Toulouse School of Economics

Master Thesis

ETE

Robustness of Machine Learning Methods Towards Time Aggregation in the Case of Predicting Asset Returns

Presented by
Peter Neis

Supervisor
Nour Meddahi

Jury
René Garcia
Eric Gautier

June 2018

Contents

1. Introduction

Predictability of equity premia and asset returns are a central problem in asset pricing research.

The well known capital asset pricing model (CAPM, Sharpe, 1964; Lintner, 1965 and Black, 1972, for empirical evidence Jensen, Black, and Scholes, 1972 and Fama and MacBeth, 1973) states that market betas suffice to describe the cross section of expected returns. This view has soon been contested, since empirical research found many other factors that added to the explanation (see for instance Banz, 1981 and Bhandari, 1988) until Fama and French (1992, 1993) proposed in the early 1990s their three-factor model.

Although the existence of many different models and empirical analysis, people questioned the predictability of returns. For instance Welch and Goyal (2007) test the variables suggested by the literature to estimate equity premia and find that they perform poorly in-sample and out-of-sample, meaning that historical average excess stock returns are the best predictor for the future. On the other hand Campbell and Thompson (2007) and Cochrane (2007) argue in two high-visibility papers for the predictability of returns. The former by showing that by adding weak assumptions, like positive slopes or that the forecast has to be positive, the forecast of excess stock returns performs better than the historical mean does. Cochrane argues that the non predictable dividend growth is a strong argument for predictability of returns. Since then the research has shifted from the question weather returns are predictable towards investigation of the goodness of predictions. Ferreira and Santa-Clara (2011) argue that the negative results from Welch and Goyal (2007) can not be reproduced when one separately forecasts the components of stock market returns and than aggregates the forecast. Lewellen et al. (2015) studies cross-sectional properties of return forecasts derived from Fama-MacBeth regressions. The forecasts vary substantially across stocks and have strong predictive power for actual returns.

Once it is assumed that asset returns are predictable, the link to machine learning is not far. Following Mohri, Rostamizadeh, and Talwalkar (2012), machine learning can be defined as computational methods using experience to make accurate predictions. This means, it consists of efficient and complex algorithms which takes, probably big amounts of, available information (the experience) to create predictions. From a statistical point of view this means nothing else than having high dimensional models combined with regularization methods to do good predictions.

Therefore machine learning seems naturally to be of interest to do predictions for asset returns. First of all because prediction is what it is all about and second because these algorithms are capable to handle the big amounts of financial data we have access to today. To be precise, predicting asset returns given past information is a sort of supervised learning where regression techniques are applied to the specific problem.

Although machine learning is nearly as old as asset pricing theory, only with the recent raise in computing power, the two topics have been connected in research. Kozak, Nagel, and Santosh (2017) give a good overview over the cross-section of empirical asset pricing research with machine learning and the authors create a stochastic discount factor, taking into account a large number of cross-sectional stock return predictors. Feng, Giglio, and Xiu (2017) show that a simple Lasso model is not stable and propose a new model-selection method to select the important factors. Freyberger, Neuhierl, and Weber (2017) use adaptive group Lasso to test which characteristics are relevant for expected return predictions and to estimate the impact of those characteristics on the expected returns. Tsai et al. (2011) use classifier ensembles of trees, neural networks and logistic regressions to predict stock returns. They distinguish between "homogeneous" (i.e an ensemble of of one method) and "heterogeneous" (i.e. an ensemble of different methods) classifier ensembles. Moritz and Zimmermann (2016) use a tree-based method

to decide which characteristics are useful to predict the cross-section of returns. They find that short-term returns are the most important characteristics.

This work is mainly based on Gu, Kelly, and Xiu (2018) who compare and apply the most common machine learning methods to the problem of measuring asset risk premia. This thesis aims at rebuilding their results and to append their work by introducing time aggregation. Explicitly this means to test whether their results are robust to more than one-month ahead forecasts. The rest of this work is structured as follows. First, the prediction methods and tests are presented. This is followed by an overview over the different machine learning methods used. The behavior of these methods is analyzed in a Monte Carlo simulation in chapter four. The results of the simulation are compared with an empirical study over data of the last 17 years conducted in the next section. The work concludes by an overview and discussion of the obtained.

2. Prediction

Before applying the different methods, the data is split into three different subsamples, "Training", "Validation" and "Testing". The idea behind this is simple. The methods are trained using the observations in the training sample. For cross-validation of the different tuning parameters, the validation sample is used. This is, for different values of the tuning parameters, a model is fitted to the training data and then is used for an out-of-sample prediction on the validation sample. The model with the tuning parameters performing best in this prediction exercise is then chosen as the final model. Since the validation data is used for cross-validation, this is not a real out-of-sample estimation. Therefore the testing data is used for real out-of-sample prediction assessment.

This invokes the natural question of the splitting rule. The simplest version follows Gu, Kelly, and Xiu (2018) and divides the simulated data in three subsamples of the same size. In a future work it would be interesting to see how the method's accuracy changes if the splitting method is changed, for instance by introducing a rolling window version, where the sub-samples are adapted every month.

This work focuses on the pure prediction of returns. Based on the mean-squared error (MSE), which is widely used in forecasting (see e.g. Elliott and Timmermann, 2016), the following statistic is used to measure and compare the quality of the prediction for the different methods:

$$L_{OOS} = \frac{\sum_{(t,i)\in test}(r_{i,t+1} - g(z_{i,t};\theta))^2}{\sum_{(t,i)\in test}(r_{i,t+1} - \frac{1}{N_1 T_1}\sum_{(t,i)\in train} r_{i,t+1})^2}$$

This is closely related to the out-of-sample R^2, the transformed tables are given in the appendix:

$$R^2_{OOS} = 1 - \frac{\sum_{(t,i)\in test}(r_{i,t+1} - g(z_{i,t};\theta))^2}{\sum_{(t,i)\in test}(r_{i,t+1} - \frac{1}{N_1 T_1}\sum_{(t,i)\in train} r_{i,t+1})^2}$$

The R^2 is a measure of the goodness of fit of a regression. To do so, it compares the fit of the regression with the fit of a simple straight line (see for instance Elliott and Timmermann, 2016). The R^2 can be negative, when the chosen model is a worse predictor than the straight line. Note that other than proposed by Gu, Kelly, and Xiu (2018) (but as done in their Internet appendix), the out-of-sample R^2 is demeaned to compare the forecast with historical mean returns.

Gu, Kelly, and Xiu (2018) focus on one-month individual stock predictability. One well observed and studied future of asset returns is that the predictability increases with the horizon. The main contribution of this work is to see how the different machine learning methods adapt to larger horizons.

The Diebold-Mariano test (Diebold and Mariano, 1995) is used to compare out-of-sample predictive accuracy of different methods pairwise. As state Gu, Kelly, and Xiu (2018), the original test assumes weak error dependence which can not reasonably assumed in the context of cross-sections of individual stocks. Therefore the test statistic is adapted in order to compare the cross-sectional mean of prediction errors between a model 1 and a model 2:

$$DM_{12} = \bar{d}_{12}/\hat{\sigma}_{\bar{d}_{12}}$$

$$d_{12,t+1} = \frac{1}{n_3}\sum_{i=1}^{n_3}\left(\left(\hat{e}_{i,t+1}^{(1)}\right)^2 - \left(\hat{e}_{i,t+1}^{(2)}\right)^2\right)$$

$\bar{d}_{12}$ denotes the mean of $d_{12,t+1}$ over the testing subsample, $\hat{\sigma}_{\bar{d}_{12}}$ denotes the corresponding Newey-West standard error. n_3 is the number of stocks in the testing subsample. $\hat{e}_{i,t+1}^{(m)}$ denotes the prediction error of model m for stock i at time $t+1$.

3. Methods

3.1. Pooled OLS

The simplest model used as a benchmark value is the simple ordinary least squares (OLS). Since here the data considered has a panel form, this is equal to the so called pooled OLS. The simplicity is due to the assumption that one can approximate the true model by a linear function of a parameter vector and explanatory variables. It is assumed that there is an individual-specific effect which enters linearly in the regression:

$$r_{i,t+1} = z_{i,t}'\theta + u_i + \epsilon_{i,t+1}$$

In this model $g(z_{i,t};\theta) = z_{i,t}'\theta$ and therefore it is a fully parametric, linear model without any interactions between the regressors. For the model to be consistent one needs a couple of assumptions: The individual stocks are mutually independent, that u_i and $\epsilon_{i,t+1}$ are independent, that $\epsilon_{i,t+1}$ is iid across individuals and time, that $\epsilon_{i,t+1}$ is uncorrelated with $z_{i,t}'$ and most importantly the random effects hypothesis $E(z_{i,t}u_i) = 0$ (see Hansen, 2017).

In most settings at least the random effect hypothesis is assumed to be violated, this would be a problem for inference and testing. Since in this setting, we are only interested in prediction ability and OLS is used as a simple benchmark model, this might not be a major concern.

The loss function is the standard l_2-loss:

$$\mathcal{L}(\theta) = \frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N}(r_{i,t+1} - z_{i,t}'\theta)^2$$

This loss function can be minimized analytically and since there are no tuning parameters, the computation is fast and not a drain on resources. Although the l_2-loss is convenient for computational issues, it is prone to give a huge importance to large errors which can be a problem for the stability of the prediction. This is of special concern for financial analysis, since heavy tails are a well observed artifact of financial data. (See fo instance Bradley and Taqqu, 2003)

Therefore adapted loss functions have been proposed which handle large errors in a more robust way. One popular robust objective function in machine learning is the Huber robust objective function, proposed first in Huber (1964), which tackled first the problem of robustness into the topic of high-dimensional statistics (Efron and Hastie, 2016). The loss function is a combination of the l_2-loss for all errors smaller than a predefined threshold and absolute loss for

large errors:

$$\mathcal{L}_{\mathcal{H}}(\theta) = \frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N} H(r_{i,t+1} - z'_{i,t}\theta, M)^2$$

where

$$H(a, M) = \begin{cases} a^2 & if\ |a| \leq M \\ 2M|a| - M^2 & if\ |a| > M \end{cases}$$

The so called Huber-threshold $M > 0$ is a tuning parameter which can be optimized using the validation sample. This robust objective function can not only be used to improve the OLS model but as well in all methods studied besides random forest.

3.2. Lasso, Ridge, Elastic Net

OLS is widely used, due to its simplicity and the fact that the number of parameters to estimate is independent of the number of observations and only depends on the number of predictor variables P used in the regression. The drawback is, that with todays growing available data and especially in cases where one has no prior knowledge (and does not want to add assumptions to the model) about which variables to use, P can be large. When P approaches the number of observation N, this can actually cause harm to the estimation, since least-squares estimators may not uniquely exist and be inconsistent.

To adapt to this problem, regularization techniques in order to reduce the number of parameters have been proposed. This is, the objective function is expanded by a penalty over the parameters, accepting that the estimator is biased. By this penalization, the econometricians hopes to achieve a better out-of-sample fit by decreasing the in-sample one (and therefore to avoid overfitting). The elastic net penalty is to add a linear combination of an l_1 and l_2-penalty to the original loss function:

$$\mathcal{L}(\theta; \lambda) = \mathcal{L}(\theta) + \lambda(1-\rho)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\rho\sum_{j=1}^{P}\theta_j^2$$

In the case when $\rho = 1$, this is equal to Ridge regression, a shrinkage method to improve the estimation by adding an l_2-penalty on the parameters to the original loss function (see Hoerl and Kennard, 1970). Ridge regression can be interpreted as an increased prior belief that θ is close to 0. Due to the ridge penalty the new estimators are shrunk closer to the believed population parameters. Note that all coefficients are shrunk by the same factor, meaning that it does not result in a sparse model since no coefficients are set to 0 but in a parsimonious one. The tuning parameter in Ridge regression λ can be found via cross-validation by minimizing the prediction error. Setting $\lambda = 0$ would result in the least squares regression while setting $\lambda = \infty$ shrunk all coefficients to zero.

When $\rho = 0$, one recovers the so called Lasso penalty. The Lasso uses an L_1-penalty to shrink coefficients and to set some coefficients to exactly zero (see Tibshirani, 1997). As a result it reduces the estimation variance and does a kind of variable selection by introducing sparsity which provides an interpretable final model. Again, λ is the tuning parameter optimized via cross-validation.

When $\rho \in (0, 1)$ the elastic net uses a linear combination of the lasso and ridge penalties. The elastic net is especially useful when the number of variables is importantly larger than the number of observations, then it often outperforms the lasso, while enjoying a similar sparsity of representation (see Zou and Hastie, 2005). Then one has to find both λ and ρ by cross-validation. This means that it is computational more intensive but might lead to improved predictions.

3.3. Regression Trees

The more data one has, where more is meant with respect to the number of observations and the number of predictor variables, the less useful are the traditional techniques presented before. One non-parametric method handling especially well big amounts of data are regression trees. Regression trees use recursive partitioning to transform a big amount of input data into predicted values. The principle is simple. At any node of the tree, the observations associated to this node are divided following a binary decision rule over one explanatory variable. At the very first node this means that all observations are evaluated to have one specific variable being larger or smaller than a defined threshold.

The so created (two) subgroups of observations can then, independently one from the other, being split again in two new subgroups due to (probably another) variable with a new threshold. The final nodes are called leaves. The average over outcomes of all observations in one specific leave is then the prediction for an observation being assigned to this final node by the decision rules.

The most common method to choose at any node the variable and the threshold to split up on is to choose the two subgroups in order to minimize the forecast error. This is equal to maximizing the difference between the subgroups. Note that since it impossible to calculate every possible following combination of the tree to find the real loss due to the choice at one node, this method maximizes differences at one node, without taking into account the impact of this decision on the difference between (potentially) following subgroups. The corresponding algorithm based on L. Breiman et al. (1984) calculates at each node the so called l_2-impurity function:

$$Imp(C, C_l, C_r) = \frac{1}{|C|} \sum_{z_{i,t} \in C_l} \left(r_{i,t+1} - \frac{1}{|C_l|} \sum_{z_{i,t} \in C_l} r_{i,t+1} \right)^2 + \frac{1}{|C|} \sum_{z_{i,t} \in C_r} \left(r_{i,t+1} - \frac{1}{|C_r|} \sum_{z_{i,t} \in C_r} r_{i,t+1} \right)^2$$

where l and r indicate respectively the following left and right subgroups, C (C_l, C_r) is the set containing all observations in the node before (after) splitting. The optimal decision rule at a given node is the one minimizing the impurity function at this node. The maximal depth of a tree L, i.e. the number of consecutive splittings, can be optimized via tuning over the validation sample.

Algorithm 1 Regression Tree

1. Define the set of observations at the initial node as the entire training set
2. For each predictor variable and each threshold level define the two subsets "left" and "right" and calculate the impurity function. Choose the predictor variable and threshold minimizing the impurity function.
3. Iterate step 2 over each newly created node until each branch attained the maximum length L or the terminal node of the branch contains only one observation
4. For each leave take the average over the returns of the observations contained in the leave

The ability of regression trees to handle big amounts of data is due to these simple, binary sub-setting rules, which create smaller and smaller bins. Another advantage of trees is that they are easy to interpret, one can see the variables and thresholds selected at each node. Therefore, one can see Regression Trees as a sort of variable selection mechanism. Still, one single tree is

prone to overfit the data and to be bad for prediction tasks. To handle these issues, two widely used ensemble methods have been proposed: Random Forests and Boosting.

3.3.1. Random Forest

Random Forest was first proposed by Leo Breiman (2001). The basic idea behind it, is to grow a relatively large number of deep and probably overfitted trees and to reduce the high variance of each individual tree by taking the average over all trees. To introduce variation between the trees, one randomizes the data for each tree. This randomization is made on two different dimensions. First, one creates for each tree a resampled dataset via bootstrap. This makes the data on which each tree is grown different and therefore decorrelates the trees. Second, at each node, the set of explanatory variables to choose one variable for splitting from (by minimizing the impurity function) is randomized. The most popular variant in the literature is to select randomly at each node 1/3 of all explanatory variables.[1]

Algorithm 2 Random Forest

1. Draw B bootstrap samples from the original training data.
2. For each $b \in [1, ..., B]$ grow a tree following the regression tree algorithm. Instead of using at each node the whole sample of regressors, only use a random subset with 1/3 of the regressors.
3. Take the average over the outputs of all B trees

Here the tuning parameters are the depth of each tree L and the number of trees B. B needs to be large enough to stabilize the prediction but if it is choose too large there is again a risk of overfit. To find the optimal values, one can predict the returns of the validation sample via the models fitted on the training sample and choose the model doing the best prediction.

Since in Random Forest, the interpretation of the trees is way harder than just for one single regression tree it is common to use variable importance plots to interpret the underlying mechanisms of the method. The idea is to measure the loss when not using a specific variable for splitting, this is every time the variable is actually used in any tree for splitting, to report the gain in the impurity function of using this variable instead of the next best one. Adding these gains for each variable over all B trees, one gets a measure for the relative importance of a variable. This implies that using a smaller randomized set of variables at each node can be compared to ridge regression since it will more equally distribute the relative importance of the variables.

3.3.2. Boosting

The main idea behind boosting is to reduce bias. This is done by growing shallow (small L) trees to fit the residuals of the last step and update the model by adding the new tree multiplied by a shrinkage parameter $\nu \in (0, 1]$ to the previous model. As the trees considered are shallow, the literature talks about weak learners

Tuning parameters for boosting are the tree depth L, the shrinkage parameter ν and the number of shallow trees to grow successively B. If B is chosen too large, the model can overfit, usually the prediction first improves in the number of trees before deteriorating when the number becomes too large.

[1] Using only the first variant of randomization, i.e. using the whole set of variables at each node, is the predecessor of Random Forest and called Bagging.

In order to use any differentiable loss function beside the classic squared-error (such as Huber) it is common to use gradient boosting, this is to build an algorithm inspired by gradient descend. The main idea behind this, is to calculate the gradient vector given the last fit and to approximate this gradient by a new shallow tree. As before, this new tree is added, shrunken by ν, to the former model. To add the shrunken tree to the last model is equal to take a step of length ν down the gradient (see for instance Efron and Hastie, 2016).

Algorithm 3 Gradient Boosting

1. Initialize the predictor as $\hat{g}_0 = 0$
2. Compute the pointwise negative gradient of the loss function $L(\cdot, \cdot)$:
$$\epsilon_{i,t+1} = - \left. \frac{\partial L(r_{i,t+1}, g)}{\partial g} \right|_{g=\hat{g}_{b-1}(z_{i,t})}$$
3. Approximate the negative gradient by a shallow tree of depth L where $\epsilon_{i,t+1}$ is the explained variable by solving:
$$\min \sum_{i=1}^{N} \sum_{t=1}^{T} (\epsilon_{i,t+1} - g(z_{i,t}; \theta, L))^2$$
4. Update the model: $\hat{g}_b = \hat{g}_{b-1} + \nu g(z_{i,t}; \theta, L)$
5. Repeat steps 2 and 3 for all $b \in [1, ...B]$. Return the final model $\hat{g}_B(\cdot; \cdot)$.

3.4. Neural Networks

After having been popular in the 1980s, Neural network became popular again in the last years for what we call "Deep Learning". In the eighties they were especially popular because of the universal approximation theorem stating that even a single hidden layer neural network with a finite number of neurons can uniformly approximate any continuous function of n real variables with support in compact subsets of $\mathcal{R}^n$ (see Cybenko, 1989 for one hidden layer with sigmoid activation function or Hornik, Stinchcombe, and White, 1989 for a multilayer theorem). Basically, a neural network is a highly parametrized model, connecting input data to intermediate steps where the data is transformed non-linearly before being reunited in an output function.

Every neural network has the same basic structure. There is an input layer consisting of the original predictors and a chosen number (≥ 0) of hidden layers. Each hidden layer consists of a number of so called "neutrons". The connection between a neuron in a hidden layer and its input from the last layer can be manifold. The case analyzed here follows the common case where each neuron in one layer is connected with each neuron/input from the previous layer. This means that the input for each neuron consists in the output of the neurons of the last layer which is together with an intercept term (called bias) multiplied by a vector of neuron specific parameters.

To introduce nonlinearity, each neuron applies an activation function on its inputs. In this setting, the same function is applied to each neuron, but neither to the initial input layer nor to the output function. There are two common nonlinear activation functions, the sigmoid function mentioned earlier and the rectified linear unit (ReLU, see for instance Nair and Hinton, 2010) function:

$$ReLU(x) = \begin{cases} 0 & if\ x < 0 \\ x & otherwise \end{cases}$$

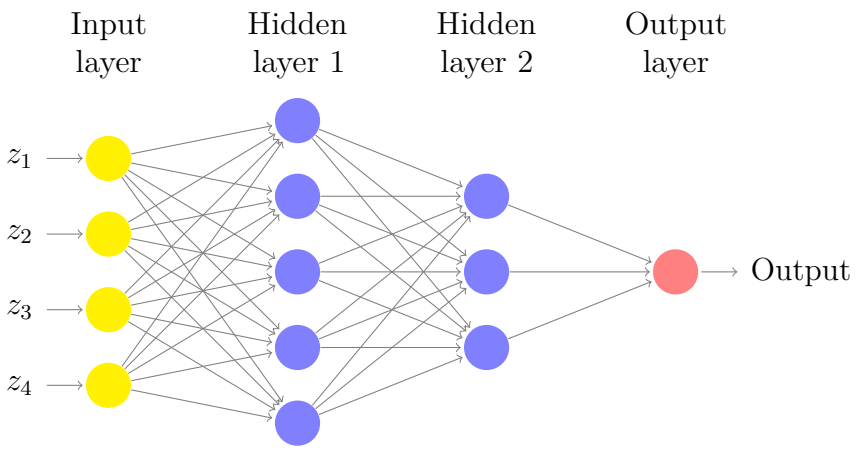

Figure 1: Feed-forward Neural Network diagram with two hidden layer with 4/5/2/1 all-connected neurons

To build up a neural network one has to make a lot of choices, the number of hidden layers, the number of neurons in each layer, which layers are connected, which activation function to use, and often which kind of regularization to apply to the network to improve the results. This is part of the strength of neural networks, they can be adapted to nearly every situation. On the other side, this means that the statistician has many choices to take which will influence the goodness of the prediction. (See for example: Stathakis (2009) or Heaton (2008))

The l_2 objective function on prediction errors with an l_1-penalization on the parameter values is used for estimating the neural network.

Here, neural networks with one (NN1) up to five (NN5) hidden layers are constructed. The first layer always has 32 neurons, the possible second one 16, the third 8, the fourth 4 and the fifth 2. This is, a geometric pyramid rule is applied. Assuming to have 100 input variables, the number of parameters gets large very fast. For instance the neural network with three hidden layers has $101 * 32 + 33 * 16 + 17 * 8 + 9 = 418065$ parameters to estimate.

To handle this huge number of parameters, together with the nonlinearity, regularization is needed to speed up the training and validation process and to increase the prediction accuracy of the model.

First, to train the model the widely used stochastic gradient descent (SGD) is applied. Before the training process starts, it splits the training data into small (usually random) batches. Here, one batch consists of the total cross-section of observations at a given time t. Starting with the most recent batch, SGD updates the initial guess on the parameter (often a gaussian sample) by evaluating the gradient only on the actual batch, then continuing updating the new guess with the second most recent batch and so on.

During the updating process the learning rate plays a critical role. This is, the updating of the parameters is slowed down by a shrinkage parameter, the learning rate. While the learning rate has traditionally been fixed for all batches and tuned via cross-validation, today it is more common to use adaptive learning rates. Two main techniques can be identified. First, momentum, helps to improve the algorithm whenever the gradient is close to a local optima. In these areas, SGD oscillates a lot and makes only marginal progress towards the local optimum. Momentum takes the gradient calculated by SGD and adding the average of the past gradients. This helps accelerate the optimization to proceed in the right direction. The second method uses an estimated second moment of the gradient for each individual parameter to perform larger or smaller updates depending on their importance. The Adaptive Moment Estimation algorithm (ADAM) proposed by Kingma and Ba (2014) combines these two methods by multiplying the learning rate by an exponentially decaying average of past gradients (momentum) as well as

dividing by exponentially decaying average of past squared gradients. Although it is questioned if adaptive gradient methods are better predictors than stochastic gradient descent (see Wilson et al., 2017), Adam is one of the most popular methods today and is widely used. In future work it could be interesting to compare different methods in the context of return prediction.

Early Stopping: Starting from an initial guess for the parameters, they are updated continuously by the SGD and Adam algorithms to minimize the predictive errors for the training data set. At the same time, after optimizing the new parameters for a batch, the prediction error from the validation sample is calculated given the new parameters. Often it happens that this validation prediction error starts to increase before the training prediction error is minimized. By stopping the optimization process when the validation prediction error starts to increase saves computation time and increases the out-of-sample predictive power of the model (see for example Prechelt, 2012).

Batch normalization is a transformation applied to each neuron after the activation function was used. The idea is to normalize the input for each neuron. To put it in a simple way, it normalizes the output of the previous hidden layer by subtracting the batch mean and dividing by the batch standard deviation and scaling and shifting the result using two tuning parameters. This allows first of all to use higher learning rates which speeds up the training process. Second, it reduces overfitting because it has a slight regularization effect. (Ioffe and Szegedy, 2015)

4. Simulations

The simulations follow closely the internet appendix of Gu, Kelly, and Xiu (2018). The idea is to create a panel of individual stocks with returns and characteristics for each stock i at time t. The characteristics for one stock are persistent over time and for each time t, the characteristics of all stocks at this time are cross-sectionally normalized. The return at time $t+1$ for stock i is constructed as the sum of a function of certain characteristics of stock i at time t, a macroeconomic component which is the same for all stocks at a time and which itself is also persistent over time and an error term. Gu, Kelly, and Xiu (2018) calibrate the model such that the average time series R^2 is 50% and the average annualized volatility is 30%. The panel of returns is constructed as:

$$r_{i,t+1} = g^*(z_{i,t}) + (c_{i1,t}, c_{i2,t}, c_{i3,t})v_{t+1} + \epsilon_{i,t+1} \ \forall t \in [1, ..., T], \forall i \in [1, ..., N]$$

The characteristics $c_{ij,t}, j = 1, ..., P$ are time series for every individual stock i. They are created as

$$c_{ij,t} = \frac{2}{N * T * P_c + 1} rank(\bar{c}_{ij,t}) - 1 \quad where \ \bar{c}_{ij,t} = \rho_j \bar{c}_{ij,t-1} + \epsilon_{ij,t}$$

with the distribution of the autoregressive coefficient ρ_j being $U[0, 1]$ and $\epsilon_{ij,t}$ following a $N(0, 1)$. This gives the two desired features, persistence of characteristics over time and cross-sectional normalization.

The explanatory variables are given as the bundle of all characteristics and all characteristics multiplied by a time series: $z_{i,t} = (1, x_t) \otimes c_{i,t}$ where x_t is a high persistent AR(1)-time series that follows $x_t = 0.9x_{t-1} + u_t$ with $u_t \sim N(0, 1 - 0.9^2)$.

$v_{t+1} \sim N(0, 0.05^2 * \mathbb{1}_1)$. In the original paper, $\epsilon_{i,t+1}$ follows a $t_5(0, 0.05^2)$ distribution. In the following is examined the effect of changing this error distribution slightly on the performance of the different methods.[2]

[2]I could not exactly reproduce the results from the original paper although the nice explanations of Dacheng Xiu who replied nearly immediately to my e-mails. Thank you.

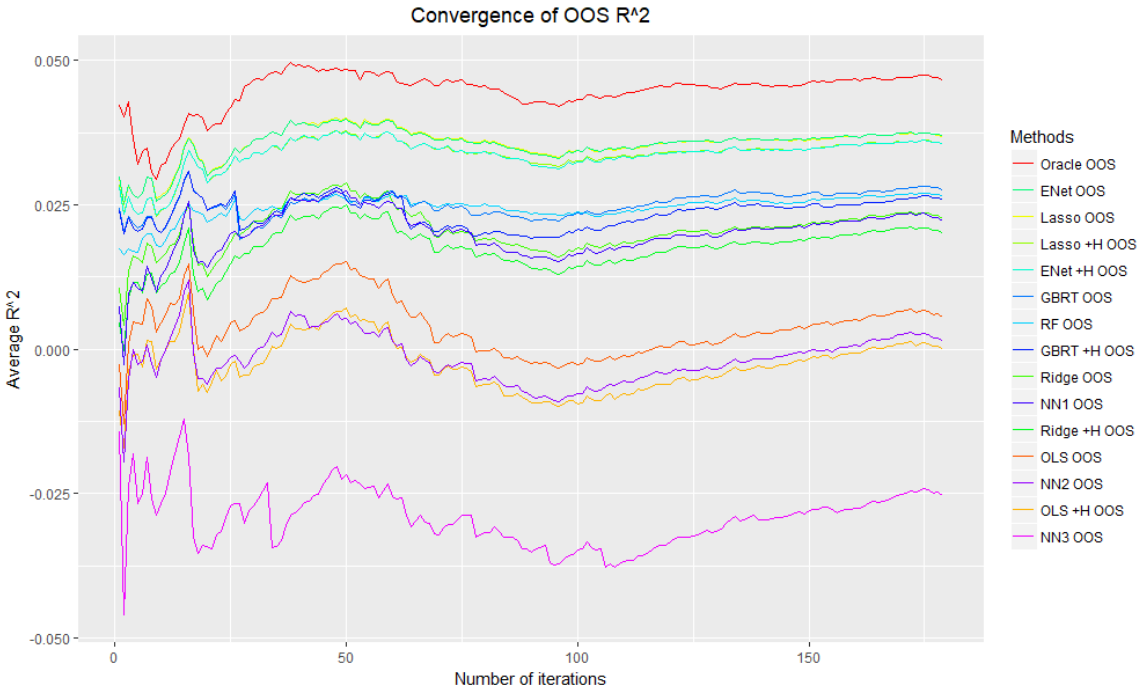

Figure 2: Convergence with number of Monte-Carlo repetitions

$g^*(\cdot)$ corresponds to the case (a) in the original paper and is a simple linear model:

$$g^*(z_{i,t}) = (c_{i1,t}, c_{i2,t}, c_{i3,t} * x_t)(0.02, 0.02, 0.02)'$$

In the following, the number of individual characteristics is set to 50. They are created for an interval of $T = 150$ periods and for $N = 200$ individual stocks. Note that in this simulation, all stocks are present in each time period. For the Monte-Carlo simulation, the number of repetitions is set to 100. Figure 2 shows that the different methods converge relatively fast with the number of repetitions. It gives some evidence that taking 100 repetitions might be a good trade off between computing time and accuracy. Nonetheless, ideally, one might verify the obtained result with a higher number of repetitions.

Following again Gu, Kelly, and Xiu (2018) the simulated data is divided into three equally sized and consecutive subsamples: "Training", "Validation" and "Testing". On the simulated data, the methods presented in chapter 3 are trained, estimated and evaluated. These methods are pooled OLS (denoted OLS), Lasso, Ridge, Elastic Net (ENet), Random Forest (RF), Gradient Boosted Regression Trees (GBRT) and Neural Networks with depth 1 (NN1) up to depth 3 (NN3). Note that no neural networks with deeper than three layers are presented as they have been shown to be worse than shallow networks and the computational capacity for this work has been limited. All methods but RF and NN1 to NN3 are estimated additionally with a huber loss function, denoted by "+H".

As stated earlier, to test the robsutness of the methods towards different data generating processes, three different distributions of $\epsilon_{i,t+1}$ were simulated. The created models were identical upon the distribution of the error term $\epsilon_{i,t+1}$. In the first, denoted $0.05 * t(5)$, it follows a Student's t-distribution with five degrees of freedom multiplied by 0.05 to get a smaller variance. For the second, $t_5(0, 0.05^2)$, the error term follows a Student's t-distribution with five degree of freedoms multiplied by 0.05 and normalized by dividing by $\sqrt{(5/(5-3))}$. This makes the equal to 0.05^2. The last model, $N(0, 0.05^2)$, follows a normal with mean 0 and variance 0.05^2.

The first and the second model both follow a Student's t-distribution with five degrees of freedom and differ only in their respective variance. In the first model, the error term has a

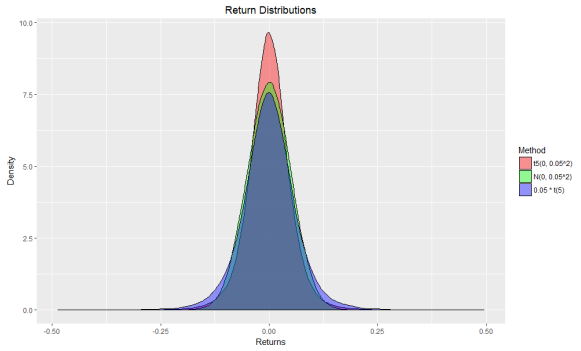

Figure 3: Error distributions for different error terms $\epsilon_{i,t+1}$

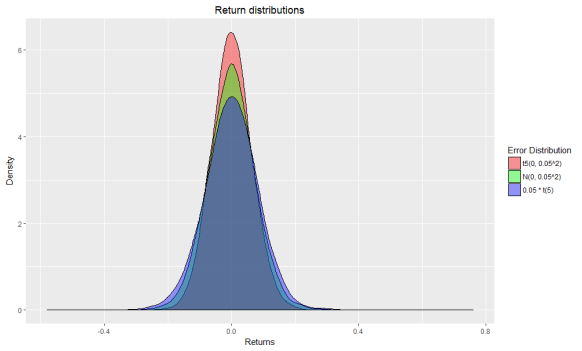

Figure 4: Returns distributions for different error terms $\epsilon_{i,t+1}$

variance of around 0.0042 while for the second model it is 0.0025. The second and the third model have the same variance (0.0025) but one follows a Student's t with 5 degrees of freedom and the other a normal distribution. This means that although they have the same variance, the second model has larger tails than the third model. Figure 3 shows the difference of the three error distributions. Figure 4 depicts the distributions (density) of the returns created with the three different models. Note that the large difference in the error distribution between $t_5(0, 0.05^2)$ and $N(0, 0.05^2)$ creates only a small difference in the return distribution.

Table 1: Differences in the distribution of $\epsilon_{i,t+1}$

Model	Distribution	Variance
$0.05 * t(5)$	Student's t	0.0042
$N_{0.5}(0, 0.05^2)$	Student's t	0.0025
$N(0, 0.05^2)$	Normal	0.0025

Note: Differences of $\epsilon_{i,t+1}$ for the 3 models used.

Table 2 compares the in-sample and out-of sample normalized MSE for the different methods between the three data generating processes. The main observation is that even though the three models differ in the magnitude of their accuracy, the global order of methods is common around all models. Lasso and Elastic Net seem to perform the best, followed by Random Forest, GBRT, Ridge and shallow Neural Networks while OLS with all predictors is the worst. Furthermore, using a Huber Loss instead of a simple L_2-Loss makes no better predictions.

For the neural networks the results are less coherent, for the first and the third model the one layer network performs the best. While for the first model, the neural network seems to give good results, for the third one, the results are worse than all other methods beside OLS. For the second model, the best neural network is the one with two layers which performs as good as Lasso and Elastic Net. This shows a high degree of non-robustness of neural networks towards different data generating processes. While Lasso and Elastic Net are always among the top performers, neural networks are in the first two but not in the third model and it is not clear whether one or two layers is better.

Comparing the first and the second model clearly shows that the later, with a lower variance, performs better for all methods. This implies that a too high variance causes a problem for prediction which is in line with mathematical intuition. Now, beside for the GBRT +H, the second model has higher predictive power for all methods compared with the third model. Remember that both models have the same variance but the second has a larger kurtosis. This

implies that larger tails, i.e. more extreme values in the distribution of the error term, lead to a better prediction.

Table 2: Loss for different DGPs

$\epsilon_{i,t+1}$	$0.05 * t(5)$		$N_{0.5}(0, 0.05^2)$		$N(0, 0.05^2)$	
L	IS	OOS	IS	OOS	IS	OOS
Oracle	0.95	0.97	0.94	0.96	0.93	0.97
OLS	0.93	1.00	0.92	1.00	0.91	1.01
OLS +H	0.94	1.01	0.92	1.01	0.92	1.02
Lasso	0.95	0.97	0.94	0.97	0.94	0.97
Lasso +H	0.96	0.97	0.94	0.97	0.94	0.97
Ridge	0.94	0.99	0.92	0.98	0.92	0.98
Ridge +H	0.94	0.99	0.92	0.99	0.92	0.99
ENet	0.96	0.97	0.94	0.97	0.94	0.97
ENet +H	0.96	0.97	0.94	0.97	0.94	0.97
rf	0.93	0.98	0.91	0.98	0.91	0.98
GBRT	0.94	0.98	0.92	0.98	0.91	0.98
GBRT +H	0.93	0.98	0.91	0.98	0.92	0.98
NN1	0.95	0.97	0.92	0.97	0.92	1.00
NN2	0.95	0.98	0.92	0.97	0.91	1.02
NN3	0.97	0.99	0.94	0.98	0.94	1.05

Note: This table gives an overview of in-sample (IS) and out-of-sample (OOS) loss for one month ahead forecasts of simulated individual stock returns for three different data generating processes. The DGPs are identical upon the distribution of the error term $\epsilon_{i,t+1}$. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. +H indicates that Huber Loss is used instead of a L2 loss function. Additionally, Oracle specifies a pooled OLS regression only with the three real input variables.

The fact that the three models have more or less the same pattern shows a certain degree of robustness towards different data generating processes. Although they can differ in the degree of goodness, the classification of the methods stays more or less the same which is good news, implying that even if we do not know the exact underlying data generating process, with choosing for instance Lasso or Elastic Net, the chances to have good predictions are high.

In Table 3 the test statistics of a pairwise Diebold-Mariano test are depicted. The simulated data used are one-month ahead forecasts with the third model as underlying DGP. In the table, a method in one row is compared to the methods in the columns. A positive test-statistic means that the method of the corresponding row is more accurate than the method in the corresponding column. For significance, the numbers can be interpreted similar to the classic t-test: values larger than 1.96 imply that the accuracy of two methods is significantly different at a 5% or better level. The signs of the pairwise comparison are for most methods as expected. OLS is significantly less accurate than Lasso, Ridge and Elastic Net. Lasso and Elastic Net are both more accurate than Ridge and neural networks with two and three layers. GBRT is significantly more accurate than the neural network with three layers. For all other methods, the difference in accuracy is not significantly different.

For asset return prediction with classic estimation techniques, it is a well known fact, that the larger the horizon (up to a certain point), the better is the predictability (see for instance Lewellen et al., 2015 for a result for monthly and annual prediction). Table 9 tries to recover this feature for the simulations with the third model to investigate weather this is as well the case for machine learning methods. The results are ambiguous. The three month forecasts are

Table 3: Diebold-Mariano Test

p	Lasso	Ridge	ENet	RF	GBRT	NN1	NN2	NN3
OLS	**-2.34**	**-2.04**	**-2.34**	0.02	-1.06	-1.22	0.07	0.58
Lasso		**2.16**	0.00	1.00	0.00	-0.05	**2.73**	**2.25**
Ridge			**-2.16**	0.35	-0.78	-0.87	0.87	1.37
ENet				1.00	0.00	-0.05	**2.73**	**2.25**
RF					-1.80	-1.89	0.02	0.48
GBRT						-0.09	1.93	**2.50**
NN1							**2.83**	**4.73**
NN2								0.90

Note: This table gives the adapted Diebold-Mariano test statistics to compare methods pairwise. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. It can be interpreted as how a row-method performs compared with a column-method. In bold are values with a significance level of 5% or better.

better compared with the one step ahead ones (beside for the neural networks with two and more layers). For a six months forecast the OLS and neural network predictions becomes worse but the other methods still do even better than for three months. For a twelve months forecast, the oracle is still doing better than with less time aggregation but now the only methods which improve are GBRT and OLS, all other methods do not take advantage of the larger horizon. Comparing the order of the methods, it stays more or less the same for one, three and six months forecasts. Only for the larges aggregation, twelve months, GBRT performs better than Elastic Net and Lasso. The fact that some methods do not perform better for the largest horizon can be connected to the data generating process which can not reflect all aspects of reality. More interestingly, the order of the methods in the twelve months case changes with respect to the one month case.

Table 10 compares the prediction performance for one-month ahead forecasts given different sample sizes. The simulations are done with three increasing time periods: $T = 150$, $T = 300$ and $T = 450$. Not surprisingly, all methods perform better, the larger the sample. More interestingly, the order of the methods is robust towards the sample size. Lasso and Elastic Net do the best prediction in all three cases. The difference between the methods seems to become smaller the larger the sample. Especially OLS and the three neural networks benefit largely from the increased sample.

5. Empirical Study

The data for the empirical study comes from two different sources. First the panel of individual stock characteristics and returns is collected using the code of Green, Hand, and Zhang (2017) where the authors collect monthly data on US stocks from Compustat, CRSP and IBES through the WRDS database.[3] The data contains all stocks listed at the NYSE, AMEX or NASDAQ in the period August 1975 to December 2017. In average there are more than 4000 individual stocks each month.

Second, eight macroeconomic variables are included in the dataset. For this, the actualized data from Welch and Goyal, 2007 is taken. The variables are described in the following. Dividend

[3]The code is available on Jeremiah Green's website

Table 4: Out-of-sample Loss for different time aggregations

L_{OOS}	1 month	3 months	6 months	12 months
Oracle	0.97	0.95	0.94	0.94
OLS	1.01	0.99	1.01	0.99
OLS +H	1.02	1.00	1.02	1.00
Lasso	0.97	0.95	0.95	0.96
Lasso +H	0.97	0.95	0.95	0.95
Ridge	0.98	0.97	0.96	0.98
Ridge +H	0.99	0.97	0.97	0.98
ENet	0.97	0.95	0.95	0.96
ENet +H	0.97	0.95	0.95	0.96
RF	0.98	0.96	0.96	0.96
GBRT	0.98	0.96	0.95	0.95
GBRT +H	0.98	0.96	0.95	0.95
NN1	1.00	0.98	1.03	1.04
NN2	1.02	1.07	1.11	1.13
NN3	1.05	1.10	1.12	1.13

Note: This table gives an overview of out-of-sample loss for one month up to 12 months aggregated forecasts of simulated individual stock returns. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. +H indicates that Huber Loss is used instead of a L2 loss function. Additionally, Oracle specifies a pooled OLS regression only with the three real input variables.

Table 5: Out-of-sample loss for different sample sizes

L_{OOS}	150 months	300 months	450 months
Oracle	0.97	0.95	0.95
OLS	1.01	0.97	0.95
OLS +H	1.02	0.97	0.96
Lasso	0.97	0.95	0.95
Lasso +H	0.97	0.95	0.95
Ridge	0.98	0.96	0.95
Ridge +H	0.99	0.96	0.95
ENet	0.97	0.95	0.95
ENet +H	0.97	0.95	0.95
RF	0.98	0.96	0.96
GBRT	0.98	0.96	0.95
GBRT +H	0.98	0.96	0.95
NN1	1.00	0.96	0.95
NN2	1.02	0.96	0.96
NN3	1.05	0.97	0.96

Note: This table gives an overview of out-of-sample loss for forecasts of simulated individual stock returns for different time periods. The periods are $T = 150$, $T = 300$ and $T = 450$. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. +H indicates that Huber Loss is used instead of a L2 loss function. Additionally, Oracle specifies a pooled OLS regression only with the three real input variables.

Table 6: Out-of-sample Loss for different time aggregations

L_{OOS}	1 month	3 months	6 months	12 months
OLS	1.22	4.06	5.26	7.23
Lasso	1.00	1.01	1.01	1.02
Ridge	1.08	1.21	1.52	2.10
ENet	1.00	1.00	1.00	1.00
RF	1.00	1.01	1.01	1.02
GBRT	1.01	1.04	1.02	1.03
NN1	1.00	1.01	1.01	1.17
NN2	1.00	1.01	1.01	1.07
NN3	1.00	1.00	1.00	1.03

Note: This table gives an overview of out-of-sample loss for one month up to 12 months aggregated forecasts for returns of selected assets between 2001 and 2017. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers.

Price Ratio (dp) computed as the difference of the log of 12 month moving sums of dividends of the S&P 500 index and the log of prices. Earnings Price Ratio (ep) is the difference between the log of 12-month moving sums of earnings of the S&P 500 index and the log of prices. Book-to-Market Ratio (bm) is the ratio of book value to market value for the Dow Jones Industrial Average. Net Equity Expansion (ntis) is the ratio of 12-month moving sums of net issues over the end-of-year market capitalization at the NYSE. Treasury-bill Rate (tbl) is the 3-Month Treasury Bill: Secondary Market Rate from the FRED. Term Spread (tms) is the difference between the yield on long-term U.S bonds and the Treasury-bill Rate. Default Yield Spread (dfy) is the difference between BAA and AAA-rated corporate bond yields. Stock Variance (svar) is the sum of squared daily returns on the S&P 500 index.

The panel of stock-level characteristics ($z_{i,t}$) is constructed by multiplying for each stock at each time all individual characteristics ($c_{i,t}$) by the macroeconomic variables and a constant (x_t): $z_{i,t} = x_t \otimes c_{i,t}$.

Due to computational and time constraints, not the whole dataset could be used. Therefore, the data used is restricted to the period from January 2001 to December 2017, this is equal to a total of 204 months (or 17 years). Furthermore, to reduce the number of variables further, only stocks which are present in at least 197 months whole period are selected. This leads to 1380 different stocks being considered. Note that this selection is not random and is suspected to have a large impact on the results. For data cleaning issues, all variables where more than 10% of the total number of observations have missing values are deleted and the remaining missing values in the data set are replaced by the median of the cross-section. This leads to 77 stock-level and eight macroeconomic predictors. Deleting a couple of predictors with all values being equal to 0 gives a total of 675 explanatory variables.

In this empirical study the same methods as in the simulation are used. Since the methods with a Huber-loss did not seem to perform significantly better (or even worse) than the ones with a classic l_2-loss, they are not included.

In Table 6 one can find the out-of-sample performance of the different method applied to the selected data for four different time aggregations: 1-month ahead forecast, 3-month ahead forecast, 6-month ahead forecast and 12-month ahead forecast. One can see that OLS performs bad in all four situations. In contrast with the literature, none of the methods improves with a larger forecast. Most methods do not change much with a larger horizon. Those who do

Table 7: Diebold-Mariano Test

p	Lasso	Ridge	ENet	RF	GBRT	NN1	NN2	NN3
OLS	**-5.99**	**-7.26**	**-5.99**	**-6.02**	**-6.25**	**-5.76**	**-5.93**	**-5.9**3
Lasso		**4.00**	-0.95	-1.17	1.26	1.68	0.48	0.25
Ridge			**-4.00**	**-4.05**	**-4.13**	**-3.59**	**-3.93**	**-3.93**
ENet				-1.09	1.26	1.73	0.51	0.28
RF					1.67	1.83	1.01	0.79
GBRT						-0.11	-0.96	-1.03
NN1							-0.98	-1.24
NN2								-1.19

Note: This table gives the adapted Diebold-Mariano test statistics to compare methods pairwise. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. It can be interpreted as how a row-method performs compared with a column-method. In bold are values with a significance level of 5% or better.

are OLS and Ridge. Ridge is already outperformed by the other methods (beside OLS) in the one-month ahead scenario but it becomes worse the larger the horizon. Interestingly, the neural networks seems to perform quite good, especially the deepest one with three layers. This is in contradiction with the results from the simulations.

In Table 7, the nine methods are compared in pairwise Diebold-Mariano tests. The first result is that OLS is significantly worse than all other methods, this was expected. Furthermore, Ridge is less accurate than all methods beside OLS. All other results are not significantly different. This is interesting especially for the neural networks which underperformed in the simulations.

6. Conclusion

Predictability of asset returns is one of the big questions in the field of asset pricing. Many models with different predictors have been proposed over the last decades. Lately machine learning started to be applied by researcher to the field. The high dimensional methods which are combined under the name machine learning are constructed for prediction and permit to researchers to use a large amount of predictors. Recent research gives some evidence that the predictions of asset returns can indeed be improved by using high dimensional techniques.

The fundamental work of Gu, Kelly, and Xiu (2018) tries to give an overview over different methods. Using data from the US stock markets, they find impressively good predictions. This work tried to rebuild their results and to test their robustness to variations of the predictive task, especially time aggregation. In a first part a Monte Carlo simulation was performed. Varying the error term of the simulated returns did change the values of the goodness of the prediction, but not the order between methods. Lasso and Elastic Net were the most accurate predictors, followed by Random Forest, GBRT, Ridge and shallow Neural Networks while pooled OLS over all predictors did the worst. Surprisingly, using Huber Loss instead of L_2-Loss to put less weight on outliers, did not improve the predictions.

While these results are encouraging, comparing different time aggregations (1, 3, 6 and 12-months ahead forecasts) is more ambiguous. While three-months forecasts are more accurate than one-month ones, this is surprisingly not the case for all methods for even larger horizons. Still, the order of the methods stays more or less the same for one, three and six months forecasts. Only for a twelve-months ahead forecast, GBRT becomes the most accurate. In a last step,

the predictions for different sample sizes are compared. Not surprisingly, all methods perform better, the larger the sample and the order of the methods is robust towards the sample size. This is true even though not all methods benefit equally from the increased sample. Especially neural networks improved largely when the sample size is increased.

In the second part, an empirical study was conducted on a small sample of US data over a period from 2001 to 2017. Again, the main interest is not the absolute value of the predictions but the relative accuracy of the methods compared with each other. The forecasts were calculated for one, three, six and twelve-months aggregations. As expected, pooled OLS performed bad whatever the time aggregation. Surprisingly, in contrast with the literature, none of the methods improved with a larger aggregation. Comparing the methods one by one, only OLS and Ridge performed badly, all others were not significantly different among each other.

The simulations and the empirical study have some limitations. For the simulations, one would like to have more Monte Carlo repetitions and for the empirical study the sample should be increased and all observations included. Both limitations could falsify the results and rerunning the simulations and the empirical study with more time and computational capacity would be a first improvement of the work.

Furthermore the splitting method used here is very simple, just divide the initial data in three disjoint subsamples for training, validation and testing. In a future work, it would be interesting to investigate how the results change with the proportions of the subsamples. Finally one could try to use more sophisticated and interesting methods such as rolling or fixed windows.

References

Banz, Rolf W (1981). "The relationship between return and market value of common stocks". In: *Journal of financial economics* 9.1, pp. 3–18.

Bhandari, Laxmi Chand (1988). "Debt/equity ratio and expected common stock returns: Empirical evidence". In: *The journal of finance* 43.2, pp. 507–528.

Black, Fischer (1972). "Capital market equilibrium with restricted borrowing". In: *The Journal of business* 45.3, pp. 444–455.

Bradley, Brendan O and Murad S Taqqu (2003). *Financial risk and heavy tails*, pp. 35–103.

Breiman, L. et al. (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Campbell, John Y and Samuel B Thompson (2007). "Predicting excess stock returns out of sample: Can anything beat the historical average?" In: *The Review of Financial Studies* 21.4, pp. 1509–1531.

Cochrane, John H (2007). "The dog that did not bark: A defense of return predictability". In: *The Review of Financial Studies* 21.4, pp. 1533–1575.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Diebold, Francis X and Roberto S Mariano (1995). "Comparing Predictive Accuracy". In: *Journal of Business & Economic Statistics* 13.3.

Efron, Bradley and Trevor Hastie (2016). *Computer age statistical inference*. Vol. 5. Cambridge University Press.

Elliott, Graham and Allan Timmermann (2016). *Economic Forecasting*. Princeton University Press.

Fama, Eugene F and Kenneth R French (1992). "The cross-section of expected stock returns". In: *the Journal of Finance* 47.2, pp. 427–465.

– (1993). "Common risk factors in the returns on stocks and bonds". In: *Journal of financial economics* 33.1, pp. 3–56.

Fama, Eugene F and James D MacBeth (1973). "Risk, return, and equilibrium: Empirical tests". In: *Journal of political economy* 81.3, pp. 607–636.

Feng, Guanhao, Stefano Giglio, and Dacheng Xiu (2017). "Taming the factor zoo". In: Chicago Booth Research Pape 17-04.

Ferreira, Miguel A and Pedro Santa-Clara (2011). "Forecasting stock market returns: The sum of the parts is more than the whole". In: *Journal of Financial Economics* 100.3, pp. 514–537.

Freyberger, Joachim, Andreas Neuhierl, and Michael Weber (2017). *Dissecting characteristics nonparametrically*. Tech. rep. National Bureau of Economic Research.

Green, Jeremiah, John RM Hand, and X Frank Zhang (2017). "The characteristics that provide independent information about average us monthly stock returns". In: *The Review of Financial Studies* 30.12, pp. 4389–4436.

Gu, Shihao, Bryan T Kelly, and Dacheng Xiu (Apr. 2018). "Empirical Asset Pricing Via Machine Learning". In: URL: `https://ssrn.com/abstract=3159577`.

Hansen, Bruce (2017). *Econometrics*. University of Wisconsin. URL: `http://www.ssc.wisc.edu/~bhansen/econometrics/`.

Heaton, Jeff (2008). *Introduction to Neural Networks for C#, 2Nd Edition*. 2nd. Heaton Research, Inc.

Hoerl, Arthur E and Robert W Kennard (1970). "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1, pp. 55–67.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5, pp. 359–366.

Huber, Peter J. (Mar. 1964). "Robust Estimation of a Location Parameter". In: *Ann. Math. Statist.* 35.1, pp. 73–101.

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

Jensen, Michael C, Fischer Black, and Myron S Scholes (1972). "The capital asset pricing model: Some empirical tests". In: *Studies in the Theory of Capital Markets*. Ed. by editor.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv:1412.6980*.

Kozak, Serhiy, Stefan Nagel, and Shrihari Santosh (Nov. 2017). *Shrinking the Cross Section*. Working Paper 24070. National Bureau of Economic Research.

Lewellen, Jonathan et al. (2015). "The Cross-section of Expected Stock Returns". In: *Critical Finance Review* 4.1, pp. 1–44.

Lintner, John (1965). "Security prices, risk, and maximal gains from diversification". In: *The journal of finance* 20.4, pp. 587–615.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2012). *Foundations of machine learning*. MIT press.

Moritz, Benjamin and Tom Zimmermann (2016). "Tree-based conditional portfolio sorts: The relation between past and future stock returns". In:

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

Prechelt, Lutz (2012). "Early stopping - but when?" In: *Neural networks: tricks of the trade*. Springer, pp. 53–67.

Sharpe, William F (1964). "Capital asset prices: A theory of market equilibrium under conditions of risk". In: *The journal of finance* 19.3, pp. 425–442.

Stathakis, D (2009). "How many hidden layers and nodes?" In: *International Journal of Remote Sensing* 30.8, pp. 2133–2147.

Tibshirani, Robert (1997). "The lasso method for variable selection in the Cox model". In: *Statistics in medicine* 16.4, pp. 385–395.

Tsai, Chih-Fong et al. (2011). "Predicting stock returns by classifier ensembles". In: *Applied Soft Computing* 11.2, pp. 2452–2459.

Welch, Ivo and Amit Goyal (2007). "A comprehensive look at the empirical performance of equity premium prediction". In: *The Review of Financial Studies* 21.4, pp. 1455–1508.

Wilson, Ashia C et al. (2017). "The marginal value of adaptive gradient methods in machine learning". In: *Advances in Neural Information Processing Systems*, pp. 4151–4161.

Zou, Hui and Trevor Hastie (2005). "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320.

A. Tables Simulations

Table 8: R^2 for different DGPs

$\epsilon_{i,t+1}$	$0.05 * t(5)$		$N_{0.5}(0, 0.05^2)$		$N(0, 0.05^2)$	
$R^2(\%)$	IS	OOS	IS	OOS	IS	OOS
Oracle	5.10	2.88	6.32	4.29	6.64	3.24
OLS	6.70	-0.52	8.17	-0.23	8.64	-1.18
OLS +H	6.23	-0.76	7.65	-0.89	8.15	-2.13
Lasso	4.46	2.57	6.06	3.42	5.88	2.97
Lasso +H	4.41	2.58	6.05	3.25	6.14	2.90
Ridge	6.52	1.10	7.96	1.70	8.36	1.47
Ridge +H	6.25	1.10	7.67	1.39	8.09	1.07
ENet	4.39	2.48	6.07	3.41	6.07	2.97
ENet +H	4.31	2.55	6.15	3.21	6.12	2.91
rf	7.00	1.99	9.22	2.34	9.37	2.06
GBRT	6.48	2.23	8.29	2.32	8.74	2.31
GBRT +H	6.55	2.11	8.57	2.03	8.46	2.42
NN1	5.15	2.46	8.39	2.92	8.37	0.44
NN2	5.41	1.91	8.00	3.28	8.55	-1.87
NN3	3.26	1.32	5.73	1.61	6.07	-4.96

Note: This table gives an overview of in-sample (IS) and out-of-sample (OOS) R^2s for one month ahead forecasts of simulated individual stock returns for three different data generating processes. The DGPs are identical upon the distribution of the error term $\epsilon_{i,t+1}$. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. +H indicates that Huber Loss is used instead of a L2 loss function. Additionally, Oracle specifies a pooled OLS regression only with the three real input variables.

Table 9: Out-of-sample R^2 for different time aggregations

R^2(%)	1 month	3 months	6 months	12 months
Oracle	3.24	4.88	5.89	5.98
OLS	-1.18	1.24	-1.09	0.73
OLS +H	-2.13	0.31	-2.05	-0.23
Lasso	2.97	4.49	5.29	4.44
Lasso +H	2.90	4.54	5.10	4.51
Ridge	1.47	3.32	3.72	2.13
Ridge +H	1.07	2.91	3.32	1.81
ENet	2.97	4.53	5.28	4.39
ENet +H	2.91	4.59	4.98	4.41
RF	2.06	3.83	4.05	3.98
GBRT	2.31	3.91	4.91	4.98
GBRT +H	2.42	4.00	5.03	4.85
NN1	0.44	1.71	-2.61	-4.05
NN2	-1.87	-6.71	-10.90	-12.83
NN3	-4.96	-9.94	-12.31	-13.38

Note: This table gives an overview of out-of-sample R^2s (in %) for one month up to 12 months aggregated forecasts of simulated individual stock returns. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. +H indicates that Huber Loss is used instead of a L2 loss function. Additionally, Oracle specifies a pooled OLS regression only with the three real input variables.

Table 10: Out-of-sample R^2 for sample sizes

R^2(%)	150 months	300 months	450 months
Oracle	3.24	4.50	5.24
OLS	-1.18	3.32	4.66
OLS +H	-2.13	2.97	4.41
Lasso	2.97	4.59	5.09
Lasso +H	2.90	4.55	5.00
Ridge	1.47	3.86	4.84
Ridge +H	1.07	3.73	4.67
ENet	2.97	4.61	5.10
ENet +H	2.91	4.57	4.99
RF	2.06	3.68	4.35
GBRT	2.31	3.95	4.77
GBRT +H	2.42	3.95	4.79
NN1	0.44	3.97	4.82
NN2	-1.87	3.55	4.43
NN3	-4.96	3.38	4.30

Note: This table gives an overview of out-of-sample R^2s (in %) for forecasts of simulated individual stock returns for different time periods. The periods are $T = 150$, $T = 300$ and $T = 450$. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers. +H indicates that Huber Loss is used instead of a L2 loss function. Additionally, Oracle specifies a pooled OLS regression only with the three real input variables.

B. Tables Empirical Study

Table 11: Out-of-sample Loss for different time aggregations

R^2	1 month	3 months	6 months	12 months
OLS	-21.78	-305.88	-425.91	-622.84
Lasso	-0.03	-0.66	-0.97	-1.78
Ridge	-8.47	-20.87	-52.31	-109.72
ENet	-0.02	-0.46	-0.36	-0.23
RF	0.08	-0.79	-1.14	-2.12
GBRT	-0.57	-3.66	-2.25	-3.08
NN1	-0.50	-0.92	-0.55	-17.32
NN2	-0.12	-0.51	-0.62	-6.68
NN3	-0.07	-0.44	-0.32	-2.60

Note: This table gives an overview of out-of-sample R^2 for one month up to 12 months aggregated forecasts for returns of selected assets between 2001 and 2017. Methods used are pooled OLS (OLS), Lasso and Ridge regression, Elastic Net (ENet), Random Forest(RF), Gradient Boosted Regression Trees (GBRT) and Neural Nets with one (NN1) up to 3 (NN3) layers.