FACULTY OF ECONOMICS AND BUSINESS ADMINISTRATION

Ensembles in stock price prediction

Word count: 10572

Joachim Depovere
Student number: 01605983

Promotor/ Supervisor: Prof. dr. Matthias Bogaert

Master's Dissertation submitted to obtain the degree of:

Master of Science in Business Engineering
Main subject: Data analytics

Academic year: 2020 – 2021

GHENT UNIVERSITY

Confidentiality agreement

Permission

I declare that the content of this Master's Dissertation may be consulted and/or reproduced, provided that the source is referenced.

Name student: Joachim Depovere

Foreword

This master dissertation marks the end of a five-year study in business engineering with data analytics as specialization. Diving deeper into the world of data analytics has been a great journey, both enriching and challenging at times. In these challenges I was always supported by great people around me which I would like to thank explicitly.

Firstly, I would like to express my gratitude towards my promotor, Matthias Bogaert. Because of his intervention I could quickly transition to a new interesting subject after my first subject was cancelled in the context of Covid-19. Despite his busy agenda as newly appointed professor I was provided quickly with helpful guidance. I really liked the structured and practical approach which still left plenty of room for me to work independently, adding my own ideas.

Secondly, a big thank you to my parents who supported me throughout all those years, without them, this opportunity would simply not have been possible. Especially when a challenge occurred, they were always there to provide advice and support.

Thirdly, my friends were a great source of motivation. Their capabilities and perseverance brought out the best in me. In addition, by studying together, a good time schedule was maintained.

Lastly, I would like to mention Stack Overflow, their forum is a great asset during programming. Coming across similar problems as the ones you are facing is both comforting and very helpful.

Table of contents

List of abbreviations

Acc	Accuracy
Ada	Adaboost
AUC	Area Under the Curve
BagTree	Bagged Tree
BS	Brier Score
CB	Cat Boost
DT	Decision Tree
EMH	Efficient Market Hypothesis
ERT	Extremely Randomized Trees
F	F-measure
FinPM	Financial Performance Measures
FN	False Negative
FP	False Positive
H	H-measure
HCE	Hill Climbing Ensemble
KNN	K Nearest Neighbors
LGB	Light Gradient Boosting
LogR	Logarithmic Regression
MCE	Mean Cross Entropy
MCS	Multiple Classifier Systems
NB	Naïve Bayes
NN	Neural Network
Prec	Precision
RF	Random Forest
ROI	Return On Investment
RotF	Rotation Forest
SA	Simple Averaging
Sens	Sensitivity
SR	Severity Ratio
Stack	Stacking
SVM	Support Vector Machines
TP	True Positive
TN	True Negative
WA	Weighted Averaging
XGB	eXtreme Gradient Boosting

List of figures

List of tables

Abstract

In the past years, many researchers have been investigating how to predict the stock market as accurately as possible. Obviously, this is one of the most challenging tasks in the financial world and ensembles are promising in this context. The objective of this paper is to benchmark a broad range of published ensembles based on a selected set of performance metrics. Ballings et al. already published in 2015 a benchmarking study showing that ensembles perform better than single classifiers by gathering data from 5767 publicly listed European companies (Ballings et al., 2015). Since then, several new ensemble models have emerged. This study will include eighteen models containing twelve ensembles in an updated benchmark, using the same dataset and considering six carefully selected performance metrics. This research confirms the superiority of ensembles compared to single classifiers and shows that heterogeneous ensembles work well in stock prediction. However, these heterogeneous ensembles do not statistically outperform all homogeneous models like for example random forest. Accordingly, we recommend random forest for stock price direction considering its ease of use and interpretability. This study contributes to the existing theory and practices by benchmarking a substantial amount of ensemble methods concerning stock price prediction. In addition, this paper considers multiple important performance metrics. To the best of our knowledge this has never been done so comprehensively in the stock market domain.

1. Introduction

In the past decades, stock price prediction has been extensively studied in literature. By minimizing the forecasting error, the investment risk is minimized which results in a financial gain (Manish & Thnmozhi, 2011). Even a slight improvement in forecasting performance could potentially result in high profits (Ballings et al., 2015; Halbleib & Pohlmeier, 2012). However, predicting the stock market is very difficult, even impossible according to two well-known theories. Firstly, the random walk theory (formulated from the martingale model), stipulates that stock prices take a random and unpredictable path that makes all methods of predicting stock prices ineffective in the long run (Umoru et al., 2020). In addition, the theory explicitly points out that historical prices and present stock performance (technical analysis) cannot be used to predict the future performance of stock prices. This supports Gujarati & Porter (2009) assertation that stock prices are fundamentally random. This means that a fundamental analysis, which makes use of financial ratios to predict the stock market, does not outperform a random model. It is therefore difficult to benefit from speculation in stock trading. Secondly, according to the Efficient Market Hypothesis (EMH) it is impossible to outperform the overall market by means of expert stock selection or market timing. The proponents of the EMH explain that the intrinsic value of a stock is always equal to its current price (Qian & Rasheed, 2007). Nevertheless, after more than half of a century of research, no consensus has been reached on the presence nor the absence of the validity of this hypothesis (Leković, 2019).

Models based on machine learning are most promising in showing evidence against the Market Hypothesis. Advancements in computing power, as well as the availability of large datasets, did lead to the introduction of techniques such as decision trees and neural networks being used in stock price prediction (Albanis & Batchelor, 2007; Nti et al., 2020). Some studies have already attempted to predict the relationship between the available information and the stock returns using simple linear models, although with little success (Butler et al., 2014). This is in line with the assumption that the relationship between the stock returns and the available information is non-linear (Tsai et al., 2011; Umoru et al., 2020). Non-linear machine learning algorithms like decision trees and neural networks solve this issue. One of the most performant and popular techniques to come up with predictions of nonlinear relationships are ensembles (Tsai et al., 2011). These ensembles are constructed by combining machine learning algorithms which compensate each other for individual weaknesses (Gomez & Rojas, 2018). In the financial world, the vast majority of

articles supports the performance superiority of ensembles over single classifiers (Ampomah, Qin, Nyame, et al., 2020; Ballings et al., 2015; Basak et al., 2019; Tsai et al., 2011). If the combined single classifiers are diverse and independent, the prediction error of the ensemble decreases significantly.

Despite the superior performance of ensembles in stock price prediction, there are only a few thorough benchmarking studies available using fundamental analysis (Albanis & Batchelor, 2007; Ballings et al., 2015; Tsai et al., 2011). However, these studies only benchmark a limited number of ensembles. In comparison to other, larger scale benchmark studies (e.g., Lessmann et al., 2015), these studies lack three dimensions: (i) the inclusion of novel classification algorithms from other financial domains like credit scoring and bankruptcy prediction, (ii) the benchmarking of both heterogeneous and homogeneous ensembles and (iii) the use of a broad range of different performance metrics. Hence, this paper contributes to literature by providing an extensive benchmarking of 18 classifiers, by checking their performance in terms of Area Under the Curve (AUC), H-measure (H), Brier Score (BS), Mean Cross Entropy (MCE), Precision (Prec) and Accuracy (Acc).

The remainder of this paper is structured as follows. Section 2 reviews the literature concerning ensembles in stock price prediction. Next, section 3 explains the methodology used to benchmark all 18 classifiers and explains them briefly. In addition, the selected performance metrics and calibration methodology are explained into more depth at the end of the section. Section 4 examines the results after which section 5 concludes the dissertation and explains the practical implications. Lastly, section 6 defines the limitations and equips the reader with ideas for further research.

2. Literature review

2.1 Ensembles

Ensembles have been applied for the first time at the end of the 1980s. Over the following years various names have been used in literature, all meaning exactly the same: ‘‘Ensemble Classifier”, ‘‘Classifier Fusion”, ‘‘Combined classifier”, ‘‘Multiple-classifier systems (MCS)”, ‘‘Ensemble Learning”, and ‘‘Hybrid classifiers” (Graña & Corchado, 2014). The technique of combining multiple classifiers has proven to be an excellent predictor in a broad range of different contexts (e.g., pattern recognition (Jan & Verma, 2020), predicting natural phenomena (Kaloop et al., 2020) and medical diagnosis (J. Zhang & Chen, 2019)). Similar to these fields, the financial sector saw great opportunities in using ensembles, for example in credit scoring to support decision making in the retail credit business (Lessmann et al., 2015). Given its immense popularity, ensembles have also been used extensively for stock price prediction. Most efforts have been devoted to predicting the most likely direction of a stock rather than its price level, because it has often resulted in more accurate trading results (Reza et al., 2020). Consequently, in this paper we won’t try to predict the exact stock price level but rather the most likely direction.

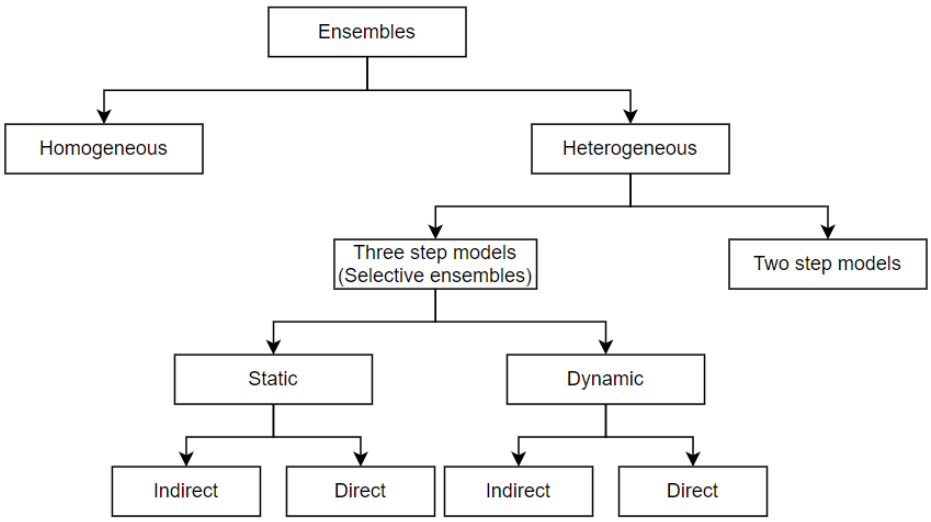

Figure 1 Ensemble framework

To structure the different ensemble methods, Figure 1 was constructed above. Firstly, ensembles can be homogeneous or heterogeneous. Every ensemble is built onto multiple base classifiers. For

homogeneous ensembles, the base classifiers all belong to the same algorithm. On the contrary, for heterogeneous ensembles, the learning algorithms of the base classifiers are different (van Rijn et al., 2018). Secondly, ensembles can consist of two or three steps (selective ensembles). For two step ensembles, these steps are called base model development and forecast combination. Selective ensembles add a third step: after the development of the base models a selection procedure is used to select a suitable subset of models to enter the ensemble (Lessmann et al., 2015). Thirdly, these selective ensembles can be static or dynamic. For static ensembles the selection procedure is executed once in contrast to dynamic ensembles (Oliveira et al., 2018) where the selection is executed multiple times. Finally, the static and dynamic selection procedures can be based on maximizing the predictive accuracy, which is called the direct approach, or on maximizing the diversity among the base models (Lessmann et al., 2015), which is called the indirect approach. An example of such selective dynamic direct approach is given in Feng et al. (2018). In this specific case, the selection procedure takes the relative costs of type I and type II error into account while selecting classifiers.

2.2 Stock price prediction

This study focusses on the prediction of stock price direction using fundamental analysis. When using fundamental analysis, the intrinsic value of a stock is considered (Albanis & Batchelor, 2007). In 1995, Fama & French (1995) already suggested that company fundamentals are correlated with excess returns. A fundamental analysis is obtained by analyzing the historical financial statements of a firm. From these statements financial ratios are derived, like debt, cashflow and current ratio (Tsai et al., 2011). Focusing on the data used in literature, it follows that only several studies (Albanis & Batchelor, 2007; Ballings et al., 2015; Tsai et al., 2014) use purely financial ratios to predict the stock market. This might be due to the sensitive data (e.g., revenue of companies) and due to procedures defined to guarantee privacy, protection and not disclosure of the data (Carta et al., 2019). On the contrary, note that studies using time series data (technical indicators) for stock direction prediction are abundant (Ampomah, Qin, & Nyame, 2020; Atsalakis & Valavanis, 2009; Basak et al., 2019; Carta et al., 2019; Moon et al., 2018; Nti et al., 2020; Zhou et al., 2019). In this case the prediction data is derived from past trading activities such as past stock prices and volumes. Consequently, this analysis does not focus on the intrinsic value of a stock, but rather on extrapolations from historical price patterns (Tsai et al., 2011). However, when the time series data is converted into input variables (e.g. relative strength index, price rate of change, Williams percentage rate, …) like in Basak et al. (2019), the classification task is very similar to a

classification using financial ratios. In this regard, classification algorithms based on technical indicators are included in Table 1 below.

2.3 Ensembles in stock price prediction

Table 1 provides an extensive overview of the available literature, categorized on the kind and number of ensembles used, as well as the performance metrics and the statistical tests used. All studies listed in this table elaborate on one or multiple classification ensembles to predict the stock price direction. On average, 3.5 ensembles are discussed in each paper. The number of ensembles is counted as per following example, a study with 4 homogeneous boosting methods using different single classifiers is counted as 1 ensemble.

Table 1 Ensembles in stock price prediction

Study	Ensembles				Performance metrics			Stat. tests
	Homo - geneous	Hetero - geneous	Stacking	#	AUC	ACC	Other	
Albanis & Batchelor (2007)	x	x		5		x		x
Ampomah, Qin, & Nyame (2020)	x			6	x	x	Prec, Sens, Spec, F	x
Ballings et al. (2015)	x			3	x			x
Basak et al. (2019)	x			2	x	x	Prec, Sens, Spec, BS, F	
Carta et al. (2019)	x			5		x	FinPM	
Chen et al. (2017)	x			1		x	Sens	
Jiang et al. (2020)	x	x	x	8	x	x	Prec, Sens, F	
Khaidem et al. (2016)	x			1	x	x	Prec, Sens, Spec	
Khan et al. (2020)	x			4	x		Prec, Sens, F	
Manish & Thnmozhi (2011)	x			1			Sens	
Mokoteli-Mokoteli et al. (2019)				5	x	x		
Moon et al. (2018)		x		1	x			
Nti et al. (2020)	x			2	x	x		
Ocak & Seker (2012)		x		1		x		
Patel et al. (2015)	x			1		x	F	
Qian & Rasheed (2007)		x	x	3		x		
Rodriguez & Rodriguez (2004)	x			4	x			
Tsai et al. (2011)	x	x		8		x	FinPM	
This study	**x**	**x**	**x**	**12**	**x**	**x**	**Prec, MCE, BS, H**	**x**

2.3.1 Ensembles

The ensembles are divided into two major categories: homogeneous and heterogeneous ensembles. In addition, a column ‘stacking’, which is a subcategory of the heterogeneous ensembles, is added to underline the absence of stacking frameworks in literature. While heterogeneous ensembles contribute to algorithm induced diversity, homogeneous ensembles add to the literature of data induced diversity. Three main data induced variations can be derived: bagging, boosting and random subspace. Firstly, bagging which generates multiple predictions formed by making bootstrap replicates of the learning set (Breiman, 1996). The most common way to aggregate these predictions is majority voting. Secondly, in contrast to bagging, boosting doesn’t train multiple predictions in parallel but applies a sequential method. A base learner is trained from an initial bootstrapped training cluster. Consequently, the distribution of the training samples is adjusted based on the performance of the base learner, meaning that the wrongly classified training samples get more attention in the future. Next, the second base learner is trained based on the adjusted sample distribution. This process repeats itself until several base learners are applied (Du et al., 2020). Finally, the predictions are usually combined using majority voting. Thirdly, random subspace generates multiple predictions by training on randomly sampled predictors with replacement.

While simple heterogeneous ensembles combine multiple different base learners, stacking adds a new dimension. Stacking, also called stacked generalization, combines the knowledge from a batch of base learners by implementing in a second step another single classifier with the first base learners’ prediction results as input (Jiang et al., 2020). This second step single classifier is also called the meta-learner. Meta-learners try to combine the predictions of the base-learners by learning their biases and correlations (Qian & Rasheed, 2007). In summary, bagging tries to reduce variance, boosting tries to reduce deviance and stacking tries to improve the overall prediction result (Du et al., 2020).

From Table 1 it is visible that only six studies implement heterogeneous ensembles and two discuss stacking. While literature states that heterogeneous classifier ensembles offer a slightly better performance (not significant) than the homogeneous ones (Tsai et al., 2011). In this study three different classifiers are combined using both bagging and majority voting. There are a lot of ways to combine different classifiers of which some are benchmarked in Albanis & Batchelor (2007). According to this study, by combining 5 classification techniques, the unanimity principle performs best as combining technique. This principle only classifies an instance as positive, if each individual

classifier does this separately as well. In addition, the absence of stacking frameworks in stock price direction is confirmed in Jiang et al. (2020). This study successfully implements ensemble learning algorithms and deep learning techniques into a stacking method in the prediction of stock price direction. The study shows that a stacking framework with machine learning techniques significantly improves the prediction performance. The proposed stacking framework is identified as top performer obtaining a higher level of accuracy, F-score and AUC value than several deep learning algorithms. A lasso based meta classifier was perfectly able to automatically weight and select the optimal base learners for the stock price direction problem.

2.3.2 Performance metrics and statistical tests

Table 1 also indicates that there is a large variety in performance metrics. The vast majority are directly or indirectly derived from four base measures. These base measures can be found in the four quadrants of the so-called confusion matrix. It is important to stress that the numbers in these four quadrants are measures and not metrics. There is some confusion about this terminology as explained in Sagiroglu et al. (2017). According to Table 1, Accuracy and Area Under the Curve are applied the most, respectively 10 and 13 times out of 18 papers. Unfortunately, due to the popularity of these metrics, a substantial amount of studies limits themselves to only one of these, including (Ballings et al., 2015). Focusing only on Area Under the Curve can lead to misleading results. According to Hand (2009) the AUC uses different misclassification cost distributions for different classifiers. For example, misclassifying the positive class is punished more severely by one classifier compared to another. To solve these problems multiple performance metrics can be used (Basak et al., 2019; Feng et al., 2018) or even combined (Caruana & Niculescu-Mizil, 2006; Jiang et al., 2020; Lessmann et al., 2015; T. Zhang & Chi, 2020) to score the ensemble methods. However, combining for example the AUC with the partial Gini coefficient could result in a less robust metric due to the apparent linear relationship between both (Schechtman & Schechtman, 2016). Prudence is required because this partial Gini index should not be confused with the Gini impurity measure for decision trees. Besides the standard performance evaluation metrics, such as accuracy or precision, some studies like (Tsai et al., 2011) test the decision making capacity of their model in monetary terms. In this way, they mimic a real investor by making a quarterly hold or sell decision based on the used machine learning model. The results in terms of ROI are than compared to a simple buy and hold strategy.

In the last column of Table 1 the use of statistical tests to rank the benchmarked algorithms is indicated. Different ranking methods are applied: chi-squared (Albanis & Batchelor, 2007), Friedman (Ballings et al., 2015) and Kendall's coefficient of concordance which is simply the normalization of Friedman (Ampomah, Qin, & Nyame, 2020). This in contrast to e.g. (Khan et al., 2020) where the ensembles are ranked based on accuracy without the use of any statistical test.

2.4 Gaps in literature

To investigate whether these issues are only related to stock price prediction, Table 5 is included in Appendix A. where the ensemble methods are summarized per financial domain (Financial distress, bankruptcy and credit scoring). Apart from the credit scoring domain, we can identify 4 major gaps in literature. Firstly, the number of articles benchmarking heterogenous ensembles in stock price prediction, is very limited. Secondly, these benchmarking studies don't include selective ensembles, in contrast to other financial domains like credit scoring (Feng et al., 2018; Lessmann et al., 2015). Thirdly, there is a gap concerning the limited number of different performance metrics. Different types of indicators are needed to reflect different notions of classifier performance (Lessmann et al., 2015). Only a few studies, e.g. (Basak et al., 2019; Feng et al., 2018) use a comprehensive set of evaluation metrics. Lastly, statistical tests to rank the benchmarking results are lacking in many studies. Taking these considerations into account, it is clear that 6 years after the publication of (Ballings et al., 2015) a lot has changed and an update is needed. Hence this study, (i) includes both homogeneous and heterogenous ensemble techniques, (ii) compares a much broader range of ensembles with some brand-new ensemble techniques from other financial domains, (iii) takes into account several different evaluation metrics, (iv) tests statistically the outcomes for each evaluation metric in a scientific way.

3. Methodology

3.1 Data

In this dissertation the same dataset is used as in Ballings et al. (2015). This dataset contains 5716 publicly listed European companies covering a wide range of different industries. Each observation consists of 80 financial indicators and a binary encoded variable which equals one for a stock price going up by at least 15% within the next year and which equals zero otherwise. To give a better insight in the imbalance of the dataset three possible cut-offs are depicted in Table 2.

Table 2 Class imbalance

	Negative class (0)	*Positive class (1)*
Threshold 15%	3394 (59%)	2322 (41%)
Threshold 25%	3865 (68%)	1851 (32%)
Threshold 35%	4252 (74%)	1464 (26%)

While the class imbalance of this dataset is tackled by oversampling the positive class in Ballings et al. (2015), this is skipped in this study because of four reasons. Firstly, metrics are used that decouple classifier performance from class skewness and error costs (Fawcett, 2006). Good examples include the AUC and H-measure. Secondly, by calibrating predictions prior to assessing them, it is possible to compare different classifiers based on the same ground. This calibration sanitizes a classifier's score distribution and prevents imbalance from indirectly affecting the BS or KS (Lessmann et al., 2015). Thirdly, the imbalance depicted in Table 2 is low for the 15% threshold, which is the threshold focus of this study. Lastly, the ability of different algorithms to be sensitive to class imbalance should not be discarded as it contributes to the real-life implementation.

3.2 Experimental set up

While the data is already preprocessed, scaling has not been done yet. The different features do not have the same range which could result in serious issues for algorithms like k-nearest neighbors, support vector machines or neural networks. For example, the k-nearest neighbors algorithm computes Euclidean distances which are difficult to compare if different features have different magnitudes. To transform the data, min max normalization and standardization are commonly used, however these two approaches are confused a lot in literature. To make the

distinction clear, both formulas are denoted below, where μ is the mean and σ the standard deviation.

Min max normalization: $x' = \frac{x-\mu}{x_{max}-x_{min}}$ Standardization: $x' = \frac{x-\mu}{\sigma}$

In this study standardization is only applied to the SVM, KNN and the ANN algorithm because these algorithms do not assume any distribution of the data and are affected by absolute values. This is also empirically tested on the train and validation set of the first fold, only these three algorithms resulted in a higher AUC.

To avoid data leakage, the data is split up into a training and a test set. This splitting percentage is not agreed upon in literature, e.g. 70/30 train test split is used in (Ampomah, Qin, Nyame, et al., 2020) and 80/20 train test split in (Rodriguez & Rodriguez, 2004). While using these fixed splits is simple and clear, it tends to be less robust in comparison to cross validation. Picking a different split could result in a different performance, to cancel this effect, cross validation executes the train test split multiple times. In this study, taking the recommendation of Dietterich (1998) into account, a 5x2 cross validation will be computed to minimize the influence of the variability of the training set. While this method produces train and test sets, a validation set is still needed to tune the hyperparameters of the classification algorithms. In Lessmann et al. (2015) a second 5x2 cross validation is computed to obtain the validation set. However, the gain in robustness does not necessarily outweigh the extra computation cost. Consequently, in this study an 60/40 split is computed on each training fold.
While splitting the dataset into multiple folds, class imbalance could become worse. In the worst-case scenario, the test set may not contain any instance of the minority class at all (Raschka, 2018). This can be prevented by stratification, according to Kohavi (1995) this is generally a better set up, both in terms of bias and variance, compared to regular cross-validation. Following this recommendation, the 5x2 cross validation is stratified by maintaining the original class proportion in the resulting folds. Accordingly, the stratification is also done for the inner train and validation split. This was implemented with the `rsample` r-package by Silge et al. (2021).

For some designs the combination of a large hyperparameter grid and a 5x2 cross validation results in substantial computation times. For example, for algorithms like support vector machines, four parameters are tuned which increases the grid size exponentially. To make this tuning more

effective, this study will implement a random search for each multi-dimensional hyperparameter tuning. A lot of studies implement a grid search to tune the hyperparameters, however this is not the optimal way. Bergstra & Bengio (2012) show empirically as well as theoretically that randomly chosen trials are more efficient for hyper-parameter optimization in comparison to trials on a grid. To clarify this further, two hyperparameters used to tune a rotational forest in this study are depicted below in a grid search and compared with a random search.

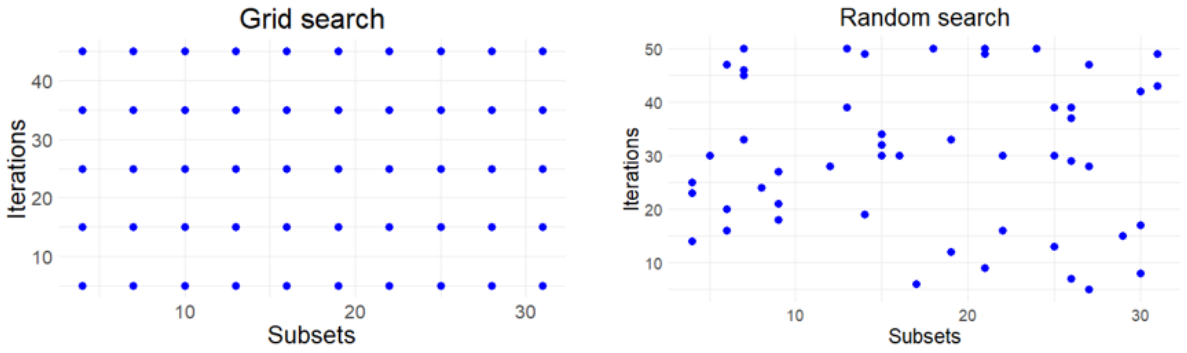

Figure 2 Comparison search procedure rotational forest

If we would project all possible outcomes of one parameter on its axis, random grid search would clearly investigate a broader set of values. This favors the random search because for most data sets, only a few data specific hyper-parameters really matter (Bergstra & Bengio, 2012). While these important hyper-parameters are tuned more in depth, the others receive less attention.

To increase computation speed further, the future_map function from the `furrr` r-package (Vaughan & Dancho, 2021) is used as much as possible. This iterative function makes use of parallel processing through all cores of the processor which leads to a substantial increase in speed. All supervised learning algorithms are computed using R (version 4.0.4). These computations were executed locally on an Intel Core i7-8565U 1.80Ghz CPU.

3.3 Classification algorithms

To clarify this section, Table 6 has been added in Appendix B.. This table summarizes for all 18 classification algorithms the classifier family, abbreviation and the set of hyperparameters used.

3.3.1 Single classifiers

While single classifiers are not directly the focus of this benchmark study, they form the basis of ensembles. All four single classifiers trained in Ballings et al. (2015) are implemented and will be very briefly discussed.

Firstly, a **logistic regression** is trained using a lasso regularization to avoid overfitting. The `glmnet` r-package (Friedman et al., 2010) was used, setting the elastic net mixing parameter to one indicating a lasso regularization which is the default setting. In addition the lambda parameter is tuned based on the values used in (Moon et al., 2018).

Secondly, a **k-nearest neighbor** model is programmed, deploying the `FNN` r-package (Beygelzimer et al., 2019). The most important parameter K indicates the number of closest neighbors which is tuned for a broad range of values. Important to mention is that all features have been standardized before entering the machine learning model.

Thirdly, a **support vector machines** model is created. This model was developed first by Vapnik (1999). The main idea is that the points closest to the separating hyperplane, called the support vectors, are more important than the others and receive a non-zero weight in the algorithm (Weng et al., 2018). For this algorithm, four hyperparameters: a cost (regularization term), gamma (not for linear kernels), kernel and degree can be tuned. The regularization, or cost parameter can be set, to account for misclassifications resulting from the trade-of between the separating margin and the misclassification of the classes. For example, for a lower cost, a larger margin will be accepted, reducing the training accuracy, and preventing overfitting. Both the cost and gamma hyperparameter are tuned based on Caruana & Niculescu-Mizil (2006). The kernel function can also be further specialized, each of these kernels differ in the way distinctions between classes are made (Mokoteli-Mokoteli et al., 2019). Actually, any function that satisfies Mercer's condition can be used as kernel function (Huang et al., 2005). The most frequently implemented kernels according to Ballings et al. (2015): linear, radial, polynomial and sigmoid are used in this study. The linear kernel only requires variations in the regularization parameter, the gamma parameter is set to $\frac{1}{data\ dimention}$ by default. On the other hand, the polynomial kernel needs an additional

parameter, namely a degree which was tuned together with the cost, gamma and kernel function in a random search. The SVM function from the `e1071` r-package (Hornik et al., 2021) was used to program this algorithm. Note that the SVM algorithm doesn't produce probability predictions.

Lastly a **neural network** was trained. This algorithm is in essence a linear combination of explanatory variables adjusted to a transfer function (Rodriguez & Rodriguez, 2004). Before computing the algorithm, all features are normalized. In this paper a feedforward artificial neural network is used with four layers: an input layer with the 80 numeric input features, a first hidden layer, a second hidden layer, and an output layer with two output nodes (one for each class). These are generally the most common neural nets in use (Dongare et al., 2012). Both hidden layers use a relu (rectified linear function) activation function and the output layer uses a sigmoid function, ideal for the classification task. The model is compiled using the binary cross-entropy loss function, which is known to work well for a binary classification, and the Adam optimizer.

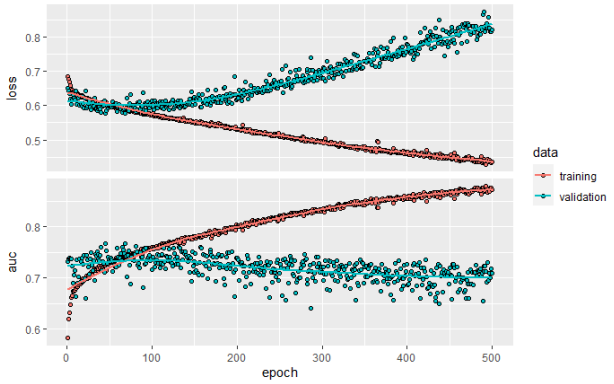

Figure 3 NN on first fold no regularization

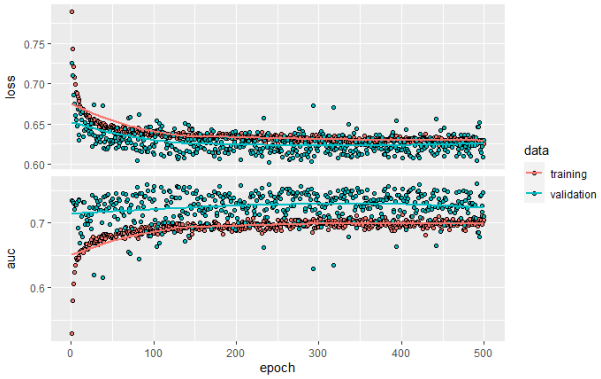

Figure 4 NN on second fold with regularization and dropout

In Figure 3, a neural net is trained, without regularization on the first fold of the 5x2 cross validation set. It is clearly visible that after 100 epochs the training data keeps on increasing in performance, however the validation set follows a contradictory pattern. This indicates that the model has become too specific and will perform bad on new, unseen data samples.

Just like in almost every machine learning algorithm, a regularization term is needed to prevent overfitting. In this study a L2 regularization parameter is added to combat overfitting. In addition, to further combat overfitting, dropout is implemented. The main idea is to randomly drop units from a neural network during training. This prevents the units from co-adapting too much (Strivastava, 2013). According to Strivastava (2013) it can be observed that the performance is insensitive for dropout percentages p if $0.4 \leq p \leq 0.8$. While the dataset is relatively small for deep learning, this

study will take a 0.4 dropout rate. Figure 4 shows that due to the dropout technique and L2 regularization, the validation is performing better than the training data. According to the minimum of the loss curve in Figure 4, 100 epochs are allowed with a batch size of 64 instances. To find the best model hyperparameters the nodes in the two hidden layers and the regularization lambda are tuned. Conceptually, nodes in successively higher layers abstract successively higher-level features from preceding layers (Dongare et al., 2012). Accordingly, the tuned number of nodes in the second layer are always set smaller than the number of nodes in the first hidden layer. To program the NN, the deep learning library of Keras is used, which runs on top of Tensorflow.

In addition, two extra single classifiers are added. A **decision tree**, which classifies an instance by filtering it down a tree from the root node to a leaf node (Qian & Rasheed, 2007). The tree is determined by: a function to measure the quality of the split, the stopping criterium, a method to assign a class or probability distribution at the leaf nodes and a posterior pruning process to simplify the tree structure (Abellán & Mantas, 2014). Reciprocally this study implements the Gini coefficient as splitting criterium, a maximum tree dept of 30 (default), a probability distribution from the leaf nodes and a complexity parameter which is tuned based on the different complexity parameters used in Kattan et al. (2012). The decision tree is implemented by making use of the `rpart` r-package (Therneau et al., 2019).
The last single classifier is a **naïve bayes**, which makes the strong assumption that features are independent given the class. While this is a strong assumption, naïve bayes often competes well with more sophisticated classifiers (Rish, 2001). The naïve bayes classifier is implemented with the `e1071` r-package of Hornik et al. (2021). The Laplacian smoothing parameter is tuned empirically where the null value indicates that no smoothing is applied.

3.3.2 Homogeneous ensembles

3.3.2.1 **Bagging.** Bagging, also called bootstrap aggregation is the first and most basic ensemble method of this study. By making bootstrap replicates, groups of observations are selected on which base classifiers are trained in parallel. This process makes the training dataset of every base classifier by default independent of the others (Ampomah, Qin, & Nyame, 2020). The outputs of each base classifier are combined by majority voting. For instable prediction methods, this procedure can give substantial gains in accuracy (Breiman, 1996). An example of such prediction method is the decision tree which builds very different models when applied to different training

sets (Abellán & Mantas, 2014). Consequently, in this study, a homogeneous bagging model, with decision trees as base classifiers, is trained. The training process is computed with the `ranger` r-package (Wright et al., 2020). This package is a very fast implementation of a random forest, however by setting the try-parameter (number of features selected) to 80 (total amount of features) the model boils down to bagging. As recommended by Breiman (2001) a large number of trees (500) is used in our model.

3.3.2.2 **Random forest.** Clearly random forest is very similar to bagging based on decision trees. However, to increase randomness, "the random subspace" method is applied, which does a random selection of a subset of all features to grow each tree (Breiman, 2001). Again, the `ranger` r-package is used, now tuned for the mtry parameter. The standard number of features used to obtain good results is the half of the total number of features (40) (Abellán & Castellano, 2017), however according to Ballings et al. (2015) this mtry parameter should be set to the square root of the number of features (9). So, to be certain, the selected number of features is tuned to 40, 10 and some additional values.

3.3.2.3 **Rotational forest.** Rotational forest proposed by Rodríguez et al. (2006) further increases diversity among the base classifiers by combining bagging with feature extraction. The features are divided into K non-overlapping subsets of equal size. A principal component analysis (PCA) is applied to each K subset of features (Abellán & Castellano, 2017). All principal components are retained to preserve the variability information in the data. Thus, K axis rotations take place to form the new features for a base classifier (Rodríguez et al., 2006). Two main parameters will be tuned: K and the number of base classifiers or bagging iterations (L). By using the `rotationForest` r-package of Rodríguez et al. (2006) K and L are tuned based on du Jardin (2019) and Lessmann et al. (2015).

3.3.2.4 **Extremely randomized trees.** Extremely randomized trees by Geurts et al. (2006) still implements random feature selection, however it differs from a random forest in two ways. The observations to build a tree are not resampled (bootstrapped) and a random split is used. Instead of seeking for the most discriminative thresholds, random cut-points are chosen and thus regarded as the splitting rules for the decision trees. This kind of algorithm usually allows to decrease the model's variance a bit more, at the cost of a slight increase in bias (Jiang et al., 2020). Besides the reduction in variance, a main advantage of the resulting algorithm is computational efficiency (Geurts et al., 2006). The selected number of features is tuned for the same values as random

forest and the number of random cuts for each predictor is varied. The ERT algorithm is implemented by the `extraTrees` r-package (Simm & de Abril, 2015).

Boosting, in contrast to bagging, trains the models in a sequential way. Accordingly, each model run, dictates what features the next model focuses on, by using weights. Just like bagging, boosted trees decrease the variance of the single estimate as they combine several estimates from different models (Mokoteli-Mokoteli et al., 2019).

3.3.2.5 **Stochastic adaboost.** The first boosting algorithm considered is Stochastic adaboost. In this sequential algorithm, the weights of the training samples, which are correctly classified by the current classifier will decrease, while the weights of the samples which are misclassified will increase (X. Zhang et al., 2016). Subsequent base learners are tweaked in favor of those instances misclassified by preceding classifiers (Ampomah, Qin, Nyame, et al., 2020). After training, the final output is constructed by a weighted vote of the base classifiers. A classifier with a small error will receive a larger weight in this voting process. This boosting method is implemented by the `ada` r-package developed by Culp et al. (2016). To prevent overfitting, the base classifier trees are constructed with a maximum of 8 nodes and depth of 3 following Friedman (2001). Tuning is done on the amount of iterations.

3.3.2.6 **Extreme gradient boosting.** Extreme gradient boosting is another tree-based ensemble which has been winning in several data contests (Jiang et al., 2020). This algorithm is based on a gradient boosting technique introduced by Friedman (1999). While Adaboost reweights each observation after each boosting step, gradient boosting learns to predict the error. This is done by trying to better classify the residuals, or the misclassified samples of the previous iteration in the next iteration (Basak et al., 2019). In comparison to gradient boosting, extreme gradient boosting applies the second order Taylor expression for the objective function. Therefore, it can process both the first and second order derivatives in parallel to accelerate the convergence process while training. Furthermore, a regularization term, which can smooth the contributions of each decision tree, is added to prevent overfitting (Jiang et al., 2020). The objective function which XGB tries to minimize is given below.

$$obj(\theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{k} \gamma T + \frac{1}{2}\lambda \|w\|^2$$

In this equation, the first summation contains the loss function and the second summation the complexity to prevent overfitting, with γ the complexity parameter, T the number of leaf nodes, λ a fixed coefficient and $\|w\|^2$ the ℓ2-norm of leaf weight. To train the model the `XGBoost` r-package of He et al. (2021) is used. The XGB algorithm contains numerous tuning parameters which require careful selection. By default, the learning rate is defined automatically based on the dataset properties and the number of iterations (He et al., 2021). This automatically defined value should be close to the optimal one, so we do not tune this parameter explicitly. The maximum tree depth is set to 7 to limit tree complexity based on du Jardin (2019). The number of iterations and the subsample hyperparameter are tuned. If the subsample parameter equals to 0.5, half of the data is used to grow trees, which improves speed of training and generalization. However, a higher number of boosting iterations increases computation time and is more prone to overfitting, consequently a tradeoff must be made.

3.3.2.7 **Light gradient boosting.** In addition, a light gradient boosting model is trained, developed by a team from Microsoft to reduce the implementation time of XGB (Y. Zhang & Haghani, 2015). LGB supports efficient parallel training just like XGB however, the main difference comes from the way in which the three is grown. As depicted in Figure 5, an XGB grows trees level wise, while Light gradient boosting forms decision trees leaf wise (Jiang et al., 2020).

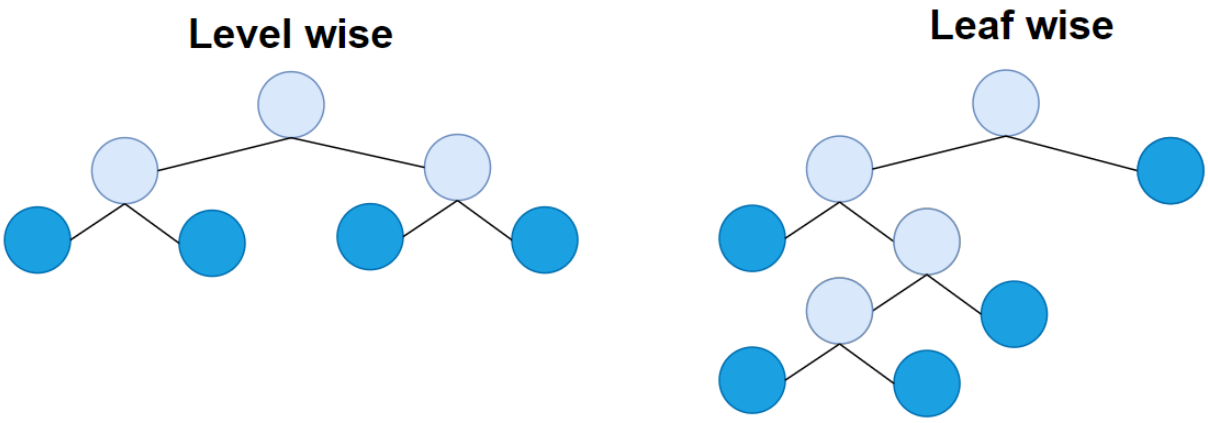

Figure 5 Decision tree splitting process

By growing leaf wise, the algorithm can reduce loss faster compared to a level wise structure. Note that because of this structure, extra carefulness is required in terms of overfitting. Following the reasoning of XGB, the same tuning is applied. According to Jiang et al. (2020) the algorithm could also be tuned for the regularization parameter alpha and lambda however, the fixed maximum tree depth, subsampling and the automated learning rate should be enough to prevent overfitting.

3.3.2.8 **Catboost.** The last and most recent developed homogeneous ensemble of this study is catboost. This open-sourced algorithm was developed by Dorogush et al. (2018) in 2018. Catboost stands for categorical boosting indicating that it smartly transforms categorical features to numeric. While our dataset does not contain categorical values, it is still an interesting technique for our benchmark because the algorithm uses a different schema compared to LGB and XGB for calculating leaf values when selecting the tee structure. In addition Dorogush et al. (2018) argue that catboost outperforms the existing state-of-the-art implementations of gradient boosted decision trees. Catboost grows a level-wise balanced tree. The feature-split pair that brings the lowest loss for each level is selected. This split is then used for all the nodes on the same level which is a new approach called minimum variance sampling. This approach decreases both the number of samples needed for each iteration of boosting and increases the model quality significantly (Ibragimov & Gusev, 2019). Just like XGB and LGB the max depth is fixed to 7 to prevent overfitting and the learning rate, sample and rounds are tuned. The algorithm is implemented with the `catboost` r-package (Dorogush et al., 2018).

3.3.3 Heterogeneous ensembles

The heterogeneous ensembles discussed in this section make use of all previously discussed classifiers of which the predictions were stored. These different base classifiers have different views on the data which has been proven to help develop more reliable, robust and generalized classifier models (Sesmero et al., 2015). For these algorithms no additional r-packages were used because they have too little customization.

3.3.3.1 **Simple average.** The first heterogeneous ensemble technique averages the results of multiple base classifiers. In this study we trained and tuned 6 single classifiers and 8 homogeneous classifiers. For all 6 performance metrics, 10 folds, 2 validation splits and 14 classifier the best predictions were stored resulting in 1680 lists of predictions. This first heterogeneous ensemble is constructed by simply averaging over the predictions of all 14 base classifiers.

3.3.3.2 **Weighted average**. This technique is very similar to simple average only the different predictions from the base classifiers are combined whilst using a weighting factor. The weights (W_i) are calculated based on the performance of the base models on the validation data:

$$W_i = \frac{X(C_i, D_v)}{\sum_{i=1}^{I} X(C_i, D_v)}$$

Where $X(C_i, D_v)$ denotes the performance metric of classifier (C_i) on the validation set (D_v). According to the setup this weighting is done for each separate performance metric. When these weights are computed they can be used to combine the predictions on the test data.

3.3.3.3 **Stacking**. Stacking, short for stacked generalization, is probably one of the most frequently discussed heterogeneous ensembles. This machine learning method has not one but two layers, where the first layer consists of the binary predictions of all base models that enter the ensemble. The second layer conducts a weighting by training a second classifier (the meta classifier) with the binary predictions as input. Naively applying a second-stage model to base learners that are trained on the full training set is dangerous because it rewards overfit learners (Sharabiani, 2016). Therefore, the best training predictions stored for each classifier are used to make binary predictions. These predictions are combined and used as input to train a meta classifier. This meta classifier should be robust to multicollinearity because each classifier is predicting the same phenomenon. Accordingly, in this study, a lasso logarithmic regression is used as meta classifier with L2 regularization to handle multicollinearity. The regularization parameter is tuned on a 40/60 train/validation split of the binary inputs. Next, the stored test predictions are set to binary. By applying the trained and tuned logarithmic classifier on these binary predictions, the final probabilities for each stock are computed.

3.3.3.5 **Hill-climbing ensemble selection**. The last heterogeneous ensemble of this study is a static direct model. Just like the previously mentioned heterogeneous ensembles, a selection procedure is applied to get a good combination of base learners. To get the optimal one, every combination should be tested on a validation set. This quickly becomes iteratively heavy, even for 14 classifiers. To tackle this, hill climbing (or forward selection) is applied. In the initialization phase, the best performing model on the validation set enters the ensemble subset. Next, this ensemble subset is updated by iteratively looking for the best second base classifier to add. This is repeated until the performance does not increase anymore. Note that after adding a classifier, it can still be picked a second time. Therefore, the ensemble prediction can effectively be either a simple or weighted average depending on whether all selected base models are unique (Caruana et al., 2014). When the subset is selected, it can be applied on the test predictions, resulting in the final performance. Note that the ensemble subset obtained could be a local instead of a global optimum.

3.4 Model evaluation criteria

To start this section, it is important to mention is that some metrics focus on type 1 errors while others on type 2 errors. These errors are derived from the confusion matrix depicted in Table 3. Type 1 errors are located in the upper right quadrant, indicating a negative sample wrongly classified as positive. In the opposite quadrant, in the lower left, type 2 errors indicate a positive sample wrongly classified as negative.

		True class	
		Positive	Negative
Predicted class	Positive	TP	FP
	Negative	FN	TN

Table 3 Confusion matrix

The question arises which error is worst in the stock classification domain. This depends on your strategy: if you buy only the stocks that are classified as 1 (rise) than false positives (type 1) could really hurt your profit. False negatives (type 2) are an opportunity loss, they do not directly result into costs for the investor. Another strategy could be that the investor shorts the stocks classified as 0. However, this study implements a 15% rise cut off, so the stocks classified as 0 still contain rising stocks which would hurt the investor's portfolio. This study assumes the prediction results will be used to buy the stocks classified as increasing more than 15%. Following this reasoning a type 1 error is considered as more severe which we will consider while selecting the evaluation criteria.

In contrast to Ballings et al. (2015) this study evaluates the classification algorithms with multiple performance metrics. This study will take six performance metrics into account: AUC, H, BS, MCE, ACC and Prec. While the AUC is a very good cut-off independent metric, it has some limitations, as described in the literature review. Every metric has some advantages and disadvantages, consequently combining a set of metrics, thus evaluating different aspects of performance is done (Jiang et al., 2020; Lessmann et al., 2015; T. Zhang & Chi, 2020) or proposed (Caruana & Niculescu-Mizil, 2006). The deployed metrics mix contains two metrics of each of the three classes

defined in Lessmann et al. (2015): (i) assessing the accuracy of the scorecard (e.g. Area under the curve), (ii) assessing the accuracy of the scorecard's predictions (e.g. Brier score), (iii) assessing the correctness of the scorecard's categorical prediction (e.g. Accuracy). In (Caruana & Niculescu-Mizil, 2006) the same groups are identified which are respectively called: (i) ordering/rank metrics, (ii) probability metrics and (iii) threshold metrics.

3.4.1 Rank metrics

From the first group the AUC and H-measure are selected. The extensively applied AUC measures the probability that a randomly chosen positive case receives a higher score than a randomly chosen negative one. Considering the limitations of the AUC, the H-measure is reported as well, which overcomes the problem of different misclassification costs for different classifiers (Hand, 2009). A severity ratio controls the severity effect of misclassifying a class 0 instance in comparison to misclassifying a class 1 instance. Per example a severity ratio of 2 implies that false positives cost twice as much as false negatives. This ratio seems adequate for this study considering a type 1 error is considered as more severe. In addition, (C. Anagnostopoulos et al., 2012) argues that it would be strange to treat both classes symmetrically in case of unbalance. Accordingly, the misclassifications of the smaller class are considered more serious. The AUC is computed by the `AUC` r-package of (Ballings & Van den Poel, 2013) and the H-measure is calculated by using the `hmeasure` r-package of Christoforos Anagnostopoulos (2019). The calculation on which both packages are based are depicted below.

$$\text{Area under the curve} = \int_0^1 \frac{TP}{(TP+FN)}\, d\, \frac{FP}{(FP+TN)} = \int_0^1 \frac{TP}{P} d \frac{FP}{N}$$

The parameters used for the AUC calculations are all derived from a confusion matrix which is depicted in Table 3.

$$T_c = \underset{t}{\operatorname{argmin}}\, 2(c * \pi_0 * \frac{FP}{N}(t) + (1-c) * \pi_1 (1 - \frac{TP}{P}(t))$$

$$L_w = \int_c L(c; T_c) * w(c) dc$$

$$\text{H-measure} = 1 - \frac{L_w}{L_w^{max}}$$

To calculate the H-measure three crucial equations are needed. The second part of the first equation denotes the total cost. Where c is chosen based on the severity ratio $\frac{c}{1-c}$. In this study the SR is set to two to penalize type 2 errors more severely. The class priors π_0 , π_1 are respectively equal to $\frac{TN+FP}{n}$, $\frac{TP+FN}{n}$. In the second equation the averaged minimum cost-weighted loss is given. In this equation $L(c;T)$ is the minimum weighted loss, where the right threshold is chosen by minimizing the total cost $L(c;t)$ for each value of c. In the last step, the third equation is normalized with the maximum value L_w can take. The result is subtracted from 1 so a higher H-measure indicates better performance. For a more detailed description Anagnostopoulos et al. (2012) is referred to.

3.4.2 Probability metrics

From the second group the BS is selected which is the mean squared error between the probabilities and the zero/one responses. The exact calculation is as follows:

$$\text{Brier score} = \frac{1}{N} \sum_{t=1}^{N} (y_i - p(y_i))^2$$

Where N is the number of observations, $p(y_i)$ the forecasted probabilities and y_i the actual outcomes. In addition, the mean cross entropy is selected also called the log loss. This log loss function is used in several classifiers and used in the probabilistic setting when interested in predicting the probability that an example is positive (Caruana & Niculescu-Mizil, 2004). Just like the brier score, this performance metric needs the classifier to produce probabilities for all the samples. According to the formula below the mean cross entropy is calculated. The parameters are the same as used in the Brier score calculation.

$$\text{Mean Cross Entropy} = -\frac{1}{N} \sum y_i * \ln(p(y_i)) + (1 - y_i) * \ln(1 - p(y_i))$$

If we take for example one observation for which y_i = 1 and $p(y_i)$ = 0.9 than the MCE would result in 0.1 which is very good. The smaller both Brier score and Mean Cross Entropy the better the prediction.

3.4.3 Threshold metrics

Lastly, the accuracy and precision are selected from the threshold metrics. Threshold metrics make use of a fixed threshold which is usually 0.5, as followed in this study. Consequently, these metrics don't consider closeness to the threshold, only whether it is below or above a threshold. Optimizing the accuracy considers both type 1 and 2 errors equally, while precision specifically checks the classifiers ability to identify stocks that increase 15% by minimizing type 1 errors. Mathematically the accuracy and precision metrics are calculated as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

3.5 Calibration

For metrics who require probability predictions like the brier score and mean cross entropy a problem arises. Not all machine learning algorithms predict probabilities that are probabilistically meaningful (= calibrated probabilities). Probabilistically meaningful means that a probability of e.g. 0.4 indicates that in the long run 40% of the events will occur. For example, uncalibrated probabilities are produced by maximum margin methods like SVM and boosted trees which tend to push predicted probabilities away from 0 and 1 forming sigmoid shaped curves. On the other hand, naïve bayes tend to do the opposite and push the predicted probabilities to 0 and 1, forming uncalibrated probabilities. This phenomena is caused by the unrealistic independence assumption (Caruana & Niculescu-Mizil, 2006). The calibration or reliability diagrams in Figure 6 divide the predicted probabilities into 10 bins, depicted on the x-axis, the y-axis shows the fraction of samples that are classified as one. Below each calibration diagram the amount of observations per bin are shown. As some bins contain a very small amount of observations, it is hard to interpret the reliability diagram for these. The diagrams are based on the first fold of the stock dataset and clearly visualize the deviation of the naïve bayes probabilities to 0 and 1. If the probabilities are calibrated, they should closely follow the dashed line through the origin, which is illustrated by the logarithmic regression probabilities. As expected, the support vector machines calibration plot shows a sigmoid

shaped curve, however it is surprisingly well calibrated. This is because the used `svm` r-package already implements calibration by default when returning probabilities.

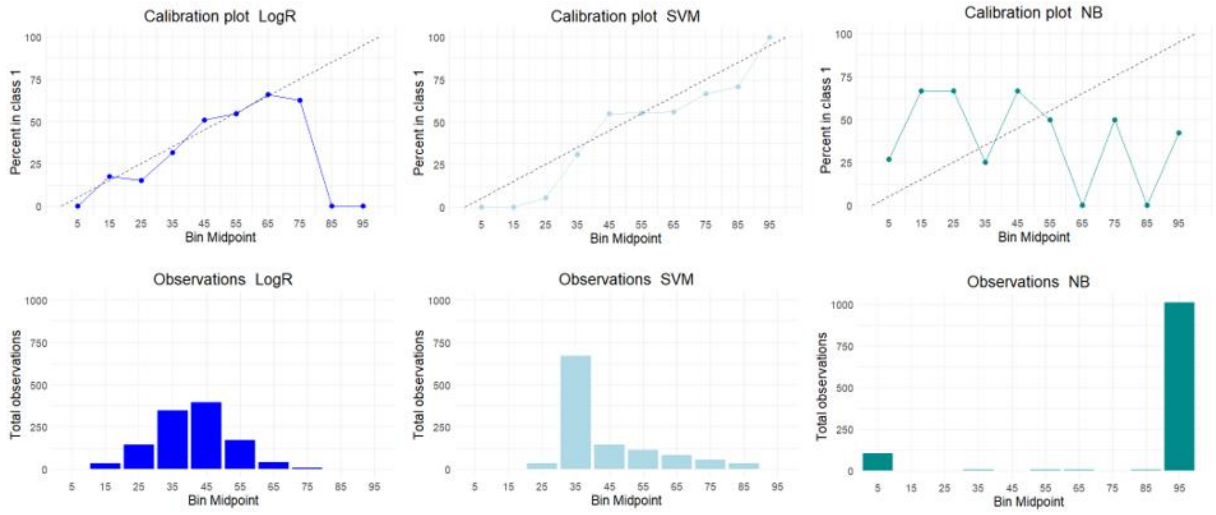

Figure 6 Reliability and observation plot of LogR, SVM and NB before Platt scaling

To tackle calibration issues Platt scaling is applied to all classifiers (Platt, 1999). While being fully aware that some classifiers like a logarithmic regression are calibrated by default, we want to treat every classifier in the same way. In essence plat scaling runs a logistic regression model on the output of a validation dataset. This trained logistic regression is afterwards used to calibrate the test probabilities. While the logistic regression implies linearity, someone could argue that using a nonlinear calibration technique like Isotonic regression would perform better. However, according to Bequé et al., (2017) Platt scaling consistently outperforms Isotonic regression on data set sizes similar the stock data under consideration. After applying the method of Platt the calibration plots improve as depicted in Figure 7 below.

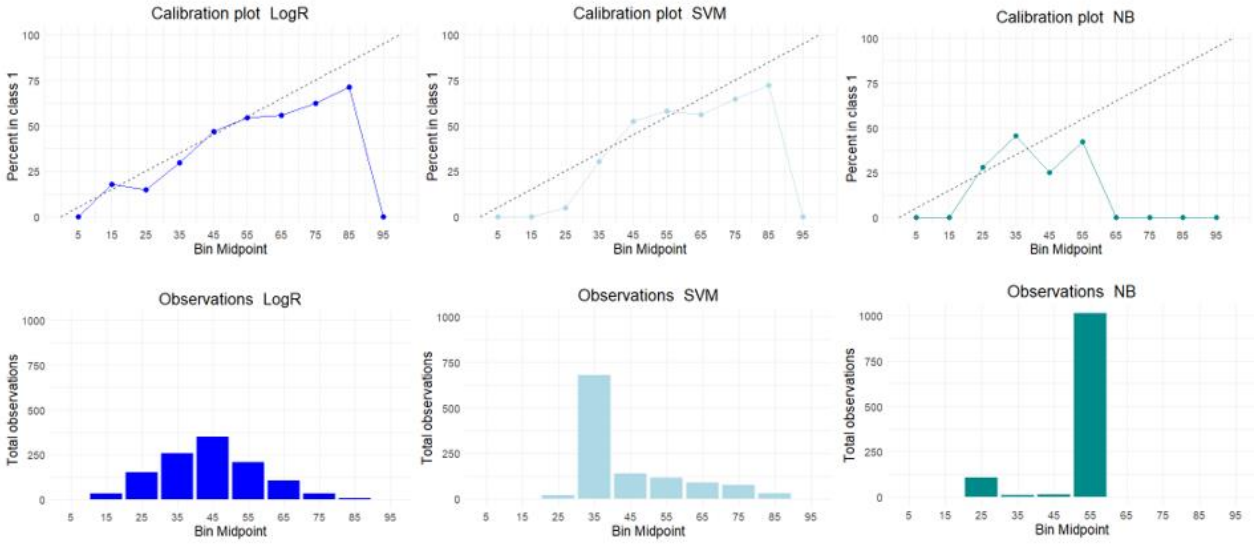

Figure 7 Reliability and observation plot of LogR, SVM and NB after Platt scaling

Lastly, the question arises how the other performance metrics under consideration are influenced by this calibration method. Because of the monotonic nature of the sigmoid function, the rank of the calibrated probabilities is preserved. Therefore, the two rank measures, AUC and H-metric should almost not change due to calibration. To check this statement and to see to which extent the other metrics changed a sensitivity analysis is conducted in Appendix C.. In this sensitivity analysis the results with calibration are compared to the results before calibration.

3.6 Algorithm comparison

In this subsection all 18 different classifiers are compared by making use of six performance metrics. Firstly, we rank each classifier for each fold and each performance metric. The best performing classification algorithm of the first fold gets a rank of 1, the second-best performing rank 2, etc. Afterwards, the ranks are averaged across all 10 folds of the 5x2 cross validation. Another approach could have been to first average the performance metric over all folds and then rank the classifiers. However, this results in slightly less accurate rankings. Next, the rankings are averaged over all performance metrics resulting in a final average rank. Because the classifiers are ranked for each performance metric separately, there is no need to scale the different performance metrics as explained in Caruana & Niculescu-Mizil (2006).

The average rankings over 10 folds per performance measure will be the bases of the non-parametric Friedman test. The Friedman test statistic for each performance metric is calculated according to this formula:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left(\sum_{j=1}^{k} AvgR_j^2 - \frac{k(k+1)^2}{4} \right)$$

Where N is the number of folds (10) and k the number of classifiers (18). The average ranks computed over the 10 folds for each classifier j are denoted by AvgR_j^2 . The test statistic has as null hypothesis that all performance between classifiers is equal. If the calculated chi squared is greater than a critical value (27.59) derived from table A4 in Sheskin (2003) with alpha (0.05) and the degrees of freedom (k-1) the null hypothesis is rejected. In the last row of Table 4 each calculated chi squared Friedman test is clearly much larger than the critical value indicating that the classifiers are statistically different for each metric.

Consequently, a post hoc test can be executed to know which classifiers are different from others. The null hypothesis states that a classifier is performing equal to the control classifier. As control classifier the best performing classifier is chosen for each performance metric. The z values are calculated following:

$$z = \frac{AvgR_i - AvgR_j}{\sqrt{\frac{k(k+1)}{6n}}}$$

Where $AvgR_j$ is the average rank over all k folds of the control classifier and $AvgR_i$ the average rank of the compared classifier. To calculate the corresponding p value for each z value the `pnorm` r-function is applied. In addition, following the recommendation of García et al. (2010) the p values are adjusted taking into account that multiple tests are conducted which increases the chances on a type one error (data dredging). According to a post hoc benchmark study of García et al. (2010) the Hommel and Rom post hoc tests are identified as most powerful. Both are sequential methods implying several correction steps depending on the results of prior steps. This study implements the Hommel procedure, ordering the p-values from smallest to largest (step-up approach) and correct for the previous number of tests in each step.

4. Discussion of results

Table 4 visualizes the final result after running the program for all classifiers and performance metrics. The code to reproduce these results can be consulted via the Github link in Appendix D. In the columns, each metric is depicted, which is an average over 10 folds, the rows depict all classifiers. The order of the rows corresponds to the buildup of this dissertation, with firstly the single classifiers, secondly the homogeneous classifiers and lastly the heterogeneous classifiers. For each classifier the adjusted p values are given between brackets, comparing the classifier with the best ranked classifier of the column indicated by empty brackets. Note that for the precision metric, this best ranked classifier is different from the absolute best classifier. The adjusted p value is underlined when significant, taking a 5% significance level into account. All these adjusted p values are based on rankings as explained in section 3.6. To give a rough indication of classifier performance the last column is added, which calculates the average across all columns and attributes the best average rank to one, the second-best to two until all classifiers are ranked.

Table 4 Final result with 5x2 CV, tuning and calibration

	AUC		H		BS		MCE		Prec		Acc		Rank
LogR	.650	(.000)	.085	(.000)	.226	(.000)	.642	(.000)	.563	(.000)	.622	(.000)	16
KNN	.678	(.000)	.113	(.000)	.218	(.000)	.633	(.000)	.592	(.002)	.639	(.001)	13
SVM	.674	(.000)	.111	(.000)	.221	(.000)	.632	(.000)	.457	(.000)	.641	(.002)	14
NN	.678	(.000)	.111	(.000)	.219	(.000)	.628	(.000)	.375	(.000)	.629	(.000)	15
DT	.641	(.000)	.077	(.000)	.226	(.000)	.644	(.000)	.558	(.000)	.630	(.000)	17
NB	.600	(.000)	.063	(.000)	.242	(.000)	.677	(.000)	.084	(.000)	.583	(.000)	18
RF	**.728**	**()**	.174	(.503)	**.205**	**()**	**.595**	**()**	.626	(.064)	.670	(.834)	2
BagTree	.725	(.675)	.173	(.503)	**.205**	(.671)	**.595**	(.706)	.632	(.196)	**.671**	**()**	3
RotF	.700	(.002)	.141	(.001)	.214	(.000)	.616	(.001)	.612	(.031)	.662	(.834)	11
ERT	.722	(.571)	.168	(.297)	.207	(.503)	.599	(.706)	.622	(.035)	.665	(.819)	6
Ada	.723	(.675)	.168	(.215)	.208	(.256)	.605	(.181)	.631	(.122)	.672	(.834)	5
XGB	.717	(.281)	.167	(.358)	.207	(.388)	.603	(.297)	.612	(.031)	.662	(.333)	9
LGB	.713	(.040)	.156	(.020)	.210	(.027)	.606	(.106)	.618	(.037)	.663	(628)	10
CB	.719	(.420)	.166	(.108)	.208	(.261)	.600	(.673)	.610	(.018)	.666	(.834)	8
SA	.724	(.601)	.172	(.476)	.212	(.003)	.613	(.003)	.676	**()**	.667	(.834)	7
WA	.725	(.675)	**.181**	**()**	.211	(.018)	.610	(.040)	.669	(.738)	.668	(.834)	4
Stack	.683	(.000)	.132	(.000)	.219	(.000)	.630	(.000)	.659	(.449)	.669	(.834)	12
HCE	**.728**	(.675)	.174	(.503)	**.205**	(.675)	**.595**	(.706)	**.682**	(.590)	**.671**	(.834)	**1**
χ^2_{17}	149.383		148.421		156.653		156.653		113.656		125.467		

From this table several conclusions emerge. Firstly, the HCE selection procedure often gets stuck in the initialization phase by just selecting the best classifier (mostly RF). Only for the Precision metric, multiple classifiers are selected, improving performance. Because the initialization procedure starts by selecting the best classifier and updates the selection only when performance is improved, HCE will always be superior or equal in performance to the selection subset. For the H-measure, we notice that a higher performance is found for the Weighted Average. Therefore, it becomes apparent that the HCE selection not always finds a global optimum but can get easily stuck in a local optimum.

Secondly, based on the adjusted p values of the single classifiers, there is strong evidence against the hypothesis that single classifiers perform equal to the best ensemble method in this benchmark. Every pairwise comparison with a single classifier results in an adjusted p value, far below the five percent significance level. This finding is in line with previous studies (Ampomah, Qin, Nyame, et al., 2020; Ballings et al., 2015; Basak et al., 2019; Tsai et al., 2011).

Thirdly, overall, the homogeneous ensembles do not perform very different compared to the heterogeneous ensembles. This in contrast to Lessmann et al. (2015) where most heterogeneous ensembles perform statistically better than homogeneous ensembles. A possible explanation for this discrepancy might be the difference in the nature of the problem. Where this study works with stock data, Lessmann et al. (2015) uses a credit scoring data set. Another explanation could be a lack of diversity in the pool of classifiers from which the ensembles were selected. However, this seems unlikely because for example a KNN, NN and RF implement very different techniques which should result in diverse predictions. In line with Lessmann et al. (2015) stacking seems to be implementing the worst selection method.

Fourthly, from the three gradient boosted methods implemented in this study (XGB, LGB and CB) the LGB ensemble is for four out of six performance metrics statistically different from the best classifier. While the benchmarking study of Al Daoud (2019) concludes that LGB is the fastest and most accurate of the three, we can only confirm its superiority in speed. This disparity could be due to the application in different domains as well as to the difference in the size of the datasets. Whereas Al Daoud (2019) is situated in the credit scoring domain with data ranging from 50000 to 307507 rows our study implements a stock dataset with only 5716 rows but with more features. Lastly, considering the homogeneous classifiers, the Rotational Forest algorithm slightly underperforms.

5. Conclusion and practical implications

This study updates Ballings et al. (2015) by implementing a more elaborate benchmark. The main purpose remained to investigate which classifier performs optimally for the stock price prediction task. Therefore, 18 classifiers have been benchmarked, considering six performance metrics. Each algorithm is carefully tuned and an additional calibration step is computed. The results imply that the Hill climbing ensemble is very hard to outperform. However, in most cases the HCE simply selects the Random Forest ensemble. While RF is very easy to implement and is even able to give an indication of feature importance, HCE needs a large set of base models, is much harder to implement and is in essence a black box. Consequently, considering deployment effort and interpretability this study recommends the random forest algorithm as the way to tackle the stock price prediction task.

These results strongly confirm the potential of ensembles in stock price prediction with reported AUC's reaching 73%, precisions reaching 68% and accuracies 67%. This constatation should raise some doubt on the efficient market hypothesis even amongst its biggest proponents. Nevertheless, when more and more investors would use random forest, it would suffer from alpha decay, implying that these financial players try to make the same predictions, however, the first one will take most of the profit. Therefore, it could be worthwhile to consider heterogeneous ensembles because of their higher difficulty to deploy, consequently having suffered less from alpha decay.

6. Limitations and further research

In terms of generalization, it is hard to claim that the results of this study are valid for all stock datasets. Whilst our study focusses on European stocks, results could be different in other markets. In addition, although this study only focusses on one dataset, further research should be conducted considering multiple datasets to support generalization. In addition, while the focus of this study is only on fundamental data, it would be enriching to add a selection of technical, fundamental, economical and even sentiment variables.

In terms of the time window of the prediction, various setups are possible. Considering a business perspective, the question arises which time window generates most profit. Further studies could even opt for a Spark streaming setup which continuously updates the prediction model. Finally, while this study does not report strong arguments in favor of heterogeneous ensembles, these could be investigated further. In particular by selecting base classifiers based on diversity measures like the Q statistic (Kuncheva et al., 2000) or by inventing more superior search procedures, the performance could increase even further.

Appendices

Appendix A.

Table 5 Ensembles in other financial domains

Study	Domain	Ensembles				Performance metrics			Stat. tests used?
		Homo - geneous	Hetero - geneous	Stacking	#	AUC	ACC	Other	
A et al. (2020)	Financial distress	x			4	x		Sens, Prec, F	
Abellán & Castellano (2017)	Credit scoring	x			5	x	x		x
Abellán & Mantas (2014)	Credit scoring	x			2	x			x
Du et al. (2020)	Financial distress	x	x		5	x	x	Sens, Prec	
du Jardin (2018)	Bankruptcy	x	x		6	x			
du Jardin (2019)	Bankruptcy	x	x		6	x	x		
Ekinci & Erdal (2017)	Bankruptcy	x			4	x	x	Sens, Spec	
Feng et al. (2018)	Credit scoring	x	x		9	x	x	Prec,	x
Finlay (2011)	Credit scoring	x	x		10	x	x	Prec, PG	x
García et al. (2019)	Credit scoring & bankruptcy	x			7			Sens, Spec	
Lessmann et al. (2015)	Credit scoring	x	x	x	25	x	x	PG, H, BS	x
T. Zhang & Chi (2020)	Credit scoring	x	x		2	x		Sens, Prec, GM, F, MCC, BM	x
Tsai et al. (2014)	Bankruptcy	x			6		x		x
This study	**Stocks**	**x**	**x**	**x**	**12**	**x**	**x**	**Prec, MCE, BS, H**	**x**

*Abbreviations have the following meaning: AUC = Area Under the Curve, Acc = Accuracy, Prec = Precision, Spec = Specificity, PG = Partial Gini, BS = Brier Score, H = H-measure, F = F-score, FinPM = Financial Performance Measures

Appendix B.

Table 6 Overview classification techniques

Classifier family	Classifier	Abbreviation	Tuning parameters	# in grid search
Single classifiers	Logistic Regression	LogR	lambda = [10^{-3}, 10^{-2}, …, 10^{4}]	all
	K-Nearest Neighbor	KNN	k-neighbors = [1, 5, …, 150]	all
	Support Vector Machines	SVM	cost = [10^{-7}, 10^{-6}, …, 10^{2}] gamma = [10^{-3}, 10^{-2}, …, 10] degree = [2,3] kernal = linear, radial, poly, sigmoid	25
	Artificial Neural Network	ANN	first_nodes = [20, 40, 50, 80] second_nodes = [10, 20, 25, 40] lambda = [10^{-3}, 10^{-2}, 10^{4}]	25
	Decision Tree	DT	complexity = [0, 0.02, …, 0.3]	all
	Naïve bayes	NB	laplace = [0.01, 0.1, …, 100]	all
Homogeneous ensembles	Bagging (DT)	BagDT		
	Random Forest	RF	mtry = [20, 30, …, 60]	all
	Rotational Forest	RotF	subsets (K) = [4, 5, …, 40] iterations (L) = [5, 6, …,50]	25
	Extremely Randomized Trees	ERT	mtry = [20, 21, …, 60] cuts = [1, 2, 3]	25
	Stochastic Adaboost	SAda	iterations = [50, 150, …, 350]	all
	Extreme Gradient Boosting	XGB	max tree depth = 6 learning rate = [0.1, 0.2, 0.6] subsample = [0.7, 0.8, …, 1] rounds = [50, 75, …, 150]	25
	Light Gradient Boosting	LGB	max tree depth = 6 learning rate = [0.1, 0.2, 0.6] subsample = [0.7, 0.8, …, 1] rounds = [50, 75, …, 150]	25
	CatBoost	CB	max tree depth = 6 learning rate = [0.1, 0.2, 0.6] subsample = [0.7, 0.8, …, 1] iterations = [50, 75, …, 150]	25
Heterogeneous ensembles	Simple Averaging	SA		
	Weighted Averaging	WA		
	Stacking	Stack	lambda = [10^{-3}, 10^{-2}, …, 10^{4}]	all
	Hill Climbing Ensemble	HCE		

Appendix C.

From this sensitivity analysis it can be deducted that Platt scaling does have its advantages and disadvantages. As expected, the rank measures are not really affected by calibration. The probabilities of both the KNN and NB algorithm are clearly better calibrated, indicated by the huge drop in MCE. Unfortunately, because of the fixed threshold, both Prec and Acc metrics are affected differently, which is disadvantageous for our comparison. Note that this difference increases diversity among classifiers, which seems to improve the heterogeneous ensembles (SA, WA and HCE).

Table 7 Calibration sensitivity analysis

	no calibration						calibration					
	AUC	H	BS	MCE	Prec	Acc	AUC	H	BS	MCE	Prec	Acc
LogR	.650	.085	.226	.643	.561	.622	.650	.085	.226	.642	.563	.622
KNN	.682	.116	.218	Inf	.567	.639	.678	.113	.218	.633	.592	.639
SVM	.675	.111	.221	.631	.542	.638	.674	.111	.221	.632	.457	.641
NN	.677	.112	.219	.629	.457	.634	.678	.111	.219	.628	.375	.629
DT	.635	.067	.227	.644	.558	.629	.641	.077	.226	.644	.558	.630
NB	.623	.063	.553	Inf	.414	.438	.600	.063	.242	.677	.084	.583
RF	.727	.175	**.205**	**.593**	.626	.672	**.728**	.174	**.205**	**.595**	.626	.670
BagTree	.726	.174	**.205**	.594	.619	.670	.725	.173	**.205**	**.595**	.632	**.671**
RotF	.698	.144	.216	.620	.630	.657	.700	.141	.214	.616	.612	.662
ERT	.722	.169	.206	.598	.607	.667	.722	.168	.207	.599	.622	.665
Ada	.724	.171	.208	.600	.615	.666	.723	.168	.208	.605	.631	.672
XGB	.720	.165	.207	.599	.605	.665	.717	.167	.207	.603	.612	.662
LGB	.713	.156	.209	.605	.632	.657	.713	.156	.210	.606	.618	.663
CB	.721	.166	.206	.597	.613	.663	.719	.166	.208	.600	.610	.666
SA	**.728**	.177	.209	.606	.613	**.674**	.724	.172	.212	.613	.676	.667
WA	**.728**	**.180**	.207	.599	.615	.673	.725	**.181**	.211	.610	.669	.668
Stack	.689	.141	.217	.624	**.663**	.672	.683	.132	.219	.630	.659	.669
HCE	.727	.175	**.205**	**.593**	.638	.672	**.728**	.174	**.205**	**.595**	**.682**	**.671**

Appendix D.

Following link contains the data set used for this study, the r-code, all stored predictions and performance results:

https://github.ugent.be/jjdpover/Ensembles-in-stock-price-prediction.git

References

Abellán, J., & Castellano, J. G. (2017). A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Systems with Applications*, *73*, 1–10. https://doi.org/10.1016/j.eswa.2016.12.020

Abellán, J., & Mantas, C. J. (2014). Improving experimental studies about ensembles of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, *41*(8), 3825–3830. https://doi.org/10.1016/j.eswa.2013.12.003

Al Daoud, E. (2019). Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset. *International Journal of Computer and Information Engineering*, *13*(1), 6–10.

Albanis, G., & Batchelor, R. (2007). Combing heterogeneous classifiers for stock selection. *Intelligent Systems in Accounting, Finance and Management*, *176*(January), 161–176. https://doi.org/10.1002/isaf

Ampomah, E. K., Qin, Z., & Nyame, G. (2020). Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement. *Information (Switzerland)*, *11*(6). https://doi.org/10.3390/info11060332

Ampomah, E. K., Qin, Z., Nyame, G., & Botchey, F. E. (2020). Stock market decision support modeling with tree-based adaboost ensemble machine learning models. *Informatica (Slovenia)*, *44*(4), 477–489. https://doi.org/10.31449/INF.V44I4.3159

Anagnostopoulos, C., Hand, D. J., & Adams, N. M. (2012). *Measuring classification performance : the hmeasure package.* 1–15. https://cran.r-project.org/web/packages/hmeasure/vignettes/hmeasure.pdf

Anagnostopoulos, Christoforos. (2019). *Package ‘ hmeasure .’ 1*.

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques - Part II: Soft computing methods. *Expert Systems with Applications*, *36*(3 PART 2), 5932–5941. https://doi.org/10.1016/j.eswa.2008.07.006

Ballings, M., & Van den Poel, D. (2013). *AUC: Threshold independent performance measures for probabilistic classifiers*. https://cran.r-project.org/web/packages/AUC/index.html

Ballings, M., Van Den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, *42*(20), 7046–7056. https://doi.org/10.1016/j.eswa.2015.05.013

Basak, S., Kar, S., Saha, S., Khaidem, L., & Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers. *North American Journal of Economics and Finance*, *47*(June 2018), 552–567. https://doi.org/10.1016/j.najef.2018.06.013

Bequé, A., Coussement, K., Gayler, R., & Lessmann, S. (2017). Approaches for credit scorecard calibration: An empirical analysis. *Knowledge-Based Systems*, *134*, 213–227. https://doi.org/10.1016/j.knosys.2017.07.034

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Beygelzimer, A., Kakadet, S., Langford, J., Arya, S., Mount, D., & Li, S. (2019). *Package ‘ FNN .’*

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1023/A:1018054314350

Breiman, L. (2001). Random forests. *Random Forests*, 1–122. https://doi.org/10.1201/9780429469275-8

Butler, M., Guzman, M., & Turner, R. (2014). *A MODEL FOR STOCK P RICING*.

Carta, S., Corriga, A., Ferreira, A., Recupero, D. R., & Saia, R. (2019). A holistic auto-configurable ensemble machine learning strategy for financial trading. *Computation*, *7*(4), 1–25. https://doi.org/10.3390/computation7040067

Caruana, R., Ksikes, A., & Crew, G. (2014). “Ensemble selection from libraries of models.” *Proceedings of the Twenty-First International Conference on Machine Learning.* . https://doi.org/10.1145/1015330.1015432

Caruana, R., & Niculescu-Mizil, A. (2004). Data mining in metric space: An empirical analysis of supervised learning performance criteria. *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 69–78.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *ACM International Conference Proceeding Series*, *148*, 161–168. https://doi.org/10.1145/1143844.1143865

Chen, Y. J., Chen, Y. M., & Lu, C. L. (2017). Enhancement of stock market forecasting using an improved fundamental analysis-based approach. *Soft Computing*, *21*(13), 3735–3757. https://doi.org/10.1007/s00500-016-2028-y

Culp, M., Johnson, K., & Michailidis, G. (2016). *Package "ada" Title The R Package Ada for Stochastic Boosting*.

Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, *10*(7), 1895–1923. https://doi.org/10.1162/089976698300017197

Dongare, A. D., Kharde, R. R., & Kachare, A. D. (2012). Introduction to Artificial Neural Network (ANN) Methods. *International Journal of Engineering and Innovative Technology (IJEIT)*, *2*(1), 189–194.

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: Gradient boosting with categorical features support. *ArXiv*, 1–7.

du Jardin, P. (2019). Forecasting bankruptcy using biclustering and neural network-based ensembles. *Annals of Operations Research*. https://doi.org/10.1007/s10479-019-03283-2

Du, X., Li, W., Ruan, S., & Li, L. (2020). *CUS-heterogeneous ensemble-based financial distress prediction for imbalanced dataset with ensemble feature selection*. *97*. https://doi.org/10.1016/j.asoc.2020.106758

FAMA, E. F., & FRENCH, K. R. (1995). Size and Book-to-Market Factors in Earnings and Returns. *The Journal of Finance*, *50*(1), 131–155. https://doi.org/10.1111/j.1540-6261.1995.tb05169.x

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Feng, X., Xiao, Z., Zhong, B., Qiu, J., & Dong, Y. (2018). Dynamic ensemble classification for credit scoring using soft probability. *Applied Soft Computing Journal*, *65*, 139–151. https://doi.org/10.1016/j.asoc.2018.01.021

Friedman, J. (1999). Greedy Function Approximation : A Gradient Boosting Machine Author (s): Jerome H . Friedman Source : The Annals of Statistics , Vol . 29 , No . 5 (Oct ., 2001), pp . 1189-1232 Published by : Institute of Mathematical Statistics Stable URL : http://www. *The Annals of Statistics*, *29*(5), 1189–1232.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, *33*(1), 1–22.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, *180*(10), 2044–2064. https://doi.org/10.1016/j.ins.2009.12.010

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Gomez, D., & Rojas, A. (2018). *An Empirical Overview of the No Free Lunch Theorem and Its Effect on Real-World Machine Learning Classificatio*. *2733*(2015), 2709–2733. https://doi.org/10.1162/NECO

Graña, M., & Corchado, E. (2014). *A survey of multiple classifier systems as hybrid systems*. *16*, 3–17. https://doi.org/10.1016/j.inffus.2013.04.006

Gujarati, D. N., & Porter, D. C. (2009). Single-equation regression models. In *Introductory Econometrics: A Practical Approach*.

Halbleib, R., & Pohlmeier, W. (2012). Improving the value at risk forecasts: Theory and evidence from the financial crisis. *Journal of Economic Dynamics and Control*, *36*(8), 1212–1228. https://doi.org/10.1016/j.jedc.2011.10.005

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning*, *77*(1), 103–123. https://doi.org/10.1007/s10994-009-5119-5

He, T., Chen, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., & Lin, M. (2021). *Package ' xgboost .'* https://doi.org/10.1145/2939672.2939785>.This

Hornik, K., Weingessel, A., Leisch, F., & Davidmeyerr-projectorg, M. D. M. (2021). *Package 'e1071.'*

Huang, W., Nakamori, Y., & Wang, S. Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers and Operations Research*, *32*(10), 2513–2522. https://doi.org/10.1016/j.cor.2004.03.016

Ibragimov, B., & Gusev, G. (2019). Minimal Variance Sampling in Stochastic Gradient Boosting. *ArXiv*, *NeurIPS*.

Jan, Z., & Verma, B. (2020). Multiple strong and balanced cluster-based ensemble of deep learners.

Pattern Recognition, *107*, 107420. https://doi.org/10.1016/j.patcog.2020.107420
Jiang, M., Liu, J., Zhang, L., & Liu, C. (2020). An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms. *Physica A: Statistical Mechanics and Its Applications*, *541*(258), 122272. https://doi.org/10.1016/j.physa.2019.122272
Kaloop, M. R., Kumar, D., Zarzoura, F., Roy, B., & Hu, J. W. (2020). A wavelet - Particle swarm optimization - Extreme learning machine hybrid modeling for significant wave height prediction. *Ocean Engineering*, *213*(July), 107777. https://doi.org/10.1016/j.oceaneng.2020.107777
Kattan, M., Chun, F. K.-H., Graefen, M., Haese, A., & Karakiewicz, P. I. (2012). Recursive Partitioning. *Encyclopedia of Medical Decision Making*, 1–60. https://doi.org/10.4135/9781412971980.n280
Khaidem, L., Saha, S., & Dey, S. R. (2016). *Predicting the direction of stock market prices using random forest*. *00*(00), 1–20. http://arxiv.org/abs/1605.00003
Khan, W., Ghazanfar, M. A., Azam, M. A., Karami, A., Alyoubi, K. H., & Alfakeeh, A. S. (2020). Stock market prediction using machine learning classifiers and social media, news. *Journal of Ambient Intelligence and Humanized Computing*, *0123456789*. https://doi.org/10.1007/s12652-020-01839-w
Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference of Artificial Intelligence*, *March 2001*.
Kuncheva, L. I., Whitaker, C. J., & Shipp, C. A. (2000). *Is Independence Good For Combining Classifiers ?* 168–171.
Leković, M. (2019). Evidence for and Against the Validity of Efficient Market Hypothesis. *Economic Themes*, *56*(3), 369–387. https://doi.org/10.2478/ethemes-2018-0022
Lessmann, S., Baesens, B., Seow, H. V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124–136. https://doi.org/10.1016/j.ejor.2015.05.030
Manish, K., & Thnmozhi, M. (2011). Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest. *SSRN Electronic Journal*, 1–16. https://doi.org/10.2139/ssrn.876544
Mokoteli-Mokoteli, T., Ramsumar, S., & Vadapalli, H. (2019). the Efficiency of Ensemble Classifiers in Predicting the Johannesburg Stock Exchange All-Share Index Direction. *Journal of Financial Management, Markets and Institutions*, *07*(02), 1950001. https://doi.org/10.1142/s2282717x19500014
Moon, K. S., Jun, S., & Kim, H. (2018). Speed up of the majority voting ensemble method for the prediction of stock price directions. *Economic Computation and Economic Cybernetics Studies and Research*, *52*(1), 215–228. https://doi.org/10.24818/18423264/52.1.18.13
Nti, I. K., Adekoya, A. F., & Weyori, B. A. (2020). Efficient Stock-Market Prediction Using Ensemble Support Vector Machine. *Open Computer Science*, *10*(1), 153–163. https://doi.org/10.1515/comp-2020-0199
Ocak, I., & Seker, S. E. (2012). Estimation of elastic modulus of intact rocks by artificial neural network. *Rock Mechanics and Rock Engineering*, *45*(6), 1047–1054. https://doi.org/10.1007/s00603-012-0236-z
Oliveira, D. V. R., Cavalcanti, G. D. C., Porpino, T. N., Cruz, R. M. O., & Sabourin, R. (2018). K-Nearest Oracles Borderline Dynamic Classifier Ensemble Selection. *Proceedings of the International Joint Conference on Neural Networks*, *2018-July*. https://doi.org/10.1109/IJCNN.2018.8489737
Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications*, *42*(1), 259–268. https://doi.org/10.1016/j.eswa.2014.07.040
Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, *10*(3), 61–74.
Qian, B., & Rasheed, K. (2007). Stock market prediction with multiple classifiers. *Applied Intelligence*, *26*(1), 25–33. https://doi.org/10.1007/s10489-006-0001-7
Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *ArXiv*.
Reza, H., Masoud, A., Moghadam, E., & Dehghan, M. (2020). Expert Systems with Applications HBoost : A heterogeneous ensemble classifier based on the Boosting method and entropy measurement. *Expert Systems With Applications*, *157*, 113482. https://doi.org/10.1016/j.eswa.2020.113482
Rish, I. (2001). An empirical study of the naive Bayes classifie. *Physical Chemistry Chemical Physics*, *3*(22), 4863–4869. https://doi.org/10.1039/b104835j
Rodríguez, J. J., Kuncheva, L. I., & Alonso, C. J. (2006). Rotation forest: A New classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1619–1630.

https://doi.org/10.1109/TPAMI.2006.211
Rodriguez, P. N., & Rodriguez, A. (2004). Predicting stock market indices movements. *Computational Finance and Its …*, *November 2004*. http://pnrodriguez.com/files/IndexMovements.pdf
Sagiroglu, S., Temizel, T. T., Baykal, N., & Canbek, G. (2017). *Binary Classification Performance Measures / Metrics : 4*, 21–26.
Schechtman, E., & Schechtman, G. (2016). The Relationship between Gini Methodology and the ROC curve. *SSRN Electronic Journal*, 1–7. https://doi.org/10.2139/ssrn.2739245
Sesmero, M. P., Ledezma, A. I., & Sanchis, A. (2015). Generating ensembles of heterogeneous classifiers using Stacked Generalization. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *5*(1), 21–34. https://doi.org/10.1002/widm.1143
Sharabiani, M. T. A. (2016). *Multi-stage heterogeneous ensemble meta-learning with hands-off user-interface and stand-alone prediction using principal components regression : The R package EnsemblePCReg*. *Breiman 2001*, 1–32.
Sheskin, D. J. (2003). Parametric and non parametric statistical procedures: Third edition. *Handbook of Parametric and Nonparametric Statistical Procedures: Third Edition*, 1–1193.
Silge, J., Chow, F., Kuhn, M., & Wickham, H. (2021). *Package ' rsample .'*
Simm, J., & de Abril, I. M. (2015). Package ' extraTrees .' *October*, 1–7.
Strivastava, N. (2013). Improving Neural Networks with Dropout. *Integration of Climate Protection and Cultural Heritage: Aspects in Policy and Development Plans. Free and Hanseatic City of Hamburg*, *26*(4), 1–37.
Therneau, T., Atkinson, B., & Ripley, B. (2019). rpart: Recursive partitioning for classification, regression and survival trees. *CRAN R Package Version 4.1-15*. https://cran.r-project.org/package=rpart
Tsai, C. F., Hsu, Y. F., & Yen, D. C. (2014). A comparative study of classifier ensembles for bankruptcy prediction. *Applied Soft Computing Journal*, *24*, 977–984. https://doi.org/10.1016/j.asoc.2014.08.047
Tsai, C. F., Lin, Y. C., Yen, D. C., & Chen, Y. M. (2011). Predicting stock returns by classifier ensembles. *Applied Soft Computing Journal*, *11*(2), 2452–2459. https://doi.org/10.1016/j.asoc.2010.10.001
Umoru, B., Udobi-Owoloja, P. I., Nzekwe, G. U., Iyiegbuniwe, W. C., & Ezike, J. E. (2020). *Are stock returns predictable? The myth of efficient market hypothesis and random walk theory using Nigerian market data*. *4*(07), 115–130.
van Rijn, J. N., Holmes, G., Pfahringer, B., & Vanschoren, J. (2018). The online performance estimation framework: heterogeneous ensemble learning for data streams. *Machine Learning*, *107*(1), 149–176. https://doi.org/10.1007/s10994-017-5686-9
Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, *10*(5), 988–999. https://doi.org/10.1109/72.788640
Vaughan, D., & Dancho, M. (2021). *Package ' furrr .'*
Weng, B., Lu, L., Wang, X., Megahed, F. M., & Martinez, W. (2018). Predicting short-term stock prices using ensemble methods and online data sources. *Expert Systems with Applications*, *112*, 258–273. https://doi.org/10.1016/j.eswa.2018.06.016
Wright, M. N., Wager, S., & Probst, P. (2020). A Fast Implementation of Random Forests. *CRAN Repository*, 25.
Zhang, J., & Chen, L. (2019). Clustering-based undersampling with random over sampling examples and support vector machine for imbalanced classification of breast cancer diagnosis. *Computer Assisted Surgery*, *24*(sup2), 62–72. https://doi.org/10.1080/24699322.2019.1649074
Zhang, T., & Chi, G. (2020). A heterogeneous ensemble credit scoring model based on adaptive classifier selection: An application on imbalanced data. *International Journal of Finance and Economics*, *November 2019*, 1–14. https://doi.org/10.1002/ijfe.2019
Zhang, X., Li, A., & Pan, R. (2016). Stock trend prediction based on a new status box method and AdaBoost probabilistic support vector machine. *Applied Soft Computing Journal*, *49*, 385–398. https://doi.org/10.1016/j.asoc.2016.08.026
Zhang, Y., & Haghani, A. (2015). A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, *58*, 308–324. https://doi.org/10.1016/j.trc.2015.02.019
Zhou, F., Zhang, Q., Sornette, D., & Jiang, L. (2019). Cascading logistic regression onto gradient boosted decision trees for forecasting and trading stock indices. *Applied Soft Computing Journal*, *84*, 105747. https://doi.org/10.1016/j.asoc.2019.105747