Bachelor Thesis International Econometrics & Operations Research

FINAL VERSION

Directional Forecasting of Equity Risk Premiums using Machine Learning

Jeffrey Zwart (498269)

July 3, 2021

Supervisor: T. van der Zwan

Second assessor: M. van der Wel

Abstract

This paper uses machine learning methods to predict the level and direction of monthly U.S. equity risk premiums. For directional predictions, a multinomial approach is used to put more emphasis on large absolute returns. It is found that there exist large economic gains for investors who use a long-short trading strategy based on machine learning predictions for the level and direction of excess returns. In particular, a long-short strategy using neural network forecasts for the level of excess returns outperforms the benchmark buy-and-hold strategy of the S&P 500 by a large margin. Although a similar long-short strategy that uses directional forecasts improves upon this benchmark, there is no indication that directional forecasts can improve upon the level forecasts in an economic sense.

The views stated in this thesis are those of the author and not necessarily those of the supervisor, second assessor, Erasmus School of Economics or Erasmus University Rotterdam

Contents

1 Introduction

Forecasting equity risk premiums is a well known problem in the financial literature (see, e.g., Welch and Goyal (2008) and Rapach, Strauss, and Zhou (2010)). Since the topic is of great importance to many groups in the investment world, a large number of methods and models have been developed with the objective of making good forecasts, and more importantly achieving good returns. Enormous improvements in computing power and lower data-storage costs have contributed to the popularity of machine learning methods in many different fields, including the field of stock return prediction, see Weigand (2019). There are two reasons that make machine learning methods suitable for predicting equity risk premiums, according to Gu, Kelly, and Xiu (2020). First, machine learning methods are often specialized for prediction tasks, focusing less on interpreting the relationships in the model. Since the underlying return process is highly complex and ambiguous, it is difficult to find an interpretable model for it. Second, researchers have put forward a substantial number of variables with forecasting power for equity risk premiums (see Welch and Goyal (2008)), and these variables can be highly correlated. Classical methods such as Ordinary Least Squares (OLS) do not perform well in such setting, see Gunst and Webster (1975). Machine learning methods on the other hand provide ways to overcome these problems, for example by using dimensionality reduction and variable selection techniques.

Gu et al. (2020) use a variety of machine learning methods to forecast equity risk premiums, and they find that they dominate classical regression-based models, underlining the usefulness of machine learning in finance. In particular, they find that neural networks and tree methods are the best predictive models. They show that there are large economic gains for investors using machine learning forecasts, in the form of better Sharpe ratios.

While predicting the level of equity risk premiums is a popular research area, there is also a great amount of literature on predicting the direction of equity risk premiums, see for example Karhunen (2019) and Nyberg (2011). Leitch and Tanner (1991) argue that the direction of stock returns is the best criterion for predictability because this criterion is more correlated with the profits investors are seeking than standard statistical error measures. In addition, Leung, Daouk, and Chen (2000) conclude that classification models outperform level estimation models in terms of predicting the direction of returns and maximizing returns from trading.

In this research, I use the linear regression, random forest, and artificial neural network models of Gu et al. (2020) to predict the level of equity risk premiums in the U.S. during the period 2002-2016. In addition to this, I predict the direction of the equity risk premiums. Similar to many machine learning models, random forests and neural networks can be used

for both regression and classification tasks. Afterward, I compare the performance of level and directional forecasting. Therefore, the main contribution of this research to the literature is the large-scale comparison of level and directional forecasting of equity risk premiums of individual stocks using machine learning methods.

I find, using a large dataset on monthly U.S. equities, that large economic gains can be obtained by using level and directional forecasts of excess returns from machine learning methods. In particular, when the level and directional forecasts from neural networks are used in a long-short trading strategy, an investor can generate financial returns that are far larger than those obtained via a buy-and-hold strategy of the S&P 500 Index. Although using directional forecasts in a trading strategy is found to be profitable, the returns are even greater when using level forecasts. Hence, the benefits of directional forecasts over level forecasts are found to be limited.

2 Literature

Machine learning methods have many applications within the field of finance. For example, Hutchinson, Lo, and Poggio (1994), among others, use neural networks to predict derivatives prices. Butaru et al. (2016) use classification methods to predict credit card defaults. The area of stock returns also receives much attention from researchers, possibly in part due to its potentially lucrative nature. For example, Rapach, Strauss, and Zhou (2013) use lasso regressions to predict global equity market returns, and Abe and Nakayama (2018) make use of deep learning to predict stock returns in the cross-section.

Many studies have demonstrated that the direction of stock returns is predictable, see for instance Christoffersen and Diebold (2006) and Hong and Chung (2003). There are several approaches for directional forecasting. The most straightforward approach is to use sign prediction, which is a binary classification problem. In that case, the problem is to predict whether the stock return is positive or negative. Examples that follow this approach are Nyberg (2011), Fiévet and Sornette (2018), and Karhunen (2019). In line with other research, Hong and Chung (2003) find evidence that the direction of equity risk premiums is predictable, but they conclude that the evidence is strongest for the large absolute equity risk premiums.

This leads to directional forecasting using multiple thresholds, such that the problem becomes a multinomial classification problem. For example, one can set two thresholds in such a way that there are three classes that represent large negative returns, small absolute returns, and large positive returns. This multinomial approach could be more effective for stock returns than sign prediction. The reason for this is that stock returns data are extremely noisy by nature. The noise stems from the fact that if returns were easily predictable the market

would have moved the prices already. This noisiness of the data results in a low signal-to-noise ratio. Indeed, Chung and Hong (2007) argue that small absolute returns are simply noise and that the signals of large absolute returns are more valuable. Hence, according to Nevasalmi (2020), directional forecasting with multiple classes could isolate this noise, and put more emphasis on large absolute returns. In addition, Nevasalmi (2020) argues that another benefit of the multinomial classification approach is that it allows for a richer set of trading strategies. For example, instead of only the buy and sell signals that are obtained via sign prediction, one can obtain buy, neutral, and sell signals. The extra signals can also be important for market timing, which is an important asset allocation problem. Therefore, I use the multinomial instead of the binary approach.

3 Data

This research uses the same data as Gu et al. (2020). This data is from March 1957 until December 2016 and consists of monthly observations. It includes 94 stock-level predictors similar to those detailed in Green, Hand, and Zhang (2017), and 74 dummy variables corresponding to the first two digits of the Standard Industrial Classification (SIC) codes.[1] I match this data with monthly data on 8 macroeconomic variables of Welch and Goyal (2008).[2] Appendix F of Gu et al. (2020) contains more information on each of the used variables.

Total monthly individual equity returns are obtained from the Center for Research in Security Prices (CRSP) for all firms listed on the NYSE, AMEX, and NASDAQ, and these are then matched with the stock-level and macroeconomic predictors. Similar to Gu et al. (2020), I use the 3-Month Treasury-bill rate as a proxy for the risk-free rate to calculate individual excess returns.[3]

I follow the approach of Gu et al. (2020) by imputing the cross-sectional median of each characteristic at each month t for each stock. The characteristics *secured* and *realestate* are not available before 1985, and therefore I delete these two characteristics from the data set. Removing them is justified because Gu et al. (2020) find that they have low importance in each model. In addition, I remove the Standard Industrial Classification (SIC) variables, given their modest importance in combination with very high memory usage. Further, I delete the rows that contain missing values even after imputing the cross-sectional median, as well as the rows that do not contain a valid return value. This step removes all data before October

[1] Available from *https://dachxiu.chicagobooth.edu/*.

[2] Available from *http://www.hec.unil.ch/agoyal/*.

[3] Accidentally, in this paper I use a risk-free rate that is slightly higher than the actual risk-free rate based on the 3-Month Treasury-bill. Therefore, there is a slight bias in the excess returns. Later in this paper, when evaluating the buy-and-hold strategy of the S&P 500 Index, the correct risk-free rate is used.

1974. After all these steps, there are 3,271,838 observations left. Finally, the characteristics are cross-sectionally ranked into the [-1, 1] interval, because the relative value of a characteristic is likely to be more informative than its absolute value.

Using the notation of Gu et al. (2020), the set of predictors for the return of stock i at time $t+1$ is given by $z_{i,t}$. It contains the 92 stock-level characteristics and a number of stock-macro interaction terms. Gu et al. (2020) find that in general, there are certain categories of stock characteristics that have much predictive power. The most important characteristics are those based on price trends and liquidity. For computational and memory reasons, I only include interactions for these categories of variables. The specific list of 14 characteristics that are selected can be found in Table 6 in the Appendix. For each selected characteristic, there are 8 interaction terms since there are 8 macroeconomic variables. Thus, $z_{i,t}$ is a vector with $92 + 14 \times 8 = 204$ elements.

4 Methodology

In this section, I describe the training procedures, methods, and evaluation techniques used throughout the rest of this text. I select a subset of the methods used in Gu et al. (2020) based on their simplicity, computational time, or predictive performance. Specifically, I select the linear model using three predictor variables (named OLS-3+H in Gu et al. (2020)) because of its simplicity and reasonable performance. In addition, I select the random forest model (RF) and neural networks (NN1-NN5) based on their good performance.[4]

For predicting the level of equity risk premiums, I use the additive prediction error model for an asset's excess return $r_{i,t+1}$, which reads

$$r_{i,t+1} = E_t[r_{i,t+1}] + \epsilon_{i,t+1}, \tag{1}$$

where $E_t[r_{i,t+1}] = g^*(z_{i,t})$. $i = 1, \ldots, N_t$ is the index of a stock at time t, and months are indexed by $t = 1, \ldots, T$. Note that the number of stocks differs across months, as private companies can go public for example. Our goal is to find the function $g^*(\cdot)$ that, given a set of predictor variables $\mathrm{z}_{i,t}$, maximizes the explanatory power of realized $r_{i,t+1}$ in an out-of-sample setting.

It is important to note that the function $g^*(\cdot)$ does not depend on time or individual stocks. Hence, the model uses information from the entire panel and can be used for predicting individual risk premiums.

For directional forecasting of equity risk premiums, the equity risk premiums should first

[4]The reason for not selecting the boosted trees model is that it is unable to make extensive use of parallel computing due to the way this model is constructed, which results in slow training.

be labeled. In this research, I use three classes, and by using the notation of Nevasalmi (2020) the response variables are obtained as

$$R_{t,i}(c_1, c_2) = \begin{cases} 1 & \text{if } r_{i,t} < c_1 \\ 2 & \text{if } c_1 \leq r_{i,t} \leq c_2 \\ 3 & \text{if } r_{i,t} > c_2 \end{cases}. \tag{2}$$

The question is now how to set the thresholds c_1 and c_2. Chung and Hong (2007) suggest that the thresholds could be based on trading costs, whereas Linton and Whang (2007) use a variety of return quantiles. Following Nevasalmi (2020), I use the lower and upper quartiles of the return series for c_1 and c_2, respectively. This labeling approach prevents overly unbalanced classes because the minimum amount of observations that are in a class is about 25% of the total. I refer to Ganganwar (2012) for more details on the problems of unbalanced data. Using these thresholds, classes 1, 2, and 3 naturally correspond to sell, neutral, and buy recommendations.

This section is organized as follows. First, in section 4.1 I outline the procedure for training, validating, and testing each model. Sections 4.2, 4.3, and 4.4 describe the selected models via their functional form, objective function and optimization procedure. Section 4.5 shows the evaluation methods that I use. Lastly, section 4.6 discusses methods to examine the variable importance in the random forest and neural network models.

4.1 Training, validation, and testing

In order to make conclusions about the predictive performance of methods, the methods should be tested in a realistic environment. Therefore, I split the data into a training, validation, and test set. The train, validation, and test set are from 1974-1991, 1992-2001, and 2002-2016, respectively. This particular split ensures that like in Gu et al. (2020), there is 18 years of training data and that the test sample is sufficiently long. I refit the models once every year, and extend the training sample by 1 year, keeping the size of the validation sample fixed but shifted 1 year forward. This means that I make out-of-sample predictions with a trained model over the subsequent 12 months, before refitting it.

The validation samples are necessary because many machine learning models are prone to overfitting. Overfitting means that a model makes good predictions on the training data, but that it generalizes badly to out-of-sample data. The goal is to find a model that gives quality predictions in an out-of-sample setting. Parameters that control the learning process are called hyperparameters and selecting the right hyperparameters can prevent model overfitting. One should search for the set of hyperparameters that produces the best model for out-of-

sample predictions. Therefore, the hyperparameters are chosen adaptively using the validation set. Using the best set of hyperparameters, the model is trained on the training set, after which it can be used to make predictions on the test set. This procedure ensures that the predictions are made on data that has not been used in the model selection and training process. The hyperparameter tuning schemes are specified in Table 7 in the Appendix.

4.2 OLS-3+H

The first model I describe is the linear model with only three predictor variables (OLS-3+H). These variables are size (mvel1), book-to-market (bm), and momentum (mom12m). This model, which has been proposed by Lewellen (2014), is parsimonious and acts as a benchmark for more complex models. It is important to note that this model can only be used to predict the level of the equity risk premiums, and is not suitable to perform classification tasks.

This model assumes that the function $g^*(\cdot)$ is linear in the three predictors, such that

$$g^*(z_{i,t};\theta) = z'_{i,t}\theta, \tag{3}$$

where $z_{i,t}$ in this case only contains the three predictor variables, and θ is a parameter vector.

The classic objective function of the linear model is the "l2" objective function, which in the current context can be written as

$$\mathcal{L}(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(r_{i,t+1} - g^*(z_{i,t};\theta))^2, \tag{4}$$

where N and T denote the number of stocks in each month and the number of months respectively. To simplify notation, this notation assumes a balanced panel of stocks. The pooled OLS estimator is obtained by minimizing this objective function. However, one of the stylized facts of stock returns is the presence of fat tails, see Cont (2001). The quadratic nature of equation (4) makes that the resulting parameters are heavily affected by outliers, deteriorating the forecasting performance. One way to overcome this problem is to use a robust loss function such as the one proposed by Huber (1992). This loss function is more robust to outliers because it uses a quadratic loss for small errors, and a linear loss for larger errors, such that (4) can be rewritten as

$$\mathcal{L}_{\mathcal{H}}(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}H(r_{i,t+1} - g^*(z_{i,t};\theta),\xi)^2, \tag{5}$$

with

$$H(x;\xi) = \begin{cases} x^2 & \text{if } |x| \leq \xi \\ 2\xi|x| - \xi^2 & \text{if } |x| > \xi \end{cases}. \tag{6}$$

4.3 Random Forests

A very different type of model is the tree, which is used extensively in machine learning literature. This model is particularly useful for incorporating nonlinear interaction terms of the covariates. First, I discuss regression trees, which are suitable for predicting the level of equity risk premiums. For directional forecasting, one should use classification trees, which are similar to regression trees and are also discussed.

A regression tree is built using the iterative CART algorithm of Breiman, Friedman, Stone, and Olshen (1984). First, at the root of the tree, the data is split into two subsets using a single predictor and some threshold. The predictor and the threshold are chosen in such a way that the resulting subsets (nodes) have the lowest weighted Mean Squared Error (MSE), where the weights are based on the number of instances in each subset. Then, the subsets are split using the same logic, and this process continues in a recursive fashion. This process stops when, for example, the maximum depth of the tree is reached, which has to be specified ex-ante. The resulting tree approximates the function $g^*(\cdot)$ with the mean value of the target variable within each leaf node.

Mathematically, using similar notation as Gu et al. (2020), the forecast of a regression tree with K leaf nodes and depth L can be written as

$$g(z_{i,t};\theta,K,L)=\sum_{k=1}^{K}\theta_k\mathbf{1}_{\{z_{i,t}\in C_k(L)\}}, \tag{7}$$

where $C_k(L)$ is the kth partition of the data, and θ_k is the mean value of the target variable within that partition.

Although the process of building a classification tree is similar to that of a regression tree, there a few differences. First, the objective at each data split is different due to the absence of a continuous target variable. Instead of minimizing the weighted MSE, the cost function becomes the weighted Gini impurity. Gini impurity is defined as

$$G_i=1-\sum_{k=1}^{n}p_{i,k}^2, \tag{8}$$

where $p_{i,k}$ is the ratio of instances belonging to class k to the total number of training instances in node i. Hence, each split ensures that the two resulting nodes are as pure as possible. Second, forecasts are made in the same way as in equation (7), but θ_k becomes the mode class of that particular partition.

Although trees have some desirable properties such as their straightforward interpretation, the models are highly prone to overfitting due to their extreme flexibility. Therefore, it is more common to use trees in ensemble models, which are types of models that combine

multiple individual models to create a better model than the individual ones separately. In particular, I consider the random forest model, proposed by Breiman (2001), which combines predictions from a large number of weakly correlated trees into a single prediction.

The random forest model uses a procedure called bagging, which consists of drawing many different bootstrap samples from the data, fitting individual trees to each bootstrap sample, and combining individual tree predictions. The bootstrap procedure forces each sample to be independent of the others. The core idea is that the individual trees are prone to overfitting and that aggregating the predictions of multiple independent and weakly correlated trees reduces the model variance. The weak correlation between the individual trees is achieved by only selecting a random subset of the predictors at each split, which forces some trees to make the first few splits on predictors that are not the most informative. This results in a lower correlation among the trees, reducing the variance in the model. The prediction of the random forest for regression is the mean of the individual regression tree predictions. For classification, the prediction is based on the majority vote of the individual trees. Figure 1 visualizes a random forest with only 4 individual trees.

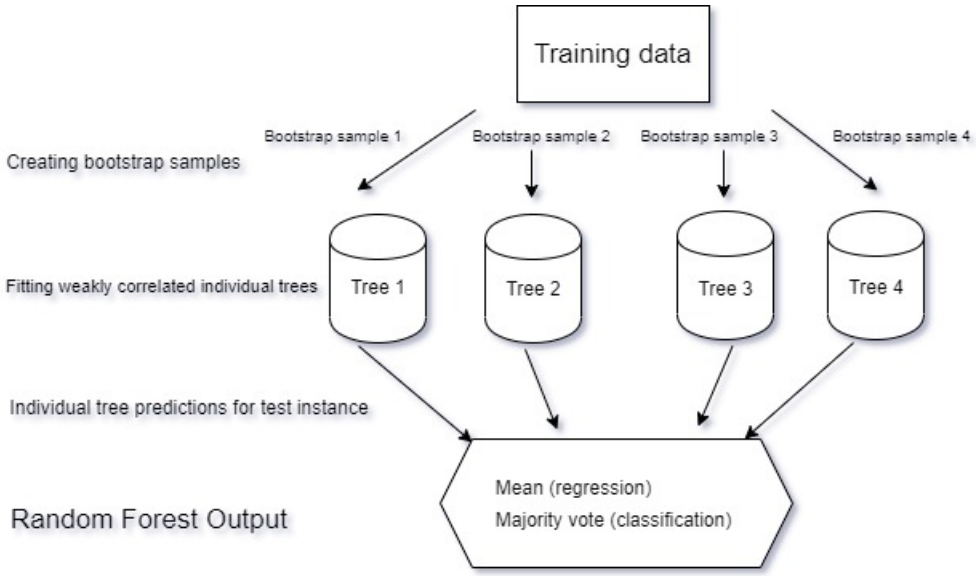


Figure 1: Visualization of a random forest model with only 4 individual trees.

The most important hyperparameters for training a random forest are the depth of the trees, the number of randomly chosen predictors at each split, and the number of individual trees. The individual trees of a random forest can be fitted in parallel, which greatly benefits the training time.

4.4 Neural Networks

Another nonlinear method is the artificial neural network, which is heavily used in practice. It is especially used to model complex nonlinear relationships, which are found in fields such as computer vision and voice recognition. In addition, it can be used for both regression and classification tasks. One disadvantage of a neural network is that it is a 'black-box' method. That is, it is hard to interpret the relationships in the approximated function $g^*(\cdot)$.

In this research, I use basic feed-forward neural networks, which consist of an input layer, one or multiple hidden layers, and one output layer. Figure 2 visualizes an example architecture of a neural network with one hidden layer. The network is fully connected, meaning that each of the nodes in one layer is connected to all nodes in the following layer. In this example, the input layer takes 5 inputs and it has a hidden layer of 3 nodes. The edges between the nodes represent the weight parameters.

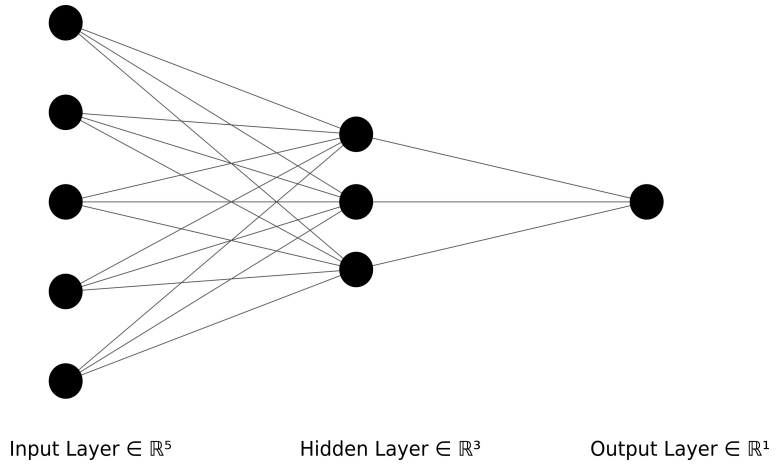


Figure 2: Artificial neural network with 1 hidden layer.

The inputs for the nodes in the hidden and output layers are linear combinations of the node outputs of the previous layer, including a constant term. The hidden layer nodes use what is called an activation function to transform their input into an output. This output is then passed forward to the next layer. For regression tasks, the output layer contains one single neuron which contains the predicted value. For directional forecasting with three classes, there are three output neurons. It is common to use the softmax function on the whole output layer to obtain probabilistic predictions for each of the three classes. Specifically, let u_1, u_2, u_3 be the inputs for the first, second, and third output neuron, respectively. Then, the final probabilistic outputs u_1', u_2', u_3' using the softmax function are given by

$$u_i' = \frac{e^{u_i}}{\sum_{k=1}^{3} e^{u_k}} \quad \text{for} \quad i = 1, 2, 3. \tag{9}$$

The predicted class is then the class corresponding to the greatest probability.

A popular activation function is the rectified linear unit (ReLU), which is defined as

$$ReLU(x) = \max(0, x). \tag{10}$$

This activation function speeds up the learning process as some nodes will be inactive. Following Gu et al. (2020), I use this activation function for each of the nodes in the hidden layers.

Finding the best network architecture for a specific task is not feasible for computational reasons. Hence, Gu et al. (2020) consider only five network architectures. The first architecture has 1 hidden layer, with 32 hidden nodes (NN1). The second has 2 hidden layers, with 32 and 16 nodes, respectively (NN2). The third has 3 hidden layers, with 32, 16, and 8 nodes, respectively (NN3). The fourth has 4 hidden layers, with 32, 16, 8, and 4 nodes, respectively (NN4). Finally, the fifth contains 5 hidden layers with 32, 16, 8, 4, and 2 nodes, respectively (NN5). The choice for the number of nodes in each hidden layer is based on the geometric pyramid scheme of Masters (1993).

In the context of predicting the level of equity risk premiums, the weight parameters are estimated by minimizing the MSE using Stochastic Gradient Descent (SGD) and by using the backpropagation algorithm of Rumelhart, Hinton, and Williams (1986). An in-depth discussion of the backpropagation algorithm is beyond the scope of this paper. In essence, the backpropagation efficiently computes the gradient of the loss function with respect to all the weight parameters, such that the weight parameters can be updated after each iteration. For directional forecasting, I use one-hot encoding for the target variable. The weight parameters are then estimated by minimizing the cross-entropy, which in this paper is defined as

$$L = -\sum_{j=1}^{N}\sum_{k=1}^{3} y_{j,k} \cdot \ln(u'_{j,k}), \tag{11}$$

where $y_{j,k}$ is 1 if the observation j of the current batch belongs to class k, and 0 else.

The training consists of multiple epochs, which is the number of times the entire training set passes through the network for training. For each epoch, the training set is split into batches, and passing one batch through the network is called an iteration. Following Gu et al. (2020), I use 100 epochs and a batch size of 10000. Hence, N in equation (11) equals 10000.

The large number of parameters and the high nonlinearity make neural networks prone to overfitting. Therefore, in line with Gu et al. (2020), I use learning rate shrinkage using the algorithm of Kingma and Ba (2014), early stopping, and batch normalization.[5]

[5]Batch normalization standardizes each layers' inputs for each batch. This method speeds up the learning process and can help stabilize the learning process. I refer to Ioffe and Szegedy (2015) for a more detailed discussion on batch normalization.

4.5 Forecast evaluation

4.5.1 Statistical performance

Level and directional predictions require different evaluation measures. First, I describe the measures that are solely used for level forecasts. Following Gu et al. (2020), I use the out-of-sample R^2, denoted as R^2_{oos}, as a simple metric to assess the predictive performance of the methods. This metric is defined as

$$R^2_{oos} = 1 - \frac{\sum_{(i,t)\in test}(r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{(i,t)\in test} r^2_{i,t+1}}, \tag{12}$$

such that only the observations in the test sample are used. Note that it pools the predictive performance across firms and time into one global assessment of each model. The predictions are compared to a naive forests of zero, as this gives more realistic results in this setting, see Gu et al. (2020).

In addition, I use the Diebold-Mariano (DM) test of Diebold and Mariano (2002) to compare the predictive accuracy of two competing models. Gu et al. (2020) point out that that there could be strong dependence of prediction errors in the cross-section. They argue that a comparison of cross-sectional means of prediction errors from each model overcomes this problem. The test statistic is defined as $DM_{12} = \bar{d}_{12}/\hat{\sigma}_{\bar{d}_{12}}$, where

$$d_{12,t+1} = \frac{1}{n_{3,t+1}} \sum_{i=1}^{n_{3,t+1}} \left(\left(\hat{e}^{(1)}_{i,t+1}\right)^2 - \left(\hat{e}^{(2)}_{i,t+1}\right)^2 \right). \tag{13}$$

Note that the prediction errors of model 1 and 2 for stock i at time t are given by $e^{(1)}_{i,t+1}$ and $e^{(2)}_{i,t+1}$, respectively. The number of stocks at time $t+1$ in the testing sample is denoted by $n_{3,t+1}$.

In this paper, the performance measure for directional forecasts is classification accuracy. This is the most common measure for the evaluation of classification models. Classification accuracy is defined as the proportion of correctly classified observations of the total number of observations.

4.5.2 Economical performance

Although statistical performance measures provide a rough indication of the forecasting quality of a model, Leitch and Tanner (1991) argue that good statistical performances do not automatically result in economic profits. Hence, to assess the economical performance of the level and directional predictions, I consider trading strategies.

For the level predictions, I create an equally weighted long-short decile spread strategy similar to Gu et al. (2020). For the directional forecasts, I use a similar strategy based on the

sell, neutral, and buy recommendations. Specifically, the strategy sells the stocks that are predicted class 1 and buys the stocks that are predicted class 3. Therefore, class 2 can intuitively be viewed as a 'filter' of noise, and the predictions belonging to this class are neglected. This zero-net-investment strategy uses equal weights within the long and short sides of the strategy and will be referred to as the class spread strategy. Each strategy is compared to a buy-and-hold strategy of the S&P 500 Index using its absolute return, annual Sharpe ratio, and monthly average turnover.

The average monthly turnover quantifies how much portfolio rebalancing is required in a strategy. I use the turnover measure of Gu et al. (2020), which is defined as

$$\text{turnover} = \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i}\left|w_{i,t+1} - \frac{w_{i,t}(1+r_{i,t+1})}{1+\sum_{j} w_{j,t}r_{j,t+1}}\right|\right). \tag{14}$$

In this notation, $w_{i,t}$ is defined as the weight in the portfolio of stock i at month t.

4.6 Variable importance

I use the same notion of variable importance as Gu et al. (2020) for predicting the level of equity risk premiums. That is, I look at the reductions of in-sample R^2 from setting one covariate to zero while using the best-estimated model for every training sample. The predictors that correspond with large reductions in R^2 are considered to be of great importance for a model.

5 Results

5.1 Monthly returns

Table 1 and Figure 3 show the out-of-sample performance of each of the models in terms of R^2_{oos}. It appears that all models except RF have little predictive power and do not produce predictions that are better than constant predictions of zero, contradicting the results of Gu et al. (2020). Only RF produces a substantial R^2_{oos} of 0.65% on the entire sample of stocks. In addition, the models do not have a particularly strong performance on the subsample of large stocks, which differs from Gu et al. (2020). In fact, the performance of some models is better on the subsample of small stocks. NN1 is the best performing neural network, though it has a very modest R^2_{oos} of 0.03%. Adding more hidden layers does often not lead to better performance, which suggests that there are few benefits to deep learning.

Table 1: Monthly out-of-sample forecasting performance in terms of percentage R^2_{oos}.

	OLS3+H	RF	NN1	NN2	NN3	NN4	NN5
All	-0.17	0.65	0.03	-1.18	-1.92	-0.78	-1.02
Top 1000	-0.15	-1.47	-1.60	-2.57	-1.05	-1.23	-0.73
Bottom 1000	-0.21	0.67	0.39	-1.01	-3.27	-1.04	-1.19

This table reports the one-month-ahead predictive performance of 7 different models using R^2_{oos} in percentage terms for the full panel of stocks from 2002 - 2016. In addition, it reports R^2_{oos} values for subsamples of large and small stocks, in which only the largest and smallest 1000 stocks per month are included, respectively.

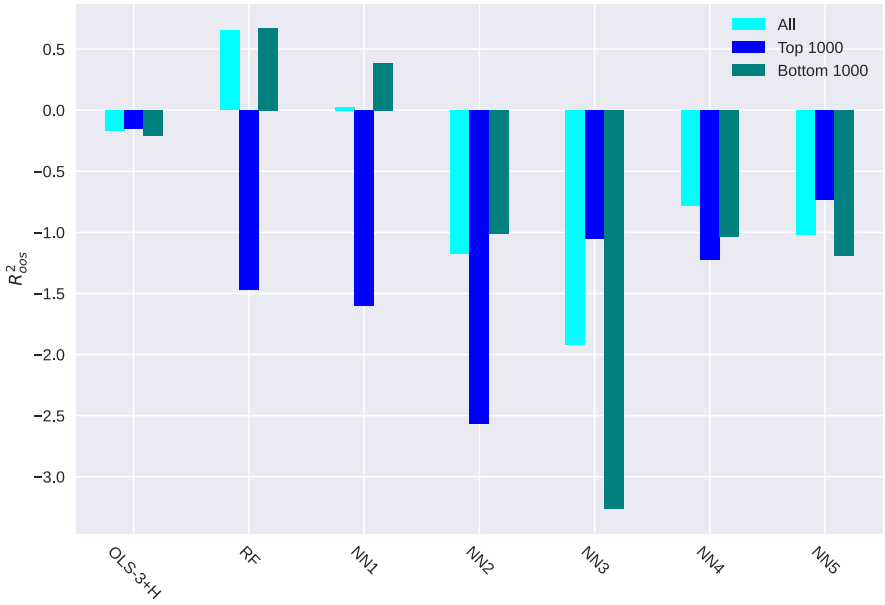


Figure 3: Monthly out-of-sample forecasting performance in terms of percentage R^2_{oos}.

There are a few factors that could have caused the above results to be substantially different from those in Gu et al. (2020). First, I use less than a quarter of their number of predictors, which might have resulted in fewer signals and hence lower predictive power. Second, I use a smaller dataset and a different and shorter test period. Third, for RF I used only a subset of their hyperparameter values for computational reasons.

Table 2 shows pairwise Diebold-Mariano (DM) test statistics to compare the forecasting accuracy between the models. It can be seen that the forecasts of OLS-3+H are significantly more accurate than those of NN2 and NN3. RF outperforms OLS-3+H and all neural networks, though the result is only statistically significant for NN3. Comparing the neural networks, we see that NN1 outperforms the others in a statistically significant way. In sum, we find that RF performs best, followed by NN1 and the parsimonious OLS-3+H.

Table 2: Prediction comparison using Diebold-Mariano tests.

	RF	NN1	NN2	NN3	NN4	NN5
OLS-3+H	0.52	0.32	-1.75*	-2.89*	-1.15	-1.58
RF		-0.34	-1.13	-1.69*	-0.88	-1.04
NN1			-3.29*	-3.77*	-2.60*	-2.97*
NN2				-1.46	1.20	0.47
NN3					3.66*	2.92*
NN4						-1.32

This table contains Diebold-Mariano (DM) test statistics for the one-month-ahead predictions of 7 different models. Positive (negative) values indicate that the column (row) model outperforms the row (column) model. To test for significance, I use a one-sided test in the direction of the sign of the test statistic. Significance at the 5% level is indicated by an asterisk.

Figure 4 shows the twenty most important variables within each nonlinear model based on reductions in R^2. For RF, the most important variable is short-term reversal (mom1m) which is in line with Gu et al. (2020). However, Gu et al. (2020) found that short-term reversal was also the most important covariate in neural networks, but I find different results. In particular, it appears that liquidity variables such as market value (mvel1_norm) and dollar volume (dolvol) are at least as important as momentum variables for neural networks. Figure 7 in the Appendix shows the overall variable importance ranking in the machine learning models. The predictors are ordered based on the sum of ranks across models. The color mapping within each column is such that dark colors correspond to important predictors and light colors correspond to less important predictors. Overall, market value, short-term reversal, and dollar volume are the three most important predictors in the nonlinear models.[6]

5.2 Annual returns

Table 3 and Figure 5 display the predictive performance at the annual horizon. To go from the monthly to the annual horizon, the monthly returns are grouped by company and year, then annualized and finally converted to obtain a 'mean' monthly return for a particular company in a particular year. The same procedure is used for the monthly predictions, and using equation (12) we can again assess forecasting performance, but this time at the annual horizon. It can be seen that the annual results are in line with the monthly. That is, RF is still the best performing model, and the predictive power, in general, is low. These results show

[6]Note that the characteristics that contain interaction terms have the highest overall importance because each of these characteristics corresponds to 9 predictors instead of 1. Hence, it is not valid to compare the importance of one characteristic that includes interaction terms to the importance of one that does not.

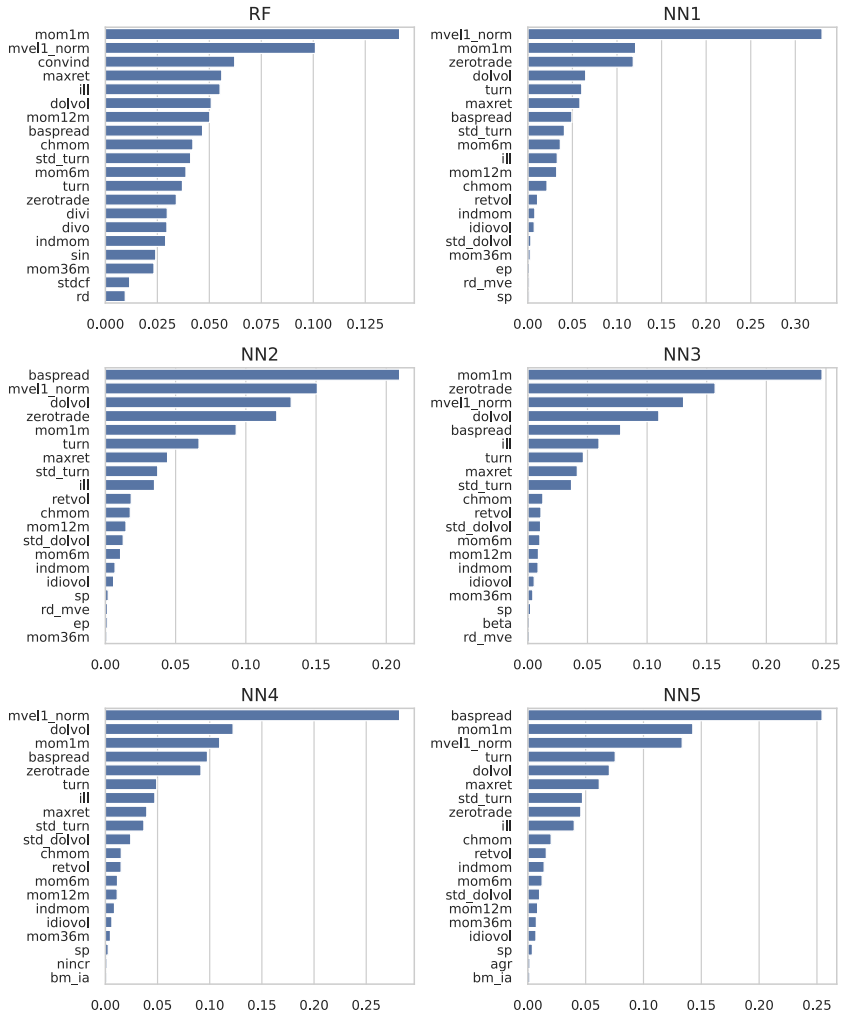


Figure 4: Variable importance by model. It is the average reduction in R^2 over all training samples, and the weights within each model are normalized such that the sum is 1.

that the machine learning models, besides having difficulty exploiting short-term inefficiencies, have difficulty predicting risk premiums over different business cycles.

Table 3: Annual out-of-sample forecasting performance in terms of percentage R^2_{oos}.

	OLS-3+H	RF	NN1	NN2	NN3	NN4	NN5
All	-1.36	2.80	-14.45	-22.97	-27.67	-22.22	-21.68
Top 1000	-5.67	-0.99	-2.11	-14.35	-4.74	-4.35	-0.27
Bottom 1000	0.57	-1.31	-31.08	-41.49	-59.08	-48.44	-44.25

This table reports the one-year-ahead predictive performance of 7 different models using R^2_{oos} in percentage terms for the full panel of stocks from 2002 - 2016. In addition, it reports R^2_{oos} values for subsamples of large and small stocks, in which only the largest and smallest 1000 stocks per year are included, respectively. The subsamples are constructed using the mean monthly market value of a company.

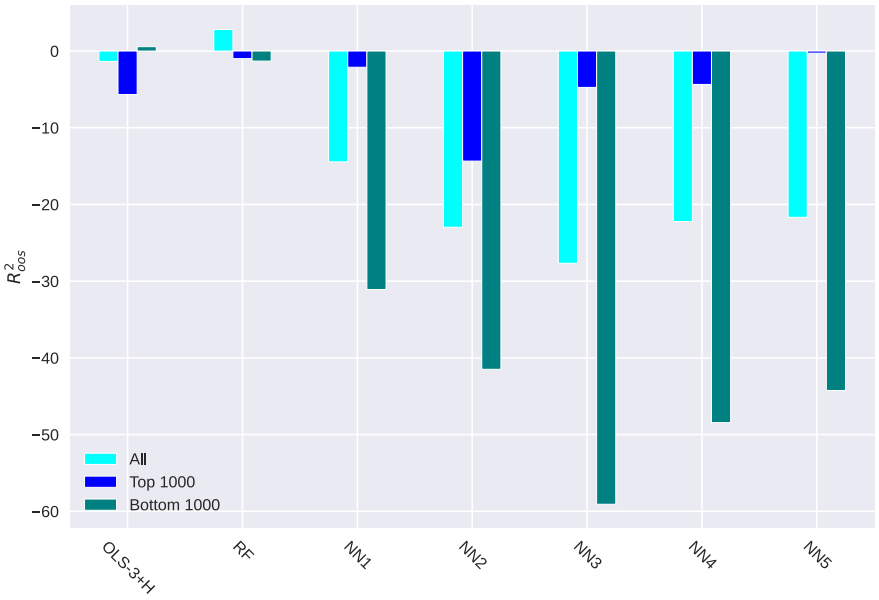


Figure 5: Annual out-of-sample forecasting performance in terms of percentage R^2_{oos}.

5.3 Directional forecasting

To avoid forward-looking bias, the first and third quartiles of the excess returns are determined using only the first training set (October 1974 - December 1991).[7] The class distribution in the test set is different from the one in the first training set. Specifically, more than half of the observations of the test set fall into the majority class. A naive strategy of always predicting the majority class thus results in an accuracy of 54.4%, which acts as a benchmark.[8]

Table 4 reports the classification accuracy of each model. For the whole sample of stocks, the models slightly improve over the benchmark of 54.4%. In particular, NN5 produces the best accuracy of 55.04%. In the subsample of large stocks, the naive strategy is difficult to beat. However, RF and NN5 are the best in terms of accuracy. The accuracy figures for the subsample of small stocks are substantially higher than the benchmark of 48.25%, with RF and NN5 producing the highest accuracy.

Table 8 in the Appendix reports the confusion matrices of the models. It can be seen that RF never predicts class 3. Thus, RF has difficulty extracting signals that produce high returns. Although the classification accuracy of this model is relatively high, it is unlikely that this strategy is economically profitable, because it never produces buy recommendations. This result, therefore, supports the argument of Leitch and Tanner (1991) that good statistical per-

[7]The first and third quartile are -8.77% and 4.78%, respectively.

[8]The accuracy benchmarks for the subsamples of large and small stocks are 60.8% and 48.25%, respectively.

Table 4: Classification accuracy.

	RF	NN1	NN2	NN3	NN4	NN5
All	0.5550	0.5478	0.5462	0.5473	0.5453	0.5504
Top 1000	0.6097	0.6024	0.6007	0.6054	0.6018	0.6079
Bottom 1000	0.5151	0.5042	0.5014	0.5011	0.4996	0.5061

This table reports the classification accuracy of 6 different models for the full panel of stocks from 2002 - 2016. In addition, it reports the accuracy for subsamples of large and small stocks, in which only the largest and smallest 1000 stocks per month are included, respectively.

formance alone is not sufficient to generate economic profits. Economically, the costliest mistakes are to predict class 1 if the actual class is 3, or 3 if the actual class is 1. In those cases, the strategy goes long (short) a stock that provides a negative (positive) return. Although the number of these mistakes differs across models, it is difficult to determine what the exact effect of this is on the performance of each model in a trading strategy.

5.4 Trading simulations

Table 5 reports the performance measures of the decile and class spread strategies. The decile spread strategy shows strong performance, especially in combination with the forecasts of the neural network models. NN4 yields the highest annualized return and Sharpe ratio, which are 37.17% and 2.46 respectively. The benchmark buy-and-hold strategy of the S&P 500 Index produces an annualized return of 3.3% and a Sharpe ratio of 0.38. Hence, this benchmark strategy is easily outperformed by both trading strategies. There are a few models that have weak performance. First, the OLS-3+H model produces an annualized return of -0.25%. Second, RF in combination with the class spread strategy produces negative returns, which is likely to be a result of the fact that it never produces buy recommendations. Also note that the statistical performance of RF was strong for level forecasts, but that it is economically outperformed by other models that had a weaker statistical performance.

Trading costs are an important element of a strategy, and therefore it is important to look at the average monthly turnover. From Table 5 it can be seen that the average monthly turnover using a decile spread strategy is more than 200%. Using the class spread strategy, the turnover is approximately 30 percentage points lower for each model. The reason for the relatively low turnover of the class spread strategy using RF is that it goes long the risk-free rate instead of class 3 stocks, which limits the amount of rebalancing.

Becker and Leschinski (2018) find that the average bid-ask spread for U.S. individual stocks is around 0.05% during the period 2004 - 2017. Therefore, even after accounting for the high

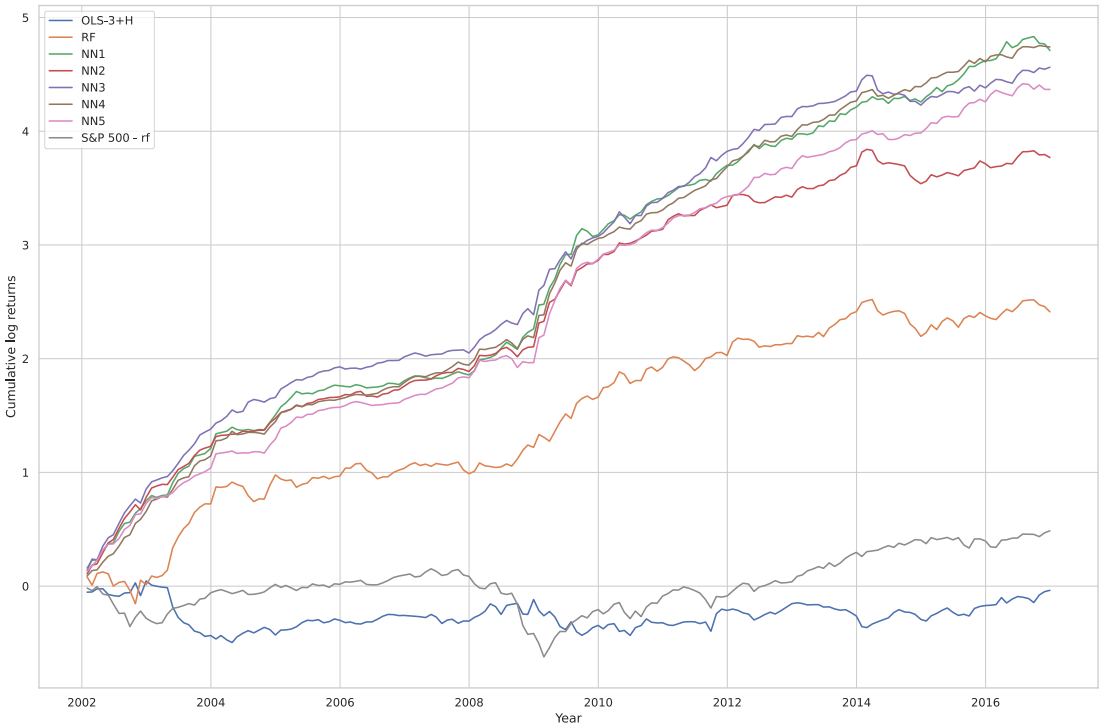
turnover rates and thus significant trading costs, both the decile spread and the class spread strategies produce returns that are greater than those corresponding to a buy-and-hold strategy of the S&P 500 Index. Comparing the decile spread strategy with the class spread strategy, I conclude that the decile spread strategy outperforms the class spread strategy in terms of absolute returns and risk-reward. Although the decile spread strategy has a higher turnover ratio and thus higher trading costs than the class spread strategy, this is easily offset by its much higher annualized return.

Table 5: Performance measures of top-bottom decile spread strategy and class spread strategy

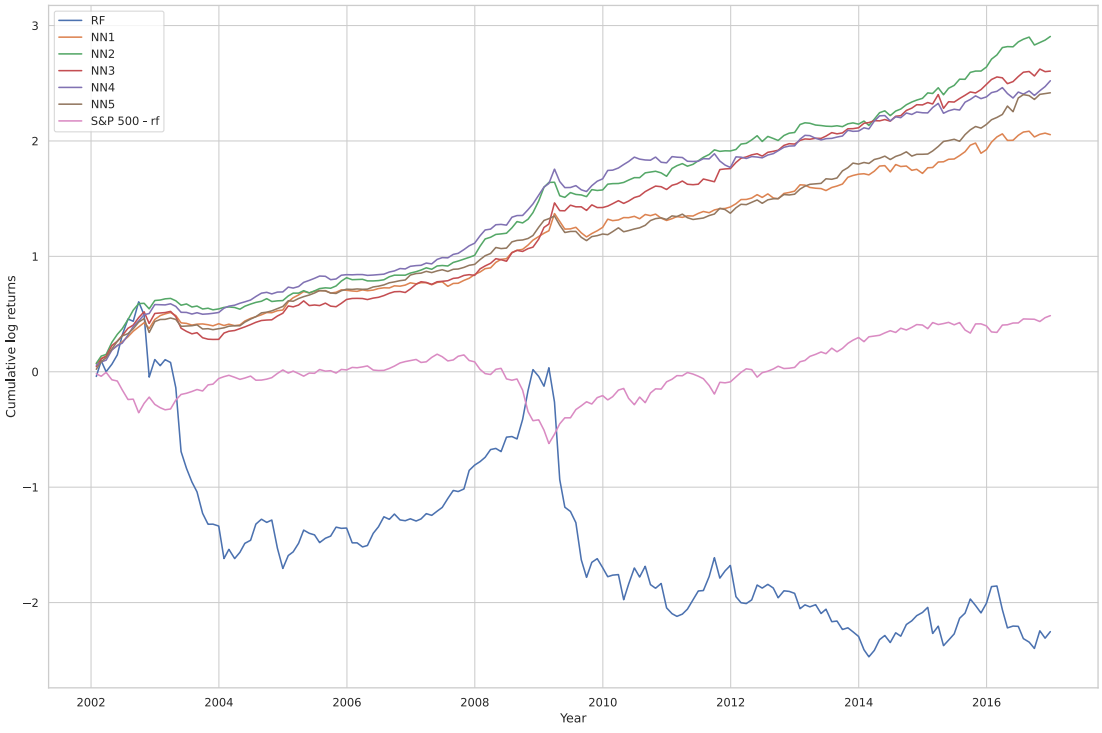
	Decile spread			Class spread		
Model	Return	SR	Turnover	Return	SR	Turnover
OLS-3+H	-0.25	0.05	276	-	-	-
RF	17.44	0.96	211	-13.94	-0.15	107
NN1	36.89	2.20	233	14.68	1.37	203
NN2	28.56	1.83	233	21.37	1.81	196
NN3	35.55	2.15	230	18.97	1.51	210
NN4	37.17	2.46	235	18.31	1.57	210
NN5	33.80	2.19	234	17.49	1.67	208

This table reports the annualized return in percent (Return), annual Sharpe ratio (SR), and average monthly turnover in percent (Turnover) for two different trading strategies using forecasts from different models. The decile spread strategy uses level predictions for equity risk premiums, and the class spread strategy uses directional forecasts. The results are based on the test sample from 2002 - 2016.

Figures 6a and 6b show the cumulative returns of the two trading strategies. It can be seen that the different types of neural network models produce similar returns over time and that they provide the highest returns in general. Whereas the buy-and-hold strategy of the S&P 500 Index produced large negative returns around 2008, the strategies using machine learning models actually produced significant positive returns. This can be explained by the strong performance of the short side of the portfolios. The cumulative returns of the neural networks in Figure 6b appear to lie on a straighter line than the ones in Figure 6a, suggesting that the returns of a class spread strategy are less volatile. This is supported by the Sharpe ratios in Table 5. For example, a decile spread strategy in combination with NN2 yields an annualized return of 28.56% and a Sharpe ratio of 1.83, whereas a class spread strategy yields a much lower annualized return of 21.37% but a similar Sharpe ratio of 1.81. Hence, this implies that the return volatility of the class spread strategy is significantly lower.



(a) Decile spread strategy (level forecasts)



(b) Class spread strategy (directional forecasts)

Figure 6: Cumulative log returns for the decile spread strategy (a) and class spread strategy (b) using different models over the period 2002-2016. The zero-net-investment portfolio holding the S&P 500 Index in excess of the risk-free rate is plotted as a benchmark. Transaction costs are not accounted for.

6 Conclusion

This paper uses machine learning methods to predict the level and direction of monthly U.S. equity risk premiums in the period 2002-2016. I consider 92 stock-level characteristics and their interactions with macroeconomic variables as predictors. For directional forecasting, I use the multinomial approach of Nevasalmi (2020) in order to put more emphasis on predicting large absolute excess returns.

Generally, the level forecasts do not outperform constant forecasts of zero as measured by the out-of-sample R^2. By means of Diebold-Mariano tests, I conclude that the level predictions of a random forest model are better than those of neural network models and of a simple linear model. The predictors that are found to be the most informative across all machine learning methods are market value and short-term reversal, in line with Gu et al. (2020). The directional predictions of a random forest and neural networks yield more accurate results than naive forecasts that predict the most frequent class. This result is particularly strong for smaller stocks.

I create two trading strategies. The first is the decile spread strategy, which uses the level forecasts to buy the top decile of stocks and sell the bottom decile of stocks based on the predictions every month. The second is the class spread strategy, which uses the directional forecasts to buy the stocks with a buy recommendation and sell the stocks with a sell recommendation every month.

It is found that the decile spread strategy using the predictions of a feed-forward neural network with 4 hidden layers yields the highest absolute return and risk-adjusted return. Although the class spread strategy outperforms a buy-and-hold strategy of the S&P 500 Index, its returns are not as high as those of the decile spread strategy. The only advantage of the class spread compared to the decile spread strategy is that its returns are less volatile and that its monthly turnover rate is lower. Even after accounting for transaction costs, both strategies are able to outperform the buy-and-hold strategy of the S&P 500 Index. Therefore, I conclude that using machine learning methods for level and directional forecasting of equity risk premiums can result in large financial returns.

For further research, it could be interesting to create trading strategies based on the directional forecasts for small stocks only, given the relatively high directional predictability for these stocks. In addition, it is interesting to examine what the exact impact of transaction costs is on the performance of the trading strategies, given that the turnover rates are high. Lastly, one could try to find optimal values for the labeling thresholds, because there is a trade-off between balanced classes and strong signals, see Nevasalmi (2020).

References

Abe, M., & Nakayama, H. (2018). Deep learning for forecasting stock returns in the cross-section. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 273–284).

Becker, J., & Leschinski, C. (2018). *Directional predictability of daily stock returns* (Tech. Rep.). Hannover Economic Papers (HEP).

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

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

Butaru, F., Chen, Q., Clark, B., Das, S., Lo, A. W., & Siddique, A. (2016). Risk and risk management in the credit card industry. *Journal of Banking & Finance*, *72*, 218–239.

Christoffersen, P. F., & Diebold, F. X. (2006). Financial asset returns, direction-of-change forecasting, and volatility dynamics. *Management Science*, *52*(8), 1273–1287.

Chung, J., & Hong, Y. (2007). Model-free evaluation of directional predictability in foreign exchange markets. *Journal of Applied Econometrics*, *22*(5), 855–889.

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues.

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

Fiévet, L., & Sornette, D. (2018). Decision trees unearth return sign predictability in the s&p 500. *Quantitative Finance*, *18*(11), 1797–1814.

Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, *2*(4), 42–47.

Green, J., Hand, J. R., & Zhang, X. F. (2017). The characteristics that provide independent information about average us monthly stock returns. *The Review of Financial Studies*, *30*(12), 4389–4436.

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

Gunst, R., & Webster, J. (1975). Regression analysis and problems of multicollinearity. *Communications in Statistics-Theory and Methods*, *4*(3), 277–292.

Hong, Y., & Chung, J. (2003). Are the directions of stock price changes predictable? statistical theory and evidence.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics* (pp. 492–518). Springer.

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A nonparametric approach to pricing

and hedging derivative securities via learning networks. *The Journal of Finance*, *49*(3), 851–889.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).

Karhunen, M. (2019). Algorithmic sign prediction and covariate selection across eleven international stock markets. *Expert Systems with Applications*, *115*, 256–263.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leitch, G., & Tanner, J. E. (1991). Economic forecast evaluation: profits versus the conventional error measures. *The American Economic Review*, 580–590.

Leung, M. T., Daouk, H., & Chen, A.-S. (2000). Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of forecasting*, *16*(2), 173–190.

Lewellen, J. (2014). The cross section of expected stock returns. *Forthcoming in Critical Finance Review, Tuck School of Business Working Paper*(2511246).

Linton, O., & Whang, Y.-J. (2007). The quantilogram: With an application to evaluating directional predictability. *Journal of Econometrics*, *141*(1), 250–282.

Masters, T. (1993). *Practical neural network recipes in c++*. Morgan Kaufmann.

Nevasalmi, L. (2020). Forecasting multinomial stock returns using machine learning methods. *The Journal of Finance and Data Science*, *6*, 86–106.

Nyberg, H. (2011). Forecasting the direction of the us stock market with dynamic binary probit models. *International Journal of Forecasting*, *27*(2), 561–578.

Rapach, D. E., Strauss, J. K., & Zhou, G. (2010). Out-of-sample equity premium prediction: Combination forecasts and links to the real economy. *The Review of Financial Studies*, *23*(2), 821–862.

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

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Weigand, A. (2019). Machine learning in empirical asset pricing. *Financial Markets and Portfolio Management*, *33*(1), 93–104.

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

A Appendix

Table 6: List of characteristics that are used to create interaction terms.

Characteristic	Description	Category
mom1m	1-month momentum	Momentum
mom6m	6-month momentum	Momentum
mom12m	12-month momentum	Momentum
mom36m	36-month momentum	Momentum
chmom	Change in 6-month momentum	Momentum
indmom	Industry momentum	Momentum
maxret	Recent maximum daily return	Momentum
turn	Share turnover	Liquidity
std_turn	Volatility of liquidity	Liquidity
mvel1	Firm size	Liquidity
dolvol	trading volume (in dollars)	Liquidity
ill	Illiquidity	Liquidity
zerotrade	Zero trading days	Liquidity
baspread	Bid-ask spread	Liquidity

This table lists the 14 stock-level characteristics that are used to create interaction terms with the 8 macroeconomic variables. The category corresponding to each variable is also reported. More information about these variables can be found in Appendix F of Gu et al. (2020).

Table 7: Hyperparameter tuning schemes

Method	**Hyperparameter**	**Values**
OLS-3+H	Huber loss (ξ)	Sklearn default
RF	Tree depth	{1, 2, 3, 4}
	Number of trees	300
	Number of features at each split	{3, 5, 10}
NN1-NN5	"l1" penalty	{0.00001, 0.0001, 0.001}
	Learning rate	{0.001, 0.01}
	Number of epochs	100
	Batch size	10000
	Early stopping patience	5
	Adam parameters	Keras default

This table reports the considered hyperparameter values for each model. The values are similar to those in Gu et al. (2020) for replication purposes.

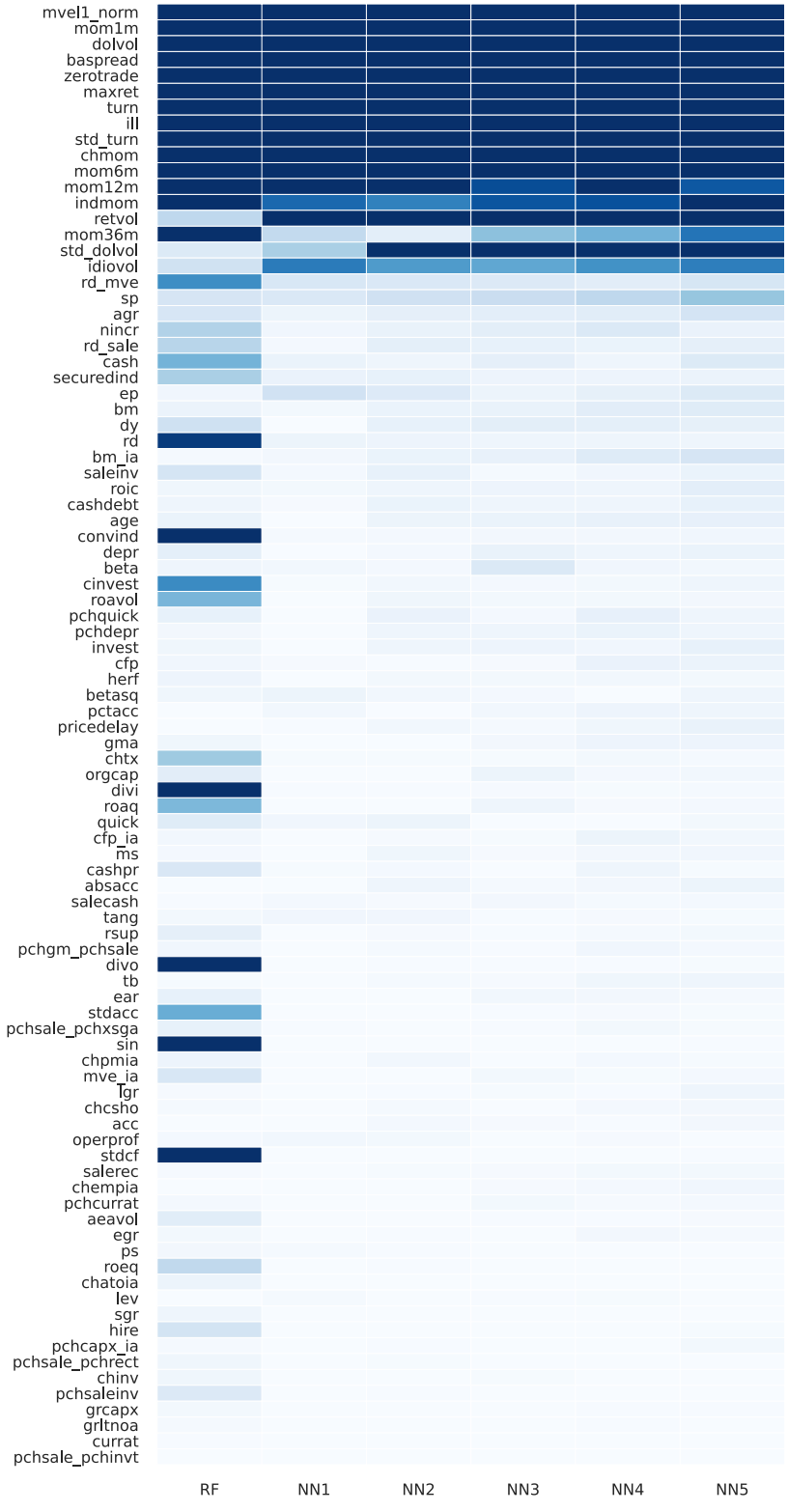


Figure 7: Variable importance ranking across models. The color gradients in a column display the importance of each predictor in a model. Since the importance of the predictors can be highly skewed in distribution, I use the interval $[0, 1]$ for the color mapping where the interval is in terms of percentage reductions in R^2.

Actual class \ Predicted class	1	2	3
1	33,395	163,696	0
2	21,382	582,765	0
3	25,174	283,763	0

(a) Confusion matrix RF

Actual class \ Predicted class	1	2	3
1	52,804	98,012	46,275
2	48,185	494,620	61,342
3	51,036	197,166	60,735

(b) Confusion matrix NN1

Actual class \ Predicted class	1	2	3
1	58,842	97,706	40,543
2	56,671	493,838	53,638
3	58,600	196,677	53,660

(c) Confusion matrix NN2

Actual class \ Predicted class	1	2	3
1	54,442	100,720	41,929
2	52,360	500,418	51,369
3	53,371	202,825	52,741

(d) Confusion matrix NN3

Actual class \ Predicted class	1	2	3
1	54,805	98,918	43,368
2	53,652	495,164	55,331
3	56,042	197,509	55,386

(e) Confusion matrix NN4

Actual class \ Predicted class	1	2	3
1	53,019	99,310	44,762
2	47,117	501,342	55,688
3	48,779	203,456	56,702

(f) Confusion matrix NN5

Table 8: (a)-(f) report confusion matrices for the models RF, NN1, NN2, NN3, NN4, and NN5.