Erasmus University Rotterdam

Erasmus School of Economics

Bachelor Thesis

International Bachelor Econometrics and Operations Research

Application of Machine Learning for Equity Premium Forecasting

Author:

Fabian Bermana

Student Number:

524259

Supervisor:

Prof. Kleen, Onno

Second Assessor:

van Os, Bram

Abstract

Forecasts of the equity premium are important for asset managers. Machine learning algorithms are applied and compared with econometric models. Models are estimated under expanding windows and AveW, combined with the historical average. Important aspects of the methods are considered, such as consistency, parallel runtime on a cloud cluster, and most importantly, accuracy. Our results show several, but not all machine learning methods are viable for this application. We find that K-Nearest Neighbors and Support Vector Regression have significantly higher performance compared to other models, and additionally exhibit outperformance during periods of extreme positive returns, contrary to previous literature.

July 2022

Contents

1 Introduction

The Equity Premium Puzzle is still a prominent topic in financial economics. First described by Mehra and Prescott (1985), investors and asset managers are interested in improving its predictability. We work with a challenging dataset, where predictors are often weak and the 'best' model changes over time. There is abundant previous research claiming to find good models and predictors. However, prominent papers such as Welch and Goyal (2008) indicate that most are unable to outperform the historical average prediction. Researchers such as Pesaran and Timmermann (1995) have indicated that this is due to the time-varying predictive power of factors and uncertainty during model selection.

We evaluate the performance of several machine learning methods for forecasting the equity premium. Following the paper of Zhang et al. (2020), these models are estimated under both the traditional recursive window scheme and a more advanced AveW technique, which attempts to deal with parameter instability and structural breaks by dividing the estimation sample into several periods, which is combined further with the historical average as shrinkage target.

A diverse range of machine learning models are evaluated. These are: the simple K-Nearest Neighbors from Fix and Hodges (1989), kernel-augmented Support Vector Regression as described in Smola and Schölkopf (2004), the popular Random Forest model from Hastie et al. (2009), the AdaBoost boosting algorithm proposed by Drucker (1997), and three variants of neural networks, including the Elman RNN from Elman (1990), the LSTM from Hochreiter and Schmidhuber (1997), and the GRU from Cho et al. (2014). Additionally, we show that faster computation times are possible with the application of a widely-available commercial cloud platform.

These models are evaluated against the historical average prediction as the benchmark, by comparing the MSE of their forecasts. Evaluations are done with the popular dataset of Welch and Goyal (2008) updated in 2022, including observations from December 1926 to December 2016. From this data, we obtain 12 predictor variables and create forecasts of the excess return on the CRSP value-weighted stock market index.

From our methods, we obtain several key insights. First, the results seen in the paper of Zhang et al. (2020) are highly reproducible, even when they are applied on the updated dataset. The AveW estimation method and HA combination allow several econometric methods to outperform the historical average benchmark. When applying machine learning methods, we find that algorithms requiring less training time and having lower possible hyperparameter combinations to be more appropriate for our application due to computational restrictions. Some of these machine learning methods, such as K-Nearest Neighbors are improved by AveW and

HA combination, but others such as Support Vector Regression are not. In terms of accuracy, we find that K-Nearest Neighbors and RBF-Kernel Support Vector Regression have very high outperformance over the benchmark. Finally, we find very interesting forecasting results of K-Nearest Neighbors and RBF-Kernel Support Vector Regression, where these methods perform the best in periods of highly positive returns, which stands in contrast to most models which perform well in periods of downturn.

Our paper is structured as follows. Section 2 is the Literature Review, where we discuss related research and propose the main research question and related sub-questions. Section 3 is the Data section, where have a short overview on the datasets utilized. Section 4 is the Methodology, where we provide more details on the models, metrics, and tests used for answering the main research question. Section 5 is the Results, which contains the output of our methods and a discussion of what they mean. Section 6 is the Conclusion, where we answer the research question, look through the limitations of our results, and suggest further research topics.

2 Literature Review

Throughout the years, many models and variables have been suggested for predicting the equity premium. Welch and Goyal (2008) examines the performance of several of these variables. Their results show most models perform poorly both in-sample and out-of-sample, failing to improve upon simply predicting the historical mean. They attribute this to the instability of models and frequent structural change. Intuitively, as economic conditions change, it is reasonable to guess that interactions between economic variables change as well.

Methods from the machine learning field have become widely used for financial applications. While machine learning is a broad term with various interpretations, we will consider the following definition: algorithms specialized in prediction, which require few or no assumptions on the nature of the data, and allow for flexible forecast models. Results from papers such as Gu et al. (2020) indicate that machine learning based forecasts are able to perform more than twice as well for asset pricing, compared to more traditional regression methods.

We are interested how these methods may also be used for forecasts of the equity premium. This motivates our main research question: "How can we apply machine learning methods to improve equity premium predictability?"

While the estimation of forecast models are traditionally done through expanding or rolling window estimation as described in Elliott and Timmermann (2016), Zhang et al. (2020) show that the average windows (AveW) method used by Pesaran and Timmermann (2007) allows for better forecasting accuracy, for every model considered by the authors. This method averages the same model class, estimated over windows to accommodate structural breaks. This AveW forecast is then combined again with the historical using the approach seen in Lin et al. (2016), further improving the forecast accuracy of all models used; even allowing the kitchen sink forecasts to perform better than the benchmark. As we aim to improve upon these statistical and econometric methods, we must first replicate the results of Zhang et al. (2020), leading to our first sub-question: "To what extent can we replicate the forecasting performance of Zhang et al. (2020)?"

Many recent papers on asset pricing advocate the use of machine learning methods. Feng et al. (2018) show how deep learning methods as an extension of non-linear econometric models can be applied for predicting the market risk premium. Chen et al. (2019) utilize deep neural networks to predict individual stock returns, proposing a novel three-network model to incorporate the theoretical structure of a stochastic discount factor. Rossi (2018) utilizes boosted regression trees for monthly forecasts of stock return and volatility, creating forecasts based on many conditioning variables without strict assumptions such as linearity, improving the performance of mean-variance portfolio allocation. Bianchi et al. (2021) show that neural networks and trees are also applicable in forecasting the returns of other asset classes, in this case bonds. Comparisons of multiple machine learning models in the context of risk premium prediction are done by Gu et al. (2020). Based on the literature on machine learning in empirical asset pricing, we propose the second sub-question: "Which machine learning methods are appropriate for equity premium forecasting?". To answer this sub-question, we consider the following criteria for an appropriate model. First, a model should perform consistently well, without too much dependence on a perfect implementation. Second, the model should be computable in a reasonable amount of time on a representative modern machine.

While we are able to measure forecasting performance through simply evaluating loss functions, various tests exists which allow a more concrete comparison of forecasts. Zhang et al. (2020) include two main performance measures for each model. The R^2_{OS} is a convenient, interpretable metric to evaluate the performance of a complex model relative to a benchmark. To provide a more statistical judgement, the MSFE-adjusted test developed by Clark and West (2007) can be used to give a probabilistic measure of forecast quality improvement compared to the benchmark. Finally, we construct simple mean-variance portfolios and calculate their $\triangle(ann\%)$ and Sharpe ratio, to provide a more economic measure of forecasting performance. We rely on these methods to assess the performance of our models, leading to our third sub-question: "Are machine learning methods able to produce more accurate forecasts of the equity premium?"

3 Data

3.1 Equity Premium Prediction

The dataset used in this paper contains the same variables as Welch and Goyal (2008), updated with more recent observations and with changes to several data points. From this dataset, we follow Zhang et al. (2020) and include the following 12 predictor variables: dividend–price ratio (DP), log dividend yield (DY), log earnings–price ratio (EP), stock variance (SVAR), book-to-market ratio (BM), net equity expansion (NTIS), Treasury bill rate (TBL), long-term return (LTR), term spread (TMS), default yield spread (DFY), default return spread (DFR), and inflation (INFL). For replication purposes, we use an initial training period of (1926:12–1956:12) and an out-of-sample period (1957:01–2016:12).

Descriptive statistics and a correlation matrix of the data are provided in Appendix B. Note that we can already see the challenge from Figure 1: None of our predictors are highly correlated

Note that while Zhang et al. (2020) use excess returns calculated from the S&P 500 index as the dependent variable, we use the excess returns on the CRSP value-weighted index (EQPREM) to represent the equity premium, which is also included in the data of Welch and Goyal (2008).

4 Methodology

This section details the econometric methods used in our paper. We begin by replicating selected results from the paper of Zhang et al. (2020). The same forecast models are replicated, using the same estimation methods. The performance of these models are evaluated with the same performance metrics. We extend the research through the addition of several machine learning methods and check if the findings still hold when we apply both the forecast models of Zhang et al. (2020) and the machine learning methods to the updated data.

4.1 Econometric Forecast Models

Our paper includes the same forecasting models as in Zhang et al. (2020), which for convenience we will refer to as "Econometric Forecast Models". As the standard model, we use the "kitchen sink" regression described by Welch and Goyal (2008), representing a simple model requiring OLS assumptions while ignoring model uncertainty and parameter instability. The model is defined as

$$r_{t+1} = \boldsymbol{X}_t \boldsymbol{\beta} + e_{t+1}, \quad t = 0, 1, \ldots, T-1 \tag{1}$$

with r_{t+1} being EQPREM at time $t+1$, $\boldsymbol{X}_t$ a $1 \times (p+1)$ vector of predictors as Section described in 3.1, $\boldsymbol{\beta}$ the parameter vector, e_{t+1} a vector of error terms, p the number of predictors, and T the number of time periods or sample size. We also include six forecast models, with short descriptions of them below.

1. Least Absolute Shrinkage and Selection Operator (Lasso): A method for variable selection and regularization during estimation of linear models, popularized by Tibshirani (1996). Compared to OLS, Lasso can be viewed as introducing ℓ_1 norm penalty to the least-squares objective function. Allows coefficients to become zero, and may improve prediction accuracy as it removes less useful predictors.
2. Elastic Net (ElasticNet): Another method for variable selection and regularization, first described by Zou and Hastie (2005). In addition to Lasso's ℓ_1 norm penalty, also adds the ℓ_2 of Ridge regression in an attempt to combine both their advantages.
3. Bayesian Model Averaging (BMA): Model averaging relies on the hypothesis that a combination of several models may produce better forecasts compared to a single one. The BMA forecast as described by Hoeting et al. (1999) utilizes the BIC of individual models to produce an approximation of their analytically optimal BMA weights.
4. Mallows Model Averaging (MMA): Uses the weights minimizing a Mallows criterion. As described in Hansen (2007), this method asymptotically gives the lowest squared error, given that the individual linear regression models are conditionally homoskedastic. Additionally, Hansen (2008) shows that minimizing the Mallows criterion also minimizes the 1-step mean-squared forecast error.
5. Jackknife Model Averaging (JMA): A model averaging technique first described my Hansen and Racine (2012) which allows for heteroskedasticity; asymptotically optimal and gives the lowest expected squared error when there is heteroskedasticity. Weights are determined by minimizing a leave-one-out cross-validation criterion.
6. Weighted-average least squares (WALS): A general model averaging technique first described by Magnus et al. (2010). Deals with the disadvantages of BMA, as the algorithm has linear-time computational complexity, and lowers reliance on a prior probability distribution specification.

MMA and JMA utilize a nested model scheme, where individual linear regressions are made, with the larger models always containing the predictors of the smaller ones. BMA enumerates all possible combinations of predictors and estimates a linear regression on each. Note that more detailed descriptions of these models are included in the paper of Zhang et al. (2020), and Appendix B.1 also includes notes on our implementation.

4.2 Estimation Methods

The following forecast model estimation techniques are used by Zhang et al. (2020), representing both a traditional and more advanced method.

First, we have the Recursively Expanding Window. Each forecast model is estimated using all of the data before some time t, and the model is used to predict a variable's realization at time $t+1$. Whenever we move the estimation point, the data that was previously used for forecasting is then added to the estimation set. For more details on the expanding window, we refer to Elliott and Timmermann (2016).

The Average Windows (AveW) forecasting method as described by Pesaran and Pick (2011) is also utilized. The idea is similar to model averaging, where the forecasts of multiple models are combined. However, we now combine the forecasts of models from different estimation windows. This is also expected to help deal with structural breaks in the data, due to the inclusion of different windows. Let t_0 be the end of the observation window, $m = 10$ the number of estimation windows, with $w_i = w_{\min} + \frac{i-1}{m-1}(t_0 - w_{\min})$, for $i = 1, 2, \ldots, m$ the size of estimation window i. This produces estimation windows $\mathbb{W}_i = \{r_{t+1}, X_t\}_{t=t_0-w_i}^{t_0-1}$, for $i = 1, 2, \ldots, m$. The AveW forecast is then formulated as

$$\hat{r}_{t_0+1}^{\text{AveW}} = \frac{1}{m}\sum_{i=1}^{m} \hat{r}_{t_0+1}\left(\mathbb{W}i\right), \tag{2}$$

with $\hat{r}_{t_0+1}\left(\mathbb{W}i\right)$ as the forecast of a model estimated over window $\mathbb{W}_i$.

4.3 Combining Forecasts

A novel forecast combination approach is the HA combination from Zhang et al. (2020), where the AveW forecasts are then combined with the historical average. This is shown below as

$$\hat{r}_{t_0+1}^{C} = (1-\delta)\hat{r}_{t_0+1}^{\text{HA}} + \delta \hat{r}_{t_0+1}^{\text{AveW}}, \tag{3}$$

where $\hat{r}_{t_0+1}^{\text{HA}}$ is the historical average forecast, $\hat{r}_{t_0+1}^{\text{AveW}}$ the AveW forecast, and δ the forecast combination weight. This weight can then be set through performance-based methods, e.g. giving larger weights to models with lower mean-squared forecast error. Although it is possible to derive theoretically optimal weights, research such as Rapach and Zhou (2013) indicate that when the forecasts from different models are correlated it is difficult to precisely estimate them precisely, causing poor practical performance. As such, Zhang et al. (2020) decide to use a simple equal weighted approach. They hypothesize that this approach "diversifies" the construction of forecasts to help the combined forecast become less sensitive to model uncertainty.

4.4 Evaluation Metrics

The out-of-sample R^2 as described in Campbell and Thompson (2008) is a convenient metric for measuring the performance of forecast models, with its calculation described below as

$$R^2_{\text{OS}} = 1 - \frac{\text{MSFE}^M}{\text{MSFE}^{\text{bmk}}} = 1 - \frac{\frac{1}{q}\sum_{t=t_0+1}^{t_0+q}\left(r_t - \hat{r}_t^M\right)^2}{\frac{1}{q}\sum_{t=t_0+1}^{t_0+q}\left(r_t - \bar{r}_t\right)^2}, \tag{4}$$

with $[t_0+1, t_0+q]$ the time interval for forecast evaluation, MSFE^M and MSFE^{bmk} mean-squared forecast errors of a complicated model and a benchmark model respectively, and q the number forecasts made. The benchmark model will always be the historical average forecast, $\bar{r}_{t_0+1} = \frac{1}{t_0}\sum_{t=1}^{t_0} r_t$. R^2_{OS} is favorable for several reasons. It is simple to interpret, as it is a measure of the percentage increase or reduction in MSFE of a complicated model compared to the benchmark. A positive value shows that a forecast model performs better than the benchmark, and a negative value indicates worse performance.

Clark and West (2007) show that it is possible to test the significance of positive R^2_{OS} values, by testing for significant difference in MSFE. The MSFE-adjusted statistic is defined as follows. Let $\hat{y}_{1t,t+\tau}$ and $\hat{y}_{2t,t+\tau}$ be the τ-steps ahead forecasts of the benchmark and sophisticated model respectively, at time t. Define adj as the sample average of $(\hat{y}_{1t,t+\tau} - \hat{y}_{2t,t+\tau})^2$, which we will use as adjustment. $\text{MSFE}_1 = P^{-1}\sum(y_{t+\tau} - \hat{y}_{1t,t+\tau})^2$ and $\text{MSFE}_2 = P^{-1}\sum(y_{t+\tau} - \hat{y}_{2t,t+\tau})^2$ represent the MSFE of models 1 and 2 respectively. Our null hypothesis is then $H_0 : \text{CW} = \text{MSFE}_1 - (\text{MSFE}_2 - adj) \leq 0$ vs $H_1 : \text{CW} > 0$. For simplicity, we define

$$\hat{f}_{t+\tau} = (y_{t+\tau} - \hat{y}_{1t,t+\tau})^2 - \left[(y_{t+\tau} - \hat{y}_{2t,t+\tau})^2 - (\hat{y}_{1t,t+\tau} - \hat{y}_{2t,t+\tau})^2\right] \tag{5}$$

which we regress on a constant, and perform the test by using the resulting t-statistic, with the number of forecasts made as the degrees of freedom for a test of zero coefficient. Note that this is an approximation, as the distribution of the CW-statistic is non-standard.

To provide a more clear economic meaning of our results, we also calculate the sharpe ratio, and the $\triangle(ann\%)$ metric expressed in annualized percentage return. This value is interpreted as the gain in utility for an investor in terms of additional management fees they would be willing to pay due to the performance improvement provided by constructing a mean-variance portfolio based on a model's forecasts. Additional information on its calculation is provided by Zhang et al. (2020).

4.5 Machine Learning Models

We create forecasts based on several popular machine learning methods and compare their performance to the six sophisticated models included by Zhang et al. (2020). For this section, denote $\{(y_i, x_i)\}$ as a set of dependent variable-predictor vector pairs, (y_{pred}, x_{pred}) a point to be predicted, and $\hat{y}$ the predicted value of y_{pred} made by the model.

Most, if not all machine learning methods require a choice of hyperparameters. These refer to the parameters of a model or algorithm that are chosen before it sees the training data, and will affect both in-sample and out-of-sample performance. This paper uses the simple grid search method to find the best choice of hyperparameters. One major disadvantage of this method is its computation time. To illustrate, let there be a model that requires κ hyperparameters, where for each hyperparameter we search along n_i possible values for $i = 1, \ldots, \kappa$. Assuming a search on each hyperparameter combination takes the same amount of time to evaluate, (which does not hold in practice), grid search will have a time complexity of $\mathcal{O}(n_1 n_2 \cdots n_\kappa)$. Given the nature of our investigation that require models to be re-estimated once per forecast (expanding window) and ten times per forecast (AveW), computation time could mean that some machine learning methods are less appropriate for this task. Note that additional implementation details are provided in Appendix B.2.

4.5.1 K-Nearest Neighbors Regression (KNRUnif, KNRDist)

The K-nearest neighbors algorithm is a supervised, non-parametric method. This algorithm was first proposed by Fix and Hodges (1989), and produces a prediction based on the K data points that have the lowest measure of distance to the predicted point. While Devroye et al. (2013) includes several theoretical results on consistency when using the algorithm for classification, most literature on its use for regression are on implementation details.

For the simplest variant, training a nearest neighbors model is only a matter of storing the data points (y_i, x_i) from the training set. Prediction of some point (y_{pred}, x_{pred}) is more computationally intensive, as described in Imandoust et al. (2013). First, calculate a distance metric d_i, e.g. the Euclidean or Mahalanobis distance, between all x_i of the training set and x_{pred}. These distances are sorted in ascending order. Let A be the set data points with the K smallest values of d_i. The model prediction is calculated as $\hat{y} = \sum_{j \in A} \gamma_j y_j$, with $\{\gamma_j\}$ a set of weights, usually uniform (KNRUnif) or distance-based (KNRDist).

4.5.2 Support Vector Regression (SVRRBF, SVRSigmoid)

While originally used for classification, an extension on support vector machines allows the method to be used on regression problems. This was first seen in Drucker et al. (1996), and

Smola and Schölkopf (2004) gives a detailed overview on how this is done and further developments on the algorithm. With $\{(y_i, x_i)\}$ as training data, we aim to find a function $f(x)$ that maps x_i to y_i with the restriction that the distances between all y_i and $f(x_i)$ are at least some constant ϵ. For this algorithm, this function is always a dot product of x_i. In the linear case, we have $f(x) = \langle \theta, x_i \rangle + \nu$ where θ is a vector of weights. This gives a linear optimization problem that can be written as

$$\text{minimize} \quad \frac{1}{2}||\theta||^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \tag{6}$$

$$\text{s.t.} \quad y_i - \langle \theta, x_i \rangle - \nu \leq \varepsilon + \xi_i, \tag{7}$$

$$\langle \theta, x_i \rangle + \nu - y_i \leq \varepsilon + \dot{\xi}_i^*, \tag{8}$$

$$\xi_i,\, \xi_i^* \geq 0, \tag{9}$$

with ξ_i and ξ_i^* slack variables to allow optimization with infeasible constraints, $C > 0$ constant penalty on the objective function.

There are various methods to introduce non-linearity. A naive approach would be to add squares of predictors to the function. However, this would cause the optimization problem to become quadratic, yielding a high computational cost. A better approach is to use an implicit mapping between predictors can be made by using a kernel $k(x, x') := \langle \Phi(x),\, \Phi(x') \rangle$. We utilize both the radial basis function (SVRRBF) and sigmoid (SVRSigmoid) kernels.

4.5.3 Random Forest Regression (RandomForest)

An ensemble method that utilizes the output of multiple regression trees to produce forecasts. Breiman et al. (2017) describes the use of trees for both regression and classification problems. Regression trees are decision trees with continuous values for its leaves, and are grown by splitting nodes such that the split minimizes a loss criterion, most commonly the mean squared error. However, trees tend to over-fit training data resulting in high variance and low bias. This causes low out-of-sample performance. Random forests as described in Hastie et al. (2009) utilize multiple uncorrelated trees to produce better forecasts. Individual trees are grown with bootstrapped samples, and their forecasts are aggregated (Bagging). Let $\{\mathbf{Z}^b\}_{b=1}^B$ be the set of bootstrap samples, $\{\hat{f}^b(x)\}_{b=1}^B$ the set of regression trees trained on each bootstrap sample b. The bagging prediction is computed as

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x), \tag{10}$$

for B the number of trees. In addition to bootstrapping for each tree grown, random forests apply an additional method when growing the regression trees: when searching for a split, only a subset of the predictors will be used. The reason for this is that a few predictors might be very strong and will be included in many trees. By removing predictors when splitting, correlations between individual trees can be reduced.

4.5.4 AdaBoost Regression (AdaBoost)

Another ensemble method commonly used with decision trees. Unlike the random forest which attempts to create a number of low-correlated trees, AdaBoost is a boosting algorithm, which iteratively trains lower-performing forecast models with respect to the performance of its predecessors to build better performing models.

The AdaBoost regression is described in Drucker (1997). To initialize the algorithm, set weights $\{w_i\}$, $w_i = 1/N$ for each item in the training set $\{(y_i, x_i)\}$, N the size of the training set. The following procedure is repeated until a stopping criterion is reached, which can be some iteration limit J. Let j be the iteration number. Take a bootstrap sample with size N_1, $\{(y_k, x_k)\}_{k=1}^{N_1}$ with replacement, with $p_i = w_i/\Sigma_{i=1}^{N} w_i$ as the probability observation i of the training set is included in the sample. Train the regression tree on all observations k from the bootstrap sample to produce the model $\hat{f}_j(x)$. With the model, produce predictions $\{\hat{y}_{k,j} = \hat{f}_j(x_k)\}_{k=1}^{N_1}$ and calculate a loss function L_k on each prediction, the average loss $\overline{L} = \sum_{k=1}^{N_1} L_k p_k$, and $\boldsymbol{\beta}_j = \frac{\overline{L}}{1-\overline{L}}$ as the measure of predictor confidence. Then, update each weight as $w_i = \beta^{**}[1 - L_i]$. After all iterations are completed, we obtain models $\{\hat{f}_j(x)\}_{j=1}^{J}$, and model confidence measures $\{\boldsymbol{\beta}_j\}_{j=1}^{J}$. These are used to produce the AdaBoost forecast

$$\hat{f}_{Ada}(x) = inf \left\{ y \in Y : {}_{j:\, \hat{f}_j(x) \leq\, y} \log(1/\boldsymbol{\beta}_j) \geq \frac{1}{2} \sum_j \log(\mathbf{1}/\boldsymbol{\beta}_j) \right\}. \tag{11}$$

This iterative procedure would intuitively cause over-fitting. However, Wyner et al. (2017)'s derivations show AdaBoost's empirical effectiveness can be explained from its property as a weighted ensemble classifier, effectively becoming a "random forest of forests".

4.5.5 Neural Networks (RNN, LSTM, GRU)

Neural networks are models which rely on groups of nodes, each of which receives input vectors, and calculates one or more mathematical functions to produce outputs. Wang (2003) provides a description of the traditional neural network. Each node has parameters commonly called weights and biases. Weights $V_{a,b}$ of node a multiply the b^{th} element of input vector x of length L_a (Note the subscript on the length, as different nodes in a network may have different input lengths), which are then summed together to produce a scalar along with the bias term b_a. The resulting sum is then passed through a non-linear function $g(.)$, called the activation function. This results in the node's output h_a, as shown below as

$$h_a = g(\sum_{b=1}^{L_a} V_{ab} x_b + b_a). \tag{12}$$

Groups of nodes are organized into layers. In general the outputs of one layer becomes the input for the next. The strength of neural networks comes from the universal approximation theorem(s); it is proven that multi-layer neural networks can represent any computable function. We will be using three types of neural networks that have empirically been shown to perform well on data in the form of sequences.

The RNN, proposed by Elman (1990) is a simple method to incorporate a time or sequence structure into neural network models. This is done by adding an additional input to each node: its own output from the previous time period. While this would seem as if each time period adds an additional layer to the network, replacing regular neural network nodes with recurrent nodes does not add additional layers. Another type of recurrent network we use is the LSTM, which was proposed by Hochreiter and Schmidhuber (1997). This type of node is composed of multiple mathematical functions and a cell that stores long-term information. Gates which act similarly to simple neural network nodes (also containing inputs, parameters, and outputs) determine how much information from the cell is kept (forget gate), and what information is used to update the cell (input gate). The LSTM node has become one of the most popular methods for modeling sequence data due to its high empirical performance. Lastly, we use the GRU node introduced by Cho et al. (2014). Similar to the LSTM nodes, GRU nodes utilize gates to control the flow of information. But unlike the LSTM, GRU nodes combine the forget and input gates into a single gate and uses a single hidden-cell state.

4.6 Computational Methods

The increase in available computational power has made the use of complex statistical methods more practical. Several methods from Section 4.1 are known to be computationally expensive (e.g. BMA). The machine learning methods in Section 4.5 may require even more, due to cross-validation. This could present a problem, especially when AveW is used; a reasonable guess is computation time increases by a factor of m compared to expanding window.

While Moore's law was reasonably accurate in predicting the doubling of single-thread processor speeds every two years, industry experts such as Sutter et al. (2005) foresee that additional gains in computational power have to come from more processors instead of higher clock speeds. As such, using a more recent processor model will not allow much faster runtime.

Fortunately, all of our estimation schemes in Section 4.2 belong in the class of embarrassingly parallel algorithms, which easily allows multiprocessing. Every forecast made is independent of all the others; no intermediate results need to be shared and only input data needs to be separately parsed. As such, our computer code can be written in a way that utilizes multiple CPU cores running in parallel, and can expect computation time to be approximately inversely related to the number of CPU cores used. It will not be exactly inversely related due to differing overheads required when creating the different inputs of each forecast.

5 Results

Model Name	R^2_{OS} **(%)**	**CW Test**	**Time (s)**
Expanding Window			
LinearRegression	-7.72%	0.51	9
Lasso	-0.56%	-0.32	16
ElasticNet	-0.56%	-0.31	14
BMA	-2.06%	-1.31	446
MMA	-3.39%	0.27	16
JMA	-4.01%	0.63	19
WALS	-3.97%	0.38	31
AveW			
LinearRegression	-2.02%	1.97**	9
Lasso	0.59%	1.72**	138
ElasticNet	0.77%	1.90**	133
BMA	-0.81%	1.01	2,894
MMA	-0.04%	1.77**	64
JMA	0.21%	2.09**	73
WALS	0.05%	1.94**	198
AveW + HA			
LinearRegression	1.14%	1.97**	-
Lasso	0.66%	1.72**	-
ElasticNet	0.76%	1.90**	-
BMA	0.20%	1.01	-
MMA	1.05%	1.76**	-
JMA	1.40%	2.14**	-
WALS	1.22%	1.94**	-

((a)) Econometric Forecast Models

Model Name	R^2_{OS} **(%)**	**CW Test**	**Time (s)**
Expanding Window			
KNRUnif	0.19%	2.10**	65
KNRDist	0.03%	1.98**	62
SVRRBF	1.37%	3.20***	72
SVRSigmoid	-0.02%	1.52*	63
RandomForest	0.23%	1.56*	785
AdaBoost	0.17%	1.20	1,896
AveW			
KNRUnif	0.85%	2.61***	449
KNRDist	0.61%	2.46***	461
SVRRBF	0.82%	2.91***	365
SVRSigmoid	-0.87%	1.30*	331
RandomForest	0.51%	1.87**	7,153
AdaBoost	-0.97%	0.75	16,100
AveW + HA			
KNRUnif	1.24%	2.61***	-
KNRDist	1.16%	2.46***	-
SVRRBF	1.30%	2.91***	-
SVRSigmoid	0.31%	1.30*	-
RandomForest	0.50%	1.87**	-
AdaBoost	-0.01%	0.75	-
Expanding Window - 12 Period Reestimation			
RNN	-7.41%	-226.74	864
LSTM	-3.68%	-378.51	1,249
GRU	-7.41%	-226.74	841

((b)) Machine Learning Models

Note: This table contains statistics measuring out-of-sample forecast evaluation. R^2_{OS} contains out-of-sample R^2 values. CW Test refers to the MSFE-Adjusted test statistic of Clark and West (2007). Asterisks (*) indicate significance at 10% (*), 5% (**), and 1% (***) levels. Time refers to the real-time running time required for model estimation, forecasting, and forecast evaluation. AveW + HA forecasts were derived from pre-computed AveW values; as such computation time is not relevant. (a) denotes running time without multiprocessing (not run on multiple CPU cores).

Table 1: Forecast Evaluation Results

This section discusses the main results of our research. Evaluation of the models in Section 4.1 and Section 4.5 were run on a GCP (Google Cloud) E2 machine with 16 vCPUs and 16GB RAM. We believe this machine is a representative example of a modern desktop computer. The advantage of using a cloud platform is that it enables anyone to access additional computational power without having to buy more hardware. In practice, we used three instances to receive results earlier.

5.1 Econometric Forecast Models

From our investigation, we found that the results of Zhang et al. (2020) were reasonably replicable, with several key differences. This can be seen in Table 1(a), which contains metrics computed from forecasts made using the methods from 4.1, and can begin to compare our results with theirs.

Under expanding window estimation, we observe a very similar pattern in R^2_{OS} values, with LinearRegression performing the worst and Lasso very close to the historical average. In our case, ElasticNet and JMA perform slightly worse. The performance of JMA may be explained by the implementation details of JMA as seen in Appendix B.1, where the ordering of nested models is randomized. With ElasticNet, the difference is most likely due to the changes in the data. The values of CW Test statistics also suggest similar outcomes, where none of the forecasts are significantly better than the historical average even at a 10% significance level.

AveW estimation also provides similar benefits as seen in Zhang et al. (2020). All of the econometric forecast models are improved by the forecasting scheme, allowing Lasso, ElasticNet, JMA, and WALS to beat the historical average in terms of R^2_{OS} and confirmed by the CW Test at 5% significance. These results indicate that the AveW scheme does help produce forecasts more robust to structural breaks and time-varying predictor strength, at least for these popular econometric models.

HA combination manages to improve the performance of most models. This is especially true for LinearRegression, BMA, and MMA, which previously had negative R^2_{OS} values. ElasticNet is the anomaly, where forecasting performance slightly drops. Across the board, the significance of CW Test statistics does not change compared to AveW. For the econometric forecast models,

Computation times follow a similar pattern as Zhang et al. (2020), with LinearRegression taking the least amount of time and BMA requiring much more. Additionally, our use of parallel computing shows major improvements in computation time. Zhang et al. (2020) indicate that BMA with AveW requires more than 20,000 seconds to complete, while ours requires only around 2,900 seconds, which is nearly a 10-fold improvement. One may suggest that this is due to implementation or programming language differences. However, sources such as Virtanen et al. (2020) indicate that the speed of Python is not significantly different from Matlab for numerically-heavy operations, suggesting that the speed increase comes from elsewhere, namely multiprocessing. One thing to note is that although it is unfortunate WALS was not parallelized as explained in B.1, computation times remain quite low, with AveW WALS not taking more than 50% longer than the parallelized AveW Lasso.

5.2 Machine Learning Methods

We have found that many, but not all of our machine learning models from 4.5 are able to produce good forecasts of the equity premium. These results are displayed in Table 1(b).

The traditional expanding window estimation produces surprisingly good results. Most of the machine learning methods, except for SVRSigmoid, already beat the benchmark as shown by positive R^2_{OS}. SVRRBF performs exceptionally well, producing the highest R^2_{OS} value across all models estimated under expanding window. The CW test statistics confirm this as well, showing that both variants of K-nearest neighbors outperform the historical average at 5% significance, and SVRRBF at 1% significance.

However, the AveW estimation does not provide universal performance improvements. While both variants of K-nearest neighbors receive large improvements in terms of R^2_{OS} and significance of the CW test, we observe lower performance for both variants of support vector regression and AdaBoost. These results suggest that AveW estimation is not universally applicable for improving forecasts.

Similar to the previous results in 5.1, HA combination improves most, but not all machine learning models estimated under AveW, with RandomForest being the exception. While this allows both variants of K-nearest neighbors to have very significant CW test statistics, it is interesting to note that for SVRRBF, HA combination does not produce better forecasts as the simple expanding window estimation. From this, we suggest that AveW and HA combination should only be applied for specific machine learning models.

Among the K-neighbors, support vector, and tree-based models, computation times are quite different. The K-neighbors and support vector run fairly quickly, and even on AveW would not require more than 10 minutes to complete. The tree-based models are much more expensive, each taking more than an hour to complete. Given the poorer performance and much higher computation times of the tree-based models, we indicate our preference for the K-neighbors and support vector models.

Results for the neural network models were placed in a separate section, as we had to apply a different estimation scheme for them: the models were re-estimated after every 12 periods instead of one model per period. This was unfortunately caused by high computation time. We can see that even for the fastest GRU model, the regular expanding window scheme would take approximately $800 \cdot 12 = 9600$ seconds to complete, which with AveW could increase that further to around $9600 \cdot 10 = 96000$ seconds. Even so, we can see from a glance they perform very poorly. As indicated in Appendix B.2, our implementation includes very few runs of cross-validation, which is likely one of the reasons for the poor performance. Additional cross-validation is infeasible, unless we can access much more computational power, possibly through GPUs. Our results suggest that neural networks may be a poor choice when we are required to re-estimate the model many times.

5.3 Forecasting Extreme Periods

Period	**Observations**	**LinearRegression**	**WALS**	**SVRRBF**	**KNRUnif**
Moderate Absolute Returns					
$0 < \|r_t\| \leq 0.5$	299/720	-16.44%	-8.03%	-2.87%	-1.45%
Extreme Absolute Returns					
$\|r_t\| \geq 0.5$	234/720	-0.26%	0.51%	-0.12%	-0.01%
$\|r_t\| \geq 1.0$	96/720	2.19%	1.70%	1.37%	1.14%
$\|r_t\| \geq 1.5$	52/720	-1.36%	-0.88%	1.61%	1.24%
$\|r_t\| \geq 2.0$	39/720	4.02%	3.18%	2.08%	2.08%
Extreme Positive Returns					
$\mathrm{r}_t \geq 0.5$	124/720	-3.63%	-1.03%	12.33%	7.14%
$\mathrm{r}_t \geq 1.0$	52/720	-1.34%	-0.61%	7.47%	2.71%
$\mathrm{r}_t \geq 1.5$	25/720	-6.40%	-4.14%	6.02%	2.17%
$\mathrm{r}_t \geq 2.0$	14/720	-5.17%	-3.29%	4.07%	1.17%
Extreme Negative Returns					
$\mathrm{r}_t \leq 0.5$	110/720	3.51%	2.23%	-14.07%	-8.02%
$\mathrm{r}_t \leq 1.0$	44/720	5.51%	3.88%	-4.40%	-0.35%
$\mathrm{r}_t \leq 1.5$	27/720	2.70%	1.73%	-1.94%	0.49%
$\mathrm{r}_t \leq 2.0$	25/720	7.68%	5.76%	1.29%	2.45%

Note: This table contains R^2_{OS} values for a selection of models, divided over periods of extreme returns. Here, r_t denotes returns normalized over the out-of-sample period of 1957:01–2016:12. All model forecasts were computed with the HA combination method.

Table 2: Model Performance During Extreme Periods

We find very interesting results when our models are evaluated for different magnitudes of extreme returns. Following Zhang et al. (2020), we examine the performance of our models during periods of extreme market movement. To provide additional insight, we include computations of R^2_{OS} when normalized returns are positive.

Overall, the four models in Table 2 show underperformance compared to the historical average benchmark during periods of moderate normalized returns. Economists such as Fama and French (1989) have theorized that the equity premium is more predictable during economic downturns, suggesting that predictions based on complex models would perform better during extreme downturns. We observe this reflected LinearRegression and WALS, where they show much higher R^2_{OS} when normalized returns are negative.

While SVRRBF and KNRUnif also perform better in periods of extreme market movements, we observe that this occurs when extreme normalized returns are positive. Meanwhile, these two models seem to do much worse when extreme normalized returns are negative. These results stand in contrast to previous literature and call for further investigation, suggesting the existence of models better at capturing the data-generating process of the equity premium when markets are rising. A combination of models performing better in extreme rising and falling markets, and the historical average prediction during moderate times, may be able to provide better overall predictions.

5.4 Portfolio Performance

From Table 3, we can see that the econometric forecast models produce lower performance overall compared to Zhang et al. (2020), both in terms of $\triangle(\mathbf{ann}\%)$ and Sharpe ratio values. In particular, we do not observe the spectacular $\triangle(ann\%)$ values for portfolios constructed using forecasts from LinearRegression, WALS, and MMA. In fact for BMA, this value becomes negative; an investor would not want to pay management fees if a fund would rely on this model alone.

Fortunately, our machine learning models show more favorable results. KNRUnif, KNRDist, and SVRRBF produce very high utility for investors, exceeding all the econometric forecast models. Portfolios constructed using forecasts of these models are also better in terms of the Sharpe ratio. These results are of particular interest to asset managers, as a fund utilizing these models could be more marketable.

Model	$\triangle$(**ann**%)	**Sharpe**
Econometric Forecast Models		
LinearRegression	0.700	0.107
LassoCV	0.550	0.119
ElasticNetCV	0.666	0.123
BMA	-0.069	0.091
MMA	0.743	0.110
JMA	0.907	0.115
WALS	1.077	0.118
Machine Learning Models		
KNRUnif	1.615	0.140
KNRDist	1.558	0.139
SVRRBF	2.270	0.154
SVRSigmoid	0.618	0.106
RandomForest	0.346	0.115
AdaBoost	-0.215	0.089

Note: This table contains $\triangle(ann\%)$ and Sharpe ratio values evaluated over the out-of-sample period of 1957:01–2016:12. All model forecasts were computed with the HA combination method.

Table 3: Portfolio Performance Metrics

6 Conclusion

In our research, we aim to answer the following main question: "How can we apply machine learning methods to improve equity premium predictability?". To do this, we first investigated whether the performance improvements obtained by Zhang et al. (2020) through AveW estimation and HA combination are replicable on the updated data of Welch and Goyal (2008). We then proceed to apply the same estimation schemes to multiple machine learning methods. The out-of-sample R^2 metric and the MSFE-Adjusted test statistic of Clark and West (2007) are then used to evaluate the performance of our models.

Running the models of Zhang et al. (2020), we find that their results are nearly perfectly replicable, with some minor exceptions. JMA performs worse on expanding window, and HA combination does not improve ElasticNet. Most of the time, for the econometric forecast models, we see that AveW estimation is better than expanding window, and HA combination is better than both. We also find that when using HA combination, the performance of LinearRegression and WALS is much better when standardized equity premiums

Applying our machine learning methods, we find that some are more suited than others for our investigation. K-Nearest Neighbors, Random Forest, and AdaBoost perform consistently on different estimation methods and are improved by the advanced AveW estimation and the HA combination forecast. Meanwhile, Support Vector Regression is more difficult to use, as it relies on a more perfect implementation to achieve the best results. Neural network models are not well suited for this problem, as they are too computationally intensive and require much more complicated cross-validation to perform well.

Several classes of machine learning methods are able to produce benchmark-beating forecasts of the equity premium. A few, namely Uniform-Weighted K-Nearest Neighbors and RBF Kernel Support Vector Regression perform significantly better than the other models we ran, achieving high out-of-sample R^2 at 1% significance, which had never been seen beforehand. When used for portfolio construction, these models allow for both higher utility and risk-adjusted return compared to portfolios constructed using only econometric forecast models. In addition, we find the surprising result that these two methods perform better when standardized returns are positive, while other models are performing better during periods of negative standardized returns.

Overall, we conclude that machine learning can be utilized to produce good-quality forecasts of the equity premium. Not all machine learning methods are applicable to the problem, and some require a careful choice of estimation scheme and hyperparameter tuning. However, when done properly, machine learning allows significant outperformance often found during periods with different return characteristics compared to models previously considered.

From our investigation, we propose a direct area for further research: combining our predictive models with a directional prediction of the stock market. If we are able to find sufficiently accurate predictions of the stock market direction, we can leverage the fact that we have models performing much better during rising and falling stock markets to produce more accurate forecasts of the equity premium. We also discuss our limitation on computing power: some of our models, such as the neural networks, were unable to perform well. This was likely because we did not properly tune their hyperparameters. In practice, financial institutions interested in using our methods will have significantly more computing resources, and may be able to achieve better results.

As a final note, we would like to thank our thesis supervisor Prof. Onno Kleen for consistent guidance and helpful input throughout the research process.

References

Andersen, M. S., J. Dahl, and L. Vandenberghe (2010). Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Mathematical Programming Computation 2*(3), 167–201.

Bianchi, D., M. Büchner, and A. Tamoni (2021). Bond risk premiums with machine learning. *The Review of Financial Studies 34*(2), 1046–1089.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (2017). *Classification and regression trees.* Routledge.

Campbell, J. Y. and S. B. Thompson (2008). Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies 21*(4), 1509–1531.

Chen, L., M. Pelger, and J. Zhu (2019). Deep learning in asset pricing. *arXiv preprint arXiv:1904.00745*.

Cho, K., B. Van Merriënboer, D. Bahdanau, and Y. Bengio (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Clark, T. E. and K. D. West (2007). Approximately normal tests for equal predictive accuracy in nested models. *Journal of econometrics 138*(1), 291–311.

Devroye, L., L. Györfi, and G. Lugosi (2013). *A probabilistic theory of pattern recognition*, Volume 31. Springer Science & Business Media.

Drucker, H. (1997). Improving regressors using boosting techniques. In *ICML*, Volume 97, pp. 107–115. Citeseer.

Drucker, H., C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik (1996). Support vector regression machines. *Advances in neural information processing systems 9*.

Elliott, G. and A. Timmermann (2016). *Economic Forecasting.* Princeton University Press.

Elman, J. L. (1990). Finding structure in time. *Cognitive science 14*(2), 179–211.

Fama, E. F. and K. R. French (1989). Business conditions and expected returns on stocks and bonds. *Journal of financial economics 25*(1), 23–49.

Feng, G., J. He, and N. G. Polson (2018). Deep learning for predicting asset returns. *arXiv preprint arXiv:1804.09314*.

Fix, E. and J. L. Hodges (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique 57*(3), 238–247.

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

Hansen, B. E. (2007). Least squares model averaging. *Econometrica 75*(4), 1175–1189.

Hansen, B. E. (2008). Least-squares forecast averaging. *Journal of Econometrics 146*(2), 342–350.

Hansen, B. E. and J. S. Racine (2012). Jackknife model averaging. *Journal of Econometrics 167*(1), 38–46.

Hastie, T., R. Tibshirani, J. H. Friedman, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*, Volume 2. Springer.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Hoeting, J. A., D. Madigan, A. E. Raftery, and C. T. Volinsky (1999). Bayesian model averaging: a tutorial (with comments by m. clyde, david draper and ei george, and a rejoinder by the authors. *Statistical science 14*(4), 382–417.

Imandoust, S. B., M. Bolandraftar, et al. (2013). Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. *International Journal of Engineering Research and Applications 3*(5), 605–610.

Joblib Development Team (2020). Joblib: running python functions as pipeline jobs.

Lin, H., C. Wu, and G. Zhou (2016). Forecasting corporate bond returns: An iterated combination approach. *Available at SSRN 2346299*.

Magnus, J. R., O. Powell, and P. Prüfer (2010). A comparison of two model averaging techniques with an application to growth empirics. *Journal of econometrics 154*(2), 139–153.

Mehra, R. and E. C. Prescott (1985). The equity premium: A puzzle. *Journal of monetary Economics 15*(2), 145–161.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani,

S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Pesaran, M. H. and A. Pick (2011). Forecast combination across estimation windows. *Journal of Business & Economic Statistics 29*(2), 307–318.

Pesaran, M. H. and A. Timmermann (1995). Predictability of stock returns: Robustness and economic significance. *The Journal of Finance 50*(4), 1201–1228.

Pesaran, M. H. and A. Timmermann (2007). Selection of estimation window in the presence of breaks. *Journal of Econometrics 137*(1), 134–161.

Rapach, D. and G. Zhou (2013). Forecasting stock returns. In *Handbook of economic forecasting*, Volume 2, pp. 328–383. Elsevier.

Rossi, A. G. (2018). Predicting stock market returns with machine learning. *Georgetown University*.

Smola, A. J. and B. Schölkopf (2004). A tutorial on support vector regression. *Statistics and computing 14*(3), 199–222.

Sutter, H. et al. (2005). The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's journal 30*(3), 202–210.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*(1), 267–288.

Tietz, M., T. J. Fan, D. Nouri, and B. Bossan (2017, July). skorch: A scikit-learn compatible neural network library that wraps pytorch.

Virtanen, P., R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen,

E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17*, 261–272.

Wang, S.-C. (2003). Artificial neural network. In *Interdisciplinary computing in java programming*, pp. 81–100. Springer.

Welch, I. and A. Goyal (2008). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies 21*(4), 1455–1508.

Wyner, A. J., M. Olson, J. Bleich, and D. Mease (2017). Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research 18*(1), 1558–1590.

Zhang, H., Q. He, B. Jacobsen, and F. Jiang (2020). Forecasting stock returns with model uncertainty and parameter instability. *Journal of Applied Econometrics 35*(5), 629–644.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology) 67*(2), 301–320.

A Data

	count	mean	std	min	25%	50%	75%	max
EQPREM	1080.0	0.005052	0.054502	-0.339221	-0.020066	0.009416	0.034981	0.346243
DP	1080.0	-3.365963	0.460030	-4.523640	-3.584079	-3.338036	-3.023517	-1.873246
DY	1080.0	-3.361231	0.458045	-4.530894	-3.583574	-3.332392	-3.020019	-1.912904
EP	1080.0	-2.732454	0.416647	-4.836482	-2.939130	-2.785351	-2.458990	-1.774952
SVAR	1080.0	0.002887	0.005777	0.000072	0.000705	0.001267	0.002453	0.070945
BM	1080.0	0.571922	0.265128	0.120510	0.346138	0.546952	0.751562	2.028478
NTIS	1080.0	0.017616	0.025549	-0.055954	0.006365	0.017275	0.027534	0.177040
TBL	1080.0	0.034343	0.031039	0.000100	0.003800	0.030000	0.051825	0.163000
LTR	1080.0	0.004780	0.024481	-0.112400	-0.007000	0.003150	0.016100	0.152300
TMS	1080.0	0.017179	0.013113	-0.036500	0.008700	0.017750	0.026200	0.045500
DFY	1080.0	0.011311	0.006956	0.003200	0.006900	0.009100	0.013300	0.056400
DFR	1080.0	0.000309	0.013629	-0.097500	-0.005025	0.000500	0.005500	0.073700
INFL	1080.0	0.002436	0.005354	-0.020548	0.000000	0.002422	0.005081	0.058824

Note: This table contain descriptive statistics for our data. Some are taken directly from Welch and Goyal (2008), while items such as EQPREM are derived from it.

Table 4: Descriptive Statistics

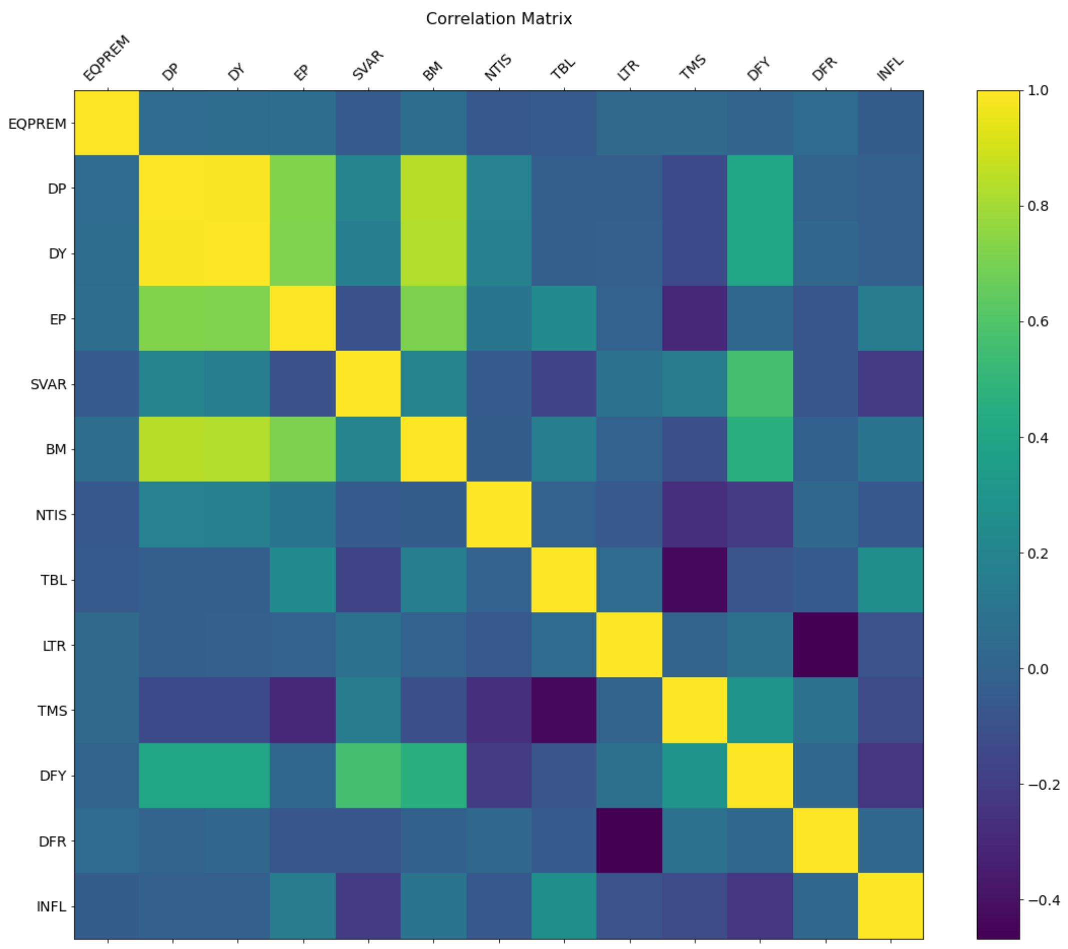

Figure 1: Correlation Matrix

B Implementation

- Parallel computation: Joblib Development Team (2020) package allows more robust parallelization of functions

B.1 Econometric Forecast Models

- Lasso: Implementation in Python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation.
- ElasticNet: Implementation in Python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation.
- BMA: Model averaging of individual linear regressions. Models estimated on all possible combinations of predictors are included.
- MMA: Python port of Matlab code from Hansen and Racine (2012). Initialization modified for ordering of nested models: First, Lasso is run to get initial included variables based on

estimated coefficient size. Then, variables with zero Lasso coefficient are randomly ordered, before estimation of the nested models. Includes use of the optimizer from Andersen et al. (2010).

- JMA: Python port of Matlab code from Hansen and Racine (2012). Initialization modified for ordering of nested models: First, Lasso is run to get initial included variables based on estimated coefficient size. Then, variables with zero Lasso coefficient are randomly ordered, before estimation of the nested models. Includes use of the optimizer from Andersen et al. (2010).
- WALS: Run with Matlab code of Magnus et al. (2010) called from a Python environment. Implementation is not parallelized, as we found that a new Matlab engine needed to be initialized for every forecast using our current expanding window and AveW framework, creating a computation overhead which is more expensive than simply sequentially producing forecasts.

B.2 Machine Learning Models

- KNRUnif: Implementation in python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation on: 50 values of number of neighbors.
- KNRDist: Implementation in python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation on: 50 values of number of neighbors.
- SVRRBF: Implementation in python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation on: 10 values on strength of regularization.
- SVRSigmoid: Implementation in python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation on: 10 values on strength of regularization.
- RandomForest: Implementation in python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation on: 3 values on minimum samples per leaf, 2 values of maximum features per split.
- AdaBoost: Implementation in python from Pedregosa et al. (2011). Estimation using (regular) 5-Fold cross-validation on: 3 values of learning rate.
- RNN: Implementation in python from Paszke et al. (2019), wrapped in Tietz et al. (2017) for compatibility with our forecasting framework. Estimation using (regular) 3-Fold cross-validation on: 2 values of learning rate, 2 values of number of layers, 2 values of layer size.
- LSTM: Implementation in python from Paszke et al. (2019), wrapped in Tietz et al. (2017) for compatibility with our forecasting framework. Estimation using (regular) 3-Fold cross-

validation on: 2 values of number of layers, 2 values of layer size.

- GRU: Implementation in python from Paszke et al. (2019), wrapped in Tietz et al. (2017) for compatibility with our forecasting framework. Estimation using (regular) 3-Fold cross-validation on: 2 values of learning rate, 2 values of number of layers, 2 values of layer size.