Erasmus University Rotterdam

Erasmus School of Economics

Bachelor Thesis Econometrics and Operations Research

Improving Mean Variance Portfolio Construction With Machine Learning Stock Price Forecasts

A different approach to forecasting the sample mean for mean variance optimization

Hans Becker *(433128)*

Supervisor:
Dr. X. Xiao
Second assessor:
S.H.L.C.G. Vermeulen

Date final version: July 5, 2020

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.

Abstract

A well known problem in constructing mean-variance portfolios is obtaining good forecasts for the sample mean of stock returns. This research will use the machine learning methods of Gu, Kelly and Xiu (2020) to forecast equity risk premia and use these forecasts instead of the first sample moment to construct mean-variance portfolios. The machine learning methods implemented use dimension reduction, penalized objective functions, tree ensemble methods and deep learning methods. A large dataset with 94 stock level characteristics is used to train the models. It is shown that machine learning models are able to outperform the OLS benchmark, but are not able to outperform the more parsimonious OLS-3 model with 3 preselected variables when it comes to constructing mean-variance optimized portfolios. OLS-3 is the only model that is able to outperform using the sample mean for mean-variance portfolios. It is also shown that some machine learning methods are not particularly suited to forecast returns to use in the mean-variance framework, due to the fact that mean-variance optimization can lead to extreme weights and in combination with forecast errors might lead to poor performance of portfolios.

Contents

1 Introduction

The question if stock returns are predictable has been around for ages. Measuring the premium received on equity is a thoroughly researched and written about subject with many relevant applications that has resulted in a 'zoo' of possible predictive factors. Many different methods and possible indicators have already been tried to forecast equity risk premiums. Recently machine learning methods have started to gain traction in the field of quantitative finance for forecasting equity risk premiums (Gu, Kelly, & Xiu, 2020). This research will use machine learning forecasts of equity risk premiums to construct mean-variance optimized portfolios as proposed by Markowitz (1952) to see if this approach is able to beat the sample mean approach. Using equity premium forecasts based on a large amount of predictive characteristics to construct mean-variance portfolios has not been done before and this will be this papers contribution to the literature.

There is a lot of literature on estimation errors in both the sample mean and sample variance-covariance matrix. Mean-variance portfolios are sensitive to small perturbations in the first and second sample moments such that small estimation errors can lead to extreme weights. It has been shown that the sample mean is most vulnerable to estimation errors (Merton, 1980), and this research will replace the sample mean as a naive forecast of returns by machine learning forecasts of equity risk premiums. The idea is to see if machine learning methods can improve upon the sample mean when estimating mean-variance portfolios. Even though there is a lot of literature on estimation errors in mean-variance optimization and on shortcomings of the methods, there is still a case to make for this path of investigation. Many investors are still risk-optimizers in the sense that they want maximum return with minimal risk. But since investors will not only look at the first and second sample moment when making investment decisions but also look for other signals when investing, using a large set of characteristics to forecast returns is a good approach.

The machine learning methods used in this research are Penalized Linear (Elastic-Net), Principal Components Regression, Partial Least Squares, Random Forests and Neural Networks. These models each have their own pros and cons and will be benchmarked against Ordinary Least Squares. Since OLS is known to be prone to overfit in a high-dimensional setting, a smaller OLS model will also be estimated, using only 3 preselected characteristics.

2 Literature

This research is based on research done by Gu, kelly and Xiu (2020) which examines multiple machine learning methods for forecasting equity risk premiums. They show that, with a large set of predictor variables, considerable gains can be made by applying machine learning methods. Especially regression trees and neural networks seem promising, they attribute this to the fact that these methods are universal approximators. They use a large set of predictors, containing 94 stock level characteristics, 8 macro-economic variables and 74 industry dummy variables. The stock level characteristics and macro variables are combined with a kronecker product such that only stock level characteristics enter the models, resulting in a whopping

920 predictor variables. With the forecasts made by the machine learning methods several types of portfolios are constructed. These include common factor portfolios and subcomponents of these factor portfolios. Next they construct machine learning portfolios by sorting the forecasts into ten deciles and use equal weights and value weights for computing the portfolio return of these ten deciles. They also construct a net-zero-investment portfolio by buying the highest decile and selling the lowest decile. They show that the gains that can be made by applying machine learning are substantial. This research will extend upon the work done by Gu, Kelly and Xiu (2020) by investigating whether further gains in portfolio performance can by made by using the equity premium forecasts as a replacement for the historical average mean of returns in the mean-variance portfolio. All machine learning methods except Gradient Boosted Regression Trees and Generalized Linear Models will be replicated in this research. In some cases estimation methods will have to be adjusted to fit the scope of this research, since time and computing power are limited. The machine learning portfolios constructed by Gu, Kelly and Xiu (2020) will also be replicated, to get an indication of the performance of the various methods on an aggregate level.

Work done by DeMiguel et al (2007) has shown that mean-variance optimization does not beat a naive benchmark with equal weights. They find that the gains made from optimally choosing a portfolio are offset by estimation errors in the first and second sample moment. They discuss a plethora of approaches for solving this problem. There is hope though, Merton (1980) shows that estimates of the variance-covariance matrix are much more accurate than their respective expected return estimates. By combining improved estimates of expected returns with mean-variance optimization we might see an improvement in portfolio performance over using the sample mean.

There has been quite a bit of work done on improving upon the mean-variance model by implementing machine learning methods. Almost all of the approaches tried center around finding better estimates for the first and second sample moments. Forecasting equity risk premia is hardly considered in the literature. The approach taken by Ban, Karoui and Lim (2018) is a form of regularization and shrinkage to deal with estimation errors in the sample mean and sample variance. Fastrisch, Paterlini and Winker (2015) take a similar approach by penalizing the 1-norm of the portfolio weights vector, a technique known as ℓ_1-regularization. This performs both regularization and variable selection, mitigating overfitting. Ao, Li and Zheng (2019) derive a new approach for constructing an optimal mean-variance portfolio based on the sample mean and sample variance-covariance. They are able to attain maximum expected return by combining the mean-variance problem with high-dimensional sparse-regression methods.

This research will try to expand upon the existing literature by combining improved forecasts of stock prices based on a large set of predictor variables and plugging these into the mean-variance framework. There has been little research in this direction yet, and it makes sense to further investigate. Investors considering the entire market will usually not only base their decisions on the first and second sample moments, which are backward looking. Investors are both forward looking and backward looking at the same time, trying to identify indicators that may signal economic down or upturn or firm specific indicators that may move an

investor to abandon certain stocks or more heavily invest in them.

3 Data

This research draws from the collection of predictive signals that is provided by Gu, Kelly and Xiu (2020) which covers the period March 1957 to December 2016. This collection of 94 predictive signals or characteristics is based on the work of Green et al. (2017). Approximately 30% of the characteristics data is missing which are filled in with the cross-sectional median at each month for each stock, following the approach of Gu, Kelly and Xiu (2020). Observations for which no cross-sectional median can be computed for a characteristic have to be dropped. The characteristics 'realestate' and 'secured' are not filled by the cross-sectional mean up to December 1986, so all observations up to 1986 are also dropped. Stock price data is retrieved from CRSP based on the permanent company number ('permno') provided in the dataset of Gu, Kelly and Xiu. After cleaning the data there remain 2.601.336 observations with on average 6774 observations per month running from January 1987 to December 2016. A plot of the number of companies per month is provided in figure 3. What stands out is the peak in the number of listed companies before the dot-com bubble of the 2000s after which the number of listed companies declines. This downward trend is know as the 'U.S. listing gap' (Whitten, 2015).

To speed up learning time, this research does not use the eight macroeconomic variables detailed in Welch and Goyal (2008). Gu, Kelly and Xiu use these macroeconomic variables to construct a kronecker product with the characteristics such that only stocks level characteristics enter the models. The computation of this kronecker product and the subsequent model fitting is an expensive operation, especially using this large dataset. This research therefore also foregoes this procedure too aid in learning time of the models, which is considerable given the size of the dataset. The same goes for the industry dummy variables based on the SIC-codes as done in Gu, Kelly and Xiu (2020).

4 Methodology

4.1 Forecasting Equity Risk Premiums

First, the general structure for forecasting equity risk premiums will be discussed, after which a short summary of each of the used machine learning methods will be given. Explanation of the various machine learning methods will be kept short and where necessary readers are informed on further reading on a subject. Implementation of all the models is done in Python, due to the availability of machine learning packages such as Scikit-Learn and Keras. A detailed description of the used packages is provided in table 4.

Lets consider the equity risk premium which is defined as follows

$$r_{i,t+1} = E_t(r_{i,t+1}) + \epsilon_{i,t+1} \tag{1}$$

With

$$E_t(r_{i,t+1}) = g(z_{i,t}) \tag{2}$$

Here stocks are indexed from $i = 1, ..., N_t$ (note that the number of stocks per month can vary) and months are indexed from $t = 1, ..., T$. The conditional expectation of returns depends on the specification of the functional form of $g(\cdot)$ with $z_{i,t}$ being a P-dimensional vector of predictor variables. The idea is to find a specification of $g(\cdot)$ that maximizes the out-of-sample explanatory power of a model.

4.1.1 Sample splitting and out-of-sample prediction

Sample splitting is important in the case of machine learning due to the need to optimize, or 'tune' as it is commonly referred to in machine learning literature, a models' hyperparameters. Hyperparameters don't directly interact with the data, but are essential to the functional form of a model. In the case of, for example, dimension reduction methods a common hyperparameter is the number of dimensions K to use in the final model. The final hyperparameter is found by 'trying' multiple dimensions. The specific algorithm for finding hyperparameters for each of the models will be discussed in the subsequent sections.

The sample is split into a training sample, validation sample and an out-of-sample section for forecasting. The training sample is used for actually estimating the model and the validation sample is used for tuning a models hyperparameters. A model is first estimated using the training sample after which predictions are made with the validation sample from which a loss function is computed. The aim is to find hyperparameters for which the loss is minimized on the validation sample, in the hope of improving the out-of-sample performance. This is done by adaptively changing the values of the hyperparameters through some algorithm and recomputing the validation loss for each new set of hyperparameters.

Because machine learning methods are computationally intensive, the models are only refitted once a year. The length of the training sample is 10 years, that of the validation sample 5 years. Since the sample runs from 1986 to 2016, this gives 15 years of out-of-sample observations. The estimated models are used to compute one month ahead forecasts for the next out-of-sample year. To fully illustrate this approach, consider the following example; When a model is trained on data from January 1987 to December 1997 and validated on data from January 1998 to December 2001 the out of sample forecasts run from February 2002 to January 2003, such that January 2002 is used to forecast February 2002 and so on. This approach ensures that out-of-sample forecasting is done solely with data that hasn't entered the model in any way, avoiding forward-looking bias.

4.1.2 Ordinary Least Squares

One of the most basic specifications of $g(\cdot)$ is a linear specification

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

Where θ is the vector of parameters. The objective function to be minimized is the following

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

Minimizing this objective function gives ordinary least squares (OLS). OLS has the drawback that it handles 'fat tails' poorly, which, in the context of financial return data which exhibits fat tails, will result in inefficient forecasts. OLS is also known to be prone to overfitting in a high dimensional context, suggesting the use of regularization methods. This model serves as a benchmark to compare the machine learning methods against. Also, a model with linear specification with only three characteristics is fitted to compare to OLS with all characteristics as done in Gu, Kelly and Xiu (2020). The characteristics that enter this OLS-3 model are 'book-to-market', 'value' and 'momentum', which are considered to be the most important stock characteristics in the literature (Lewellen, 2015). Since OLS doesn't require tuning of hyperparameters, the validation sample is not used.

4.1.3 Weighted Least Squares

As stated, financial return data exhibits fat tails and is prone to overfitting, so it makes sense to look at a weighted least squares estimates replacing objective function (4) by

$$\mathcal{L}_W(\theta) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} w_{i,t} \big(r_{i,t+1} - g(z'_{i,t}\theta)\big)^2 \tag{5}$$

Where each observation is weighted by $w_{i,t}$. To mitigate the effect of large outliers on the estimates, Gu, Kelly and Xiu (2020) make use of the Huber robust objective, which is defined as follows

$$\mathcal{L}_H(\theta) = \frac{1}{NT} = \sum_{i=1}^{N} \sum_{t=1}^{T} H\big(r_{i,t+1} - g(z_{i,t}; \theta, \xi)\big) \tag{6}$$

Where $H(\cdot)$ is defined as

$$H(x;\xi) = \begin{cases} x^2 & \text{if } |x| \leq \xi \\ 2\xi|x| - \xi^2 & \text{if } |x| > \xi \end{cases} \tag{7}$$

The Huber function in (7) is a combination of a squared loss function for small errors like in OLS, and an absolute loss function for larger errors. This results in smaller weights for large errors giving them less influence on the coefficient estimates. Huber regression doesn't have hyperparameters that need to be optimized, so the validation sample for this method is left untouched.

4.1.4 Penalized Linear

To prevent OLS from overfitting it is also possible to mechanically deteriorate the objective function by adding a penalty term. The idea behind this is that since OLS is prone to overfitting, it will fit a specification that is 'too complicated' resulting in a model that performs well in-sample but responds poorly to new data. Penalizing the objective function of OLS 'dumbs down' the model, eventually resulting in a

more parsimonious model that performs better out of sample. The process of dumbing down a model is called regularization and is the first part of Elastic-Net. Another way of improving model performance is by performing variable selection and thus reducing noise. This can result in considerable improvements, especially in the context of stock return characteristics which can be highly correlated. The basic form of penalized linear regression is the same as in (3) but has a penalty term appended:

$$L(\theta\ ;\ \cdot) = L(\theta) + \phi(\theta\ ;\ \cdot) \tag{8}$$

There are many choices for the penalty term $\phi(\theta\ ;\ \cdot)$, Gu, Kelly and Xiu (2020) adopt the elastic net penalty (Zou & Hastie, 2005) which performs variable selection and regularization and is a combination of LASSO and Ridge regression

$$\phi(\theta; \lambda, \rho) = (1-\rho) \underbrace{\lambda \sum_{j=1}^{P} |\theta_j|}_{\text{LASSO penalty}} + \rho \underbrace{\frac{1}{2}\lambda \sum_{j=1}^{P} \theta_j^2}_{\text{Ridge penalty}} \quad \lambda \geq 0, \theta \geq 0 \tag{9}$$

This penalty term involves two hyperparameters; λ and ρ[1]. ρ controls the tendency of the modeller to favor one method over the other. Setting $\rho = 0$ gives the LASSO penalty and setting $\rho = 1$ gives Ridge regression. This research adopts the approach of setting $\rho = 0.5$ as done by Gu, Kelly and Xiu (2020). λ is estimated by cross-validation. The sklearn model used doesn't allow for precise specification of the validation sample, but rather lets the modeler choose the fraction that is used for validation. So the model is given the training and validation sample as training sample and the validation fraction is set to 0.33.

4.1.5 Principal Components Regression and Partial Least Squares

Next two dimension reduction techniques are discussed. These methods aim at reducing the number of predictor variables while trying to capture as much signal in the reduced variables as possible. The first approach, principal components regression (PCR), aims to find linear combinations of the predictor variables that best preserve the covariance structure of the predictor variables. This is a two step approach, where first principal components analysis (PCA) is applied after which the components with the most explanatory power are regressed on the returns with OLS. Partial least squares (PLS) differs from PCR because it not only takes into account the set of predictors Z, but also considers the returns to be estimated R. Basically what PLS tries to do is try to find a linear combination of the predictor variables that maximizes the predictive power for the returns, whereas PCR tries to select a linear combination of the predictors that retains the most variation.

Both models use the vectorized version of equations (1) - (3)

$$R = Z\Theta + E \tag{10}$$

[1]In the literature ρ often also gets referred to as ℓ_1-ratio

Where R is the $NT \times 1$ vector of returns $r_{i,t+1}$, Z is the $NT \times P$ stacked matrix of predictor variables $z_{i,t}$ and E is the $NT \times 1$ vector of residuals.

PCR and PLS both reduce the original number of predictors P down to a number of linear combinations K such that the forecasting model for both methods is given by

$$R = (Z\Omega_K)\Theta_K = \tilde{E} \tag{11}$$

Where Ω_K is a $P \times K$ matrix that reduces the original predictor matrix Z. PCR chooses the weight matrix Ω_K recursively. The j^{th} linear combination solves

$$w_j = \arg\max_w \ Var(Zw) \quad \text{s.t.} \quad w'w = 1, \quad Cov(Zw, Zw_l) = 0, \quad l = 1, 2, ..., j-1 \tag{12}$$

The weights used to construct the j^{th} PLS component solve

$$w_j = \arg\max_w \ Cov^2(R, Zw) \quad \text{s.t.} \quad w'w = 1, \quad Cov(Zw, Zw_l) = 0, \quad l = 1, 2, ..., j-1 \tag{13}$$

Note the returns entering in (13) whereas (12) only considers the variance matrix of the predictor variables Z. Gu, Kelly and Xiu (2020) are unclear about how the number of linear combinations K is chosen. This research adaptively estimates K by starting at $K = 1$ and gradually increasing the number of components. For each K a model is estimated. With this model forecasts are made on the validation sample. These forecasts are used to compute the Mean Squared Prediction Error (MSPE) which is given by

$$\text{MSPE} = \frac{1}{N} \sum_{i=1}^{N_t} \sum_{t=1}^{T} (r_{i,t} - \hat{r}_{i,t})^2 \tag{14}$$

. The number of components that give the lowest MSPE is chosen for the final model. To improve estimation time, an early stopping rule is implemented. If the MSPE doesn't improve for 10 iterations, the best model up until then is chosen. The maximum number of components than can be chosen is set to 50. A detailed description of how K is determined is given in algorithm 1

4.1.6 Random Forests

Random Forests (RF) are quite a different set of models compared to the models discussed so far. Gradient Boosted Regression Trees (GBRT) are not considered, unlike in Gu, Kelly and Xiu (2020). Unfortunately the *Sci-kit learn* implementation is not fast enough for large samples and without parallel computing this becomes infeasible. *Sci-kit learn* does offer an experimental version of is GBRT estimator called Histogram-Based Gradient Boosting but this does not offer the desired loss funcions yet.

RF is form of regression trees, complemented with ensemble learning methods to mitigate overfitting which standard regression trees are prone to. Regression trees split observations into rectangular subspaces based on the values of the predictor variables and then fit a simple model to the data in each rectangle, such as the average over all predictor variables. The splits, or branches, in the tree are chosen such that the forecast error is minimized. An advantage of regression trees is that they can represent both continuous

and discrete data, in the latter case it is called a decision tree, and the interpretation of estimation results is very straightforward. They are also good at modeling non-linearities in the data. As mentioned, regression trees are prone to overfitting due to fitting large, deep trees and are generally not robust to small changes in the data. Finding an optimal tree is shown to be NP-complete (Hyvfil & Rivest, 1975).

The prediction tree with K leaves and depth L can be written as

$$g(z_{i,t};\theta,K,L)=\sum_{k=1}^{K}\theta_k\mathbf{1}_{\{z_{i,t}\in C_k(L)\}} \tag{15}$$

where $C_k(L)$ is one of K partitions of the data. Each partition has a vector of coefficients θ, such that when an observation falls into a partition, it is set equal to the sum of all coefficients in the partition. The coefficients, as stated, are usually the average of characteristics for the observations that fall into that partition.

The loss function used for computing the forecast error at each branch in the tree used by Gu, Kelly and Xiu (2020) is the "ℓ_2 impurity" which is defined as follows

$$H(\theta,C)=\frac{1}{|C|}\sum_{z_{i,t}\in C}(r_{i,t+1}-\theta)^2 \tag{16}$$

where $|C|$ denotes the number of observations in set C. Note that θ is equal to the forecast for $r_{i,t+1}$

RF combines the forecasts of many different trees. It selects B random samples with replacement and fits a tree to each of these samples. This procedure is called bootstrap aggregation or bagging. In each tree Random Forests also select a random subsample of the features to estimate the tree with. The idea behind this is that there may be a few very strong predictor variables on which a lot of trees may split. By random selecting a subsample of the predictors this problem is mitigated and correlation among fitted trees is reduced. This eventually gives a better predictor when all the trees are aggregated. The final tree is given by

$$\hat{g}(z_{i,t};L,B)=\frac{1}{B}\sum_{b=1}^{B}\hat{g}_b(z_{i,t};\hat{\theta}_b,L) \tag{17}$$

where $\hat{g}_b(\cdot)$ on the RHS of the equation is an individual decision tree as in (15). The important hyperparameters are the number of trees to be estimated B, the maximum depth of each tree and the number of features to randomly select. Random Forests do not make use of cross-validation, so the validation sample is left untouched.

4.1.7 Neural Networks

One of the most powerful machine learning methods are neural networks. These are highly suited to finding nonlinear interactions in the data. The neural networks considered by Gu, Kelly and Xiu (2020) are 'feed-forward' which means that the interconnected neurons of the network do not contain any cycles. They are constructed by combining an input layer of the predictors with multiple hidden layers, which manipulate the data in some prespecified way and an output layer that transforms the manipulated data into output.

All of the nodes in one layer are connected to all of the nodes in the next layer. Thus, each node takes the values of all of the preceding nodes as input and uses an 'activation function' combined with node specific parameters to compute a value for the node, which it passes on to the next layer. All of the outputs of the last layer are aggregated into an output forecast. Choosing the right number of hidden layers, nodes and the correct activation function is currently still more of an art form than an exact science, so Gu, Kelly and Xiu (2020) consider multiple variations of neural networks with the number of nodes in the hidden layers chosen according to the 'geometric pyramid' rule. The first neural network (NN1) has 1 hidden layer, with 32 neurons. NN2 has two hidden layers with 32 and 16 neurons. NN3 has 3 hidden layers with 32, 16 and 8 neurons. NN4 has 4 hidden layers with 32, 16, 8 and 4 neurons. NN5 has 5 hidden layers with 32, 16, 8, 4, and 2 neurons. The non-linear activiation function choosen by Gu, Kelly and Xiu is the rectified linear unit (ReLU) function. This activation function can results in 'dying' nodes. Due to the way missing values in the dataset are filled (cross-sectional median for each month) and the fact that the dataset is smaller this may result in forecasts that are very similar when characteristics are similar for different observations. To prevent this dying of nodes, LeakyReLU was introduced. It introduces the parameter α into the ReLu function which ensures that nodes always have a value. Usually α is chosen to be small, such that the effect on results is small, this research adopts $\alpha = 0.1$

$$\text{LeakyReLU}(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{otherwise} \end{cases} \tag{18}$$

A single layer neural network is given in figure 1. It has inputs $x[0], x[1], x[2]$, a hidden layer with three nodes $h[0], h[1], h[2]$ and an output layer with $\hat{y}$.

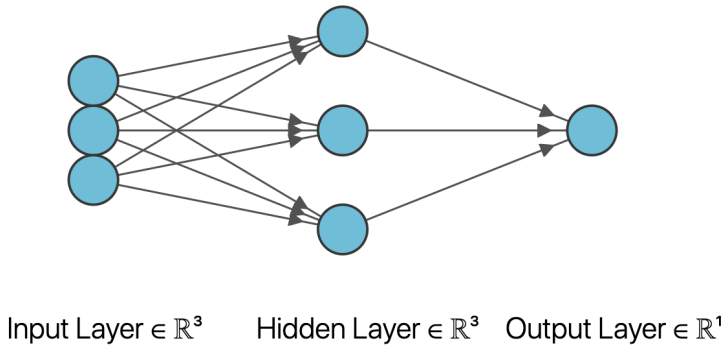

Figure 1: Single layer neural network

The value of each node is computed as follows

$$h[0] = \text{LeakyReLU}(w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + b[0])$$
$$h[1] = \text{LeakyReLU}(w[0,1] * x[0] + w[1,1] * x[1] + w[2,1] * x[2] + b[1])$$
$$h[2] = \text{LeakyReLU}(w[0,2] * x[0] + w[1,2] * x[1] + w[2,2] * x[2] + b[2])$$

where the weights w are learned from the data. The number of weights to be optimized increases dramatically with the number of hidden layers and nodes added to the network making a 'brute force' approach infeasible. The aim of a neural network is to minimize a loss-function, which in this case is the Mean Squared Error (MSE). This is done by gradient descent, which in this context is called back-propagation. The back-propagation algorithm depends on the chain rule for estimating the gradient which can easily be derived due to the way a neural network is constructed. The details of the back-propagation algorithm and fitting of the neural network are outside of the scope of this research, and I would like to refer to Hastie, Tibshirani and Friedman (2009) who give a detailed explanation of the inner workings of neural networks.

To aid estimation time, the ensemble method used by Gu, Kelly and Xiu, where multiple networks are trained with different seeds and forecasts are averaged over the model predictions is not implemented. Keras allows for precise specification of the validation sample, which is used to compute the out-of-sample MSE. This MSE is used for an early stopping rule, which says that if the MSE on the validation sample doesn't improve for 5 iterations, the best model up until then is chosen. These reduces overfit and estimation time. The maximum number of iterations of the gradient descent algorithm is set to 100 and is referred to as the number of 'epochs'.

4.2 Portfolio Construction

4.2.1 Mean Variance Optimization

The extension on Gu, Kelly and Xiu (2020) revolves around constructing mean-variance optimized portfolios using the machine learning forecasts instead of the mean average return. Markowitz (1952) derived an efficient portfolio that maximizes utility for investors who's only concern is the mean average return and the risk associated with these returns.

$$\max_{x_t} = x_t^T \mu_t - \frac{\gamma}{2} x_t^T \Sigma_t x_t \tag{19}$$

where x_t is the investors utility and γ the investors aversion to risk. Since the investors risk appetite is not relevant in this research it will be set to $\gamma = 1$. The solution to this optimization problem, such that utility is maximized, is $x_t = (1/\gamma)\Sigma_t^{-1}\mu_t$ with the vector of relative portfolio weights being

$$w_t = \frac{\Sigma_t^{-1}\mu_t}{\mathbf{1}_N \Sigma_t^{-1}\mu_t} \tag{20}$$

The strategy is implemented as follows. At time t all characteristics and returns up to r_t are known. Forecasts $\hat{r}_{t+1}$ are made and substituted for μ_t in 20. Σ_t is computed based on the returns known up until time r_t and weights w_t are computed. The portfolio return is then computed as $r_{p,t+1} = w_t * r_{t+1}$. Since we want to know if machine learning methods can provide better performance compared to using the sample mean, the same approach will be applied using the sample mean of returns.

Since computing a covariance matrix for a large number of stocks may result in large estimation errors and because unconstrained mean-variance weights can take extreme values the stocks are sorted into 5 groups by forecasts from high to low. Equal weights are chosen for the stocks within each group such that the return of one group is the average of all returns in the group. To compute the covariance matrix, the historical returns also have to be sorted into groups. This is done by taking the 'permno' of the stocks in each forecast group and selecting these from the historical returns and then again computing the groups return as the average of all returns in the group. This approach ensures that only stocks that are in each forecast group are used to compute the covariance matrix.

4.3 Performance Measures

To evaluate the forecasts of the various models the same performance measure as in Gu, Kelly and Xiu (2020) are applied: the out of sample R^2 which is defined as follows:

$$R^2 = 1 - \frac{\sum_{(i,t)\in T_3}(r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{(i,t)\in T_3} r_{i,t+1}^2} \tag{21}$$

Here T_3 means that the R^2 is computed from forecasts made with data that doesn't enter the model either in the training stage or when tuning model hyperparameters. It is important to note that the denominator is the sum of squared excess returns without demeaning. Gu, Kelly and Xiu argue that it doesn't make sense to compare the forecasts to historical average returns since historical returns underperform a naive forecast of zero due to the noisiness of historical average returns on the level of stock returns.

To evaluate the performance of the various portfolios constructed the following performance measures will be considered; the Sharpe ratio, maximum drawdown and average monthly turnover.

The Sharpe ratio (Sharpe, 1962) is a measure of risk adjusted return of an asset or portfolio. It is one of the most straight forward performance measures and very important for the mean-variance investor and is given by

$$SR_p = \frac{R_p - R_f}{\sigma_p} \tag{22}$$

Here R_p is the portfolio return, R_f is the return of the risk-free rate and σ_p is the standard deviation of the portfolio return. Since forecasts are of equity risk premiums, which are in excess of the risk-free rate, R_f will be set to 0. A higher Sharpe ratio implies a better performance.

Maximum drawdown is the maximum loss incurred from a peak before a portfolio is back up to the peak

level over the entire observation time. It gives a sense of the risk a portfolio carries and is defined as

$$\text{Maximum Drawdown} = \max_{0 \leq t_1 \leq t_2 \leq T} (Y_{t_1} - Y_{t_2}) \tag{23}$$

Where Y_t is the cumulative log return from date 0 to t.

Average monthly turnover is a measure of how much a portfolio has to be rebalanced. A 1/N portfolio generally has low turnover where other more sophisticated portfolios may have a larger turnover. Average monthly turnover is defined as follows.

$$\frac{1}{T} \sum_{t=1}^{T} \left(\sum_i \left| w_{i,t+1} - \frac{w_{i,t}(1 + r_{i,t+1})}{1 + \sum_j w_{j,t} r_{j,t+1}} \right| \right) \tag{24}$$

Here $w_{i,t}$ is the weight of stock i at time t. In this formulation it averages the absolute change in monthly portfolio weights.

5 Results

5.1 Predictive performance

	OLS	OLS-3	OLS-H	E-Net	PLS	PCR	RF	NN1	NN2	NN3	NN4	NN5
R^2	-0.462	0.233	-7.494	-0.141	0.229	-0.116	0.456	0.108	0.275	0.193	-2.211	0.025
Top 1000	-1.399	0.424	-31.834	-2.731	0.060	-1.015	0.541	0.208	0.595	0.099	-0.451	-0.058
Bottom 1000	0.059	0.294	-1.422	-0.007	0.462	0.129	0.692	0.261	0.318	0.322	0.007	0.108

Table 1: This table reports the monthly out-of-sample R^2 as defined in (21) for the entire panel of stocks. The bottom 1000 and top 1000 stocks are sorted by value

When evaluating the predictive power of the forecasts (see table 1) it is clear that most machine learning methods are able to outperform the OLS benchmark. What stands out is that OLS-3 with the preselected characteristics, 'book-to-market', 'value' and 'momentum' performs remarkably well. The only machine learning methods that are able to outperform OLS-3 are RF and NN2. Gu, Kelly and Xiu don't report results for OLS, but only for OLS-H. Since the dataset our research uses is significantly smaller in terms of the number of characteristics used it makes sense to also include OLS without Huber weights, which is actually able to outperform OLS with Huber weights. Of the five neural networks, the neural network with 2 hidden layers performs best. These results are more or less in line with Gu, Kelly and Xiu (2020) where all models were able to outperform the standard OLS benchmark. Where our results differ is that OLS-3 performs remarkably well, even better than some machine learning forecasts, where in Gu, Kelly and Xiu (2020) most machine learning forecasts were able to easily outperform the OLS-3 model. Based on the out-of-sample R^2 it is hard to tell whether the models are better at forecasting large or small value stocks, since this is different per model.

The performance of PLS is also remarkable, given the relative simplicity of the approach. In this context it performs better than PCR in contrast to Gu, Kelly and Xiu (2020), where PLS and PCR had comparable performance.

5.2 Portfolio performance

First a set of machine learning forecasts is constructed as in Gu, Kelly and Xiu (2020) to further analyze the forecasting performance. Since this research will sort stocks into 5 quantiles for the mean-variance section it makes sense to also do this for the machine learning portfolios, which differs from Gu, Kelly and Xiu who use 10 quantiles. When aggregating the stocks into 5 quantiles (table 2) the predicted returns increase monotonically as expected, since the quantiles are constructed by sorting the forecasts from high to low. This is also expected for the average realized return, but this not always the case, suggesting the models have a hard time correctly predicting the magnitude of the return. It must be said that this is also a notoriously hard problem and even predicting the sign of the return is a challenge. Also, the fact that the 8 macroeconomic variables are not included may play a part in the poor performance. Since these variables are able to give a monthly overview of the state of the economy, a lot of possibly useful information does not enter into forecasts. Nevertheless, the forecasts from RF and NN1-5 are better able to predict returns and the average forecasted return rises monotonically in most cases. Unfortunately Sharpe ratio's never reach high enough values to actually be interesting to consider for investing purposes. This is in contrast to Gu, Kelly and Xiu who are able to get better Sharpe ratios from the machine learning portfolios. Again, this may be attributed to the smaller dataset used in this research, especially leaving out the macro economic indicators.

Also included are the high minus low portfolios, where the top quantile is bought and the bottom quantile is sold to construct a zero net investment portfolio. None of the methods is able to outperform the highest quantiles, again suggesting that the magnitude of forecasts is off by quite a bit.

5.3 Extension: mean-variance portfolio performance

First, a few remarks about the results for mean-variance optimized portfolios in table 3. The 1M loss and 1M gain are computed as percentages, while the maximum drawdown is computed as the cumulative log return from time 0 to time t thus leading to the fact that 1M loss looks to be larger than the maximum drawdown, while they are computed differently. The turnover is computed from the quantile weights that are computed for each month. Each of these quantiles contains a large number of stocks and the stocks in each quantile can change from month to month. So this measure can be seen as an indication of how much the weights vary on average from month to month and not as an exact measure of turnover since this would require more detailed information on which stocks are in which quantile in each month. It is used as an indicator for the stability of the mean-variance weights. Also the performance measures for the S&P500 index are included to compare machine learning methods against the market.

When looking at the statistics of the mean-variance optimized portfolios a few things stand out. A number of models have high standard deviations and very high drawdowns and 1M losses. These models being OLS-H, PLS, PCR and RF. This can be attributed to the fact that mean-variance optimization can lead to extreme long and short positions. When the forecasts error is high for a particular month this can lead to significant effects in the choices of weights, leading to large losses. That there are problems in the weight choices is also reflected in the turnover which is relatively high for these methods. OLS-H, PLS and PCR are all based on OLS, which is very prone to overfitting the data resulting in estimation errors. Even though the R^2 of PLS is reasonable it still contains outliers that results in bad portfolio performance. From these results it can be concluded that there is still quite a bit of forecast error present and that these methods need further improvement. All in all it is obvious that these methods are not (yet) suited for portfolios construction.

We see a different picture when looking at OLS-3, ENet and NN1-5 which are quite well behaved, even when compared to the S&P 500 index. The best performing portfolio in terms of Sharpe ratio is by far the OLS-3 model. The NN1-5 models also perform well compared to the S&P 500 and also compared to the OLS benchmark. They also have a lower turnover compared to the OLS-3 model suggesting that the weights are more stable. What is also interesting to note is that both the Sharpe ratios and standard deviations have improved for NN1-5 and OLS-3 when comparing them to the sorted machine learning portfolios (table 2). When looking at the magnitude of forecasted returns, NN1-5 differs to the other models in that it is usually conservative in forecast returns compared to methods based on the linear model. This is reflected in the smaller 1M loss and 1M gain.

Unfortunately only the OLS-3 portfolio shows an improvement over using the sample mean in terms of Sharpe ratio and turnover. NN1-5 come closest in terms of Sharpe ratio but are not able to outperform the sample mean portfolio. NN1-5 are superior in terms of turnover. Comparing the results of the machine learning portfolios to using the sample means tells us that there is still quite some estimation error in the machine learning forecasts, leading to underperformance. This also tells us that on an aggregate level, using the sample mean as a forecast for returns is quite a reasonably approach. Especially considering its simplicity and easy interpretation compared to the machine learning methods.

These results suggest the use of restrictions on the weights of the mean-variance portfolios to mitigate the extreme returns. Nevertheless, the results from OLS-3 and NN1-5 are promising and further investigation with the full dataset might be interesting. In figure 2 a plot of the cumulative returns of OLS3, NN1-5 and the S&P index is included, showing that most methods are able to do better than the S&P500 which is commonly used as a benchmark for stock portfolios. The performance of the other portfolios is quite erratic, leading to extremely large positive or negative cumulative returns, hence they have not been included in the graph. It is not directly clear how to deal with these abberations. They are caused by estimation errors which result in a poor choice of mean-variance weights. It might also be useful to investigate certain shrinkage techniques for dealing with these estimation errors, although it would make more sense to try to improve

upon the NN1-5 models to increase portfolio performance.

	OLS				OLS-3				OLS-H			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low	-0.718	0.626	5.240	0.120	0.447	0.673	6.460	0.104	-3.390	0.490	4.910	0.100
2	0.488	0.857	4.470	0.192	1.160	0.981	5.590	0.176	0.809	0.790	4.840	0.163
3	1.020	0.844	4.500	0.188	1.200	0.811	4.900	0.166	1.100	0.783	5.170	0.151
4	1.640	0.867	4.600	0.188	1.230	0.775	5.050	0.153	1.570	0.928	5.110	0.182
High	3.280	0.869	5.770	0.151	1.370	0.368	5.750	0.064	5.970	0.841	4.640	0.181
H - L	3.996	0.243	3.486	0.069	0.921	-0.304	4.613	-0.066	9.361	0.351	1.686	0.208
	PLS				PCR				E-Net			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low	-0.258	0.773	4.640	0.166	-0.742	0.713	5.390	0.132	-0.329	0.844	5.920	0.142
2	0.553	0.878	4.640	0.189	-0.176	0.739	5.130	0.144	0.051	0.958	5.240	0.183
3	0.889	0.861	5.150	0.167	-0.046	0.748	5.110	0.146	0.077	0.867	5.340	0.162
4	1.280	0.802	5.830	0.138	0.107	0.724	6.000	0.121	0.145	0.930	5.340	0.174
High	2.260	0.769	6.780	0.114	0.606	0.722	6.920	0.104	2.180	0.749	4.430	0.169
H - L	2.516	-0.003	3.839	0.000	1.348	0.008	5.096	0.001	2.508	-0.094	2.878	-0.032
	RF				NN1				NN2			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low	0.730	0.560	4.840	0.116	1.210	0.680	4.830	0.141	0.912	0.696	4.780	0.146
2	0.961	0.614	4.410	0.139	1.310	0.704	4.520	0.156	1.070	0.755	4.700	0.161
3	1.110	0.714	4.470	0.160	1.360	0.746	4.610	0.162	1.170	0.763	4.650	0.164
4	1.280	0.863	4.790	0.180	1.410	0.811	4.400	0.184	1.250	0.728	4.240	0.172
High	1.770	1.180	6.090	0.194	1.460	0.780	4.300	0.182	1.410	0.895	4.660	0.192
H - L	1.036	0.624	3.460	0.180	0.254	0.100	2.027	0.049	0.497	0.198	2.047	0.097
	NN3				NN4				NN5			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low	0.897	0.704	4.870	0.145	0.598	0.689	4.930	0.140	0.431	0.702	4.890	0.143
2	1.060	0.771	4.530	0.170	0.704	0.808	4.550	0.178	0.509	0.787	4.540	0.173
3	1.150	0.757	4.510	0.168	0.771	0.732	4.500	0.163	0.559	0.720	4.580	0.157
4	1.240	0.695	4.390	0.158	0.836	0.689	4.370	0.158	0.609	0.739	4.430	0.167
High	1.350	0.859	4.670	0.184	0.909	0.811	4.310	0.188	0.666	0.707	4.060	0.174
H - L	0.456	0.155	2.218	0.069	0.311	0.121	2.332	0.052	0.234	0.004	2.185	0.002

Table 2: This table reports the performance of machine learning portfolios. Predictions are sorted into 5 quantiles from high to low. For each quantile, the average forecasted return (Pred) is reported, the actual return (Avg), the standard deviation of actual returns (SD) and the quantiles Sharpe ratio (SR). All returns are reported in percentages. The stocks within each quantile are weighted by value.

	Avg (%)	Std (%)	Max 1M loss (%)	Max 1M Gain (%)	Annul. Sharpe	MaxDD (%)	Turnover (%)
S&P500	0.458	4.141	16.942	10.772	0.348	74.560	-
Sample mean	9.734	18.769	50.301	70.958	0.749	75.071	3.797
OLS	6.415	34.408	148.72	321.400	0.145	91.910	20.874
OLS-3	0.935	3.496	8.618	22.300	1.030	23.277	2.234
OLS-H	-7.027	94.221	1044.003	199.671	-0.024	628.021	119.772
PLS	-8.133	90.835	786.702	176.170	0.573	183.731	61.262
PCR	-2.543	66.678	686.266	481.333	0.225	339.543	24.20
ENet	1.326	10.282	40.266	32.902	-0.208	73.521	3.391
RF	4.783	72.41	479.349	773.995	0.115	171.552	18.005
NN1	0.643	3.319	12.235	8.971	0.575	68.938	1.320
NN2	0.652	4.187	17.425	14.144	0.447	87.471	1.384
NN3	0.533	3.464	13.763	10.703	0.407	77.121	1.230
NN4	0.473	3.211	9.977	9.595	0.454	62.707	1.123
NN5	0.412	3.022	11.046	14.051	0.382	65.119	1.117

Table 3: This table reports results for the mean-variance optimized portfolios. It reports the average monthly return, the standard deviation of monthly returns. Minimum and maximum monthly return. The annualized Sharpe ratio, maximum drawdown and the monthly turnover

6 Conclusion

This research took the dataset and machine learning methods of Gu, Kelly and Xiu (2020) to forecasts equity risk premia and use these forecasts to construct mean-variance optimized portfolios. The machine learning methods range from OLS to dimension reduction techniques, random trees and deep learning methods. I was able to replicate most machine learning methods in quite a similar fashion, even though my means were quite limited. The predictive performance of the machine learning methods was comparable to what Gu, Kelly and Xiu (2020) found, except for the fact that OLS-3 is able to outperform most machine learning methods in this research, whereas this was not the case for Gu, Kelly and Xiu.

Evaluating portfolio performance is done by constructing machine learning portfolios as done in Gu, Kelly and Xiu (2020) by sorting stocks into quantiles based on their forecast and then determining performance measures for each quantile. The aim is to have monotonically increasing realized returns and Sharpe ratios. The performance is further evaluated by constructing a zero-net-investment portfolio by buying the highest quantile and selling the lowest quantile. When evaluating portfolio performance I found that most machine learning methods have a hard time correctly estimating the magnitude of forecasts. This is generally a hard problem, but in this case may also be attributed to the fact that the dataset used does not include macroeconomic indicators that signal how the overall economy is doing, resulting in poorer performance. Unfortunately none of the methods are able to give performance that is comparable to Gu, Kelly and Xiu

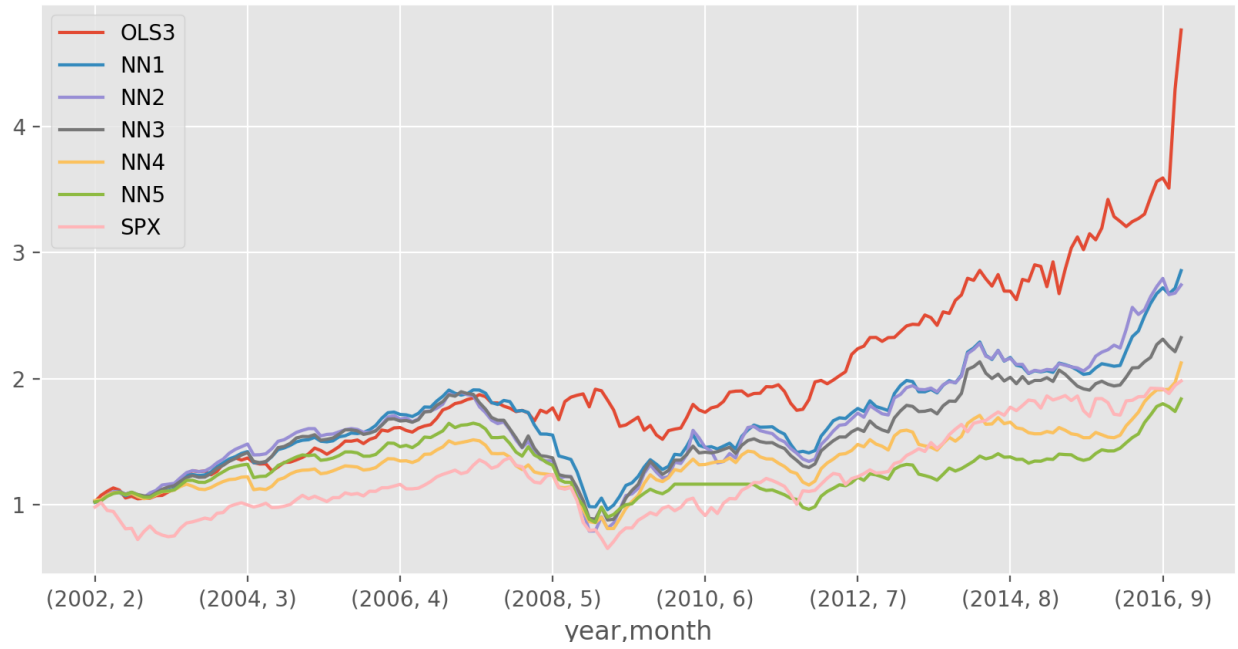

Figure 2: Plot of the cumulative returns of OLS3, NN1-5 and the S&P 500 index

(2020) when it comes to portfolio performance.

When comparing the performance of mean-variance portfolios it is found that using the sample mean compared to machine learning forecasts of equity returns is clearly superior. On an aggregate level forecast errors for the machine learning methods seems to be higher, leading to poorer mean-variance performance. The only method that was able to outperform the sample mean in terms of Sharpe ratio is OLS-3. On the other hand, when comparing machine learning methods to the S&P500, it is found that NN1-5 perform better in terms of Sharpe ratio. Some machine learning methods can also give erratic performance, leading to very high or low returns in certain months. This is due to the fact that small perturbations in the forecasts can lead to large changes in mean-variance weights. When forecast errors are present this can lead to poor performance. This suggest the use of regularization or shrinkage methods to combat forecast errors. Also the investigation of weight restrictions seems in place, to mitigate the tendency of mean-variance weight to choose large positions.

Using machine learning methods to forecast equity prices with a large dataset and subsequently using these forecasts to estimate mean-variance optimized portfolios has not been done before. It has been shown that machine learning methods in this setting are not yet able to outperform a naive sample mean for mean-variance optimization. The main reason this is the case is due to the fact that a smaller dataset is used compared to the paper this research is based on. The underperformance of the machine learning portfolios compared to Gu, Kelly and Xiu also partially confirms this. The use of a larger dataset would require more computing power, and implementing such an approach would be a good path for further investigation.

7 Discussion

One of the obvious differences between this research and that of Gu, Kelly and Xiu (2020) is the datasets that are used to train the models. We only adopted the 94 stocks level characteristics from 1986 onwards and did not include interactions with macro-economic variables as done in Gu, Kelly and Xiu (2020). The use of SIC industry dummies in Gu, Kelly and Xiu (2020) might also explain part of the underperformance of the machine learning models in this research compared to their results. So to improve forecasts it would be interesting to use the entire dataset. This was mainly a practical limitation, since I did not have the computing power at my disposal to efficiently train the models with this large dataset. Investigating how to go about this practial aspect is something that is not discussed a lot in the literature and would be interesting to further investigate, opening the door to large scale machine learning research for students.

As mentioned before, mean-variance portfolios can give large weights, and it would be interesting to investigate methods that either restrict the individual size of weights or whether it is allowed to have negative weights (short-selling). This would alter the mean-variance optimization problem and therefore likely require quadratic programming solutions. Luckily quite a bit of work has been done in this area and it would be a nice addition or extension to this research to further investigate how the choice how weights can be improved.

A further look into why the performance of the machine learning portfolios in this research differ so much from the results found in Gu, Kelly and Xiu (2020) might also be warranted to see how this research can be improved to get better equity return forecasts. It might be expected that predictive performance gets somewhat better on an aggregate level, but this was not the case. Further analyzing the forecasts in each quantile would possibly explain why performance was lacking.

References

Ao, M., Li, Y., & Zheng, X. (2019). Approaching mean-variance efficiency for large portfolios. *The Review of Financial Studies*, *32*(7), 2890–2919.

Ban, G.-Y., Karoui, N. E. K., & Lim, A. E. B. (2018). Machine learning and portfolio optimization. *Management Science*, *3*(64), 1136–1154.

DeMiguel, V., Garlappi, L., & Uppal, R. (2007). Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy. *The Review of Financial Studies*, *22*(5), 1915–1953.

Fastrich, B., Paterlini, S., & Winker, P. (2015). Constructing optimal sparse portfolios using regularization methods. *Computational Management Science*, *12*(3), 417–434.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics*(28), 337–374.

Friedman, J., Hastie, T., & Tibshirani, R. (2009). *The elements of statistical learning: Data mining, inference and prediction* (2nd ed.). Springer-Verlag, New York.

Green, J., Hand, J. R. M., & Zhang, X. F. (2017). The characteristics that provide independent information about average u.s. monthly stock returns. *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*, 2223–2273.

Hyvfil, L., & Rivest, R. L. (1975). Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, *5*(1), 15–17.

Jagannathan, R., & Ma, T. (2003). Risk reduction in large portfolios: Why imposing the wrong constraints helps. *Journal of Finance*(58), 1651–1684.

Lewellen, J. (2015). The cross-section of expected stock returns. *Critical Finance Review*, *4*, 1–44.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Markowitz, H. M. (1959). *Portfolio selection*.

Merton, R. C. (1980). On estimating the expected return on the market: An exploratory investigation. *National Bureau of Economic Research*(No. w0444).

Polson, N. G., Scott, J., & Willard, B. T. (2015). Proximal algorithms in statistics and machine learning. *Statistical Science*, *30*(4), 559–581.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 197–336.

Sharpe, W. (1962). A simplified model for portfolio analysis. *Management Science*, *9*(2).

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.

Whitten, A. (2015). *Why are there so few public companies in the u.s.?* Retrieved 17-6-2020, from `https://www.nber.org/digest/sep15/w21181.html`

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, *67*(2), 301–320.

A Model implementation details

	OLS	OLS-H	PLS	PCR	ENet	RF	NN
package	sklearn	sklearn	sklearn	sklearn	sklearn	sklearn	keras
version	0.23.1	0.23.1	0.23.1	0.23.1	0.23.1	0.23.1	2.3.1
modelname	LinearRegression	HuberRegressor	PLSRegression	PCA LinearRegression	ElasticNetCV	RandomForestRegressor	Sequential
Hyperparameters			K	K	l1_ratio = 0.5	MaxDepth = 6 #Trees = 300	Epochs = 100 Patience = 5 Learning rate = ADAM default LeakyReLU alpha = 0.1

Table 4: Details of the Python packages used and the hyperparameters involved in training the models

B Figures

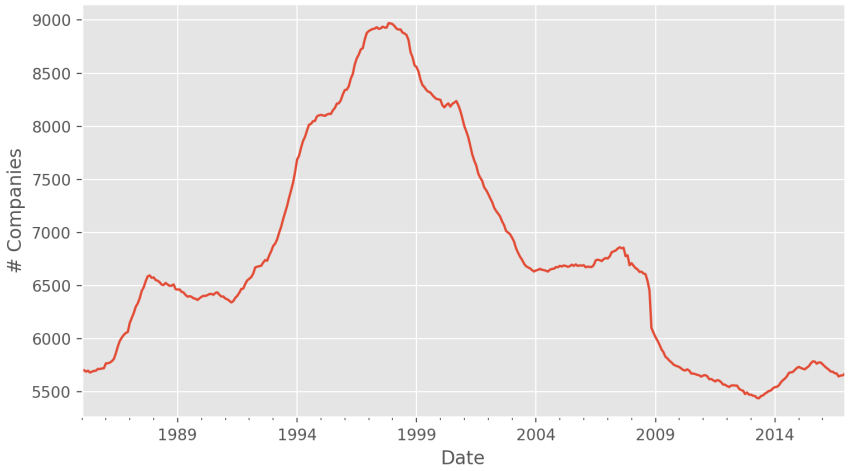

Figure 3: Number of companies in the sample per month

C Hyperparameter tuning

Algorithm 1: Pseudocode for tuning K in dimension reduction methods

```
K ← 0 ;
K_opt ← 0;
MSE ← ∞ ;
EarlyStopping ← 0 ;
while K < 50 do
    Estimate model with K components;
    Compute forecasts from validation sample;
    Compute MSE_K from forecasts ;
    if MSE_K < MSE then
        EarlyStopping ← 0 ;
        K_opt ← K ;
        MSE ← MSE_K ;
    else
        EarlyStopping ← EarlyStopping + 1;
        if EarlyStopping = 10 then
            Stop;
        end
    end
end
```