Online Appendix:
Bond Risk Premia with Machine Learning

Daniele Bianchi* Matthias Büchner† Andrea Tamoni‡

April 14, 2020

*School of Economics and Finance, Queen Mary University of London, Mile End Rd, London, E1 4NS, UK. E-mail: d.bianchi@qmul.ac.uk Web: http://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://mbuechner.com

‡Department of Finance, Rutgers Business School, 1 Washington Park, Newark, NJ 07102. E-mail: andrea.tamoni.research@gmail.com Web: https://andreatamoni.meltinbit.com

A A Simple Motivating Framework

Relying on the large literature on time varying risk premia (see e.g. Campbell and Cochrane, 1999, Wachter, 2006 and Buraschi and Jiltsov, 2007) we present a simple model with external habit formation which leads to the Quadratic Linear model.[1]

The representative agent maximizes

$$E\left[\int_0^\infty u(C_t, X_t, t)dt\right],$$

where the instantaneous utility function is given by

$$u(C_t, X_t, t) = \begin{cases} e^{-\rho t}\frac{(C_t - X_t)^{1-\gamma}}{1-\gamma} & \text{if } \gamma > 1 \\ e^{-\rho t}\log(C_t - X_t) & \text{if } \gamma = 1 \end{cases}$$

where X_t is an external habit level as in Campbell and Cochrane (1999). Consider now the Surplus Consumption Ratio

$$S_t = \frac{C_t - X_t}{C_t}.$$

We model external habit formation as in Pastor and Veronesi (2005), i.e. we assume:

$$\begin{aligned} S_t &= e^{s_t} \\ s_t &= a_0 + a_1 z_t + a_2 z_t^2 \\ dz_t &= k_z(\bar{z} - z_t)dt + \sigma_z dW_{c,t}. \end{aligned}$$

Pastor and Veronesi (2005) show that by choosing a_i appropriately (in particular, $a_2 < 0$), then $s_t < 0 \to S_t \in [0, 1]$. In addition, we must have $\frac{\partial s(y)}{\partial y} = a_1 + 2a_2 y_t > 0$ so that positive shocks to consumption $dW_{c,t}$ translate into positive shocks to the surplus consumption ratio S_t. The

[1] This material is based on the 2015 Version of Pietro Veronesi's lecture notes on "Topics in Dynamic Asset Pricing".

rest of the model is defined by:

$$dc_t = g_t dt + \sigma_c dW_{c,t}$$

$$dq_t = i_t d_t + \sigma_q dW_{q,t}$$

where we let $c_t = \log C_t$ and $q_t = \log Q_t$ be log consumption and log inflation. Finally assume that $\boldsymbol{X}_t = (g_t, i_t, z_t)'$ follows the process

$$d\boldsymbol{X}_t = K(\Theta - \boldsymbol{X}_t)dt + \Sigma d\boldsymbol{W}_t \ , \tag{A.1}$$

where $d\boldsymbol{W}_t$ is a vector of Brownian motions.

In this economy, the SDF is given by $M_t = e^{\eta t - \gamma\left(c_t + a_0 + a_1 z_t + a_2 y_t^2\right) - q_t}$ and the interest rate has a linear quadratic structure

$$r_t = \delta_0 + \gamma g_t + i_t + \delta_z z_t + \delta_{zz} z_t^2 \tag{A.2}$$

Finally, denote the zero coupon bond price by $P(\boldsymbol{X}_t, t; T)$. Given the specification of the model (A.1), the price of the zero coupon bond $P(\boldsymbol{X}_t, t; T)$ is the solution to the Partial Differential Equation (PDE)

$$rZ = \frac{\partial P}{\partial t} + \frac{\partial P}{\partial \boldsymbol{X}} K(\Theta - \boldsymbol{X}_t) + \frac{1}{2} tr\left(\frac{\partial^2 P}{\partial \boldsymbol{X} \partial \boldsymbol{X}'} \Sigma\Sigma'\right)$$

subject to the final condition $P(\boldsymbol{X}_T, T; T) = 1$. Using the method of undetermined coefficients and exploiting the risk free rate equation (A.2), we can verify that the log bond price is given by

$$\log P(\boldsymbol{X}_t, t; T) = A(t; T) + \mathbf{B}(t, T)'\mathbf{X}_t + \mathbf{X}_t'\mathbf{C}(t; T)\mathbf{X}_t$$

where $A(t; T), \mathbf{B}(t, T)$ and $\mathbf{C}(t; T)$ satisfy a set of ODEs - see Ahn et al. (2002) and Leippold and Wu (2003).[2] Hence, the bond pricing formula is also linear-quadratic with factors given by consumption growth g_t, expected inflation i_t and habit z_t.

For a fairly general framework with non-linear dynamics under the historical measure, and

[2]More precisely, we conjecture $P(\boldsymbol{X}_t, t; T) = e^{A(t;T) + \mathbf{B}(t,T)'\mathbf{X}_t + \mathbf{X}_t'\mathbf{C}(t;T)\mathbf{X}_t}$, we compute derivatives $\frac{\partial P}{\partial t}$, $\frac{\partial P}{\partial \boldsymbol{X}}$, and $\frac{\partial^2 P}{\partial \boldsymbol{X} \partial \boldsymbol{X}'}$, we substitute r and partial derivatives in the PDE, and we collect terms.

encompassing many equilibrium models with recursive preferences and habit formation see Le et al. (2010). Finally, note that besides habit-based term structure models, non-linearities are also featured in state-dependent, learning-based models (see, e.g. Veronesi, 2004).

B Pairwise Test of Predictive Accuracy

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 reject 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).

Figure B.1 confirms the statistical significance of the conclusions reached in Sections 3.1 and 3.2 of the manuscript by implementing such test. The figure reports the significance of the performance gaps while the direction can be inferred by looking at Tables 1-2 of the manuscript. We color-coded the statistical significance of the test, with lighter color suggesting a stronger rejection of the null (H_0: the pairs have the same performance).

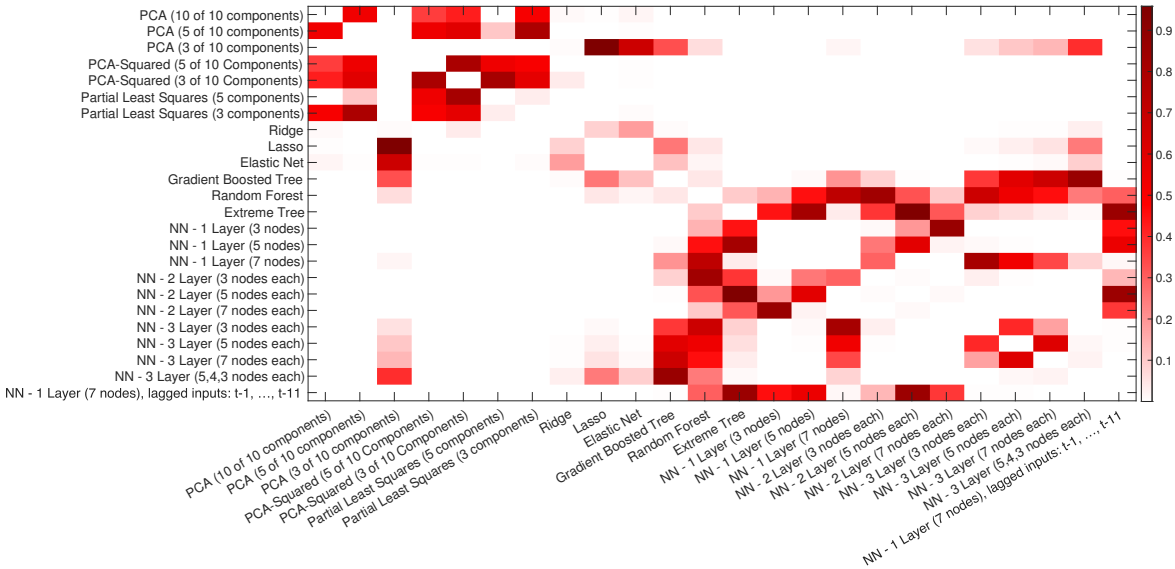
The top panel reports the analysis when yields are the only predictors. We observe that: (1) the performances across classes of machine learning methods tend to differ markedly, with penalized regressions that have significantly worse accuracy relative to trees, and NNs (white cells); (2) shallow learners and NNs with 2-layers often tend to have similar performance (darker red); however, increasing the depth of the NN further worsen the performance (lighter red); (3) lagged forward rates do not improve the predictive accuracy of a shallow learner (red cell).

The bottom panel reports the pairwise DM tests when the macro variables from the FRED-MD database are added to forward rates. The test confirms that we observe that the depth of the network matters, as testified by the statistically significant out-performance (lighter cells) of a three-layer hybrid network (NN 3 Layer, fwd rates direct) over a shallow and two-layer hybrid network. At the same time the figure confirms that a shallow network with group ensembling

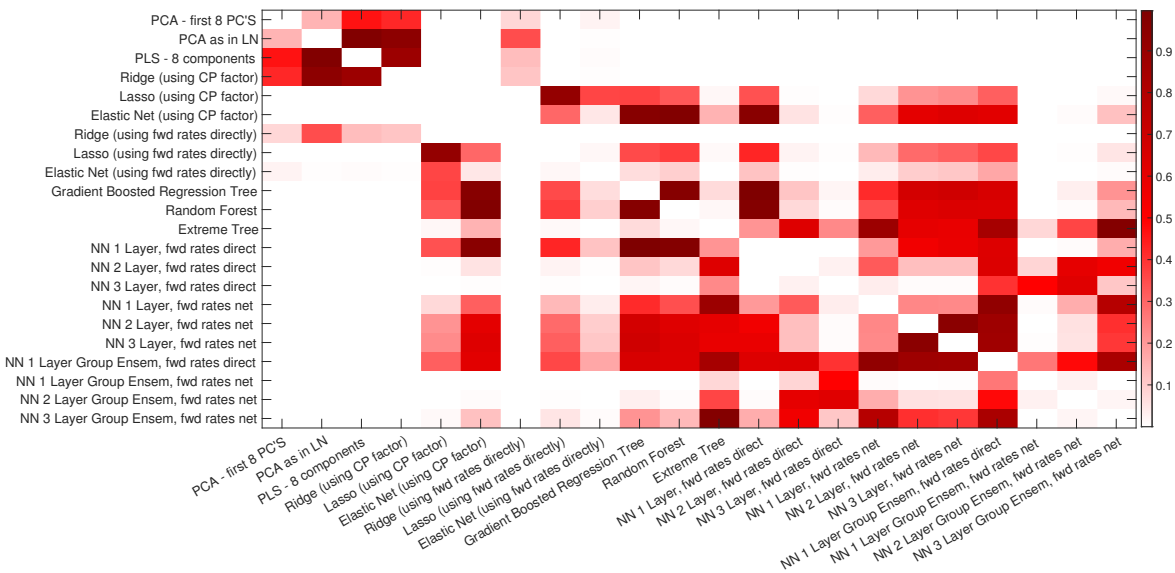
performs on par with a deeper, three-layer network that does not distinguish between economic categories.

FIGURE B.1: **Pairwise Tests of Predictive Accuracy**

This figure shows the results of a pairwise test of predictive accuracy based on Diebold and Mariano (1995) and extended by Harvey et al. (1997). The figure reports the results when forecasts are based only on the forward rates (top panel) and when both macroeconomic variables and forward rates are used (bottom panel). The red color scale indicates the statistical significance of the test, that is the lighter the color the lower the p-value for the null hypothesis that the forecasts between a pair are not statistically different.



(a) Forecast with forward rates



(b) Forecast with macro + fwd rates

C Dissecting predictability: Further discussion and additional results

C.1 Bond return predictability and cumulative SSE

To identify the periods in which the models perform best, we follow Welch and Goyal (2008) and compute the difference in the cumulative sum of squared errors (SSE) for the EH model versus the machine learning model of interest, $\Delta CumSSE$. Positive and increasing values of $\Delta CumSSE$ suggest that the model under consideration generates more accurate point forecasts than the EH benchmark.

Figure C.1 plot the $\Delta CumSSE$ for the best performing regression tree specification, i.e., extreme tree (Panels (a) and (c)), and for the best performing neural network, namely the *NN 1 Layer (3 nodes)* – when forecasting with only the forward rates (Panel (b)) – and *NN 1 Layer Group Ensem + fwd rate net* – when including also macroeconomic variables (Panel (d)) – (see Tables 1-2 of the manuscript for reference). We focus on the ten-year bond maturity.

[Insert Figure C.1 about here]

In general the plots show that the various machine learning models perform well relative to the EH model as manifested by lines that are increasing for most periods. Indeed, only when we consider exclusively yields-based predictors, we then observe occurrence of decreasing graphs (i.e. periods where the models underperform against this benchmark). However, these are few isolated occasions (e.g., around 2001 and 2008). Interestingly, adding macroeconomic variables improves the predictive accuracy of the models relative to the benchmark. Comparing panel (c) to (d), we note that: (1) the performance of extreme trees is particularly effective in few, rather prolonged, periods, namely 1992 to 1996 and the aftermath of the financial crisis; (2) the performance of the NN with group ensembling is instead characterized by an almost steady improvement in performance.

Another interesting aspect to investigate is whether expected bond returns could have

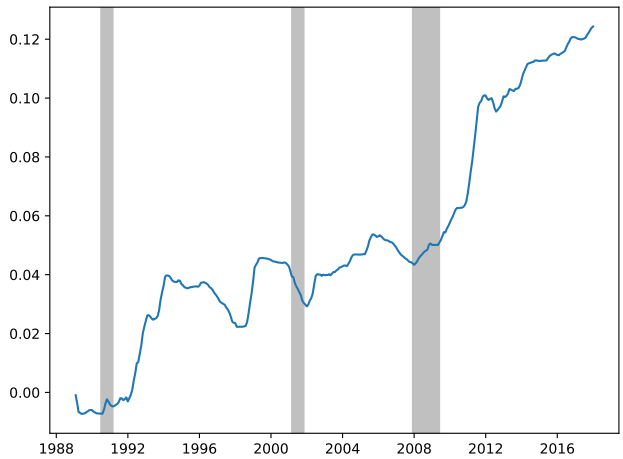
evolved differently in response to availability of technology.[3] However, referring to our plots of out-of-sample R^2 over time, even in later years when adoption of neural networks spread, we still notice the outperformance of neural networks vis-a-vis the expectation hypothesis.

Overall, we take these patterns as rather reassuring that the value-added through machine learning based forecasts is rather pervasive and not concentrated in isolated events.

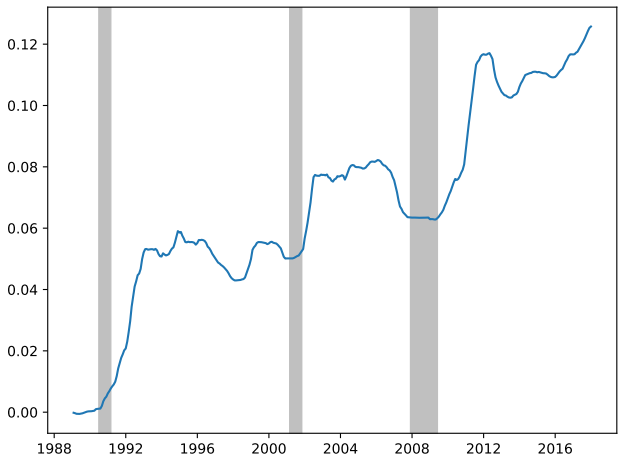
[3]The theory necessary to apply our networks was available in the 1990 when our out-of-sample period starts. The overarching concept of back-propagation in multi-layer neural networks was introduced in Werbos (1974). The use of automatic differentiation (AD) that is necessary for fast computation of gradients during back-propagation was used in Werbos (1982). An early example of a study using neural networks for prediction in a finance context, on gas markets to be precise, with similar models as used by us is Werbos (1988) and others exist before 1990. What is less clear is whether the necessary computational power was available to estimate the models in a reasonable amount of time. Also, while the most fundamental building blocks were in place in the late 1980s, regularization concepts such as dropout (Srivastava et al., 2014) and batch normalization (Ioffe and Szegedy, 2015) were only introduced later.

FIGURE C.1: **Squared Forecast Errors Across Time**

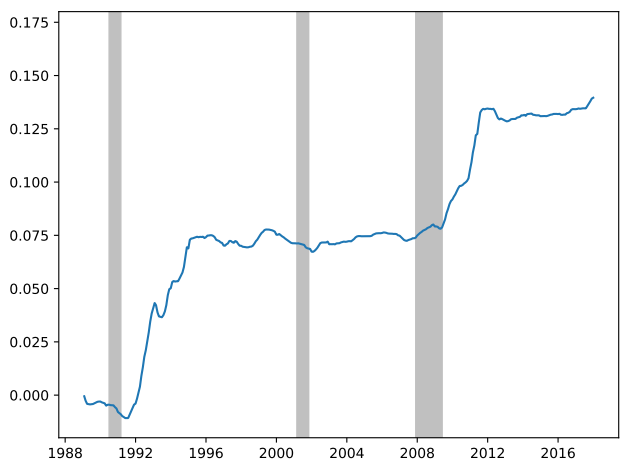
This figure plots the time series of difference in squared forecast errors from a given model versus the expectations hypothesis. We scale the forecast errors by the variance of the dependent variable times $T-1$, i.e. each month t we plot the value attained by $\frac{(\hat{\epsilon}_{t+1}^{EH})^2-(\hat{\epsilon}_{t+1}^{Model})}{(T-1)\text{Var}(rx_{t+1})}$. The out-of-sample period starts in 1990. The expectation hypothesis uses all data starting from the in-sample period in 1971:08. The figures present results for the 10-year maturity and focus on the best performing regression tree and neural network when forecasts are generated either based on forward rates only or by macroeconomic variables + forward rates.



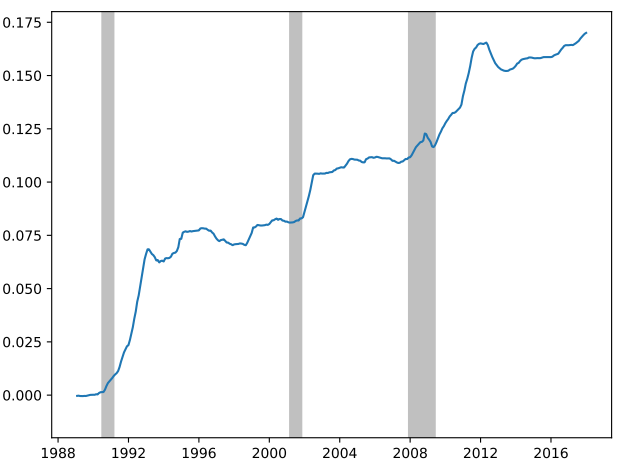
(a) Extreme tree, forward rates, 10-year maturity



(b) Neural net, forward rates, 10-year maturity



(c) Extreme tree, macro + fwd rates, 10-year maturity



(d) Neural net, macro + fwd rates, 10-year maturity

C.2 Bond return predictability and shape of the yield curve

TABLE C.1: **Conditional Forecast Accuracy: Double Sorts on Prevailing Yield Curve Level & Slope**

This table reports the forecast accuracy when conditioning on the prevailing shape of the yield curve, i.e. its level and slope. As a measure of of yield curve level we use the 2-year yield, while the measure of yield curve slope is the difference between the 10-year and 2-year yield. We sort all observations in our out-of-sample period based on the median level and slope of the yield curve prevailing at the start of the holding period. The double sort is performed unconditionally. We denote observations below the median by "Low" and observations above the median by "High". For example, "Level Low - Slope High" refers to all observations for which the prevailing yield curve level was below the median, while the yield curve slope was above the median. The median yield curve level over our out-of-sample period is 3.84% and the median yield curve slope is 1.33%. Forecast accuracy is proxied by the R^2 in regressions of realized returns, $xr_{t+1}^{(i)}$, on the predicted bond risk premium, $\hat{xr}_{t+1}^{(i)}$: $xr_{t+1}^{(i)} = \alpha + \beta \hat{xr}_{t+1}^{(i)} + \epsilon_{t+1}^{(i)}$, where the regressions are performed using all the observations that fall into the four distinguished yield curve shape cases. Predicted bond risk premia stem from forecasting either with only the forward rates (Panel A) – using *NN 1 Layer (3 nodes)* – or with forward rates plus macroeconomic variables (Panel B) – using *NN 1 Layer Group Ensem + fwd rate net* – (see Table 1-2 in the manuscript for reference). The out-of-sample predictions are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12. For each case, we report R^2, the mean fitted value, the p-values for the hypothesis tests $H_0 : \alpha = 0$ and $H_0 : \beta = 1$, and the fraction of the sample period falling into the respective case.

Panel A: Forecasting with Forward Rates

	R^2 (%)	Mean Fitted Value	p-val ($\alpha = 0$)	p-val ($\beta = 1$)	Sample Fraction
All	21.97	4.33%	0.69	0.10	100.0%
Level Low - Slope Low	7.79	-2.27%	0.00	0.91	9.8%
Level Low - Slope High	16.18	6.26%	0.76	0.70	40.2%
Level High - Slope Low	23.79	3.60%	0.00	0.15	40.2%
Level High - Slope High	58.84	5.96%	0.09	0.00	9.8%

Panel B: Forecasting with Forward Rates + Macro

	R^2 (%)	Mean Fitted Value	p-val ($\alpha = 0$)	p-val ($\beta = 1$)	Sample Fraction
All	27.95	4.33%	0.91	0.21	100.0%
Level Low - Slope Low	14.06	-2.27%	0.00	0.99	9.8%
Level Low - Slope High	18.48	6.26%	0.83	0.42	40.2%
Level High - Slope Low	28.16	3.60%	0.01	0.12	40.2%
Level High - Slope High	65.75	5.96%	0.06	0.01	9.8%

C.3 Relative importance of macroeconomic variables: Further discussion

To study the relative importance of macroeconomic variables we rely 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.

In principle, the partial derivative of the target variable with respect to the sample average of each input, in Equation 7 in the manuscript, is a deterministic object. As such, one may question the robustness and reliability of our estimates given the stochastic nature of the training process of neural networks (see, e.g., Hansen and Salamon, 1990 and Dietterich, 2000). The training process of NNs is stochastic in so far that the optimization path will generally depend on its initialization. Therefore, different initializations of the network weights can lead to different outcomes. To address this issue, we estimate the relative importance from different starting values of the neural network weights, and average the results. To be precise, we initialize network weights using the He (2015a) normal initialization procedure and use 100 different, but fixed seeds. Then we construct predictions from all network estimates and calculate the corresponding partial derivatives as in Equation 7 in the manuscript. The final result is obtained by averaging across the estimated models.

Alternative methodologies for measuring relative variable importance in NNs 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 is computationally too expensive to execute. 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 5 in the manuscript shows the relative importance of each input variable based on the gradient in Equation 7 of the manuscript. The figure shows 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.

C.4 Interactions within or across categories: Results

TABLE C.2: **Magnitude of Cross- and Within-group Interactions**

This table reports the sum of the absolute value of the cross-group (Panel A) and within-group (Panel B) interactions obtained from an ensembled neural network with one hidden layer ("Groups ensemble") and a (non-ensembled) neural network with three hidden layers ("Fully connected network"). See Table 2 in the manuscript for their performance (row "NN 3 Layer (32, 16, 8 nodes), fwd rates direct" and "NN 1 Layer Group Ensem (1 node per group), fwd rates Net (1 layer: 3 nodes)"). Interactions magnitudes are computed by numerically approximating the network Hessian's, i.e. the second derivatives of network outputs with respect to two distinct network inputs. The NNs take as inputs both macroeconomic variables and forward rates. The out-of-sample prediction errors are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12.

Panel A: Sum of Cross-Group Hessian Absolute Values

Model	2y	3y	4y	5y	7y	10y
Fully connected network	60.24	108.61	153.22	184.43	247.84	336.49
Groups ensemble	0.04	0.07	0.09	0.11	0.14	0.18

Panel B: Sum of Within-Group Hessian Absolute Values

Model	2y	3y	4y	5y	7y	10y
Fully connected network	11.93	21.30	30.12	36.03	48.51	65.77
Groups ensemble	16.93	32.05	45.09	54.70	78.85	117.05

C.5 Model uncertainty: Results

TABLE C.3: **Alternative Model Combination Strategies**

This table reports the out-of-sample R^2_{oos} obtained using a large panel of macroeconomic variables and forward rates to predict annual excess bond returns for different maturities. In addition to the best performing neural network – *NN 1 Layer Group Ensem + fwd rate net* (see Table 2 in the manuscript), we compute the R^2_{oos} for three alternative model combination strategies. The first is a recursive full cross-validation that selects every five years not only the dropout rate and the L1/L2 penalties, but also the number of layers, the nodes per macro group, and the nodes in fwd rate net. The second and third model combination are based on a weighted average of each neural network forecasts with weights that are calculated as the inverse of the validation loss or, alternatively, simply as $1/N$ where N is the number of neural networks estimated. The out-of-sample R^2_{oos} is calculated by considering the expectations hypothesis as a benchmark. That is, we compare the forecasts obtained from each methodology to the prediction based on the historical mean of bond excess returns. In addition to the R^2_{oos} we report the p-value calculated as in Clark and West (2007) in parentheses. The out-of-sample prediction errors are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12.

Models	R^2_{oos}						R^2_{oos} EW
	$xr^{(2)}_{t+1}$	$xr^{(3)}_{t+1}$	$xr^{(4)}_{t+1}$	$xr^{(5)}_{t+1}$	$xr^{(7)}_{t+1}$	$xr^{(10)}_{t+1}$	$xr^{(EW)}_{t+1}$
NN 1 Layer Group Ensem (1 node per group), fwd rate net (1 layer: 3 nodes)	20.0%	25.6%	29.5%	31.2%	33.6%	36.3%	34.0%
	(0.002)	(0.001)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
Inverse Val. Loss Weighted Model (across all NNs)	22.3%	25.6%	29.1%	30.8%	32.0%	34.4%	32.6%
	(0.002)	(0.001)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
Equally Weighted Model (across all NNs)	22.0%	25.2%	28.6%	30.4%	31.6%	33.9%	32.1%
	(0.002)	(0.001)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
Full CV Model	24.7%	27.6%	30.4%	31.7%	32.3%	34.7%	33.4%
	(0.001)	(0.001)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)

TABLE C.4: **Stability of the Neural Network Ranking**

This table reports how often the four best performing neural networks are selected throughout the sample. More specifically, we select the top 4 models based on the unconditional average of (inverse) validation loss. Then we count how often the four models rank 1st, 2nd, 3rd and 4th, in terms of their inverse validation error. The sum of the values in each column equals the number of periods in our out-of-sample period. This gives an approximate measure of how often one model ranks on top in terms of validation loss. The out-of-sample prediction errors are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12.

	Ranking			
Model	1st place	2nd place	3rd place	4th place
NN 1-Layer Group Ensem + Fwd. Rate Net	170	42	115	21
NN 1-Layer Group Ensem + Fwd. Rates	137	50	154	7
NN 3-Layer + Fwd. Rates	31	112	16	189
NN 2-Layer Group Ensem + Fwd. Rates Net	10	144	63	131

C.6 Model performance and lagged macroeconomic variables

We assess the robustness of the performance of our non-linear methods with respect to the timing of the macroeconomic predictors. Information flow to investors could occur with a lag such that macroeconomic information is not available right away, therefore leading to our models overfitting. In order to rule out that this is the case, we follow Rapach and Zhou (2019) and lag all macroeconomic variables by one month to account for data becoming available with a delay. Note, we do not lag market-based variables including interest rates, exchange rates, stock market data and the oil price since these variables are available to investors virtually in real-time.

In Table C.5 we summarize the results of the recursive forecasting exercise with lagged macroeconomic variables. Overall, we find that the performance of the non-linear models that are at the center of our analysis is robust to lagging the macroeconomic predictors by one month. Also, the ordering of models in terms of their predictive performance is basically unchanged: the set of neural networks outperforms the regression tree approaches as long as the NNs are deep enough or use group ensembling. We continue to find that a shallow group ensembled neural network performs on par with or better than a deeper neural network that does not group variables according to their macroeconomic character. In particular, the ensembled neural network with 1 hidden layer per group of macroeconomic variables and a network for the forward rates continues to perform best out of all models studied: the overall model performance (R^2 EW of 34.0%, c.f. Table 2 in the manuscript) without lagging the predictors is on par with the model performance when predictors are lagged (R^2 EW of 32.8%).

Finally, let us observe that Table 2 in the manuscript and Table C.5 below rely on two different frameworks, and demand to be interpreted accordingly. Manuscript Table 2 relies on the "economic agents know" framework: the econometrician has limited information relative to *economic agents who*, in contrast, *know* the history of prior macroeconomic data and how bond returns react to real quantities. This is the relevant framework for us, since our main objective is, through the lens of machine learning methods, to provide a better understanding of the dynamics of bond risk premia and its economic drivers, rather than to create a trading

strategy. For further discussions on how to interpret out-of-sample tests see, e.g., Atanasov, Moller and Priestley (2019).

TABLE C.5: **Forecasting Annual Holding-period Returns with Forward Rates and Lagged Macroeconomic Variables**

This table reports the out-of-sample R^2_{oos} obtained using forward rates and a large panel of macroeconomic variables to predict annual excess bond returns for different maturities when macroeconomic variables are lagged by one month. Given that macroeconomic variables that are market-based are available in close to real-time, variables pertaining to the interest rates, exchange rates, stock market categories as well as oil price are not lagged. All other variables are lagged by one month. To compute the out-of-sample R^2_{oos} we compare the forecasts obtained from each methodology to the expectation hypothesis (i.e., prediction based on the historical mean). In addition to the R^2_{oos} we report the p-value for the null hypothesis $R^2_{oos} \leq 0$ calculated as in Clark and West (2007). Notice that we report a p-value only when the R^2_{oos} is positive. The out-of-sample prediction errors are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12. Penalized regressions are estimated including macro-economic variables plus 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 an additional regressor in the output layer ("fwd rates direct") or by estimating a separate network for forward rates and ensembling both macro and forward rates networks in the output layer ("fwd rates net").

Models	R^2_{OOS}						p-value						R^2_{OOS} EW	p-value EW
	$xr^{(2)}_{t+1}$	$xr^{(3)}_{t+1}$	$xr^{(4)}_{t+1}$	$xr^{(5)}_{t+1}$	$xr^{(7)}_{t+1}$	$xr^{(10)}_{t+1}$	$xr^{(2)}_{t+1}$	$xr^{(3)}_{t+1}$	$xr^{(4)}_{t+1}$	$xr^{(5)}_{t+1}$	$xr^{(7)}_{t+1}$	$xr^{(10)}_{t+1}$	$xr^{(EW)}_{t+1}$	$xr^{(EW)}_{t+1}$
Gradient Boosted Regression Tree	15.3%	18.3%	15.5%	22.2%	24.2%	21.7%	0.000	0.003	0.006	0.003	0.004	0.005	24.8%	0.3%
Random Forest	20.1%	13.5%	15.5%	20.4%	25.6%	29.8%	0.007	0.006	0.006	0.002	0.002	0.001	27.7%	0.1%
Extreme Tree	22.8%	14.1%	16.5%	23.0%	23.4%	30.4%	0.005	0.004	0.006	0.002	0.004	0.002	26.2%	0.3%
NN 1 Layer (32 nodes), fwd rates direct	9.3%	11.3%	17.5%	19.9%	22.9%	26.3%	0.002	0.001	0.000	0.000	0.000	0.000	21.8%	0.0%
NN 2 Layer (32, 16 nodes), fwd rates direct	16.5%	21.5%	24.7%	27.5%	29.0%	31.6%	0.000	0.000	0.000	0.000	0.000	0.000	28.6%	0.0%
NN 3 Layer (32, 16, 8 nodes), fwd rates direct	22.2%	25.6%	29.8%	31.5%	32.1%	34.4%	0.000	0.000	0.000	0.000	0.000	0.000	32.6%	0.0%
NN 1 Layer (32 nodes), fwd rates net (1 layer: 3 nodes)	4.2%	15.0%	19.8%	24.7%	26.5%	29.6%	0.006	0.003	0.002	0.001	0.001	0.001	25.9%	0.1%
NN 2 Layer (32,16, nodes), fwd rates net (1 layer: 3 nodes)	12.6%	19.7%	23.2%	25.5%	27.1%	29.2%	0.006	0.002	0.001	0.001	0.001	0.001	27.1%	0.1%
NN 3 Layer (32,16, 8 nodes), fwd rates net (1 layer: 3 nodes)	10.0%	17.0%	22.1%	25.3%	27.5%	30.9%	0.014	0.005	0.003	0.001	0.001	0.001	27.2%	0.1%
NN 1 Layer Group Ensem (1 node per group), fwd rates direct	22.1%	25.6%	28.5%	30.7%	30.8%	32.0%	0.001	0.000	0.000	0.000	0.000	0.000	31.3%	0.0%
NN 1 Layer Group Ensem (1 node per group), fwd rate net (1 layer: 3 nodes)	19.0%	24.5%	28.7%	30.8%	32.8%	34.9%	0.003	0.002	0.001	0.001	0.001	0.001	32.8%	0.1%
NN 2 Layer Group Ensem (2,1 nodes per group / hidden layer), fwd rate net (2 layer: 3 nodes)	14.8%	21.7%	25.6%	28.0%	29.5%	31.5%	0.008	0.003	0.002	0.001	0.001	0.000	29.9%	0.1%
NN 3 Layer Group Ensem (3, 2, 1 nodes per group / hidden layer), fwd rate net (3 layer: 3 nodes)	13.7%	20.5%	24.8%	27.1%	29.1%	32.0%	0.011	0.004	0.002	0.001	0.001	0.001	29.1%	0.1%

D Economic Value of Excess Bond Returns Forecasts: Additional Discussions and Tests

D.1 The asset allocation framework for a power utility investor

We consider the investment decision of an agent that selects the weights on the risky n-period bonds $\boldsymbol{w}_t = \left[w_t^{(2)} \dots w_t^{(10)}\right]$ versus a one-period bond that pays a risk-free rate equal to $y_t^{(1)}$. In the main text we discuss an investor with mean-variance utility; here we discuss an extended framework whereby a representative investor has a power utility of the form,

$$U\left(\boldsymbol{w}_t, \boldsymbol{xr}_{t+1}\right) = \frac{\left[\left(1 - \boldsymbol{w}_t'\iota\right)\exp\left(y_t^{(1)}\right) + \boldsymbol{w}_t'\exp\left(y_t^{(1)}\iota + \boldsymbol{xr}_{t+1}\right)\right]^{1-\gamma}}{1-\gamma}, \qquad \gamma > 0 \tag{D.1}$$

where γ captures the investor's risk aversion and ι is a vector of ones. We follow Campbell and Viceira (1999), Campbell and Viceira (2004) and Gargano et al. (2019) and assume excess bond returns are jointly log-normal distributed so that the excess returns on a portfolio of treasury bond can be approximated by

$$R_{p,t+1} = 1 + y_t^{(1)} + \boldsymbol{w}_t'\boldsymbol{xr}_{t+1} + \frac{1}{2}\boldsymbol{w}_t'\boldsymbol{\sigma}_{t+1|t}^2 - \frac{1}{2}\boldsymbol{w}_t'\Sigma_{t+1|t}\boldsymbol{w}_t \tag{D.2}$$

where $\Sigma_{t+1|t}$ denotes the covariance matrix of the excess bond returns, and we denote with $\boldsymbol{\sigma}_{t+1|t}$ its diagonal elements. Campbell and Viceira (2004) show that under log-normality of excess returns, the optimal allocation on a maturity-specific bond can be defined as

$$w_t^{(n)} = \frac{1}{\gamma\left(\sigma_{t+1|t}^{(n)}\right)^2}\left[\widehat{xr}_{t+1}^{(n)} + \left(\sigma_{t+1|t}^{(n)}\right)^2/2\right] \tag{D.3}$$

Given these optimal set of weights, the realized utility for, say, the univariate case can be computed by plugging $\boldsymbol{w}_t$ into equation (D.1).

Similar to the mean-variance case (see Section 5.1 of the) we proxy for $\Sigma_{t+1|t}$ by using a rolling sample variance estimator as in Thornton and Valente (2012), and set the coefficient

of relative risk aversion equal to $\gamma = 5$. Further, to test if the annualized CER values are statistically greater than zero, we employ again a Diebold and Mariano (1995) test. Specifically, for a power utility investor that selects a single risky asset with maturity n using when the forecast from a NN, we estimate the regression

$$u^{(n)}_{t+1,NN} - u^{(n)}_{t+1,EH} = \alpha^{(n)} + \varepsilon_{t+1}$$

where

$$u^{(n)}_{t+1,j} = \frac{1}{1-\gamma}\left[(1-\omega^{(n)}_{t,j})\exp(y^{(1)}_t) + \omega^{(n)}_{t,j}\exp(y^{(1)}_t + xr^{(n)}_{t+1})\right]^{1-\gamma}$$

and $j = \{EH, NN\}$.

D.2 Further results on the economic value of return forecasts

In addition to the benchmark case that restricts the weights to be in the interval $-1 \leq w_t^{(n)} \leq 2$ to prevent extreme investments, we also consider two alternative scenarios. The first scenario restricts the optimal weights to be non-negative, i.e., $w_t^{(n)} \in (0, 0.99)$. Such scenario ensures that the expected utility is finite even in the case of unbounded returns (see Geweke, 2001 for a discussion). The second scenario leaves the portfolio weights unrestricted but instead restricts the bond returns to fall in the interval (−100%, +100%). Similar to restricting the weights to be non-negative, such restriction prevents the expected utility from being unbounded (see Johannes et al., 2014).

The results in Table D.1 confirm our results in the main text (c.f. Section 5 of the manuscript). For instance, Panel A shows that including macroeconomic information tend to improves the economic performance with respect to a model which exploits only information from the yield curve, and that forecasts from the best performing neural network achieves a higher utility with respect to a competing non-linear model such as extreme trees.

Panel B shows that by winsorizing the returns instead of the optimal weights the conclusion remains qualitatively the same; again macroeconomic information brings additional economic value and even more so when neural networks are used. As a whole, these results are in line with the main evidence described in Section 5 of the manuscript.

TABLE D.1: **Economic Significance of Bond Predictability: Robustness**

This table reports two robustness exercises concerning the out-of-sample economic performance reported in Table 5 of the manuscript. Panel A reports the results for the weights restricted to be non-negative, whereas Panel B restricts the realized returns on each maturity to be between -100% and 100%. We report results for an investor with either mean-variance or power utility and a coefficient of risk aversion equal to five. The models are benchmarked against the expectation hypothesis. The table reports both multi-asset results and the case of a single risky asset. In the univariate asset allocation case, the investor selects either the two- or the ten-year bond, along with the one-year short-rate. In the multivariate case, the investor selects bond excess returns across the six maturities, two- to five-, seven- and ten-years. The asset allocation decision is based on the predictions implied by either the best performing regression tree specification, i.e., extreme tree, or the best performing neural network, namely the *NN 1 Layer (3 nodes)* – when forecasting with only the forward rates – and *NN 1 Layer Group Ensem + fwd rate net* – when including also macroeconomic variables – (see Table 1-2 of the manuscript for reference). The row Δ reports the value added by NN relative to extreme tree within each application ("Fwd rates" and "Fwd + Macro"). The column Δ reports the value added by "Fwd+Macro" relative to "Fwd rates" within each model (NN and extreme tree). The models are benchmarked against the expectation hypothesis. The out-of-sample predictions are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12. Statistical significance is based on a one-sided Diebold and Mariano (1995) test extended by Harvey et al. (1997) to account for autocorrelation in the forecasting errors. We flag in bold those values that are statistically significant at the 5% confidence level.

Panel A: Non-negative weights

		2-year maturity			10-year maturity			All		
		Fwd rates	Fwd + Macro	Δ	Fwd rates	Fwd + Macro	Δ	Fwd rates	Fwd + Macro	Δ
Mean-Variance	Neural net	0.017	**0.025**	0.007	**0.915**	**1.242**	**0.318**	**2.340**	**3.867**	**1.527**
	p-value	(0.322)	(0.000)	(0.557)	(0.011)	(0.000)	(0.041)	(0.010)	(0.011)	(0.000)
	Extreme tree	0.034	0.011	-0.023	**0.958**	**0.863**	0.159	**2.812**	**3.376**	**0.555**
	p-value	(0.416)	(0.512)	(0.449)	(0.003)	(0.000)	(0.473)	(0.009)	(0.009)	(0.004)
	Δ	-0.017	0.013		**-0.043**	**0.379**		**-0.472**	**0.491**	
	p-value	(0.597)	(0.471)		(0.261)	(0.014)		(0.023)	(0.015)	
Power Utility	Neural net	**0.045**	**0.098**	**0.053**	**1.235**	**1.765**	**0.530**	**2.624**	**4.133**	**1.509**
	p-value	(0.000)	(0.000)	(0.010)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
	Extreme tree	0.007	**0.075**	**0.068**	**1.507**	**1.198**	**-0.308**	**2.568**	**2.995**	0.427
	p-value	(0.763)	(0.000)	(0.004)	(0.000)	(0.000)	(0.010)	(0.000)	(0.000)	(0.242)
	Δ	**0.038**	0.023		-0.272	**0.566**		0.056	**1.138**	
	p-value	(0.034)	(0.057)		(0.055)	(0.000)		(0.820)	(0.000)	

Panel B: Winsorized returns

		2-year maturity			10-year maturity			All		
		Fwd rates	Fwd + Macro	Δ	Fwd rates	Fwd + Macro	Δ	Fwd rates	Fwd + Macro	Δ
Mean-Variance	Neural net	0.044	0.088	0.045	**0.910**	**2.331**	**1.413**	**2.551**	**3.143**	**0.592**
	p-value	(0.110)	(0.091)	(0.098)	(0.000)	(0.000)	(0.000)	(0.011)	(0.006)	(0.012)
	Extreme tree	0.028	0.078	0.050	**1.191**	**2.171**	**0.970**	1.156	**2.756**	**1.600**
	p-value	(0.261)	(0.076)	(0.094)	(0.023)	(0.000)	(0.000)	(0.063)	(0.001)	(0.002)
	Δ	0.016	0.009		-0.282	0.160		**1.395**	**0.387**	
	p-value	(0.171)	(0.519)		(0.067)	(0.087)		(0.016)	(0.023)	
Power Utility	Neural net	**0.056**	**0.110**	**0.054**	**1.170**	**3.143**	**1.961**	**2.294**	**4.411**	**2.117**
	p-value	(0.012)	(0.011)	(0.034)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
	Extreme tree	0.022	0.023	0.001	**1.518**	**2.681**	**1.162**	**2.601**	**3.712**	**1.111**
	p-value	(0.486)	(0.324)	(0.577)	(0.000)	(0.000)	(0.006)	(0.000)	(0.000)	(0.024)
	Δ	0.034	**0.087**		**-0.348**	**0.459**		-0.307	**0.697**	
	p-value	(0.093)	(0.032)		(0.041)	(0.034)		(0.169)	(0.039)	

D.3 Economic drivers of portfolio performance

In this section, we investigate the potential drivers of the economic gains from bond return predictability. Intuitively one may expect large economic gains during bad times, when uncertainty and disagreement are large. Indeed Sarno et al. (2016) find large economic gains from predictability of bond returns during times with high macroeconomic uncertainty.

Specifically, we study the relation between the realized utility obtained in the portfolio analysis of Section 5 in the manuscript and the structural risk factors presented in Section 6.2 of the manuscript. Table D.2 reports the results when we employ as predictors the macro variables from the FRED-MD database in addition to forward rates. Panels A and B show the analysis for the CER obtained by a mean-variance and power utility investor, respectively.

Most of the results confirm the evidence in Table 7 in the manuscript for our forecasts of excess bond returns. Indeed, across panels, we find that bond volatility is only weakly related to realized utility. The link between our realized utility and nominal disagreement is positive and statistically significant whereas we find only a weak association with real disagreement. This confirms the recent findings of Gargano et al. (2019) that inflation disagreement is an important driver of portfolio performance.[4] Other variables stand out as important determinants of variation in realized utility, namely the risk aversion proxies and (macro and inflation) uncertainty. Interestingly, in a kitchen sink regression where we preselect predictors based on their statistical significance (specification (vi)), we continue to find that variation in risk aversion (as proxied by the measure of Bekaert et al., 2019), and macroeconomic uncertainty are important drivers of the economic gains.

The results in Appendix Table D.3 show the case for yield-only based forecasts. The results substantiate further our conclusion that the relation between utility gains from our portfolio analysis is the strongest with risk aversion and time-varying uncertainty.

In all, the evidence in Table 7 of the manuscript for our forecasts of excess bond returns, and Tables D.2 and D.3 in this appenidx for the risk-adjusted economic gains, paints a consistent picture of a significant link between time-varying risk aversion and risk, and bond risk premia.

[4] Gargano et al. (2019) call the cross-sectional inter-quartile range in GDP and CPI forecasts uncertainty. We follow Buraschi et al. (2019) and refer to the cross-sectional inter-quartile range in forecasts as to disagreement.

This link is obfuscated when using future realized returns but becomes apparent when using expected returns obtained from machine learning methods. Therefore, models like Bekaert et al. (2009) and Creal and Wu (2018) that combine time variation in economic uncertainty with changes in risk aversion seem to be a promising avenue for research.

TABLE D.2: **Drivers of Portfolio Performance (Forecasts based on Forward Rates + Macro Variables)**

This table reports the regression estimates of economic gain obtained from a mean-variance (panel A) and power utility (panel B) on a set of structural determinants of risk premia (see discussion in the paper for details). The economic gains are based on the predictions implied by the best performing neural network when both yields and macroeconomic variables are considered as predictors, namely the *NN 1 Layer Group Ensem + fwd rate net* (see Table 2 of the manuscript for reference). We standardize both left and right hand variables, so that a 1-standard deviation change in the right hand variables implies a β-standard deviation in the dependent variable. We report the regression estimates as well as Newey-West p-values. Bold font indicates significance at the 5% level. The out-of-sample predictions are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12.

Panel A: Mean-variance utility

	$DiB(g)$	$DiB(\pi)$	$-Surplus$	$RAbex$	$UnC(g)$	$UnC(\pi)$	$TYVIX$	$\sigma_B^{(n)}$	$R^2(\%)$
(i)	0.07	0.05							6.76
	(0.06)	(0.14)							
(ii)			0.03						15.51
			(0.65)						
(iii)				**0.02**					6.59
				(0.01)					
(iv)					**0.07**	**0.08**			25.61
					(0.00)	(0.00)			
(v)							0.06	-0.01	0.50
							(0.64)	(0.75)	
(vi)		0.03	0.03	**0.02**	**0.08**	**0.09**			29.52
		(0.25)	(0.78)	(0.01)	(0.00)	(0.00)			

Panel B: Power utility

	$DiB(g)$	$DiB(\pi)$	$-Surplus$	$RAbex$	$UnC(g)$	$UnC(\pi)$	$TYVIX$	$\sigma_B^{(n)}$	$R^2(\%)$
(i)	-0.01	**0.03**							10.27
	(0.34)	(0.02)							
(ii)			**0.56**						20.84
			(0.00)						
(iii)				**0.02**					9.18
				(0.00)					
(iv)					**0.08**	**0.13**			10.16
					(0.04)	(0.00)			
(v)							0.02	-0.00	0.90
							(0.14)	(0.99)	
(vi)		0.02	0.04	**0.02**	**0.03**	**0.02**			33.77
		(0.29)	(0.06)	(0.00)	(0.01)	(0.01)			

TABLE D.3: **Drivers of Portfolio Performance (Forecasts based on Forward Rates only)**

This table reports the regression estimates of economic gain obtained from a mean-variance (panel A) and power utility investor (panel B) on a set of structural determinants of risk premia (see discussion in the paper for details). The economic gains are based on the predictions implied by the best performing neural network when only yields are considered as predictors, namely the *NN 1 Layer (3 nodes)* (see Table 1 of the manuscript for reference). We standardize both left and right hand variables, so that a 1-standard deviation change in the right hand variables implies a β-standard deviation in the dependent variable. We report the regression estimates as well as Newey-West p-values. Bold font indicates significance at the 5% level. The out-of-sample predictions are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12.

Panel A: Mean-variance utility

	$DiB(g)$	$DiB(\pi)$	$-Surplus$	$RAbex$	$UnC(g)$	$UnC(\pi)$	$TYVIX$	$\sigma_B^{(n)}$	$R^2(\%)$
(i)	**0.06**	0.03							12.30
	(0.03)	(0.12)							
(ii)			**0.02**						10.41
			(0.00)						
(iii)				**0.01**					6.99
				(0.01)					
(iv)					**0.04**	**0.12**			8.28
					(0.00)	(0.00)			
(v)							0.01	-0.01	-1.16
							(0.72)	(0.38)	
(vi)		0.03	0.01	**0.02**	**0.03**	**0.03**			18.12
		(0.14)	(0.07)	(0.02)	(0.00)	(0.01)			

Panel B: Power utility

	$DiB(g)$	$DiB(\pi)$	$-Surplus$	$RAbex$	$UnC(g)$	$UnC(\pi)$	$TYVIX$	$\sigma_B^{(n)}$	$R^2(\%)$
(i)	-0.02	0.02							15.58
	(0.12)	(0.09)							
(ii)			**0.38**						20.38
			(0.00)						
(iii)				**0.01**					9.18
				(0.03)					
(iv)					**0.09**	**0.10**			7.24
					(0.02)	(0.02)			
(v)							**0.02**	-0.01	6.47
							(0.02)	(0.12)	
(vi)		0.02	0.05	**0.01**	**0.03**	**0.02**			30.08
		(0.09)	(0.09)	(0.01)	(0.01)	(0.01)			

E Algorithmic Procedures

In this section we provide details on the algorithmic procedures used for each class of models implemented in the main empirical analysis. We start from the simple penalized regressions, e.g., lasso, ridge, and elastic net. We then turn to non-linear methods starting with shallow regression trees and random forest. We conclude by discussing the different neural network specifics.

E.1 Partial least squares

Following the extant practice (see, Ch.3.5 Friedman et al., 2001) Partial Least Squares (PLS) is constructed iteratively as a two-step procedure: in the first step we regress excess bond 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 excess bond returns and the predictors. In the second step, excess bond 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.[5]

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 excess bond 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.[6] PLS does not require the calibration of hyperparameters as the derived input directions are deterministically obtained by the two-step procedure outlined above. In this respect, unlike penalized regressions

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

[6] In particular, the m-th direction solves:

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

no shrinkage/regularization parameters are required to be calibrated.

E.2 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

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

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. (E.1b) and (E.1c)). 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

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

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

$$-\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \alpha - \boldsymbol{x}_i^\top \hat{\beta}) + \lambda(1-\mu)\beta_j + \lambda\mu \tag{E.3}$$

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 λ and μ, 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 $j = 1, \ldots, p$.
Set ϵ 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* $j = 1, \ldots, p$ **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 λ, μ;

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)

$$\hat{\beta}^{\text{Ridge}} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{E.4}$$

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 circumvent issues incurred by matrix inversion. Thus, we use a singular value decomposition (SVD) of the matrix $\boldsymbol{X}$ with the form

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top \tag{E.5}$$

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

$$\boldsymbol{X}\hat{\beta}^{\text{Ridge}} = \boldsymbol{U}\boldsymbol{D}\left(\boldsymbol{D}^2 + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{D}\boldsymbol{D}^\top \boldsymbol{y}. \tag{E.6}$$

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

E.3 Tree-based methods

Regression trees can approximate any a priori unknown function while keeping the interpretation from a recursive binary tree. However, with more than two inputs, the interpretation is less obvious as trees like the one depicted in Figure 1 of the manuscript grow exponentially in size. Nevertheless the algorithmic procedure is equivalent. Suppose one deals with a partition

of M regions $\mathcal{A} = \{A_1, \ldots, A_M\}$ of the vector of yields $\boldsymbol{y}_t$ such that

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

By minimizing the sum of squared residuals, one can show that the optimal estimate $\hat{\beta}_m$ is just the average of the excess bond 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.

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 *de-correlated* 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 also consider an extended version of the random forest procedure which is called "Extremely Randomized Trees" (Geurts et al., 2006) . While similar to ordinary random forests, in that they still represent an ensemble of individual trees, extreme trees have two main distinguishing features: first, each tree is trained using the whole training sample (rather than a bootstrap sample); and second, the top-down splitting in the tree learner is randomized. That means that instead of computing the optimal cut-point locally for each input variable under consideration, a random cut-point is selected. In other words, with extreme trees the split of the trees is stochastic; with random forests the split is instead deterministic.

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).

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 $\{R_l(d), l = 1, \dots, 2^{d-1}\}$ **do**

    Given splitting variable j and split point s define regions

$$R_{\text{left}}(j,s) = \{X \mid X_j \le s, X_j \cap \tilde{R}\} \quad \text{and} \quad R_{\text{right}}(j,s) = \{X \mid X_j > s, X_j \cap \tilde{R}\}$$

    In the splitting regions set

$$c_{\text{left}}(j,s) \leftarrow \frac{1}{|R_{\text{left}(j,s)}|} \sum_{x_i \in R_{\text{left}(j,s)}} y_i(x_i) \quad \text{and} \quad c_{\text{right}}(j,s) \leftarrow \frac{1}{|R_{\text{right}(j,s)}|} \sum_{x_i \in R_{\text{right}(j,s)}} y_i(x_i)$$

    Find j^*, s^* that optimize

$$j^*, s^* = \underset{j,s}{\text{argmin}}\Big[\sum_{x_i \in R_{\text{left}(j,s)}} (y_i - c_{\text{left}}(j,s))^2 + \sum_{x_i \in R_{\text{right}(j,s)}} (y_i - c_{\text{right}}(j,s))^2 \Big]$$

    Set the new partitions

$$R_{2l}(d) \leftarrow R_{\text{right}}(j^*, s^*) \quad \text{and} \quad R_{2l-1}(d) \leftarrow R_{\text{left}}(j^*, s^*)$$

  end

end

Output: A fully grown regression tree T of depth D. The output is given by

$$f(x_i) = \sum_{k=1}^{2^L} \text{avg}(y_i \mid x_i \in R_k(D)) \mathbf{1}_{\{x \in R_k(D)\}},$$

i.e. the average response in each region R_k at depth D.

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 hyperparameters 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 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 boosted 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 $t = 1, \ldots, F$ **do**

  for $i = 1, \ldots, N$ **do**

    Compute negative gradient of loss function evaluated 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 hyperparameter.

end

Output: $f_F(x)$ is the gradient boosted regression tree output.

E.4 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 parameters with zero mean and unit variance or variations thereof such as He et al. (2015b) 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.

Algorithm 5: Stochastic Gradient Decent with Nesterov Momentum

Initialize the vector of neural network parameters θ_0 and choose momentum parameter γ. Determine the learning rate η and set $v_0 = 0$.

while *No convergence of* θ_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 θ_t of the trained network.

Batch normalization reduces the amount of variability of predictors by adjusting and scaling the activations. This increases the stability of the neural network and 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.

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

Algorithm 6: Batch Normalization per mini-batch

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

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 from simply memorizing 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 7: Early Stopping

Initialize the validation error $\epsilon = \inf$ and define a patience ϕ, also set $k = 0$
while $k < \phi$ **do**
  Update θ 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 θ'

Finally, it is important to highlight that we use a form of forecast averaging / ensembling, i.e. we train multiple copies of networks with different seeds for the randomly drawn initial network weights. Using fixed seeds will in general lead to replicable results. Nevertheless, different seeds will produce different forecasts as discussed also in Gu et al. (2018). Therefore, in order to reduce prediction variance, we average over forecasts from networks initialized with different seeds. To be precise, for each time t we initialize 100 models with different but fixed seeds. The 100 models are then trained and as part of the training we obtain the validation sample loss. The validation sample loss is then used to select the 10 out 100 models with the smallest validation sample error. Finally, we average the forecasts of those 10 in-sample best performing models.

F Computational Details

For our implementation of the various machine learning techniques in Python we utilize the well-known packages `Scikit-Learn`[7] and `Tensorflow`[8] in the `Keras`[9] 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 hyperparameters. 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,

[7] `http://scikit-learn.org/stable/`, as of 26th October 2018
[8] `https://www.tensorflow.org/`, as of 26th October 2018
[9] `https://keras.io/`, as of 26th October 2018

L1 / L2 penalties, drop-out, and batch normalization.

F.1 Setup

Since the forecasting exercise in this paper is iterative and since we use model averaging, the computational challenge becomes sizable. 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`[10] 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.

F.2 Full cross-validated neural network vs. group-ensemble

In Section C.5 we compare our best performing group-ensemble model (labeled as NN 1 Layer Group Ensem (1 node per group), fwd rate net (1 layer: 3 nodes)) against two different types of model averaging schemes, i.e., weighting based on the inverse of the validation loss and an equal-weighted model, as well as a fully cross-validated (CV) network. While the logic of the weighting schemes for the model-averaging may be intuitive, the specifications for the fully CV network may be not. Table F.1 gives an outline of these specifications vis-a-vis the choice made for the group-ensemble structure.

A number of aspects should be discussed; in the full CV setting we recursively choose the number of hidden layers (between 1 and 2) as well as the number of nodes for each group of macroeconomic variables. Similarly, the number of nodes in the forward rate net is allowed to change whereas for the group-ensemble it is kept fixed. In addition, the full CV allows

[10] `https://docs.python.org/3.4/library/multiprocessing.html`, as of 26th October 2018

TABLE F.1: **Specifics of Full Cross-Validation vs. Group-Ensembling**

This table reports the hyperparameters for the full cross-validated network vs. the shallow network with group ensemble (see Table C.3). The out-of-sample predictions are obtained by a recursive forecast which starts in January 1990. The sample period is from 1971:08-2018:12.

Hyperparameter Set	Full CV	CV for group ensemble
Number of Layers	1, 2	1
Nodes Per Group	1, 2, 3	1
Nodes in Fwd Rate Net	1, 3	3
Dropout - Group Net	0.1, 0.3, 0.5	0.1, 0.3, 0.5
Dropout - Fwd Rate Net	0, 0.3	0
L1/L2 Penal - Group Net	0.01, 0.001, 0.0001	0.01, 0.001
L1/L2 Penal - Fwd Rate Net	0.001, 0.0001	0.0001
Combinations per retraining	432	6
CV frequency	5 years	5 years

for a bit more flexibility in the L1/L2 penalty terms for both the group-specific networks as well as for the forward rates. Increasing the flexibility comes at the cost of increased computational expense (423 possible combinations vs. 6). To keep the computational cost manageable we employ a procedure referred to as randomized cross-validation that randomly draws combinations of hyperparameters from the set of available combinations. Specifically, we perform randomized cross-validation over 60 specifications from the the set of hyperparameters above. Note, even under randomized cross-validation as we employ it here the effective training time increases by a factor of 10 (60 specifications vs 6 specifications).

References

Ahn, Dong Hyun, Robert F. Dittmar, and Andrew Ronald Gallant (2002) "Quadratic Term Structure Models: Theory and Evidence," *Review of Financial Studies*, Vol. 15, No. 1, pp. 243–288, 1.

Atanasov, Victoria, Stig Vinther Moller, and Richard Priestley (2019) "Consumption Fluctuations and Expected Returns," *The Journal of Finance*, Vol. n/a, No. n/a.

Bekaert, Geert, Eric Engstrom, and Yuhang Xing (2009) "Risk, uncertainty, and asset prices," *Journal of Financial Economics*, Vol. 91, No. 1, pp. 59–82.

Bekaert, Geert, Eric C. Engstrom, and Nancy R. Xu (2019) "The Time Variation in Risk Appetite and Uncertainty," NBER Working Papers 25673, National Bureau of Economic Research, Inc.

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

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.

Buraschi, Andrea and Alexei Jiltsov (2007) "Habit Formation and Macroeconomic Models of the Term Structure of Interest Rates," *The Journal of Finance*, Vol. 62, No. 6, pp. 3009–3063.

Buraschi, Andrea, Ilaria Piatti, and Paul Whelan (2019) "Subjective Bond Risk Premia and Belief Aggregation,"Technical report.

Campbell, John Y and John H Cochrane (1999) "By force of habit: A consumption-based explanation of aggregate stock market behavior," *Journal of political Economy*, Vol. 107, No. 2, pp. 205–251.

Campbell, John Y and Luis M Viceira (1999) "Consumption and portfolio decisions when expected returns are time varying," *The Quarterly Journal of Economics*, Vol. 114, No. 2, pp. 433–495.

——— (2004) "Long-horizon mean-variance analysis: A user guide," *Manuscript, Harvard University, Cambridge, MA*.

Clark, Todd E and Kenneth D West (2007) "Approximately normal tests for equal predictive accuracy in nested models," *Journal of econometrics*, Vol. 138, No. 1, pp. 291–311.

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

Creal, Drew D. and Jing Cynthia Wu (2018) "Bond Risk Premia in Consumption-based Models," NBER Working Papers 22183, National Bureau of Economic Research, Inc.

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

Dietterich, Thomas G (2000) "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, pp. 1–15.

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.

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

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.

Gargano, Antonio, Davide Pettenuzzo, and Allan Timmermann (2019) "Bond Return Predictability: Economic Value and Links to the Macroeconomy," *Management Science*, Vol. 65, No. 2, pp. 508–540.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel (2006) "Extremely randomized trees," *Machine learning*, Vol. 63, No. 1, pp. 3–42.

Geweke, John (2001) "A note on some limitations of CRRA utility," *Economics letters*, Vol. 71, No. 3, pp. 341–345.

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.

Hansen, Lars Kai and Peter Salamon (1990) "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 12, No. 10, pp. 993–1001.

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 (2015a) "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

——— (2015b) "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

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

Johannes, Michael, Arthur Korteweg, and Nicholas Polson (2014) "Sequential Learning, Predictability, and Optimal Portfolio Returns," *Journal of Finance*, Vol. 69, No. 2, pp. 611–644, April.

Le, Anh, Kenneth J. Singleton, and Qiang Dai (2010) "Discrete-Time AffineQ Term Structure Models with Generalized Market Prices of Risk," *The Review of Financial Studies*, Vol. 23, No. 5, pp. 2184–2227, 03.

Leippold, Markus and Liuren Wu (2003) "Design and Estimation of Quadratic Term Structure Models," *Review of Finance*, Vol. 7, No. 1, pp. 47–73.

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.

Nesterov, Yuri (1983) "A method for solving the convex programming problem with convergence rate $O(1/\hat{k2})$," *Dokl. Akad. Nauk SSSR*, Vol. 269, pp. 543–547.

Pastor, Lubos and Pietro Veronesi (2005) "Rational IPO Waves," *The Journal of Finance*, Vol. 60, No. 4, pp. 1713–1757.

Rapach, David and Guofu Zhou (2019) "Sparse Macro Factors," ssrn working paper.

Sarno, Lucio, Paul Schneider, and Christian Wagner (2016) "The economic value of predicting bond risk premia," *Journal of Empirical Finance*, Vol. 37, pp. 247–267.

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.

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.

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.

Veronesi, Pietro (2004) "Belief-dependent Utilities, Aversion to State-Uncertainty and Asset Prices,," crsp working papers, Center for Research in Security Prices, Graduate School of Business, University of Chicago.

Wachter, Jessica A. (2006) "A consumption-based model of the term structure of interest rates," *Journal of Financial Economics*, Vol. 79, No. 2, pp. 365–399, February.

Welch, Ivo and Amit Goyal (2008) "A comprehensive look at the empirical performance of equity premium prediction," *The Review of Financial Studies*, Vol. 21, No. 4, pp. 1455–1508.

Werbos, Paul J. (1974) "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. dissertation, Harvard University.

——— (1982) *Applications of advances in nonlinear sensitivity analysis*, pp. 762–770: Springer.

——— (1988) "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, Vol. 1, No. 4, pp. 339–356.

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.

Zou, Hui and Trevor Hastie (2005) "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.