The usefulness of financial ratios using machine learning

A study of forecasting asset returns and bankruptcy using financial ratios

By: Milan Muller, 501225

Supervisor: Zwan, T. van der

Second assessor: Wel, M. van der

A thesis presented for the degree of

Bachelor Econometrics

Erasmus School of Economics

ERASMUS UNIVERSITY ROTTERDAM

The Netherlands

June 30 2021

Contents

Abstract

I perform a comparative analysis of machine learning methods using financial ratios as a base predictor set to analyse the predictability for two different topics. The first topic is asset pricing and the second is bankruptcy. This is done by analysing stocks between the period of 1970 and 2020 in the US. I conclude that financial ratios have no great predictive power for forecasting asset returns for both a quarterly and annual horizon. None of the used machine learning models (Trees, neural networks and elastic net) were able to outperform the OLS-3 model using only the features momentum, size and book-to-market. However, I find promising results for predicting bankruptcy. I achieve with a gradient boosted classifier a model with 14.75% false positive rate (Type I error) and 18.68% false negative rate (Type II error). The F1 score which combines these two rates in one metric, for this model is 15.79% compared to the F1 score of 12.22% that is achieved with a simple logit model. This is even further improved by the meta labeling extension. Meta labelling is a two-step modelling procedure where the second model learns from the first model. This procedure results in a model with a F1 score of 19.14% which is substantially higher than the stand alone models. The models also agree on most of the same predictors but the neural network failed to perform well, possibly due to the low number of observations.

1 Introduction

Using financial return in conducting firm specific analysis has always been a popular fundamental approach to analyse firms (see Lewellen (2004) and Tian and Yu (2017)). Analysing firms is a broad research area, for example one can be interested in predicting the excess returns of company or predicting if a firm is likely to default to avoid risky assets. Wang et al. (2014) suggested that there is no exact theory for corporate failure and it needs the exploratory efforts to identify characteristics to predict bankruptcy by trial and error. It would therefor be interesting to see if there are any useful ratios that have predictive power.

One other heavily studied in the literature is predicting asset returns like in Feng et al. (2018). There are many approaches and models introduced in recent literature, with the goal to get a better understanding of the equity's risk premiums.[1] This is useful for investors to make better investment decisions. With non-linear machine learning models becoming more popular, it is interesting to compare those new methods to classical linear models. This is done in the research of Gu et al. (2020). Where they compare different machine learning models on a large fundamental data set to forecast risk premiums.

In this research I focus on exploring the predictive power of financial ratios in the asset pricing and bankruptcy prediction area. To explore the predictive power I use several machine learning models. The models are: elastic net, boosted trees, random forest and a forward feed neural network. The elastic net is a linear model that can be used for high-dimensional data sets. This linear model is useful to compare the performance of non-linear models with. I compare the models by out-of-sample forecasts. For the asset pricing models I use the Diebold-Mariano test to test for statistical difference between the models. For the bankruptcy topic I use several classification metrics including the F1 score to compare the performance across models. To prevent over-fitting which is a main problem in machine learning, I will split the data set into a training and test set where the test set is not used for fitting and tuning. There will be looked at quarterly and annual cross-sectional stock data. The data used contains around 30.000 stocks listed on NYSE, NASDAQ and AMEX during the period of 1970-2020.

In the data section I will tell how I constructed the data set and where the data is from. After that in the methodology I will discuss the models structural form for both the regression case and classification case and also its strengths and weaknesses. Finally in the result section I discuss the results for the asset pricing topic where I conclude that the models using the large financial ratio data set are not capable of outperforming the benchmark model. For the bankruptcy topic I conclude that the models are able to extract useful information out of the ratios to predict bankruptcy with a low false negative rate. The meta labelling extension which is explained in the the methodology achieves a improved false positive while maintaining the same false negative rate, which there for increases the F1 score.

2 Literature

The empirical literature on asset return prediction can be split into 2 ways: the first one is to model stock characteristics as a cross-sectional regressions across stocks at one point in time using a few lagged characteristics to predict future returns. This is demonstrated by Fama and French (2008) and Lewellen (2014). The second way is to use historical returns as a time-series to predict future returns using time-series regressions like Nelson et al. (2017). This can also be extended by using a few macro economic variables, as in Feng G. and D. (2020), Welch and A. (2008) and Koijen and V. (2011).

A big limitation that these two ways share is that they are poor in handling a huge amount of predictor variables that the literature assembled during the last years. The challenge is to extract the new forecast power of a variable while handling the false discovery problem. Machine learning models can help to overcome these limitations. Machine learning has already appeared in recent asset pricing literature. Like Rapach D. E. and G. (2013) used a LASSO method to predict global equity market returns where they found that LASSO in capable of selection useful predictors in a large data set. Also several papers about using neural networks to predict derivatives prices like in Yao and L. (2000). Recently there also papers about studying the cross-section of stock returns with machine learning models. Kozak S. and S. (2020) and Freyberger J. and M. (2020) use shrinkage methods to approximate a stochastic discount factor and a nonlinear function for expected returns. They found that allowing for non-linearity can improve the predictability significantly.

This motivated me to compare non-linear models with linear models. This LASSO idea can also be used for neural networks to handle a large predictor set which I will use. To research if company ratios have any predictive power to forecast asset returns I will use the machine learning models used in Gu et al. (2020). I replicate most of the methods here only with a company ratio data set.
The literature of predicting bankruptcy already exists like in (du Jardin (2016), Danenas and Garsva (2015) and Tsai et al. (2014)). However this research contributes by this by exposing the models to a broad range of financial ratios. Some of these ratios are also found in Barboza et al. (2017) but I increase the amount of predictors by 5 times to explore the information of a broader set and to see if the machine learning models are able to capture the most useful ones. The extension I will conduct called meta labeling, which is introduced in de Prado (2018), motivates me to test this idea on the bankruptcy data. This is a new kind of ensemble method which is not used a lot in the literature yet.

3 Data

3.1 Predicting assest returns

I will look at US stocks that were listed on NYSE in the period between 1970-01 to 2020-12. For each stock there will be 74 financial ratios including book-to-market (bm), price-earnings (p/e) dividend-payout-ratio (dpr) and 4 momentum variables. The full ratio description can be found at the COMPUSTAT website. Stock returns are known to be noisy, So a quarterly frequency is used. This is also in line with the update frequency of most of the ratios. The total stock count is over 4500 stocks with a average of over 1500 stocks each quarter. All this data is collected through the database of CRSP and COMPUSTAT. The missing financial ratios are imputed via the cross-sectional median of the stocks in that period. However, the data from COMPUSTAT is in a monthly frequency. I transformed this to a quarterly frequency by taking in account financial ratios publicity dates. Since most of the companies announce their results in the same quarter, I took the quarters FEB-APRIL, MAY-JUL, AUG-OKT and NOV-JAN. This way the new public results can be used in the upcoming month. For the annual horizon I take aggregate this to yearly data starting on may.

There will also be 8 macro economic variables which are: dividend-price ratio (dp), earnings-price ratio (ep), book-to-market ratio (bm), net equity expansion (ntis), Treasury-bill rate (tbl), term spread (tms), default spread (dfy), and stock variance (svar). These follow the description of Welch and A. (2008). To define the excess returns (r_e) of a stock I use the 1-month US treasury as the risk-free rate (r_f) and use the total holding return as asset return (r_a). Which will be calculated as follow:

$$r_{e,(i,t)} = r_{a,(i,t)} - r_{f,t} \tag{1}$$

The predictor set $\mathbf{x_{i,t}}$ for each stock i in month t is defined as:

$$\mathbf{x_{i,t}} = \mathbf{z_{i,t}} \otimes \mathbf{c_t} \tag{2}$$

Where $\mathbf{z_{i,t}}$ are the stock characteristics and $\mathbf{c_t}$ the macro economic variables plus a constant. So in total there are $74 * (8 + 1) = 666$ predictors plus one constant term.

Since stock fundamentals are sometimes updated or released with delay with respect to their quarter, it causes a look-ahead bias[2], if I use this information at that quarter if it was at that time not available. To avoid this look-ahead bias of predictors I use the publicity date of variables to match them to the closest following quarterly return. This makes sure I only use the information that was public at that moment of time.

To train the models, tune their hyper-parameters[3] and test their performance I use a train-validation-test split. The train data consists of 15 consecutive years, the validation set the following 5 years and the test set the 5 years after that, totalling 25 years of data for each run. This is used as a moving window with a move of 5 years. So in total there are 6 runs.

3.2 Predicting bankruptcy

To predict bankruptcy I look at a broader stock data set which is expanded by adding all stocks from the AMEX and NASDAQ to the data set described above, where the frequency of the data is yearly. The total amount of bankrupt companies is 590 (company is labelled as bankrupt by the CRSP delist code 574). The total of non-bankrupt companies is over 20.000 in this period (1970-2020). So this is an extreme imbalanced data set.

The train set is from 1970 to 2002 containing 375 bankrupt firms which have at least 3 years of financial ratios available before declared bankrupt. The non-bankrupt firms in the train set is also 375 where the year of the data is randomly chosen for each firm. Also these firms were not declared bankrupt through the entire period of the train and validate set.

The validation set contains 93 non-bankrupt and bankrupt firms in the period in from 2003 to 2010. Here the same constraints holds as the train.

The test set consist of all the available non-bankrupt firms (5840) and 122 bankrupt firms in the period in from 2011 to 2020. The constraints to include the stocks is that it needs at least 2 years of data.

For all the bankrupt firms I take the previous years data if the company is declared bankrupt in the last half year else I take the data from 2 years ago. The reason that the train and validate set contain only a subset of the non-bankrupt firms is to account for the imbalance of classes during the training period. Since there are way fewer observations than the asset return data set, I only take the 74 financial ratios as predictors.

Also it is important to note that almost all of these ratios are not highly correlated. This is important for the logistic model to perform well if because if there is a high correlation between variables, the model will cause unstable coefficient estimations. In figure 8 in the appendices you can find a visual Pearson correlation matrix where it can be seen that is little correlation between the predictors.

Again the missing data is imputed via a cross sectional median from that year. In figure 7 in the appendices it can be seen that for most of the ratios there is no big difference between missing data for bankrupt and non-bankrupt firms.

3.3 Data Transformation

Most of the machine learning algorithms needs some sort of data normalization or standardization to centralize the data. This is important for the convergence of the optimization algorithms they use. Not all models are subject to this like random forest and boosted trees. There are many available scalars to transform data. In this research the min-max and a quantile based robust scalar are used as hyper-parameters for the elastic net and neural network. The min-max scalar formula looks as follow:

$$z = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3}$$

This scalar transforms the data in the range of $[0, 1]$. One main drawback is that it is highly influenced by outliers since it uses the minimum and maximum only. To also use a scalar that is more robust to outliers, I use a robust scalar:

$$z = \frac{x - x_{median}}{x_{Q(0.95)} - x_{Q(0.05)}} \tag{4}$$

Both of these scalars will be applied so I can check if outliers have a impact on the results.

4 Methodology

The models used in this research are based on the paper of Gu et al. (2020). For each of these models there will be a short overview including the statistical/functional form and the estimation procedure. Finally and importantly is how to deal with over-fitting[4]. This is a common pitfall when using machine learning methods especially when using financial data where the signal-to-noise ratio is low. If a model is over-fitted on a data set it will most likely perform badly on unseen data and so not be suitable for forecasting. The asset's excess return is in its most general form described as an additive prediction error model:

$$r_{i,t+1} = \mathbf{E}(r_{i,t+1}|_t) + \epsilon_{i,t+1} \tag{5}$$

$$\mathbf{E}(r_{i,t+1}|_t) = g^*(\mathbf{x}_{i,t}) \tag{6}$$

Where $i = 1, 2, ..., N$ for the stocks and $t = 1, 2, ..., T$ for the months. The objective is to find a representation of the $\mathbf{E}(r_{i,t+1}|_t)$ as a function of the prediction variables that maximizes out-of-sample explanatory. Those predictors are denoted as P-dimensional vector $x_{i,t}$ and it is assumed that the expected return $g^*(\mathbf{x}_{i,t})$ is a function of those predictors. The bankruptcy predicting models are based on the same models as the asset return prediction models but they are converted to a binary classification version.

4.1 Elastic net regularization

4.1.1 Predicting asset returns

The simple linear model is well known in the econometric field where each predictor is regressed on the target variable where the model assumes a simple linear form for the conditional expected return in equation 2:

$$g(x_{i,t}) = x_{i1,t}\beta_1 + x_{i2,t}\beta_2 + ... + x_{iP,t}\beta_P = \mathbf{x}_{i,t}^\top \boldsymbol{\beta} \tag{7}$$

The criteria to estimate is a L2 loss function or also called 'mean squared error' and there is an analytical solution to find its minimum:

$$\hat{\boldsymbol{\beta}} \equiv \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \sum_{t=1}^{T} (r_{i,t+1} - \mathbf{x}_{i,t}^\top \boldsymbol{\beta})^2 \tag{8}$$

One pitfall of using OLS to estimate this model is that in presence of many predictors, the model becomes inefficient or even inconsistent. Even though there are $N * T \gg P$ observations it is important to compare P only with N because stock returns share strong cross-sectional dependency. To overcome this issue the elastic net regularization is used as in Zou and Hastie (2005). This regularization adds two extra terms to the criteria used to estimate the linear model.

$$L(\boldsymbol{\beta}; \lambda, \rho) = \sum_{i=1}^{N} \sum_{t=1}^{T} (r_{i,t+1} - \mathbf{x}_{i,t}^\top \boldsymbol{\beta})^2 + \lambda\rho \sum_{j=1}^{P} \beta_j^2 + \lambda(1-\rho) \sum_{j=1}^{P} |\beta_j| \tag{9}$$

The first term is the mean squared error (L2 loss) that ordinary least squares uses. When setting $\rho = 1 \; or \; 0$ than the second and third terms come from Ridge or LASSO regression methods respectively. They are also known as $L2$ and $L1$ regularization. The Ridge is a shrinkage method, which means that it draws all coefficients closer to 0 but not exactly 0. This is useful when there are highly correlated independent variables (multicollinearity). But it is not usable for variable selection, because it will not set variables to 0. This is due the fact that the closer a variable gets to 0 the smaller the gradient becomes and thus has less impact on changing the variable. On the other hand the LASSO is a variable selection method. This regularization does both continuous shrinkage and variable selection simultaneously, due the nature of the $L1$ penalty. In this case the gradient will stay the same regardless of how close it is to 0. So, this results in setting a variable to 0 faster. But for highly correlated variables it tends to select only one and does not care which variable and this is not useful if I want to select the variables with the highest prediction power. The elastic net combines both the $L1$ and $L2$ penalties of the Ridge and LASSO methods to get in the middle of these regularization methods. Figure 1 visualizes these penalties.

To estimate the model parameters, the minimum of criteria 1.4 needs to found. Because this loss function is convex it can easily be found because there is a unique minimum. To solve this convex problem, the accelerated proximal gradient algorithm is used. The hyper-parameters λ and ρ are tuned via the validation set.

$$\hat{\boldsymbol{\beta}} \equiv \arg\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}, \lambda, \rho) \tag{10}$$

So to conclude, the linear elastic net regression handles the high dimensionality by automatic selection and shrinkage of the variables. Possible limitations of the linear model is that it assumes and captures only linear relationships and only considers one variable at a time. So there is no interaction between predictors, which may miss significant patterns that are in the data.

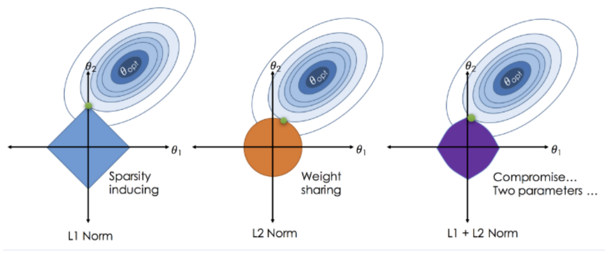

Figure 1: Comparison $L1$, $L2$ and $L1 + L2$ penalties

4.1.2 Predicting bankruptcy

To transform the above model to a classification model the linear specification is replaced with a logit model:

$$p(Bankrupt)_i = \frac{e^{\beta_0+\beta_1 x_i+\ldots+\beta_P x_i}}{1+e^{\beta_0+\beta_1 x_i+\ldots+\beta_P x_i}} \tag{11}$$

This forces the outcome of the value to be between 0 and 1 which will be the probability of being a bankrupt company according to the model. I also use a other loss function because the mean squared error doesn't work well with binary outcomes. This is because using MSE results in a non-convex function when used with binary outcomes. The following loss function is specially designed for binary outcomes and its called binary cross-entropy or log loss:

$$L(p(Bankrupt)) = -\frac{1}{N}\sum_{i=1}^{N} y_i log(p(Bankrupt)) + (1 - y_i)log(1 - p(Bankrupt)) \tag{12}$$

The same estimation and regularization as described as above is used to estimate the optimal parameters to minimize this loss function.

4.2 Boosted regression and random forests

Trees have become a popular machine learning model to incorporate interaction between variables. As a result this allows for non-linear relationships which can unveil patterns that simple linear models can not. A tree consists of nodes, leaves and branches. Figure 2 shows a basic example of a regression tree. At the top is the node that represents the entire sample. At each node the sample is split into two by a splitting rule. A node where there is no split is called a leave node. The predicted value is the average of the observations that are in the leave node. As you can see at a depth of L in the tree there can be $L - 1$ interactions between variables. In the example the data is split into two at the first step. In the second step this is also the case and now there is a interaction between predictor X2 and X1, this can be extended to multiple predictors. This also results in non-linear relationships as you can see in the plot of the example. A simple linear model is not capable of doing both of these things.

The models can easily be changed to a binary classification because they use a binning method.

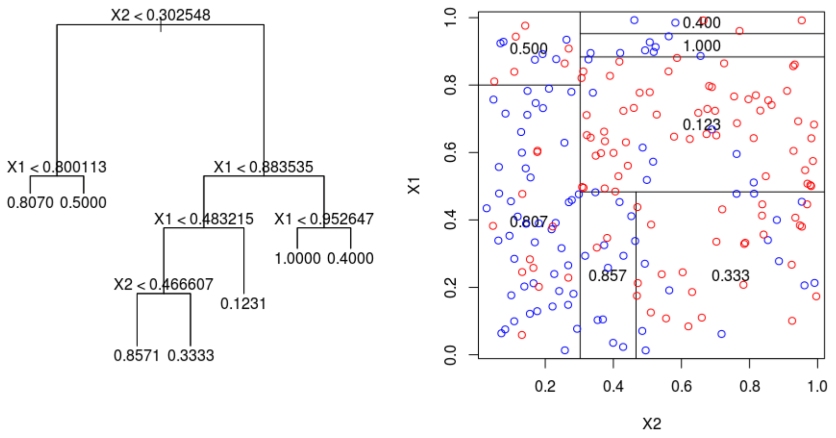

Figure 2: Basic example regression tree

The prediction of a tree τ with K leaves and depth L can be written as follows:

$$g(x_{i,t}; \theta, K, L) = \sum_{k=1}^{K} \theta_k \mathbf{1}[x_{i,t} \in C_{k(L)}] \tag{13}$$

Where $C_K(L)$ is one of the K leave nodes and each leave node is the result of a maximum of L binary splitting rules based on the predictors. θ_k denotes the sample average of the samples that end up in the leave node k.
To grow a tree there needs to be a heuristic to choose a variable at each node and also a boundary value of that variable that splits the sample into two. This heuristic needs to have a so called 'impurity', which measures the loss and it selects the variable and bin that minimizes this impurity at each node. In this research the popular $L2$ impurity is used.

One advantage of trees is that there is no need to normalize[5] or standardize[6] the predictors and as well naturally accommodates categorical and numerical data in the same model. When depth L of the tree is increased, the amount of splits is increased as well this causes also more interaction between predictors. This results fast in a over-fitted model. This is also the biggest drawback of the trees, they are likely to over-fit.

Over-fitting is a main drawback of machine learning models. Especially in the case where the signal-to-noise ratio is low. This is due the fact that the models can allow a huge amount of degrees of freedom which makes the model almost perfectly on the train data. However if there is a lot of noise in the data, it is likely that the model found patterns in the noise that do not mean anything. So, if the underlying data changes a little bit than the patterns don't hold anymore which results in a failing model for unseen data. This problem is also called the bias-variance trade off. If you want a lower bias than the variance of the model will increase and the other way around. To prevent over-fitting one can reduce the number degree of freedom of the models. With trees this can be done by selecting the max depth a tree can go or how many leaves it can have. See Neal et al. (2018) for information about the bias-variance trade-off.

4.2.1 Boosted trees

Two regularization methods for trees are used in this research: The first one is 'boosting'. Boosting recursively combines forecasts of many weak learners (shallow trees eg: short depth $L = 1$) so that each preceding learner uses the error of the previous to create a new forecast. But the forecasts improvement of the second tree is shrunken with a factor $\alpha \in (0, 1)$ also called 'learning rate', this is to prevent over-fitting on the residuals. This is repeated until there are B trees in the ensemble.[7]

- $f_1(x_{i,t}) \approx r_{i,t+1} \implies \epsilon_1 = r_{i,t+1} - \alpha f_1(x_{i,t})$

- $f_2(x_{i,t}) \approx r_{i,t+1} - \alpha f_1(x_{i,t}) \implies \epsilon_2 = r_{i,t+1} - \alpha f_1(x_{i,t}) - \alpha f_2(x_{i,t})$
- $f_3(x_{i,t}) \approx r_{i,t+1} - \alpha f_1(x_{i,t}) - \alpha f_2(x_{i,t})$

This results in three hyper-parameters (L, α, B, f). B is the number of consecutive trees that are trained on the preceding error, α (learning rate) is the ratio that the tree contributes to previous error, L is the depth a individual tree can go and f the amount of randomly selected predictors the tree can use. usually the lower the f, α and L the higher the bias and the lower the variance.

4.2.2 Random forests

The other regularization method is using a random forest model which is also a ensemble method. Like the name 'forest' suggest, the model is a collection of trees that combine their forecasts into one average forecast. This is a more general approach also called bootstrap[8] aggregation or 'bagging' Breiman (2001). It will draw B different bootstrap samples of the data to fit separate trees to each. By doing so it will reduce the correlation between the samples. To prevent over-fitting one can also choose a random subset of the predictors for each tree, also called 'dropout' method. This will result that not a strong predictor does not dominate in all trees. For this model there are three hyper-parameters: (L, B, f) where f is the number of predictors in each tree.

Both of these model have the hyper-parameter B which is the total number of trees. This parameter has less of a impact when the size increases after a certain amount. This parameter will be set high enough so stable models will be created but will not be tuned any further.

4.3 Neural networks

4.3.1 Predicting asset returns

Currently one of the most powerful machine learning methods is the Neural Network. One of the main advantages is that the model allows to non-linearly transform and make combinations of the predictors to unveil possible hidden patters in the data. There are many variants but this research will focus on the 'feed-forward' networks. The model consist of a input layer, hidden layers and a output layer. The input layer consist of the predictors. This will be passed on to possibly more than one hidden layer which is in state to transform the predictors of the preceding layer. This will be aggregated in the end by the output layer, the predicted value.

In figure 3 you see a simple neural network consisting of three input neurons one hidden layer with four neurons and one output neuron who's value can be used for evaluation. Each of hidden neurons will get the input variables plus a constant $((1, x_1, x_2, x_3),)$ and transforms the output as via a non-linear activation function[9] as (for first hidden neuron) $x_2^{(1)} = f(\beta_{1,0}^{(0)} + \sum_{j=1}^{4} x_j^{(0)} \beta_{1,j}^{(0)})$, This can then be extended by more hidden layers where the next hidden layer neurons will receive the outputs of the preceding hidden layer neurons. In the end the output neuron will transform the proceedings hidden layers output to approximate the asset's excess return function as following:

$$g(x;\beta) = \beta_0^{(1)} + \sum_{j=1}^{4} x_j^{(1)} \beta_j^{(1)} \tag{14}$$

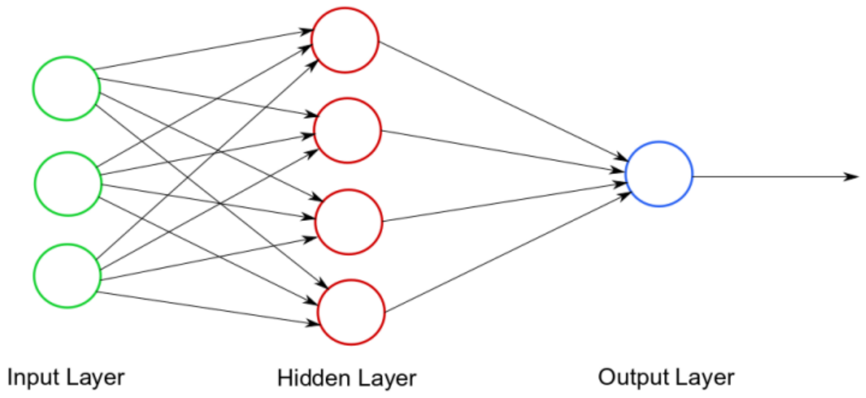

Figure 3: Basic neural network

There are a lot of hyper-parameters to be tuned like the number of hidden layers, how many neurons in each of those hidden layers and also for the non-linear activation function. This research will focus on the model used in Gu et al. (2020), where there are 3 hidden layers with the amount of neurons according to the geometric pyramid rule by Masters (1993). There are many activation functions like sigmoid, ReLu and softmax. Here the ReLu will be used since it has gained popularity for regression problems in the literature. For the output layer there is no special activation function but a simple linear function. The ReLu and neural network look as follow:

$$ReLu(x) = \begin{cases} 0, & \text{if x < 0} \\ x, & \text{otherwise} \end{cases} \tag{15}$$

$$x_k^{(l)} = ReLu\left(x^{(l-1)\prime} \beta_k^{(l-1)}\right) \tag{16}$$

$$g(x;\beta) = x^{(L-1)\prime} \beta^{(L-1)} \tag{17}$$

Where $K^{(l)}$ is the number of neurons in layer $l = 1,2,3$. $x_k^{(l)}$ is the output of neuron k and layer l, to initialize this in the input layer it is set to $x^{(0)} = (1, x_1, ..., x_P)^T$.

To estimate this model the I use the MSE loss function on the output neuron prediction. Unlike trees that require a 'greedy'[10] heuristic, training a neural network allows for updating all parameters simultaneously. This is a big advantage of neural networks over trees. However, due the high degree of non-linearity, non-convexity and high number of parameters, makes it computationally intensive. In this research I use the stochastic gradient decent method (SGD) is used to optimize the objective function. To control the learning rate the algorithm of Kingma and Ba (2014) is used. To prevent over-fitting there are four methods used: $L1$ penalization on the weight parameters (like LASSO), early stopping[11], batch normalization[12] and a ensemble approach.

4.3.2 Predicting bankruptcy

For the bankruptcy model I make the following changes. The amount of predictors is a lot smaller, so I use a simple network with only one hidden layer of 30 neurons that is almost half of the predictors set size. Also to convert it into a binary classification the linear output activation function is replaced with sigmoid activation function. The output will be between 0 and 1. This has the same form as described in the logit model (11). Also the loss function is the same as in (12).

4.4 Model evaluation

4.4.1 Predicting asset returns

To access the prediction performance of equation 2, the out-of-sample R^2 is used defined as:

$$R^2_{oos} = 1 - \frac{\sum_{(i,t)\in\tau_3}(r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{(i,t)\in\tau_3} r^2_{i,t+1}} \tag{18}$$

τ_3 defines the test sample set, whose data is never used for fitting or tuning the model. Where the denominator is not demeaned, because I want to measure how much the model explains the total variance of the returns. Usually predicting the returns with the historical mean under-performs the 0 prediction that's why the metric is not demeaned like the normal r-squared. To test if the model are significantly different from each other, the Diebold Mariano test is used (DM) Dybold and Mariano (1994). The DM statistic will be calculated by averaging the DM for each year in τ_3. This is done to prevent the auto-correlations in the errors which is important for the asymptotic normality. The Newey-West standard errors will be used to make the standard deviation more robust. Where $n_{3,t+1}$ is the size of stocks in testing sample $t+1$. The statistic looks as follow:

$$DM = \frac{\sum_{i=1}^{n_{3,t+1}}[(\hat{e}^{(1)}_{i,t+1})^2 - (\hat{e}^{(2)}_{i,t+1})^2]}{n_{3,t+1} * \hat{\sigma}} \tag{19}$$

4.4.2 Predicting bankruptcy

One common method to evaluate the performance of binary classification models is the accuracy of the model. That is, the ratio of correctly classified instances with respect to the total. In this research this is a useless method since if I would predict all instances as non-bankrupt this would lead to a accuracy of $5840/5962 = 97.95\%$. But ofcourse this model would be useless. To overcome this issue there are other metrics available which I will use:

- Type I error: $\frac{FP}{TN+FP}$, the ratio of positive instances that were predicted to be negative.
- Type II error: $\frac{FN}{TP+FN}$, the ratio of negative instances that were predicted to be positive.
- Precision: $\frac{TP}{TP+FP}$, the ratio of correctly specified positive instances.
- F1: $\frac{2*TP}{2*TP+FP+FN}$, the weighted average of Precision and Recall (Recall = 1 - Type I error).

The F1 score is on its own a good metric to compare the overall performance across models. During training and validating the amount of non-bankrupt and bankrupt is the same so the accuracy score will be used to tune the hyper-parameters. Usually there is a trade off between type I and II errors and one can decide which one they find more important. In this case there is a trade-off between: a model that it almost not miss bankrupt firms but at the cost it will predict more non-bankrupt as bankrupt (lower type I precision and higher type II) or a model that is more precise but misses more bankrupt cases. To analyse the trade of this I use the receiver operating characteristic (ROC) curve. This curve shows the trade off between the true positive rate and the false positive rate when adjusting the threshold. The threshold is the probability that cut-offs the positive cases from the negative ones. In this research I use a threshold of 0.5. The ROC is perfect when it has a area under the curve (AUC) of 1. a unskilled model has a AUC of 0.5.

4.5 Feature analysis

To investigate what the contribution of a feature is in a model, I use a method called feature importance permutation. The method works as follow:

1) Select one of the features or one of the grouped features (In the asset return case, a ratio and the ratio multiplied with the macro features).

2) Shuffle the observations in the train set randomly for each of the selected feature.

3) Test the modified train set and measure its R^2_{oos} / F1-score decrease with respect to the original R^2_{oos} / F1-score.

4) Repeat step 2 & 3 multiple times (I did it 10 times) to get a more robust average percentage.

5) Repeat for the remaining features or grouped features and normalize the average percentage changes so that it sums up to 1 (This allows for comparing across models).

The reason why I don't refit the model with removing the grouped features, is that it is too computational heavy to be feasible. This approach is less accurate but it is way faster. By shuffling the observations the model will fail more if the model relies on the features, so this method reveals the most important predictors of the model. Methods like random forest have the advantage that feature importance is easily available through the model, but to compare across models I use this method for all the models.
to investigate the financial ratios, I will get the features corresponding to this ratios totalling in 9 predictors each to form the group as in step 1. For the bankrupt model I will analyse all the features separately.

4.6 Extension predicting bankruptcy: Meta labeling

The idea of meta labeling is first introduced in de Prado (2018). Meta labeling is useful when you want to achieve higher F1-scores. First, build a model that achieves a low type I error, even if the precision is not high. Second, build a model that uses the predicted probabilities of the first model to learn how to use the primary model as input, so it can filter false negatives. This will usually result in higher F1 scores. This is a kind of ensemble method where it uses more than one model to make predictions. The procedure to apply meta labeling is as follow:

1) Train a primary model with all the predictors available and predict the probabilities of positive instances.

2) Use the probabilities in step 1 in a second model together with all the predictors as input features.

3) Create the meta labels: 1 if the primary model classified the instance correctly (using a threshold 0.5 in this research), 0 otherwise.

4) Train the second model with the updated labels and feature set.

5) A instance will be classified if both models classify the instance as positive otherwise it will be classified as negative.

This method allows to compare different types as models by using different models for the primary and secondary model. I will test all the combinations between the models but I will leave either random forest or gradient boosted based on the weaker model. I do this because the trees have similar model structure and meta labelling benefits from combining two different models. Also this results in $3 * 2 = 6$ two-step models instead of $4 * 3 = 12$.

5 Results

5.1 Predicting asset returns

The hyper-parameters used to train the models can be found in table 6 in the appendices. Table 1 shows the performance of the out-of-sample R^2 metric for all the machine learning models used and also the benchmark OLS-3 model. The Benchmark model preselects size, momentum and book-to-market predictors. The R^2 is a average of all the testing periods that start at 1990 and ends in 2020. The first row reports the R^2 of all the companies in the testing period. The results show for all models a very low R^2, this indicates that all models have low predict power to predict quarterly asset returns. The gradient boosted regression tree performs the worst. This is probably the cause of tuning the hyper-parameters wrong for the model, but due to high computational complexity of the model I was not able to rerun the model with a expanded set of hyper-parameters. Gu et al. (2020) reported in their research that by testing the model on the top 1000 and bottom 1000 companies each month by market size, will produce different outcomes in the R^2. I did the same for the top 100 annd bottom 100. The top 100 companies by market size each quarter show a large negative performance. Table 2 conducts the analysis to a annual horizon. The models still have a similar performance

	OLS-3	Elastic Net	Random Forest	Boosted Trees	NN3
All	0.59%	0.62%	0.49%	-5.06%	0.48%
Top 100	-3.17%	-0.87%	-3.49%	-12.27%	-1.43%
Bottom 100	0.27%	0.30%	0.88%	-2.12%	0.26%

Table 1: In this table, the quarterly R^2_{oos} percentage performance is shown for the model. Also the top-100 stocks and bottom-100 stocks by market equity size are tested as a sub-sample on the model that is fitted on all the stocks.

compared to each other but I see that the R^2 is now approximately five times larger, this increase shows that the models are able to forecast annual returns (which persists over business cycle frequencies) better than quarterly returns (which are more exposed to specific seasonlity of a company). Again I see lower values for the top 100 firms. Table 3 shows if the performance differences in table 1 are actually statistically significant.

	OLS-3	Elastic Net	Random Forest	Boosted Trees	NN3
All	2.09%	2.59%	2.42%	2.34%	2.78%
Top 100	-6.62%	0.29%	-1.06%	0.14%	-0.12%
Bottom 100	0.82%	1.35%	2.51%	1.03%	1.56%

Table 2: Annual return forecasting R^2_{oos} (see table 1 for the explanation). The quarterly data is aggregated annually starting the year in May.

The table reports pairwise Diebold-Marnio statistics of a column model versus a row model. A positive statistic indicates that the column model outperforms the row model. Bold numbers indicate a significance at the 5% level, which are calculated with Newey-West standard errors. I conclude from this table that no model outperforms the simple benchmark OLS-3 model. So the machine learning models are not able to extract any extra use-full predictive power out of the large predictor set. This is most likely the cause that the financial ratios contain almost no predictive information for assets returns and that it is not the models fault that it performs bad. Finally to compare the most important predictors used by the models, the permutation feature importance procedure is used as explained in the methodology. Figure 4 shows these results. The models seem not to agree on the most important features. Also the three features of the OLS-3 model are not in the top features of the models. This is a sign of low predictive power in the features, which causes the models to not select the same features but more random low predictive ones. This is also in line with the low R^2 performance.

	Elastic Net	Random Forest	Boosted Trees	NN3
OLS-3	-0.13	0.326	**-29.66**	**-2.20**
Elastic Net	-	0.40	**-28.88**	**-4.85**
Random Forest	-	-	**-21.29**	-1.20
Boosted Trees	-	-	-	**28.24**

Table 3: This table shows pairwise Diebold-Mariano test statistics. A positive numbers indicates that the column model outperforms the row model. Bold means a 5% significance.

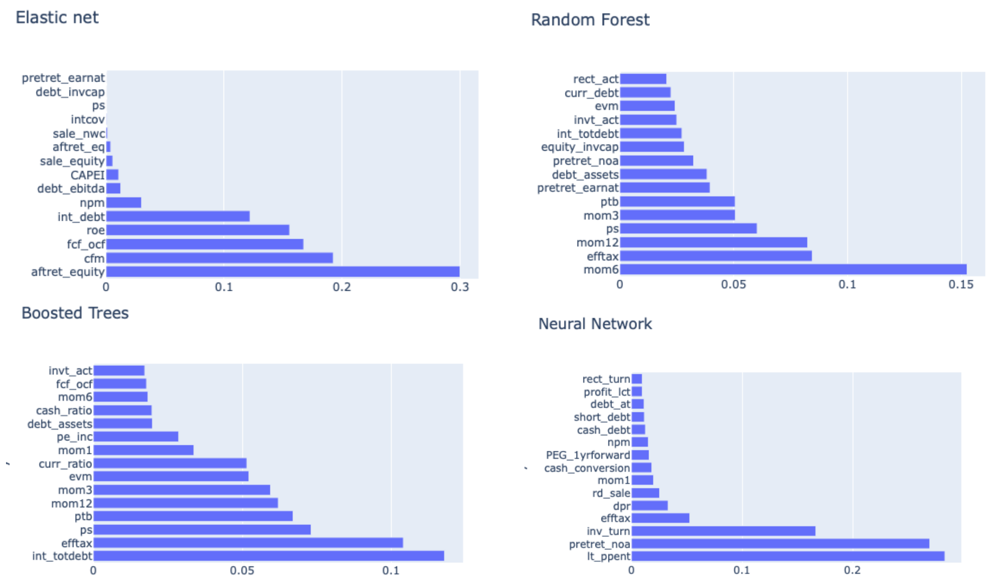

Figure 4: Average variable importance for the top-15 most influential variables in each model.

5.2 Predicting bankruptcy

The hyper-parameters used to train the models can be found in table 7 in the appendices. Table 4 shows the outcome of the machine learning models on the unseen test set. The gradient boosted trees shows the best performance in both the type 1 and type 2 errors, precision (P), area under the curve (AUC) and also in the F1 score. The model predicts 104 out of 122 bankrupt companies correct out of 5962 companies in total. The promising neural network performs the worst but this can be the cause of a wrongly constructed model or the low number of observations to train the model on. Important to note is that all the models after tuning train the final model on both the train and validation set. Since I use a early stopping method for the neural network, I can not train the final model on both the train and validation set. This also causes to a fewer observations of training which can be the cause of the poorer performance. For all the models I see a high rate of false positive (type 2 error) as expected because of the extreme imbalanced cases (non-bankrupt: 5840 & bankrupt: 122, in the test set). All of the results in the table are constructed with a threshold of 0.5, which means if the probability of bankrupt is higher than 0.5 it will be classified as bankrupt otherwise non-bankrupt.

Model	TP	FP	TN	FN	Type I Error (%)	Type II Error (%)	AUC (%)	P (%)	F1 (%)
Logistic	98	1385	4455	24	19.67	23.72	83.65	6.61	12.22
Random Forest	99	1195	4645	23	18.85	20.46	90.12	7.65	13.98
Boosted Trees	104	1091	4749	18	**14.75**	**18.68**	**90.42**	**8.70**	**15.79**
Neural Network	94	1396	4444	28	22.95	23.90	83.14	6.39	11.66

Table 4: This table shows classification metrics for all the models on unseen test data to predict bankruptcy from 2011 to 2020. Type I error means the portion of bankrupt companies that were predicted to be non-bankrupt. Type II error means the portion of non-bankrupt companies that were predicted to be bankrupt. AUC is the area under the ROC curve, and P is precision. Bold indicates that the model has the best performing metric.

Figure 5 shows the Receiver operating characteristic curve (ROC) for each of the models. This curve shows the change of true/false positive rate when using different thresholds. The gradient boosted trees model shows the most promising model followed by the random forest, which is also in line with the results in table 4.

Figure 6 shows the variable importance of the financial ratios used in the models. The neural network has no preferred top features that is relies on. This is maybe also the reason that the model did not perform well compared to the other models since it was not able to capture the most predictive ratios. The other three models share some ratios in their most important variables. The logistic model has less in common than the random forest and boosted trees. This shows that allowing for interaction of predictors can reveal new patterns and increase predictive power. Some important ratios are: Price/Sales (ps), Total Debt/Total As-

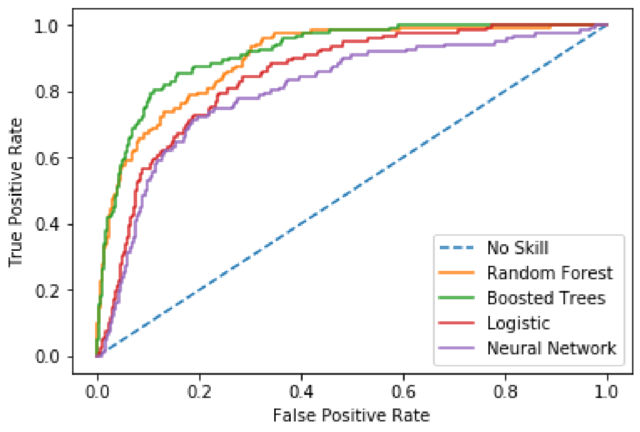

Figure 5: Receiver operating characteristic (ROC) curve of all the models on the unseen test data to test bankruptcy from 2011 to 2020. Gradient boosted trees shows greatest predictive performance.

sets (debt-assets), Total Debt/Capital (debt-capital), Pre-tax Profit Margin (ptpm), Inventory/Current Assets (invt-act) and Price/Book (ptb). These ratios are also found in Barboza et al. (2017) but in a different form like percentage change or absolute change.

As explained in the methodology a method called meta-labelling is used to lower this type 2 error with the goal of minimum increase of type 1 error. Table 2 show the results of these 2-step procedure modelling. Here the first model represents the model that calculated the probability of bankrupt which is than used as a feature in the second model. This allows that the second model learns how to use the first model to make better predictions. If both models agree on a true (bankrupt) prediction that it is classified as true otherwise as false (non-bankrupt). The table show promising results. All the 2-step models have higher F1-scores expect for the neural network / logistic model (these were also the least performing models in table 4). The Boosted Trees / Logistic model has the highest F1 score, which is more than 20% higher than the boosted trees model alone. For all the models I see that the false positive cases is around 300-500 lower which is a great improvement. For some models this at a cost of having more false negatives (which is a bad thing since I don't want to misclassify bankrupt companies). For the Boosted Trees / Neural Network model I see that type 1 error rate is the same as the best model in table 1 but the type 2 error is a lot lower. So again this model performs better than the best model of table 1 if I are only interested in lowering the type 2 error without increase type 1 error

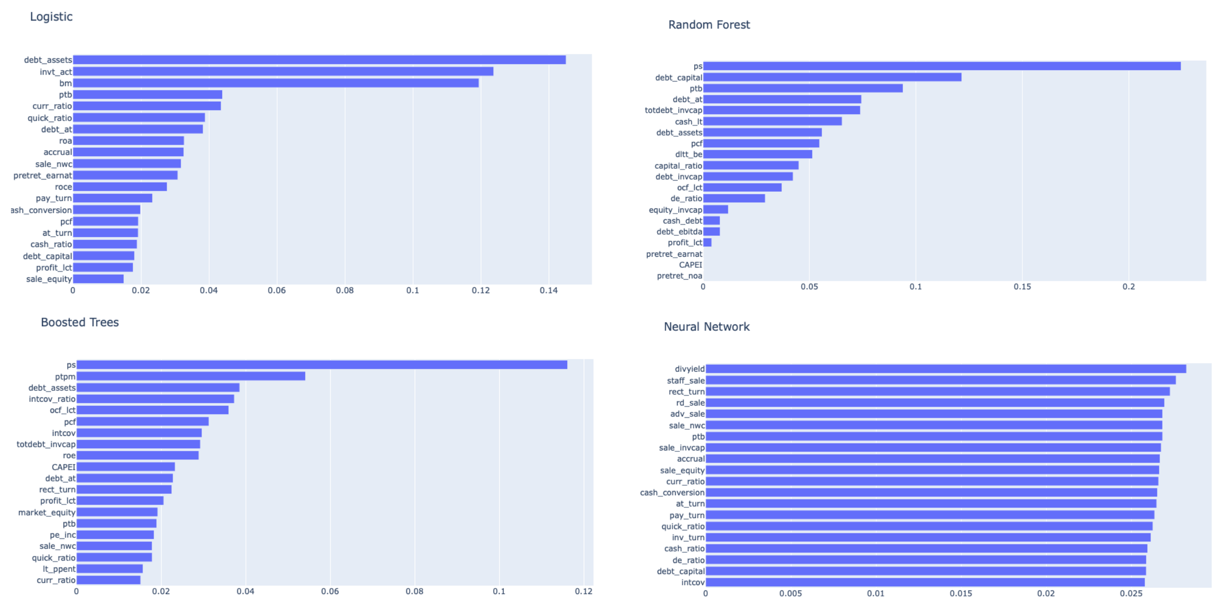

Figure 6: Variable importance for the top-20 most influential variables in each model. Variable importance is an average over all training samples. Variable importance within each model is normalized to sum to one.

1st Model / 2nd Model	TP	FP	TN	FN	Type I Error	Type II Error	P (%)	F1 (%)
Boosted Trees / Logistic	98	804	5036	24	19.67	13.77	**10.86**	**19.14**
Boosted Trees / Neural Network	104	923	4917	18	**14.75**	15.80	10.13	18.10
Logistic / Neural Network	83	870	4970	39	31.97	14.90	8.71	15.44
Logistic / Boosted Trees	93	779	5061	29	23.77	**13.34**	10.67	18.71
Neural Network / Boosted Trees	87	831	5009	35	28.69	14.23	9.48	16.73
Neural Network / Logistic	88	993	4847	34	27.87	17.00	8.14	14.63

Table 5: This table shows classification metrics for all the 2 step models with meta labelling on unseen test data to predict bankruptcy from 2011 to 2020. Type I error means the portion of bankrupt companies that were predicted to be non-bankrupt. Type II error means the portion of non-bankrupt companies that were predicted to be bankrupt. Bold indicates that the model has the best performing metric.

6 Conclusion

Using financial ratios as the base of a predictor set, I perform comparative analysis of methods in the machine learning area by predicting asset return and bankruptcy. My results show that using ratios as predictors for asset returns perform poorly and that there is almost no predictive power. The elastic net achieved the a R^2 of 0.62% and 2.59% for the quarterly and annual horizon respectively. All the models: Elastic net regression, random forest, gradient boosted trees and a neural network were not able to outperform the benchmark model OLS-3 (using only a momentum, book-to-market and size predictor). This results is also the same for an annual horizon instead of a quarterly. This is all concluded using Diebold-Mariano statistics to test for significance of better predictability. Also using permutation feature importance I analysed the feature importance across the models. The result of this is that the models fail to agree on the same features. This indicated that the ratios contain little information for asset pricing and may not be suitable to use as predictors. However with predicting bankruptcy, the ratios achieved promising predictive metrics and these were even improved by the meta labeling extension. This meta labeling is a two-step modelling approach where the second model learns from the first models mistakes. The gradient boosted trees achieved a F1 score of 15.79% and the two-step modelling using the gradient boosted trees and the logistic elastic net improved this to 19.14%. Also I conclude that the complex models like neural networks are harder to tune and require more knowledge to do so, since I was not able to let the neural network outperform the simple logit model. The random forest and gradient boosted classifier showed the best results and shared some of their most used predictors in the model like: Price/Sales (ps), Total Debt/Total Assets (debt-assets) and Total Debt/Capital (debt-capital). These ratios are also found in Barboza et al. (2017) but in a different form like percentage change or absolute change. This research did not focus on feature selection methods but I used a automatic regularization penalty. Some issues in this research were the training and tuning of the models when there are a lot of variables and observations.

A possible extension could be looking how to decrease the number of observations that are closely related in the asset return pricing problem since most of the models assume independent and identically distributed observations (I.D.D.) but this is clearly not the case. For the bankruptcy prediction a interesting extension would be to implement macro economic variables because the number of bankruptcy firm per year seems to non constant over the years. Also future studies should extend the analysis to incorporate the time effects of variables since I only used the cross-sectional effect which assumes a constant over the entire period which is a strong assumption that is probably not valid.

Notes

[1]A risk premium is the investment return an asset is expected to yield in excess of the risk-free rate of return. An asset's risk premium is a form of compensation for investors. It represents payment to investors for tolerating the extra risk in a given investment over that of a risk-free asset.

[2]Look-ahead bias is a type of bias that occurs when a study or simulation relies on data or information that was not yet available or known during the time period being studied. It generally leads to inaccurate results from a study or simulation

[3]In machine learning, a hyper-parameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are derived via training.

[4]Over-fitting is the case where the overall cost is really small, but the generalization of the model is unreliable. This is due to the model learning "too much" from the training data set

[5]This technique is to re-scale features value with the distribution value between 0 and 1 is useful for the optimization algorithms, such as gradient descent, that are used within machine learning algorithms that weight inputs (e.g., regression and neural networks).

[6]The result of standardization (or Z-score normalization) is that the features will be re-scaled to ensure the mean and the standard deviation to be 0 and 1, respectively.

[7]Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.

[8]The bootstrap method is a re-sampling technique used to estimate statistics on a population by sampling a data set with replacement.

[9]Activation Functions make the decision of whether or not to pass a signal to the next layer. They take in the weighted sum of inputs plus a constant as an input

[10]A greedy algorithm is an algorithmic strategy that makes the best optimal choice at each small stage with the goal of this eventually leading to a globally optimum solution. This means that the algorithm picks the best solution at the moment without regard for consequences. It picks the best immediate output, but does not consider the big picture, hence it is considered greedy.

[11]Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation data set.

[12]Batch normalization (also known as batch norm) is a method used to make artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling

Appendices

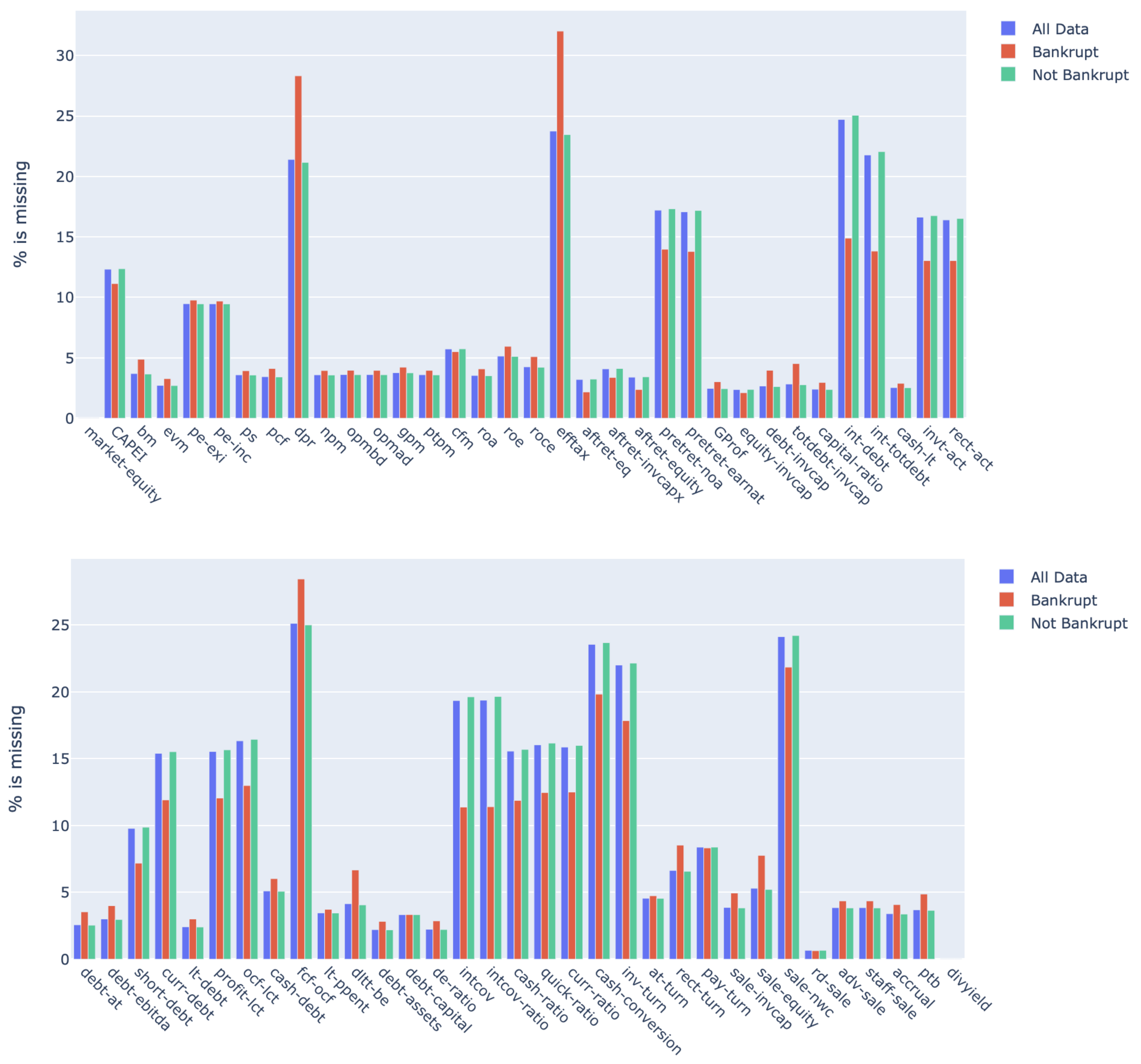

Figure 7: Total observations that are missing in percentage for all the ratios. These are filled by a median imputer with the median of each year.

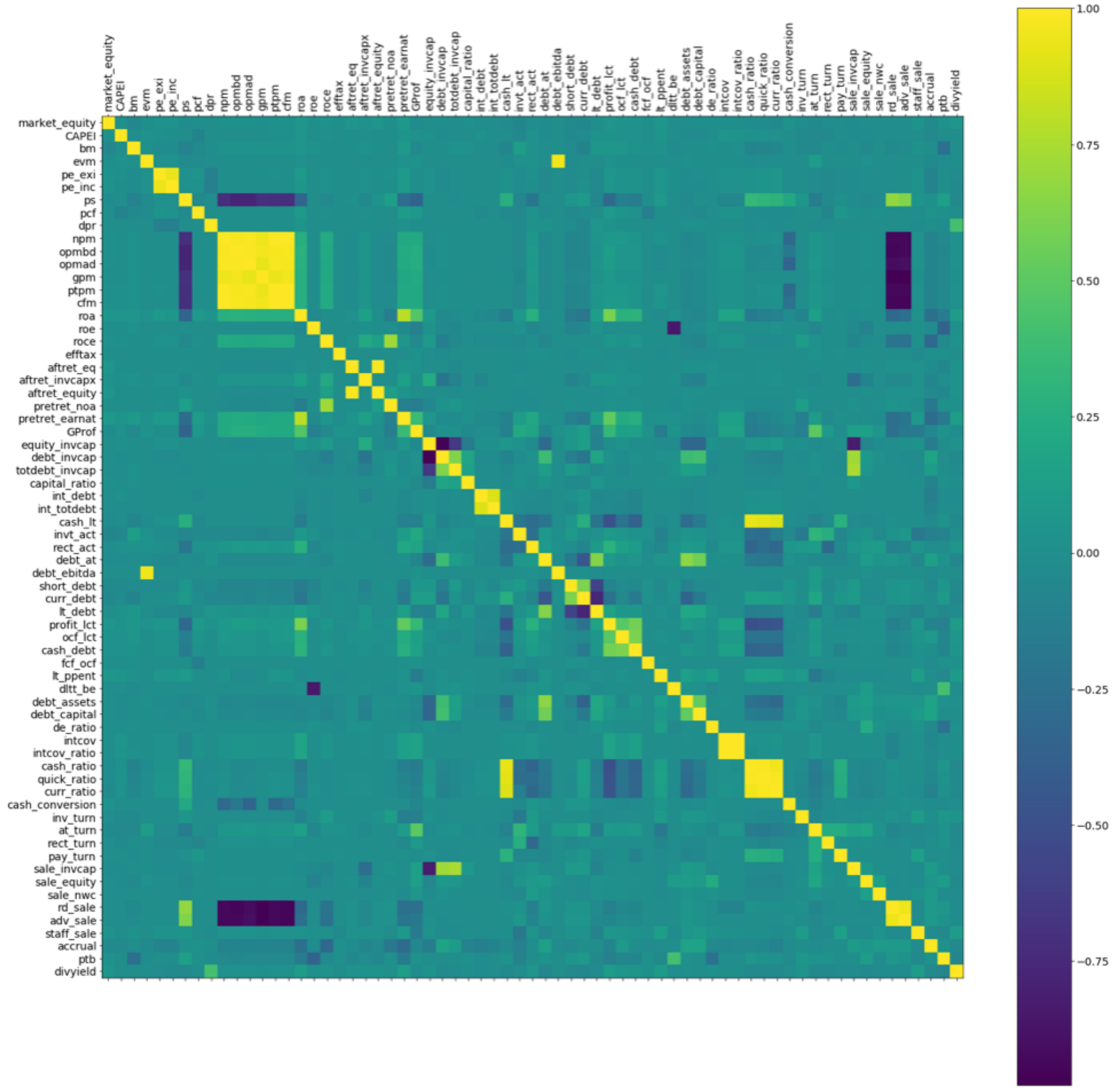

Figure 8: Pearson correlation between all the ratios.

	OLS-3	Elastic Net	Random Forest	Boosted Trees	NN3
Robust Scaler Q(0.05) & Q(0.95)	×	✓	×	×	✓
Min Max Scaler	×	✓	×	×	✓
others	-	$\rho \in \{0.25, 0.5, 0.75\}$	Depth=1-8	$LR \in \{0.1, 0.01\}$	LR $\in$ {0.1, 0.0
		$\lambda \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	NF $\in \{10, 30, 60\}$	NF $\in \{10, 30, 60\}$	L1 $\in$ {10^{-3}, 10
			Trees=300	Depth=1-8	Epochs=100
				Trees=1-300	Patience=5
			split=MSE	Loss and split=MSE	Adam param=De

Table 6: Tested hyper-parameters for each of the regression models for asset return predictability.

	Logistic	Random Forest	Boosted Trees	Neural Network
Robust Scaler Q(0.05) & Q(0.95)	✓	×	×	✓
Min Max Scaler	✓	×	×	✓
others	$\rho \in \{0, 0.25, 0.5, 0.75, 1\}$	Depth=1-10	$LR \in \{0.1, 0.01, 0.001\}$	LR $\in \{0.1, 0.01\}$
	$\lambda \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	NF $\in \{5, 10, 15, 20\}$	NF $\in \{5, 10, 15, 20\}$	L1 $\in \{10^{-3}, 10^{-5}\}$
		Trees=500	Depth=1-10	Epochs=1000
			Trees=1-500	Patience=5
		Split=gini	Loss and split=MSE	Adam param=Defau

Table 7: Tested hyper-parameters for each of the classification models to predict bankruptcy.

References

Barboza, F., Kimura, H., & Altman, E. (2017). Machine learning models and bankruptcy prediction. *Expert Systems with Applications*, *83*, 405–417. https://doi.org/https://doi.org/10.1016/j.eswa.2017.04.006

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1).

Danenas, P., & Garsva, G. (2015). Selection of support vector machines based classifiers for credit risk domain. *Expert Systems with Applications*, *42*(6), 3194–3204.

de Prado, M. L. (2018). *Advances in financial machine learning* (1st). Wiley Publishing.

du Jardin, P. (2016). A two-stage classification technique for bankruptcy prediction. *European Journal of Operational Research*, *254*(1), 236–252.

Dybold, F. X., & Mariano, R. S. (1994). Dybold, f. x., and r. s. mariano. comparing predictive accuracy. *NBER Technical Working Paper*.

Fama, E. F., & French, K. R. (2008). Dissecting anomalies. *The Journal of Finance*.

Feng, G., He, J., & Polson, N. G. (2018). Deep learning for predicting asset returns.

Feng G., G. S., & D., X. (2020). Taming the factor zoo: A test of new factors. *Journal of Finance*.

Freyberger J., N. A., & M., W. (2020). Dissecting characteristics nonparametrically. *Review of Financial Studies*.

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

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

Koijen, R., & V., N. S. (2011). Predictability of returns and cash flows. *Annual Review of Financial Economics*.

Kozak S., N. S., & S., S. (2020). Shrinking the cross section. *Journal of Financial Economics*.

Lewellen, J. (2004). Predicting returns with financial ratios. *Journal of Financial Economics*, *74*(2), 209–235. https://doi.org/https://doi.org/10.1016/j.jfineco.2002.11.002

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

Masters, T. (1993). Practical neural network recipes in c++. *Academic Press*, (4).

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., & Mitliagkas, I. (2018). A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*.

Nelson, D. M. Q., Pereira, A. C. M., & de Oliveira, R. A. (2017). Stock market's price movement prediction with lstm neural networks. *2017 International Joint Conference on Neural Networks (IJCNN)*, 1419–1426. https://doi.org/10.1109/IJCNN.2017.7966019

Rapach D. E., S. J. K., & G., Z. (2013). International stock return predictability: What is the role of the united states? *Journal of Finance*.

Tian, S., & Yu, Y. (2017). Financial ratios and bankruptcy predictions: An international evidence. *International Review of Economics Finance*, *51*, 510–526. https://doi.org/https://doi.org/10.1016/j.iref.2017.07.025

Tsai, C.-F., Hsu, Y.-F., & Yen, D. C. (2014). A comparative study of classifier ensembles for bankruptcy prediction. *Applied Soft Computing*, *24*, 977–984.

Wang, G., Ma, J., & Yang, S. (2014). An improved boosting based on feature selection for corporate bankruptcy prediction. *Expert Systems with Applications*, *41*(5), 2353–2361.

Welch, I., & A., G. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies*.

Yao, L. Y., J., & L., T. C. (2000). Option price forecasting using neural networks. *Omega*.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(25).