Bond Risk Premia with Machine Learning[*]

Daniele Bianchi[†] Matthias Büchner[‡] Andrea Tamoni[§]

First draft: December 2017. This draft: November 15, 2018

Abstract

We propose, compare, and evaluate a variety of machine learning methods for bond return predictability in the context of regression-based forecasting and contribute to a growing literature that aims to understand the usefulness of machine learning in empirical asset pricing. The main results show that non-linear methods can be highly useful for the out-of-sample prediction of bond excess returns compared to benchmarking data compression techniques such as linear principal component regressions. Also, the empirical evidence show that macroeconomic information has substantial incremental out-of-sample forecasting power for bond excess returns across maturities, especially when complex non-linear features are introduced via ensembled deep neural networks.

Keywords: Machine Learning, Deep Neural Networks, Forecasting, Bond Returns Predictability, Empirical Asset Pricing, Ensembled Networks.

JEL codes: C38, C45, C53, E43, G12, G17.

[*]We are grateful to Bryan Kelly, Marcin Zamojski and participants at the workshop "Predicting Asset Returns" in Örebro for useful comments and suggestions. We thank the Centre for Scientific Computing at the University of Warwick for support with the super-computing clusters.

[†]Warwick Business School, University of Warwick, Scarman Road, Coventry CV4 7AL, UK. E-mail: Daniele.Bianchi@wbs.ac.uk Web: whitesphd.com

[‡]Warwick Business School, University of Warwick, Scarman Road, Coventry CV4 7AL, UK. E-mail: matthias.buechner.16@mail.wbs.ac.uk Web: http://bitly.com/2kZFenc

[§]Department of Finance, London School of Economics and Political Science, Houghton Street, London WC2A 2AE, UK. E-mail: a.g.tamoni@lse.ac.uk Web: http://www.lse.ac.uk/finance/people/profiles/AndreaTamoni.aspx

1 Introduction

The recent advancements in the field of econometrics, statistics, and computer science have spurred the interest in dimensionality reduction and model selection techniques as well as predictive models with complex features such as sparsity and non-linearity, both in finance and economics.[1] Over the last two decades, however, the use of such methods in the financial economics literature was mostly limited to data compression techniques such as principal component and latent factor analysis.[2] A likely explanation for the slow adoption of advances in statistical learning is that these methods are not suitable for structural analysis and parameters inference (see Mullainathan and Spiess, 2017). Indeed, machine learning methods are primarily focused on prediction, that is to produce the best out-of-sample forecast of a quantity of interest based on some conditioning information.

The suitability of machine learning methodologies for predictive analysis makes them particularly attractive in the context of financial asset returns predictability and risk premia measurement (see, e.g., Gu et al., 2018). As a matter of fact, while many problems in economics rely on the identification of primitive underlying shocks and structural parameters, the quantification of time variation in expected returns is intimately a forecasting problem. This practical view complements the theory-driven approach which often provides the building blocks for the empirical analysis of financial markets. Modeling the predictable variation in Treasury bond returns, which is the focus of this paper, provides a case in point. Theory provides guidelines on what variables can be plausibly considered as "predictors", such as past yields, macroeconomic information, or both. However, the actual functional form of the mapping between the predictors and future bond excess returns is left unspecified a priori (see, e.g., Duffee, 2013).

In this paper we propose, compare, and evaluate a variety of machine learning methods for the prediction of Treasury bond excess returns within a regression-based context. The research design follows the structure outlined in Gu et al. (2018), whereby a comparison of different machine learning

[1]See, e.g., Rapach et al. (2013), Feng, Giglio and Xiu (2017), Freyberger, Neuhierl and Weber (2017), Giannone, Lenza and Primiceri (2017), Giglio and Xiu (2017), Heaton, Polson and Witte (2017), Kozak, Nagel and Santosh (2017), Feng, Polson and Xu (2018), Bianchi, Billio, Casarin and Guidolin (2018), Gu, Kelly and Xiu (2018), Kelly, Pruitt and Su (2018), and Sirignano, Sadhwani and Giesecke (2018).

[2]In economics, the initial idea of compression technique can be probably traced back to Burns and Mitchell (1946) who argue for a business cycle indicator that is common across macroeconomic time series. This idea was formally modeled by Geweke (1977) and Sargent and Sims (1977) who provide evidence in favor of reducing the number of predictors. Since then principal component analysis and factor analysis have been widely adopted in financial economics for forecasting problems involving many predictors. Prominent examples have been provided by Stock and Watson (2002a; 2002b; 2006), Forni and Reichlin (1996, 1998), Bai and Ng (2003, 2006, 2008), De Mol et al. (2008), and Boivin and Ng (2006a), among others.

techniques is based on their out-of-sample predictive performance. Our contribution is threefold: first, we show empirically that machine learning algorithms, and neural networks in particular, are useful to detect predictable variations in bond returns, as indicated by the higher out-of-sample predictive R^2s relative to benchmark data compression (e.g. linear combinations of forward rates as in Cochrane and Piazzesi, 2005, and factors extracted from a large panel of macroeconomic variables as in Ludvigson and Ng, 2009) and penalized regression techniques. Our out-of-sample evidence suggests that the economic significance of violations of the Expectations Hypothesis may be large, and challenges some recent evidence based on classical linear predictive system (see, e.g., Thornton and Valente, 2012). Second, we show that macroeconomic information has substantial out-of-sample forecasting power for bond excess returns across maturities, especially when complex non-linear features are introduced via deep neural networks. Our evidence suggests that non-linear combinations of macroeconomic variables contain information about future interest rates that is not entirely spanned by the current term structure. Third, we provide evidence that the predictability of future short-term bond returns is primarily driven by financial variables, within the context of neural networks. In particular, a sensitivity analysis, based on the partial derivative of future bond returns with respect to each predictor, shows that, for e.g., the S&P composite index and the effective federal funds rate rank at the top among the most relevant predictors. However, notice that the composition of the set of best predictors turns out to be quite heterogeneous over the term structure.

The implications of using machine learning methodologies for bond returns predictability are far from trivial. Forecasting bond excess returns requires a careful approximation of the a priori unknown mapping between the investors' information set, the one-period yields and excess bond returns (see, e.g., Duffee, 2013, p. 391-392). By using machine learning methodologies, one can agnostically investigate the properties of such mapping at a general level via regression-based predictive analysis.

In the empirical analysis we investigate the out-of-sample performance of a variety of machine learning techniques for forecasting Treasury bond excess returns for different maturities. In particular, we consider a set of candidate methodologies including standard linear least squares estimates, penalized linear regressions, partial least squares, regression trees, random forests, and neural networks. All these methods fall under the heading of "supervised learning" in the computer science literature.[3] Although not exhaustive, this list arguably covers the spectrum of modern statistical

[3]The main difference between a "supervised" and an "unsupervised" statistical learning is that the former explicitly employs the information embedded in the target variable to summarize the information of the inputs. That is, the mapping between the quantity of interest y and the predictors $\boldsymbol{x}$ is learned by using information on the joint distribution.

learning techniques (see, e.g., Friedman et al., 2001).[4] These methods are evaluated in contrast to traditional dimensionality reduction techniques such as Principal Component Analysis (PCA), which arguably represents an almost universal approach to regression-based forecasting of Treasury bond returns (see, e.g., Duffee, 2013). In addition, factor models have been widely shown to be a benchmark that is hard to beat in terms of forecasting power (see, e.g., Stock and Watson, 2002a,b).[5]

Our set of empirical results shows that machine learning methods reliably detect movements in expected bond returns. More specifically, within the context of two traditional research designs – one that exploits information only in yields as in Cochrane and Piazzesi (2005), and one that makes also use of information from a large dataset of hundreds of macroeconomic indicators as in Ludvigson and Ng (2009) – we provide evidence that a deep neural network attains positive out-of-sample R^2 and significantly outperforms both the benchmark PCA forecasts and the alternative supervised learning methods – these methods displaying out-of-sample predictive R^2 which are in negative territory. In addition, the empirical evidence unequivocally show that the information embedded in macroeconomic variables is not subsumed by the yield curve.

Delving further into the comparison of performance across methodologies, a clear pecking order emerges: linear penalized regressions, such as the Lasso originally proposed by Tibshirani (1996) and the "Elastic Net" proposed by Zou and Hastie (2005b), achieve a better out-of-sample performance than traditional PCA regressions. This is particularly true when only information in the current term structure is used to measure bond expected returns. Yet, allowing for non-linearities substantially improves the out-of-sample performance, especially for longer maturity bonds. We find that a neural network with more than two layers unambiguously improves the prediction of one-year holding period bond excess returns with predictive R^2 being several orders of magnitude higher than linear penalized regressions and PCA forecasts.[6] This result suggests that a universal functional approximator like

Unsupervised learning is normally implemented for data compression, e.g. PCA, and does not explicitly condition on the quantity of interest y to summarize the information content in $\boldsymbol{x}$.

[4]Other methodologies such as reduced rank regression, project pursuit regression, sliced inverse regressions, support vector machines, and linear discriminant analysis could also be considered. Such methods can be classified as "shallow" learners comparable to more traditional penalized regressions and principal component analysis (see Ripley, 1994, Friedman et al., 2001 and Polson and Sokolov, 2017). For this reason we limit the empirical application to the use of penalized regressions, principal component regressions and partial least squares.

[5]Interestingly, PCA can be interpreted as a particular type of neural network structure called Autoencoder. This allows to re-frame PCA within a typical shallow learning framework and to isolate the pure contribution of using the response variable, i.e., the bond excess returns, in learning the mapping between yields, macroeconomic information and expected bond returns.

[6]Notice that throughout the paper we adopt the convention of numbering the layers including also the output layer. For instance, a four-layer neural network entails three hidden layers with non-linear activation functions and an output layer. Similarly, a two-layer network is made by a single hidden layer in addition to the output layer. An alternative

neural networks may be effective in determining the mapping between current yields, macroeconomic variables, and expected bond returns. This finding is consistent with the evidence in Gu et al., 2018 on stock returns. When considering a range of different neural network specifications from shallow (one hidden layer) to deeper networks (up to three hidden layers), we find that the out-of-sample predictive R^2 increases almost monotonically. Again, this is consistent with Gu et al. (2018), who show in the context of stock returns that the performance of a neural network peaks for a three hidden-layer specification and then deteriorates. A pairwise Diebold and Mariano (1995) test confirms the statistical significance of the outperformance of penalized regressions with respect to traditional PCAs and, in turn, of neural networks with respect to penalized regressions.

At a broad level, the empirical results remain intact across different subsamples. The inclusion of the period after 2008/12, which includes unconventional monetary policies and interest rates hitting the zero-lower bound, substantially deteriorates the out-of-sample performance of all methodologies. However, in line with the recent literature (see e.g. Fernández-Villaverde et al., 2015) that argues for the importance of acknowledging the role of nonlinearities in analyzing the dynamics of nominal interest rates when a zero lower bound is included, we do find that the predictive R^2 remains largely in favor of a non-linear deep neural network specification with three hidden layers. Also, when we predict bond returns using only yields, we find evidence that a deeper network is needed to better proxy for the non linear mapping between bond returns and yields during the full sample. Indeed, when the financial crisis is excluded, a NN with three layers (and five nodes) achieves performance that is on par with or better than that of a NN with four layers (and pyramidal nodes). On the other hand, when the crisis is included, this order is reversed; in this case the NN with four layers (and pyramidal nodes) achieves performance that are at par or better than a NN with three layers throughout the maturity structure. Overall, our results suggest that the success of neural networks is largely due to their ability to capture complex non-linearities in the data, and the "right" depth of the network may vary with the severity of these non-linear relations.

One additional contribution of this paper is to attempt an analysis of the drivers of the outperformance of deep neural networks. To this end, within the exercise that relies only on information contained in term structure of interest rates, we investigate the incremental contribution of deep networks in forecasting the changes in the first three principal components of the covariance matrix of yields. An uncontroversial result of the term structure literature is that the first three principal

convention is followed by, e.g., Feng et al. (2018) who count only the hidden layers.

components – dubbed level, slope and curvature – summarize almost 99% of the information in the cross-section of yields. As a result, by investigating the ability of neural networks to forecasts these three principal components one can shed some light on the origins of the increasing out-of-sample predictability. For instance, if neural networks help forecasting only the third principal component one can conclude that their out-of-sample outperformance is primarily due to a better forecasting of the term structure curvature. Our evidence suggests that - when using yields only as predictors - the neural networks improve primarily the forecast of the level of the term structure while the incremental contribution for the slope and the curvature is much more modest. However, when we turn to the forecasting exercise that exploits both macroeconomic and financial information in addition to yields, we find that the factors extracted from the neural networks not only contribute to the ability to predict the level of the yield curve, but also the slope. This is consistent with the idea that the slope of the yield curve is related to the state of the economy, and a neural networks is able to extract the relevant information from the large set of macroeconomic variables used.

Although a structural interpretation of the results obtained from neural networks is somewhat prohibitive, it is instructive to have a broad understanding of what variables might be driving the predictions, especially in relation to our exercise that makes use of a large panel of macroeconomic variables. To this end, we design an evaluation procedure which investigate the marginal relevance of single variables based on the partial derivative of the target variable with respect to sample average of each input. These partial derivatives represent the sensitivity of the output to the ith input, conditional on the network structure and the average value of the other input variables (see Dimopoulos et al., 1995), and are akin to the betas of a simple linear regression.

Empirically, we provide evidence of a significant heterogeneity in the relative importance of macroeconomic variables across bond maturities. For instance, the importance of the effective fund rates and the S&P composite index tends to decrease as the maturity increases. Conversely, variables related to the housing sector and inflation substantially dominate the ranking of the most relevant predictors. Furthermore, we calculate the relative importance from the partial derivatives averaged for each class of input variables as labeled in McCracken and Ng (2016). In the same way as for the single variables we show that the predictability on the short-term maturity is dominated by the stock market and financial variables in general, whereas variables more correlated with economic growth such as consumption and output tend to be more relevant for the long-end of the yield curve.

1.1 Related Literature

This paper contributes to at least three main strands of literature. First, this paper adds to a large literature on bond returns predictability. Several studies provide statistical evidence in support of bond returns predictability by means of variables such as forward spreads (e.g., Fama and Bliss, 1987), yield spreads (e.g., Campbell and Shiller, 1991), and linear combinations of forward rates (see Cochrane and Piazzesi, 2005). Interestingly, the debate about bond returns predictability is far from settled: in an important study Thornton and Valente (2012) show that models based on forward rates or yields do not outperform the no-predictability benchmark. We contribute to this literature by showing that, after accounting for potential non-linear relations between yield-based predictors and bond returns, there is evidence in favor of out-of-sample predictability of bond excess returns.

Following the spirit of Litterman and Scheinkman (1991), researchers often summarize term structures by a small set of linear combinations of yields. Yet, recent studies show that there is substantial information about future excess returns that is not embedded in the current yield curve (see, e.g., Cooper and Priestley, 2008, Ludvigson and Ng, 2009, Duffee, 2011b, Joslin et al., 2014, Cieslak and Povala (2015), and Gargano et al., 2017). We contribute to this debate by showing that a decoupled ensemble neural network in which non-linear latent features of forward rates and macroeconomic variables are initially extracted separately, and then joined at the output level allows to reach a substantially higher out-of-sample predictive R^2. In this respect, our paper reinforces the evidence in favor of unspanned macroeconomic information to forecast bond excess returns. Our approach based on neural network ensembles extends the literature by showing a novel way of modeling the (non-linear) relation between the term structure, macroeconomic variables, and bond returns.

Particularly relevant for our analysis is the approach proposed by Ludvigson and Ng (2009, p. 5034) who acknowledge that "factors that are pervasive for the panel of data [input] need not be important for predicting [the output]" and propose a three-step forecasting procedure where a subset of principal components extracted from a large panel of macroeconomic variables is selected according to the BIC criteria before running the bond returns forecasting regressions. In line with this intuition, we provide evidence that non-linear supervised learning methodologies such as neural networks are useful to exploit the information in predictors other than yields, and to improve the measurement of expected bond returns as testified by the increase in out-of-sample predictive R^2 relative to other linear methods.

Second, we contribute to a growing literature that explores the use of machine learning methodologies in empirical asset pricing. Early attempts are Kuan and White (1994), Lo (1994), Hutchinson et al. (1994), Yao et al. (2000), who introduced the use of artificial neural networks in economics and finance. More recent work by Kelly and Pruitt (2013), Kelly and Pruitt (2015), Kozak, Nagel and Santosh (2017), Feng et al. (2017), Freyberger et al. (2017), Giglio and Xiu (2017), Feng, Polson and Xu (2018), Kelly, Pruitt and Su (2018), and Messmer (2017) further show the advantages and promises of traditional shrinkage/regularization methods and data compression for equity markets. In the context of a comprehensive evaluation of machine learning methodologies, Gu et al. (2018) provide evidence that non-linear supervised learning can substantially improve forecasts of expected stock excess returns, allowing for a better measurement of the equity risk premium. Heaton et al. (2017) and Feng et al. (2018) further develop a deep neural network framework for portfolio selection and the machine-driven construction of investment strategies. We contribute to this literature by simultaneously exploring a wide range of machine learning methods to measure bond risk premia, with a particular emphasis on non-linear neural networks.

The application of machine learning for forecasting bond excess returns is further motivated by the aftermath of the great financial crisis, a period where non-linearities in the dynamics of the term structure, and consequently in bond excess returns, may play a dominant role. For instance, Bauer and Rudebusch (2016) show that conventional linear dynamic term structure models severely violate the zero lower bound on nominal interest rates. Similarly, Fernández-Villaverde et al. (2015) provide evidence on the importance of explicitly considering non-linearities in analyzing the dynamics of nominal interest rates when unconventional monetary policies are adopted.

Finally, our paper connects to a growing literature that aims to understand the advantages and properties of deep neural networks in empirical finance. Deep neural networks have a long history and have been proved successful in a wide range of fields, including Artificial Intelligence, image processing, and neuroscience. Early results on stochastic recurrent neural networks (a.k.a Boltzmann machines) were published in Ackley et al. (1985). Jones (2006), Heaton et al. (2017) and Sirignano et al. (2018) constitute recent applications of deep learning hierarchical models to derivatives prices, smart indexing in finance and mortgage risk, respectively. Recently Lee (2004) demonstrates a connection of neural networks with Bayesian non parametric techniques, and Polson and Sokolov (2017) discuss a Bayesian and probabilistic approach to deep learning. We refer to Schmidhuber (2015) for a comprehensive historical survey of deep learning and its applications. We contribute to this literature by showing how deep and ensemble neural networks can achieve out-of-sample performance gains versus linear additive models. This is a far cry from both linear models and the relatively crude, ad hoc methods of regularization and data compression techniques commonly used in empirical asset pricing.

The rest of the paper is organized as follows. Section 2 provides a discussion of why machine learning techniques can prove useful to measure bond expected returns within the context of regression-based predictive regressions. Section 3 outlines the machine learning methodologies used in the paper. Section 4 and 5 describe the design of the empirical applications and the results. Section 6 delves further into the performance of neural networks by investigating the implications for the prediction of the level, slope and curvature of the term structure as well as macroeconomic activity. Section 7 concludes.

2 Motivating Framework

This section provides a motivation for the use of machine learning to predict treasury bond excess returns, that is for measuring bond risk premia. The discussion is framed within the context of regression approaches to forecasting treasury yields. Consider a zero-coupon bond with maturity $t+n$ and a payoff of a dollar. Denote its (log) price and (continuously compounded) yield at time t by $p_t^{(n)}$ and $y_t^{(n)} = -\frac{1}{n}p_t^{(n)}$. The superscript refers to the bond's remaining maturity. The (log) excess return to the n-year bond from t to $t+1$, when its remaining maturity is $n-1$, is denoted by $xr_{t+1}^{(n)} = p_{t+1}^{(n-1)} - p_t^{(n)} - y_t^{(1)}$. An identity links the current yield to the sum, during the bond's lifetime, of one-period yields and excess returns

$$y_t^{(n)} = \frac{1}{n}E_t\left(\sum_{j=0}^{\infty} y_{t+j}^{(1)}\right) + \frac{1}{n}E_t\left(\sum_{j=0}^{\infty} xr_{t+j,t+j+1}^{(n-j)}\right) . \qquad (1)$$

These expectations hold regardless of the information set used for conditioning, as long as the set contains the yield $y_t^{(n)}$. Assume that investors' information set at time t can be summarized by a latent k-dimensional state vector $\boldsymbol{x}_t$. In particular, $\boldsymbol{x}_t$ represents the information that investors use at time t to predict bond yields and excess returns for all future periods $t+1, t+2, \ldots, t+h$. Using this assumption in the identity (1) produces

$$y_t^{(n)} = \frac{1}{n}\sum_{j=0}^{\infty} E_t\left(y_{t+j}^{(1)} \mid \boldsymbol{x}_t\right) + \frac{1}{n}\sum_{j=0}^{\infty} E_t\left(xr_{t+j+1}^{(n-j)} \mid \boldsymbol{x}_t\right) \ .$$

It is easy to see that the yield on the left-hand side cannot be a function of anything other than the state vector, since only $\boldsymbol{x}_t$ shows up on the right-hand side. Hence we can write

$$\boldsymbol{y}_t = f(\boldsymbol{x}_t; N) \ ,$$

where we stack time-t yields on bonds with different maturities in a vector $\boldsymbol{y}_t$, and the maturities of the bonds are in the vector N. If we also assume there exists an inverse function such that $\boldsymbol{x}_t = f^{-1}(\boldsymbol{y}_t; N)$, i.e. each element of $\boldsymbol{x}_t$ has a unique effect on the yield curve, then we can write

$$E_t\left[xr_{t+1}^{(n)}\right] = g(\boldsymbol{y}_t; N) \ , \tag{2}$$

for some function $g(\boldsymbol{y}_t; N)$. Put differently Eq. (2) says that the time-t yield curve contains the information necessary to predict future values of $\boldsymbol{x}_t$, and thus the one-period future yields and bond excess returns. That is, the vector $\boldsymbol{x}_t$, or equivalently a combination of yields (or forward rates), is all that is needed to forecast bond excess returns for all future horizons and maturities.

Standard practice posits that the function $g(\cdot)$ is linear, so that we can write $\boldsymbol{x}_t$ as a portfolio of yields. Following Litterman and Scheinkman (1991), it is also common to use the first three principal components – dubbed level, slope and curvature – to proxy for these (linear) combination of yields. Linearity of $g(\cdot)$ together with sparsity in the space of principal components gives rise to the traditional regression-based forecasting of bond excess returns

$$E_t\left[xr_{t+1}^{(n)}\right] = \hat{\alpha} + \hat{\boldsymbol{\beta}}^{\top}\boldsymbol{x}_t \qquad \text{where} \qquad \boldsymbol{x}_t = \boldsymbol{W}\boldsymbol{y}_t + b \ , \tag{3}$$

where the columns of $\boldsymbol{W}$ form an orthogonal basis for directions of greatest variance (which is in effect an eigenvector problem), and b captures the average reconstruction error. This framework is known in the machine learning literature as Principal Component Regression (PCR) where the quantity of interest is regressed onto the derived inputs from PCA (see, Ch.3.5 Friedman et al., 2001). Practically, the linear predictive system outlined in Eq. (3) represents a two-step procedure where researchers extract the latent factors $\boldsymbol{x}_t$ first, and then learn the regression coefficients $\hat{\theta} = \left(\hat{\alpha}, \hat{\boldsymbol{\beta}}^\top\right)$ by minimizing some form of residual sum of squares.

Eq. (2) implies that the period-t cross-section of yields contains all information relevant to forecasting future yields. That is, one has to rule out the presence of additional variables with equal and opposite effects on expected future short rates and expected future excess returns. However, state variables that drive this kind of variation drop out of the left side in (1), hence the yields do not necessarily span the information set used by investors to forecast future yields.[7] This means that, if information at time t other than the yields is helpful to predict future yields and excess returns, then investors could use that information in addition to $\boldsymbol{x}_t$. Such additional information is often considered in the form of macroeconomic variables. More specifically, one could consider an extended predictive regression in the form

$$E_t\left[xr_{t+1}^{(n)}\right] = \hat{\alpha} + \hat{\boldsymbol{\beta}}^\top \boldsymbol{x}_t + \hat{\boldsymbol{\gamma}}^\top \boldsymbol{F}_t \tag{4}$$

where $\boldsymbol{F}_t \subset f_t$ and f_t is an $r \times 1$ vector of latent common factors extracted from a $T \times N$ panel of macroeconomic data with elements $m_{it}, i = 1, \dots, N,\ t = 1, \dots, T$, and $r \ll N$. This is the framework originally proposed by Ludvigson and Ng (2009). The distinction between F_t and f_t is important since "factors that are pervasive for the panel of data need not be important for predicting $xr_{t+1}^{(n)}$" (Ludvigson and Ng, 2009, p. 5034). Specifically, Ludvigson and Ng (2009) follow a three-step procedure. First, the first eight latent common factors of the macroeconomic variables $\hat{f}_{1t}, \dots, \hat{f}_{8t}$ are estimated. Second, an information criterion is used to select a subset of these factors (and possibly nonlinear functions of those factors) to forecast bond excess returns.[8] Finally, conditional on the chosen specification of the factors, the coefficients $\hat{\boldsymbol{\theta}} = \left(\hat{\alpha}, \hat{\boldsymbol{\beta}}^\top, \hat{\boldsymbol{\gamma}}^\top\right)$ in regression (4) are estimated through least squares methods. As far as the latent state $\boldsymbol{x}_t$ is concerned, Ludvigson and Ng (2009) follows Cochrane and Piazzesi (2005) whereby $\boldsymbol{x}_t$ is a linear combination of all forward rates at time t.

Both the approaches outlined in equations (3) and (4) are essentially based on three main assumptions. First, they assume that the relationship between future bond excess returns and current yields, macroeconomic information, or both, is linear. That is, the latent factors which are assumed

[7]The hidden factor models successfully introduced by Duffee (2011b) and Joslin et al. (2014) capture this idea.

[8]The specification proposed by Ludvigson and Ng (2009) is a linear function of five out of the eight extracted principal components of the form

$$\gamma' \boldsymbol{F}_t = \gamma_1 F_{1t} + \gamma_2 F_{1t}^3 + \gamma_3 F_{3t} + \gamma_4 F_{4t} + \gamma_5 F_{8t}\ .$$

Notice that although F_{1t} enters in the specification with a cubic exponential, it is still originally extracted as a linear combination of the original macroeconomic inputs.

to summarize the investors' conditioning information are linear combinations of the input variables, regardless of their origin. The assumption of linearity is potentially restrictive. As equation (2) suggests, the theory does not grant a linear relation between future bond excess returns and the current investors' information. This raises the possibility that a more precise measurement of bond risk premia can be obtained by using non-linear transformations of the data. This is an avenue that has also been advocated by Stock and Watson (2002a, p. 154) within the context of forecasting macroeconomic time series.

Second, the benchmark implementation of regression-based forecasts of bond excess returns using principal components as outlined in Eq. (3)-(4) implies that no direct use of the response variable, i.e., the bond excess returns, is made to learn about the state variables $\boldsymbol{x}_t$ and $\boldsymbol{F}_t$. This is not surprising as data compression methods such as PCA fall under the labeling of "unsupervised learning"; the algorithm is left to its own device to discover and present the structure of interest in the input data (the yields or macroeconomic variables) without the use of the output (the bond excess returns). To fix ideas, take the yields $\boldsymbol{y}_t$ as the only investors' information; unsupervised learning methods assume that $\boldsymbol{x}_t$ can be extracted by using the marginal distribution of $\boldsymbol{y}_t$ rather than the joint distribution of $\boldsymbol{y}_t$ and the bond excess returns.[9] However, Eq. (2) suggests that bond excess returns play the implicit role of conditioning argument, namely, one should be able to tailor the extraction of hidden latent states $\boldsymbol{x}_t$ to the response variable $xr_{t+1}^{(n)}$. "Supervised learning" algorithms such as the Lasso, Elastic Net, partial least squares, regression trees and neural networks, which explicitly condition on the response variables to summarize the information in the predictors, may arguably prove useful to overcome the limitations of standard data compression methods such as PCA and latent factor analysis (FA).

Third, traditional PCA and FA are based on the assumption that all variables could bring useful information for the prediction of future bond excess returns, although the impact of some of them could be small. However, PCA or FA is not a black box where we can simply add any number of predictors and then be sure that the extracted factors will provide an optimal summary. Boivin and Ng (2006b) formalize this argument by providing evidence that the structure of the common components is sensitive to the input variables, and that not always more data means more sensible estimates. In this respect, one may want to "select" the variables that actually matter for forecasting

[9]More generally, the lack of success in extracting the optimal state variables, in a forecasting sense, is not measured by looking at the expected loss over the joint distribution of bond excess returns and yields, but rather depends on ex-post heuristic arguments like the goodness-of-fit of the predictive regression in Eq. (3).

bond excess returns. Penalized regressions, such as Lasso and Elastic Net, as well as neural networks with "drop-out" regularization – see, e.g., Srivastava et al. (2014) – allow to exploit the entire span of the input variables without imposing that they all carry useful information for the prediction of bond excess returns.

The existing literature on bond return predictability has vastly ignored the potentiality of machine learning techniques to address the issue of non-linearity and variable selection. Arguably, this comes at the expense of not fully capturing the extent to which yields and macroeconomic variables are relevant for the measurement of expected bond returns. This is the focus of our paper.

3 Competing Machine Learning Methodologies

This section describes the set of machine learning methodologies implemented for the measurement of bond risk premia, with a particular emphasis on the dichotomy between unsupervised vs supervised learning. PCA represents the most popular instance of a large class of machine learning algorithms that fall under the heading of dimensionality-reduction, projection methods, or data compression. Prominent examples in finance and economic are provided by Forni and Reichlin (1996, 1998), Stock and Watson (2002a,b, 2006), Bai and Ng (2003), Cochrane and Piazzesi (2005), Bai and Ng (2006, 2008), Boivin and Ng (2006a), Diebold and Li (2006), De Mol et al. (2008), and Ludvigson and Ng (2009), among others. Given the wide use of PCA in financial economics, and bond returns predictability in particular, we take PCA as our benchmarking predictive strategy. In addition, PCA offers a natural understanding of neural networks via an Autoencoder. An autoencoder is a type of neural network whose outputs are its own inputs. More specifically, Appendix A shows that, under linearity of the activation functions, PCA is analogous to a two-layer autoencoder.

For each method, we discuss the objective function and the algorithmic procedure used to estimate the hyper-parameters. For instance, hyper-parameters include the weights for each activation function in hidden layers of the neural networks as well as the penalization parameters in the penalized regressions.

Overfitting is a major concern when training complex machine learning algorithms. In the empirical asset pricing literature one splits the data in a training (in-sample) period and a test (out-of-sample) period to evaluate model performance. However, pure data learning approaches lack prior beliefs on the intrinsic nature of the data generating process. On the one hand, this fact makes machine learning methods highly flexible. On the other hand, it requires a more careful approach to tackle model complexities explicitly (see Bishop et al., 1995). In practice, it is common to split the data in three sub-samples: a training set used to train the model, a validation set used to evaluate the estimated model on an independent data set, and a testing set which represents the out-of-sample period in a typical forecasting exercise.

The training sample represents the actual part of data that is used to train the model, e.g., weights and biases in neural networks, and it is subject to different tuning parameter values. The size of the training set often depends on the application. Some models and applications need substantial data to train upon, so in this case it would be intuitive to utilize a training set as large as possible. The optimal trade-off between the size of the training and validation samples is ultimately an empirical question.

The validation sample represents the part of data that is used to provide an unbiased evaluation of a model fit. The purpose is to simulate an out-of-sample test to provide a measure of the expected prediction error based on an independent dataset. We follow Gu et al. (2018) and construct the validation sample as follows: conditional on the estimates from the training set we produce forecasting errors over the validation sample. Next, we use the prediction errors over the validation sample to iteratively search the hyper-parameters that optimize the objective function.[10] It is trivial to see that predictions in the validation set are not truly out-of-sample as they are used to tune the model hyper-parameters.

The third sub-sample, or the testing sample, contains observations that are not used for estimation or tuning. Being truly out-of-sample, this sub-sample can be used to test the predictive performance of the model.

There is a variety of splitting schemes that could be considered (see Arlot et al., 2010 for a comprehensive survey of cross-validation procedures for model selection). In our empirical exercise we keep the fraction of data used for training and validation fixed at 85% and 15% of the in-sample data, respectively. The training and the validation samples are consequential. In this respect, we do not cross-validate by randomly selecting independent subsets of data in order to preserve the time-series dependence of both the predictors and the target variables. Forecasts are produced recursively

[10]For instance, in Lasso-type penalized regressions the validation sample is used to calculate the amount of sparsity that produces the best forecasts given the estimates in the training sample.

by using an expanding window procedure, that is we re-estimate a given model at each time t and produce out-of-sample forecasts for non-overlapping one-year holding period excess returns. Figure 1 provides a visual representation of the sample splitting we adopt in the empirical analysis.

[Insert Figure 1 about here]

The blue area represents the sum of the training and validation sample. The red area represents the testing sample. Notice that for some of the methodologies we consider, validation is not required. For instance, neither standard linear regressions nor PCA requires a pseudo out-of-sample period to validate the estimates. In these two cases, we adopt a traditional separation between in-sample vs out-of-sample period, where the former consists of the sum of the training and the validation data.

3.1 Simple and Penalized Linear Regressions

The simple linear regression model has been the mainstay of returns predictability over the past decades and remains one of the most important tools in the empirical asset pricing literature (see, e.g., Ang and Bekaert, 2006, Campbell and Thompson, 2007, Cochrane, 2007, and Welch and Goyal, 2007). To fix ideas consider a typical linear model as Eq. (3) in which one uses the cross-section of yields $\boldsymbol{y}_t$ as predictors instead of the principal components $\boldsymbol{x}_t$.

There are a variety of methods to estimate the parameters $\boldsymbol{\theta} = \left(\alpha, \boldsymbol{\beta}^\top\right)$ of a linear predictive regression model, but by far the most popular is standard least squares which minimizes

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{t}\sum_{\tau=1}^{t-1}\left(xr_{\tau+1}^{(n)} - \alpha - \boldsymbol{\beta}^\top \boldsymbol{y}_\tau\right)^2 \tag{5}$$

where $\tau = 1, \ldots, t$ represents the in-sample period up to time t. Despite its popularity, linear predictive models are bound to fail in the presence of many predictors. In practice, the out-of-sample performance of least squares estimates – in fact, even of maximum likelihood and Bayesian inference with uninformative priors – tend to deteriorate as the number of predictors increases, a fact that is well known as the curse of dimensionality (see, e.g., Stein, 1956). As pointed out in Gu et al. (2018), the curse of dimensionality leads to overfitting in sample and can be particularly problematic in applications with low signal-to-noise ratio as it is typical the case for forecasting asset returns.

Confronted with a large set of predictors in linear regression models a popular strategy is to impose sparsity in the set of regressors via a penalty term. The idea is that by focusing on the selection of a sub-set of variables with the highest predictive power out of a large set of predictors, and discarding the least relevant ones, one can mitigate in-sample overfitting and improve the out-of-sample performance of the linear model.[11] In its most general form a penalized regression entails a penalty term in the objective function (5), that is[12]

$$\mathcal{L}(\boldsymbol{\theta};\cdot) = \underbrace{\mathcal{L}(\boldsymbol{\theta})}_{\text{Loss Function}} + \underbrace{\phi(\boldsymbol{\beta};\cdot)}_{\text{Penalty Term}} . \tag{6}$$

Depending on the functional form of the penalty term, the regression coefficients can be regularized and shrunk towards zero, completely set to zero, or a combination of the two. More specifically, the penalization term can take the following form (see, Ch.3 Friedman et al., 2001),

$$\phi(\boldsymbol{\beta};\cdot) = \begin{cases} \lambda \sum_{j=1}^{p} \beta_j^2 & \text{Ridge regression} \quad (7a) \\ \lambda \sum_{j=1}^{p} |\beta_j| & \text{Lasso} \quad (7b) \\ \lambda\mu \sum_{j=1}^{p} \beta_j^2 + \frac{\lambda(1-\mu)}{2} \sum_{j=1}^{p} |\beta_j| & \text{Elastic net} \quad (7c) \end{cases}$$

The ridge regression (c.f. 7a) shrinks the regression coefficients (excluding the intercept) by imposing a penalty on their size. The parameter λ controls the amount of shrinkage, that is the larger the value of λ, the greater the amount of shrinkage/regularization.[13] By imposing a size constraint, ridge regressions alleviate the concern that predictors may be highly correlated in the linear regression model. If that is the case, coefficients are poorly determined as a large coefficient on a given variable can be offset by a similarly large negative coefficient on a correlated pair.[14] The lasso (c.f. 7b) is a shrinkage regression method like ridge, but with important differences. Unlike ridge, the nature of the L_1 constraint shrinks those coefficients sufficiently small to be exactly zero. In this respect, whereas ridge is a dense model with a closed-form solution, the lasso is a sparse model in which there

[11]A similar approach in the Bayesian literature is the spike-and-slab prior introduced by George and McCulloch (1993) which allows to implement variable selection through a data-augmentation framework (see Giannone et al., 2017 for a discussion).

[12]Notice that the intercept α is not included in the penalty terms. Penalization on the intercept would make the optimization procedure dependent on the initial values chosen for the bond excess returns; that is, adding a fixed constant to the bond excess returns would not simply result in a shift of the prediction by the same amount.

[13]Ridge regressions are not scale invariant, and so normally the inputs are standardized before solving the objective (6)-(7a).

[14]Interestingly, in the case of orthonormal inputs one can show that the solution of the ridge regression is equivalent to a scaled version of the least squares estimates.

is no closed-form solution (see Giannone et al., 2017). As a whole both methods apply a different transformation to the least squares estimates: ridge regression does a proportional shrinkage whereas the lasso rescales each coefficient truncating at zero (see, Ch. 3 p. 69 Friedman et al., 2001, for a comprehensive discussion).

Finally, the Elastic net penalty (c.f. 7c) introduced by Zou and Hastie (2005b) represents a compromise between the ridge and the lasso. The elastic net selects variables like the lasso and shrinks the coefficients of the highly correlated predictors like the ridge. Specifically, the first term $\lambda\mu\sum_{j=1}^{p}\beta_j^2$ tends to average highly correlated regressors, while the second term $\lambda(1-\mu)/2\sum_{j=1}^{p}|\beta_j|$ encourages a sparse solution. The parameter λ for ridge and lasso, and the pair λ, μ for elastic-net are estimated adaptively on the validation sample by using a cyclical coordinate descent method as proposed by Wu et al. (2008) and extended by Friedman et al. (2010).

3.2 Regression Trees and Random Forests

Regression trees are based on a partition of the input space into a set of "rectangles". Then, a simple linear model is fit to each rectangle. They are conceptually simple, yet powerful, and therefore highly popular in the machine learning literature. A regression tree is produced as a set of recursive binary partitions, that is we first split the regression space into two regions, and predict the target variable in each region where the split-point is chosen to achieve the best fit. Then, one or both regions are further split in two and the process goes on until some stopping rule is applied. In this respect, at each step the data are classified and the relationship between the target and the response variables is approximated within each partition. To fix ideas, let us consider the one-step ahead prediction of the holding period excess return of a one-year treasury bond, i.e., $xr_{t+1}^{(1)}$, based on two predictors, say, the two-year and the five-year yields, denoted by $y_t^{(2)}$ and $y_t^{(5)}$. Figure 2 displays an example of a binary partition (left panel) and the corresponding regression tree (right panel).

[Insert Figure 2 about here]

We first split $y_t^{(2)}$ at some threshold value p_1. Then the region $y_t^{(2)} \le p_1$ is split at $y_t^{(5)} = p_2$ while the region $y_t^{(2)} > p_1$ is split at $y_t^{(2)} = p_3$. Finally, the region $y_t^{(2)} > p_3$ is split at $y_t^{(5)} = p_4$. As a result, the input space is partitioned in the five regions $A_1, \dots, A_5$ as shown in the left panel. The same model can be reported as a binary tree as shown in the right panel.

A key advantage of a regression tree is that it can approximate any a priori unknown function while keeping the interpretation from the recursive binary tree. With more than two inputs the interpretation is less obvious as trees like the one depicted in Figure 2 grow exponentially in size. Nevertheless the algorithmic procedure is equivalent. Suppose one deals with a partition of M regions $\mathcal{A} = \{A_1, \dots, A_M\}$ of the vector of yields $\boldsymbol{y}_t$ such that

$$g(\boldsymbol{y}_t; N) = \sum_{m=1}^{M} \beta_m \mathbb{I}\left(\boldsymbol{y}_t \in A_m\right) \ .$$

By minimizing the sum of squared residuals, one can show that the optimal estimate $\hat{\beta}_m$ is just the average of the bond excess returns in that region, i.e., $\hat{\beta}_m = E\left[xr_{t+1}^{(1)} \middle| \boldsymbol{y}_{1:t} \in A_m\right]$. Finding the optimal partition by using a least squares procedure is generally infeasible, however. We thus follow Friedman (2001) and implement a gradient boosting procedure. Gradient boosting in a tree context boils down to combining several weak trees of shallow depth.[15]

Boosting is a technique for reducing the variance of the model estimates and increasing precision. However, trees are "grown" in an adaptive way to reduce the bias, and thus are not identically distributed. An alternative procedure would be to build a set of *decorrelated* trees which are estimated separately and then averaged out. Such modeling framework is known in the machine learning literature as "Random Forests' (see Breiman, 2001). It is a substantial modification of bagging (or bootstrap aggregation) whereby the outcome of independently drawn processes is averaged to reduce the variance estimates. Bagging implies that the regression trees are identically distributed – that is the variance of the average estimates, as the number of simulated trees increases, depends on the variance of each tree times the correlation among the trees. Random forests aim to minimize the variance of the average estimate by minimizing the correlation among the simulated regression trees.

We follow existing literature and train random forests by randomly choosing subsets of regressors for splitting at each step of the tree. This lowers the correlation across predictions at the benefit of reducing the overall variance of the estimates. The individual trees are trained using the Classification and Regression Trees (CART) algorithm (see, Ch.9 Friedman et al., 2001).

[15]The number of weak learners ensembled is set to 10 and the maximum depth to three.

3.3 Partial Least Squares

The benchmark PCA regressions as outlined in Eq. (3)-(4) project the bond excess returns across different maturities onto a linear combination of the input variables (yields, macroeconomic variables, or both). The linear combinations of the inputs are derived without the use of the dependent variable, that is, principal components are uniquely based on the marginal distribution of the predictors.

A similar data compression methodology falls under the heading of Partial Least Squares (PLS). Unlike PCR, with PLS the common components of the predictors are derived by conditioning on the joint distribution of the target variable and the regressors. Like PCR, partial least squares is not scale invariant, so we assume that inputs are standardized to have mean zero and unit variance.

Following the extant practice (see, Ch.3.5 Friedman et al., 2001) PLS is constructed iteratively as a two-step procedure: in the first step we regress bond excess returns on each predictor $j = 1, \ldots, p$ separately and store the regression coefficient ψ_j. The first partial least squares direction is constructed by multiplying the vector of coefficients by the original inputs, that is $\boldsymbol{x}_1 = \boldsymbol{\psi}'\boldsymbol{y}_t$. Hence the construction of $\boldsymbol{x}_1$ is weighted by the strength of the relationship between the bond excess returns and the predictors. In the second step, bond excess returns are regressed onto $\boldsymbol{x}_1$ giving the coefficient θ_1. Then all inputs are orthogonalized with respect to $\boldsymbol{x}_1$. In this manner, PLS produces a sequence of $l < p$ derived inputs (or directions) orthogonal to each other.[16]

Notice that since the response variable is used to extract features of the input data, the solution path of PLS represents a non-linear function of bond excess returns. Stone and Brooks (1990) and Frank and Friedman (1993) show that, unlike PCA which seeks directions that maximize only the variance, the PLS maximizes both variance and correlation with the response variable subject to orthogonality conditions across derived components.[17] PLS does not require the calibration of hyper-parameters as the derived input directions are deterministically obtained by the two-step procedure outlined above. In this respect, unlike penalized regressions no shrinkage/regularization parameters are required to be calibrated.

[16]It is easy to see that for $l = p$ we go back to usual linear least squares estimates similar to PCR.

[17]In particular, the mth direction solves:

$$\begin{aligned} &\max_{\gamma} \ \mathrm{Corr}^2\left(xr^{(n)}, \boldsymbol{y}\gamma\right) \cdot \mathrm{Var}\left(\boldsymbol{y}\gamma\right) \\ &\text{subject to} \quad \|\gamma\| = 1, \quad \gamma'\Sigma\hat{\psi}_j = 0, \quad j = 1, \ldots, m-1 \end{aligned}$$

3.4 Neural Networks

Neural networks (NN) represent a widespread class of supervised learning methods that has been developed in different fields, such as biostatistics, image processing, neuroscience, and artificial intelligence. The central idea of NN is to extract complex non-linear combinations of the input data by conditioning on both the target (bond excess returns) and the inputs (yields, macro variables, or both). As a result, the latent states $\boldsymbol{x}_t, \boldsymbol{F}_t$ and the parameters $\alpha, \boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top$ in Eq. (3)-(4) are estimated jointly through a non-linear specification which generalizes the class of linear models (see, Ch.11 Friedman, Hastie and Tibshirani, 2001).

We focus our analysis on traditional "feed-forward" networks or multi-layer perceptrons (MLP). An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. As a result, MLPs can distinguish data that are not linearly separable. The result is a powerful learning method which can approximate virtually any continuous function with compact support (see Kolmogorov, 1957; Diaconis and Shahshahani, 1984; Cybenko, 1989; Hornik, Stinchcombe and White, 1989). The advantage of feed-forward networks is that they do not require to manipulate, extend, or order the input data unlike other specifications such as recurrent neural networks (RNN), Convolutional Neural Networks (CNN), Long-Short Term Memory networks (LSTM) and Neural Turing Machines (NTM).

Figure 3 shows two simple examples of neural networks implemented in our empirical analysis. The left panel shows a "shallow" network which consists of an output layer and a single hidden layer. The right panel shows a "deep" neural network which consists of an output layer and three hidden layers.

[Insert Figure 3 about here]

The green circles represent the input variables, e.g., the cross-section of yields. The purple circles represent the fully connected hidden nodes. The red circles represent the output variables, that is the bond excess returns across maturities.

A general definition of a multi-layer neural network is as follows. Let $h_1, \ldots, h_L$ be univariate activation functions for each of the L (hidden) layers of the network. These represent non-linear transformation of weighted data. We denote $\boldsymbol{z}_l$ the l-th layer which is a vector of length equal to the number of nodes (or neurons) in that layer, such that $\boldsymbol{z}_0 = \boldsymbol{y}_t$ is the vector of inputs. The explicit structure can be summarized as a composition of univariate semi-affine functions, i.e.,

$$H^{\boldsymbol{W},b} := h_1^{\boldsymbol{W}_1,b_1} \circ \ldots \circ h_L^{\boldsymbol{W}_L,b_L} \ , \tag{8}$$

with

$$h_l^{\boldsymbol{W}_l,b_l}\left(\boldsymbol{z}_{l-1,t}\right) = h_l\left(\boldsymbol{W}_l \boldsymbol{z}_{l-1,t} + b_l\right), \qquad \forall 1 \le l \le L \ , \tag{9}$$

where the matrices of weights $(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L)$ and the biases $(b_1, \ldots, b_L)$ are the objects to be estimated. Interestingly, Eq. (2) suggests that NN can be particularly suitable since a finite composition of univariate semi-affine functions like Eq. (8) have been shown to uniformly approximate any unknown compact and continuous function with arbitrary accuracy (see Funahashi, 1989; Hornik, 1993).

In the case of predictive regressions based on the cross-section of yields the neural network is just a hierarchical model of the form

$$\begin{aligned}
E_t\left[xr_{t+1}^{(n)}\right] &= \hat{\alpha}_n + \hat{\boldsymbol{\beta}}_n^{\top} \boldsymbol{x}_t && \text{(Output layer)} \\
\boldsymbol{x}_t &= h_L\left(\boldsymbol{W}_L \boldsymbol{z}_{L-1,t} + b_L\right) && \text{(Hidden layer } L\text{)} \\
\boldsymbol{z}_{L-1,t} &= h_{L-1}\left(\boldsymbol{W}_{L-1} \boldsymbol{z}_{L-2,t} + b_{L-1}\right) && \text{(Hidden layer } L-1\text{)} \\
&\vdots \\
\boldsymbol{z}_{1,t} &= h_1\left(\boldsymbol{W}_1 \boldsymbol{y}_t + b_1\right) \ . && \text{(Hidden layer 1)}
\end{aligned}$$

By introducing the non-linear transformations $H^{\boldsymbol{W},b}$, a deep neural network enlarges the class of linear models, and allows to capture complexities in the data which cannot be discovered within a standard linear framework. Based on the above architecture, a shallow network is simply a NN with $L = 1$, which consists of a low dimensional composition function:

$$E_t\left[xr_{t+1}^{(n)}\right] = \hat{\alpha}_n + \hat{\boldsymbol{\beta}}_n^{\top} \boldsymbol{x}_t \qquad \text{where} \qquad \boldsymbol{x}_t = h\left(\boldsymbol{W} \boldsymbol{y}_t + b\right) \tag{10}$$

By looking at Eq. (10) it is immediate to see that for a linear activation function $h^{\boldsymbol{W},b}(\boldsymbol{y}_t)$ the NN framework can be thought of as a predictive factor model as the one described in Eq. (3).

We investigate the predictive performance of different network architectures. The depth and width of the network is relevant to the extent that the hidden layers detect unobservable features in the data. We start from a shallow neural network with a single hidden layer and expand the network to a deeper four-layer structure, that is the output layer in addition to three hidden layers. We limit the structure to four layers as it is difficult to support a richer parametrization, namely more layers, without a substantially larger amount of observations.

We explore different specification in terms of number of nodes for each hidden layer depending on the empirical application. For instance, when forecasting bond excess returns based on forward rates only, we consider both a specification with three and five nodes for a two-layer neural network which contrast the performance of a three- and five-factor principal component regression. For a deeper network we explore architectures in which the number of nodes is constant across layers. In addition, we follow Gu et al. (2018) and implement an alternative network structure in which the number of nodes decreases from the direction of input to the output by an approximate geometric rate as suggested in Masters (1993). A comparison of the out-of-sample predictive performance of different NN architectures allows us to discuss the trade-off between depth and width of the network structure.

The activation function for each node is applied to each element (see Figure 3) and can take various forms. Commonly used non-linear activation functions are the hyperbolic tangent (tanh), the sigmoid, e.g., $h(z) = 1/(1+\exp(-z))$, and the Rectified Linear Unit (ReLU) $h(z) = \max(z,0)$. We follow existing literature and utilize ReLU functions, which are computationally attractive and have the advantage of avoiding vanishing gradient problems that affect both the tanh and the sigmoid (see Jarrett et al., 2009, Nair and Hinton, 2010, Glorot et al., 2011, Messmer, 2017, Polson and Sokolov, 2017, Feng et al., 2018 and Gu et al., 2018). For the shallow neural network Eq. (10), the ReLU function is defined as $\boldsymbol{x}_t = \max(\boldsymbol{W}\boldsymbol{y}_t + b,\ 0)$.

3.4.1 Optimization and Regularization

Each neural network specification is trained by minimizing a mean square loss function with a penalizing term to induce regularization in the weight estimates. We follow Goodfellow et al. (2016) and adopt an elastic-net type of loss function in which both an L1 and L2 penalty terms are added to the mean square error (c.f. Eq.(7c)). In the spirit of elastic-net, these two penalty terms induce sparsity in the set of estimated weights as well as impose regularization on those weights which are not shrunk to zero.

The estimates of the weights and parameters of a neural network are solutions of a non-convex optimization problem. As a result, conditional on the number of layers and nodes, conventional estimation procedures of neural network heavily build on stochastic optimization routines. A standard approach is to implement a Stochastic Gradient Descent (SGD) algorithm to train the model parameters. The idea underlying the SGD is to evaluate the loss function based on a small set of randomly drawn weights and to iteratively approach the (local) minimum through back propagation. We implement an extension of the standard SGD by incorporating a Nesterov momentum component as proposed by Sutskever et al. (2013).[18]

In order to mitigate legitimate concerns about the possibility of over-fitting the data, we apply a variety of regularization techniques in addition to the use of the penalized mean squares error loss function outlined above. More specifically, we follow Gu et al. (2018) and implement early stopping, batch normalization (BN) and averaging over a set of alternative forecasts. In addition, we also investigate the advantage of ensembling at the output level separately trained sub-networks, one for each groups of macroeconomic variables. Interestingly, we show that the gap between shallow and deep neural networks is reduced as the complexity of the network structure increases.

We refer the reader to Gu et al. (2018) and Appendix (B) for a description of early stopping, batch normalization and averaging. In addition to their setting we further utilize drop-out regularization. Drop-out regularizes the choice of the number of hidden units in a layer (see Srivastava et al., 2014). This can be achieved if we drop units of the hidden layer and then establish which probability p gives the best results. A pre-selection grid-search produced a probability $p = 30\%$ which is then used to randomly drop nodes in each layer at each iteration. Similar to batching, drop-out works because introduces multiple implicit ensembles that share the same weights. The underlying idea is that for each training set, one randomly removes a fraction of the neurons connections. Effectively, one momentary has a subset of the original neural net that runs inference and gets its weights update. Notice that drop-out and BN are similar in spirit. However, their combined use does not necessarily

[18]Notice that using a momentum based algorithm further accelerates the convergence of the optimizer in the direction of the global minimum. However, if momentum is chosen too large, overshooting of the optimum becomes likely. We calibrate the momentum parameter by performing an extensive grid search for each model specification. Notice also that there exist more advanced algorithms such as AdaGrad and AdaDelta that adjust the learning rate automatically and do not need parameter tuning. We performed the estimates using these algorithms as well. However, the empirical results show that these more sophisticated algorithms at their recommended parameters produce models that do not generalize as well as a reasonably well tuned plain stochastic gradient descent algorithm with momentum. Further results on the grid search results and algorithmic investigation are available upon request.

brings additional advantages. Ioffe and Szegedy (2015) suggest that on the one hand batch normalization provides similar regularization benefits, whereas, on the other hand, drop-out together with BN could provide conflicting results in some type of architectures. However Li et al. (2018) show that the conflict between drop-out and BN can be mitigated by applying drop-out after BN, conditional on having a low drop-out probability. In our setup BN is implemented after feeding into the ReLu activation function but before drop-out. BN is applied after the activation since over the course of the training process the input normalization vanishes and a problem referred to as covariate shift occurs. As far as dropout goes, it is applied after activation as suggested by Li et al. (2018).

4 Research Design

The research design we implement to compare machine learning methodologies is similar to Gu et al. (2018).

We compare machine learning methodologies against two benchmark frameworks based on the principal component regressions. All of the machine learning methods are designed to approximate the empirical specification $E_t\left[xr_{t+1}^{(n)}\right] = g(\boldsymbol{y}_t; N)$ defined in Eq. (2) as well as its extension which includes macroeconomic information which is not spanned by the current term structure.

In particular, the first application concerns the forecasting of future bond excess returns based on the cross-section of yields as originally proposed by Cochrane and Piazzesi (2005). The bond return data are taken from the Fama-Bliss dataset available from the Center for Research in Securities Prices (CRSP) and contains observations on one- through five-year zero-coupon U.S. Treasury bond prices. These data are used to construct one-period yields, forward rates and bond excess returns as described in Section 2. We focus on one-year holding period returns. The sample observations are monthly and cover the period 1964:1-2016:12.

The second application consists of forecasting future bond excess returns based on both forward rates and a large panel of macroeconomic variables as proposed by Ludvigson and Ng (2009). We consider a balanced panel of $N = 128$ monthly macroeconomic and financial variables. A detailed description of how variables are collected and constructed is provided in McCracken and Ng (2015). The series were selected to represent broad categories of macroeconomic time series: real output and income, employment and hours, real retail, manufacturing and sales data, international trade, consumer spending, housing starts, inventories and inventory sales ratios, orders and unfilled orders, compensation and labor costs, capacity utilization measures, price indexes, interest rates and interest rate spreads, stock market indicators, and foreign exchange measures. This dataset has been widely used in the literature (see, e.g., Stock and Watson, 2002a, 2006; Ludvigson and Ng, 2009), and permits comparison with previous studies.

The empirical application is conducted by recursively forecasting the non-overlapping one-year ahead holding period bond returns in excess of the short-term rate. We considered both a short sample from 1964:01 to 2008:12 and the enlarged sample from 1964:01 to 2016:12. The latter includes the regime of unconventional monetary policies and interest rates at the zero-lower bound.

We initially divide the sample in three parts: training, validation and testing sample. The training and validation periods together account for 70% of the data and the remaining 30% is for out-of-sample testing. We recursively refit machine learning methods at each time t, meaning each time we increase the in-sample period by one monthly observation. Such recursive monthly estimation scheme allows to incorporate the most recent updates from the yield curve and the set of macroeconomic and financial variables.[19]

When enlarging the in-sample period we roll it forward to include the most recent information in a recursive fashion but keep constant the ratio between the training and the validation sample. In this respect we always retain the entire history of the training sample, thus its window size gradually increases. By keeping the proportion of the training and validation sets fixed, the validation sample gradually increases as well. The result is a sequence of performance evaluation measures corresponding to each recursive estimate. Although computationally expensive this has the benefit of leveraging more information for prediction.

A complete description of the computational specifications is provided in Appendix C. For our implementation of the various machine learning techniques in Python we utilize the well-known packages `Scikit-Learn`. Since the forecasting exercise in this paper is iterative and since we use model averaging, the computational challenge becomes sizeable. For that reason, we perform all computations on a high performance computing cluster consisting of 84 nodes with 28 cores each, totaling to more than 2300 cores. We parallelize our code using the Python `multiprocessing` package.

[19]Notice this is different from the implementation of machine learning methods for stock returns, where trading signals from firm characteristics are often updated once per year meaning that retraining of the models could be performed with lower frequency (see Gu et al., 2018).

4.1 Forecast Evaluation

To compare the predictive performance of each individual machine learning methodology we first calculate the out-of-sample Mean Squared Prediction Error (MSPE), i.e.,

$$MSPE_s^{(n)} = \frac{1}{T - t_0 - 1} \sum_{t=t_0}^{T-1} \left(xr_{t+1}^{(n)} - \widehat{xr}_{s,t+1}^{(n)} \right)^2 \tag{11}$$

where $\widehat{xr}_{s,t+1}^{(n)}$ is the one-step ahead forecast of the bond excess returns for maturity n and model $\mathcal{M}_s$, and t_0 is the date of the first prediction. For the sample that ends in 2008:12, the first forecast is generated in 1994:10. For the sample that includes the financial crisis, the first forecast is generated in 2000:05.

In addition to the MSPE we compare the forecasts obtained from each methodology to a naive prediction based on the historical mean of bond excess returns. In particular, we calculate the out-of-sample predictive R^2 as suggested by Campbell and Thompson (2007). The R^2_{oos} is akin to the in-sample R^2 and is calculated as

$$R^2_{oos} = 1 - \frac{\sum_{t_0=1}^{T-1} \left(xr_{t+1}^{(n)} - \widehat{xr}_{s,t+1}^{(n)} \right)^2}{\sum_{t_0=1}^{T-1} \left(xr_{t+1}^{(n)} - \overline{xr}_{t+1}^{(n)} \right)^2} \tag{12}$$

where $\overline{xr}_{t+1}^{(n)}$ is the one-step ahead prediction error obtained based on the historical mean. When $R^2_{oos} > 0$, the predictive regression model has better MSPE than the benchmark historical average returns. We follow Gu et al. (2018) and implement a pairwise test as proposed by Diebold and Mariano (1995) (DM) to compare the predictions from different models. Diebold and Mariano (1995) show that the asymptotic normal distribution can be a very poor approximation of the test's finite-sample null distribution. In fact, the DM test can have the wrong size, rejecting the null too often, depending on the sample size and the degree of serial correlation among the forecast errors. To address this issue, we adjust the DM test by making a bias correction to the test statistic as proposed by Harvey et al. (1997).

5 An Empirical Study of US Treasury Bonds

5.1 Bond Return Predictability and the Yield Curve

We start by forecasting excess returns of Treasury bonds with the yield curve. A classical benchmark is represented by a principal component regression, reported here for reader convenience:

$$E_t\left[xr_{t+1}^{(n)}\right] = \hat{\alpha} + \hat{\boldsymbol{\beta}}^\top \boldsymbol{x}_t \ ,$$

i.e. one-year holding period excess returns are regressed on principal components of the Treasury term structure, i.e. $\boldsymbol{x}_t = \left[PC_{1,t} \ \ldots, PC_{k,t}\right]$, where the latter are extracted as in Eq. (3). We either use the first three or the first five PCs. The case with five PCs essentially corresponds to the setting in Cochrane and Piazzesi (2005), where excess returns are regressed on a linear combination of short-rate, $y_t^{(1)}$, and four forward rates for loans between $t+n-1$ and $t+n$, $f_t^{(n)}, n = 2, \ldots, 5$.

Panel A of Table 1 displays the out-of-sample MSPE and the R^2_{oos} for two benchmarking principal component regressions. The sample period is from 1964:01 to 2008:12 and does not includes the extensive period of unconventional monetary polices implemented in the aftermath of the great financial crisis of 2008/2009. In addition, Panel A reports the predictive performance of two alternative data compression methodologies, namely partial least squares and an autoencoder.

[Insert Table 1 about here]

The first two rows show the predictive performance of principal component regressions for the case with $k = 3$ and $k = 5$ principal components, respectively. The result are quite disappointing with the predictive R^2 being solidly in the negative region across different maturities. A more parsimonious representation with only the first three PCs significantly outperforms the predictive regression with the five yields. Alternative data compression strategies such as PLS and autoencoding show a similar performance to the case with three PCs. Although the results from PLS might be somewhat surprising being a complex, albeit linear, supervised learning method, the performance from the autoencoder has to be expected since the linear activation function of the autoencoder makes it equivalent to PCA (see Appendix A.1 for a formal discussion).

Panel B displays the results from various configurations of linear penalized regressions in addition to a standard OLS. Few comments are in order. First, as expected the performance of a simple linear model which takes forward rates as inputs is equivalent to a PCR that includes all five principal components. Ridge regression also performs poorly out-of-sample with constantly negative predictive R^2_{oos} across bond maturities. This is somewhat expected. In Appendix B we show formally that ridge regression can be seen as a smooth version of PCRs in which the smallest singular values of the sample variance-covariance matrix are penalized the most. The third and fourth row of Panel B show that sparse modeling may help to improve the forecasting performance of the current term structure. The R^2_{oos} for both the lasso and elastic-net are only mildly negative.

Panel C shows the results obtained by introducing non-linearity in the prediction through non-linear activation functions and hidden layers, i.e., neural networks, and through both boosted regression trees and random forests. Few interesting features emerge. First, supervised learning produces substantial gains in terms of MSPE and R^2_{oos}. That is the forecasting accuracy sensibly improves by comparing, say, a shallow neural network vs. a principal component regression. For instance, for the 2-year bond maturity, the out of sample MSPE of a shallow learner with five neurons and a single hidden layer is 28% (=1.94/2.69) lower than a PCR based on the first five principal components. When we look at the longer 5-year maturity bond, this gain remains substantial with a MSPE which is about 32% (=17.8/26.17) lower than the one obtained from PCR.

Second, non-linearity matters. For instance, for the 2-year bond maturity, the MSPE of a two-layer NN with three neurons is 6% and 7% lower than the MSPE obtained by using lasso and elastic-net, respectively. Such gain increases to 14% and 16% for a 5-year bond maturity. The outperformance of NNs is even larger when we compare NN to principal component regression in which squared PCs are added into the model (see row 3 and 4 in panel A). This result highlights that whereas the 3 PCs explain most of the cross-sectional variability of yields, it is not guaranteed that a linear combination of the yields provide the best forecast of bond returns. It is this latter fact that is exploited by a non-linear neural network, leading to its outperformance.

Third, the depth of the network impacts the ability to predict future excess bond returns. For instance, a deep (four-layer) neural network improves the accuracy of the forecast by around 10% on average across maturities with respect to a shallow (two-layer) learner. As a matter of fact, by comparing the predictive R^2_{oos}, the outperformance of a deep NN relative to a shallow structure is evident. Finally, whereas the results from the neural networks hint that non-linearities are important to improve the measurement of expected bond excess returns based on the current term-structure, both a boosted regression tree and random forests do not show a similar result. Specifically, at least in the context of this application, both regression trees and random forests do not significantly improve the out-of-sample performance with respect to sparse linear regression models.

The third row of Panel C represents an interesting case. In their influential paper, Cochrane and Piazzesi (2005) conclude that lags of forward rates (dated $t-1$ and earlier) contain information about the excess returns that is not in month t forward rates. As Cochrane and Piazzesi (2005) note, this result is inconsistent with the logic that the time-t term structure contains all information relevant to forecasting future yields and excess returns (c.f. Eq. (2)). We therefore ask whether the flexibility of NN can help reconcile the theoretical assumption that yields at time t already incorporate all information about the term structure that is needed to understand bond risk premia. To this end, the third row in Panel C reports the results obtained by feeding the NN with the five forward rates at time t *and* lagged forward rates from time $t-11$ to $t-1$. By comparing the third row to results from deeper neural networks like the NN with three layers (row 6), it is clear that we cannot improve upon a neural network that uses just the month-t forward rates. In other words, consistent with Eq. (2), we cannot find extra information that is contained in lagged values of the yield as opposed to just using the time-t term structure.

We now extend the results in Table 1 by including the period from 1964:01 to 2016:12 which covers the aftermath of the great financial crisis. As before forecasts are generated recursively and the initial training and validation samples consist of 70% of the data, that is the first prediction is generated as of 2000:05. By including an extensive period of unconventional monetary policy measures and interest rates at the zero lower bound we aim to further investigate the ability of machine learning methods to improve the measurement of bond risk premia. Table 2 reports the results.

[Insert Table 2 about here]

At a broad level the results are consistent with Table 1. As a matter of fact, although the predictive performance tends to deteriorate across models, we still find that sparse linear regressions tend to improve upon data compression methodologies, and that in turn non-linearity matters as shown by significantly lower MSPE obtained from neural networks. Panel A provides evidence that both principal component regressions and PLS massively underperform a forecast based on the conditional mean as proved by large and negative R^2_{oos}s. Panel B shows that the performance of sparse linear regression models remains substantially the same.

The empirical evidence provided in Panel C essentially confirms that (1) we can improve bond risk premia measurement by acknowledging that the function $g(\boldsymbol{y}_t; N)$ in Eq. (2) can be non-linear, and (2) such improvement is a function of the neural network specifications, i.e., a deeper neural network with three hidden layers tend to outperform a shallow learner with a single set of hidden neurons. Interestingly, the performance of random forests substantially improves over the larger sample. With the only exception of the 2-year bond, all R^2_{oos} are positive; in fact, random forests slightly outperform a shallow NN with a single hidden layer. Consistent with studies that show that non-linearities matter in the zero-lower bound period (see Fernández-Villaverde et al., 2015,Bauer and Rudebusch, 2016, and Gust et al., 2017, among others), we find evidence that a deeper network is needed to better proxy for the non linear mapping between bond returns and yields during the full sample. Indeed, when the financial crisis is excluded, see Table 1, we see that a NN with three layers and five nodes achieves the best performance for 4- and 5-year maturities, and is at par with a NN with four layers (and pyramidal nodes) at 3-year maturity. On the other hand, Table 2 shows that when the crisis is included then the NN with four layers (and pyramidal nodes) achieve performance that are better than a NN with three layers throughout the maturity structure.

Tables 1-2 provide quantitative evidence that explicitly accounting for non-linearities can significantly improve the out-of-sample performance of a regression-based forecast of bond excess returns based on the current term structure. We now implement a pairwise test of predictive accuracy both for the short and the long sample based on test proposed by Diebold and Li (2006) and extended by Harvey et al. (1997).

Figure 4 reports in gray the pairs which have a performance difference which is statistically significant at a conventional 5% (or lower) significance level, and in black the pairs which have either a lower significance or no significance at all, i.e., a p-value greater than 0.10.

[Insert Figure 4 about here]

Top panel reports the test results for the sample implementation until 2008:12. Notice the figure reports the significance of the performance gaps while the direction can be inferred by looking at Tables 1-2. The first conclusion is that there is some clustering of performances across methodologies. For instance, few specifications of neural networks tend to perform similarly, although a deep neural network with the number of nodes descending from the input to the output layer clearly show a performance that is statistically higher than all competing strategies. Again, data compression methodologies tend to perform similarly, as shown by a cluster of non-significant DM statistics from "PCAsq3" (a principal component regression with $k = 3$ and factors squared) to "PLS3" (a PLS with three latent factors). The picture that emerges for the longer sample (bottom panel) is slightly different. The performance among shallow neural networks is almost indistinguishable. On the other hand, the performances across classes of machine learning methods tend to markedly differ.

5.2 Bond Return Predictability and Macroeconomic Variables

Next, we consider the set-up where information embedded in the yield curve does not subsume information contained in macro variables. In this case the relevant benchmark regression is given by Eq. (4), reported here for reader convenience:

$$E_t\left[xr_{t+1}^{(n)}\right] = \hat{\alpha} + \hat{\boldsymbol{\beta}}^\top \boldsymbol{x}_t + \hat{\boldsymbol{\gamma}}^\top \boldsymbol{F}_t \ ,$$

where the factors F_t have now the potential to serve as the model's state vector beyond yields only. To ensure comparability with the existing literature we adopt as a benchmark the specification proposed by Ludvigson and Ng (2009), whereby $\boldsymbol{F}_t$ is a subset of the first eight principal components extracted from a large cross section of macroeconomic variables and $\boldsymbol{x}_t$ represents a linear combination of forward rates as proposed by Cochrane and Piazzesi (2005), a.k.a. the CP factor.

Similar to the forecasting of bond excess returns based on the yield curve only, we estimate all machine learning methods for both a short sample which does not include the aftermath of the financial crisis and an enlarged sample until 2016:12. Table 3 reports the results for the short sample.

[Insert Table 3 about here]

Panel A displays the benchmark specification proposed by Ludvigson and Ng (2009) (LN henceforth). In addition, we report the results from two alternative data compression methods, namely PLS and auto-encoding. By looking at Panel A we confirm results in the literature (see Duffee, 2011b; Joslin et al., 2014) that there is information in macro variables that is not embedded in the yield curve, and yet it is useful for forecasting bond excess returns. As a matter of fact, the R^2_{oos} obtained by adding $\boldsymbol{F}_t$ to the CP factor are much higher than the ones obtained by simply using a linear combination of yields (c.f., first row of Panel A in Table 1).

The performances of both PLS and the auto-encoder are substantially lower than that of LN, especially for shorter-term maturities. This is not surprising. The specification proposed by Ludvigson and Ng (2009) includes also non linear transformations of the macro PCs, i.e., the cubic function in the first estimated factor. Neither PLS nor auto-encoding are designed to embed non-linearities. This is a first hint that non-linear features in macroeconomic information could play a significant role for the forecasting of bond excess returns.

Panel B shows the results from two alternative implementations of sparse and regularized linear regressions. The first implementation employs directly the CP factor as additional regressor; this specification ensures a closer comparability with respect to Ludvigson and Ng (2009). Instead, the second implementation treats the whole set of forward rates as additional regressors with respect to macroeconomic variables. Two interesting aspects emerge. First, employing unrestricted forward rates generally leads to better out-of-sample performance. Second, elastic-net substantially outperform both the ridge regression and the lasso with a R^2_{oos} that turns from negative to positive for bonds with longer maturities.

Turning to non-linear machine learning methods, Panel C in Table 3 shows the results obtained from three alternative specifications of neural networks. The first specification can be thought of as a "hybrid" modeling framework in the sense that forward rates are simply included as an additional predictor in the output layer. Figure 5 shows a visual representation of such network structure.[20]

[Insert Figure 5 about here]

Such structure simulates the idea of Ludvigson and Ng (2009) in which the latent factors $\boldsymbol{F}_t$ are extracted from a large cross-section of macroeconomic variables and the forward rates are included as a linear combination as proposed by Cochrane and Piazzesi (2005). We label this structure where forward rates have been pre-processed a "hybrid neural network". The second specification ensembles two separate networks at the output layer level: one for the forward rates and one for the

[20]The green circles represent the input variables, that is the macroeconomic variables. The purple circles represent the fully connected hidden nodes. The red circles represent the output variables, that is the bond excess returns across maturities. The yellow circle represents the additional predictor, that is the linear combination of forward rates.

macroeconomic variables. As a result, the possibly non-linear interactions among forward rates and macroeconomic variables are modeled separately. The third specification entails a collection of networks, one for each group of macroeconomic variables, that are trained in parallel and ensembled at the output layer level. The groups of macroeconomic variables are constructed according to the classification provided by McCracken and Ng (2016). We call this structure "groups ensembling". To the best of our knowledge, these specifications have not been proposed before in the empirical asset pricing literature.

The empirical evidence confirms that non-linearity matters. For instance, a NN with CP as separate predictor and a single hidden layer achieves a MSPE which is 23% (=1.50/1.96) lower than the best performing elastic net specification for a 2-year bond excess returns. The improvement remains substantial at longer maturities. For example, for a 4-year bond maturity, the out-of-sample MSPE obtained from a simple two-layer neural network is 23% (=10.20/13.28) lower than the best performing elastic net. Similarly, the MSPE of the neural network is 24% lower (=15.32/20.11) when we look at the 5-year maturity bond.

The performance of a network that ensembles two separate structures, one for the macroeconomic variables and one for the forward rates stands out. For instance, a shallow network that separates the modeling of forward rates and macro factors improves the out-of-sample MSPE by 6% on average across maturities.

Interestingly, as far as the short sample is concerned, Panel C shows that the depth of the network plays only a marginal role. When focusing on the hybrid NN, we observe that a shallow network outperforms both the NN with three- and four layers. When focusing on the NN that ensembles two separate structures, a three-layer NN performs generally better than a NN with four layers. Despite the fact that a shallow network may perform better than deep NNs in this sample period, we observe that a careful choice of the structure has a great impact on the performance. E.g. comparing the two-layer hybrid NN with a two-layer groups ensembling model, we observe that the latter performs better across maturities.

Panel C in Table 3 also shows that the performances of both boosted regression trees and random forests improve substantially when using a large panel of macroeconomic information. In fact, the random forest performs better than the hybrid network in which the forward rates are forced to enter linearly in the model, and at par of a three-layer NN that ensembles two separate structures. Opposite to the conclusion reached for the sample that excludes the zero lower bound, restricting the linear combination of forward rates is beneficial for the elastic net specification in the long sample.

Table 4 shows the results from the extended sample which covers the period from 1964:01 to 2016:12. Some of the results confirm the discussion above. Dense modeling such as data compression techniques and ridge regression tends to perform poorly out of sample. Sparse modeling performs marginally better, especially when both regularization and shrinkage are considered, i.e., elastic net regression.

[**Insert Table 4 about here**]

As far as neural networks are concerned, the results slightly change. Overall, the depth of the network now plays a much more significant role in the full sample. Focusing on the hybrid structure, Table 4 shows that a four-layer network is clearly superior to a shallow network. Also, when focusing on NNs that ensemble two separate structures instead, Table 4 shows that a NN with four layers outperforms a NN with three layers (thus reversing the ordering shown in Table 3). Nevertheless, a shallow neural network with groups ensembling compares favorably against a deeper network with macro and forward rates modeled separately.

In addition, for short 2- and 3-year maturities, random forests perform almost on par with a deep neural network setting in which the CP factor is added separately. However, when compared to a deep ensemble network, random forests tend to substantially underperform as the bond maturity increases.

Figure 7 shows the pairwise performances according to the DM test. The top panel shows the results for the short sample. The results suggest that the differences in the performance are broadly significant and no evident clusters of performance similarity can be observed.

[**Insert Figure 7 about here**]

The bottom panel shows the DM test results for the longer sample. The picture that emerges is rather different with the statistical differences across models being weaker. There seems to be no statistical difference in the performance between some of the NN specifications. In particular, within the class of NNs that ensemble two separate structures, we cannot detect statistically significant differences across shallow and deep networks. Similarly, the performance of PCA regressions are often not statistically different than those of some penalized regressions.

5.2.1 Relative Importance of Macroeconomic Variables

Although a structural interpretation of the results obtained from neural networks is somewhat prohibitive, it is instructive to understand what variables might be driving the results shown in Tables 3-4. To this end, we investigate the marginal relevance of single variables based on the partial derivative of the target variable with respect to the sample average of each input. That is, we calculate $\frac{\partial xr_{t+1}^{(n)}}{\partial y_{it}}\Big|_{y_{it}=\overline{y}_i}$, where $\overline{y}_i$ represents the in-sample mean of the input variable i. As a matter of fact, the partial derivatives are similar to the betas of a simple linear regression. Indeed, these partial derivatives effectively represent the sensitivity of the output to the ith input, conditional on the network structure and the average value of the other input variables (see Dimopoulos et al., 1995).

Alternative methodologies have been proposed in the literature. For instance, Sung (1998) proposed a stepwise method that consists of adding or rejecting one input variable at a time and noting the effect on the Mean Squared Error (MSE). Based on the changes in MSE, the input variables can be ranked according to their importance in several different ways depending on different arguments. The major drawback of stepwise methods is that at each step of the selection algorithm a new model is generated that requires training. For an exercise like ours, in which there are more than 120 input variables and forecasts are generated recursively this could be computationally expensive. Alternatively, Lek et al. (1995) and Lek et al. (1996) propose to study each input variable successively when the others are blocked at fixed values. Depending on the users' needs one can set single inputs to their sample mean, their median, or simply to zero (see, e.g., Gu et al., 2018). The relative importance of a given input is then investigated by looking at the changes in the MSE.

Figure 8 shows the relative importance of each input variable calculated based on the gradient of the output with respect to each input, where the gradient is evaluated at the in-sample mean of the value of the input. For the ease of interpretation we report the rescaled value of the gradient such that a value of zero means that a variable is least relevant and a value of one means that a variable is the most important in relative terms. The gradient-at-the-mean is calculated for each time t of the recursive forecasting, then averaged over the out-of-sample period. The results are further averaged across the bond maturities.

[Insert Figure 8 about here]

The benchmark network specification is a deep neural network with four layers (output layer plus three hidden layers). Macroeconomic variables and forward rates are modeled separately through ensembling at the output layer level.

The left panel shows the results for the short sample period from 1964:01 to 2008:12. Few interesting facts emerge by looking at the most relevant variables. For instance, the S&P composite index, the effective federal funds rate, and the spread between Aaa-rated corporate bonds, rank in the top three positions. Scrolling down the list, variables that pertain to the housing market (e.g., housing starts), inflation (e.g., CPI: all items), interest rate spreads (e.g., 10- and 5-year treasury minus the fed funds rate) all occupy the top spots in the relative importance ranking.

The right panel shows the results for the long sample period from 1964:01 to 2016:12, which fully includes the increase in the sub-prime mortgage defaults and the following great financial crisis of 2008/2009. Despite few nuances, the ranking remains largely similar to the short sample. For instance, the spread between Moody's Aaa corporate bond yields and the fed funds rate still represents the third most relevant input variable. The S&P composite index goes from the top to the fourth slot out of 128, that is, still represents a valuable source of information to forecast bond excess returns.

The results of Figure 8 are averaged out across maturities. However, in principle the effect of predictors can be heterogeneous over the term structure. Figures 9 and 10 show that this is indeed the case. Figure 9 shows the results for the short sample.

[Insert Figure 9 about here]

The ranking for the 2-year (left panel) bond excess returns is somewhat similar to the left panel of Figure 8: the S&P composite index and the spread between Moody's Aaa corporate bond yields and the fed funds rate rank first and third, respectively. The effective fed funds rate ranks sixth. Sector-specific measures of inflation, such as CPI apparel and CPI commodities also occupy the top of the list. The right panel shows the results for the 5-year bond excess returns. Here the picture that emerges is substantially different. Except for the federal funds rate which ranks fifth, the variables related to the housing sector and inflation substantially dominate the ranking. Notice that, however, the differences in relative importance are less strong as shown by a less steep decrease after the top positions.

The heterogeneity in the driving factors of expected excess returns between the short- and the long-end of the yield curve is confirmed by looking at the relative importance over the long sample. Figure 10 shows the results. Again, the spread between Aaa-rated corporate and treasury bond yields is particularly relevant, together with the S&P composite index and measures of inflation (e.g., CPI apparel). However, also variables related to economic growth (e.g., personal consumption expenditures) turn out to be highly relevant when the sample period includes the great financial crisis.

[Insert Figure 10 about here]

The fact that we are dealing with a large of panel of input variables makes it difficult to detect any systematic pattern in the economics of expected future bond excess returns. We address this issue by calculating the relative importance from the gradients averaged for each class of input variables as labeled in McCracken and Ng (2016). This allows to give some indication on the which economic category dominates. Figure 11 shows the results.

[Insert Figure 11 about here]

The top row displays the results for the sample period that goes from 1964:01 to 2008:12. The left (right) column reports the results for the 2-year (5-year) bond excess returns. In order of importance, we find that the network performance relies predominantly on (commodity and consumer) prices, and measures of the aggregate stock market. We also find some evidence that the neural network loads on measures of consumption and new manufacturing orders. Finally, we find little relation to unemployment and aggregate credit conditions. Interestingly, each category has a differential impact across maturities. For a 5-year maturity variables more correlated with economic growth such as consumption, output and income have greater or equal relevance to financial variables. On the other hand, (commodity and consumer) prices tend to have a negligible effect at longer maturities.

The bottom row of Figure 11 reports the results for the sample that goes from 1964:01 to 2016:12. The left panel shows that variables related to the aggregate stock market tends to dominate for 2-year bonds. We also observe that the impact of variables related to economic growth, such as labor market, output and income, and consumption tend to be significantly higher for the long end of the curve. Finally, variables related to the future prospects of output growth, such as orders and inventories, are not present at the short end of the curve but tend to become dominant for longer maturities.

6 Understanding the Performance of Neural Networks

6.1 Forecasting Principal Components of Yields and Macro Variables

In this section we provide an heuristic interpretation of the performance of the neural networks outlined above based on the Campbell and Shiller (1991) accounting identity. Such identity posits that forecasts of future yields (or their principal components) using current yields are necessarily also forecasts of expected log returns to bonds (see, e.g., Duffee, 2013).[21] Thus, to investigate the main drivers of the increased predictive ability of the latent factors extracted by the NN, we now first move to forecast the year-on-year changes in the first three latent factors extracted from the cross-section of forward rates by a standard principal component analysis. We denote these principal components as $PC_{1,t}, PC_{2,t}, PC_{3,t}$. The auxiliary forecasting regressions are

$$PC_{i,t+12} - PC_{i,t} = b_0 + \boldsymbol{b}_1^\top \mathcal{P}_t + \boldsymbol{b}_2^\top \boldsymbol{x}_t + \epsilon_{i,t+1} \qquad \text{for} \quad i = 1, 2, 3 \quad ,$$

where we stack the first three principal components of the term structure in the vector $\mathcal{P}_t$ and denote by $\boldsymbol{x}_t$ the hidden factors extracted by the NN (see figure 3). Panel A of Table 5 reports the in-sample R^2 of a predictive regression where the dependent variables are the changes in the level, slope, and curvature of the term structure and the independent variables are the same principal components plus the hidden factors extracted from a set of shallow and deep neural networks.[22]

[Insert Table 5 about here]

The first row provides a benchmark based on the sole vector $\mathcal{P}_t$. As noted in Duffee (2011a, 2013), there is weak evidence that changes in the first principal component (level) are forecastable ($R^2 = 16.8\%$ being the lowest) whereas the slope and curvature are unquestionably forecastable with 28.1% (58.8%) of the variation in slope (curvature) that is predictable.

Looking at the second and third row we observe that the factors extracted from neural networks contribute substantially to the predictability of the level. On the other hand, the statistical evidence

[21] The accounting identity is

$$y_{t+1}^{(n-1)} - y_t^{(n)} = \frac{1}{n-1}\left(y_t^{(n)} - y_t^{(1)}\right) - \frac{1}{n-1} xr_{t+1}^{(n)} \; .$$

[22] We do not consider an enlarged vector $\mathcal{P}_t = (PC_{1,t}, \ldots, PC_{5,t})$ since Duffee (2013) already concludes that principal components other than the first three do not contribute much to forecasts of slope and curvature.

for slope and curvature forecasts - after controlling for the standard three principal components - is weak. We conclude that there is substantial information in the time-t term structure not only about future values of slope and curvature, but also about the level. Standard principal components are not entirely able to extract all the relevant information about the level. Both a shallow learner and a deep neural network are successful in extracting such information about the future level of the curve, an information which leads to excess returns being more predictable out-of-sample. The evidence for the large sample seems to be consistent; in fact, while the predictive ability of the three PCs for the slope decreases from 28.08% to 26.38%, the incremental explanatory power of the neural networks increases and helps raising the R^2 to about 30%.

We now investigate whether the factors extracted from macroeconomic variables have explanatory power for the year-on-year changes in the level, slope and curvature of the yield curve. Panel B of Table 5 reports the results. The first row provides in-sample statistical evidence that changes in the first three principal component are forecastable by using macroeconomic information. This is in line with existing literature (see, e.g., Diebold and Li, 2006). The last three rows show that the factors extracted from neural networks have a substantial incremental predictive power for both samples. In fact, the inclusion of the factors from the NNs produce in-sample R^2 which are, on average, five times higher than the ones obtained using only macro PCAs. Importantly, the factors extracted from the NNs not only contribute to the ability to predict the level of the yield curve, but also the slope. This is consistent with the idea that the slope of the yield curve is related to the state of the economy, and a NN is able to extract the relevant information from the large set of macroeconomic variables used.

Finally, we investigate whether the ability of the hidden factors extracted from the NNs to forecast bond returns is related to the the increasing ability to forecast future changes in aggregate macroeconomic conditions. In particular, Panel C of Table 5 reports the results for the forecasts of the first three principal components extracted from a panel of data with 128 individual macro and financial series. Ludvigson and Ng (2009) showed that each PC is particularly related to a handful of underlying variables in our macro panel dataset. For example, the first PC loads heavily on measures of employment and production, and on measures of capacity utilization and new manufacturing orders. The second PC is highly correlated to interest rate spreads. The third loads most heavily on measures of inflation and price pressure. As a results, by testing the additional explanatory power of the factors extracted from NNs we implicitly test their ability to provide information about macroeconomic variables related to the term structure, such as measures of real economic activity and price indexes.

The first row provides in-sample statistical evidence that changes in the first three principal component are forecastable. The last three rows show that the factors extracted from neural networks have incremental predictive power for all the three macro PCs with respect to the latent factors extracted by PCA. Interestingly, these results are in line with those in Figure 8 in which we show the relative importance of groups of predictors in forecasting bond excess returns. In fact the most relevant categories highlighted by our exercise correspond to the very same categories associated with the first three PCs.

6.2 Forecasting Macroeconomic Time Series

We note that any labeling of the factors extracted from the panel of macroeconomic variables maybe hard to interpret. As pointed out by Boivin and Ng (2006a) factors identification is influenced by the amount of information used, i.e. how many and which macroeconomic variables are considered; indeed it is often the case that none of the factors correspond exactly to precise economic concepts such as industrial production, unemployment or inflation. To address this issue we extend and complement the results of Panel C in Table 5 by investigating the forecasting properties of the factors extracted from the NNs when the dependent variable is the year-on-year growth rate of measures of real economic activity (total industrial production; real personal income less transfers; real manufacturing and trade sales; and number of employees on nonagricultural payrolls) and price indexes (the consumer price index; the personal consumption expenditure implicit price deflator; the consumer price index (CPI) less food and energy; and the producer price index for finished goods). We follow Stock and Watson (2002a) and consider a simple AR(1) process as a benchmark methodology. Panel A of Table 6 reports the results for the short sample.

[Insert Table 6 about here]

Two interesting facts emerge: first, the addition of the latent NN factor to the AR(1) significantly decreases the MSE. This is true for each macroeconomic time series series, from industrial production to the production price index (PPI). Of particular interest are the results on forecasting year-on-year inflation. The NNs significantly improves the results obtained from the AR(1) in isolation. Second, as far as forecasting macroeconomic variables is concerned, the depth of the network does not seem to play a significant role. Indeed, the outperformance with respect to the AR(1) alone remains virtually unchanged by increasing the number of hidden layers. Panel B reports the results for the longer sample that goes from 1964:01 to 2016:12. The conclusions are broadly similar to those in Panel A. The addition of the latent factors from the NNs substantially decreases the MSE with respect to the AR(1). Yet, the depth of the neural network does not play virtually any role.

Overall our results point to the fact that the NN factor extracted from a large set of macroeconomic variables are useful to forecast not only bond returns but also macroeconomic variables related to the term structure, such as measures of real economic activity and price indexes.

7 Conclusion

In this paper we have shown that machine learning techniques can lead to more accurate out-of-sample forecasts of bond excess returns. In particular, we have provided empirical evidence that accounting for non-linearity in the functional mapping between bond excess returns and the state variables substantially improves upon the classical two-step approach whereby one first compresses the cross-section of yields and/or macroeconomic variables into a low-dimensional vector (principal component analysis), and then uses this vector to forecast future bond excess returns. Instead, machine learning techniques – and neural networks in particular – allow to leave the functional form of the mapping between bond risk premia and the investors' information set unspecified. This is in line with standard bond pricing theory which grants only the existence of an unknown, hence not necessarily linear, returns–predictors function.

By looking at the empirical results two conclusions emerge. First, non-linearity matters. This is true both in the context of yield-only forecasting regressions as well as in the context of bond returns regressions whereby yield-based variables are augmented with macroeconomic information. Second, a deep neural network with non-linear activation functions substantially outperforms both dense and sparse linear regression frameworks as well as data compression techniques such as principal component regressions and partial least squares. These results represent a first attempt to assess the benefit of using machine learning in a well-designed prediction framework for bond returns.

References

Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski (1985) "A learning algorithm for boltzmann machines," *Cognitive Science*, Vol. 9, No. 1, pp. 147 – 169.

Ang, Andrew and Geert Bekaert (2006) "Stock return predictability: Is it there?" *The Review of Financial Studies*, Vol. 20, No. 3, pp. 651–707.

Arlot, Sylvain, Alain Celisse et al. (2010) "A survey of cross-validation procedures for model selection," *Statistics surveys*, Vol. 4, pp. 40–79.

Bai, Jushan and Serena Ng (2003) "Determining the Number of Factors in Approximate Factor Models," *Econometrica*, Vol. 70, No. 1, pp. 191–221.

——— (2006) "Confidence Intervals for Diffusion Index Forecasts and Inference for Factor-Augmented Regressions," *Econometrica*, Vol. 74, No. 4, pp. 1133–1150.

——— (2008) "Forecasting economic time series using targeted predictors," *Journal of Econometrics*, Vol. 146, No. 2, pp. 304–317, October.

Bauer, Michael D and Glenn D Rudebusch (2016) "Monetary policy expectations at the zero lower bound," *Journal of Money, Credit and Banking*, Vol. 48, No. 7, pp. 1439–1465.

Bianchi, Daniele, Monica Billio, Roberto Casarin, and Massimo Guidolin (2018) "Modeling systemic risk with Markov switching graphical SUR models," *Journal of Econometrics, forthcoming.*

Bishop, Christopher M (1995) "Regularization and complexity control in feed-forward networks."

Bishop, Chris, Christopher M Bishop et al. (1995) *Neural networks for pattern recognition*: Oxford university press.

Boivin, Jean and Serena Ng (2006a) "Are more data always better for factor analysis?" *Journal of Econometrics*, Vol. 132, No. 1, pp. 169–194.

——— (2006b) "Are more data always better for factor analysis?" *Journal of Econometrics*, Vol. 132, No. 1, pp. 169–194.

Breiman, Leo (2001) "Random forests," *Machine learning*, Vol. 45, No. 1, pp. 5–32.

Breiman, Leo, Jerome Friedman, Charles J. Stone, and R.A. Olshen (1984) *Classification and Regression Trees*: Taylor & Francis.

Burns, Arthur F. and Wesley C. Mitchell (1946) *Measuring Business Cycles*: National Bureau of Economic Research, Inc.

Campbell, John Y. and Robert J. Shiller (1991) "Yield Spreads and Interest Rate Movements: A Bird's Eye View," *Review of Economic Studies*, Vol. 58, No. 3, pp. 495–514.

Campbell, John Y and Samuel B Thompson (2007) "Predicting excess stock returns out of sample: Can anything beat the historical average?" *The Review of Financial Studies*, Vol. 21, No. 4, pp. 1509–1531.

Cieslak, Anna and Pavol Povala (2015) "Expected Returns in Treasury Bonds," *Review of Financial Studies*, Vol. 28, No. 10, pp. 2859–2901.

Cochrane, John H (2007) "The dog that did not bark: A defense of return predictability," *The Review of Financial Studies*, Vol. 21, No. 4, pp. 1533–1575.

Cochrane, John H. and Monika Piazzesi (2005) "Bond Risk Premia," *American Economic Review*, Vol. 95, No. 1, pp. 138–160, March.

Cook, R Dennis et al. (2007) "Fisher lecture: Dimension reduction in regression," *Statistical Science*, Vol. 22, No. 1, pp. 1–26.

Cooper, Ilan and Richard Priestley (2008) "Time-varying risk premiums and the output gap," *The Review of Financial Studies*, Vol. 22, No. 7, pp. 2801–2833.

Cybenko, George (1989) "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, Vol. 2, No. 4, pp. 303–314.

De Mol, Christine, Domenico Giannone, and Lucrezia Reichlin (2008) "Forecasting using a large number of predictors: Is Bayesian shrinkage a valid alternative to principal components?" *Journal of Econometrics*, Vol. 146, No. 2, pp. 318–328.

Diaconis, P. and M. Shahshahani (1984) "On Nonlinear Functions of Linear Combinations," *SIAM Journal on Scientific and Statistical Computing*, Vol. 5, No. 1, pp. 175–191.

Diebold, Francis X and Canlin Li (2006) "Forecasting the term structure of government bond yields," *Journal of econometrics*, Vol. 130, No. 2, pp. 337–364.

Diebold, Francis X and Robert S Mariano (1995) "Comparing predictive accuracy," *Journal of Business & economic statistics*, Vol. 20, pp. 134–144.

Dimopoulos, Yannis, Paul Bourret, and Sovan Lek (1995) "Use of some sensitivity criteria for choosing networks with good generalization ability," *Neural Processing Letters*, Vol. 2, No. 6, pp. 1–4.

Duffee, Greg (2011a) "Forecasting with the term structure: The role of no-arbitrage restrictions," Economics Working Paper Archive 576, The Johns Hopkins University,Department of Economics.

Duffee, Gregory R. (2011b) "Information in (and not in) the Term Structure," *Review of Financial Studies*, Vol. 24, No. 9, pp. 2895–2934.

Duffee, Gregory (2013) *Forecasting Interest Rates*, Vol. 2 of Handbook of Economic Forecasting, Chap. 0, pp. 385–426: Elsevier.

Fama, Eugene F and Robert R Bliss (1987) "The information in long-maturity forward rates," *The American Economic Review*, pp. 680–692.

Feng, Guanhao, Stefano Giglio, and Dacheng Xiu (2017) "Taming the Factor Zoo," Fama-Miller Working Paper 24070, Chicago Booth.

Feng, Guanhao, Jingyu He, and Nicholas G Polson (2018) "Deep Learning for Predicting Asset Returns," *arXiv preprint arXiv:1804.09314*.

Feng, Guanhao, Nicholas Polson, and Jianeng Xu (2018) "Deep Factor Alpha," Chicago Booth Research Paper 23527, Chicago Booth.

Fernández-Villaverde, Jesús, Grey Gordon, Pablo Guerrón-Quintana, and Juan F Rubio-Ramirez (2015) "Nonlinear adventures at the zero lower bound," *Journal of Economic Dynamics and Control*, Vol. 57, pp. 182–204.

Forni, Mario and Lucrezia Reichlin (1996) "Dynamic Common Factors in Large Cross-Sections," *Empirical Economics*, Vol. 21, No. 1, pp. 27–42.

——— (1998) "Let's Get Real: A Factor Analytical Approach to Disaggregated Business Cycle Dynamics," *The Review of Economic Studies*, Vol. 65, No. 3, pp. 453–473.

Frank, LLdiko E and Jerome H Friedman (1993) "A statistical view of some chemometrics regression tools," *Technometrics*, Vol. 35, No. 2, pp. 109–135.

Freyberger, Joachim, Andreas Neuhierl, and Michael Weber (2017) "Dissecting Characteristics Nonparametrically," CESifo Working Paper Series 6391, CESifo Group Munich.

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

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001) *The elements of statistical learning*, Vol. 1: Springer series in statistics New York, NY, USA:.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani (2010) "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, Vol. 33, No. 1, p. 1.

Friedman, Jerome, Trevor Hastie, Holger Hfling, and Rob Tibshirani (2007) "Pathwise Coordinate Optimization," *The Annals of Applied Statistics*, Vol. 1, No. 2, pp. 302–332.

Funahashi, Ken-Ichi (1989) "On the approximate realization of continuous mappings by neural networks," *Neural networks*, Vol. 2, No. 3, pp. 183–192.

Gargano, Antonio, Davide Pettenuzzo, and Allan Timmermann (2017) "Bond return predictability: Economic value and links to the macroeconomy," *Management Science*.

George, Edward I and Robert E McCulloch (1993) "Variable selection via Gibbs sampling," *Journal of the American Statistical Association*, Vol. 88, No. 423, pp. 881–889.

Geweke, J. (1977) *The Dynamic Factor Analysis of Economic Time Series*: D.J Aigner and A.S. Goldberger, eds. (NorthHolland, Amsterdam).

Giannone, Domenico, Michele Lenza, and Giorgio Primiceri (2017) "Economic predictions with big data: The illusion of sparsity," *Working Paper*.

Giglio, Stefano and Dacheng Xiu (2017) "Inference on Risk Premia in the Presence of Omitted Factors," NBER Working Papers 23527, National Bureau of Economic Research, Inc.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011) "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio (2016) *Deep learning*, Vol. 1: MIT press Cambridge.

Gu, Shihao, Bryan T. Kelly, and Dacheng Xiu (2018) "Empirical Asset Pricing via Machine Learning," Chicago Booth Research Paper 18-04, Chicago Booth.

Gust, Christopher, Edward Herbst, David López-Salido, and Matthew E Smith (2017) "The empirical implications of the interest-rate lower bound," *American Economic Review*, Vol. 107, No. 7, pp. 1971–2006.

Harvey, David, Stephen Leybourne, and Paul Newbold (1997) "Testing the equality of prediction mean squared errors," *International Journal of forecasting*, Vol. 13, No. 2, pp. 281–291.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015) "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

Heaton, J. B., N. G. Polson, and J. H. Witte (2017) "Deep learning for finance: deep portfolios," *Applied Stochastic Models in Business and Industry*, Vol. 33, No. 1, pp. 3–12.

Hornik, Kurt (1993) "Some new results on neural network approximation," *Neural networks*, Vol. 6, No. 8, pp. 1069–1072.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989) "Multilayer feedforward networks are universal approximators," *Neural networks*, Vol. 2, No. 5, pp. 359–366.

Hutchinson, James M, Andrew Lo, and Tomaso Poggio (1994) "A Nonparametric Approach to Pricing and Hedging Derivative Securities via Learning Networks," *Journal of Finance*, Vol. 49, No. 3, pp. 851–89.

Ioffe, Sergey and Christian Szegedy (2015) "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*.

Jarrett, Kevin, Koray Kavukcuoglu, Yann LeCun et al. (2009) "What is the best multi-stage architecture for object recognition?" in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153.

Jones, Christopher S. (2006) "A Nonlinear Factor Analysis of SP500 Index Option Returns," *The Journal of Finance*, Vol. 61, No. 5, pp. 2325–2363.

Joslin, Scott, Marcel Priebsch, and Kenneth J. Singleton (2014) "Risk Premiums in Dynamic Term Structure Models with Unspanned Macro Risks," *The Journal of Finance*, Vol. 69, No. 3, pp. 1197–1233.

Kelly, Bryan and Seth Pruitt (2013) "Market Expectations in the Cross-Section of Present Values," *The Journal of Finance*, Vol. 68, No. 5, pp. 1721–1756.

——— (2015) "The three-pass regression filter: A new approach to forecasting using many predictors," *Journal of Econometrics*, Vol. 186, No. 2, pp. 294–316.

Kelly, Bryan, Seth Pruitt, and Yinan Su (2018) "Characteristics Are Covariances: A Unified Model of Risk and Return," NBER Working Papers 24540, National Bureau of Economic Research, Inc.

Kolmogorov, A. K. (1957) "On the Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of One Variable and Addition," *Doklady Akademii Nauk SSSR*, Vol. 114, pp. 369–373.

Kozak, Serhiy, Stefan Nagel, and Shrihari Santosh (2017) "Shrinking the Cross Section," NBER Working Papers 24070, National Bureau of Economic Research, Inc.

Kuan, Chung-Ming and Halbert White (1994) "Artificial neural networks: an econometric perspective," *Econometric Reviews*, Vol. 13, No. 1, pp. 1–91.

Lee, Herbert K. H. (2004) *Baysian Nonparametrics via Neural Networks*, Vol. 2 of ASA-SIAM Series on Statistics and Applied Probability, Chap. 0, pp. 385–426: ASA-SIAM.

Lek, Sovan, Alain Belaud, Ioannis Dimopoulos, J Lauga, and J Moreau (1995) "Improved estimation, using neural networks, of the food consumption of fish populations," *Marine and Freshwater Research*, Vol. 46, No. 8, pp. 1229–1236.

Lek, Sovan, Marc Delacoste, Philippe Baran, Ioannis Dimopoulos, Jacques Lauga, and Stephane Aulagnier (1996) "Application of neural networks to modelling nonlinear relationships in ecology," *Ecological modelling*, Vol. 90, No. 1, pp. 39–52.

Li, Xiang, Shuo Chen, Xiaolin Hu, and Jian Yang (2018) "Understanding the disharmony between dropout and batch normalization by variance shift," *arXiv preprint arXiv:1801.05134*.

Litterman, R. and J. Scheinkman (1991) "Common Factors Affecting Bond Returns.," *Journal of Fixed Income*, Vol. 1, pp. 54–61, September.

Lo, Andrew (1994) "Neural networks and other nonparametric techniques in economics and finance.," *AIMR Conference Proceedings – CFA Institute*, No. 9, pp. –.

Ludvigson, Sydney C. and Serena Ng (2009) "Macro Factors in Bond Risk Premia," *Review of Financial Studies*, Vol. 22, No. 12, pp. 5027–5067, December.

Masters, Timothy (1993) *Practical neural network recipes in C++*: Morgan Kaufmann.

McCracken, Michael W. and Serena Ng (2015) "FRED-MD: A Monthly Database for Macroeconomic Research," Working Papers 2015-12, Federal Reserve Bank of St. Louis.

McCracken, Michael W and Serena Ng (2016) "FRED-MD: A monthly database for macroeconomic research," *Journal of Business & Economic Statistics*, Vol. 34, No. 4, pp. 574–589.

Messmer, Marcial (2017) "Deep Learning and the Cross-Section of Expected Returns."

Mullainathan, Sendhil and Jann Spiess (2017) "Machine learning: an applied econometric approach," *Journal of Economic Perspectives*, Vol. 31, No. 2, pp. 87–106.

Nair, Vinod and Geoffrey E Hinton (2010) "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814.

Nesterov, Yuri (1983) "A method for solving the convex programming problem with convergence rate O(1/k$\hat{2}$)," *Dokl. Akad. Nauk SSSR*, Vol. 269, pp. 543–547.

Polson, Nicholas G. and Vadim Sokolov (2017) "Deep Learning: A Bayesian Perspective," *Bayesian Analysis*, Vol. 12, No. 4, pp. 1275–1304, 12.

Rapach, David E., Jack K. Strauss, and Guofu Zhou (2013) "International Stock Return Predictability: What Is the Role of the United States?" *Journal of Finance*, Vol. 68, No. 4, pp. 1633–1662.

Ripley, Brian D (1994) "Neural networks and related methods for classification," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 409–456.

Sargent, Thomas and Christopher Sims (1977) "Business cycle modeling without pretending to have too much a priori economic theory," Working Papers 55, Federal Reserve Bank of Minneapolis.

Schmidhuber, Jurgen (2015) "Deep learning in neural networks: An overview," *Neural Networks*, Vol. 61, pp. 85 – 117.

Sirignano, Justin A., Apaar Sadhwani, and Kay Giesecke (2018) "Deep Learning for Mortgage Risk," economics working paper archive, Stanford Working Paper.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014) "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958.

Stein, Charles (1956) "Inadmissibility of the usual estimator for the mean of a multivariate normal distribution," *Proceedings of the Third Berkeley Symposium in Mathematical Statistics and Probability*, Vol. 1, pp. 197–206.

Stock, James H and Mark W Watson (2002a) "Macroeconomic Forecasting Using Diffusion Indexes," *Journal of Business & Economic Statistics*, Vol. 20, No. 2, pp. 147–162, April.

——— (2002b) "Forecasting Using Principal Components From a Large Number of Predictors," *Journal of the American Statistical Association*, Vol. 97, pp. 1167–1179, December.

Stock, James H. and Mark Watson (2006) "Forecasting with Many Predictors," Vol. 1: Elsevier, 1st edition, Chap. 10, pp. 515–554.

Stone, Mervyn and Rodney J Brooks (1990) "Continuum regression: cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal components regression," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 237–269.

Sung, AH (1998) "Ranking importance of input parameters of neural networks," *Expert Systems with Applications*, Vol. 15, No. 3-4, pp. 405–411.

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton (2013) "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147.

Thornton, Daniel L and Giorgio Valente (2012) "Out-of-sample predictions of bond excess returns and forward rates: An asset allocation perspective," *The Review of Financial Studies*, Vol. 25, No. 10, pp. 3141–3168.

Tibshirani, Robert (1996) "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.

Welch, Ivo and Amit Goyal (2007) "A comprehensive look at the empirical performance of equity premium prediction," *The Review of Financial Studies*, Vol. 21, No. 4, pp. 1455–1508.

Wu, Tong Tong, Kenneth Lange et al. (2008) "Coordinate descent algorithms for lasso penalized regression," *The Annals of Applied Statistics*, Vol. 2, No. 1, pp. 224–244.

Yao, Jingtao, Yili Li, and Chew Lim Tan (2000) "Option price forecasting using neural networks," *Omega*, Vol. 28, No. 4, pp. 455–466.

Zou, Hui and Trevor Hastie (2005a) "Method of Conjugate Gradients for Solving Linear Systems," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 67, No. 2, pp. 301–320.

——— (2005b) "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 67, No. 2, pp. 301–320.

Table 1: **Out-of-Sample Results with Forward Rates: Short Sample**

This table reports the out-of-sample Mean Squared Prediction Error (MSPE) and R^2_{oos} obtained using forward rates to predict excess bond returns for different maturities and across methodologies. The out-of-sample prediction errors are obtained by a recursive forecast which starts using 70% of the full sample length and recursively add one observation at a time in an expanding fashion. The sample period is from 1964:10 to 2008:12, monthly. The first forecast is generated as of 1994:10.

	Mean Squared Prediction Error				$R^2_{oos}(\%)$			
	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$
Panel A: PCA, PLS and Autoencoder								
PCA (5 components)	2.69	9.21	17.25	26.17	-41.5	-31.3	-28.9	-27.2
PCA (3 components)	2.64	9.09	17.11	25.97	-38.4	-29.5	-27.9	-26.2
PCA-Squared (5 Components)	2.53	8.69	16.37	24.87	-33.1	-23.9	-22.4	-20.8
PCA-Squared (3 Components)	2.59	9.02	16.93	25.80	-36.3	-28.5	-26.5	-25.4
PLS (5 components)	2.69	9.21	17.25	26.17	-41.5	-31.3	-28.9	-27.2
PLS (3 components)	2.64	9.09	17.09	25.98	-38.7	-29.5	-27.8	-26.2
Autoencoder	2.56	8.82	16.51	25.05	-34.7	-25.7	-23.4	-21.7
Panel B: Simple and Penalized Linear Regressions								
OLS	2.69	9.21	17.25	26.17	-41.5	-31.3	-28.9	-27.2
Ridge	2.59	8.92	16.80	25.35	-35.8	-27.2	-25.6	-23.2
Lasso	1.94	6.95	13.43	20.46	-1.8	1.0	-0.4	0.6
Elastic Net	1.95	7.01	13.52	20.91	-2.5	0.0	-1.1	-1.6
Panel C: Regression Trees and Neural Networks								
Boosted Regression Tree	2.16	7.37	14.17	21.15	-13.3	-5.0	-5.9	-2.8
Random Forests	2.13	6.99	12.61	20.58	-12.1	0.4	5.7	0.0
NN - 2 layer (5 nodes), lagged inputs $t-1,\ldots,t-11$	1.76	5.99	10.82	15.97	7.6	14.6	19.1	22.4
NN - 2 layer (5 nodes)	1.94	6.45	11.90	17.80	-1.9	8.1	11.0	13.5
NN - 2 layer (3 nodes)	1.82	6.41	11.79	17.53	4.3	8.7	11.8	14.8
NN - 3 layer (5 nodes each)	1.75	5.92	10.72	15.95	7.9	15.6	19.9	22.5
NN - 3 layer (3 nodes each)	1.82	6.10	11.36	16.56	4.6	13.0	15.1	19.6
NN - 4 Layer (3 nodes each)	1.76	5.97	11.00	16.20	7.4	15.0	17.8	21.3
NN - 4 Layer (4,3,2 nodes each)	1.69	5.91	10.83	16.07	11.0	15.7	19.0	21.9

Table 2: **Out-of-Sample Results with Forward Rates: Long Sample**

This table reports the out-of-sample Mean Squared Prediction Error (MSPE) and R^2_{oos} obtained using forward rates to predict excess bond returns for different maturities and across methodologies. The out-of-sample prediction errors are obtained by a recursive forecast which starts using 70% of the full sample length and recursively add one observation at a time in an expanding fashion. The sample period is from 1964:10 to 2016:12, monthly. The first forecast is generated as of 2000:05.

	Mean Squared Prediction Error				$R^2_{oos}(\%)$			
	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$
Panel A: PCA, PLS and Autoencoder								
PCA (5 components)	2.39	8.06	14.93	22.49	-78.7	-62.3	-48.2	-38.8
PCA (3 components)	2.06	6.92	12.94	19.26	-54.0	-39.2	-28.4	-18.8
PCA-Squared (5 Components)	1.97	6.66	12.51	19.18	-47.3	-34.1	-24.1	-18.4
PCA-Squared (3 Components)	1.79	6.05	11.12	16.70	-33.8	-21.8	-10.4	-3.0
PLS (5 components)	2.39	8.06	14.93	22.49	-78.7	-62.3	-48.2	-38.8
PLS (3 components)	2.21	7.54	13.87	20.85	-64.7	-51.9	-37.6	-28.6
Autoencoder	2.01	6.68	12.29	18.37	-50.4	-34.5	-22.0	-13.3
Panel B: Simple and Penalized Linear Regressions								
OLS	2.39	8.06	14.93	22.49	-78.7	-62.3	-48.2	-38.8
Ridge	2.15	7.21	13.47	20.28	-60.4	-45.1	-33.6	-25.1
Lasso	1.34	5.00	10.10	15.97	-0.5	-0.6	-0.2	1.4
Elastic Net	1.36	5.04	10.17	16.35	-1.5	-1.5	-0.9	-0.9
Panel C: Regression Trees and Neural Networks								
Boosted Regression Tree	1.49	5.28	10.59	16.67	-11.4	-6.4	-5.0	-2.8
Random Forests	1.44	4.71	9.12	13.42	-7.5	5.2	9.5	17.2
NN - 2 layer (5 nodes), lagged inputs $t-1,\dots,t-11$	1.43	4.65	8.59	13.03	-6.8	6.3	14.7	19.6
NN - 2 layer (5 nodes)	1.49	4.80	8.99	13.59	-11.3	3.4	10.8	16.1
NN - 2 layer (3 nodes)	1.47	4.82	9.03	13.56	-10.1	2.9	10.4	16.3
NN - 3 layer (5 nodes each)	1.37	4.58	8.31	12.50	-2.2	7.8	17.5	22.9
NN - 3 layer (3 nodes each)	1.48	4.77	8.75	13.03	-10.7	4.0	13.2	19.6
NN - 4 Layer (3 nodes each)	1.46	4.67	8.55	12.84	-8.9	6.0	15.2	20.8
NN - 4 Layer (4,3,2 nodes each)	1.28	4.37	8.21	13.07	4.6	11.9	18.6	19.3

Table 3: **Out-of-Sample Results with Macroeconomic Variables and Forward Rates: Short Sample**

This table reports the out-of-sample Mean Squared Prediction Error (MSPE) and R^2_{oos} obtained using a large panel of macroeconomic variables and forward rates to predict excess bond returns for different maturities and across methodologies. The out-of-sample prediction errors are obtained by a recursive forecast which starts using 70% of the full sample length and recursively add one observation at a time in an expanding fashion. The sample period is from 1964:10 to 2008:12, monthly. The first forecast is generated as of 1994:10. Penalized regressions are estimated including either raw forward rates or a linear combination of forward rates as introduced by Cochrane and Piazzesi (2005) (CP). Similarly, neural networks are estimated either adding the CP factor as additional regressor in the output layer or by estimating a separate network of forward rates and ensemble both macro and forward rates network in the output layer.

	Mean Squared Prediction Error				$R^2_{oos}(\%)$			
	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$
Panel A: PCA, PLS and Autoencoder								
PCA as in Ludvigson and Ng (2009)	1.98	7.47	15.02	22.91	-3.8	-6.5	-12.3	-11.3
PLS (macro + fwd rates)	2.38	8.81	17.23	26.12	-25.0	-25.6	-28.8	-26.9
Autoencoder + CP factor	2.11	7.88	15.50	23.17	-10.6	-12.3	-15.8	-12.6
Panel B: Simple and Penalized Linear Regressions								
OLS	2.38	8.77	17.47	27.16	-25.2	-25.1	-30.6	-31.9
Ridge (macro + CP)	3.18	10.85	21.90	33.84	-67.1	-54.7	-63.7	-64.4
Lasso (macro + CP)	2.11	7.51	14.98	23.35	-11.1	-7.1	-12.0	-13.5
Elastic Net (macro + CP)	1.99	7.03	14.21	21.70	-4.3	-0.3	-6.2	-5.4
Ridge (macro + fwd rates)	3.04	10.95	21.80	34.62	-59.7	-56.0	-62.9	-68.2
Lasso (macro + fwd rates)	1.96	7.23	13.91	20.84	-2.7	-3.0	-4.0	-1.2
Elastic Net (macro + fwd rates)	1.96	7.16	13.28	20.11	-2.8	-2.0	0.7	2.3
Panel C: Regression Trees and Neural Networks								
Boosted Regression Trees	1.63	5.54	11.24	16.35	14.4	21.0	15.9	20.6
Random Forests	1.42	4.85	9.52	14.61	25.7	28.9	28.8	29.0
NN 2 Layer + CP (32 nodes)	1.50	5.38	10.20	15.32	21.4	23.4	23.8	25.6
NN 3 Layer + CP (32, 16 nodes)	1.50	5.41	10.35	15.66	21.3	23.0	22.6	23.9
NN 4 Layer + CP (32, 16, 8 nodes)	1.56	5.65	10.85	16.32	18.1	19.5	18.9	20.7
NN 2 Layer, ensemble macro and fwd	1.41	5.04	9.41	14.32	25.7	28.2	29.7	30.4
NN 3 Layer, ensemble macro and fwd	1.40	4.99	9.39	14.31	26.6	28.9	29.8	30.5
NN 4 Layer, ensemble macro and fwd	1.39	5.02	9.48	14.41	27.1	28.4	29.2	30.0
NN 2 Layer, groups ensembling + CP	1.55	5.69	11.15	16.97	18.8	18.9	16.7	17.6
NN 2 Layer, groups ensembling and fwd	1.37	5.00	9.52	14.67	27.8	28.7	28.8	28.7

Table 4: **Out-of-Sample Results with Macroeconomic Variables and Forward Rates: Long Sample**

This table reports the out-of-sample Mean Squared Prediction Error (MSPE) and R^2_{oos} obtained using a large panel of macroeconomic variables and forward rates to predict excess bond returns for different maturities and across methodologies. The out-of-sample prediction errors are obtained by a recursive forecast which starts using 70% of the full sample length and recursively add one observation at a time in an expanding fashion. The sample period is from 1964:10 to 2008:12, monthly. The first forecast is generated as of 2000:05. Penalized regressions are estimated including either raw forward rates or a linear combination of forward rates as introduced by Cochrane and Piazzesi (2005) (CP). Similarly, neural networks are estimated either adding the CP factor as additional regressor in the output layer or by estimating a separate network of forward rates and ensemble both macro and forward rates network in the output layer.

	Mean Squared Prediction Error				$R^2_{oos}(\%)$			
	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$	$rx^{(2)}_{t+1}$	$rx^{(3)}_{t+1}$	$rx^{(4)}_{t+1}$	$rx^{(5)}_{t+1}$
Panel A: PCA, PLS and Autoencoder								
PCA as in Ludvigson and Ng (2009)	1.46	5.00	9.78	14.79	-9.3	-0.7	2.9	8.7
PLS (macro + fwd rates)	2.17	6.70	12.50	19.14	-62.5	-34.8	-24.0	-18.1
Autoencoder + CP factor	1.51	5.37	11.05	17.02	-12.6	-8.1	-9.6	-5.0
Panel B: Simple and Penalized Linear Regressions								
OLS	3.04	10.19	19.76	29.87	-127.2	-105.2	-96.0	-84.3
Ridge (macro + CP)	2.41	7.28	14.20	21.89	-79.8	-46.6	-40.9	-35.1
Lasso (macro + CP)	1.47	5.23	10.67	16.99	-9.6	-5.2	-5.9	-4.9
Elastic Net (macro + CP)	1.26	4.44	8.85	13.98	6.2	10.7	12.2	13.7
Ridge (macro + fwd rates)	2.49	8.58	16.84	26.67	-85.9	-72.7	-67.1	-64.6
Lasso (macro + fwd rates)	1.39	5.08	10.01	15.78	-3.8	-2.2	0.7	2.6
Elastic Net (macro + fwd rates)	1.39	5.00	10.09	16.40	-3.8	-0.7	-0.1	-1.2
Panel C: Regression Trees and Neural Networks								
Boosted Regression Trees	1.51	4.28	8.36	13.26	-12.8	13.8	17.1	18.2
Random Forests	1.26	3.87	7.24	13.17	5.7	22.1	28.1	18.7
NN 2 Layer + CP (32 nodes)	1.92	5.09	8.21	12.17	-43.6	-2.4	18.6	24.9
NN 3 Layer + CP (32, 16 nodes)	1.54	4.25	7.39	11.06	-14.9	14.4	26.7	31.8
NN 4 Layer + CP (32, 16, 8 nodes)	1.28	3.82	6.93	10.44	-0.4	19.5	27.1	31.2
NN 2 Layer, ensemble macro and fwd	1.88	4.94	8.10	11.96	-40.7	0.6	19.6	26.2
NN 3 Layer, ensemble macro and fwd	1.17	4.17	7.33	10.94	12.3	13.8	27.2	32.5
NN 4 Layer, ensemble macro and fwd	1.04	3.76	6.81	10.28	22.3	22.1	32.5	36.6
NN 2 Layer, groups ensembling + CP	1.21	3.83	7.65	12.16	9.6	22.9	24.1	25.0
NN 2 Layer, groups ensembling and fwd	1.19	3.61	6.98	11.05	11.0	27.3	30.8	31.8

Table 5: **Diagnostics Based on Principal Components Forecasts**

This table reports the in-sample R^2 (%) of a predictive regression where the dependent variable is the year-on-year growth rate of the first three principal components extracted either from the cross-section of forward rates (Panel A and B) or a large set of macroeconomic variables (Panel C). The independent variables are the first three principal components extracted from the term structure of yields (Panel A) and the first three-principal components extracted from macroeconomic variables (Panel B and C). In addition to the PCA we also compute the in-sample R^2 obtained by adding the factors extracted from the same information sets obtained by a set of shallow and deep neural networks.

Panel A: Predicting changes in PCs of the term structure from Yields.

	Short Sample			Long Sample		
	PC # 1	PC # 2	PC # 3	PC # 1	PC # 2	PC # 3
PCA	16.83	28.08	58.78	11.38	26.38	55.79
NN 2 Layer + PCA	22.22	29.21	59.23	17.47	30.30	56.57
NN 3 Layer + PCA	23.06	28.78	59.46	19.39	29.26	56.90
NN 4 Layer + PCA	24.82	28.88	59.41	19.54	29.16	56.97

Panel B: Predicting changes in PCs of the term structure from Macro Variables.

	Short Sample			Long Sample		
	PC # 1	PC # 2	PC # 3	PC # 1	PC # 2	PC # 3
PCA	15.47	39.61	3.26	6.75	35.84	2.92
NN 2 Layer + PCA	52.00	58.98	16.16	22.94	48.40	11.19
NN 3 Layer + PCA	62.03	57.18	14.05	20.07	44.43	7.34
NN 4 Layer + PCA	55.07	53.19	10.86	22.91	42.72	5.58

Panel C: Predicting changes in PCs of macroeconomic variables.

	Short Sample			Long Sample		
	PC # 1	PC # 2	PC # 3	PC # 1	PC # 2	PC # 3
PCA	38.97	37.73	52.28	39.20	39.67	50.06
NN 2 Layer + PCA	42.20	47.16	62.63	42.95	45.74	57.46
NN 3 Layer + PCA	40.59	42.87	61.95	40.54	43.56	55.37
NN 4 Layer + PCA	40.18	42.39	60.86	40.42	43.48	56.98

Table 6: **Diagnostics Based on the Forecasts of Macroeconomic Variables**

This table reports the in-sample Mean Squared Error (MSE) of a predictive regression where the dependent variable is the year-on-year growth rate of a set of macroeconomic time series. The independent variables are the latent factors extracted from a large panel of macroeconomic variables. We follow Stock and Watson (2002a) and consider a simple AR(1) process as a benchmark methodology. The first raw report the MSE for the benchmark AR(1), and for each competing model we report its MSE relative to the MSE from the AR(1). Panel A reports the results for the sample from 1964:01 to 2008:12, whereas Panel B shows the results for the longer sample that goes from 1964:01 to 2016:12.

Panel A: Short Sample

Method	Ind. Prod.	Income	Man. & Sales	Employment	CPI	PCE Def.	Core CPI	PPI
AR	4.13	2.11	3.85	1.47	1.66	1.43	0.98	2.77
NN 2 Layer + AR	0.81	0.85	0.80	0.85	0.78	0.72	0.83	0.86
NN 3 Layer + AR	0.89	0.90	0.86	0.92	0.82	0.76	0.85	0.88
NN 4 Layer + AR	0.93	0.92	0.92	0.95	0.80	0.76	0.86	0.87

Panel B: Long Sample

Method	Ind. Prod.	Income	Man. & Sales	Employment	CPI	PCE Def.	Core CPI	PPI
AR	4.48	2.43	4.13	1.52	1.85	1.35	0.94	3.24
NN 2 Layer + AR	0.85	0.86	0.84	0.85	0.76	0.72	0.83	0.82
NN 3 Layer + AR	0.94	0.94	0.92	0.95	0.79	0.74	0.84	0.85
NN 4 Layer + AR	0.92	0.92	0.91	0.94	0.80	0.76	0.85	0.84

Table A.1: **Explained Variance of PCA vs. Linear Autoencoder**

This table reports the explained variance of the factors extracted from a standard Principal Component Analysis (PCA) and an auto-encoder where the hidden factors have been extracted without using the bond excess returns. To keep comparability we employ a two-layer structure with one hidden layer and linear activation functions.

Panel A: Explained Variance (%), PCA and Autoencoder

Forward Rates			Macro Variables		
# Factors	PCA	Autoencoder	# Factors	PCA	Autoencoder
1	95.00	93.13	1	15.03	15.98
2	98.76	98.02	2	22.22	23.91
3	99.48	99.23	8	47.12	47.29
4	99.81	99.74	32	79.11	78.86
5	100.00	100.00	128	100.00	99.79

Figure 1: **Sample Splitting**

This figure shows the sample splitting used for cross-validation of the hyper-parameters of the penalized regressions, i.e., Lasso, Elastic Net, Ridge, and the neural networks. The forecasting exercise involves and expanding window that starts from 70% of the original data. The blue area represents the sum of the training and validation sample. The former consists of the first 85% of the observations while the latter consists of the final 15% of observations. The training and the validation samples are consequential and not randomly selected in order to preserve the time series dependence. The red area represents the testing sample, which consists of the non-overlapping one-year holding period excess returns of treasury bonds.

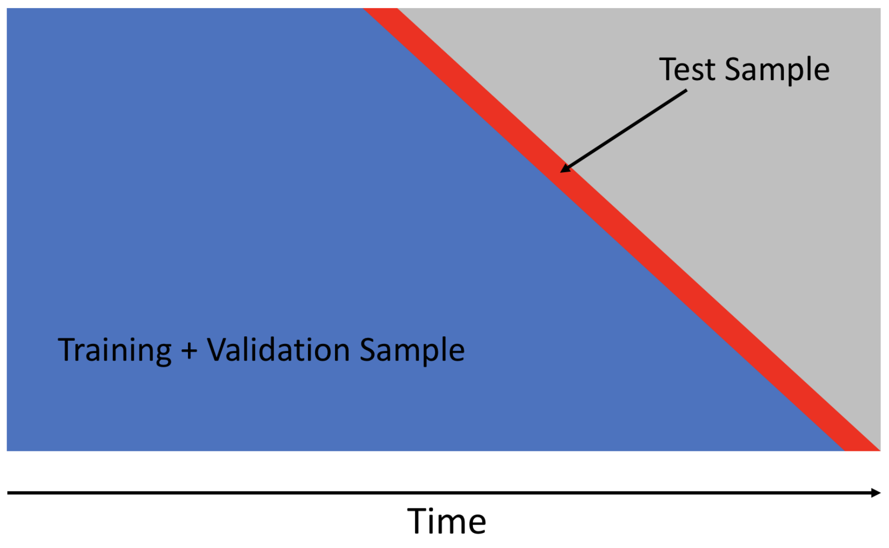
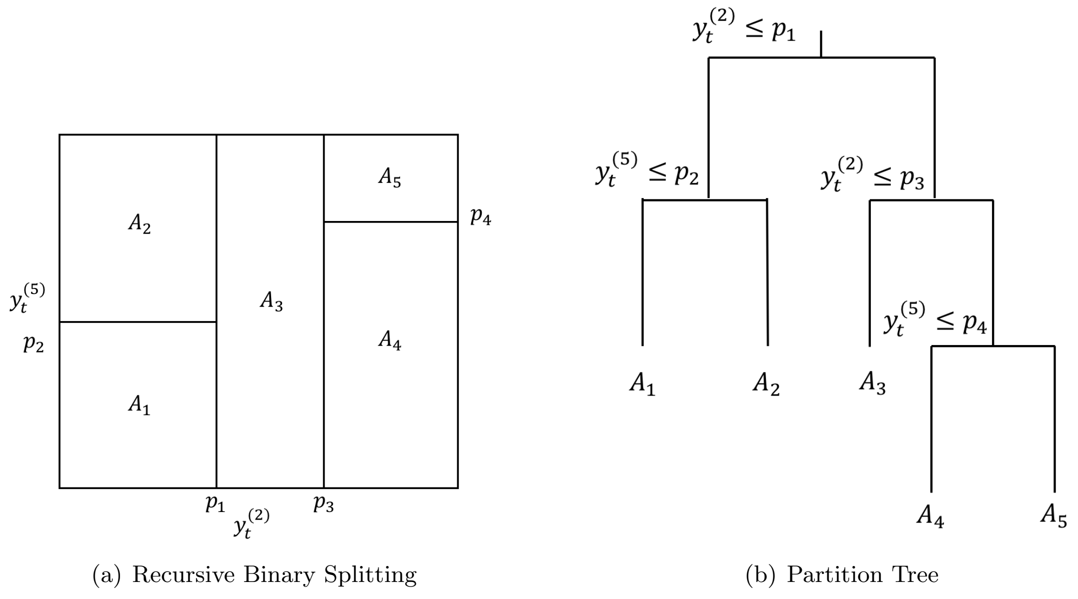


Figure 2: **Example of a Regression Tree**

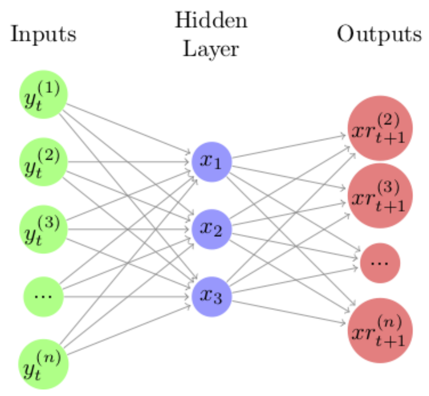
This figure shows an example of a regression trees for a predictive regression with a univariate target variable, e.g., the holding period excess return of a one-year treasury bond, and two predictors, e.g., the two-year and the five-year forward rates, which we label $y_t^{(2)}$ and $y_t^{(5)}$. Left panel shows the partition of the two-dimensional regression space by recursive splitting. Right panel shows the corresponding regression tree.



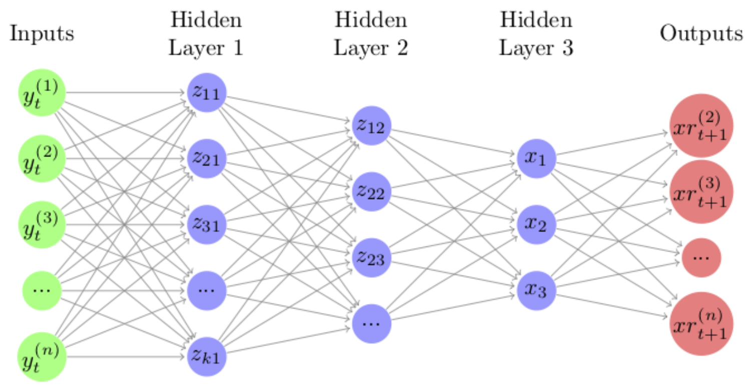
(a) Recursive Binary Splitting (b) Partition Tree

Figure 3: **Examples of Neural Networks**

This figure shows two examples of neural networks. The left panel shows a "shallow" network which consists of an output layer and a single hidden layer. The right panel shows a "deep" neural network which consists of an ouput layer and three hidden layers. The green circles represent the input variables, that is the cross-section of yields, macroeconomic variables of both. The purple circles represent the fully connected hidden nodes. The red circles represent the output variables, that is the bond excess returns across maturities.



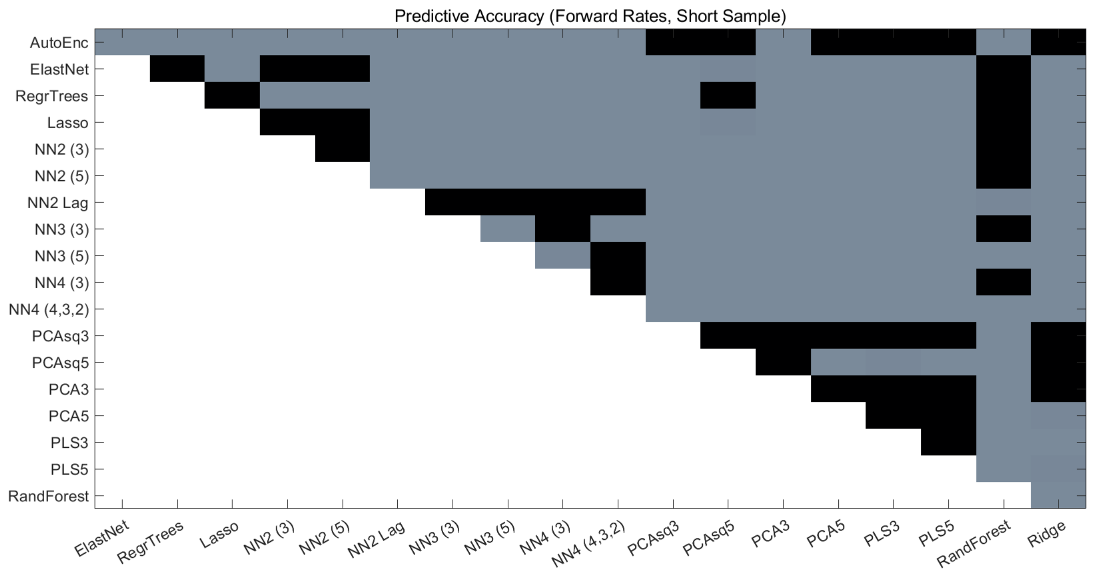
(a) Shallow Neural Network



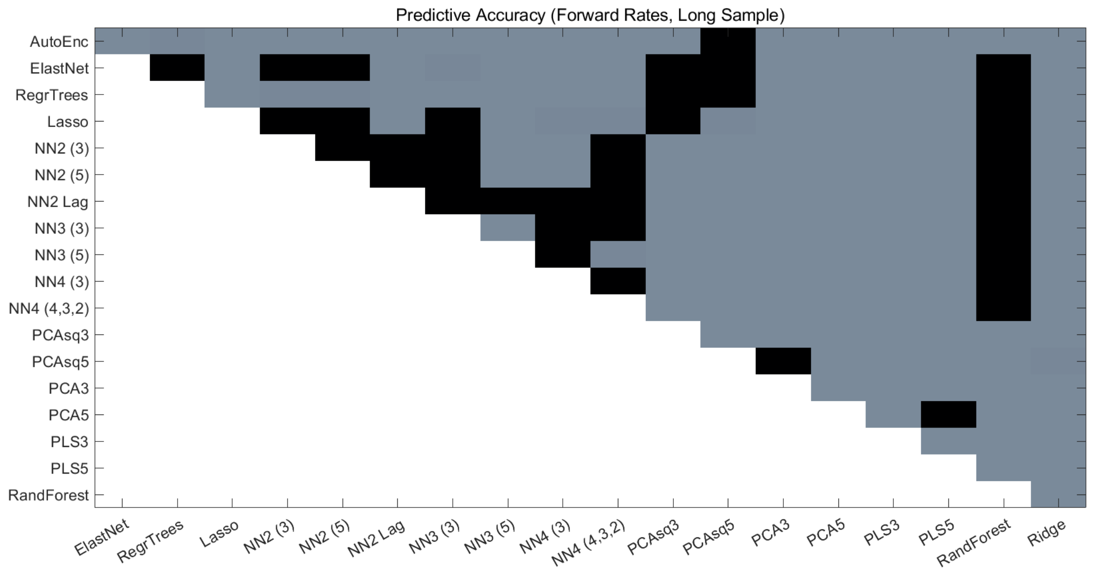
(b) Deep Neural Network

FIGURE 4: **Pairwise Tests of Predictive Accuracy Based on Forward Rates**

This figure shows the results of a pairwise test of predictive accuracy based on test proposed by Diebold and Li (2006) and extended by Harvey et al. (1997). The figure reports the results when forecasting is based on the forward rates. We report in grey the pairs which have a performance difference which is statistically significant at the least at a conventional 5% significance level and in black the pairs which have either a lower significance or no significance at all, i.e., a p-value greater than 0.10. Top panel shows the results for the sample implementation until 2008:12. Bottom panel shows the results from a larger sample that ends in 2016:12 and includes the regime of non-conventional monetary policies implemented in the aftermath of the great financial crisis.



(a) Short Sample



(b) Long Sample

Figure 5: **Examples of a Neural Networks with Additional Predictors**

This figure shows two examples of a three-layer neural network with an exogenous predictor. In particular, this structure simulate the idea of Ludvigson and Ng (2009) in which the latent factors $\boldsymbol{F}_t$ are extracted from a large cross-section of macroeconomic variables and the forward rates are included as a linear combination as proposed by Cochrane and Piazzesi (2005). The green circles represent the input variables, that is the cross-section of macroeconomic variables. The purple circles represent the fully connected hidden nodes. The red circles represent the output variables, that is the bond excess returns across maturities. The yellow circle represents the additional predictor.

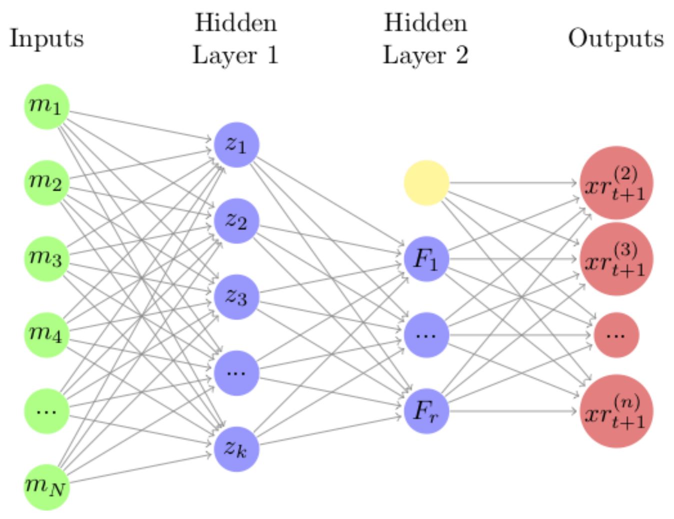
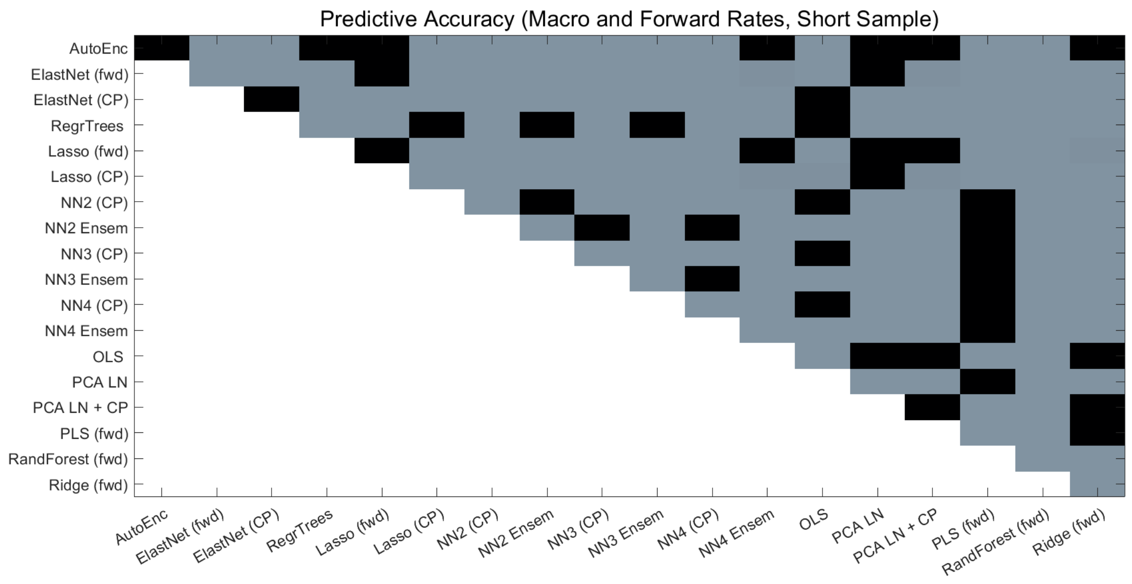
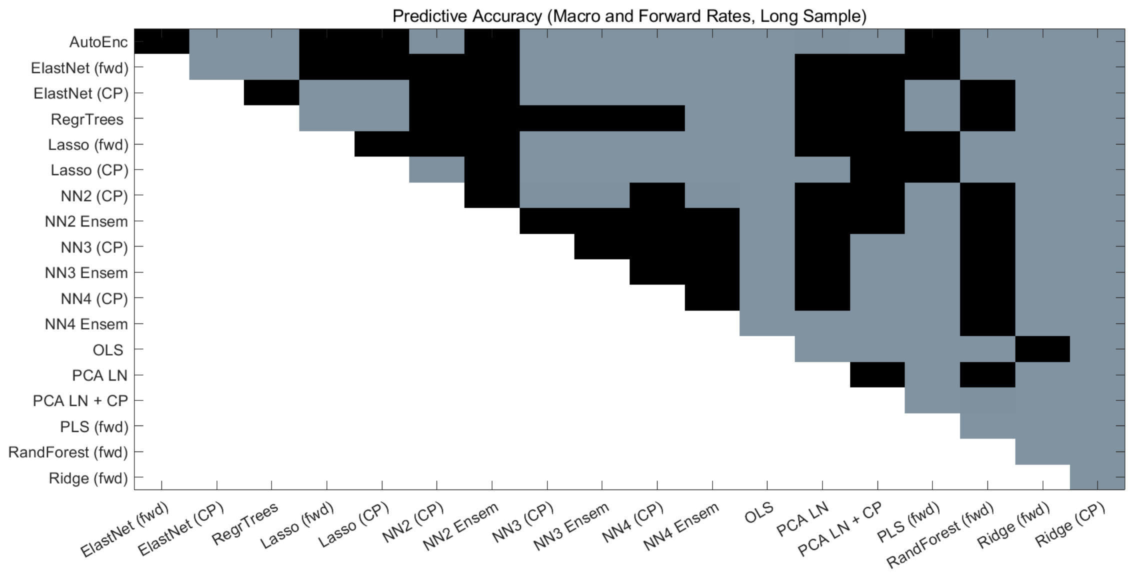


Figure 6: **Pairwise Tests of Predictive Accuracy Based on Macro Variables and Forward Rates**

This figure shows the results of a pairwise test of predictive accuracy based on test proposed by Diebold and Li (2006) and extended by Harvey et al. (1997). The figure reports the results when forecasting is based on both the forward rates and a large panel of macroeconomic information. We report in grey the pairs which have a performance difference which is statistically significant at the least at a conventional 5% significance level and in black the pairs which have either a lower significance or no significance at all, i.e., a p-value greater than 0.10. Top panel shows the results for the sample implementation until 2008:12. Bottom panel shows the results from a larger sample that ends in 2016:12 and includes the regime of non-conventional monetary policies implemented in the aftermath of the great financial crisis.



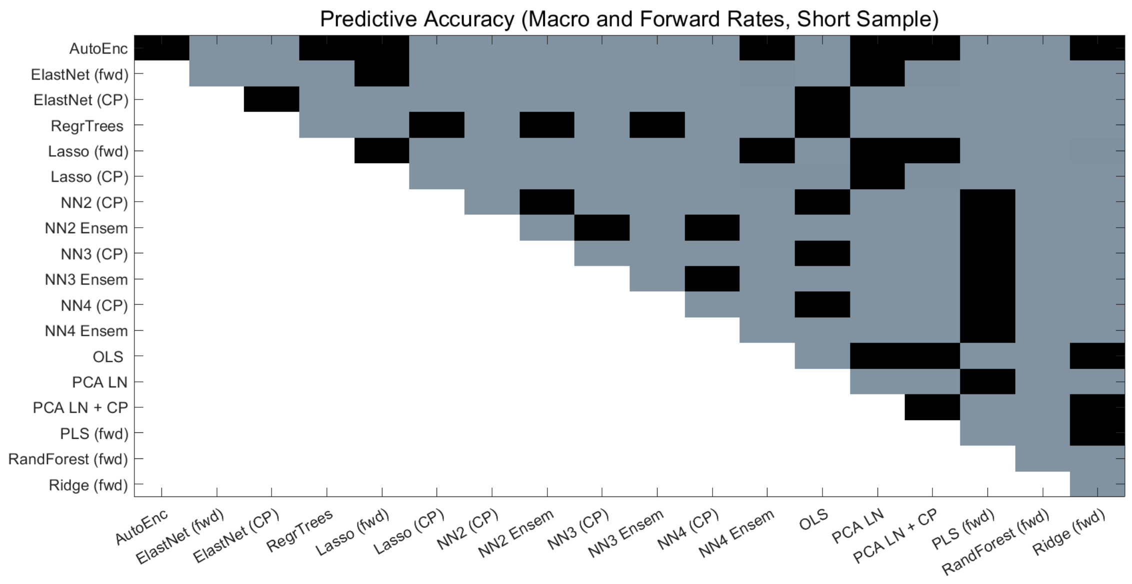
(a) Short Sample



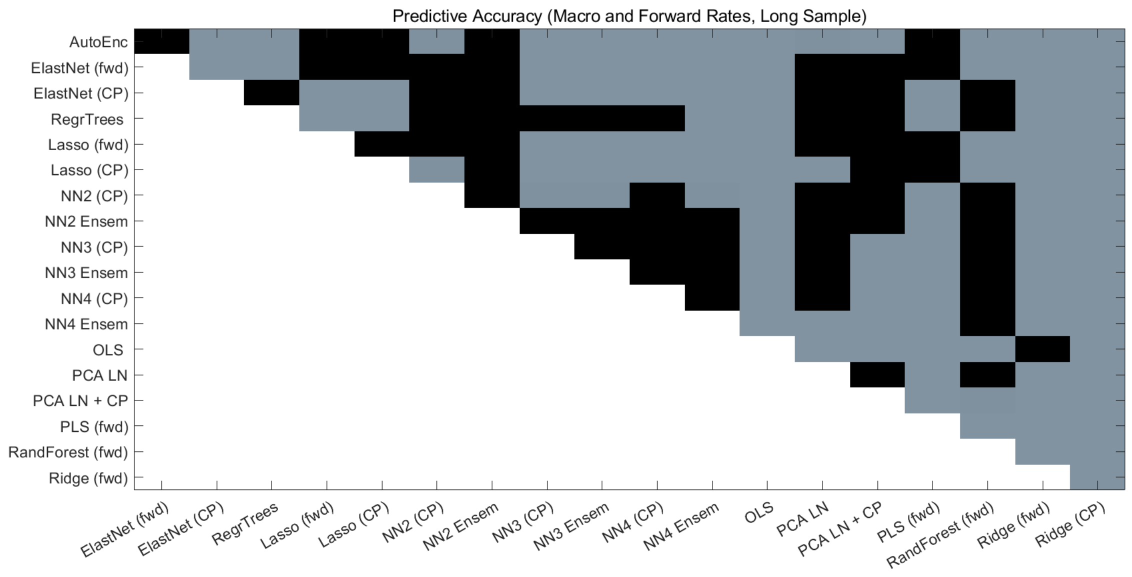
(b) Long Sample

Figure 7: **Pairwise Tests of Predictive Accuracy Based on Macro Variables and Forward Rates**

This figure shows the results of a pairwise test of predictive accuracy based on test proposed by Diebold and Li (2006) and extended by Harvey et al. (1997). The figure reports the results when forecasting is based on both the forward rates and a large panel of macroeconomic information. We report in grey the pairs which have a performance difference which is statistically significant at the least at a conventional 5% significance level and in black the pairs which have either a lower significance or no significance at all, i.e., a p-value greater than 0.10. Top panel shows the results for the sample implementation until 2008:12. Bottom panel shows the results from a larger sample that ends in 2016:12 and includes the regime of non-conventional monetary policies implemented in the aftermath of the great financial crisis.



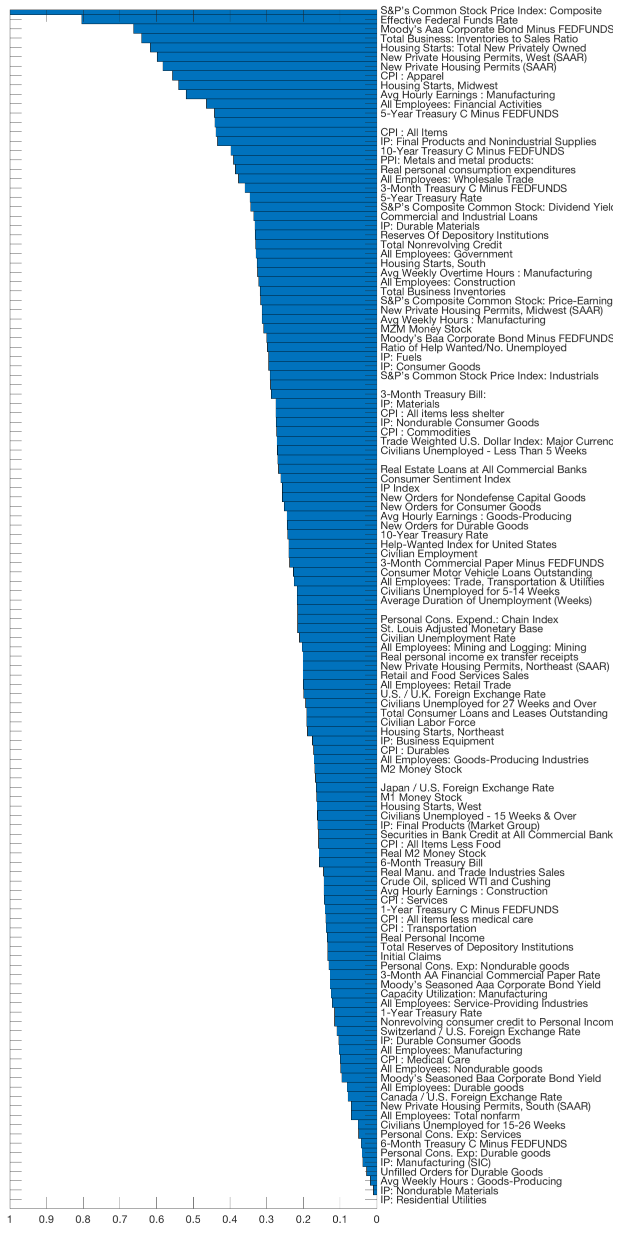
(a) Short Sample



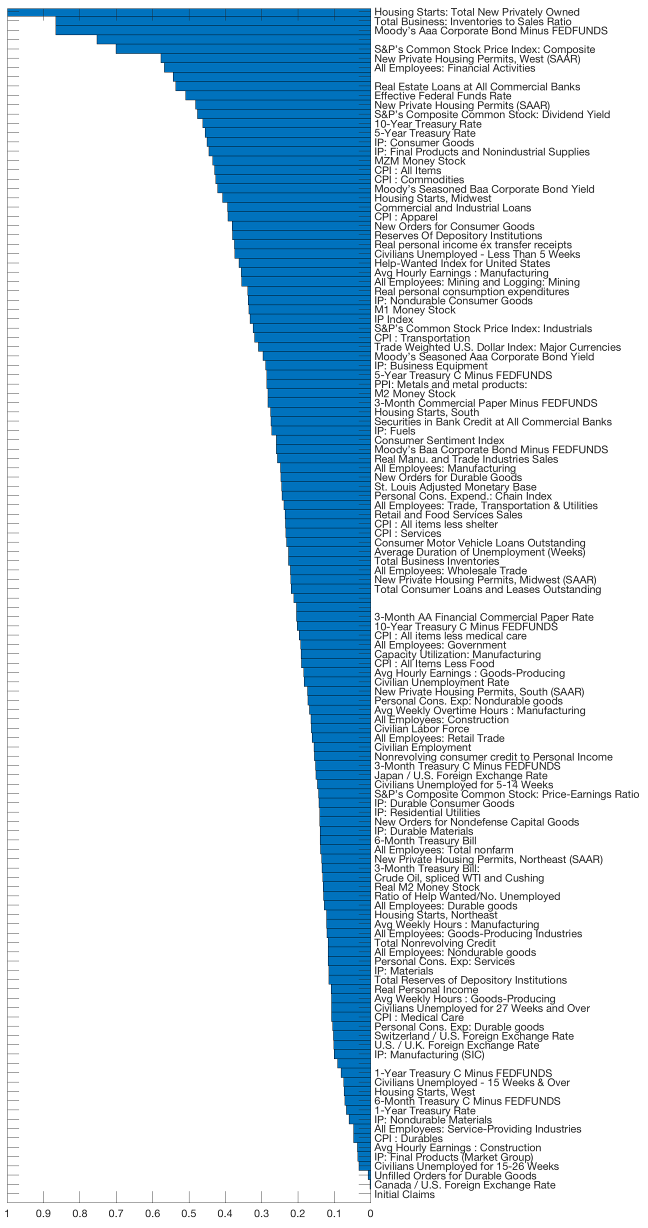
(b) Long Sample

Figure 8: **Relative Importance of Macroeconomic Variables: Average Across Maturities**

This figure shows the relative importance of each input variable obtained from a four-layer neural network. Variables are labelled according to McCracken and Ng (2016). The relative importance of each input variable is calculated based on the gradient evaluated at the in-sample mean of the value of the input. The gradient-at-the-mean is calculated for each time t of the recursive forecasting, then averaged over the out-of-sample period. The results are further averaged across the bond maturities. The left panel shows the results for the short sample period from 1964:01 to 2008:12, whereas the right panel shows the results for the long sample period from 1964:01 to 2016:12.



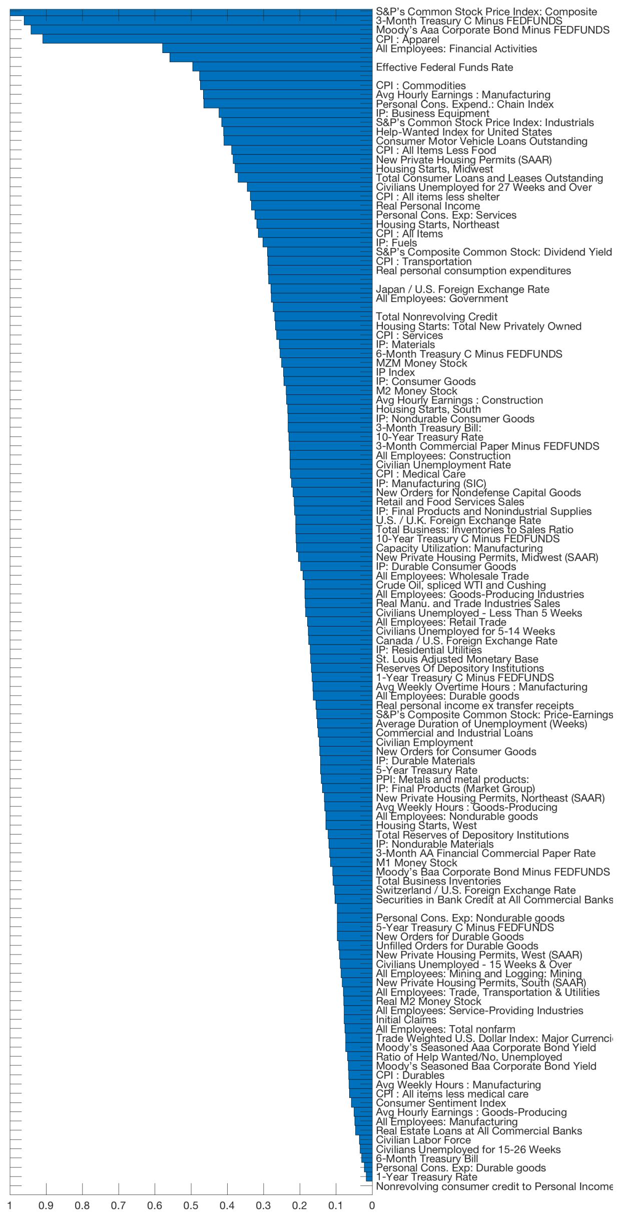
(a) Short Sample



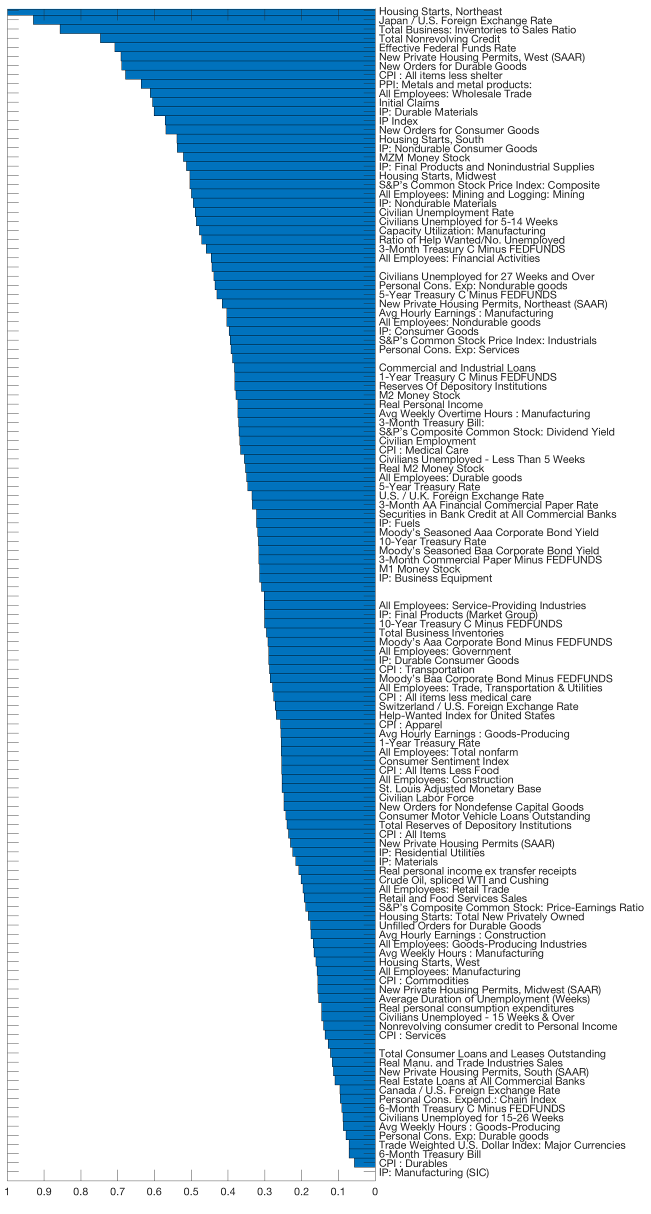
(b) Long Sample

Figure 9: **Relative Importance of Macroeconomic Variables: Each Maturity for the Short Sample**

This figure shows the relative importance of each input variable obtained from a four-layer neural network. Variables are labelled according to McCracken and Ng (2016). The relative importance of each input variable is calculated based on the gradient evaluated at the in-sample mean of the value of the input. The gradient-at-the-mean is calculated for each time t of the recursive forecasting, then averaged over the out-of-sample period. The left panel shows the results for the 2-year bond excess returns, whereas the right panel shows the results for the 5-year bond excess returns. The long sample period goes from 1964:01 to 2008:12.



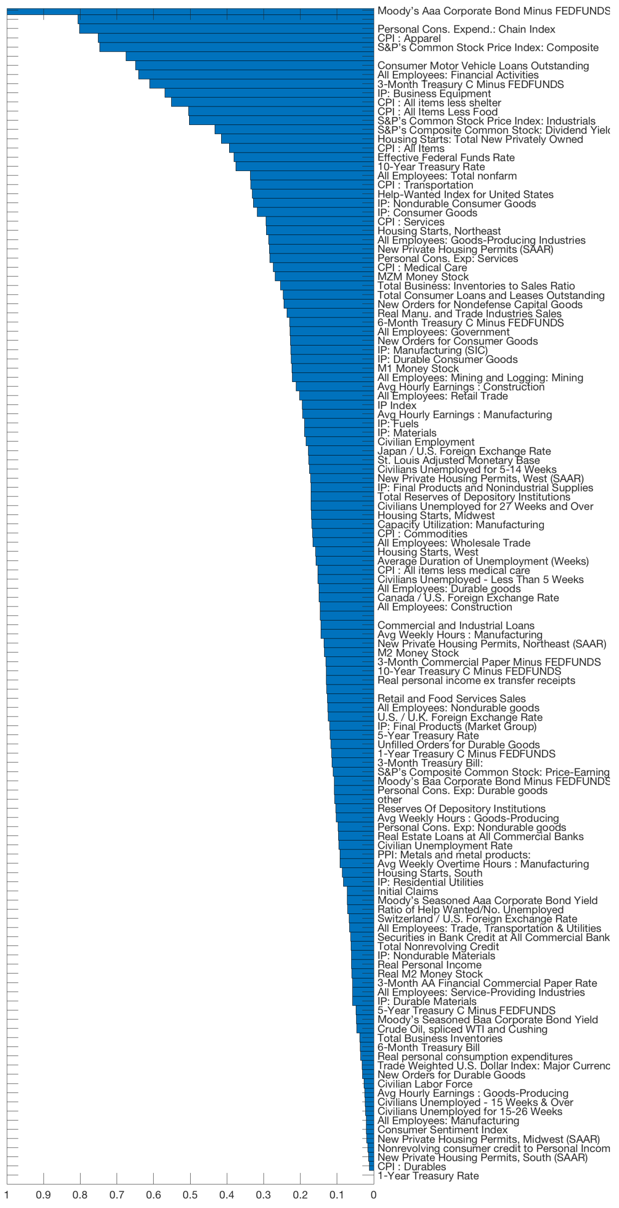
(a) 2-year Maturity



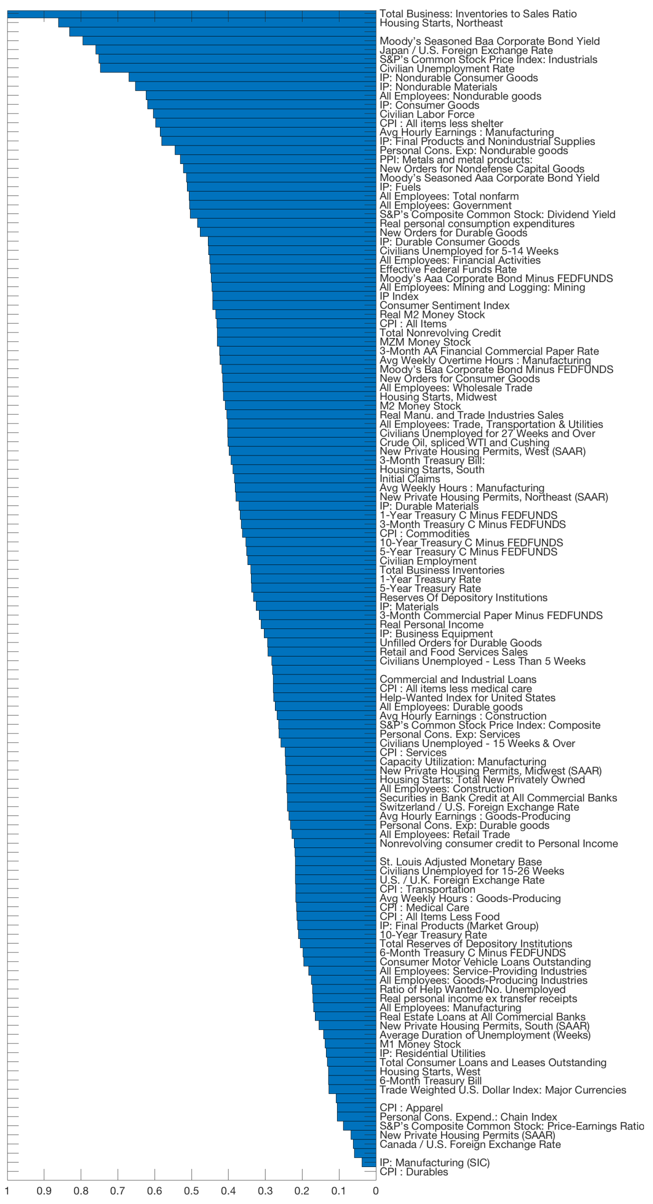
(b) 5-year Maturity

Figure 10: **Relative Importance of Macroeconomic Variables: Each Maturity for the Long Sample**

This figure shows the relative importance of each input variable obtained from a four-layer neural network. Variables are labelled according to McCracken and Ng (2016). The relative importance of each input variable is calculated based on the gradient evaluated at the in-sample mean of the value of the input. The gradient-at-the-mean is calculated for each time t of the recursive forecasting, then averaged over the out-of-sample period. The left panel shows the results for the 2-year bond excess returns, whereas the right panel shows the results for the 5-year bond excess returns. The long sample period goes from 1964:01 to 2016:12.



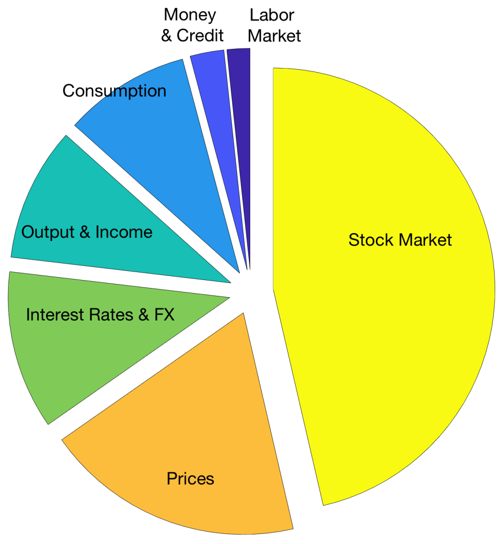
(a) 2-year Maturity



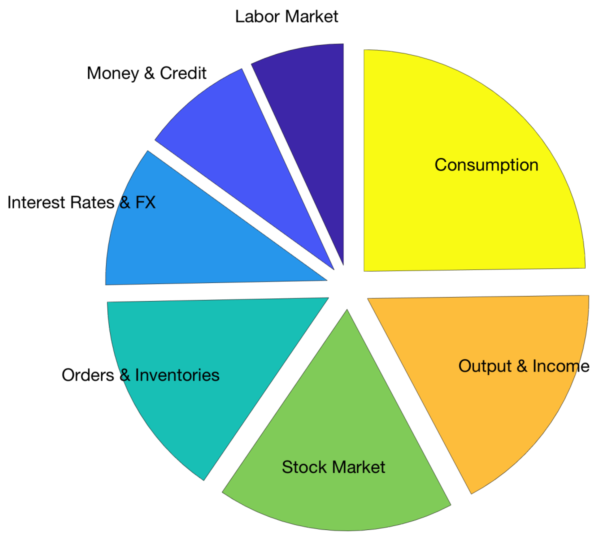
(b) 5-year Maturity

Figure 11: **Relative Importance of Groups of Macroeconomic Variables**

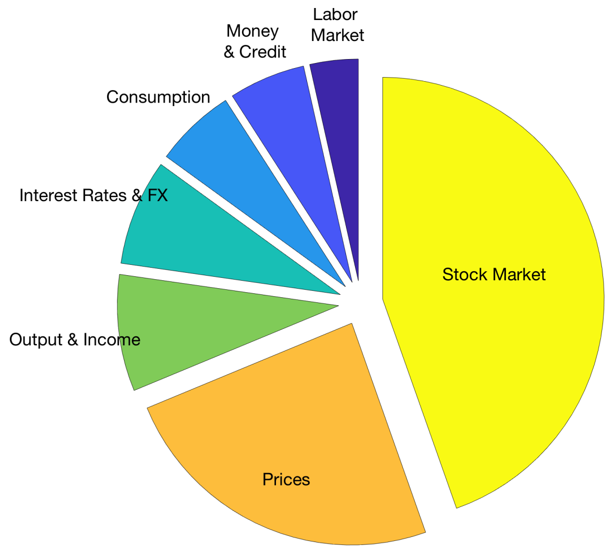
This figure shows the relative importance of each group of macroeconomic variables. Groups are labelled according to McCracken and Ng (2016). The relative importance of each input variable is calculated based on the gradient evaluated at the in-sample mean of the value of the input. The gradient-at-the-mean is calculated for each time t of the recursive forecasting, then averaged over the out-of-sample period. The results for each input variable are averaged within groups. The left (right) column reports the results for the 2-year (5-year) bond excess returns. The top (bottom) row displays the results for the short (long) sample.



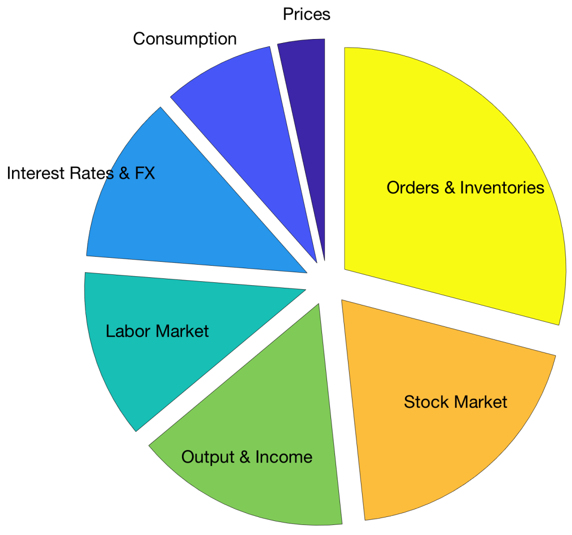
(a) Short Sample, 2-year Maturity



(b) Short Sample, 5-year Maturity



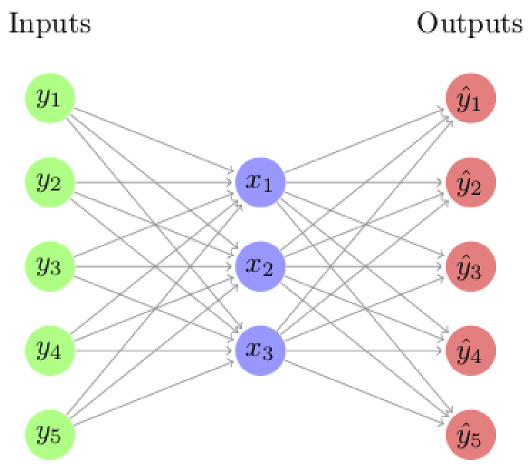
(c) Long Sample, 2-year Maturity



(d) Long Sample, 5-year Maturity

FIGURE 7.1: **Examples of an Autoencoder**

This figure shows two examples of a two-layer autoencoder. The green circles represent the input variables. The purple circles represent the fully connected hidden nodes. The red circles represent the output variables. Unlike a standard neural network, in an autoencoder the input variables and the output variables are the same.



Appendix

A Further Results

A.1 Principal Component Analysis vs. Autoencoder

PCA are closely related to a particular form of artificial neural network which is called *Autoencoder*. An autoencoder is a type of neural network whose outputs are its own inputs. Figure 7.1 shows a visual representation of a two-layer autoencoder. The aim of the autoencoder is to learn a parsimonious representation of the original input data (encoding), typically for the purpose of dimensionality reduction. Similar to PCA, autoencoding is an important compression technique which is designed to replicate the input itself via a bottleneck structure. This means that we select a model which aims to concentrate the information required to recreate the inputs.

To fix ideas, we can think of PCA as projecting the original data onto a lower-dimensional subspace $L < N$. The goal is to find an orthogonal set of L linear basis vectors $\boldsymbol{w}_j \in \mathbb{R}^N$ and the corresponding scores $\boldsymbol{x} \in \mathbb{R}^L$, such that we minimize the average reconstruction error

$$\mathcal{L}(\boldsymbol{W}) = \underset{\boldsymbol{W}}{\arg\min} \|\boldsymbol{y} - \boldsymbol{W}_1 \boldsymbol{x}\|^2 \tag{A.1}$$

subject to the constraint that $\boldsymbol{W}_1$ is orthonormal. The optimal solution is obtained by setting $\hat{\boldsymbol{W}}_1 = \boldsymbol{V}_L$ where $\boldsymbol{V}_L$ contains the L eigenvectors with largest eigenvalues of the empirical variance-covariance matrix of the input data $\boldsymbol{y}$. As a result, $\boldsymbol{x} = \boldsymbol{W}_1^\top \boldsymbol{y}$ represents a low-dimensional encoding of the original data given by an orthogonal projection onto the column space spanned by the eigenvectors.

Now take the autoencoder outlined in Figure 7.1. This structure can be formally represented as a composite function

$$\hat{\boldsymbol{y}} = \boldsymbol{W}_2^\top \boldsymbol{x} \qquad \text{for} \qquad \boldsymbol{x} = h\left(\boldsymbol{W}_1^\top \boldsymbol{y}\right) \tag{A.2}$$

where $h(\cdot)$ is the activation function, $\boldsymbol{y}$ the set of input and $\boldsymbol{x}$ the latent nodes. The weights $\boldsymbol{W} = (\boldsymbol{W}_1, \boldsymbol{W}_2)$ are trained by minimizing some loss function of the form

$$\mathcal{L}(\boldsymbol{W}) = \underset{\boldsymbol{W}}{\arg\min} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2 + \phi(\boldsymbol{W}) \tag{A.3}$$

with $\hat{\boldsymbol{y}}$ defined as in (A.2) and $\phi(\boldsymbol{W})$ some regularization penalty term. In its simplest form without penalization terms, one can fit the model and train the weights $\boldsymbol{W} = (\boldsymbol{W}_1, \boldsymbol{W}_2)$ by minimizing a squared loss function of the form

$$\mathcal{L}(\boldsymbol{W}) = \underset{\boldsymbol{W}}{\arg\min} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2 = \|\boldsymbol{y} - \boldsymbol{W}_2^\top \cdot h\left(\boldsymbol{W}_1^\top \boldsymbol{y}\right)\|^2 \tag{A.4}$$

Now assuming $h(\cdot)$ is linear, then Eq.(A.4) can be written as

$$\mathcal{L}(\boldsymbol{W}) = \|\boldsymbol{y} - \boldsymbol{W}_2^\top \boldsymbol{x}\|^2 \tag{A.5}$$

where $\boldsymbol{x} = \boldsymbol{W}_1 \boldsymbol{y}$. That is, for a linear activation function of the hidden layer the optimal solution of PCA and the autoencoder are equivalent. In particular, if $\boldsymbol{W}_2$ is estimated from the structure of the training data matrix, then we have a traditional factor model, and the $\boldsymbol{W}_1$ matrix provides the factor loadings (see Cook et al., 2007 for further discussion).

Table A.1 shows this case in point. Columns 2 and 3, report the explained variance obtained from the PCA and the linear autoencoder extracted from the cross-section of yields. It is evident how the explanatory power of the factors from the autoencoder tends to mimic closely PCA.

[Insert Table A.1 about here]

A similar results is obtained by investigating the explained variance of the latent factors from the panel of macroeconomic variables (column 5 and 6). Again, PCA and Autoencoding give similar results. However, despite they explain a similar fraction of the variance the factors from PCA and autoencoding are not alike. As a matter of fact, the projections $\hat{\boldsymbol{x}} = \boldsymbol{W}_1\boldsymbol{y}$ are not orthogonal.

A.2 Principal Component Analysis vs. Ridge Regression

Take a standard Singular Value Decomposition (SVD) of the $N \times p$ matrix of predictors $\boldsymbol{X}$, i.e.,

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}'$$

with $\boldsymbol{U}$ and $\boldsymbol{V}$ be an $N \times p$ and a $p \times p$ matrices which span the column and the row span of the original matrix $\boldsymbol{X}$, respectively. $\boldsymbol{D}$ is $p \times p$ diagonal matrix with $d_1 \geq d_2 \geq \ldots \geq d_p \geq 0$ called the singular values of $\boldsymbol{X}$. The fitted value of the OLS regression is given by

$$\boldsymbol{X}\hat{\boldsymbol{\beta}}_{ls} = \boldsymbol{X}\left(\boldsymbol{X}'\boldsymbol{X}\right)^{-1}\boldsymbol{X}'\boldsymbol{y} = \boldsymbol{U}\boldsymbol{U}'\boldsymbol{y} \tag{A.6}$$

Now take the sample covariance matrix of $\boldsymbol{X}$ can be decomposed as

$$\boldsymbol{S} = \boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}'$$

with $\boldsymbol{v}_j$ called the jth principal component direction of $\boldsymbol{X}$. Notice that the principal component $\boldsymbol{z}_j = \boldsymbol{X}\boldsymbol{v}_j = \boldsymbol{u}_j d_j$. Hence $\boldsymbol{u}_j$ represents the projection of the row vectors of $\boldsymbol{X}$ on the direction $\boldsymbol{v}_j$, scaled by d_j. One can show that the fitted value of a principal component regression with k components can be defined as

$$\boldsymbol{X}\hat{\boldsymbol{\beta}}_{pca} = \boldsymbol{U}\mathrm{diag}\left(1,\ldots,1,0,\ldots,0\right)\boldsymbol{U}'\boldsymbol{y} \tag{A.7}$$

After some simplification, the solution of the ridge regression conditional on the penalty parameter λ is given by

$$\begin{aligned}\boldsymbol{X}\hat{\boldsymbol{\beta}}_{ridge} &= \boldsymbol{X}\left(\boldsymbol{X}'\boldsymbol{X} + \lambda\mathbf{1}\right)^{-1}\boldsymbol{X}'\boldsymbol{y} = \boldsymbol{U}\boldsymbol{S}\left(\boldsymbol{D}^2 + \lambda\mathbf{1}\right)^{-1}\boldsymbol{S}\boldsymbol{U}'\boldsymbol{y}\\ &= \boldsymbol{U}\mathrm{diag}\left(\frac{d_1^2}{d_1^2+\lambda},\ldots,\frac{d_p^2}{d_p^2+\lambda}\right)\boldsymbol{U}'\boldsymbol{y}\end{aligned}$$

where $\boldsymbol{u}_j$ are the normalized principal components of $\boldsymbol{X}$. Note that since $\lambda \geq 0$, $d_j^2/\left(d_j^2+\lambda\right) \leq 1$. From here we can see that: (1) a greater amount of shrinkage is applied to directions with smaller d_j^2, (2) in contrast, in PCA regressions, singular values up to k remain intact whereas those from $k+1$ are completely removed. This means that ridge regressions can be interpreted as a dense modeling framework very much like PCA, with the difference that ridge penalize the singular values more smoothly.

B Algorithmic Procedures

B.1 Penalized Regressions

We present the algorithms utilized to estimate the penalized regression models, i.e. the Ridge, Lasso and Elastic Net regressions. To recall, penalized regressions add a penalty term $\phi(\boldsymbol{\beta};\cdot)$ to the least squares loss function $\mathcal{L}(\boldsymbol{\theta})$ where $\boldsymbol{\theta} = (\alpha, \boldsymbol{\beta}^{\top})$. The penalty terms in the individual methods are given by

$$\phi(\boldsymbol{\beta};\cdot) = \begin{cases} \lambda \sum_{j=1}^{p} \beta_j^2 & \text{Ridge regression} \quad \text{(A.8a)} \\ \lambda \sum_{j=1}^{p} |\beta_j| & \text{Lasso} \quad \text{(A.8b)} \\ \lambda\mu \sum_{j=1}^{p} \beta_j^2 + \frac{\lambda(1-\mu)}{2} \sum_{j=1}^{p} |\beta_j| & \text{Elastic net} \quad \text{(A.8c)} \end{cases}$$

Apart from the estimation of $\boldsymbol{\theta}$, we have to determine the level of the shrinkage / regularization parameters λ and μ. Usually, this is achieved by cross-validation, i.e. λ and μ are chosen from a suitably wide range of values by evaluating the pseudo out-of-sample performance of the model on a validation sample and picking the λ, μ that yield the best validation error. In the context of time series forecasts the validation sample should be chosen to respect the time-dependence of the observed data, meaning that the validation sample is chosen to follow upon the training sample used to obtain $\boldsymbol{\theta}$ in time. In the following we discuss in more detail the algorithms that are used to obtain coefficient estimates for the penalized regression models.

In contrast to Ridge, which is discussed further below, Lasso and Elastic net coefficient estimates cannot be obtained analytically because of the L^1 component that enters their respective penalty terms (c.f. Eq. (A.8b) and (A.8c)). Hence, we estimate $\boldsymbol{\theta}$ by means of cyclical coordinate descent proposed by Wu et al. (2008) and extended in Friedman et al. (2010). In our exposition of the algorithm of Friedman et al. (2010) we focus on the elastic net case since the Lasso case is contained as a special case therein (i.e. by setting $\mu = 0$).

At a high-level coordinate descent can be described as an optimization method aimed at minimizing a loss function one parameter at a time while keeping all other parameters fixed. More precisely, consider the loss function for the Elastic Net

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \alpha - \boldsymbol{x}_i^{\top}\boldsymbol{\beta})^2 + \lambda\mu \sum_{j=1}^{p} \beta_j^2 + \frac{\lambda(1-\mu)}{2} \sum_{j=1}^{p} |\beta_j| \tag{A.9}$$

where the factor two in the denominator in front of the sum is introduced to simplify subsequent expressions for the gradient of the loss function. The minimization of the loss function is unaffected by multiplication with a scalar. Denote by $\mathcal{L}(\boldsymbol{\theta})^{(k)}$ the loss function after the k-th optimization step. The gradient of the loss function with respect to β_j evaluated at its current estimate $\hat{\beta}_j^{(k)}$ is given by

$$-\frac{1}{N} \sum_{i=1}^{N} x_{ij}(y_i - \alpha - \boldsymbol{x}_i^{\top}\hat{\beta}) + \lambda(1-\mu)\beta_j + \lambda\mu \tag{A.10}$$

if $\hat{\beta}_j > 0$. A similar expression can be obtained for the case $\hat{\beta}_j < 0$ and $\hat{\beta}_j = 0$ (c.f. Friedman et al., 2007). Then, it can be shown that the optimal $\boldsymbol{\beta}$ is obtained by following the Algorithm (1). Commonly, a "warm-start" approach is used to obtain the parameter estimates over the range for λ and μ during cross-validation, meaning that when moving from one set of regularization parameters λ, μ to the next, the prior estimates $\hat{\boldsymbol{\beta}}$ are utilized as initial parameters for the subsequent coordinate descent optimization.

Algorithm 1: Coordinate Descent

```
Choose initial estimates for $\hat{\alpha} = \bar{y}$ and $\hat{\boldsymbol{\beta}}^{(0)}$ for given $\lambda$ and $\mu$, where $\bar{y}$ is the unconditional
  mean of $y$.
Standardize the inputs $x_{ij}$ such that $\sum_{i=1}^{N} x_{ij} = 0$, $\frac{1}{N}\sum_{i=1}^{N} x_{ij}^2$, for $j = 1, \dots, p$.
**Set** $\epsilon$ to desired convergence threshold
**while** *there is an improvement in the loss function, i.e.* $|\mathcal{L}(\boldsymbol{\theta})^{(k+1)} - \mathcal{L}(\boldsymbol{\theta})^{(k)}| > \epsilon$ **do**
    **for** *all predictors* $j = 1, \dots, p$ **do**
        $\hat{y}_i^{(j)} = \hat{\alpha} + \sum_{l \neq j} x_{il}\hat{\beta}_l$, i.e. the fitted value when omitting the covariate $x_{ij}$
        $\hat{\beta}_j \leftarrow \frac{S\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{y}_i^{(j)}), \lambda\mu\right)}{1+(1-\mu)}$, defines the parameter-wise update, where $S$, the
          soft-thresholding operator, is given by $S(a,b) = \begin{cases} a-b, \text{ if } a>0 \vee b<|a| \\ a+b, \text{ if } a<0 \vee b<|a| \\ 0, b \geq a \end{cases}$
    **end**
**end**
**Output:** Estimates $\hat{\beta}$ for given level of $\lambda, \mu$;
```

In contrast to Lasso and Elastic Net regressions, the Ridge regression has a closed-form solution given by (e.g., see Friedman et al., 2001, Ch. 3)

$$\hat{\beta}^{\text{Ridge}} = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y} \tag{A.11}$$

where $\boldsymbol{X}$ is the input $N \times p$ matrix of p regressors, $\boldsymbol{I}$ is an $N \times N$ identity matrix and $\boldsymbol{y}$ is the vector of dependent variables.

Although, there exists an elegant analytical solution to the Ridge regression setup, it is common to apply a matrix decomposition technique to forego issues incurred by matrix inversion. Thus, we use a singular value decomposition (SVD) of the matrix $\boldsymbol{X}$ with the form

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top \tag{A.12}$$

where $\boldsymbol{U}$ is an $N \times N$ orthogonal matrix, $\boldsymbol{V}$ is an $p \times p$ orthogonal matrix and D is an $N \times p$ diagonal matrix containing the singular values of $\boldsymbol{X}$. Then it can be shown that the fitted values are given as

$$\boldsymbol{X}\hat{\beta}^{\text{Ridge}} = \boldsymbol{U}\boldsymbol{D}\left(\boldsymbol{D}^2 + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{y}. \tag{A.13}$$

The shrinkage parameter λ is chosen by cross-validation. Alternative estimation approaches such as conjugate gradient descent (Zou and Hastie, 2005a) become relevant when $\boldsymbol{X}$ gets larger in dimension.

B.2 Tree-Based Methods

Tree-based methods such as Gradient Boosted Regression Trees or Random Forests are essentially modifications of a universal underlying algorithm utilized for the estimation of regression trees, commonly, that is the Classification and Regression Tree (CART) algorithm (Breiman et al., 1984) presented in Algorithm (2). Next, we present the Algorithm (3) used to populate random forests as suggested by Breiman (2001). Random Forests consist of trees populated following an algorithm like CART, but randomly select a sub-set of predictors from the original data. In this manner, the individual trees in the forest are de-correlated and overall predictive performance relative to a single tree is increased. The hyper-parameters to be determined by cross-validation include first and foremost the number of trees in the forest, the depth of the individual trees and the size of the randomly selected sub-set of predictors. Generally, larger forests tend to produce better forecasts in terms of predictive accuracy. Finally, Algorithm (4) delivers the gradient boosted regression tree (GBRT) (Friedman, 2001). GBRTs are based on the idea of combining the forecasts of several weak learners. The GBRT comprises of trees of shallow depth that produce weak predictions stand-alone, however, tend to deliver powerful forecasts when aggregated adequately.

Algorithm 2: Classification and Regression Trees

```
**Initialize** tree $T(D)$ where $D$ denotes the depth; denote by $R_l(d)$ the covariates in branch $l$ at depth $d$.
for $d = 1, \dots, D$ do
    for $\tilde{R}$ in $\left\{R_l(d), l = 1, \dots, 2^{d-1}\right\}$ do
        Given splitting variable $j$ and split point $s$ define regions
            $R_{\text{left}}(j,s) = \{X \mid X_j \le s, X_j \cap \tilde{R}\}$ and $R_{\text{right}}(j,s) = \{X \mid X_j > s, X_j \cap \tilde{R}\}$
        In the splitting regions set
            $c_{\text{left}}(j,s) \leftarrow \frac{1}{|R_{\text{left}(j,s)}|} \sum_{x_i \in R_{\text{left}(j,s)}} y_i(x_i)$ and $c_{\text{right}}(j,s) \leftarrow \frac{1}{|R_{\text{right}(j,s)}|} \sum_{x_i \in R_{\text{right}(j,s)}} y_i(x_i)$
        Find $j^*, s^*$ that optimize
            $j^*, s^* = \underset{j,s}{\operatorname{argmin}} \Big[ \sum_{x_i \in R_{\text{left}(j,s)}} (y_i - c_{\text{left}}(j,s))^2 + \sum_{x_i \in R_{\text{right}(j,s)}} (y_i - c_{\text{right}}(j,s))^2 \Big]$
        Set the new partitions
            $R_{2l}(d) \leftarrow R_{\text{right}}(j^*, s^*)$ and $R_{2l-1}(d) \leftarrow R_{\text{left}}(j^*, s^*)$
    end
end
**Output:** A fully grown regression tree $T$ of depth $D$. The output is given by
    $f(x_i) = \sum_{k=1}^{2^L} \operatorname{avg}(y_i \mid x_i \in R_k(D)) \mathbf{1}_{\{x \in R_k(D)\}}$,
 i.e. the average response in each region $R_k$ at depth $D$.
```

B.3 Neural Networks

A commonly used algorithm to fit neural networks is stochastic gradient descent (SGD). For this paper we make use of a modified form of gradient decent by adding Nesterov momentum (Nesterov, 1983). In comparison to plain SGD which is often affected by oscillations between local minima, Nesterov momentum (also known as Nesterov accelerated gradient) accelerates SGD in the relevant direction. Algorithm (5) outlines the procedure. It is best practice to initialize neural network

Algorithm 3: Random Forest

```
Determine forest size $F$
for $t = 1, \dots, F$ do
    Obtain bootstrap sample $Z$ from original data.
    Grow full trees following Algorithm (2) with the following adjustments:
        1. Select $\tilde{p}$ variables from the original set of $p$ variables.
        2. Choose the best combination $(j, s)$ (c.f. Algorithm (2)) from $\tilde{p}$ variables
        3. Create the two daughter nodes
    Denote the obtained tree by $T_t$
end
Output: Ensemble of $F$ many trees. The output is the average over the trees in the forest
  given as
      $$f(x_i) = \frac{1}{F} \sum_{t=1}^{F} T_t(x_i)$$
```

Algorithm 4: Gradient Boosted Regression Trees

```
Initialize a gradient bosted regression tree $f_0(x) = 0$ and determine number of learners $F$.
  Let $\mathcal{L}(y, f(x))$ be the loss function associated with tree output $f(x)$.
for $t = 1, \dots, F$ do
    for $i = 1, \dots, N$ do
        Compute negative gradient of loss function evaluted for current state of regressor
          $f = f_{t-1}$
            $$r_{it} = -\frac{\partial \mathcal{L}(y_i, f_{t-1}(x_i))}{\partial f_{t-1}(x_i)}.$$
    end
    Using the just obtained gradients grow a tree of depth $D$ (commonly, $D \ll p$ where $p$ is
      the number of predictors) on the original data replacing the dependent variable with
      $\{r_{it}, \forall i\}$. Denote the resulting predictor as $g_t(x)$.
    Update the learner $f_t$ by
            $$f_t(x) \leftarrow f_{t-1}(x) + \nu g_t(x)$$
      where $\nu \in (0, 1]$ is a hyper-parameter.
end
Output: $f_F(x)$ is the gradient boosted regression tree output.
```

parameters with zero mean and unit variance or variations thereof such as He et al. (2015) like we do in this paper. Over the course of the training process this normalization vanishes and a problem referred to as covariate shift occurs. Thus, we apply batch normalization Ioffe and Szegedy (2015) to the activations after the last ReLU layer.

Batch normalisation reduces the amount of variability of predictors by adjusting and scaling the activations. This allows to increase the stability of the neural network and increase the speed of training. The output of a previous activation is normalized by subtracting the batch mean and dividing by the batch standard deviation. This is particularly advantageous if layers without activation functions, i.e. the output layer, follow layers with non-linear activations, such as ReLU, which tend to change the distributions of the activations. Since the normalization is applied to each individual mini-batch, the procedure is referred to as batch normalization. The SGD optimization remains largely unaffected; in fact, by using batch normalization the structure of the weights is much more parsimonious. Algorithm (6) outlines the procedure from the original paper.

Finally, Algorithm (7) presents the early stopping procedure that is used to abort the training process early when the loss on the validation sample has not improved for a specific number of consecutive iterations. Early stopping is used to improve the performance of the trained models and reduce over-fitting. By evaluating the validation error it prevents the training procedure to simply memorize the training data (see Bishop, 1995 and Goodfellow et al., 2016). More specifically, by means of early stopping the training process is stopped prematurely if the loss on the validation sample has not improved for a number of consecutive epochs. In detail, our algorithm is stopped early if any of the following is true: maximum number of epochs reached the value of 1000, gradient of loss function falls below a specified threshold, or the MSE on validation set has not improved for 20 consecutive epochs. When early stopping occurs we retrieve the model with the best validation performance. Early stopping has two effects. Firstly, early stopping prevents over-fitting by aborting the training when the pseudo out-of-sample performance starts to deteriorate, hence it reduces over-fitting. Secondly, since the optimal number of weight updates is unknown initially, early stopping helps to keep the computational cost at a minimum by potentially stopping the training far before the maximum number of iterations is reached.

Algorithm 5: Stochastic Gradient Decent with Nesterov Momentum

```
Initialize the vector of neural network parameters $\theta_0$ and choose momentum parameter $\gamma$.
 Determine the learning rate $\eta$ and set $v_0 = 0$.
while No convergence of $\theta_t$ do
    $t \leftarrow t + 1$
    $v_t = \gamma v_{t-1} + \eta \nabla_{\theta_t} \mathcal{L}(\theta_t)$
    $\theta_t \leftarrow \theta_{t-1} - v_t$
end
Output: The parameter vector $\theta_t$ of the trained network.
```

Algorithm 6: Batch Normalization per mini-batch

```
Let $\mathcal{B} = x_{1,\ldots,m}$ be a mini-batch of batch-size $m$. Set parameters $\lambda, \beta$.
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$
$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$
$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$
$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$
Output: The normalized mini-batch $mathrmBN_{\gamma,\beta}(x_i)$.
```

Algorithm 7: Early Stopping

```
Initialize the validation error $\epsilon = \inf$ and define a patience $\phi$, also set $k = 0$
while $k < \phi$ do
    Update $\theta$ using Algorithm (5) to get $\theta^{(j)}$, i.e. the parameter vector at iteration $j$
    Compute loss function on validation sample $\epsilon' = \mathcal{L}_{val}(\theta; \cdot)$ if $\epsilon > \epsilon$ then
        $j \leftarrow j + 1$
    end
    else
        $j \leftarrow 0$
        $\epsilon \leftarrow \epsilon'$
        $\theta' = \theta^{(j-p)}$
    end
end
Output: The early-stopping optimized parameter vector $\theta'$
```

C Computational Specs

For our implementation of the various machine learning techniques in Python we utilize the well-known packages `Scikit-Learn`[23] and `Tensorflow`[24] in the `Keras`[25] wrapper. `Scikit-Learn` provides the functionality to estimate regression trees (both gradient boosted regression trees and random forest), partial least squares and penalized regressions (Ridge, Lasso, Elastic Net). Furthermore, we make use of numerous auxiliary functions from `Scikit-Learn` for data pre-processing such as input standardization / scaling and train-test splits. A particularly useful `Scikit-Learn` function is `GridSearchCV`, which allows streamlined systematic investigation of neural network hyper-parameters. Our neural networks are trained using `Keras` and Google's `Tensorflow`. The `Keras` wrapper provides two distinct approaches to construct neural networks, i.e. a sequential API and a functional API. The sequential API is sufficient to construct relatively simple network structures that do not require merged layers, while the functional API is used to build those networks that require merged layers as for example in the case of the exogenous addition of forward rates into the last hidden layer. Keras also implements a wide range of regularization methods applied in this paper, i.e. early-stopping by cross-validation, L1 / L2 penalties, drop-out, and batch normalization.

C.1 Setup

Since the forecasting exercise in this paper is iterative and since we use model averaging, the computational challenge becomes sizeable. For that reason, we perform all computations on a high performance computing cluster consisting of 84 nodes with 28 cores each, totalling to more than 2300 cores. We parallelize our code using the Python `multiprocessing`[26] package. Specifically, we parallelize our code execution at the point of model averaging such that for each forecasting step a large number of models can be estimated in parallel and averaged before moving to the next time step. Although it is common in applications such as image recognition to perform neural network training on GPUs, we refrain from doing so since the speed-up from GPU computing would be eradicated by the increased communication over-head between CPU and GPU as the computational effort of training an individual neural network is relatively small in our exercise.

[23]`http://scikit-learn.org/stable/`, as of 26th October 2018
[24]`https://www.tensorflow.org/`, as of 26th October 2018
[25]`https://keras.io/`, as of 26th October 2018
[26]`https://docs.python.org/3.4/library/multiprocessing.html`, as of 26th October 2018