Seminar Empirical Methods and Data Analysis

A comparison of machine learning algorithms and traditional regression-based statistical modeling for forecasting stock price returns

Lukas Ferdinand Pitz

Student-Id: 7393631

August 30, 2023

Seminar by: Professor Jörg Breitung and Assistant Professor Sven Otto

Contents

1 Introduction

Forecasting stock returns is a prominent topic within classical finance. Researchers typical aim to estimate the Risk Premium which is the excess return of a security above the Risk-free Rate, frequently approximated by US-treasury bond rates. General goal is to improve the understanding of Risk Premium pricing and its underlying determinants. Authors commonly explore two sources of innovation. Firstly, they try to identify variables, such as financial indicators, macroeconomic variables, measurements of investors expectations, energy prices and more, which help to improve predictions on stock price returns. Secondly, they assess which model is best capable of predicting stock price returns. Numerous publications consider classical regression-based statistical models, like Fama and French (1993), Fama and French (2015) or Phan, Sharma, and Narayan (2015). A relatively new branch of literature utilizes machine learning models which have demonstrated efficacy in managing high-dimensional data in various contexts. Recent contributions have been made by authors like Abe et al. (2018) or Gu, Kelly, and Xiu (2020).

Given the continuous stream of innovations in forecasting stock price returns, it is crucial to evaluate their relative predictive power using the same data base to discern their individual utility and applicability. We investigate whether regression-based and machine learning models differ in their performance to predict stock price returns. For traditional regression-based statistical models, we consider Pooled Ordinary Least Squares (OLS) and Weigthed Least Squares (WLS) models. For machine learning models, we employ Penalized Regression models, like Lasso, Ridge and Elastic Net, as well as Tree-based models, like Random Forest (RF) and Gradient Boosted Regression Tree (GBRT) models. To enable a sound comparison of model performance, we employ one data set containing annual measurements of more than 200 financial indicators across 4960 publicly listed companies. This data set is split into a training and test data set. The training data set is used to train our models and perform hyperparameter training for the machine learning models, we also refer to this subset as in-sample data. Hyperparameter tuning involves further splitting of our in-sample data into a training and validation data set (in other literature sometimes also referred to as test data set). Model performance is measured utilizing the test data set which is not used for training or tuning. Accordingly, we refer to this data set as out-of-sample data. We report individual model performance, on the one hand based on each model's in- and out-of-sample Rooted Mean Squared Error (RMSE) and on the other hand with respect to their R-squared

from a Mincer-Zarnowitz Regression, introduced by Mincer and Zarnowitz (1969). The R-squared is received from a regression of the true values on the fitted values obtained from each model respectively.

We observed that among the models under consideration, the RF model performs best with obtaining the lowest in- and out-of-sample RMSE and the highest R^2 from all Mincer-Zarnowitz Regressions. However, there is only minor variation in the out-of-sample performance across all models with all out-of-sample RMSE falling within the range of 60.91 to 61.95 percentage points. The high deviations on average and the outcomes of statistical tests on the Mincer-Zarnowitz Regressions collectively suggest an overall fairly limited forecasting capability.

The paper is organized as follows: In Section 2, we give a comprehensive review of the utilized data and its cleaning process. In Section 3, we introduce our regression-based models, namely OLS and WLS. Section 4 includes a description of the applied machine learning models and their tuning. In Section 5 and 6 we present our results and discuss limitations. We conclude in Section 7.

2 Data & Cleaning

The data used in this paper is provided by the individual *Nicolas Carbone*. He uploaded the data set with the name "*200+ Financial Indicators of US stocks (2014-2018)*" on the website *kaggle* in 2019.[1] The original data set contains yearly measurements of 223 financial indicators for 3808 to 4960 publicly listed companies in the time frame 2014 to 2018. Additionally, it includes our objective variable the stock return in percentage points of each company for the years 2015 to 2019. The author obtained the data using the API provided by *Financial Modeling Prep*. While dealing with this data set, we find certain inconsistencies like extreme outliers, missing values and identical columns. The data quality is therefore ambiguous.

Our working data set includes annual data for 3726 companies. We determine outliers separately for each feature (column). If the feature value of one observation falls below the 0.1 or above the 99.9 quantile it classifies as an outlier. We impute each outlier with its respective feature (column) mean. The *Sector* variable is converted to dummy variables and identical columns are ignored. All financial indicators with more than 30% missing values are excluded due to their low information

[1] The data set is available here.

content. Furthermore, we utilize the Variance Inflation Factor (VIF) to determine columns with high collinearity, see Appendix A for more details. Our working data set contains 136 features or 61% of the original set of financial indicators.

We split our working data set into a training and test data set. The training data set contains the cleaned data for the years 2014 to 2017 and is used to train each of our applied models. Some machine learning models require additional splitting of the training data set to specify their respective model parameters, for example by implementing cross-validation, see Appendix B. The procedure to find the optimal model parameter is called hyperparameter tuning. Our test data set contains observations for the year 2018 and is only used to assess the out-of-sample performance of each applied model. It is therefore not used either in the training or in the hyperparameter tuning of our models.

3 Applied Regression-Based Models

In the following, we present our Pooled OLS and WLS models which include all 136 financial indicators as regressors. The OLS model serves as the fundamental and widely utilized regression-based model in econometrics. To account for evidence of heteroscedasticity within our OLS model, we apply a WLS model.

3.1 Pooled Ordinary Least Squares Regression

The applied Pooled OLS model and its estimator is based on the standard loss function, displayed in equation 1. Since no applied model explicitly incorporates the time dimension of the data, time indices in all following equations are dropped.

$$\hat{\beta}_{OLS} = argmin\, S(b) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \quad i = 1, .., N \tag{1}$$

A standard Breusch-Pagan Test[2] indicates the presence of heteroscedastic error terms within our model. Consequently, the OLS estimator loses its efficiency, while being still unbiased with respect to our model trained using in-sample data. If we would assess our model performance in-sample, we could stop here, since our performance measurement, the RMSE, only utilizes the fitted-values, not the model's

[2]Breusch-Pagan Test statistic: $LM = 722.99$; p-value $\leq 2.2e^{-16}$

variance. We would expect that under heteroscedasticity and while all other assumptions hold, the OLS model still produces the lowest RMSE in-sample. But since we assess model performance out-of-sample this may not be the case. This motivates an approach which is able to account for heteroscedasticity, in our case the WLS model.

3.2 Pooled Weighted Least Squares Regression

The applied Pooled WLS model and its estimator is based on the loss function, displayed in equation 2. The WLS is a special case of the Generalized Least Squares (GLS) estimator. It accounts for heteroscedastic error terms by not assuming identical values on the diagonal of its variance-covariance matrix. Contrary to the GLS estimator, it does not account for autocorrelation by assuming all off-diagonal elements of its variance-covariance matrix to be zero.

$$\hat{\beta}_{WLS} = argmin\ S(b)_w = \sum_{i=1}^{N} w_i(y_i - \hat{y_i})^2 \quad i = 1, .., N \tag{2}$$

Since we do not know the true weights w_i of our WLS model, we need to estimate them utilizing the residuals from our OLS regression. For more details on the methodology see Fox and Weisberg (2018).

4 Applied Machine Learning Models

We employ models from two distinct machine learning categories. The initial category comprises Penalized Regression models, including an Elastic Net model that incorporates a Ridge and Lasso model. In the subsequent category of Tree-based models, we implement Random Forest (RF) and Gradient Boosted Regression Tree (GBRT) models.

4.1 Elastic Net

Penalized Regression models typically introduce some kind of shrinkage penalty term to the objective loss function. The inclusion of a penalty term aims to regularize the regression coefficients by shrinking them towards or setting them to zero.

$$min. \sum_{i=1}^{N}(y_i - X_i'\beta)^2 + \lambda \sum_{j=1}^{p}\Big(\underbrace{\alpha|\beta_j|}_{Lasso} + \underbrace{(1-\alpha)\beta_j^2}_{Ridge}\Big) \quad i = 1, .., N \; and \; j = 1, .., p \tag{3}$$

Equation 3 displays the objective loss function of an Elastic Net model. It consists of the standard residual sum of squars (RSS) loss function of an OLS model and a combination of the shrinkage penalty terms of a Lasso and Ridge model. The Lasso (Ridge) model uses the sum of the $j = 1, ..., p$ absolute (squared) coefficients as penalty. The parameter $\alpha \in [0, 1]$ dictates which model dominates. The Elastic Net model collapses to the Lasso (Ridge) model for $\alpha = 1$ ($\alpha = 0$). The shrinkage parameter $\lambda \geq 0$ controls the impact of the penalty on the regression coefficient estimates. The optimal values for α and λ are unknown and are typically derived from the in-sample data via hyperparameter tuning. For more details see James et al. (2021).

To ascertain the ideal value for α, we iterate over a sequence of 100 distinct α values spanning the interval $[0, 1]$. For each of these 100 models, we apply a range of λ values and determine for each λ its respective Mean Squared Error (MSE) via a 10-fold cross-validation, see Appendix B. We use the algorithm embedded within the *glmnet* package by Friedman et al. (2023) to select the range of λ values. The model yielding the lowest MSE is selected. Figure 1 displays the MSE for each applied α. We observe a strong convergence of the MSE for $\alpha \geq 0.1$. The model with $\alpha = 0$ has the lowest MSE among all considered models, with the overall range of different MSE values being relatively narrow. Consequently, our hyperparameter tuning procedure indicates that the optimal Elastic Net model effectively collapses to the Ridge model. Figure 5 in Appendix D displays the hyperparamter tuning process for each λ value for the Ridge model. For completeness, we also consider a Lasso model which results we discuss in Section 5.

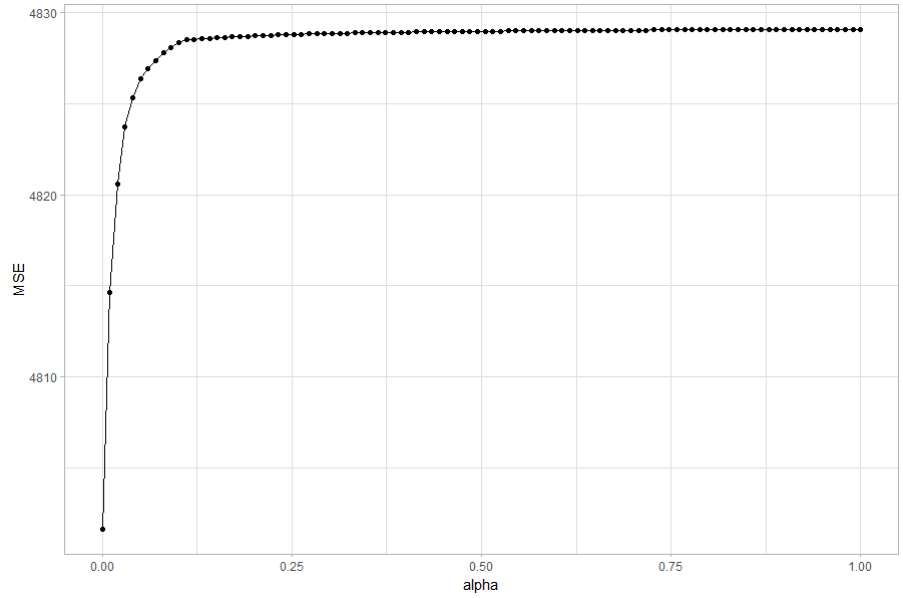

Figure 1: Hyperparameter tuning for Elastic Net model; $\alpha_k \in [0, 1]$ with $k = 1, ..., 100$

4.2 Random Forest

Tree-based methods involve partitioning the feature space into simpler regions. A basic tree model then employs the mean of training observations within these regions to predict new observations within the same region. Split variables and points, used for feature space division, are chosen sequentially via a Greedy Algorithm. This algorithm selects optimal split variables and points that maximize the reduction in the Residual Sum of Squares (RSS) at the current node, essentially identifying the variables and points leading to the highest local decrease in the applied loss function.[3] The primary issue with basic single-tree models is their proneness to overfitting in-sample data, resulting in high variance and less dependable out-of-sample predictions. Addressing this concern, Breiman (1996) introduces Bagging which involves training multiple tree models on sub-samples of the training data using bootstrapping. Subsequently, these models are used to make predictions on the out-of-sample data, with their mean forming the Bagging model's prediction. This ensemble method mitigates out-of-sample variance compared to individual tree models. A known problem is the similarity among trees due to the resemblance of bootstrap sub-samples in large data sets. Consequently, predictions from individual trees tend to be highly correlated, limiting the variance reduction when averaging predictions across models. To counteract this, Breiman (2001) proposes extending the Bagging approach by considering only subsets of all available features p at each

[3]See Breiman et al. (2017) or James et al. (2021) for more details on the exact methodology.

split while growing eahc tree, resulting in the Random Forest (RF) model.

For hyperparamter tuning of RF models two groups of parameters may be considered. On the one hand, hyperparameters of the tree models incorporated in the RF model, acting as the base learner. These encompass parameters such as the minimum number of observations in a potential node to warrant a split, denoted as $minsplit = q$. This parameter typically also determines the minimum observations within a terminal leaf, approximating $minbucket \approx \frac{q}{3}$. Additionally, the maximum depth of the trees is regulated by $maxdepth = h$. On the other hand, the second group of parameters relates to the ensemble procedure. These include the number of trees B to be integrated within the model and the number of randomly drawn features m to be considered as split variables at each node.
We start by identifying the optimal number of trees B to combine in our RF model by assessing the behavior of the out-of-bag (OOB)MSE when we average over an increasing number of trees. While doing so, we adhere to default values for the remaining hyperparameters prescribed by the authors Wright, Wager and Probst (2023) of the software package *ranger* and also adopted among others by James et al. (2021). The loss term of RF converges to a constant for an increasing number of trees indicating no overfitting, which legitimize the isolated tuning. Figure 2 indicates the convergence of the OOB MSE to a constant at around 500 trees, with the lowest OOB MSE received for the model using $B = 754$ trees.[4] Due to this convergence and limited computational resources, we specify our RF models to include $B = 500$ trees in all following models.

[4]The OOB MSE for $B = 500$ and $B = 754$ are 4827.905 and 4812.685 respectively, which is a difference of approximate 0.3%.

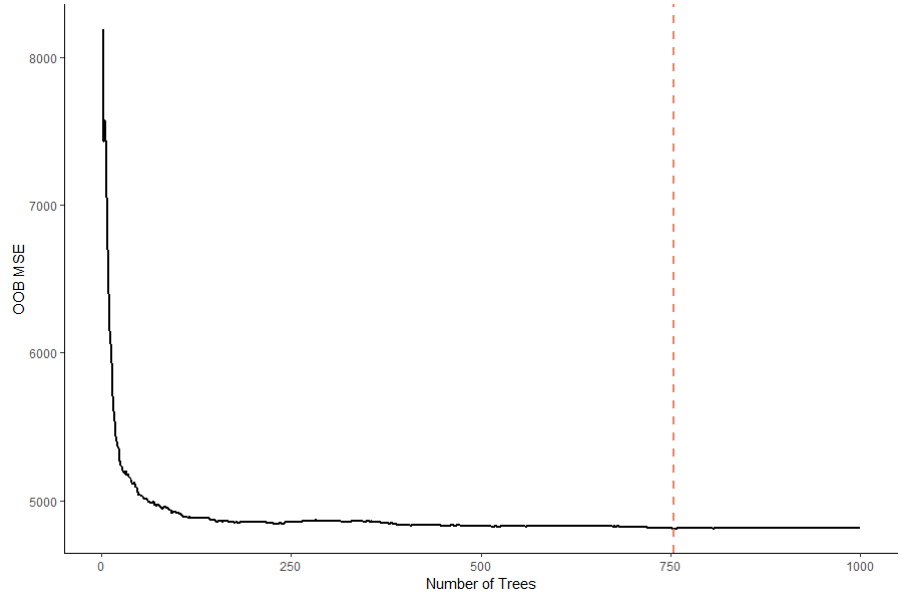

Figure 2: Hyperparameter tuning for Random Forest model
Number of trees $B = 1, ..., 1000$; lowest OOB MSE at $B = 754$, indicated by a vertical line.

We continue by determine the optimal values for the number of randomly drawn features m to be considered as splitting variables in each node and the tuning parameters of our base learner, the minimum number of observations in a potential node $minsplit = q$, the minimum number of observations in a terminal leaf $minbucket \approx \frac{q}{3}$ and the maximum depth of the trees $maxdepth = h$. A heuristic guideline suggests the floor of the square root of the number of all possible split variables ($m \approx \sqrt{p}$) for reference, see James et al. (2021).[5] With this in mind, we consider different values of m within the range of $m \in [2, 20]$. For the remaining hyperparameters we consider the following range of values: $minsplit = q \in [1, 15]$, $minbucket \in [1, 5]$ and $maxdepth \in [1, 10]$. We use a so called hyperparameter grid to evaluate different combinations of these parameters and calculate a total of 760 different RF model specifications. Table 1 presents the hyperparameter configurations for the 5 out of 760 calculated models with the lowest OOB MSE.

OOB MSE	minsplit	minbucket	maxdepth	m
4664.454	15	5	3	13
4666.002	15	5	3	10
4667.653	15	5	7	7
4667.872	15	5	7	14
4668.154	15	5	1	15

Table 1: Results for the hyperparameter tuning of the RF model

[5] With $p = 137$ features in our data set, the reference point is $m = \sqrt{137} \approx 11$.

4.3 Gradient Boosted Regression Trees

Boosting is yet another method which can be used to improve tree models by leveraging the residuals from the prior tree, sequentially refining the model. The modeling procedure commonly starts with a simple base learner, such as a shallow tree. The trained model is then used to fit the data, producing residuals in the process. These residuals are then used to train the succeeding model, improving the fit where most needed. This iterative process continues, yielding a series of interdependent trees, until a specified stopping criterion is met. Mathematically, this procedure aims to improve the loss function $L(y_i, \hat{y}_i)$, in the direction of its gradient. The loss function is at its optimum if the gradient is 0 at all N observations, i.e. $\frac{\partial L(y_i,\hat{y}_i)}{\partial \hat{y}_i} = 0$, meaning that we have a perfect fit of the model on our in-sample data. However, this potential overfitting hinders the usefulness of the model to make predictions out-of-sample. To prevent rapid overfitting, a learning rate $\lambda \in [0, 1]$ is used when updating the residuals promoting slow learning.[6]
Like Bagging and RF, the Gradient Boosted Regression Tree (GBRT) model is a ensemble method, combining multiple tree models in its final form. Yet the GBRT model does not grow multiple independent trees from bootstrap sampling. Instead, it uses the residuals of the latest fitted model to train the model which is then added to the overall fitting function to update the residuals, consequently growing lots of interdependent trees. Besides this key difference, it is possible to incorporate the idea of RF by limiting the number of features the algorithm may choose for building each tree to a randomly drawn subset.

GBRT models are highly flexible, accommodating numerous hyperparameters that can be fine-tuned according to the characteristics of the data and requirements of the task. The range of available parameters leads to a computationally intensive grid-search. Notably, unlike Bagging tree models and RF, the validation error of the GBRT model does not converge to a constant as the number of trees/iterations of the algorithm increases. This behavior arises from the iterative nature of training on the residuals of the preceding model. Initially, the GBRT model's validation error tends to decrease but eventually starts rising when the model begins to overfit, see for example Figure 6 in Appendix C. How fast the model tends to overfit depends among other things on the learning rate $\lambda \in [0, 1]$. A relatively high (low) λ leads to rapid (gradual) in-sample fit improvements and a potential fast (slow) reversal in the validation error. Consequently, rather low learning rates are beneficial to prevent rapid overfitting. However, this comes with the trade-off of requiring more

[6] For a detailed description of the algorithm see James et al. (2021), p.347.

iterations (trees) for training, thereby demanding more computational resources. We implement values of $\lambda \in [0.005, 0.1]$ in our grid-search.

To further limit the risk of overfitting, we use a random sub-sample of our observations to train each tree, employing fractions of the training data of 0.65, 0.8 and 1 within our grid-search. Additionally, we limit the use of features for growing new trees to fractions spanning the range of 0.7 to 1. The choice of these tuning parameters involve a treade-off between preventing in-sample overfitting and avoiding potential underfitting, which can occur by overly restricting the information accessible to the model.

We also fine-tune two parameters associated with the base-learner: the maximum depth of each tree and the minimum sum of weights for all observations required in a child node which is similar to the hyperparameter *minsplit* considered in the context of tuning RF models. Both these parameters contribute to curtailing in-sample overfitting, as a high depth and distinct splits enable the model to capture highly specific relations within particular subsets of the data. We apply values for the maximum depth of the tree in the range of $[5, 11]$ and values for the minimum sum of weights for all observations required in a child node in the range of $[5, 11]$.

With limited computational resources, an early stopping rule may be implemented. We stop the training of our models if for 30 consecutive iterations (i.e. 30 consecutive trained trees) the validation error does not decrease which indicates the beginning of overfitting. In total, we train and assess 768 models. Table 2 displays the five hyperparameter combinations yielding the lowest validation RMSE.

RMSE	trees	λ	max depth	min. child weight	sub-sample	feature sample
68.233	366	0.005	11	11	0.65	0.7
68.245	177	0.01	11	11	0.65	0.8
68.253	179	0.01	11	11	0.65	0.7
68.261	366	0.005	11	11	0.8	0.7
68.265	305	0.005	11	11	0.8	0.8

Table 2: Results for the hyperparameter tuning of the GBRT model

5 Results

Given our primary interest in the forecasting ability of each model, we compare performances based on their out-of-sample RMSE. For completeness we report the in-sample RMSE, both displayed in Figure 3. For a plain benchmark, we consider

the forecast performance of the historic mean, which is the mean of the stock return across all observations within our training data set. Furthermore, we evaluate the respective R-squared received from a Mincer-Zarnowitz Regression and perform a Wald-Test and individual t-tests on its coefficients, with results presented in Figure 4 below and Table 3 and 4 in Appendix C.

In Figure 3 the models are arranged in an ascending order based on their out-of-sample RMSE. The RF model performs best both out- and in-sample, while only the Ridge model also performs better than the historical mean out-of-sample. Generally, there are only marginal variation in the out-of-sample performance across all models. The RF model achieves the lowest RMSE at 60.91, whereas the OLS model records the highest RMSE at 61.95. With an average deviation of 60.91 to 61.95 percentage points from the true stock return in our out-of-sample data, the economic forecast power of these models can be regarded as poorly. We will discuss, whether these outcomes are due to inadequate models, the data at hand or a general lack of stock return predictability in section 6.

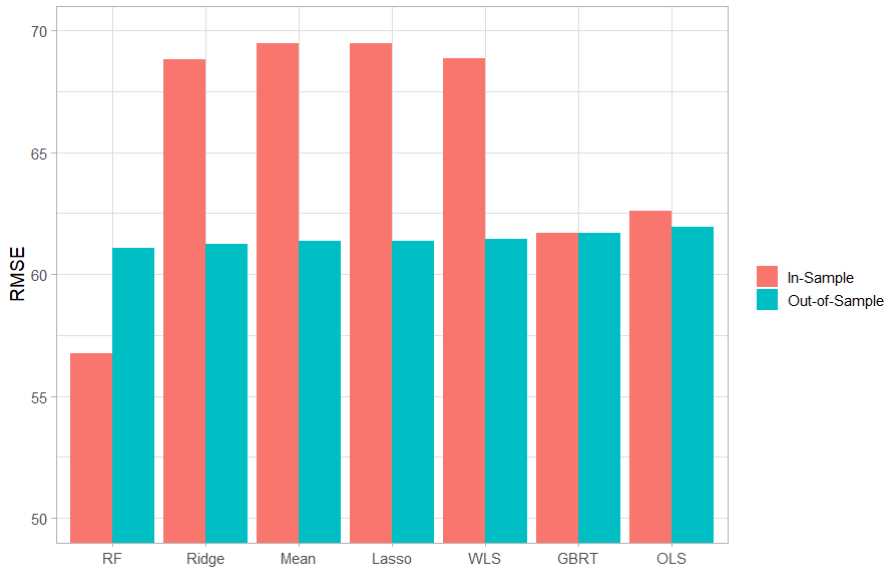

Figure 3: Comparison of in- and out-of-sample RMSE; Ordered from lowest to highest in-sample RMSE

Besides these main results, two specifics are worth mentioning. Firstly, the Lasso model, after fine-tuning its hyperparameters, shrinks all coefficients, except the constant, to zero. Consequently, the Lasso model collapses to the (historical) mean for

prediction.[7] The Ridge model performs slightly better than the historical mean and therefore the Lasso model. This is coherent since our optimal Ridge model applies such low weights to the majority of the features, leading to their quasi omission. Secondly, for all models, except the GBRT, the in-sample RMSE is higher than the out-of-sample RMSE. This phenomenon contrasts with typical forecasting scenarios. Machine Learning models in particular, tend to fit closely to in-sample data, often displaying overfitting tendencies that lead to high variance in their out-of-sample predictions. A potential explanation could be well behaved, low variance, data in the out-of-sample compared to the in-sample data set.

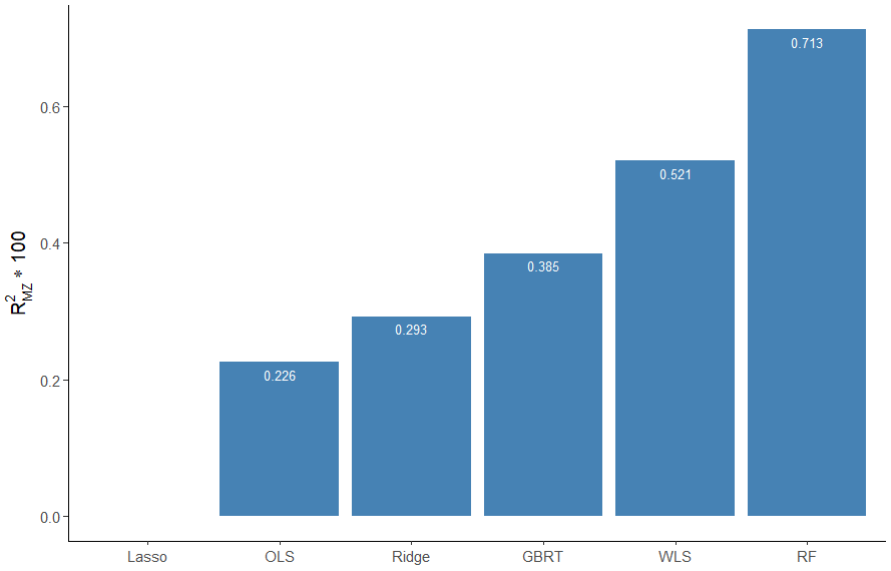

Figure 4: Mincer-Zarnowitz R-squared; y-axis scaled to percentage points; Ordered from lowest to highest R^2_{MZ}

Figure 4 presents the R-squared from a Mincer-Zarnowitz Regression scaled to percentage points. This regression involves regressing the true out-of-sample stock returns on our fitted values generated by each applied model. Again the RF model performs best compared to the other models with a R^2_{MZ} of 0.00713 or 0.713 percentage points. Consequently, the RF model helps to explain approximately 0.713 percent of the variance of the out-of-sample data. The R^2_{MZ} of the Lasso model is equal to zero, since it collapses to the constant (historic) mean of the training data. We perform a joint Wald-test and individual t-tests on the constant and coefficient of each Mincer-Zarnowitz Regression to test whether the fitted values help to ex-

[7] To account for the possible impact of specific splits during the cross-validation process used in tuning the shrinkage parameter λ, we re-run the tuning and training of our Lasso model 100 times. In all 100 iterations, all coefficients except the constant were consistently shrunk to zero.

plain the true values with results presented in Table 3 and 4 in Appendix C. The Wald-tests and individual t-tests reject the null hypothesis for each model considering a 0.01 significance level. These results indicate that the models are unable to perfectly explain the true values.

6 Limitations

We encounter two major limitations, namely the utilized data and the general predictability of stock returns.
The data set contains annual data for five consecutive years, encompassing between 3808 to 4960 companies across several industries. While we have an adequate number of cross-sectional observations, the data comprise only a low frequency of data points across a few time periods. This limitation potentially affects the models' ability to capture the relationships between the considered stock attributes and their corresponding returns, thereby affecting the reliability of predictions. Additionally, we encounter several inconsistencies within the data, leading to an ambiguous assessment of the its quality. Providers like *Bloomberg*, *yahoo Finance* or *Refinitiv Eikon* offer access to high quality and frequency data for a wide time horizon. Nonetheless, the access to this kind of data is often subject to licensing restrictions.

On the one hand, multiple studies, such as Hjalmarsson (2010), Narayan and Gupta (2015) or Avramov and Chordia (2006), have put forth models that demonstrate both in-sample and out-of-sample predictability of stock returns. On the other hand, some meta analysis, like Goyal, Welch, and Zafirov (2021), show that lots of these models and their as influential identified characteristics lose their predictive power considering new data, with these models regularly failing to beat the historical mean as predictor. The lacking ability to beat the historic mean as a predictor, see Figure 3, is therefore not unique to our particular models and application. Mclean and Pontiff (2016) argue that the observed predictability of stock returns in some publications might be due to asset miss-pricing. The authors argue that the finance industry incorporates information from publications for arbitrage, leading to price adjustment in return. Furthermore, Gu, Kelly, and Xiu (2020) undertake a comprehensive analysis that includes models like OLS, Elastic Net, RF, and GBRT, applied to a rich data set spanning 60 years of monthly data including around 6200 individual companies listed in the NYSE, AMEX and the NASDAQ.[8]

[8]NYSE - New York Stock Exchange; AMEX - American Stock Exchange; NASDAQ - National Association of Securities Dealers Automated Quotations Stock Market.

Although their models exhibit a greater capacity to account for both in-sample and out-of-sample variances, their economic applicability, considering for example stock picking, remains eminently constraint as well.

7 Conclusion

In our paper, we apply several models ranging from traditional regression-based methods like OLS to machine learning models like RF and GBRT. Considering their out-of-sample performance, they show only marginal differences in their capability of predicting stock returns. The in-sample performance of our RF model shows some improvements compared to the other models, but with the overall accuracy of the prediction being still poor. Consequently, we can not detect a meaningful difference between traditional regression-based and machine learning model's capability to predict stock returns within the scope of our available data and application. This persistent incapability throughout all our models indicates that not the applied models are the origin of the limitations, but either the utilized data or more general challenges in predicting stock returns.

To address this issue, our forthcoming research involves the application of the considered models to a data set comprising high-frequency data spanning an extended time horizon. This step is aimed at gaining further insights into predictive capability differences of traditional regression-based and machine learning models under more comprehensive and dynamic conditions.

References

Abe, M et al. (2018). "Deep learning for forecasting stock returns in the cross-section". In: *SpringerM Abe, H NakayamaAdvances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, 2018 • Springer*.

Avramov, Doron and Tarun Chordia (2006). "Predicting stock returns". In: *Journal of Financial Economics* 82 (2).

Breiman, Leo (1996). "Bagging predictors". In: *Machine Learning* 24 (2).

Breiman, Leo (2001). "Random forests". In: *Machine Learning* 45 (1).

Breiman, Leo et al. (Jan. 2017). "Classification and regression trees". In: *Classification and Regression Trees*, pp. 1–358.

Fama, Eugene F. and Kenneth R. French (1993). "Common risk factors in the returns on stocks and bonds". In: *Journal of Financial Economics* 33 (1).

Fama, Eugene F. and Kenneth R. French (2015). "A five-factor asset pricing model". In: *Journal of Financial Economics* 116 (1).

Fox, John and Sanford Weisberg (2018). *Time-Series Regression and Generalized Least Squares in R*, pp. 1–10.

Goyal, Amit, Ivo Welch, and Athanasse Zafirov (2021). "A Comprehensive Look at the Empirical Performance ofEquity Premium Prediction II". In: *SSRN Electronic Journal*.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu (May 2020). "Empirical Asset Pricing via Machine Learning". In: *The Review of Financial Studies* 33 (5), pp. 2223–2273.

Hjalmarsson, Erik (2010). "Predicting global stock returns". In: *Journal of Financial and Quantitative Analysis* 45 (1).

James, Gareth et al. (2021). *An Introduction to Statistical Learning*. Second Edition, pp. 237–252.

Mclean, R. David and Jeffrey Pontiff (2016). "Does Academic Research Destroy Stock Return Predictability?" In: *Journal of Finance* 71 (1).

Mincer, Jacob and Victor Zarnowitz (1969). "The Evaluation of Economic Forecasts". In: *Economic Forecasts and Expectations: Analysis of Forecasting Behavior and Performance*.

Narayan, Paresh Kumar and Rangan Gupta (2015). "Has oil price predicted stock returns for over a century?" In: *Energy Economics* 48.

Phan, Dinh Hoang Bach, Susan Sunila Sharma, and Paresh Kumar Narayan (2015). *Stock return forecasting: Some new evidence*.

Appendices

A Variance Inflation Factor

The Variance Inflation Factor (VIF) measures the severity of multicollinearity between predictors (features) in regression models. It estimates by how much the variance of a regression coefficient is inflated relative to the case with no multicollinearity present. The VIF is calculated by regressing each $k = 1, ..., K$ feature on the rest of the features used in the model. The R_k^2 of this regression is than used to calculate the VIF_k for the feature k, see equation 4.

$$VIF_k = \frac{1}{1 - R_k^2} \tag{4}$$

$VIF \in (1, \infty)$. $VIF = 1$ indicates that there is no correlation between feature k and the remaining considered features. As a rule of thumb $VIF \geq 10$ indicates high correlation and is cause for concern.

We utilize the VIF to detect and exclude features within our high-dimensional data set which contribute little additional information due to their severe collinearity to other considered features. This may be for example a feature which is simply a ratio of two other features also included in the data set. We calculate the VIF for each feature and then exclude the feature with the highest $VIF \geq 10$. This procedure is repeated until no feature with a $VIF \geq 10$ remains. We exclude only one feature at a time because collinearity concerns for some features may be resolved once the as highly collinear identified features is excluded. Deleting all features with a $VIF \geq 10$ may lead to the loss of relevant ones. Applying this procedure we determine 29 features to be ignored due to their high levels of collinearity to other features.

B Cross-Validation

For tuning machine learning models, the in-sample data set is commonly split into a train and test/validation set. The most basic technique uses a single data split. Problem of this approach is that the trained model depends on the exact split which impacts the generalization of the model out-of-sample. An alternative approach accounting for this problem is cross-validation where the training data is partitioned into several distinct subsets, so called folds, and each fold is used once for test-

ing/validation.

In the initial step of K-fold cross-validation, the dataset is divided into approximately equal-sized segments, creating K folds. Subsequently, the model is trained using data from $K-1$ folds while utilizing the k^{th} fold for evaluating the predictive errors, for example by using the MSE. This process iterates across all $k = 1, ..., K$ folds. Finally, the K individual estimates are combined through averaging. For instance, considering the MSE:

$$\frac{1}{K}\sum_{k=1}^{K} MSE_k$$

For more details see James et al. (2021).

C Additional Figures and Tables

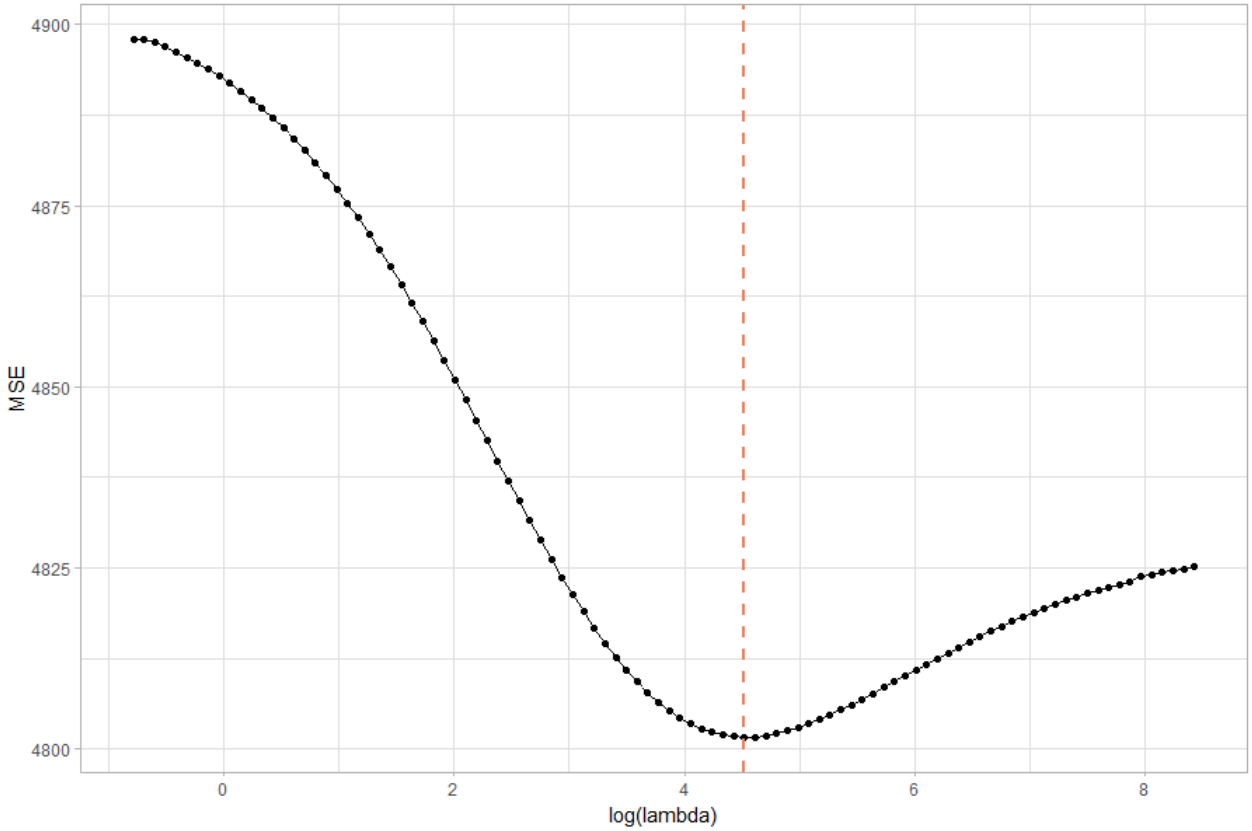

Figure 5: Hyperparameter tuning for λ of Ridge model; lowest MSE found for $\lambda = 91.70609$

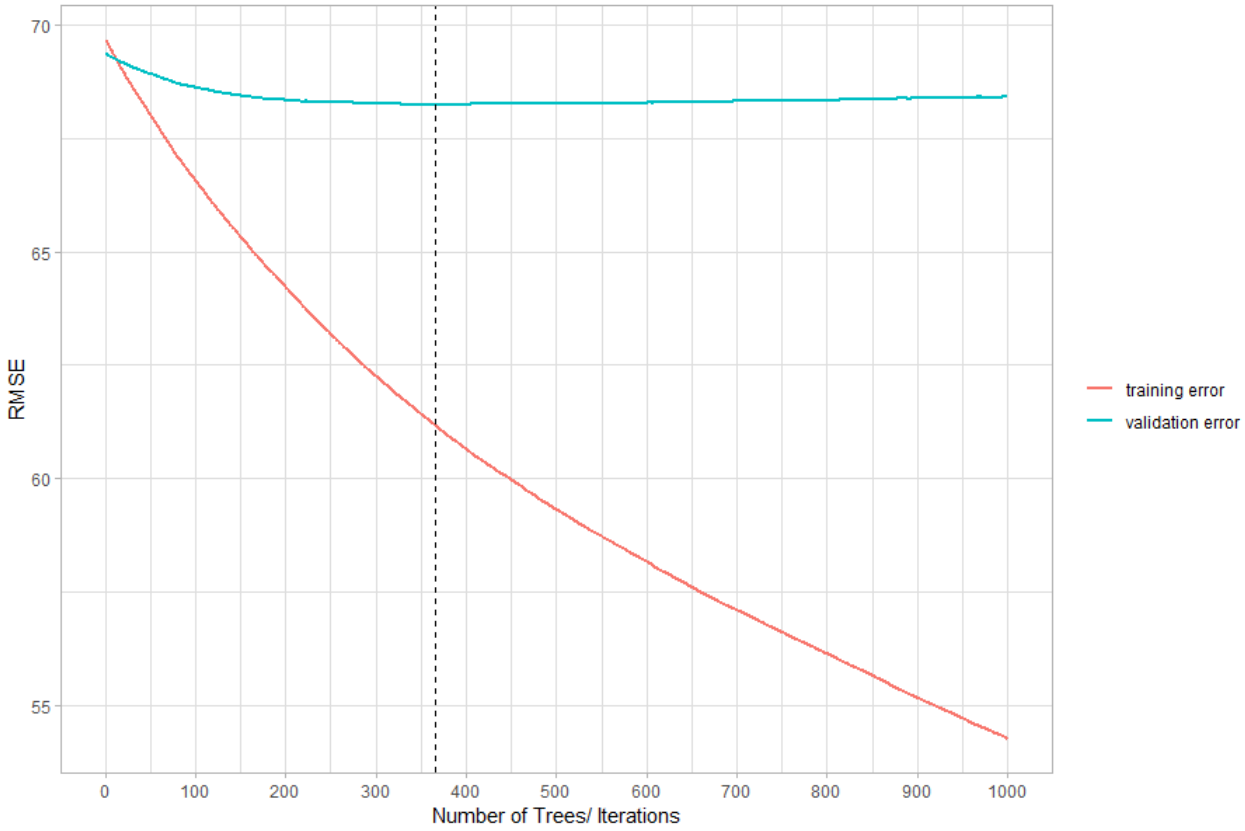

Figure 6: Hyperparameter tuning for GBRT model; using the optimal hyperparameters, lowest validation RMSE found for 366 trees.

Wald-Test of coefficients of the Mincer-Zarnowitz-Regression:

$$H_0 : \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad vs. \quad H_1 : \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Model	χ^2	p-value
OLS	300.0	0.00
WLS	243.8	0.00
Ridge	211.7	0.00
Lasso	NA	NA
Random Forest	204.6	0.00
GBRT	274.7	0.00

Table 3: Results of a Wald-Test performed on the Mincer-Zarnowitz-Regressions

a) t-test for **constant** of Mincer-Zarnowitz-Regression:

$$H_0 : \beta_0 = 0 \quad vs. \quad H_1 : \beta_0 \neq 0$$

b) t-test for **coefficient** of Mincer-Zarnowitz-Regression:

$$H_0 : \beta_1 = 1 \quad vs. \quad H_1 : \beta_1 \neq 1$$

Model	t-statistic constant	p-value	t-statistic coefficient	p-value
OLS	15.53	0.00	−119.84	0.00
WLS	12.35	0.00	−38.63	0.00
Ridge	5.15	0.00	−5.28	0.00
Lasso	NA	NA	NA	NA
Random Forest	3.29	0.00	7.51	0.00
GBRT	14.72	0.00	−117.72	0.00

Table 4: Results of a t-tests performed on the Mincer-Zarnowitz-Regression's constant and coefficient

D R-Packages used for modelling

In the following, we account for the main R-Packages used to build our models which are not included in *Base R*:

Package	Authors	Version	Model
glmnet	Friedman, Hastie et. al.	$4.1 - 8$	Lasso, Ridge and Elastic Net
randomForest	Bremann, Cutler, Liaw and Wiener	$4.7 - 1.1$	RF
ranger	Wright, Wager and Probst	0.15.1	RF
xgboost	Chen, He et. al.	1.7.51	GBRT

Further software packages are referred to in the attached code.