period, analyzing feature importance across business cycles.

**LITERATURE REVIEW**

Estrella and Mishkin (1996, 1998) conducted early work that uses financial and macroeconomic variables in a probit framework to forecast recession. They found that “stock prices are useful with one- to three-quarter horizons, as are some well-known macroeconomic indicators” for predicting recessions. For longer horizons, however, they concluded that “the slope of the yield curve emerges as the clear individual choice and typically performs better by itself out-of-sample than in conjunction with
other variables.” For recessions occurring during and before the Volcker era, negative term spreads turned out to be a very strong signal of impending recession.

The slope of the yield curve largely failed to predict the 1990–1991 recession, however. Dueker (1997, 2002) used Markov switching in the probit framework to allow for coefficient variation and investigated issues surrounding the application of probit methods to time-series data. He found that, although it is important to allow for dynamic serial correlation of term spreads in the probit framework, “allowance for general coefficient variation is not particularly significant at horizons less than one year.”

Just prior to the onset of the 2001 recession, Chauvet and Potter (2005) extended the probit method to investigate the instability of the term spread’s predictive power and the existence of structural breaks. Using Bayesian techniques and several specifications of the probit method that variously allow for business-cycle dependence and autocorrelated errors, they found that recession forecasts from their more complicated extensions of the probit framework “are very different from the ones obtained from the standard probit specification.”

As the Great Recession loomed and as the yield curve was beginning to invert again for first time since the 2001 recession, Wright (2006) reexamined the standard probit specification, looking more closely at the policy instruments and other financial variables of interest to the Federal Open Market Committee. Around that same time, King, Levin, and Perli (2007) embedded credit spreads in the probit framework along with term spreads. They also incorporated Bayesian model averaging into the analysis and found that “optimal (Bayesian) model combination strongly dominates simple averaging of model forecasts in predicting recessions.”

More recently, Fornari and Lemke (2010) extended the probit approach by endogenizing the dynamics of the regressors using a vector autoregression and studied the United States, Germany, and Japan. Liu and Moench (2016) used the receiver-operating curve to assess the predictive performance of a number of previously proposed variables. Favara et al. (2016a, 2016b) decomposed credit spreads and showed that their power to predict recession is contained in a measure of investor risk appetite called the excess bond premium (EBP). Finally, Johansson and Meldrum (2018) used the principal components of the yield curve and a measure of term premiums to predict recession, and Engstrom and Sharpe (2020) investigated the forecast power of near-term forward term spreads.

The methods used in the research described to this point (e.g., probit regression, Markov switching, Bayesian techniques) are well established in the field of econometrics. In contrast, the methods that we use in this article have roots in statistics and computer science and—though used in many industrial applications—have only in recent years found application in macroeconomic analysis. Fornaro (2016) used large panels of predictors (several to hundreds) and added to the probit framework a Bayesian methodology with a shrinkage prior for the parameters to predict recession. Ng (2014) dispensed with the probit framework altogether and applied a tree ensemble classifier to a panel of 132 real and financial features and their lags to do so. In contrast to Ng, in this article we study a small panel of just three features and investigate the recession forecasting power of neural network classifiers vis-à-vis probit methods.

Holopainen and Sarlin (2017) studied many machine learning methods for the purpose of creating an early-warning/crisis detection mechanism for the Euro area. After conducting a horse race between the methods, their focus turned to model aggregation and ensemble voting techniques for producing forecasts from multiple classifiers. They also investigated the statistical significance of their results by means of bootstrap procedures. In contrast to that work, in this article we propose an NTS cross-validation procedure and explore strategies for contending with the
time-series properties of macro-financial panel data containing multiple structural breaks. Furthermore, we conduct statistical inference and compare classifiers by means of joint omnibus (Cochrane’s $Q$) and pairwise post hoc (McNemar’s) tests of significance, rather than via bootstrap methods. We also apply the Shapley additive explanations (SHAP) framework of Lundberg and Lee (2017) to investigate feature importance and decompose the recession forecasts of a neural network classifier.

Bluwstein et al. (2020) used a variety of machine learning methods to forecast financial crises over a long period of history and multiple countries, focusing on the Shapley value decomposition of their forecasts (Joseph 2019). They used a $k$-fold cross-validation strategy and found that the machine learning methods outperform logistic regression in out-of-sample prediction. In contrast, we propose a cross-validation strategy that respects the time ordering of the data at all points and arrive at a very different conclusion.

The empirical performance of machine learning methods on problems outside of the fields of finance and economics is well documented. Fernandez-Delgado et al. (2014) studied 179 binary classifiers on the 121 datasets in the UCI Machine Learning Data Repository. They found that, on average, random forest methods achieved the highest accuracy among all families of classifiers, followed by support vector machines and neural networks, and that in many cases the differences at the top of the list were not statistically significant. A similar study by Wainer (2016), using more robust hyperparameter searches, found random forest and gradient boosting to be two of the three top-performing methods for classification, with the differences between the two methods also not statistically significant. In this article, we study neural network classifiers only.¹

**DATA**

We use three high-frequency macro-financial time series in our study, two of which have not been used previously in the literature, although their low-frequency counterparts have. The three series separately capture (1) the shape of the yield curve, (2) the state of real business conditions, and (3) the state of financial conditions at each point in time in our sample.

For a measure of term spreads, or yield curve slope (term spread hereafter), we use the 10-year Treasury spot yield of Gürkaynak, Sack, and Wright (2007), less the three-month Treasury bill discount in the Federal Reserve’s H.15 series. These data are available at daily frequency. For the purposes of estimation and cross-validation, the monthly average of the series is used, whereas for the purposes of our out-of-sample forecasting exercise, the weekly average is used. The series is plotted in the top panel of Exhibit 1.

As a measure of real business conditions, we use the Federal Reserve Bank of Philadelphia’s Aruoba-Diebold-Scotti (ADS) Business Conditions Index (Aruoba, Diebold, and Scotti 2009). This series is highly correlated with quarterly returns in the Conference Board’s Leading Economic Index (LEI), which was previously studied by Estrella and Mishkin (1998) and Dueker (1997) in the context of the present problem and is composed of many of the same underlying macro series.² The ADS Index is available at weekly frequency, whereas the LEI is available only monthly at a lag of

---

¹An early version of this article studies tree ensemble methods as well. See Puglia and Tucker (2020).

²Specifically, the index is composed of weekly initial jobless claims, monthly payroll employment, industrial production, personal income less transfer payments, manufacturing trade and sales, and quarterly real gross domestic product (GDP).
The recession indicator used in this analysis is the same as that used in most of the existing literature on recession forecasting. For any given month, it is defined as true if any of the following 12 months falls within a recession, as defined by the National Bureau of Economic Research (NBER), and is false otherwise.\textsuperscript{3} The shaded

---

\textsuperscript{3}The Federal Reserve Bank of St. Louis NBER-Based Recession Indicators for the United States from the Peak through the Trough series is used.
regions in Exhibit 1 indicate NBER recession periods but not the recession indicator as we have defined it. All data used in this article cover the period from January 1973 through February 2020.

The unconditional probability of the indicator across the dataset (January 1973 through December 2019) is 25.2%, whereas the unconditional probability of recession in any given month is 13.8%. The static moments and monthly volatility of each series are summarized in Exhibit 2.

MODELS

In this article, we study three nested models. Denote the feature values at time \( t \) as \( \text{Term Spread}_t \), \( \text{NFCI}_t \), and \( \text{ADS}_t \). Furthermore, denote the response indicator, which takes a value of 1 if there is an NBER-dated recession in any of the 12 months following month \( t \) and 0 otherwise, as \( \text{NBER}_{t+1:T+12} \). Finally, designate \( \Phi() \) as the probit link function (the standard normal cumulative distribution) and \( \text{NN}() \) as the neural network classifier algorithm over a collection of features.

The first model is univariate in the term spread. The probability of the indicator conditional on term spreads for both the probit and neural network classifiers is given by

\[
Pr_{\text{Probit}}(\text{NBER}_{t+1:T+12} = 1) = \Phi(\alpha_0 + \alpha_1 \text{Term Spread}_t) \\
Pr_{\text{Neural Network}}(\text{NBER}_{t+1:T+12} = 1) = \text{NN}(\text{Term Spread}_t)
\]

Note that a constant is included when the probit method is used. In the second model, the term spread feature is combined with the ADS Index. The probability of the indicator conditional on the features for both the probit and neural network classifier is given by

\[
Pr_{\text{Probit}}(\text{NBER}_{t+1:T+12} = 1) = \Phi(\alpha_0 + \alpha_1 \text{Term Spread}_t + \alpha_2 \text{ADS}_t) \\
Pr_{\text{Neural Network}}(\text{NBER}_{t+1:T+12} = 1) = \text{NN}(\text{Term Spread}_t, \text{ADS}_t)
\]

The third and final model combines all three variables in similar fashion.

\[
Pr_{\text{Probit}}(\text{NBER}_{t+1:T+12} = 1) = \Phi(\alpha_0 + \alpha_1 \text{Term Spread}_t + \alpha_2 \text{ADS}_t + \alpha_3 \text{NFCI}_t) \\
Pr_{\text{Neural Network}}(\text{NBER}_{t+1:T+12} = 1) = \text{NN}(\text{Term Spread}_t, \text{ADS}_t, \text{NFCI}_t)
\]

NEURAL NETWORKS

We assume familiarity with probit regression. In this section, neural network classifiers are described and contrasted with probit regression.
Classifiers

Classification, in the statistical sense, is the problem of “identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.”

Although probit regression is not typically described as such in the field of econometrics, it too is an algorithm for classification. In essence, all classification algorithms are designed to map input data to a state probability, which can be used to produce point forecasts of that state (e.g., recession in the present application).

Artificial neural networks can also be used to build classification algorithms. They do so by approximating nonlinear classification functions through a collection of nodes organized into multiple layers. The universal approximation theorem states that a feed-forward neural network can approximate arbitrary real-valued continuous functions on compact subsets of \( \mathbb{R}^n \), provided the network is either sufficiently wide or sufficiently deep. Exhibit 3 depicts a simple example of a neural network classifier with three input vectors, two hidden layers with four nodes each, and a single output node. The number of layers in the network characterizes its depth, and the number of neurons in a layer characterizes its width.

The first layer, the input layer, passes the data vectors \( x_1, x_2, x_3 \) into hidden layer 1. Each node \( y_i \) in hidden layer 1 uses a weighting vector \( a_i \) (displayed as edges or arrows into the node) to linearly transform the information from the input layer and then employs a (possibly) nonlinear activation function \( f \) to construct the layer outputs at each node:

\[
y_i = f(a_{i0} + a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3).
\] (4)

Common choices of the activation function include the hyperbolic tangent function and the logistic function, among others. In this study, the rectified linear unit is used. This procedure continues to the right for each node in each hidden layer. The last layer of the neural network, the output layer, transforms the values from the final hidden layer into the output values of the network, which in the binary classification setting is a single node denoting the indicator class probability.

We use a relatively simple neural network architecture, consisting of just two hidden layers. We make no attempt to cross-validate over the number of nodes in each layer (or over more exotic architectures, such as recurrent or convolutional networks). The number of nodes used in the two layers is fixed at nine and five, respectively, in all analysis that follows.

Neural networks are trained on data (i.e., the weights are chosen) using back-propagation and gradient descent. Used together, these methods exploit the chain rule to update node weights in a neural network so as to minimize a loss function. In this study, cross-entropy loss is targeted. All of these details are beyond the scope of the present article, however. In what follows, all that needs to be understood is that a neural network classifier maps input data to a class probability, and the exact mechanisms by which this mapping happens can be black boxed and ignored.

NOTES: This is a simple feed-forward neural network shown for illustrative purposes. It contains two hidden layers, with four nodes each. Data from the leftmost layer, the input layer, are passed into the hidden layers, which transform the data. Values from the hidden layers are passed to the rightmost layer, the output layer, which is an indicator of class probability in classification problems. Bias (constant) nodes may also be employed but are omitted in the exhibit.
Comparison to Probit Regression

Our formal results are presented in a later section. The following is a qualitative discussion of some of the salient details of the one- and two-feature models that will be estimated. The aim is to compare probit regression to a neural network classifier and build intuition on the ways in which they differ.

One-feature term spread model. Beginning with the one-feature term spread model, Exhibit 4 shows the model-implied probability of the indicator as a function of term spreads for both the probit regression and the neural network classifier. Going forward we refer to this as the decision function, for the sake of simplicity. In addition to the probability distributions, the underlying data are plotted as well, with the nonrecession indicator shown in black (0) and the recession indicator shown in red (1).

NOTES: The probability of a recession in the following 12 months as a function of 3-month/10-year term spreads is displayed in the exhibit. Recession indicators code the occurrence of an NBER recession in the next 12 months in the sample data and are shaded black (0) and red (1).
The probability of a recession in the following 12 months as a function of 3-month/10-year term spread and the ADS Index is displayed in the exhibit. Indicators code the occurrence of an NBER recession in the next 12 months in the sample data and are shaded black (0) and red (1).

**EXHIBIT 5**
Two-Feature Term Spread/ADS Model—Probit Decision Function

**NOTES:** The probability of a recession in the following 12 months as a function of 3-month/10-year term spread and the ADS Index is displayed in the exhibit. Indicators code the occurrence of an NBER recession in the next 12 months in the sample data and are shaded black (0) and red (1).

Parametric assumptions. Furthermore, although the classifier probabilities track each other closely across the term spread domain, some important differences in the tails of the distribution can be seen. For example, at a term spread of −50 bps, the data suggest that the conditional probability of recession is 100% because all observations below this level of the term spread indicate recession. Looking to the model-implied probabilities, the probit classifier places the probability at only ~70%, however, which would appear to be low. In contrast, the neural network places the probability closer to 90%, which seems to fit the empirical distribution better. Looking to the other side of the distribution, the conditional probability of the indicator when term spreads are greater than 150 bps is about 11%. Again, the neural network implies a probability much closer to the empirically observed level, whereas the probit probability appears too high but also decays asymptotically to zero as term spreads rise. These results suggest that, although the point forecasts of these classifiers will not differ greatly, the flexibility of the neural network classifier allows it to capture the empirical distribution of the recession indicator in ways the probit regression cannot.

**Two-feature term spread/ADS model.** Next, we take up the two-feature term spread/ADS model. In Exhibit 5, term spreads are plotted on the x-axis and the ADS Index is plotted on the y-axis. The black dashed line is the probit isomorphic line of 50% probability, which is the decision boundary for this classifier in two dimensions.
The isomorphic lines for 25% and 75% probability are shown as well. As in the previous plot, the underlying data are scatter plotted, with the nonrecession points shown in black and the recession points in red. The observations are also labeled with the year in which they occurred.

Before turning to the neural network probabilities, we can first consider the probit decision function. First, we might question whether a linear separator (hyperplane) is the appropriate separator for these data. Our dataset is sparse, and although a linear separator admittedly performs well near the center of the mass around the 1991 and 2001 recessions, it is less clear that a linear separator would be appropriate in the regions where term spread is negative or ADS is below −1 or so. Certainly, there is no theoretical justification for extending the probit line to infinity in both directions. As an alternative to a linear separator, we posit that there exists a critical level of term spread below which the level of ADS ceases to matter. If that were so, we would consider the recession probability everywhere in the left half-plane defined by that critical term spread level to be very high. Conversely, if macroeconomic conditions are severe enough—as in the Great Recession—it could be argued that there ceases to be any level of term spread steep enough (or perhaps any monetary policy accommodative enough) to avert recession. As such, we would consider the probability of recession everywhere in the lower half-plane defined by that critical ADS level to be very high as well.

Next, we move to the neural network decision function in Exhibit 6. Here we have plotted the decision function in detail, color coding the model probabilities. Red shading denotes a degree of probability above 50%, and gray shading denotes a degree of probability below 50%. The white region separating the red and gray is the region of 50% probability and constitutes the decision boundary. The chart has been overlaid with the contents of Exhibit 5 for comparison to the probit classifier.

In the one-feature term spread model, there was little to distinguish the decision boundary of the probit classifier from that of the neural network; for both methods the boundary existed as a point in the feature space. Here, in two dimensions, the distinction becomes more obvious. Although the probit method linearly separates the data by drawing a line (hyperplane) through the feature space, the decision boundary for the neural network classifier captures the empirical distribution of the data in ways the probit regression cannot. Although both decision boundaries coincide in the center of the mass, the neural network curves horizontally in the fourth quadrant, and vertically in the first quadrant, which appeals to our economic intuition.

Notice also in the exhibit the two triangular regions of disagreement, in which the probit regression does not forecast recession and the neural network classifier does. There are very few observations in these regions, which makes statistical inference difficult. As in the one-feature model, the point forecasts of the probit and neural network classifiers are not statistically different from one another for this model. That said, it again appears that the flexibility of the neural network classifier allows it to capture the empirical distribution of the recession indicator in ways the probit regression cannot.

The three-feature model is difficult to visualize in this manner, so we refrain from doing so. In a later section, we present more detailed results on the forecast accuracy of neural networks and probit regression over all of these models.

**ECONOMETRIC METHODS**

In this section, we present a three-step econometric method for cross-validating and conducting statistical inference on machine learning classifiers and explaining their forecasts. The method is composed of (1) an NTS cross-validation strategy that
addresses the issues posed by sparse economic data when conducting econometric analysis using machine learning methods, (2) pairwise post hoc McNemar’s tests for selecting models and algorithms from many possible candidates, and (3) Shapley value decomposition of forecasts to aid in the economic interpretation of results. Our goal is to determine what, if any, value a neural network adds to our ability to forecast recession vis-à-vis the probit method.

The dataset we use presents several challenges that motivate our overall strategy. These challenges would be common to most any econometric analysis that uses machine learning methods on macro-financial panel datasets of monthly or quarterly frequency and can be summarized as follows:

- The dataset is serially correlated, which violates the independently and identically distributed (i.i.d.) assumption required to use many of the cross-validation methods most popular in the machine learning literature, such as k-fold, leave-p-out, and so on. Furthermore, because the data are time series and not cross-sectional in nature, attempting k-fold cross-validation on the dataset would result in data peeking and overly optimistic estimations of forecast performance.

**EXHIBIT 6**
Two-Feature Term Spread/ADS Model—Neural Network Decision Function

NOTES: The probability of a recession in the following 12 months as a function of 3-month/10-year term spread and the ADS Index is displayed. Indicators code the occurrence of an NBER recession in the next 12 months in the sample data and are shaded black (0) and red (1). Results from a probit classifier are overlaid for comparison.
• The dataset likely contains one if not multiple structural breaks (delineated by the Volcker Fed, the Great Moderation, and the Financial Crisis perhaps), further violating the i.i.d. assumption and making standard time-series cross-validation methods such as sliding or expanding windows problematic.

• The indicator (recession) is unbalanced in the dataset (unconditional mean of ~25%) and occurs temporally in clusters of varying length at the end of each business cycle, further complicating the use of standard time-series cross-validation methods such as sliding or expanding windows.

• The dataset is relatively short and sparse over the joint feature space.

NTS Cross-Validation

Cross-validation is a general technique for assessing how well the results of a statistical analysis will generalize to an independent dataset. It is mainly used in applications such as this, in which the goal is prediction and an estimate of how well a machine learning classifier or regression will perform in practice is desired. In the language of machine learning, a model is given known (in-sample) data on which training is run and a dataset of unknown (or out-of-sample) data against which testing is conducted. The training exercise is conducted to select the hyperparameters of the machine learning method being used, and the testing exercise is used to estimate the (out-of-sample) forecast performance. As mentioned earlier, strategies commonly used in cross-validation with machine learning classifiers include $k$-fold and leave-$p$-out.

Because our application does not lend itself nicely to established cross-validation strategies such as $k$-fold and leave-$p$-out, we propose a novel fourfold NTS cross-validation strategy to estimate the out-of-sample forecast performance of each of the six classifiers (three models by two algorithms) under study. Nested cross-validation is a strategy originally designed by Varma and Simon (2006) for use on small datasets to address the unique difficulties they pose in machine learning applications. According to Raschka (2018), nested cross-validation “shows a low bias in practice where reserving data for independent test sets is not feasible,” which is a desirable property given our dataset.

Because we are also working in a time-series setting, we augment the standard nested cross-validation strategy to incorporate several features that make it more amenable to conducting time-series analysis on the small macro-financial panel dataset used in this article. In particular, we overlay standard nested cross-validation with an expanding window so as to respect the time ordering of the data and prevent future leakage. We add one wrinkle to this feature in that, rather than forward chaining the window over a single data point or a fixed-size block of data points, we forward chain the outer loop of the NTS cross-validation over business cycles.

The NTS cross-validation procedure we use is most easily described by means of the pseudo-code algorithm in Exhibit 7.

To be more concrete, consider the top panel of Exhibit 8, in which we plot real GDP growth over our sample period, indicating recession periods in gray above the $x$-axis. Colored regions below the $x$-axis demarcate the individual business cycles that comprise the data; each colored region begins in a period of economic growth and ends when the NBER recession period following the period of growth terminates.

The bottom panel of Exhibit 8 shows each iteration of the NTS cross-validation outer loop in a row. Within each iteration of the outer loop, a stratified/shuffled threefold cross-validation is conducted over each training set (labeled Train) to determine optimal hyperparameters. The optimal hyperparameters are then used to estimate a classifier over the full training set, and the outer loop iteration is scored by calculating forecast performance (e.g., accuracy) over the test set (labeled Test) using
EXHIBIT 7

NTS Cross-Validation Algorithm

Function: nested_time_series_cv(model, algo, data, grid, metric)

Input: model (specification of features used to forecast recession)
       algo (e.g., classification algorithm, such as probit, neural network, random forest)
       data (dataset of feature and indicator observations stratified into $k = 7$ folds by business cycle)
       grid (discrete set in algo’s hyperparameter space to be searched for optimal performance)
       metric (forecast performance metric function and basis for model/algo comparison; e.g., accuracy, AUCROC)

Output: performance (forecast performance estimate for model/algo combination, in units of metric)

OUTER LOOP (expanding window cross-validation)

For $i = 3$ to $k - 1$ do

Hold out data fold $i + 1$ for use as an outer loop test set.
Use data folds 1 to $i$ as an outer loop training set.
Stratify the outer loop training set by fold, shuffle over the stratified classes, and divide into three folds of equal size.

HYPERPARAMETER GRID SEARCH

For each $j \in grid$ do

INNER LOOP (standard threefold stratified/shuffled cross-validation)

For $m = 1$ to $3$ do

Hold out fold $m$ from the outer loop training set as an inner loop test set.
Use the folds $\{n: n \in \{1,2,3\}, n \neq (m)\}$ of the outer loop training set as an inner loop training set.
Estimate a classifier for model using algo, the inner loop train set and hyperparameters $j$.
inner score $[m]$, calculate forecast performance for inner loop $m$ using metric and the inner loop test set.

End for

hyperparameter score $[j]$, average of inner score $[1:3]$ for hyperparameters $j$.

End for each

optimal hyperparameters $[i]$, element in grid with the highest hyperparameter score

Estimate a classifier for model using algo, the outer loop train set, and optimal hyperparameters $[i]$.
outer score $[i]$, calculate forecast performance for outer loop $i$ using metric and the outer loop test set.

End for

performance $\leftarrow$ average of outer scores $[3:k-1]$.

Return performance

the estimated classifier. When all four iterations of the outer loop are complete, the model/algorithm pair is scored by the average out-of-sample forecast performance over the four iterations.

The first three business cycles are used for training in the first iteration of the outer loop owing to their relatively short durations and to ensure a large-enough sample for classifier estimation. It is our opinion that the division of the data and the NTS cross-validation strategy described earlier represents the best balance that can be achieved between (1) the size of the training/test sets, (2) the sample size of the outer loop forecast accuracy estimate, and (3) indicator imbalance in the data. Although it allows hyperparameters to be chosen and out-of-sample forecast error to be estimated simultaneously, the NTS cross-validation strategy also

- avoids the creation of a test set to be held out entirely from training,
- mitigates class imbalance in the data through the use of stratification,
- mitigates structural breaks by chaining over business cycles,
- simplifies model diagnosis by mapping outer loop scores to business cycles, and
- avoids data peeking or future leakage.
Two main alternatives to this cross-validation strategy exist, but when applied to time-series data they become problematic in that they do not always respect the time ordering of data. The first alternative is a standard $k$-fold cross-validation strategy, originally developed for use on data that obey an i.i.d. assumption. This strategy is more appropriately applied to cross-sectional data and can result in overly optimistic estimates of forecast performance when applied to time-series data like ours.

The second alternative is a hybrid approach, somewhat between a standard $k$-fold cross-validation strategy and the NTS cross-validation strategy described earlier. Although it is generally accepted that the time ordering of data must be respected when estimating classifier forecast performance, it is less clear whether it is necessary to respect the time ordering of data when determining the optimal hyperparameters of the classifier algorithm. A strategy in which optimal hyperparameters are chosen through a strategy such as $k$-fold cross-validation on the entire dataset, but forecast performance is estimated using a rolling or expanding window strategy and the optimal hyperparameters determined by $k$-fold, would respect the time ordering of data during the latter half of the exercise but violate it during the former. Holopainen and Sarlin (2017) implemented a hybrid strategy similar to this and found ultimately that “conventional statistical approaches are outperformed by more advanced machine
learning methods.” In this article, we have used the NTS cross-validation strategy to preclude any violation of the time ordering of data during the study.

Cross-validation must target a forecast performance metric. We use accuracy as the target value over which out-of-sample forecast performance is measured and hyperparameters are selected. Our choice of forecast accuracy as a scoring metric within the NTS cross-validation procedure is motivated by several considerations. Although we would have preferred to use average precision (AP) of the precision-recall curve to score forecast performance, the last business cycle in the sample may not necessarily contain a positive instance of the indicator (recession). When this is the case, AP is ill-conditioned in that business cycle during NTS cross-validation. We also considered area under the receiver operating curve, but it is ill-conditioned like AP in some circumstances. In any case, Davis and Goadrich (2006) showed that, in this setting, the area under the receiver operating characteristic curve does not give a complete picture of a classification method’s performance.5 Because accuracy and Brier score are with over the last business cycle, they remain the best candidates for scoring forecast performance. For the present analysis we have chosen to use accuracy as the target metric.

The grid search that is used in the inner loop of the NTS cross-validation procedure is the principal means by which variance is traded against bias. (This applies only to the neural network classifier because no trade-off is necessary in the case of probit regression.) Because we fix our neural network architecture, as described earlier, our grid search is greatly simplified.

Statistical Inference via McNemar’s tests

After applying the NTS cross-validation strategy to our three models, two algorithms, and data, we will be left with six estimates of forecast performance across all combinations, from which we must choose that which works best for the problem at hand. In this article, we rely heavily on the strategies suggested by Raschka (2018) in doing so.

The first stage of our model comparison amounts to what is effectively an omnibus F-test over our classifiers. More specifically, we use a Cochrane’s Q test (Cochrane 1950), which tests the null hypothesis that there is no difference between the classifier forecast accuracy estimates. Consider the group of six classifiers. The null hypothesis $H_0$ is

$$H_0: A_{1,\phi} = A_{3,NN} = A_{2,\phi} = A_{2,NN} = A_{3,\phi} = A_{3,NN}$$  \(5\)

where $A_{x,y}$ is the forecast accuracy of the classifier estimated on model $x$, algorithm $y$. If the $m = 6$ classifiers do not differ in forecast performance, then Cochrane’s $Q$ statistic, defined in the following, will be distributed approximately $\chi^2$ with $m - 1$ degrees of freedom:

$$Q_x = (m - 1) \frac{\sum_{i=1}^m G_i^2 - T_x^2}{mT - \sum_{j=1}^n m_j^2}$$  \(6\)

where $n$ is the total number of observations in the dataset; $m_j$ is the number of classifiers out of $m$ that correctly classified the $j$th observation in the dataset; $c_{x,y,j}$ is 1 if the classifier for model $x$, algorithm $y$ correctly identified the $j$th observation and 0 otherwise; and

---

5Furthermore, they showed that any method that dominates in the precision-recall space also dominates in the receiver operating characteristic space.
If the null hypothesis is rejected, it suggests that at least one classifier is greatly outperforming or underperforming the others and that further investigation is warranted. In the second stage of our model comparison, we conduct pairwise post hoc tests, with adjustments for multiple comparisons, to determine where the differences occur. In particular, we use pairwise McNemar (1947) or \( \chi^2 \) within-subjects tests between all classifiers in the grouping, which is an analogue of a Diebold–Mariano test. Where a Diebold–Mariano test would be used in regression, with continuous variables, McNemar’s test is used in binary classification, where predictions take discrete values.

McNemar’s test begins by constructing a 2 × 2 contingency table for each pairing of our six classifiers:

<table>
<thead>
<tr>
<th></th>
<th>( c_2 ) correct</th>
<th>( c_2 ) incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 ) correct</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>( c_1 ) incorrect</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

In the contingency table, A is the number of observations out of \( n \) that both classifier \( c_1 \) and \( c_2 \) classified correctly. B is the number of observations that \( c_1 \) classified correctly and \( c_2 \) identified incorrectly, and vice versa for C. D is the number of observations that both classified incorrectly. Note that \( A + B + C + D = n \) and that the accuracy of each classifier is

\[
Accuracy_1 = \frac{A + B}{n} \quad (9)
\]

\[
Accuracy_2 = \frac{A + C}{n} \quad (10)
\]

Note also that the differences between the classifiers are captured by B and C, so these are the natural quantities of interest to the problem at hand. If the null hypothesis is that B and C are equal (i.e., both classifiers have the same rate of error), then McNemar’s statistic, with a continuity correction proposed by Edwards (1948), defined as follows, will be distributed \( \chi^2 \) with one degree of freedom:

\[
\text{McNemar’s } \chi^2 = \frac{(B - C - 1)^2}{B + C} \quad (11)
\]

If the null hypothesis is rejected after application of McNemar’s test, we may conclude that the classifier with the higher forecast accuracy estimate outperforms the other and that the difference is statistically significant.

**Forecast Decomposition using Shapley Values**

After the forecast performances of all models and algorithms have been determined in NTS cross-validation and weighed against one another by means of the
The omnibus Cochrane’s $Q$ and pairwise McNemar tests presented in the previous sections, it is natural to ask which of the features under study contributed to the overall performance of the best classifier and how. In this section we outline our methods for doing so.

At this point of the analysis, we dispense with NTS cross-validation and calculate our feature importance metrics in-sample. That is, we use the entire dataset as a training sample, use $k$-fold cross-validation to determine optimal hyperparameters, estimate classifiers using the optimal hyperparameters, and then calculate feature importances using the estimated classifiers.

Feature importance metrics fall into two broad categories: those that are specific to only one or a few algorithms and those that can be applied broadly to all algorithms. In the probit literature referenced earlier, it is common practice to report the sensitivity or marginal effects of an estimated model as a measure of variable importance. This is an example of the former type of feature importance because it relies on the parametric assumptions of the probit method and cannot be applied directly to machine learning methods. There are many examples of the latter type of metric. Perhaps the most commonly used is permutation importance, or mean decrease in accuracy (Breiman 2001). For the sake of brevity, we omit its usage here.

Shapley value decomposition can be used to produce both global and local feature importances. Global feature importances score the value of a feature to a classifier as a whole through a single summary metric; no further decomposition of individual predictions is possible under this class of feature importance methods. (Probit sensitivities and mean decrease in accuracy are examples of global feature importance metrics.) If feature attribution at the level of a single observation is desired, a local method is required.

The SHAP framework of Lundberg and Lee (2017) provides an example of a local metric that can also be used to construct a global metric. Although other local feature attribution methods exist, such as LIME (Reibero, Singh, and Guerstrin 2016) and DeepLIFT (Shrikumar, Greenside, and Anshul 2017), Lundberg and Lee demonstrated that SHAP unifies these approaches and others under a more general framework for decomposing individual predictions and attributing feature importance.

At the core of the SHAP framework, the Shapley value is a concept from coalitional game theory that prescribes how gains or payouts resulting from a cooperative game should be divided among players in proportion to their contribution to the outcome. In the present case, the players are the features in the dataset, and the payout is the forecast probability output by the classifier given the observations at time $t$. More technically, consider the set $f$ of $m$ features in the dataset at time $t$ and a classifier prediction function $v$ that maps the feature values at $t$ to a probability forecast $v(f) : 2^m \rightarrow P$. If $s$ is a coalition or subset of features $s \subseteq f$, then $v(s)$ describes the worth of coalition $s$, or the total payout (i.e., probability) that the members of $s$ can expect to receive by working together.

The Shapley value suggests a way to distribute the total forecast probability to the grand coalition of all features $f$ after considering all possible coalitions and their individual worths. The Shapley value, which in the present case is the amount of probability apportioned to feature $i$ in $f$ at $t$, is given by

$$
\varphi_i(v) = \sum_{s \subseteq f \setminus \{i\}} \frac{|s|!(m-|s|-1)!}{m!} (v(s \cup i) - v(s))
$$

The second term inside the summation can be interpreted as the amount of payout fairly credited to feature $i$ if $i$ were to join coalition $s$. The summation is conducted

---

6 The payout can also be interpreted as the forecast probability of the classifier in excess of the naïve or (unconditional) mean forecast probability of the classifier over all observations.
over all subsets $s$ of $f$ not containing feature $i$, or rather all permutations of coalitions $s$ that $i$ does not belong to but could join. In effect, feature $i$’s total payout is the average of its payouts over all those permutations.

The Shapley value is considered to be a fair attribution of the total probability in the sense that it possesses a few desirable properties. Those pertinent to the present discussion are summarized as follows:

- **Efficiency**—The Shapley values for all features in the set $f$ add up to the total payout $\nu$. That is, $\sum_{i} \phi_{i} = \nu(f)$. In other words, the Shapley values for the features $f$ at time $t$ sum to the probability forecast of the classifier given the observation at time $t$.

- **Symmetry**—If two features contribute equally to all permutations of coalitions, then their Shapley values are equal. That is, if features $i$ and $j$ satisfy $\nu(s \cup \{i\}) = \nu(s \cup \{j\})$ for every coalition $s \subset f \setminus \{i,j\}$, then $\phi_{i}(\nu) = \phi_{j}(\nu)$.

- **Null Player**—If a feature cannot contribute to any coalition to which it is not a member, then its Shapley value is zero. That is, if $\nu(s \cup \{i\}) = \nu(s)$ for every coalition $s \subset f \setminus \{i\}$, then $\phi_{i}(\nu) = 0$.

Although these ideas form the basis of the SHAP framework, direct implementation of Shapley values in a machine learning setting is generally not feasible because it may require the classifier to be retrained on all the permutations of $s \subset f \setminus \{i\}$ for each feature $i$ in the dataset (as well as all observations $t$). To apply Shapley values in practice, an approximation is required. The SHAP framework unifies several local Shapley value approximation methods under a more general conditional expectation function of an abstract machine learning model. In general, each of these local approximation methods maps to a single class of machine learning algorithms. The SHAP methods relevant to the present discussion and the algorithms they map to are as follows:

- **DeepSHAP**—SHAP value approximations for deep neural networks. This method wraps DeepLIFT (Shrikumar, Greenside, and Anshul 2017). Although we study neural network classifiers in this article, the DeepSHAP framework will not integrate with our implementation, so we use the more generally applicable KernelSHAP for neural network SHAP value decomposition.

- **KernelSHAP**—SHAP value approximations for general functions. This method uses LIME (Reibero, Singh, and Guestrin 2016) to locally approximate any classifier function. We apply KernelSHAP to the neural network classifiers in this article.

### RESULTS

**NTS Cross-Validation Results**

Exhibit 9 displays the quantitative results of NTS cross-validation for all three models that we study, across both the probit and neural network algorithms. The rows are sorted in order of decreasing mean forecast accuracy, and standard errors are shown in parentheses. According to the exhibit, the probit classifiers for the two- and three-feature models appear to have performance superior to all of the neural network classifiers, before considering the statistical significance of the results.

---

7 DeepSHAP integrates with neural networks built in Tensorflow/Keras and PyTorch but not scikit-learn. We have used scikit-learn.
Exhibit 10 plots the data in Exhibit 9. Before turning to statistical inference, we may hypothesize, judging from the exhibit, that the two-feature model performs better than the one-feature model, but the three-feature model does not necessarily outperform the two-feature model. Furthermore, the differences in the probit and neural network classifier performance for any given model do not seem to be very meaningful. Only for the three-feature model does there appear to be, perhaps, a statistically significant difference between the methods.

McNemar’s Tests

After conducting a Cochrane’s Q test on the six classifiers in the preceding, the null hypothesis is rejected at the 1% level. The $p$-value for the test is essentially zero. According to the method outlined in the previous section, this justifies our use of pairwise post hoc McNemar’s tests to investigate the existence of out- or underperformance among the classifiers.

The results of the pairwise post hoc McNemar’s tests are shown in Exhibit 11. For each pairing of the six classifiers, the $p$-value of the test is indicated. Asterisks are used to denote rejection of the null hypothesis at the 1% and 10% levels.

The top two rows indicate that the two one-feature classifiers differ significantly from all other classifiers but not from each other. However, the third and fourth rows of the matrix indicate that the two-feature classifiers do not differ significantly from the three-feature classifiers. This lends some support to our hypothesis that the two-feature model outperforms the one-feature model, but the three-feature model does not outperform the two-feature model. Furthermore, for any of the three models, the probit and the neural network classifier performance is not statistically different at the 1% level. Only for the three-feature model do we observe something

### Exhibit 9
NTS Cross-Validation Mean Forecast Accuracy by Model/Algorithm

<table>
<thead>
<tr>
<th>Model</th>
<th>Algorithm</th>
<th>Mean Forecast Accuracy (SE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term Spread/ADS/NFCI</td>
<td>Probit</td>
<td>0.911 (0.037)</td>
</tr>
<tr>
<td>Term Spread/ADS</td>
<td>Probit</td>
<td>0.908 (0.039)</td>
</tr>
<tr>
<td>Term Spread/ADS</td>
<td>Neural Network</td>
<td>0.893 (0.041)</td>
</tr>
<tr>
<td>Term Spread/ADS/NFCI</td>
<td>Neural Network</td>
<td>0.891 (0.041)</td>
</tr>
<tr>
<td>Term Spread</td>
<td>Probit</td>
<td>0.814 (0.078)</td>
</tr>
<tr>
<td>Term Spread</td>
<td>Neural Network</td>
<td>0.805 (0.081)</td>
</tr>
</tbody>
</table>

**NOTES:** Mean out-of-sample forecast accuracy across all models and algorithms, using NTS cross-validation. Standard errors of the mean accuracy estimate are in parentheses.

### Exhibit 10
NTS Cross-Validation Mean Forecast Accuracy by Model/Algorithm

**NOTE:** Mean out-of-sample forecast accuracy across all models and algorithms, using NTS cross-validation.
Neural Networks, the Treasury Yield Curve, and Recession Forecasting

approaching statistical significance, but it is weak at that. In summary, it would appear that although the probit classifier accuracy estimates are everywhere higher than the neural network estimates, they are not meaningfully higher.

Shapley Value Decomposition

In closing this section, a few feature importances are presented. Recall that, having estimated the forecast accuracies of the classifiers, we dispense with NTS cross-validation from this point forward and use classifiers that have been trained on all of the data, so all results are in-sample.

Probit sensitivities are shown in Exhibit 12. As would be expected, the term spread coefficient is strongly significant, and the marginal effect is consistent with previously published estimates. The coefficients for ADS and NFCI are also very significant, and these features absorb some of the marginal effect of the term spread. The \( R^2 \) values of the regressions grow rapidly as features are added to the univariate model.

Mean absolute SHAP values for each model and classifier are shown in Exhibit 13. This global measure of feature importance allows us to compare the results of probit regression to that of a neural network classifier. The values are generally consistent across both algorithms and with the rank ordering of the probit sensitivities shown in Exhibit 12. Term spreads have the highest importance across all models, followed by the ADS Index and NFCI. The latter two features show some ability to absorb the power of term spreads.

The SHAP values can also be used to compare feature importance across the probit and neural network algorithms locally (i.e., observation by observation). Exhibit 14 scatterplots the term spread SHAP values for each observation against the level of the term spread for the three-feature model. The two scatters more or less coincide.

Hyperparameters for the machine learning classifiers have been chosen through \( k \)-fold cross-validation conducted on the entire dataset.

<table>
<thead>
<tr>
<th>No. of Features</th>
<th>Model Algorithm</th>
<th>1 Term Spread Probit</th>
<th>2 TS/ADS NN</th>
<th>2 TS/ADS Probit</th>
<th>3 TS/ADS/NFCI NN</th>
<th>3 TS/ADS/NFCI Probit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Term Spread Probit</td>
<td>0.29</td>
<td>0.00**</td>
<td>0.00**</td>
<td>0.00**</td>
<td>0.00**</td>
</tr>
<tr>
<td>1</td>
<td>Term Spread NN</td>
<td>0.00**</td>
<td>0.00**</td>
<td>0.00**</td>
<td>0.00**</td>
<td>0.00**</td>
</tr>
<tr>
<td>2</td>
<td>TS/ADS NN</td>
<td>0.39</td>
<td>0.86</td>
<td>0.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TS/ADS Probit</td>
<td>0.31</td>
<td></td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TS/ADS/NFCI NN</td>
<td></td>
<td></td>
<td>0.10*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES: This exhibit displays \( p \)-values from pairwise McNemar tests conducted between all pairs of six classifiers. *\( p < 0.1 \); **\( p < 0.01 \).
EXHIBIT 12
Probit Sensitivities

<table>
<thead>
<tr>
<th>Model/Feature</th>
<th>Term Spread</th>
<th>ADS</th>
<th>NFCI</th>
<th>McFadden</th>
<th>McKelvey–Zavoina</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Term Spread</td>
<td>–0.145 (0.010)**</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Term Spread/ADS</td>
<td>–0.125 (0.010)**</td>
<td>–0.198 (0.017)**</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-Term Spread/ADS/NFCI</td>
<td>–0.090 (0.011)**</td>
<td>–0.136 (0.018)**</td>
<td>0.111 (0.022)**</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:** The average sensitivity of the probit classifier probabilities to the features at the estimated coefficients over all observations is reported in the exhibit. Standard errors are in parentheses and calculated using a bootstrap (n = 564 observations). The McFadden and McKelvey–Zavoina $R^2$ are shown in the right columns. **$p < 0.01.$

EXHIBIT 13
Mean Absolute SHAP values

<table>
<thead>
<tr>
<th>Feature</th>
<th>1-Term Spread</th>
<th>2-Term Spread/ADS</th>
<th>3-Term Spread/ADS/NFCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model/Algo</td>
<td>Probit</td>
<td>Neural Net</td>
<td>Probit</td>
</tr>
<tr>
<td>Term Spread</td>
<td>18.6</td>
<td>20.2</td>
<td>18.2</td>
</tr>
<tr>
<td>ADS</td>
<td>15.1</td>
<td>16.0</td>
<td></td>
</tr>
<tr>
<td>NFCI</td>
<td></td>
<td></td>
<td>10.9</td>
</tr>
</tbody>
</table>

**NOTES:** Mean absolute SHAP values over the entire dataset are reported in the exhibit for each feature, model, and algorithm. All values are in percent.

EXHIBIT 14
Scatterplot of Term Spread SHAP Values for the Three-Feature Model

**NOTE:** The term spread SHAP values for the three-feature term spread/ADS/NFCI model are scatter plotted against the level of the term spread for both the probit and neural network classifier.
for positive term spreads. When term spreads are negative, however, the ability of the neural network to capture a nonlinearity in the relationship becomes visible.

Exhibit 15 is a similar plot for the ADS Index. A nonlinear relationship is visible again in the neural network classifier. The classifiers coincide roughly when the ADS Index is below 0. For positive values of the ADS Index, the neural network SHAP values are roughly constant, whereas for the probit classifier they fall as the ADS Index rises.

Finally, Exhibit 16 shows a SHAP value scatterplot for the NFCI in the three-feature model. Similarly, the nonlinearity of the neural network classifier is visible across the NFCI axis.

To summarize our conclusions, we show that the neural network classifier does not outperform the probit regression for any of the three models we study, according to the results of NTS cross-validation and pairwise post hoc McNemar’s tests on the six classifiers. This stands in contrast to a body of research documenting the superior performance of machine learning methods compared with other algorithms, including probit regression. We believe this is a consequence of the relatively small, sparse panel of macro-financial variables that has been used.

Across both algorithms, the McNemar’s tests also show that the two-feature model clearly outperforms the one-feature model, but the three-feature model does not outperform the two-feature model when measuring on the basis of forecast accuracy. That said, as we showed in a previous section, the flexibility of the neural network classifiers allows them to capture the empirical distribution of the recession indicator in ways a probit classifier cannot. By Shapley value decomposition, we have also shown in the scatterplots that the neural network classifier is able to map a nonlinear relationship between the data and the recession indicator probability in ways the probit regression cannot.

Our variable importance measure—mean absolute SHAP values—indicates that term spreads have a high degree of explanatory power, which is consistent with the literature. Adding the ADS Index to the model greatly improves forecast performance,
and the importance measure shows that this feature has a high degree of explanatory power as well. The NFCI has the lowest mean absolute SHAP value in the three-feature model, but it is only slightly lower than the ADS Index.

**DISCUSSION**

In this section, we briefly present a policy application of Shapley value decomposition in the context of recession forecasting. As mentioned in the third section, all three of the features we use are available at high frequency, which permits us to produce something close to real-time forecasts of recession.

Exhibit 17 displays the SHAP value decomposition of the out-of-sample forecast of the three-feature neural network classifier. The data are weekly and cover March 2019 to February 2020. We have chosen to conduct the policy application using this classifier, although it did not rank highest in forecast accuracy among the alternatives, because it did not underperform the probit alternative (on the basis of forecast accuracy) and because we believe it captures features of the distribution of recession over the data that the probit alternative cannot.

Note first that the baseline Shapley value is constant across the sample (~25%) and denotes the unconditional probability of the indicator in the data, or the naïve forecast. The other three colored regions represent the Shapley values for the features, and at each point in time the Shapley values sum to the recession forecast probability (black line). The term spread Shapley value (red) is everywhere positive in the exhibit, meaning that term spreads were low over the period, which raised the probability of recession. In contrast, the NFCI Shapley value (purple) is everywhere negative, meaning that financial conditions were loose, lowering the recession probability. In between these extremes, the ADS Index Shapley value oscillates around zero, meaning that real business conditions were at times slightly loose and at other times slightly tight over the period.
The three cuts to the federal funds target rate made by the Federal Open Market Committee in 2019 are displayed as black vertical lines. There appears to be a relationship between these cuts and changes in the model-implied recession probabilities. To the extent that policymakers have policy instruments that can influence the model inputs, this Shapley value decomposition allows us to map the probability of recession to the tools of monetary policy.

Exhibit 18 displays the same information from 1987 to present, the post-Volcker period. All data are monthly, rather than weekly, and the pre-2019 forecasts are in sample. NBER recessions are shaded in gray. It is clear in this figure that the term spread Shapley values were elevated in the period preceding each of the last four recessions, which confirms the power of term spreads to predict recession. In every instance that the yield curve first inverted during these pre-recession periods, new market commentary and articles in the financial press asked whether this time would be different, questioning whether the inverted yield curve would correctly forecast another recession. Shapley value compositions provide a tool for exploring this question more methodically. In the exhibit, it appears that the Shapley values for at least two of the three features used will become elevated before recession begins. If this relationship or another like it exists, it suggests a role for Shapley value decomposition in early-warning mechanisms.
CONCLUSION

We have investigated the performance of neural network classifiers vis-à-vis probit regression when forecasting US recessions using term spreads and other macro-financial data. In doing so, we proposed a novel three-step econometric method for cross-validating and conducting statistical inference on machine learning classifiers and explaining forecasts. The method is composed of (1) an NTS cross-validation strategy that addresses the issues posed by sparse economic data when conducting econometric analysis using machine learning classifiers, (2) pairwise post hoc McNemar’s tests for selecting models and algorithms from many possible candidates, and (3) Shapley value decomposition of forecasts to aid in the economic interpretation of results.

We find that probit regression does not underperform a neural network classifier in the present application, which stands in contrast to a growing body of literature demonstrating that machine learning methods outperform alternative classification algorithms. That said, neural network classifiers do identify important features of the joint distribution of recession over term spreads and other macro-financial variables that probit regression cannot. We discussed some possible reasons for our results and used our procedure to study US recessions over the post-Volcker period.
REFERENCES


Bluwinstein, K., M. Buckmann, A. Joseph, M. Kang, S. Kapadia, and Ö. Simsek. “Credit Growth, the 
Yield Curve and Financial Crisis Prediction: Evidence from a Machine Learning Approach.” Working 


(3–4): 256–266.

Davis, J., and M. Goadrich. “The Relationship between Precision-Recall and ROC Curves.” In 


1–6.


Favara, G., S. Gilchrist, K. F. Lewis, and E. Zakrjasek. “Recession Risk and the Excess Bond Pre-
mium.” FEDS notes, Board of Governors of the Federal Reserve System, 2016a.

———. “Updating the Recession Risk and the Excess Bond Premium.” FEDS notes, Board of 
Governors of the Federal Reserve System, 2016b.

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 

Fornari, F., and W. Lemke. “Predicting Recession Probabilities with Financial Variables over Multiple 

Fornaro, P. 2016. “Forecasting US Recessions with a Large Set of Predictors.” Journal of Fore-


To order reprints of this article, please contact David Rowe at d.rowe@pageantmedia.com or 646-891-2157.