ERASMUS UNIVERSITY ROTTERDAM

PWC RISK ADVISORY

MASTER THESIS

Interpretable Machine Learning in Asset Price Forecasting

SAM RIGTER (428904)

UNIVERSITY SUPERVISOR: S.H.L.C.G. VERMEULEN

COMPANY SUPERVISOR: C. VAN LOON

SECOND READER: DR. A. PICK

July 1, 2021

Abstract

We perform a comparative empirical analysis on the performance of Machine Learning models in the forecasting of risk premia on Equity, Government Bond and Commodity return classes using a set of predictor variables. Out-of-sample forecasting performance is compared for regularized linear models, tree ensemble methods and a Neural Network over an expanding window framework. Boosted Regression Trees outperform most models for all asset classes and the models best capture the return dynamics of the Government Bonds class. In addition, we adopt the SHapley Additive exPlanations (SHAP) methodology proposed by Lundberg and Lee (2017) to increase the models' transparency and interpret the return forecasts. In our analysis, we recognize partial relations between forecasts and predictors that align with macroeconomic theory.

Contents

1 Introduction **1**

2 Data **4**

2.1 Asset Returns 4

2.2 Predictor Variables 5

3 Methodology **6**

3.1 Model Specifications 6

3.1.1 Linear Models 7

3.1.2 Tree-based Models 7

3.1.3 Neural Network Model 9

3.2 Expanding Window Framework 11

3.3 Forecasting Performance Measures 13

3.4 Return Forecast Interpretation 15

4 Results **17**

4.1 Hyperparameter Configurations 18

4.2 Forecasting Performance 20

4.3 Crisis vs. Non-Crisis Robustness Analysis 24

4.4 Model Interpretation 24

5 Conclusion **28**

References **30**

Appendix A Data lags table **33**

Appendix B Crisis Robustness Results **34**

Appendix C Hyperparameter Timeseries Plots **36**

1 Introduction

Asset pricing models describe the relation between systematic risk and expected returns on assets. Traditional macroeconomic asset pricing models are built on the assumption that the relation between asset returns and macroeconomic variables is exclusively linear. The non-linear behaviour found in returns (Scheinkman and LeBaron (1989) and Hsieh (1991)) therefore makes traditional asset pricing models restrictive. Currently, with the rise of "Big Data" and the exponential increase in computing power (Schaller (1997)), asset pricing literature is adopting Machine Learning (ML) models, which provides various means to capture unobserved non-linear relations between returns and risk factors. We apply ML models to the forecasting of risk premia on Equity, Government Bond and Commodity classes and produce forecasts that outperform traditional methods. Additionally, we increase the models' transparency and interpret the forecasts produced by the models using the Shapley Additive exPlanations (SHAP), proposed by Lundberg and Lee (2017). We recognize multiple partial relations between the models' output and predictor variables that are in line with macroeconomic theory.

Accurate forecasting of asset returns is valuable to asset managers and market speculators and has an extensive history in literature. Sharpe (1964) introduces the Capital Asset Pricing Model (CAPM), which connects excess returns on equity to a shared market risk factor. Ross (1976) makes a key contribution to asset pricing by introducing the Arbitrage Pricing Theory, which expresses excess returns as a linear function of macroeconomic state variables. The risk associated with multiple macroeconomic indicators is significantly priced in asset returns, shown by Chen et al. (1986). It was later found that multiple factors are needed in asset pricing models to adequately describe the cross section of returns (Ferson and Harvey (1991), Fama and French (1993)). More recently, ML estimation methods are starting to be integrated into financial applications in literature. Due to their non-parametric structure, ML models are suited to capture potential non-linear relations (Atsalakis and Valavanis (2009)). Lee (2009) finds that Support Vector Machines outperform linear model specifications in forecasting financial time-series. Lasso regularization can be used to construct global equity returns with the lagged returns on equity across the globe (Rapach et al. (2013)). Heaton et al. (2016) show multiple financial applications using Deep Learning and demonstrate an application in portfolio selection.

The fundamental no-arbitrage condition is used by Chen et al. (2019) as criterion function to construct a Neural Network asset pricing model that outperforms all benchmark approaches in terms of Sharpe ratio. Gan et al. (2020) use deep learning to produce fast and accurate approximations in derivative pricing. Gu et al. (2020) perform a comparative analysis between ML methods in return modelling and demonstrate economic gains to investors using ML predictions.

ML tools are known for being able to achieve high forecast precision (Ahmed et al. (2010)). However, due to their non-parametric structure, most methods suffer from a lack of interpretability. ML models can be difficult to interpret and are generally considered "black-box" models. A lack of understanding and transparency of ML models can have severe consequences in finance as well as other domains (Rudin (2019)). Financial managers must be able to justify financial decisions to stakeholders, and therefore cannot fully rely on models that are not transparent. For this reason many financial institutions are skeptical of its potential (Doshi-Velez and Kim (2017)). Recently, more light is shed on ML interpretability, uncovering "black-box" models (Du et al. (2019)). Model agnostic methods that can uncover the "black-boxes" and explain patterns in the model predictions are evolving. Ribeiro et al. (2016) propose model-agnostic approaches for explaining ML predictions and review the local model agnostic Local Interpretable Model-Agnostic Explanation (LIME) method. Lundberg and Lee (2017) propose the SHAP method, which measures the contributions of feature training data to prediction output in ML models. SHAP is based on Shapley values, introduced by Shapley (1953) to compute the payout distribution across players in coalitional game theory. Layer-wise Relevance Propagation can be used to interpret factor contributions to return forecasts produced by a Neural Network specification (Nakagawa et al. (2018)). Bracke et al. (2019) compute Shapley values to address the black box problem that is present in the default risk modelling done by the Bank of England.

This research attempt to explain if ML techniques can help fulfill the demand for both accurate and explainable asset pricing models in the financial industry. In this paper, we perform an empirical study to investigate the out-of-sample forecasting performance and we interpret the forecasts produced by the models. Using a set of predictor variables that describe the current state of the U.S. economy, we model returns on U.S. equity, government bonds and commodity indices. We implement different ML solutions that are commonly used in practice, which can be categorized as regularized linear models, tree (ensemble) methods and a Neural Network architecture. The return forecasts produced by the models are compared on forecasting precision with forecasts produced by a standard linear asset pricing model. Furthermore, we adopt the SHAP methodology proposed by Lundberg and Lee (2017), to uncover the contributions of predictor variables to return forecasts in the ML models. We construct variable importance measures and partial dependence plots to gain a better understanding of the return forecasts of the estimated models.

We find that the performance of the ML models differs per asset class. Boosted Regression Trees (BRTs) outperform most models in all asset classes in terms of Mean Absolute Scaled Error (MASE). A naive model that produces forecasts by averaging over the training sample is difficult to outperform in the equity and commodity classes. The best forecast results are achieved for the Government Bonds, yielding Out-of-Sample R^2 values of above sixty percent. Furthermore, we construct feature importance and partial dependence plots for the return forecasts produced by the Random Forest (RF), BRT and Neural Network (NN) models. We find that the models capture non-linear influences of the predictor variables on the asset returns. We recognize multiple macroeconomic phenomena in the partial dependence relations between the return forecasts and the predictor variables.

The contribution of this research is two-fold: (1) we examine return forecast accuracy of the different ML asset pricing models, and (2) we interpret the forecasts produced by the models. We do this by explaining how the return forecasts of non-transparent ML models depend on the predictor variables. The return modelling of equity funds, government bonds and commodity price levels is of interest to mutual funds, pension funds, banks and other financial institutions which hold large positions in these asset classes. The exponential growth in computing power over the last decades combined with the vast amounts of (financial) data provides interesting forecasting opportunities using ML. Achieving superior investment results using asset pricing models based on ML can be of great economic value for these asset managers. But it is often desirable to be able to explain how financial models produce their output. By interpreting the model's output using SHAP, we aim to recognize relations between returns and predictor variables that are in line with well accepted macroeconomic theory. Through this analysis, we aim to give insight into the applicability of ML solutions in the finance sector. The attribution to interpretability and model transparency is not limited to asset managers only, but also for risk managers and other types of business.

The remainder of this paper is organized as follows: In Section 2 we discuss the financial return data and predictor variables that we use in our empirical analysis. We introduce the ML models, forecast accuracy measures and SHAP interpretation framework in Section 3. Section 4 presents and discusses the results of our empirical analysis. Finally, we conclude our research in Section 5.

2 Data

In this section, we discuss the data that is used in this research to model the asset returns, and the predictor variables that are used to compute the predictions. Section 2.1 describes the return data of the Equity, Government Bonds and Commodity asset classes. Section 2.2 describes the considered predictor variables in detail.

2.1 Asset Returns

We study the excess returns of N = 3 asset classes: Equity (EQ), U.S. Government Bonds (GOVBO) and Commodities (COM), in the United States over the period April 1987 - December 2019. We compute the monthly excess returns that are earned by these classes, henceforth asset class returns. This results in $T = 392$ monthly time-series observations for the asset class returns. We use the S&P500 index to measure returns on equity, which we obtain from the Yahoo Finance database. The S&P500 Index tracks 500 leading companies in leading sectors that are publicly held on U.S. stock exchanges, and hence provides a diversified representation of the overall equity return level in the U.S. market. For the Government Bonds asset class we use the returns on the one-month Treasury bills, which we obtained from the Kenneth R. French Data Library[1]. Lastly, we measure the returns on the Commodities asset class by calculating monthly returns on the Producer Price Index by Commodity: All Commodities (PPIACO), which is obtained from the Federal Reserve Economic Data (FRED)[2]. The PPIACO index indicates the selling price levels received by domestic producers of commodities for their output, thereby representing the average price level of U.S. commodities. For the Equity and Commodities classes, we are interested in the excess returns. For these classes, we construct excess returns by subtracting the one-month U.S. short rate from the monthly returns.

Table 1: Descriptive Statistics of the Asset Classes Excess Returns

Asset Class	Mean	Median	Min	Max	Std Dev	Sharpe Ratio
Equity	0.45	0.91	-22.13	10.83	4.27	0.11
Government bonds	0.25	0.235	0.00	0.79	0.20	1.23
Commodities	-0.07	-0.11	-5.41	2.80	0.96	-0.07

NOTE: This table presents descriptive statistics for monthly percentage excess returns per asset class, calculated from the monthly percentage change over index values over the sample period April 1987 - December 2019.

Table 1 presents summary statistics of the excess returns on the asset classes. As

[1]https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
[2]https://fred.stlouisfed.org/series/PPIACO

expected, the Equity class has the highest mean returns, but these are paired with the highest standard deviation which results in a lower Sharpe Ratio than that of the Fixed Income class.

2.2 Predictor Variables

We aim to predict and explain the returns of the asset classes, described in section 2.1, by including a collection K = 19 predictor variables including macroeconomic indicators, treasury yield rates, a U.S. climate index and Fama-French (Fama and French (1996)) factors in the asset pricing models.

Inspired by Bianchi et al. (2017) and Gu et al. (2020), we have selected a set of eleven macroeconomic indicators. The first one is the U.S. Personal Consumption Expenditure (PCE) index, which tracks the average consumption patterns of U.S. households. Secondly, we use the U.S. Consumer Price Index (CPI), that measures the average price level of goods and services purchased by U.S. households. The third and the fourth indicator are the Industrial Production (IP) and Gross Domestic Product (GDP), which are different measures of U.S. economic output. Furthermore, we consider the Term Spread (TS) on bonds, which is constructed as the difference between the ten-year and three-month U.S. Treasury yields. The sixth macroeconomic indicator is the Default Risk Premium (DRP) on bonds, which is calculated as the difference between Moody's Baa corporate bond yield averages, and the ten-year U.S. Treasury yields. For the seventh indicator we use the CBOE S&P-100 Volatility Index (VXO), which measures the expected volatility on the S&P-100 index.

Furthermore, we use the S&P/Case-Shiller U.S. National Home Price Index (RE) as a proxy for U.S. residential real estate prices. Additionally, we include a U.S. Unemployment Rate (UR) index which represents the number of unemployed U.S. citizens as a percentage of the total labor force. We also include a Trade Weighted U.S. Dollar Index, which represents a weighted average of the FX value of the U.S. Dollar against a set of the other leading index currencies. Finally, we include the market value of the gross federal debt (GFD) which represents the debt burden faced by the U.S. government. We obtained all macroeconomic indicators from the FRED database.

Interest rates can have predictive value on economic activity in the U.S (Kozlowski (1995)). Hence, besides the macroeconomic indicators, we include the U.S. treasury rates of the 3-month, 10-year and 30-year maturities in our set of predictors. (Dell et al. (2014) show the current interest in literature on the effect of climate change on economic output. They highlight the direct effects and risks that climate change has on the economy. This interest is directly reflected by the increasing importance of risk associated with

Environmental, Social & Governance (ESG) to financial institutions (Kotsantonis et al. (2016)). Hence, in our research, we include the U.S. Climate Extremes Index (CEI), obtained from the National Centers for Environmental Information[3] to capture the effects of the observed change the U.S. climate on the excess returns of the assets. Finally, we include the well-known Market, Size and Value factors of the 3 Factor Fama-French model by Fama and French (1996) which are known for their ability to explain and predict excess returns on Equity.

3 Methodology

To predict the asset returns we examine multiple ML approaches, introduced in Section 3.1. These are supervised ML methods which are trained using the asset returns as dependent variable and the predictor variables as explanatory variables. In Section 3.2, we describe the expanding window framework used to estimate the ML models and their hyperparameters. We compare the return forecasts by adopting the comparison methods described in Section 3.3. Finally, we interpret the return forecasts using variable importance and partial dependence plots in Section 3.4.

3.1 Model Specifications

We define $\boldsymbol{r}_{1:T} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_T)$ to be the $(3 \times T)$ asset return data matrix, where $\mathbf{r}_t = (r_{1,t}, r_{2,t}, r_{3,t})'$ is the vector of excess returns on the asset classes at time t. Furthermore, we define $\boldsymbol{f}_{1:T} = (\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_T)$ to be the $(K \times T)$ matrix of predictor data, where $\mathbf{f}_t = (f_{1,t}, f_{2,t}, \dots, f_{K,t})'$ is the vector of predictors at time t.

Using asset pricing models, we aim to accurately predict the returns of asset class i on time $t+1$, given the information available at time t. Hence, following Gu et al. (2020), we model the excess returns $r_{i,t+1}$ as:

$$r_{i,t+1} = E_t[r_{i,t+1}] + \epsilon_{i,t+1}, \qquad E_t[r_{i,t+1}] = g_i(\mathbf{f}_t)$$

for time $t = 1, \dots, T$ and class $i = \{1, 2, 3\}$. $E_t[r_{i,t+1}]$ denotes the conditional expectation at time t of the returns of asset class i during time $t+1$ and $\epsilon_{i,t+1}$ the prediction error at time $t+1$. We express the conditional return expectation as a flexible function $g_i(\cdot)$ of the $(K \times 1)$ vector $\mathbf{f}_t$ of the macroeconomic predictor variables. We now specify different functional forms of $g_i(\cdot)$.

[3]https://www.ncdc.noaa.gov/extremes/cei/definition

3.1.1 Linear Models

We first set the baseline model to a simple linear regression representation:

$$g_i(\mathbf{f}_t) = \mathbf{f}_t'\boldsymbol{\beta}_i. \tag{1}$$

Equation 1 represents a linear combination of the macroeconomic indicator data with the $(K \times 1)$ parameter vector $\boldsymbol{\beta}_i$.

To prevent overfitting in the linear model due to too many risk factors, we examine regularization methods that can shrink certain predictor coefficients to zero. This is done by including a penalty term to the OLS loss function, as described in Equation 2. These methods include Ridge ($\alpha = 0$), Lasso ($\alpha = 1$) and Elastic Net (EN) for $(0 < \alpha < 1)$, where the EN penalty is a convex combination of Ridge and Lasso.

$$\boldsymbol{\beta}_i^* = \arg\min_{\boldsymbol{\beta}_i} \sum_{t=1}^{T} (r_{i,t} - \beta_{0,i} - \mathbf{f}_t'\boldsymbol{\beta}_i)^2 + \lambda \sum_{k=1}^{K} ((1-\alpha)\beta_{k,i}^2 + \alpha|\beta_{k,i}|) \tag{2}$$

The penalty term in the loss function forces certain predictor coefficients to zero, allowing the model to prevent overfitting in the data. The hyperparameter $\lambda > 0$ represents the penalization magnitude. Increasing λ will increase the penalty and thereby force more coefficients towards zero. Setting λ to zero results in the linear baseline model from Equation 1. We will optimize for α and λ using timeseries cross-validation, as explained in Section 3.2. This results into the following (shrunk) functional form of $g_i(\cdot)$:

$$g_i(\mathbf{f}_t) = \mathbf{f}_t'\boldsymbol{\beta}_i^*. \tag{3}$$

The models described in Equation 1 and 2 can be regarded as restrictive due to their linearity (Atsalakis and Valavanis (2009)). Hence, in addition to the (shrunk) linear models, we use Regression Trees (RT), Random Forests (RF), Boosted Regression Trees (BRT) in Section 3.1.2 and Neural Networks (NN) in Section 3.1.3 to capture potential nonlinear dynamics in the asset returns.

3.1.2 Tree-based Models

RTs (Breiman et al., 1984) are a fully non-parametric estimation technique that is used to group sample returns of the training set into nodes/branches, based on their similarity in predictor values. We estimate the RT by minimizing the following cost complexity function:

$$\sum_{m=1}^{|\mathcal{T}|} \sum_{t:f_t \in R_m} (r_t - \hat{r}_{R_m})^2 + |\mathcal{T}|. \tag{4}$$

In Equation 4, $\mathcal{T}$ represents the RT, $|\mathcal{T}|$ corresponds to the number of terminal nodes in tree $\mathcal{T}$ and R_m the region in the sample space of the macroeconomic indicators corresponding to the m^{th} terminal node. The variable $\hat{r}_{R_m}$ corresponds to the return prediction for returns in a region R_m, computed by the average of the sample points that lay in that region. The additive term $|\mathcal{T}|$ in the optimization function is used to prevent overfitting in the data. This happens when the leafs of the tree grow too "deep" into the data and out-of-sample forecasts will become too variable.

Forecasts for returns can be made by classifying a test sample point to a specific terminal node m of the regression tree and then, using the average of the sample points in that node denoted by c_m, to predict the return:

$$g_i(\mathbf{f}_t) = \sum_{m=1}^{|\mathcal{T}|} c_m I_{\{\mathbf{f}_t \in R_m\}}. \tag{5}$$

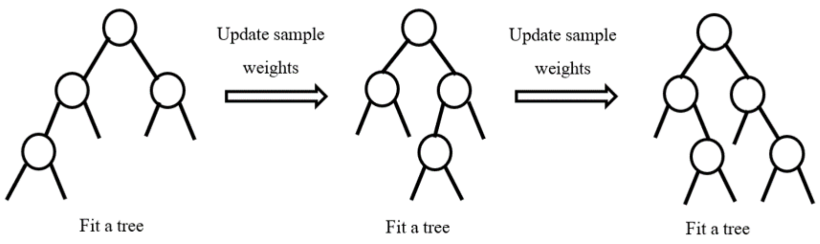
Ensemble methods, which combine the output of multiple models, can often improve forecasting precision (Dietterich (2000)). Multiple RTs can also be combined, giving rise to the use of RFs (Breiman (2001)). To estimate a RF, B RTs are trained individually using only random subsets of the predictor variables. These random subsets of the predictors decorrelate the individual trees, ensuring variance reduction in the return forecasts. If we denote the return forecast of an individual tree $b \in \{1, \ldots, B\}$ as $g_i^b(\mathbf{f}_t)$, a RF forecast can be constructed by averaging over the individual tree forecasts:

$$g_i(\mathbf{f}_t) = \frac{1}{B} \sum_{b=1}^{B} g_i^b(\mathbf{f}_t). \tag{6}$$

RTs can also be combined sequentially to boost their collective performance through weighted data sampling. When iteratively estimating new trees, more weight can be given to sample return forecasts that are inaccurate. In this way, a new tree can be estimated on an adjusted version of the training set, prioritizing on previous errors. Subsequently, errors of previous models are minimized while increasing (or "boosting") the performance of high-performing trees.

Friedman (2001) introduce Gradient Boosting Machines by employing gradient descent algorithms to minimize errors in sequential models, yielding a competitive and robust forecasting procedure. The Extreme Gradient Boosting (XGB) algorithm is introduced by Chen and Guestrin (2016), which is optimized for memory efficiency and speed, using parallel and distributed computing. We apply the XGB algorithm to compute return predictions by producing an average of the $g_i^b(\mathbf{f}_t)$ predictions made by the sequentially "boosted" regression trees as described in Equation 6. The collection of boosted RTs forms the BRT model. The process is visualized in Figure 1:

Figure 1: Regression Tree Boosting



NOTE: This figure illustrates the sequential boosting of regression trees, which is performed by the XGBoost algorithm. After fitting a new tree, more weight is given to previously inaccurately forecasted samples.

Because tree boosting algorithms produce trees to correct for the errors of previous trees on the training data, they are prone to overfitting quickly. A key hyperparameter in the XGB algorithm to control overfitting is the learning rate, denoted by η. The learning rate functions as a weighting factor for the corrections made in the new trees.

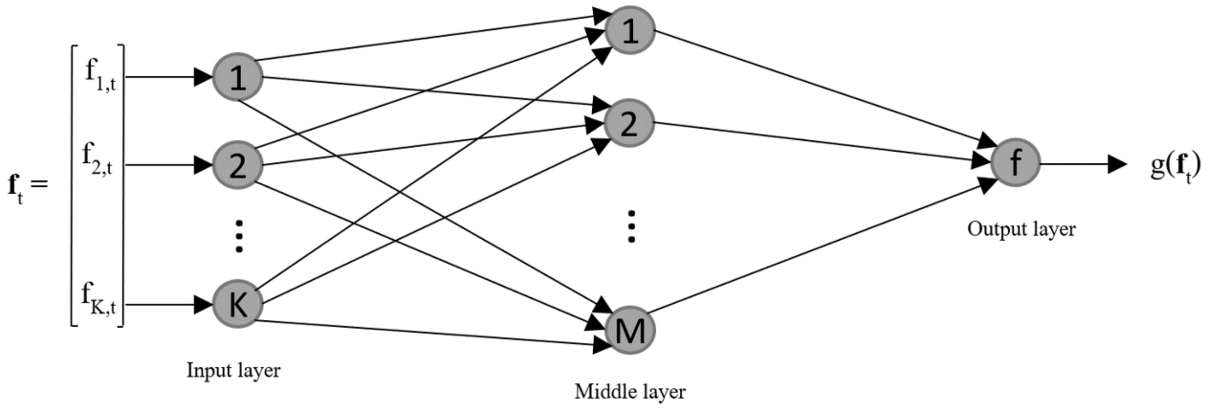
3.1.3 Neural Network Model

Finally, we use NNs to model potential non-linear risk-return specifications. NN models are a powerful ML tool due to their ability to approximately fit any functional form that is hidden in the data (Stinchcombe et al. (1989)). This flexibility is achieved from the network architecture of nodes that can pass signals through (multiple) hidden layers. This ML network is popularized through its biological interpretation, because it mimics the neurological connections found in biological brains.

NN models can provide an approximation of our return prediction function $g_i(\mathbf{f}_t)$, using a very flexible and layered structure. We illustrate the model by explaining the components of a Single Layer NN. This example is illustrated in Figure 2. The structure of such a network can be generalized to a NN that contains multiple middle layers.

The Single Layer NN consists of an input layer, a hidden layer, and an output layer. The three network layers each contain a specified number of nodes. In our context, the input layer contains K nodes $\{n_1^{(1)}, \dots, n_K^{(1)}\}$, the middle layer contains M nodes $\{n_1^{(2)}, \dots, n_M^{(2)}\}$, and the output layer contains 1 output node $\{n_1^{(3)}\}$. The superscripts refer to the input layer (1), middle layer (2) and the output layer (3).

Figure 2: Single Layer Neural Network



NOTE: This figure illustrates the Single Layer NN example. The input layer consists of K nodes, corresponding to the K predictor variables. The middle layer contains M nodes, and the output layer consists of one terminal node. Signals are passed and combined through the architecture of nodes to produce a return forecast in the output layer.

All nodes in the network receive an activation signal, which is then used to update the node value. The input nodes receive the prediction variable data of time t: $\{f_{1,t}, f_{2,t}, \dots, f_{K,t}\}$ as activation signals and these are set as their node value. These input node values are then send as signals to the M nodes of the hidden layer. Each node $n_m^{(2)}$ of the hidden layer receives an activation signal $a_{m,t}^{(2)}$ from the input layer which is a linear combination of the values of the input nodes:

$$a_{m,t}^{(2)} = \sum_{k=1}^{K} \beta_{m,k}^{(1)} f_{k,t}, \tag{7}$$

where $\beta_{m,k}^{(1)}$ is the scaling coefficient from the k^{th} input node value to the m^{th} node of the hidden layer. This node in the hidden layer processes the activation signal $a_{m,t}^{(2)}$ in a nonlinear activation function $h(a_{m,t}^{(2)})$, to compute the value of hidden layer node $n_m^{(2)}$. Subsequently, the values of the hidden nodes are sent to the output node as a linear combination, producing the activation signal $a_{1,t}^{(3)}$ for the output node $n_1^{(3)}$:

$$a_{1,t}^{(3)} = \sum_{m=1}^{M} \beta_m^{(2)} h(a_{m,t}^{(2)}). \tag{8}$$

The output node has a distinct output activation function $\sigma(\cdot)$. We use a linear output function to map the node signals back to the real number scale $\mathbb{R}$. Therefore, we choose function $\sigma(\cdot)$ as the identity function: $\sigma(a_{1,t}^{(3)}) = a_{1,t}^{(3)}$.

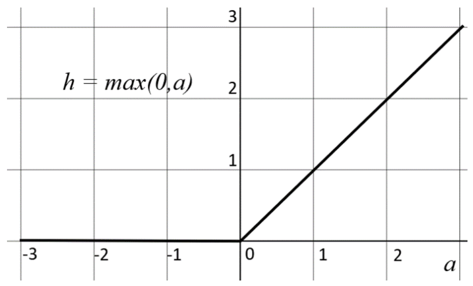
We can now express the NN return forecast as:

$$g_i(\mathbf{f}_t) = a_{1,t}^{(3)} = \sum_{m=1}^{M} \beta_m^{(2)} h(a_{m,t}^{(2)}) = \sum_{m=1}^{M} \beta_m^{(2)} h(\sum_{k=1}^{K} \beta_{m,k}^{(1)} f_{k,t}), \tag{9}$$

which we obtain by substituting Equation 7 and Equation 8 recursively in the identity output activation function $\sigma(\cdot)$. This simple example of 1 hidden layer can be extended to a NN of multiple hidden layers, which interact with subsequent streams of activation signals before reaching the output layer.

The key in this network structure is the recursive use of linear combinations and nonlinear transformations of the signals passing through the nodes. There are several activation functions, each influencing the model's performance. The activation function can be considered as a hyper parameter of the model. We use the popular Rectified Linear Unit (ReLU) $h_{ReLU}(a) = max(0, a)$ activation function, graphed in Figure 3. The ReLU activation function has a nonzero derivative which allows gradient based learning, resulting in quick convergence.

Figure 3: ReLU



NOTE: This figure graphs the Rectified Linear Unit activation function used for the node activation in the proposed NN.

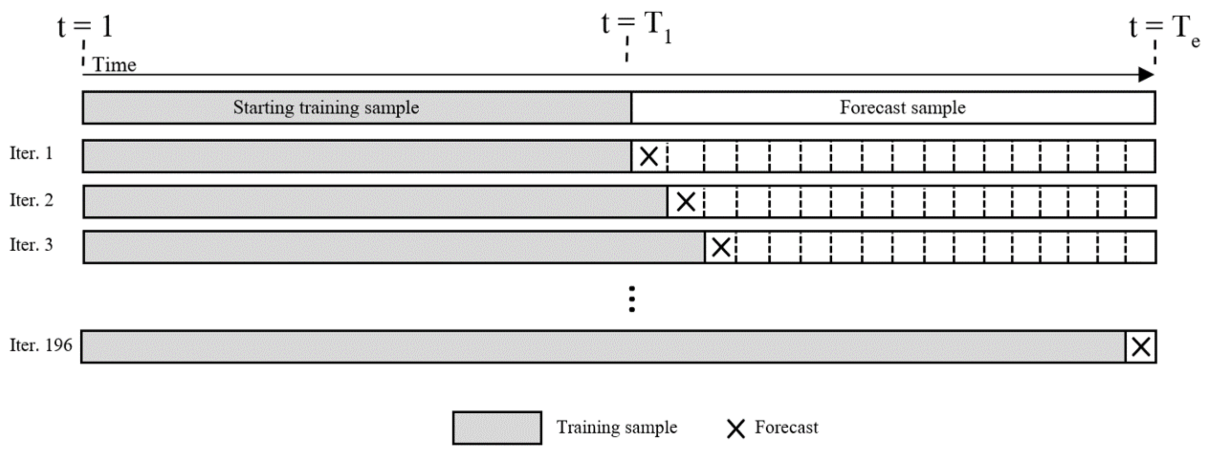
We compare the forecasting performance of the ML models with the performance of a naive asset pricing model. The naive model does not use the predictor variables and produces return forecasts by simply averaging over the return training sample.

3.2 Expanding Window Framework

We are interested in the performance of ML based asset pricing models. To measure the performance in an out-of-sample setting, we adopt an expanding window framework that iterates over our dataset. From this point, we differentiate between the end of our starting training sample and the end of the forecasting sample, which we denote by T_1 and T_e respectively. Ranging from 1987 until 2019, we have approximately 33 years of return and predictor data, consisting of 392 time series observations. From this follows that $T_e = 392$. Figure 4 visualizes the expanding window framework used to re-estimate the ML models and to produce one-month-ahead forecasts. The first 196 monthly observations of the sample are used as starting training sample, which includes observations from $t = 1$ to $t = T_1 - 1$, so $T_1 = 196$. The forecast sample ranges from

$t = T_1$ to $t = T_e$. In each iteration, the training set is expanded with the next observation from the forecast sample and then used to refit the ML models. In this way, we simulate a progression through time and the real-time monthly arrival of new data points. The updated models are used to produce the next one-month-ahead forecasts, until the end of the forecast period is reached. This results in $T_e - T_1 = 196$ expanding window iterations. As shown in Section 4.1, the optimal hyperparameter specifications can show variation while progressing through the expanding window. On the other hand, cross-validating all models in each iteration is computationally infeasible. Hence, we re-select the optimal hyperparameters for the models every 12 months. However, it is important to note that the models are still retrained every month with the new data points that enter through the expanding window.

Figure 4: Expanding Window Framework

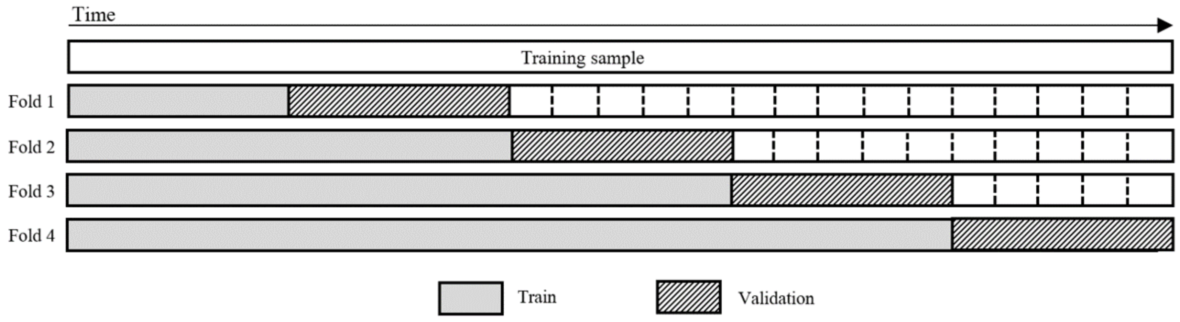


NOTE: This figure displays the expanding window structure for estimating the models and producing the forecasts. We set $T_1 = 196$ and $T_e = 392$ as the bounds of the forecasting sample.

The starting training sample is used to tune the hyperparameters of the ML specifications. Because we use timeseries data, the sample points are not independent and thus we cannot use regular k-fold cross validation. Intertemporal dependence structures like autocovariance or seasonality would make random splits invalid. The situation would arise that data from the $(i+1)^{th}$ fold would be used to forecasts of the i^{ht} fold. We adopt an adjusted cross-validation scheme that is suited for timeseries analysis, illustrated by Figure 5. The training sample is split into 5 folds. Again, an expanding window framework is used to construct subsequent training and validation samples. In the training samples, the ML models are estimated with different hyperparameter configurations and tested in the validation samples. This results in an expanding $5 - 1 = 4$ -fold cross validation scheme. We select the hyperparameter specifications that on average achieve the lowest Mean Squared Prediction Errors (MSPE) on the validation sets.

To create a realistic out-of-sample setting while forecasting, *data snooping* must be

Figure 5: Timeseries Cross-Validation



NOTE: This figure displays the timeseries cross-validation scheme that is used for the hyperparameter tuning of the ML algorithms in the first iteration of the expanding window framework.

avoided. For a forecast made at time t, only information that would be available prior to and at time t may be used. The majority of the predictor variables exhibit a publication lag, meaning that their realizations are published later than the time that they correspond to. We account for publication lags by lagging the predictor variables appropriately. We refer to Table 6 in Appendix A for the lag orders that are used.

3.3 Forecasting Performance Measures

We are interested in the predictive performance of the different asset pricing models on the test data set. As displayed in Table 1, the returns of the asset classes differ in scale and standard deviation. Hence, to compare the performance of the models across the different asset classes, we adopt the scale invariant Mean Absolute Scaled Error (MASE) measure proposed by Hyndman and Koehler (2006). MASE is constructed by scaling the forecast errors based on the in-sample Mean Absolute Error from the random walk forecast method. For a one-month-ahead return forecast $\hat{r}_{T+1}$, produced using a training sample $t = 1, \ldots, T$, the scaled forecast error is constructed as:

$$q_{T+1} = \frac{r_{T+1} - \hat{r}_{T+1}}{\frac{1}{T-1}\sum_{t=2}^{T} |r_t - r_{t-1}|}. \tag{10}$$

Then, we construct the MASE by averaging the scaled forecast errors of the forecast period $\{T_1, \ldots, T_e\}$:

$$\text{MASE} = \frac{1}{T_e - T_1} \sum_{t=T_1}^{T_e} q_t. \tag{11}$$

The MASE measure allows us to compare the performance of the ML models across the different asset classes. A MASE score smaller than one indicates that the model produces, on average, more accurate forecasts than a naive (random walk) forecast.

Additionally, we examine the Out-of-Sample R^2_{OOS} measure. For a return forecast series $\hat{r}_{t+1}$, with $t = T_1, \ldots, T_e$ we compute the R^2_{OOS} measure as

$$R^2_{OOS} = 1 - \frac{\sum_{t=T_1}^{T_e} (r_{t+1} - \hat{r}_{t+1})^2}{\sum_{t=T_1}^{T_e} (r_{t+1})^2} \tag{12}$$

The R^2_{OOS} measure indicates how much of the variance in the returns is explained by the return forecasts. Negative R^2_{OOS} values signal that a horizontal line would better explain the out-of-sample variation than the corresponding model. A R^2_{OOS} measure of one occurs when all forecast errors are zero and thus the corresponding model describes all out-of-sample variation in the returns.

To formally test for a difference in predictive accuracy, we use the Diebold-Mariano (DM) Test (Diebold and Mariano (2002)). We evaluate the test statistic for the return forecasts over the forecast sample of our dataset, consisting of $T_e - T_1 = 196$ data points. Using the DM-test we compare the return forecasts $\hat{r}^{(1)}_{i,t}$ and $\hat{r}^{(2)}_{i,t}$, for $t = T_1, \ldots, T_e$, resulting from ML model (1) and (2) for asset class i. The residuals of the forecasts can be written as:

$$e^{(1)}_{i,t} = r_{i,t} - \hat{r}^{(1)}_{i,t} \quad \text{and} \quad e^{(2)}_{i,t} = r_{i,t} - \hat{r}^{(2)}_{i,t} \qquad \text{for} \qquad t = T_1, \ldots, T_e.$$

From the forecast residuals the loss differentials can be constructed as:

$$d_{i,t} = \left(e^{(1)}_{i,t}\right)^2 - \left(e^{(2)}_{i,t}\right)^2 \qquad \text{for} \qquad t = T_1, \ldots, T_e,$$

which are related to the Mean Squared Prediction Error (MSPE) statistic. We are interested in the expectation of this loss differential, defined as $\mu = E(d_{i,t})$. This μ can be approximated by the sample mean. We define the sample mean loss differential $\bar{d}_i$, and the k'th order autocorrelation $\gamma_{i,k}$ as:

$$\bar{d}_i = \frac{1}{T_e - T_1} \sum_{t=T_1}^{T_e} d_{i,t} \qquad \text{and} \qquad \gamma_{i,k} = \frac{1}{T_e - T_1} \sum_{t=T_1}^{T_e} (d_{i,t} - \bar{d}_i)(d_{i,t-k} - \bar{d}_i) \qquad \text{for} \qquad k \leq 1.$$

Then, for $h > 0$, where $h = (T_e - T_1)^{\frac{1}{3}} + 1$ should be sufficient, because we assume the auto correlations at lags of order $(T_e - T_1)^{\frac{1}{3}} + 1$ and higher to be close to zero and insignificant. We define the DM test statistic as:

$$DM = \frac{\bar{d}_i}{\sqrt{\frac{\gamma_{0,i} + 2\sum_{k=1}^{h-1} \gamma_{k,i}}{n}}}. \tag{13}$$

Essentially, the DM test is a regular t-test where we test for the equality in means for two series of prediction errors. We use the statistic to test the H_0 hypothesis of $\mu = 0$ vs the alternative hypothesis $H_a : \mu \neq 0$. Under the null hypothesis the DM statistic has a Standard Normal distribution. Hence, we reject H_0 when $|DM| > z_{z-\alpha/2}$ where $z_{z-\alpha/2}$ is the two-sided critical value of the standard normal distribution for a $\alpha\%$ significance level.

3.4 Return Forecast Interpretation

In this subsection we present the outline of the interpretation techniques we use to better understand the forecasts produced by the models presented in Section 3.1. Doshi-Velez and Kim (2017) define interpretability in the context of ML as the ability to explain or to present in understandable terms to a human. The models that we examine vary in interpretability, for example, the (regularized) linear model specifications and the regression trees are visually comprehensible. For the linear models, one can inspect the sign, magnitude and significance of the estimated regression coefficients to understand the effect of the macroeconomic indicators on the return forecasts of the data. The regression tree model estimates can be visualized by a graphical representation, which displays the path splits towards the leaf nodes of the tree. On the other hand, RF, BRT and NN models are less transparent. Their high flexibility comes at the price of lower transparency and interpretability. We refer to the last three models as the non-transparent models.

While interpreting the ML model forecasts, we aim to answer two sub-questions:

1. Which predictor variables contribute to the return forecasts most and do these differ per model?
2. What are the partial relations between predictor variables and return forecasts?

To answer these two sub-questions we use the SHAP method introduced by Lundberg and Lee (2017) to construct variable importance measures and plot partial dependence plots.

SHAP is based on Shapley Values, which are introduced by Shapley (1953). Shapley values are used in coalitional game theory and tell how in a game the payout should be distributed fairly across the players. In SHAP, Lundberg and Lee (2017) apply this idea to ML explainability. Shapley quantifies the contribution of each player to the payout, while SHAP quantifies feature variable contributions to predictions produced by a model. These contributions can be interpreted as importance values of the predictor variables in producing the individual return forecasts in the test sample. This makes SHAP a local interpretation method, providing explanations of specific forecasts produced by our models. Its ability to explain specific return forecasts instances makes SHAP an attractive method to use in practise. Furthermore, SHAP is the only method proven to be efficient, meaning that the sum of all Shapley values of a return forecast add up to the forecast itself (Lundberg and Lee (2017)). Among other properties, SHAP is also proven to be symmetric, meaning that two features will get the same Shapley values if their marginal importance to the return forecast is equal.

For a return forecast $g(\mathbf{f}_t)$, produced by a model at time t in our test sample, SHAP specifies the output explanation as an additive model:

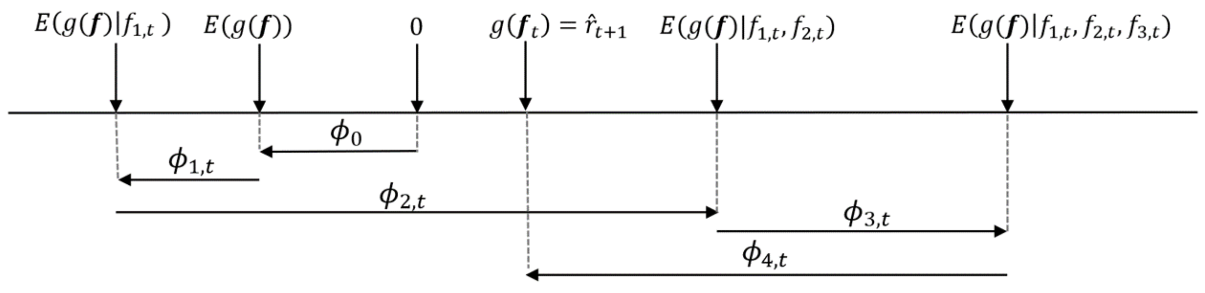
$$g(\mathbf{f}_t) = \phi_0 + \sum_{k=1}^{K} \phi_{k,t}, \tag{14}$$

which we can refer to as the "explanation model". In the explanation model, the Shapley Value $\phi_{k,t}$ is the feature attribution of feature k to the return forecast at time t. The value ϕ_0 corresponds to the expected return forecast over all test sample return predictions, $E(g(\mathbf{f}))$. Hence, we can rewrite Equation 14 and express the summation over the Shapley values $\phi_{k,t}$ as the deviation of a specific return forecast from the mean forecast:

$$\sum_{k=1}^{K} \phi_{k,t} = g(\mathbf{f}_t) - E(g(\mathbf{f})). \tag{15}$$

When for a return forecast $g(\mathbf{f}_t)$ the Shapley values $\phi_{1,t}, \ldots \phi_{K,t}$ are estimated, we can use the additive model from Equation 14 to explain how much the value of each predictor variable has contributed to the return forecast $g(\mathbf{f}_t)$. We illustrate this using an example with four predictor variables to compute a forecast in Figure 6.

Figure 6: SHAP illustration



NOTE: This figure displays the contribution of individual predictor variables to the final forecast output $\hat{r}_{t+1}$. This example uses K = 4 predictor variables to compute the forecast.

The Shapley values are computed by simulating cases where only subsets of all predictors are "present". We define $\mathcal{F}$ as the set containing all predictor variables. A Shapley value $\phi_{k,t}$ is estimated by averaging forecast differences resulting from in- and excluding the k^{th} predictor to all subsets $\mathcal{S}$ of $\mathcal{F}$. Thus, for each subset $\mathcal{S}$, we estimate the model using predictor variable sets $\mathcal{S} \cup \{k\}$ and $\mathcal{S}$, and denote these models by $g_{\mathcal{S}\cup\{k\}}(\cdot)$ and $g_{\mathcal{S}}(\cdot)$ respectively. A difference term $(g_{\mathcal{S}\cup\{k\}}(\mathbf{f}_{\mathcal{S}\cup\{k\},t}) - g_{\mathcal{S}}(\mathbf{f}_{\mathcal{S},t}))$ can be constructed, which denotes the marginal contribution of predictor variable k to the forecasts. Because the effect of a predictor variable k depends on the other predictor variables, we compute the difference term for all possible subsets $\mathcal{S}$. To compute a weighted average, the terms are scaled by the probability of the ordered combination of features derived from set $\mathcal{S}$, resulting in a weighted average expression:

$$\phi_{k,t} = \sum_{\mathcal{S}\subseteq\mathcal{F}\setminus\{k\}} \frac{|\mathcal{S}|!(|\mathcal{F}| - |\mathcal{S}| - 1)!}{|\mathcal{F}|!}(g_{\mathcal{S}\cup\{k\}}(\mathbf{f}_{\mathcal{S}\cup\{k\},t}) - g_{\mathcal{S}}(\mathbf{f}_{\mathcal{S},t})). \tag{16}$$

Estimating models on all possible subsets S would require estimating $2^{|\mathcal{F}|}$ models, which is practically infeasible. Lundberg and Lee (2017) propose efficient model-specific approximation methods for our linear models (linear SHAP), tree based methods (Tree SHAP) and NN (Deep SHAP). By restricting on specific model types, they develop faster model-specific approximations, based on sampling. These approximation techniques are implemented in the SHAP[4] package and used in this analysis.

Note that the Shapley values are estimated for all predictor variables, and for all instances in the forecasting sample. This results in K sets of $(T_e - T_1)$ Shapley values. To measure the overall importance of predictor variables in the production of the forecasts, we average over the Shapley values of the individual return forecasts. The average Shapley values from this analysis help us answer sub-question (1) of our interpretation research, because they measure the contributions of the different predictor variables to return forecasts. Additionally, we are interested in the partial relations between the predictor variables and the return forecasts produced by the models. For this reason, we produce scatter plots for the Shapley values per predictor variable. This results in partial dependence plots between the indicator values on the x-axis and the effect on the return prediction on the y-axis, which helps us answering sub-question (2) of our interpretation research.

4 Results

In this section, we forecast the asset returns introduced in Section 2.1 using the predictor variables of Section 2.2 in multiple ML models using the expanding window framework. Section 4.1 presents the hyperparameter configurations that are used to train the ML models. In Section 4.2 we present and discuss the forecasting performance of the models.

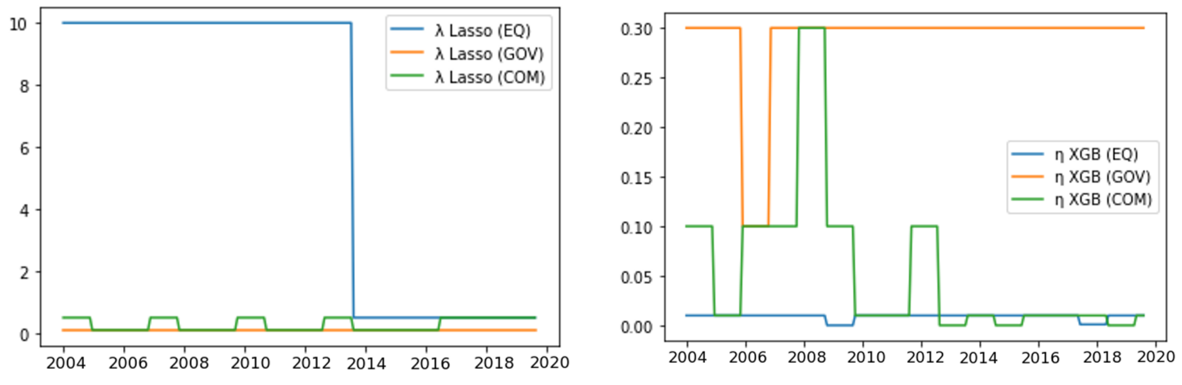
[4]https://github.com/slundberg/shap

The forecasting performance of the models is compared between crisis and non-crisis periods in Section 4.3. We interpret the forecasts produced by the models in Section 4.4 using the SHAP methodology.

4.1 Hyperparameter Configurations

The models as described in Section 3.1 are estimated for the asset classes using the expanding window framework as described in Section 3.2. We find that the optimal hyperparameter configurations differ per asset class and also vary over time. We illustrate this with timeseries plots of the λ-penalty parameter used in the Lasso models and the learning rate parameter η in the XGB algorithm in Figure 7 for the estimation of the BRT model. We refer to Appendix C for the other hyperparameter timeseries plots.

Figure 7: Time-variation in Hyperparameters



(a) λ-penalty in Lasso (b) η in XGBoost

NOTE: This figure shows how optimal hyperparameter values λ (penalty parameter of Lasso) and η (learning rate parameter of XGBoost) vary over the 196 iterations of the expanding window framework through the test period. The hyperparameters are displayed for the Equity (EQ), Government Bonds (GOV) and Commodities (COM) asset classes.

To summarize the hyperparameter configurations of the models for the different asset classes, we present the mode values of the selected hyperparameters in Table 2. We use the mode statistics to maintain nominal hyperparameter values. The mode values for the λ parameter in the Lasso model and the η from the XGB algorithm that are reported in Table 2 can be directly linked to the timeseries plots. For the Ridge and Lasso regression, the α parameter is fixed at 1 and 0 respectively, in line with Equation 2. For all asset classes, the Ridge model acquires high penalty values λ, compared to the Lasso and Elastic Net. When observing the sizes of the NN layers, lower # trees and lower max_depth hyperparameters of the commodities class, the ML models appear to prefer shallower learning for the Commodities class compared to the other classes. This could indicate that the models are prone to overfitting in the Commodities return data and prevent this by selecting shallower settings in the cross-validation steps. For the Equity and Government Bonds class, the mixing parameter α is smaller than 0.5, indicating a model preference for the Ridge penalty term in the optimization function.

Table 2: Hyperparameter Settings Equity Class

Model	α	λ	max_depth	# trees	max_features	η	layers
Equity							
Linear Regression	×	×	×	×	×	×	×
Ridge	1	10	×	×	×	×	×
Lasso	0	10	×	×	×	×	×
ElasticNet	0.3	1	×	×	×	×	×
Regression Tree	×	×	6	1	×	×	×
Random Forest	×	×	6	12	8	×	×
Boosted RT	×	×	4	10	×	0.01	×
Neural Network	×	×	×	×	×	×	$(200, 200)$
Government Bonds							
Linear Regression	×	×	×	×	×	×	×
Ridge	1	1	×	×	×	×	×
Lasso	0	0.1	×	×	×	×	×
ElasticNet	0.1	0.1	×	×	×	×	×
Regression Tree	×	×	8	1	×	×	×
Random Forest	×	×	8	10	10	×	×
Boosted RT	×	×	4	30	×	0.3	×
Neural Network	×	×	×	×	×	×	$(200, 200)$
Commodities							
Linear Regression	×	×	×	×	×	×	×
Ridge	1	10	×	×	×	×	×
Lasso	0	0.1	×	×	×	×	×
ElasticNet	0.2	0.5	×	×	×	×	×
Regression Tree	×	×	6	1	×	×	×
Random Forest	×	×	10	8	6	×	×
Boosted RT	×	×	4	10	×	0.01	×
Neural Network	×	×	×	×	×	×	$(20, \dots, 20)_5$

NOTE: This table presents the mode values of hyperparameter settings used to train the ML models. We use the mode to maintain nominal values of the hyperparameters, which are often defined as integers. In the expanding window the hyperparameter settings are re-chosen every 12 months through (k-1)-fold timeseries cross validation with k = 5.

4.2 Forecasting Performance

Using the hyperparameter configurations presented in Table 2, we iteratively re-estimate the models in our expanding window framework. While iterating through the forecasting sample we compute the one-step-ahead forecasts of the asset returns for the next months. Using the realized returns, we construct the forecast errors and use these to compute the MASE values in Table 3.

Table 3: Mean Absolute Scaled Error (MASE)

	Equity	Government Bonds	Commodities
Transparent models			
Naive Average Model	0.637	7.118	1.227
Linear regression (baseline)	0.780	0.566	1.398
Ridge	0.671	0.579	1.268
Lasso	0.631	3.906	1.231
ElasticNet	0.633	0.745	1.257
Regression Tree	0.814	0.478	1.724
Non-transparent models			
Random Forest	0.672	0.519	1.386
Boosted RT	0.631	0.494	1.229
Neural Network	0.897	0.663	1.860

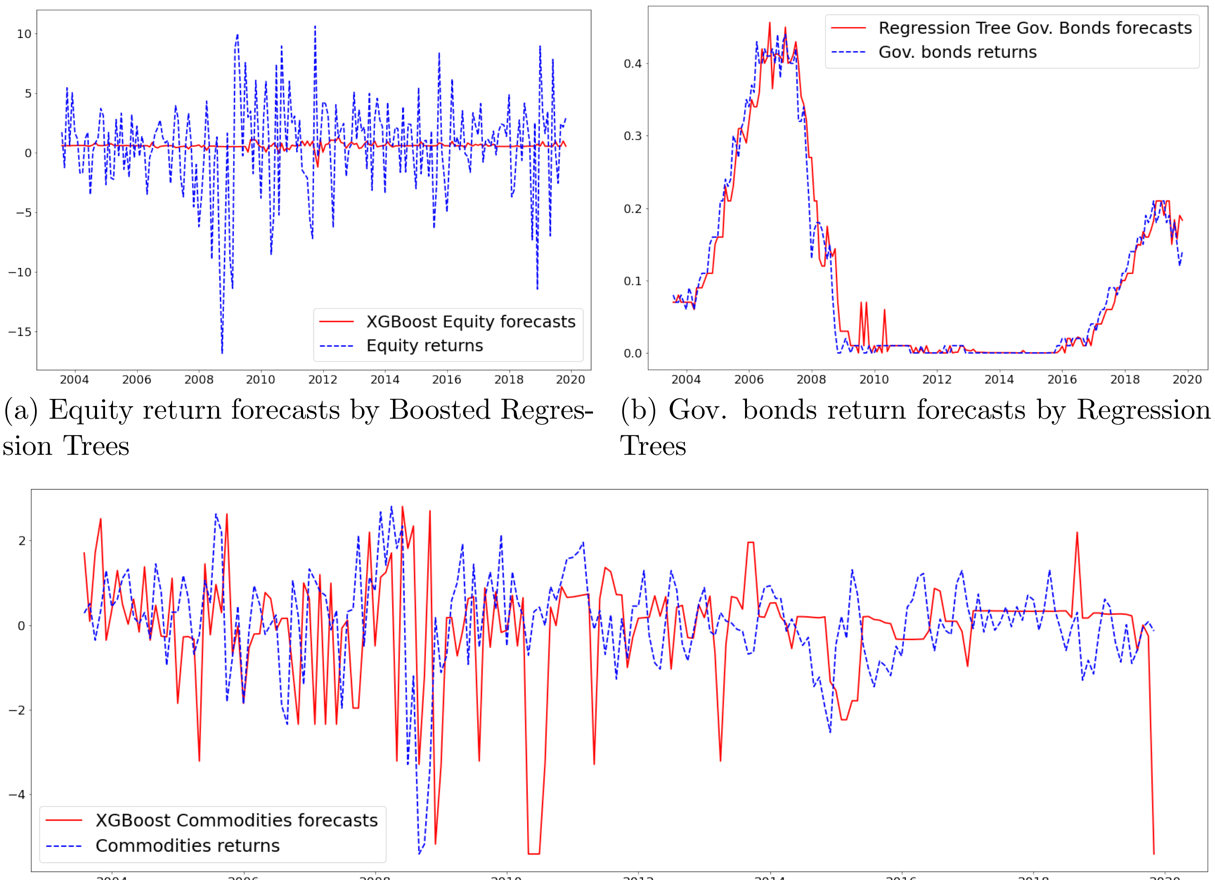
NOTE: This table presents the MASE values of the return forecasts produced by the different ML specifications over the test sample January 2003 - December 2019.

We compare naive model forecasts with the forecasts produced by the ML models. The naive forecasts are constructed by predicting the average returns value of the current training sample. For the Equity and Commodity returns, not all ML models outperform the naive model in terms of MASE. For the Government Bonds class, the forecasts produced by all ML models score lower MASE values than the naive model. As explained in Section 3.3, MASE values lower than one indicate that the model produces, on average, more accurate forecasts than the naive on-step-ahead random walk forecast, which is achieved by all ML models of the Equity class. For Equity, the ElasticNet and BRT models reach the lowest MASE values. For the Government Bonds class, the NN and Lasso are outperformed by a naive random walk model. The tree (ensemble) methods perform best for the Government Bonds class, where the BRT and RT models produce the lowest MASE values. For the Commodities class, all models produce worse forecasts than the naive one step ahead random walk model. The regularized linear models result in the lowers MASE values of the ML methods for the Commodities class.

For all asset classes, we graph the return forecasts with the lowest MASE in Figure 8. Complemented by the descriptive statistics Table 1 of the asset returns, we see that the

Equity returns are most volatile followed by the Commodity returns. In comparison to the Equity and Commodity classes, the Government bonds class returns are more persistent and also more accurately predicted. For the Equity return class, the randomness in the returns seems to force the BRT model forecasts to zero. This results in conservative forecasts that approximate a constant forecast series to minimize prediction error. The commodity return forecasts occasionally follow the trend of the real returns, but this is frequently accompanied by a lag. This can be observed around the Credit Crisis around 2008.

Figure 8: Out-of-sample Return Forecasts



(a) Equity return forecasts by Boosted Regression Trees (b) Gov. bonds return forecasts by Regression Trees

(c) Commodity return forecasts by Boosted Regression Trees

NOTE: This figure presents the out-of-sample return forecasts and realizations of the three asset classes. We select the best performing model in terms of MASE which are the BRT, RT and BRT models for the Equity, Government bonds and Commodity classes respectively. The plots range from January 2003 - December 2019 including 196 observations.

The R^2_{OOS} values per model for the different asset classes are presented in Table 4. These are generally in line with the MASE results. For the Equity and Commodities class, we generally find negative R^2_{OOS} values, indicating that a small part of the variance in the out-of-sample returns is explained by the predictor variables through the ML models. The negative R^2_{OOS} values indicate that the corresponding models do not follow the trend of the returns, and thereby fit worse than a constant model. For the Equity asset class, the Lasso and BRT models perform best in terms of MASE and in R^2_{OOS}. For the Commodities class, the Naive model and Lasso also achieve the highest R^2_{OOS}. Within the Government Bonds class, the BRT has the highest R^2_{OOS} value of 0.614.

Table 4: Out-Of-Sample R^2

	Equity	Government bonds	Commodities
Transparent models			
Naive Average Model	0.014	-1.504	-0.047
Linear Regression (baseline)	-0.386	0.612	-0.411
Ridge	-0.138	0.613	-0.239
Lasso	0.004	-0.043	-0.079
ElasticNet	-0.002	0.602	-0.157
Regression Tree	-0.998	0.613	-1.248
Non-transparent models			
Random Forest	-0.153	0.610	-0.347
Boosted RT	0.010	0.614	-0.261
Neural Network	-0.927	0.605	-1.442

NOTE: This table presents the R^2_{OOS} of the return forecasts produced by the different ML specifications over the test sample January 2003 - December 2019.

Table 5 shows the results of the DM-test. It tests for significant differences between the forecasting performance of the models. For the Equity class we find that the naive average forecast model significantly outperforms the linear model, RF and NN. The linear model is outperformed by all linear models and the BRT model. The NN has produced forecasts that are significantly less accurate than the forecasts of all other models, except the RT. The regularization methods Lasso, Ridge and Elastic Net improve return forecast accuracy significantly for the Equity class.

For the Government Bonds return class, we see mixed DM-test results between the models. The Naive model is outperformed by all ML models. This is in line with the MASE scores from Table 3. For this class, the return forecasts produced by the Lasso regression are significantly outperformed by all other models. The NN produces significantly lower forecast errors than the Naive and the Lasso model.

For the Commodities class, the naive model produces more accurate forecasts than all ML models. According to the DM test, these differences are significant for the RT, RF and NN model. The RT model is significantly outperformed by all models except the NN.

Combining our results from Tables 3, 4 and 5 we see that the relative accuracy performance differs per asset class. The variation in the Government Bonds class returns

Table 5: Diebold-Mariano Test Statistics

Equity Returns

	Linear	Ridge	Lasso	EN	RT	RF	BRT	NN
Naive	3.231*	1.679	0.981	1.315	1.832	2.063*	0.281	4.378*
Linear	0.000	-5.588*	-3.233*	-3.418*	1.250	-1.908	-3.084*	4.145*
Ridge		0.000	-1.596	-1.687	1.746	0.187	-1.550	5.859*
Lasso			0.000	0.628	1.819	2.006*	-0.353	4.391*
EN				0.000	1.818	2.010*	-0.581	4.637*
RT					0.000	-1.743	-1.834	-0.158
RF						0.000	-2.044*	4.409*
BRT							0.000	4.292*

Government Bond Returns

	Linear	Ridge	Lasso	EN	RT	RF	BRT	NN
Naive	-6.802*	-6.803*	-6.960*	-6.808*	-6.769*	-6.762	-6.789*	-6.784*
Linear	0.000	-0.367	5.947*	1.733	-0.183	0.356	-0.388	1.124
Ridge		0.000	5.948*	2.036*	-0.079	0.512	-0.278	1.417
Lasso			0.000	-5.975*	-5.875*	-5.870*	-5.931*	-5.936*
EN				0.000	-1.497	-1.459	-2.205*	-0.465
RT					0.000	0.895	-0.208	1.172
RF						0.000	-1.586	0.827
BRT							0.000	1.439

Commodity Returns

	Linear	Ridge	Lasso	EN	RT	RF	BRT	NN
Naive	1.601	0.967	0.683	1.039	3.405*	2.083*	1.210	2.891*
Linear	0.000	-4.069*	-1.790	-1.893	2.615*	-0.564	-1.468	3.389*
Ridge		0.000	-1.018	-0.797	3.434*	1.284	0.328	3.734*
Lasso			0.000	1.218	3.501*	2.597*	1.379	3.067*
EN				0.000	3.497*	2.461*	1.279	3.260*
RT					0.000	-2.880*	-3.054*	0.461
RF						0.000	-0.941	2.814*
BRT							0.000	3.187*

NOTE: This table contains the DM test statistics of the pairwise forecast comparisons between all models, for all three asset classes. A positive number indicates that the model in the row produces more accurate forecasts than the model in the column. DM statistics that are significant at a 5% confidence level the number are marked with an asterisk (*). The DM statistics are computed using the forecasts constructed on the test sample January 2003 - December 2019.

seems to be explained best by the models according to the R^2_{OOS} measure. A potential explanation could be that returns on Government Bonds exhibit less randomness and are potentially subject to less external influences. For the Equity and Commodity return classes we observe similar accuracy performances, where the linear model is outperformed by most ML methods.

4.3 Crisis vs. Non-Crisis Robustness Analysis

The test sample contains data points sampled from the Global Financial Crisis during the period of 2007 - 2008. We compare the performance of the models during crisis and the non-crisis periods. We define the data samples from 2007 and 2008 as the crisis period, resulting in 24 crisis observations. The remaining 172 observations are selected as non-crisis observations. We display the MASE and R^2_{OOS} measures for both periods in Appendix B. For all asset classes, we observe higher MASE values for the 2007-2008 period. This shows that the models produce better forecasts during non-crisis periods. Furthermore, the model-ranking based on MASE values does not show much variation between the crisis and non-crisis periods. The relative forecasting performance can thus be considered robust to being in crisis or non-crisis periods.

Like the MASE values, the R^2_{OOS} values differ between the crisis and non-crisis period. For the Government Bonds class, we see a strong increase in R^2_{OOS}, while for the Equity and Commodities classes we observe small declines. For the Government Bonds class, the increase could partially be explained by examining the returns and forecasts produced by the RT in Figure 8. By visual inspection, the dynamics of the forecasts of the Government Bonds better follow the real returns during the non-crisis period. We cannot directly observe this for the Equity and Commodities classes.

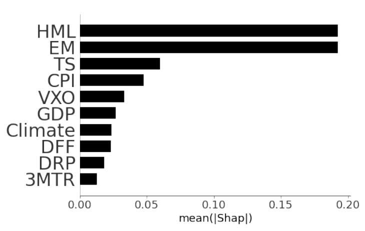
4.4 Model Interpretation

We have analyzed the forecast accuracy of the models for the different asset classes. In order to increase transparency in the forecasting of the asset returns, we use the SHAP method described in Section 3.4. We compute the additive contributions of the predictor variables to the return forecasts of the asset classes. As described in Section 3.4, we estimate the models with and without each predictor variable and thereby measure the contribution of each of the predictors to the return forecasts.

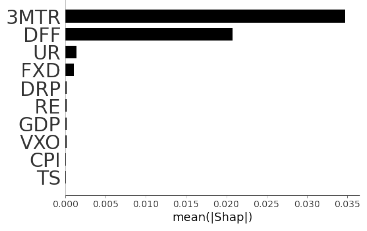
To increase the transparency of the models, we are interested in which predictor variables contribute most to the return forecasts produced by the models. For each asset class and for each model, we report the average Shapley value over the data points in the test sample. We report bar plots of the average feature importance value in Figure 9 for the RF, BRT and NN models. These models are considered as least transparent because we cannot visually inspect the model results like we can for a linear regression or RT models.

Figure 9 shows that for all three asset classes, the RF and the BRT algorithm use

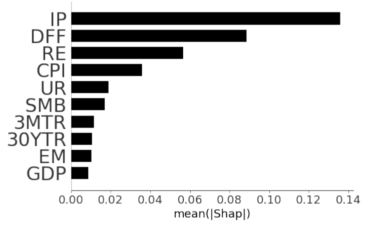
Figure 9: Feature Importance Barplots



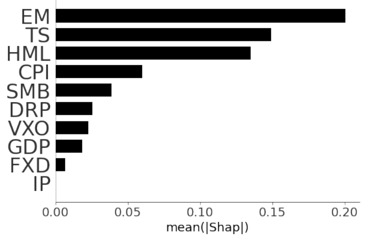
(a) RF - EQ



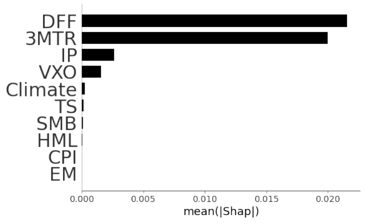
(b) RF - GOVBO



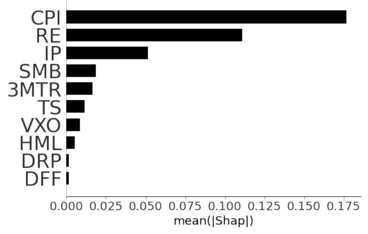
(c) RF - COM



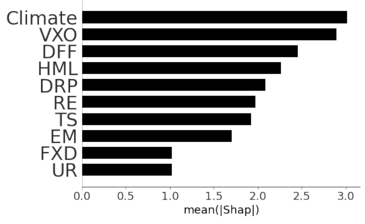
(d) BRT - EQ



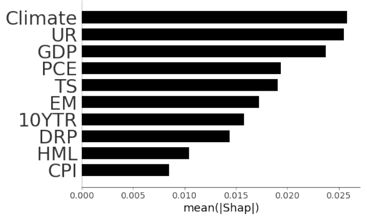
(e) BRT - GOVBO



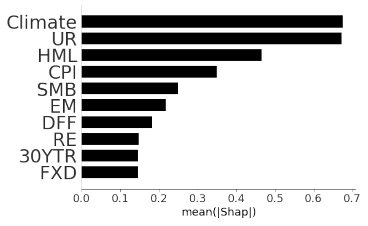
(f) BRT - COM



(g) NN - EQ



(h) NN - GOVBO



(i) NN - COM

NOTE: This figure displays Feature Importance Barplots for the Random Forest, Boosted Regression Tree and Neural Network models. They are constructed by averaging over the absolute Shapley values.

similar most influential predictor variables. These are the Fama-French factors: Excess Market return (EM) and the High-Minus-Low (HML) factor for the Equity class returns, according to the RF and BRT models. Fama and French (1996) have shown that these factors can price excess returns on equity fairly well, and our results appear to be in line with that finding. For the Government Bonds returns, the 3-Month Treasury Rate (3MTR) and the Domestic Federal Funds Rate (DFF) are the most important predictor variables, which also logically connect to Government Bonds returns. For the Commodities index, we see that Industrial Production (IP) and Real Estate (RE) are important predictor variables according to the RF and BRT models. Intuitively, there should be a connection between movements in industrial production levels and demands for commodities. The same reasoning holds for real estate price movements, as the prices of real estate will partially depend on the prices of construction materials. For the NN models, the importance of the features is spread more evenly over the predictor variables. The Climate factor appears to be the most important factor for all three asset classes according to the NN. The following important features are VXO for Equity, UR for Government Bonds and

UR for the Commodity class. Due to their extreme flexibility, NN models can capture any hidden patterns found in data. The feature importance levels of the NN models are less intuitive and not directly in line with macroeconomic theory. VXO, however, is also among the more important predictors in the RF and BRT model for the Equity class.

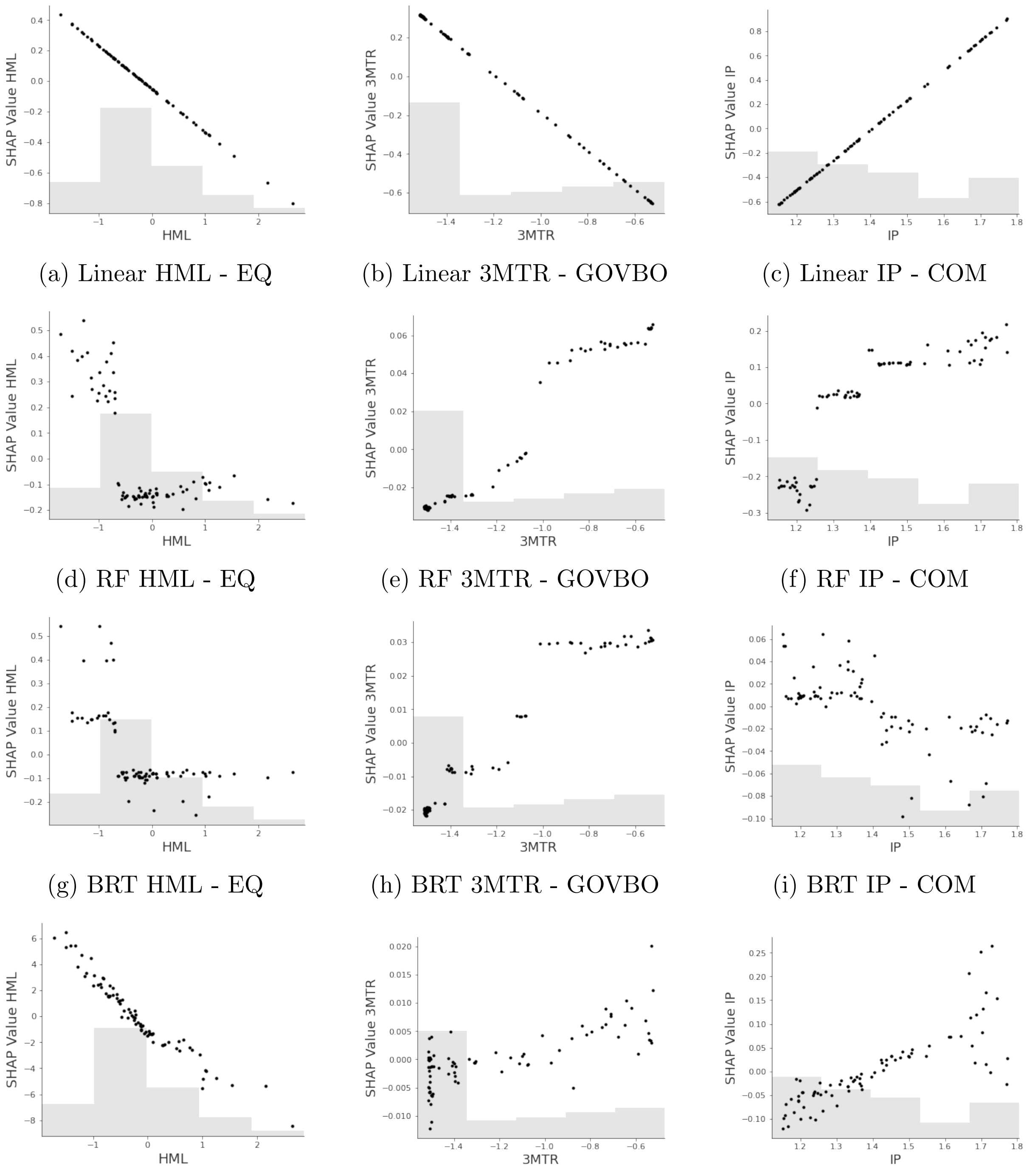
We continue increasing transparency by analyzing scatter plots of the Shapley values for the predictor variable realisations. This results in partial dependence plots of the test sample values between the predictor variables and their contributions to the forecasts produced by the model. The plots show the marginal contributions of a predictor variable on the model's output. Figure 10 displays scatter plots of the Shapley values of the most important predictor variables according to Figure 9. To illustrate the dependence patterns we display the Shapley values derived from the linear model first. As expected, these values are perfectly aligned.

Where in a Linear Regression model, the Shapley values appear on a linear line, we observe that the ML models attribute the predictor variables to the return forecast in a non-linear way. These results are in line with our assumption of non-linear dependencies that exist between the asset returns and the predictor variables. All models show a negative relation between Equity returns and the HML predictor variable. For the Government Bonds return class, however, the linear model shows a negative relation between the 3MTR rate and the returns on the Government Bonds class which contradicts to the signs captured by the ML models. An explanation could be that the predictor set consists of, among others, multiple yield variables. This can result in multicollinearity in the linear model and hence cause a bias in the sign of the coefficients.

For the RF model we observe s-curved dependence patterns between the predictor variables and the return forecasts. We observe a negative relation between the Equity returns and the HML variable. Especially low HML values lead to positive contributions to the next month's return forecasts. A potential explanation could be that this the model picks up on the short term reversal effect in stocks (Jegadeesh (1990)). For the RF models we observe positive relations between the variables 3MTR and IP and return predictions on Government Bonds and Commodities respectively. Considering that we use the returns on 1-month Treasury Bonds as Government Bonds Class returns, we expect a positive relation between the return forecasts and the 3-month Treasury yields according to the term structure of interest rates. Furthermore, we also expect a positive relation between IP levels and return forecasts on Commodities.

Compared to the RF model, the BRT shows step-wise shaped relations between the predictor values and their contribution to the return forecasts. While the RF and the BRT find similar directions of the effect of 3MTR and IP on the Government Bonds and

Figure 10: Shapley Scatter Plots



(a) Linear HML - EQ (b) Linear 3MTR - GOVBO (c) Linear IP - COM

(d) RF HML - EQ (e) RF 3MTR - GOVBO (f) RF IP - COM

(g) BRT HML - EQ (h) BRT 3MTR - GOVBO (i) BRT IP - COM

(j) NN HML - EQ (k) NN 3MTR - GOVBO (l) NN IP - COM

NOTE: This figure displays Shapely Scatter Plots for the Random Forest (RF), XGBoost (XGB) and Neural Network (NN) models, for the High-minus-low (HML), Three Month Treasury Rate (3MTR) and the Industrial Production (IP) predictor variables.

Commodity returns, they show opposite effects of HML predictor on the Equity Class.

The NN model appears to pick up on linear marginal relations rather than on s-shaped or step-wise relations. The directions of the relations are in line with those produced by the RF model. In the scatter plots for the NN we observe a dispersion of the Shapley values for extreme values of the predictor variables. This suggests that the NN shows "certain" behaviour for normal situations of the economy, but becomes more "uncertain" when extreme situations of the economy occur. This indicates that the NN model performs well with normal data, but worse in case of outliers.

In this section we performed a comparison study between the ML models by obtaining variable importance and feature dependence plots using the SHAP methodology. For the RF, BRT and NN models we inspect which variables are most influential to the forecast output. We also show the partial relations between the predictor variables and the asset return forecasts. We generally find similarities in the sign of the relation between the predictors and the forecasts, and find explanations connecting the relations to expectations according to macroeconomic theory.

5 Conclusion

This research aims to identify whether ML models are valuable in predictive asset price modelling of different asset classes. Specifically, we are interested if ML models can outperform traditional linear models in accuracy, while still maintaining explainable model results. To measure the forecasting performance of the ML models in producing one-month-ahead return predictions, we use an expanding window framework. Furthermore, we implement the SHAP methodology proposed by Lundberg and Lee (2017) to obtain variable importance measures and partial dependence plots of the predictor variables and the model forecasts in order to increase model transparency.

The methods are applied to the forecasting of excess returns on an Equity, Government bonds and Commodities asset class over the sample period of April 1987 - December 2019, using a set of predictor variables. We show that the performance of the ML models strongly differs across the different classes. The Boosted Regression Tree model overall attains high Mean Absolute Scaled Error accuracy over the three classes. The return dynamics of Government bonds are best captured by the ML models, which could be explained by their high timeseries persistence. The models generally show similar partial relationships between the predictor variables and the return forecasts. We can link most dependence structures to economic theory, which attributes to the interpretability of the models. The ability to explain the magnitude of the return forecasts make the ML applications valuable to institutional investors and risk managers. The forecasting performance ranking between the models is robust to being in crisis and non-crisis periods. However, the overall performance of the models decreases significantly during the Global Financial Crisis in the period 2007-2008 in our test sample.

In general, the SHAP methodology can be used for explaining individual model output values produced by any ML model, making it a local model agnostic method. We use SHAP for quantifying the contribution that each predictor variable adds to the return forecasts made by the ML models. In this way, we increase model transparency by uncovering the partial relations and feature importance values captured by the models. However, the SHAP results refer to how the models perceive relations between predictor variables and returns after training them on our specific dataset. The return data can contain noise and is subject to external forces that are not incorporated in our predictor variable set. Furthermore, besides the uncertainty resulting from model estimation, an additional level of uncertainty arises by approximating the Shapley values over our model forecasts and interpreting them. Hence, it is important to note that the relations found using SHAP can differ from actual relations in return dynamics. We do, however, find similarities between the partial relations and variable importance measures between the different models. Furthermore, the relations that are found seem understandable and are generally in line with macroeconomic beliefs and theory. This is a positive result that supports our choice for adopting the SHAP method.

Our analysis leaves several directions for future research. One of the restricting elements in our analysis is that the forecast accuracy results are a quantitative measure, while the interpretation section consist of qualitative results. Combining accuracy and interpretability into a single quantitative metric would result in a more efficient model ranking and comparison technique. Furthermore, while new ML algorithms and forecasting techniques are emerging quickly, our research focuses on the performance of only a subset of these models. Centering focus on specific models and investing more time in hyperparameter tuning could increase their relative performance. Furthermore, in our analysis on the Equity and Commodity classes, we considered portfolio indices. Our methodology could be applied to the forecasting of excess returns on individual stocks or commodity types. Additionally, one could simplify the forecasting exercise by solely focusing on the direction of change in asset prices, making it a classification problem instead of forecasting the magnitude.

References

Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621.

Atsalakis, G. S. and Valavanis, K. P. (2009). Surveying stock market forecasting techniques–part ii: Soft computing methods. *Expert Systems with Applications*, 36(3):5932–5941.

Bianchi, D., Guidolin, M., and Ravazzolo, F. (2017). Macroeconomic factors strike back: A Bayesian change-point model of time-varying risk exposures and premia in the US cross-section. *Journal of Business & Economic Statistics*, 35(1):110–129.

Bracke, P., Datta, A., Jung, C., and Sen, S. (2019). Machine learning explainability in finance: an application to default risk analysis.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Chen, L., Pelger, M., and Zhu, J. (2019). Deep learning in asset pricing. *Available at SSRN*.

Chen, N.-F., Roll, R., and Ross, S. A. (1986). Economic forces and the stock market. *Journal of Business*, 59(3):383–403.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Dell, M., Jones, B. F., and Olken, B. A. (2014). What do we learn from the weather? the new climate-economy literature. *Journal of Economic Literature*, 52(3):740–98.

Diebold, F. X. and Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 20(1):134–144.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Du, M., Liu, N., and Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77.

Fama, E. F. and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56.

Fama, E. F. and French, K. R. (1996). *Multifactor explanations of asset pricing anomalies*. University of Chicago Press.

Ferson, W. E. and Harvey, C. R. (1991). The variation of economic risk premiums. *Journal of Political Economy*, 99(2):385–415.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232.

Gan, L., Wang, H., and Yang, Z. (2020). Machine learning solutions to challenges in finance: An application to the pricing of financial products. *Technological Forecasting and Social Change*, 153:119928.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Heaton, J., Polson, N. G., and Witte, J. H. (2016). Deep learning in finance. *arXiv preprint arXiv:1602.06561*.

Hsieh, D. A. (1991). Chaos and nonlinear dynamics: application to financial markets. *The Journal of Finance*, 46(5):1839–1877.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688.

Jegadeesh, N. (1990). Evidence of predictable behavior of security returns. *The Journal of Finance*, 45(3):881–898.

Kotsantonis, S., Pinney, C., and Serafeim, G. (2016). Esg integration in investment management: Myths and realities. *Journal of Applied Corporate Finance*, 28(2):10–16.

Kozlowski, P. J. (1995). Money and interest rates as predictors of regional economic activity. *Review of Regional Studies*, 25(2):143–157.

Lee, M.-C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, 36(8):10896–10904.

Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Nakagawa, K., Uchida, T., and Aoshima, T. (2018). Deep factor model. *ECML PKDD 2018 Workshops*, 11(2):37–50.

Rapach, D. E., Strauss, J. K., and Zhou, G. (2013). International stock return predictability: What is the role of the united states? *The Journal of Finance*, 68(4):1633–1662.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

Ross, S. A. (1976). The arbitrage theory of capital asset pricing. *Journal of Economic Theory*, 13(3):341–360.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Schaller, R. R. (1997). Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59.

Scheinkman, J. A. and LeBaron, B. (1989). Nonlinear dynamics and stock returns. *Journal of Business*, 62(3):311–337.

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *Journal of Finance*, 19(3):425–442.

Stinchcombe, M., White, H., and Hornik, K. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Appendix A Data lags table

Table 6: Predictor Variables Overview

Data	Lag	Source
Economic Variables		
PCE	×	Federal Reserve Economic Data
CPI	1 month	Federal Reserve Economic Data
IP	1 month	Federal Reserve Economic Data
GDP	3 months	Federal Reserve Economic Data
DRP	×	Federal Reserve Economic Data
TS	×	Federal Reserve Economic Data
VXO	×	Federal Reserve Economic Data
RE	1 month	Federal Reserve Economic Data
GFD	1 month	Federal Reserve Economic Data
UR	1 month	Federal Reserve Economic Data
FXD	×	Federal Reserve Economic Data
Yields/rates		
DFF	×	Federal Reserve Economic Data
3MTR	×	Federal Reserve Economic Data
10YTR	×	Federal Reserve Economic Data
30YTR	×	Federal Reserve Economic Data
Climate variables		
CEI	12 months	National Center for Environmental Information
Fama- French factors		
EM	1 month	Kenneth R. French Data Library
SMB	1 month	Kenneth R. French Data Library
HML	1 month	Kenneth R. French Data Library

NOTE: This table presents the lag orders and sources of the predictor values used to train the ML models and to compute return forecasts. We classify the predictor variables among groups of Economic Variables, Yield/rates, Climate variables and Fama-French factors.

Appendix B Crisis Robustness Results

Table 7: Mean Absolute Scaled Error (MASE) for Crisis and Non-Crisis Periods

	Equity	Government Bonds	Commodities
Crisis Period			
Transparent models			
Naive Average Model	1.088	5.585	2.520
Linear regression (baseline)	1.168	1.277	2.988
Ridge	1.018	1.254	2.714
Lasso	1.090	3.759	2.522
ElasticNet	1.069	1.404	2.624
Regression Tree	1.488	1.101	3.189
Non-transparent models			
Random Forest	1.067	1.284	2.595
XGBoost	1.106	1.249	2.800
Neural Network	1.067	1.353	3.861
Non-Crisis Period			
Transparent models			
Naive Average Model	0.577	7.322	1.055
Linear regression (baseline)	0.728	0.471	1.186
Ridge	0.625	0.490	1.076
Lasso	0.570	3.925	1.059
ElasticNet	0.571	0.658	1.075
Regression Tree	0.694	0.416	1.563
Non-transparent models			
Random Forest	0.624	0.431	1.134
XGBoost	0.568	0.394	1.100
Neural Network	0.874	0.572	1.594

NOTE: This table presents the MASE values of the return forecasts produced by the different ML models over a crisis and a non crisis period. We select the return samples in our dataset from 2007 and 2008 as the crisis period, based on the Global Financial Crisis from 2007-2008. The remaining observations are selected as non-crisis sample points, resulting in 24 crisis observations and 172 non-crisis observations.

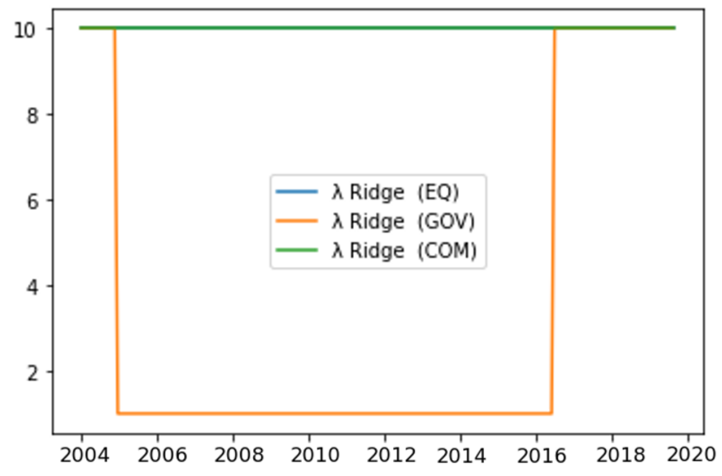
Table 8: Out-Of-Sample R^2 for Crisis and Non-Crisis Periods

	Equity	Government bonds	Commodities
Crisis Period			
Transparent models			
Naive Average Model	-0.042	-1.793	-0.010
Linear Regression (baseline)	-0.030	-0.811	-0.400
Ridge	0.097	-0.808	-0.295
Lasso	-0.045	-1.215	-0.068
ElasticNet	0.036	-0.832	-0.189
Regression Tree	-0.919	-0.804	-0.560
Non-transparent models			
Random Forest	-0.015	-0.822	-0.280
XGBoost	-0.080	-0.816	-0.327
Neural Network	0.057	-0.817	-1.236
Non-Crisis Period			
Transparent models			
Naive Average Model	0.037	-1.886	-0.093
Linear Regression (baseline)	-0.540	0.527	-0.434
Ridge	-0.240	0.526	-0.206
Lasso	0.025	-0.194	-0.102
ElasticNet	-0.019	0.520	-0.144
Regression Tree	-0.750	0.524	-2.052
Non-transparent models			
Random Forest	-0.191	0.527	-0.283
XGBoost	0.048	0.530	-0.218
Neural Network	-1.349	0.519	-1.632

NOTE: This table presents the R^2_{OOS} of the return forecasts produced by the different ML specifications over the test sample January 2003 - December 2019. We select the return samples in our dataset from 2007 and 2008 as the crisis period, based on the Global Financial Crisis from 2007-2008. The remaining observations are selected as non-crisis sample points, resulting in 24 crisis observations and 172 non-crisis observations.

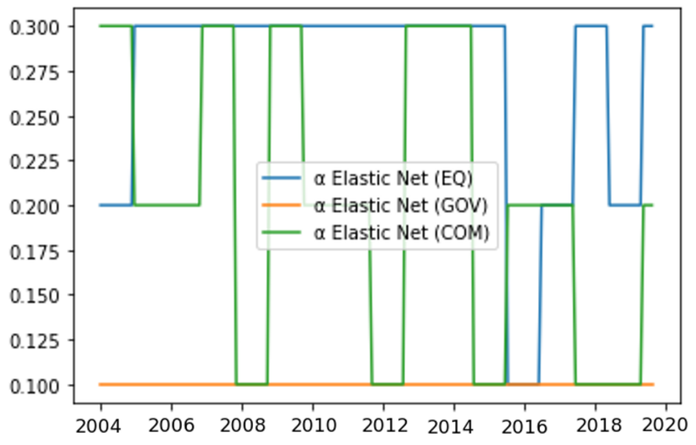
Appendix C Hyperparameter Timeseries Plots

Figure 11: Ridge hyperparameter λ timeseries plot



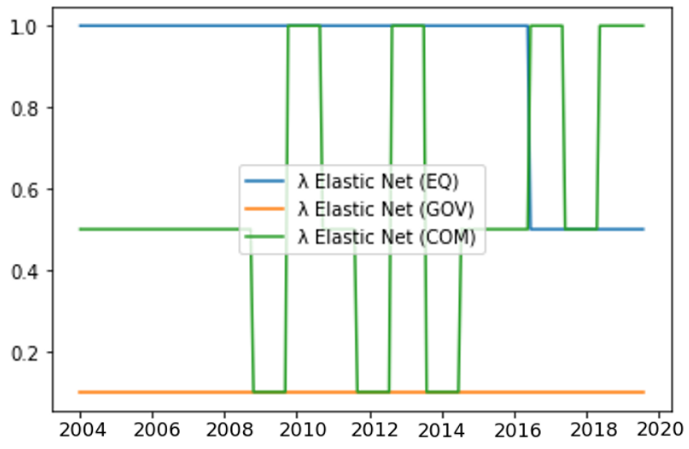
NOTE: This figure shows how the optimal hyperparameter values of λ vary over the 196 iterations of the expanding window framework through the test period for the Ridge model.

Figure 12: Elastic Net hyperparameter α timeseries plot



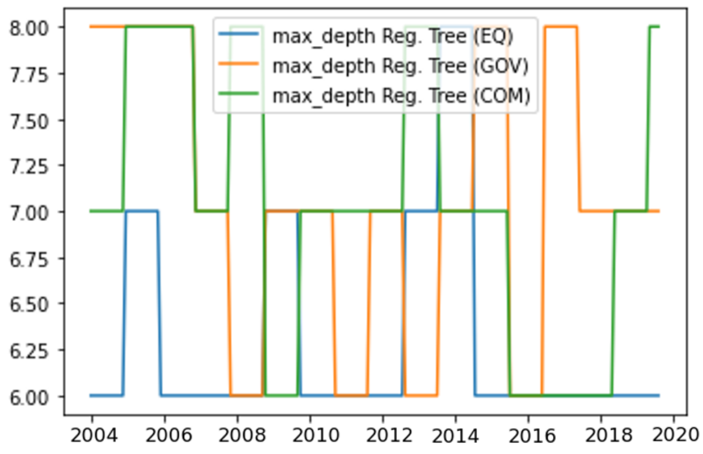
NOTE: This figure shows how the optimal hyperparameter values of α vary over the 196 iterations of the expanding window framework through the test period for the Elastic Net model.

Figure 13: Elastic Net hyperparameter λ timeseries plot



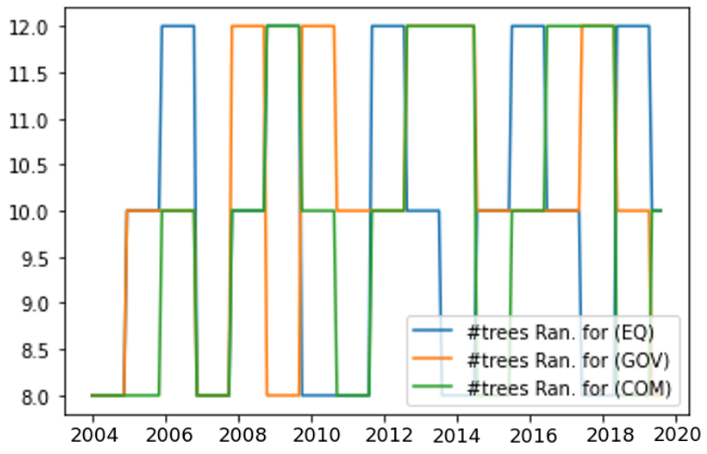
NOTE: This figure shows how the optimal hyperparameter values of λ vary over the 196 iterations of the expanding window framework through the test period for the Elastic Net model.

Figure 14: Regression Tree hyperparameter *max_depth* timeseries plot



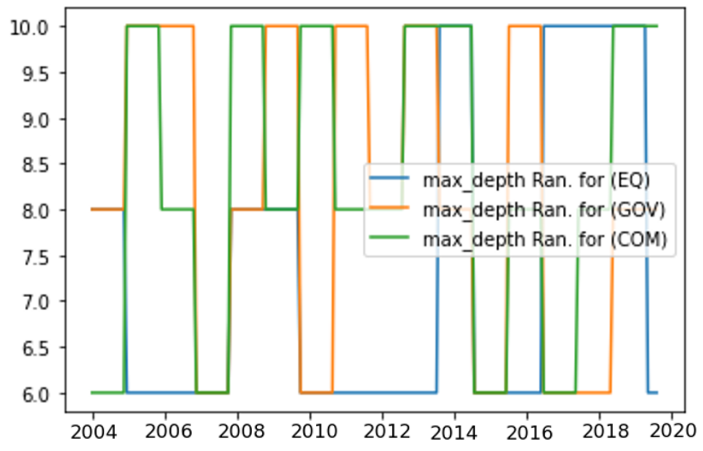
NOTE: This figure shows how the optimal hyperparameter values of *max_depth* vary over the 196 iterations of the expanding window framework through the test period for the Regression Tree model.

Figure 15: Random Forest hyperparameter #Trees timeseries plot



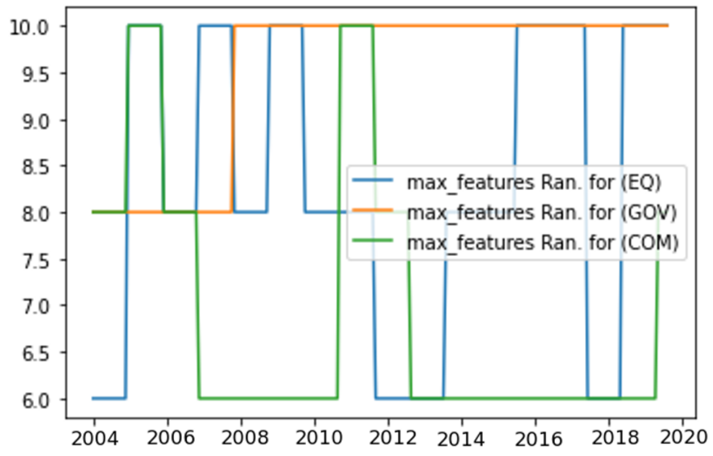
NOTE: This figure shows how the optimal hyperparameter values of #Trees vary over the 196 iterations of the expanding window framework through the test period for the Random Forest model.

Figure 16: Random Forest hyperparameter *max_depth* timeseries plot



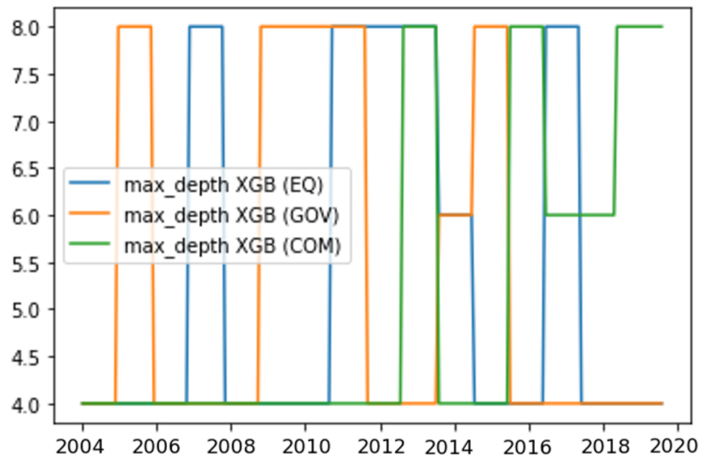
NOTE: This figure shows how the optimal hyperparameter values of *max_depth* vary over the 196 iterations of the expanding window framework through the test period for the Random Forest model.

Figure 17: Random Forest hyperparameter *max_features* timeseries plot



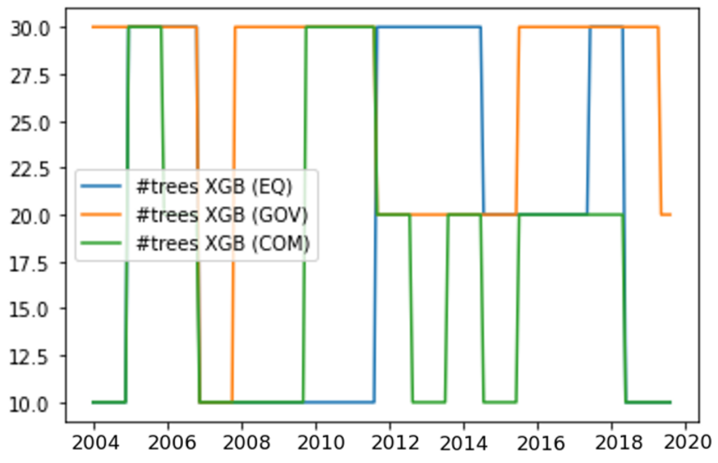
NOTE: This figure shows how the optimal hyperparameter values of *max_features* vary over the 196 iterations of the expanding window framework through the test period for the Random Forest model.

Figure 18: XGBoost hyperparameter *max_depth* timeseries plot



NOTE: This figure shows how the optimal hyperparameter values of *max_depth* vary over the 196 iterations of the expanding window framework through the test period for the XGBoost model.

Figure 19: XGBoost hyperparameter #Trees timeseries plot



NOTE: This figure shows how the optimal hyperparameter values of #Trees vary over the 196 iterations of the expanding window framework through the test period for the XGBoost model.