Stockholms universitet

Gradient Boosting vs Random Forests: Predicting the Overnight Return of the OMXS30-index

Filip Bergkvist

Kandidatuppsats i matematisk statistik
Bachelor Thesis in Mathematical Statistics

Kandidatuppsats 2022:4
Matematisk statistik
Juni 2022

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Mathematical Statistics
Stockholm University
Bachelor Thesis **2022:4**
http://www.math.su.se

Gradient Boosting vs Random Forests: Predicting the Overnight Return of the OMXS30-index

Filip Bergkvist*

June 2022

Abstract

Statistical learning is an important tool for statistical analysis in many areas of finance. In the current economy, transactions are made all across the world making different stock markets increasingly integrated. This thesis will compare the prediction accuracy of two different machine learning techniques; gradient boosting and random forest. Using only a few predictor variables consisting of stock indices, currencies and commodities the different models will try to predict whether the overnight return of the OMXS30 index is negative or not. In this binary classification problem, the gradient boosting model achieved a slightly better prediction accuracy than the random forest model and both of them outperformed the Zero rule classifier. The variable importance when predicting the response showed for both models that the Hang Seng Index, Nikkei 225 Index and Nasdaq Composite Index had the most relative influence out of the predictor variables. Even if gradient boosting outperformed random forest in prediction accuracy, the difference in computational cost between the models must be taken into account when evaluating overall model performance.The GBM-model, even with a restricted parameter space, was very time consuming to train compared to the random forest model. Which model to use in a classification task depends on how sensitive the data is. If it is of high importance to get correct classifications it could justify training a model that requires more computational cost.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: filipbergkvist@hotmail.com. Supervisor: Ola Hössjer and Nils Engler.

Acknowledgement

This is a bachelor's thesis of 15 ECTS in Mathematical Statistics at the Department of Mathematics at Stockholm University. I would like to express my sincere thanks to my supervisors Ola Hössjer and Nils Engler for their valuable guidance and support throughout the writing of this thesis.

Contents

1 Introduction

Predicting the movement of the stock market in order to maximize investment returns has historically been and currently is a challenging area for both academics as well as traders. Multiple factors such as politics, global economic conditions and financial performance have an impact on the stock price. However, the variety of parameters combined with a rich historical data base have enabled financial analysts as well as data scientists to use and explore different type of financial models in order to predict movements in prices [13].

The OMXS30 is a stock index that consists of the 30 most traded stocks listed on Nasdaq Stockholm. In the current economy, transactions are made all across the world and since many different stock markets are integrated with each other it also means that movements in some stock markets might have an impact on movements in other stock markets. The economy in the United states, for example, has a lot of influence on the world economy which in turn effects the Swedish economy [14].

This paper will compare the prediction accuracy of two different machine learning methods; gradient boosting and random forest. Both of these methods are ensemble methods and similar in the sense that they combine a number of smaller classifiers into a single classifier. Random forests is a modification of bagging in the sense that it averages the large collection of trees after building them. Every classifier is trained independently from the rest. Gradient boosting uses boosting as a method, that unlike bagging allows each of the weak learners to improve over time where all the members in the committee of weak learners cast a weighted vote. Each iteration in the algorithm a new classifier is added to a already trained ensemble [8]. Section 2 in this paper provides more detailed descriptions of these methods.

Both of the models will be trained in classifying the response using 12 predictor variables for each observation. These variables include log returns of 6 different stock indices around the world and the log returns of some currencies and commodities. The models will use these predictor variables in order to predict the overnight log return of the OMXS30 index. The variables were selected in part for evaluating how well the models can predict the opening movement of the OMXS30 index based on only a few influences from the world economy during the time that the Swedish stock market is closed. The values of the response and predictor variables are taken over a 10 year period between 2010 and 2020. The response variable will be set to 0 if the overnight log return is negative and otherwise set to 1. Deviating opening hours, removal of null values and creating a lag for some variables resulted in a final data set of 2083 observations. The order of the observations in the data was randomized before dividing the data into a training set used to fit the models and a test set used to measure the performance of the fitted models.

CFD (Contracts for difference) trading allows the trader to speculate on financial markets without the requirement to buy the underlying asset [9]. Since there exists trading platforms that enable CFD trading around-the-clock, traders have the possibility to react to news during the weekend and when the market is closed and as of such do not have to wait until the market opens [10]. As a result these financial instruments often with great accuracy can indicate the opening movement of the OMXS30-index before the market opens. The machine learning methods used in this paper with only 12 numerical predictor variables will not be outperforming that accuracy and as of such the result of the analysis will be interpreted and evaluated based on the prediction accuracy of the models compared to each other as well as compared to the Zero Rule classifier.

The relative influence of the predictor variables in predicting the overnight log return is also discussed as well as visualized in section 4.1.1 and 4.2.2. We find that for both of the models the overnight log return of the Hang Seng Index (Hong Kong) has the most relative influence in predicting the movement of OMXS30. This is an interesting result, since the intraday return of the Nikkei 225 Index (Tokyo) is calculated only an hour before the Swedish stock market opens and more than four hours after the Hang Seng Index is calculated. The SSE Composite Index (Shanghai) is calculated at the same time as the Hang Seng Index and has a low relative influence in the predictions. A hypothesis is that the Hang Seng Index contains stocks of companies that are more embedded in the world economy and as of such has more impact on the stocks in the OMXS30, but no such economical analysis is present in this paper.

The prediction accuracies of both models are compared in section 5.3. Out of a test data set of 417 observations, only two true classifications separated gradient boosting from random forest. For gradient boosting the measured prediction accuracy is 0.6954 and for random forest it is 0.6906. They both outperformed the Zero Rule classifier that had a prediction accuracy of 0.5683. Even if the GBM model performed slightly better than random forest, the computational cost when tuning the model must be taken in to account. Random forest is a model with fewer parameters to tune and is more computationally feasible than gradient boosting. Even when the training dataset is relatively small (2083 observations) and the hyperparameter space is restricted it was time consuming to tune the GBM model. Which model to use when classifying observations depends on the data and personal preferences. For some response variables it could be very important that observations are classified correctly (sensitivity) and in those cases it could be justified to train a model that requires more computational cost.

Section 2 in this paper is designed to provide the reader with a theoretical background on the concept of statistical learning, the methods used and how the performance of the models are measured. This section will also provide a description of the programming packages that are used to implement the methods in practice. A more granular description of the data is given in Section 3 before tuning the respective model on the training data in Section 4. Section 5 presents the results which are interpreted and discussed in Section 6.

2 Theory

2.1 Statistical learning

The material of section 2.1 is taken from [8].

Statistical learning is an important tool for statistical analysis and learning from data in many areas of science, finance and industry. In supervised learning, as opposed to unsupervised learning, there exists a response variable which enables the use of performance measurement when building the model. If we for example assume that the errors are additive, then supervised learning attempts to learn the model $Y = f(X) + \epsilon$, where Y is the response variable, X is a vector containing p predictor variables and ϵ is an error term, by using the observed values x_i from the training set $\tau = \{(x_i, y_i);\ i = 1, \ldots, N\}$ to "teach" the model by finding a predictor of the output $\hat{f}(x_i)$ of the true response values y_i. There are different types of scenarios with quantitative response parameter values as well as categorical ones. Regardless of which type of scale for the response the process in

building a supervised machine learning model $X \rightarrow \hat{f}(X)$ is similar. We use data to train and build a prediction model which then can be used to find predictions $\hat{f}(X)$ of new unseen response variables Y from their predictor variables X. A good learner is said to be a built model that with great accuracy predicts new data points.

The analysis in this paper treats the response variable as binary, where the two different classes depend on whether the overnight log return of the OMXS30 stock index is positive (≥ 0) or negative. This is a supervised learning problem that requires the different models to determine a prediction $G(X) \in \{0, 1\}$ of the response $Y \in \{0, 1\}$, that is a function of the predictor vector X. For the gradient boosting algorithm the predictions $\hat{Y}$ of a binary target Y will lie in $[0, 1]$, and the class label $G(X)$ is determined depending on whether $\hat{Y} > 0.5$ or not. This way the gradient boosting algorithm is used to classify the observations.

2.1.1 Decision Trees

One popular approach for representing classifiers are decision trees. They are expressed as a recursive partition of the instance space and has the properties of a directed tree. All the nodes in a directed tree except for the root node have exactly one incoming edge. Nodes with outgoing edges are called internal nodes and the nodes with no outgoing edges are the terminal nodes of the tree. Each internal node of the decision tree splits the instance space into two or more sub-spaces depending on a discrete function of the parameter values belonging to that observation. The terminal nodes, also called leaves, are assigned to a class representing the most appropriate target value or hold a probability vector indicating the probability of the target attribute having a certain value [16].

2.1.2 Classification trees

In trees where the response variable is of categorical nature and takes values $1, 2, ..., K$, the proportion of class k observations in node j is

$$\hat{p}_{jk} = \frac{1}{N_j} \sum_{x_i \in R_j} I(y_i = k) \tag{1}$$

where R_j refers to the region of the predictor space that node j corresponds to, whereas N_j is the number of observations in the training dataset whose predictor variables x_i fall into this region. The majority class in node j determines the classification of all observations in this node [8].

2.1.3 Model Selection

The material of section 2.1.3 is taken from [8].

The performance of a learning method is related to its predictive accuracy on the independent test data. Since this guides the choice of model the selection of which assessment-method to use when selecting the ultimate model is important. The generalization error is the prediction error over an independent test sample

$$Err_\tau = E[L(Y, \hat{f}(X))|\tau] \tag{2}$$

where L is the loss function, (X, Y) is drawn randomly from the distribution of the test dataset, whereas the training set τ is fixed. It is tricky to get the expected prediction error $Err = E[L(Y, \hat{f}(X))] = E[Err_\tau]$ of our estimated model $\hat{f}$, since training error is not a good estimate of the test error. The more complex the model gets, the more training data it uses to adapt the model. This decreases the bias but it also increases the variance due to overfitting. A model with zero training error is maximally overfit to the training data and will typically generalize poorly.

2.1.4 Cross validation

The material of section 2.1.4 is taken from [8].

One of the most used and simple performance assessment methods of a learning model is cross-validation. The method estimates the average generalization error $Err = E[L(Y, \hat{f}(X))]$ when the method $\hat{f}(X)$ is applied to an independent test sample from the distribution of X and Y. When dealing with large datasets, an ideal approach would be to divide the data set into training and test data and use a validation set to validate the model. When dealing with more sparse datasets, K-fold cross-validation is an effective method to assess the performance of the prediction model. The K-fold cross validation sets aside a different part of the data to fit and test the model to estimate the average error.

In order to describe K-fold cross-validation let $\kappa : \{1, ...N\} \to \{1, ..., K\}$ be an indexing function that indicates the partition to which observation i is allocated by a randomized assignment of training data into K folds of approximately equal size. The cross-validation estimate makes use of the prediction error of the fitted function $\hat{f}^{-k}(x)$ with the k:th fold of training data removed. It is defined as

$$CV(\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)). \tag{3}$$

The number of folds used, K, usually depends on the data.

2.2 Gradient boosted trees

The theory in this section follows that of chapter 10 in [8].

2.2.1 Boosting

Boosting is one of the most powerful learning ideas introduced in the last twenty years and it comes from the idea of combining many "weak" classifiers to produce a powerful "committee" [8]. A weak classifier is a classifier with an error rate on the training sample only slightly better than what would have occurred with random guessing. Given a classifier $G(X)$ that produces a prediction of the binary output variable Y. Suppose we code the two possible outcomes of this outcome variable as $Y \in \{-1, 1\}$. Then, given a vector of predictor variables X, the error rate on the training sample is

$$err = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i)) \tag{4}$$

and the error rate on future predictions is $Err = E_{XY} I(Y \neq G(X))$.

Boosting is the process of sequentially applying the weak classification algorithm to iteratively modified versions of the data, hence producing a sequence (read. "committee") of weak classifiers $G_m(x), m = 1, 2, .., M$. The numbers $a_1, ..., a_m$ of the boosting algorithm weight the classifiers $G_m(x)$ and the weighted majority vote of this sequence of weak classifiers results in the final prediction

$$G(x) = \text{sign}(\sum_{m=1}^{M} a_m G_m(x)). \tag{5}$$

The more accurate certain classifiers in the sequence of classifiers are the more influence, i.e. the higher weight, they get in the final prediction.

2.2.2 Stagewise Additive modeling

The expression (5) of the previous section visualizes boosting as a way of fitting an additive expansion in a set of basis function which in (5) is represented by the individual classifiers $G_m(x) \in \{-1, 1\}$. The basis function expansions are more generally represented by

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m) \tag{6}$$

where $\beta_m, m = 1, 2, ...M$ are the expansion coefficients and $b(x;\gamma) \in \mathbb{R}$ are usually simple functions of the multivariate argument x, characterized by a set of parameters γ.

The basis functions are normally fit by minimizing a loss function averaged over training data,

$$\min_{\{\beta_m,\gamma_m\}_1^M} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m b(x_i;\gamma_m)) \tag{7}$$

and forward stagewise modeling approximates this minimization by sequentially adding new basis functions to the expansion without changing the already added coefficients and parameters.

Algorithm 1 Forward Stagewise Additive Modeling Algorithm

1. Initialize $f_0(x) = 0$.

for $m = 1$ *to* M **do**

(a) Compute $(\beta_m, \gamma_m) = \text{argmin}_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma))$

(b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x;\gamma_m)$.

end for

In the algorithm above the optimal basis function $b(x;\gamma_m)$ and corresponding coefficient β_m are added to the current expansion $f_{m-1}(x)$ at each iteration m. This produces $f_m(x)$ and the process re-iterates.

2.2.3 Boosting trees

A classification tree, as mentioned in sections 2.1.1 and 2.1.2, partition the input space of all the parameter values into disjoint regions $R_j, j = 1, 2, ..., J$ which are represented by the terminal nodes of the tree. The predictive rule of a boosting tree is that a constant γ is assigned to each region $x \in R_j \rightarrow f(x) = \gamma_j$. This enables the tree to be expressed as

$$T(x;\theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j) \tag{8}$$

with parameters $\theta = \{R_j, \gamma_j\}_1^J$ that are obtained by solving a combinatorial optimization problem

$$\hat{\theta} = \text{argmin}_\theta \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j). \tag{9}$$

The sum of trees of the type expressed in (8), induced using the algorithm for forward stagewise modeling, generates the boosted tree model

$$f_M(x) = \sum_{m=1}^{M} T(x; \theta_m). \tag{10}$$

At each step in the forward stagewise modeling algorithm given the current model $f_{m-1}(x)$, for the region set and constants $\theta_m = \{R_{jm}, \gamma_{jm}\}_1^{J_m}$ of the next tree, the algorithm must solve for

$$\hat{\theta}_m = \text{argmin}_{\theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \theta_m)). \tag{11}$$

2.2.4 Numerical Optimization and Gradient descent

In order to solve (11) with any differentiable loss criterion numerical optimization can be used. Suppose we want to use the function $f(x)$ to predict the response y on the training data. Defining the loss

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{12}$$

and then minimizing $L(f)$ with respect to f can be viewed as a numerical optimization $\hat{f} = \text{argmin}_f L(f)$ where $f = \{f(x_1), ..., f(x_N)\} \in \mathbb{R}^N$.

The procedure of numerical optimization involves using the initial guess $f_0 = h_0$ and solving for $\hat{f} = f_M$ by using the sum of component vectors

$$f_M = \sum_{m=0}^{M} h_m, h_m \in \mathbb{R}^N \tag{13}$$

letting $f_m = \sum_{l=0}^{m} h_l$ be determined by the current parameter which is determined by the sum of the previous updates. Computing the vector h_m can be done using a method called steepest descent. The method uses a scalar p_m and the gradient g_m of $L(f)$ evaluated at $f = f_{m-1}$ to compute each increment vector $h_m = -p_m g_m$. The gradient $g_m \in \mathbb{R}^N$ has components

$$g_{im} = [\frac{\delta L(y_i, f(x_i))}{\delta f(x_i)}]_{f(x_i)=f_{m-1}(x_i)} \tag{14}$$

and the step length $p_m > 0$ is the solution to $p_m = \text{argmin}\, L(f_{m-1} - pg_m)$.

Before the iteration of the numerical optimization procedure is done the current solution gets updated by $f_m = f_{m-1} - p_m g_m$ and the full procedure is then repeated at the next iteration.

2.2.5 Gradient Boosting

If minimizing the loss of the training data $L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$ was the only objective then the steepest descent method would be advantageous. Tree predictions $T(x_i; \theta_m)$ are similar to the components of the gradient descent formula (13) mentioned in section 2.2.4 but are preferable when generalizing the function $f_M(x)$ to unseen data since the tree components $t_m = \{T(x_1; \theta_m), ..., T(x_N; \theta_m)\}^T$ are not independent but constrained to the prediction of a J_m-terminal node tree. By inducing a tree $T(x; \theta_m)$ at the mth iteration of the stagewise approach with predictions t_m that aim toward similarity of the negative gradient, i.e. by fitting a tree T to the negative gradient values $-g_{im}$, where $g_{im} = [\frac{\delta L(y_i, f(x_i))}{\delta f(x_i)}]_{f(x_i)=f_{m-1}(x_i)}$ and using squared error to measure closeness, we solve for

$$\hat{\theta}_m = \text{argmin}_\theta \sum_{i=1}^{N} (-g_{im} - T(x_i; \theta))^2. \tag{15}$$

There are different types of loss functions depending on the type of response variable. The loss function used for a categorical response is the K-class multinomial deviance. Given $p_k(x) = e^{f_k(x)} / \sum_{l=1}^{K} e^{f_l(x)}$ which ensures $0 \leq p_k(x) \leq 1$, the loss function for the K-class multinomial deviance is

$$L(y, p(x)) = -\sum_{k=1}^{K} I(y = k) f_k(x) + \log(\sum_{l=1}^{K} e^{f_l(x)}), \tag{16}$$

where $p(x) = (p_1(x), \ldots, p_K(x))$ and $y = k$ refers to the k:th possible outcome of the response variable.

2.2.6 GBM-algorithm

The gradient boosting algorithm is based on the theory of sections 2.2.1-2.2.5. It is used for classification and is summarized in Algorithm 2.
The gradient boosting procedure as described in sections 2.2.1-2.2.5 is implemented in the R-package *gbm* which is described more in section 2.6.2.

2.2.7 Hyperparameters

The optimal size for a tree in a tree building algorithm was historically estimated separately when it was built, where very large trees were first induced and later bottom-up pruned. This made the boosting procedure very costly in terms of time, especially since the method assumes that each tree is the last one in the expansion. In order to avoid the problem of computational cost, i.e. bottom-up pruning each tree for the optimal number of terminal nodes, a simple strategy is to restrict all trees to be the same size $J_m = J$ for all m. At each step in the iteration a J-terminal node tree is induced

Algorithm 2 GBM-algorithm for classification

Initialize $f_{0k}(x) = \text{argmin}_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$, for $k = 1, \ldots, K$
for $m = 1$ *to* M **do**
  for $k = 1$ *to* K **do**
    Set:
    $p_k(x) = e^{f_{k,m-1}(x)} / \sum_{l=1}^{K} e^{f_{l,m-1}(x)}$.
    1. $-g_{ikm} = I(y_i = G_k) - p_k(x_i), i = 1, 2, ..., N$
    2. Fit a regression tree to $-g_{ikm}, i = 1, 2, ..., N$ which gives the regions $R_{jkm}, j = 1, 2, ..., J_{km}$.
    3. We compute $\gamma_{jkm} = \text{argmin}_\gamma \sum_{x_i \in R_{jkm}} L(y_i, f_{k,(m-1)}(x_i) + \gamma), j = 1, 2, ...J_{km}$.
    4. Update $f_{km}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_{km}} \gamma_{jkm} I(x \in R_{jkm})$.
  end for
end for
Output: $\hat{f}(x) = f_M(x) = (f_{1M}(x), \ldots, f_{KM}(x))$
return $\hat{f}_k(x) = f_{kM}(x),\ k = 1, 2, ..., K$

and hence the tree size becomes a parameter manageable to be tuned for the user in order to maximize prediction performance on validation (or test) data.

Another parameter that can be tuned in the gradient boosting algorithm to enhance performance is the number of boosting iterations M. A large enough M will reduce the training risk $L(f_M)$ but might lead to a overfit tree. A way to select the optimal estimate of M, M^*, is to monitor the future application dependent prediction risk as a function of M on a validation sample. In the analysis this is achieved by using the R-function *gbm.perf* where a 10-fold cross-validation sample is used to select the optimal estimate of M^*.

The shrinkage parameter v controls the learning rate of the boosting procedure by adding a penalty to each contribution. From the gradient boosting algorithm in section 2.2.6, step 4 of this algorithm (within the for loop over k) is thus replaced by

$$f_{km}(x) = f_{k,m-1}(x) + v \times \sum_{j=1}^{J_{km}} \gamma_{jkm} I(x \in R_{jkm}). \tag{17}$$

It has been found by (Friedman, 2001) that smaller values of the shrinkage parameter v enhances test error, and requires more boosting iterations M (since smaller steps are taken towards the optimal solution). This is however a matter of computational cost, since small improvements in prediction accuracy on the test data might not be justified because of a very large

computational time.

2.3 Random Forests

The theory in this section follows that of chapter 15 in [8].

2.3.1 Bagging

Bagging, introduced in section 8.7 in Hastie et al (2017), is a way of estimating the prediction $\hat{f}(x)$ of $f(x) = E(Y|X = x)$ at input x. The model is fit to training data and bagging averages the prediction over a collection of bootstrap samples and thereby reduces its variance. For every bootstrap sample $Z^{*b}, b = 1, 2, ..., B$ on the training data $Z = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ a prediction $\hat{f}^{*b}(x)$ is obtained and this generates the bagging estimate

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{18}$$

In the case of trees in K-class classification that produces a classifier $\hat{G}(x) = \text{argmax}_k \hat{f}_k(x)$ the bagged estimate $\hat{f}_{bag}(x)$ (18) is a K-vector with components $[p_1(x), p_2(x), ..., p_K(x)]$, with $p_k(x)$ representing the proportion of trees that are predicting class k at x. So, the bagged classifier counts the number of "votes" from the B trees and selects the class that received the most votes, $\hat{G}_{bag}(x) = \text{argmax}_k p_k(x)$. In other words, a "committee" of trees each cast a vote for the predicted class.

2.3.2 Random Forest

Random forests (Breiman, 2001) builds a large collection of de-correlated trees and averages them, which can be seen as a substantial modification of the bagging technique described in the previous section. Since each tree is identically distributed (i.d.), the expectation of an average of B such trees is the same as the expectation of any one of them. So, the bias of the average of B trees is the same as that of the individual trees which indicates that in order to improve the performance of the model variance reduction should be the aim. If we consider B identically distributed and independent random variables, each with variance σ^2, then the variance of the average of these B random variables is σ^2/B. With pairwise correlation p between the random variables, the variance of the average is

$$p\sigma^2 + \frac{1-p}{B}\sigma^2. \tag{19}$$

We can see from the formula above that when the number of variables B get large enough the second term disappears and the size of the correlation parameter p limits the benefits of averaging. The random forest algorithm

aims to improve the variance reduction by reducing the correlation parameter p without causing too big of an increase in the variance. When growing a tree on the bootstrapped dataset the random selection of the input variables achieves the variance reduction. More specifically, in the tree growing process on a bootstrapped dataset a subset of $m \leq p$ input variables are selected at random for splitting, where p in this case represents the number of variables in the data. In classification problems, the default value for m is $\sqrt{p}$. Intuitively, by reducing the number m of variables selected at random for splitting at each tree $T(x;\theta_b)$ (where θ_b are the parameters of the bth random forest tree grown) one reduces the correlation between any pair of trees in the ensemble and thus by formula (19) one reduces the variance of the average.

The algorithm for random forest for classification is presented below in Algorithm 3.

Algorithm 3 Random Forest for Classification

1.
for $b = 1$ *to* B **do**
(a) Draw a bootstrap sample Z^{*b} of size N from the training data.
(b) Grow a random forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
i. Select m variables at random from the p variables.
ii. Pick the best variable/split-point among the m.
iii. Split the node into two daughter nodes.
end for
2. Output the ensemble of trees $\{T_b\}_1^B$.
To make a classification prediction among K classes at a new point x:
Let $\hat{C}_b(x) \in \{1, \ldots, K\}$ be the class prediction of the bth random-forest tree.
Then $\hat{C}_{rf}^B(x) = \text{majorityvote}\{\hat{C}_b(x)\}_1^B$.

2.3.3 Out-of-bag (OOB) error estimate

An important feature in fitting a random forest model on a training data set $Z = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ is the use of out-of-bag (OOB) samples. The OOB error estimate is very similar to the K-fold Cross-validation prediction error $CV(\hat{f})$, in the sense that for each observation $z_i = (x_i, y_i)$ its random forest is constructed by averaging only those trees corresponding to bootstrap samples in which z_i did not appear. Once the OOB error stabilizes the training of the model can be stopped since the sufficient number of trees have been resampled.

2.4 Variable importance

2.4.1 GBM

For tree based boosting methods the approximate relative influence, developed by Friedman (2001), of a variable x_l is defined as follows: Consider first a regression tree T, and define the relative importance of x_l for the fit of this tree as

$$I_l^2(T) = \sum_{t \in \text{splits on } x_j} I_t^2 \tag{20}$$

where I_t^2 is the empirical improvement of split number t of T when splitting on x_l at that point [15].

For K-class classification each of the K separate models $f_k(x), k = 1, 2, ...K$ are consisting of a sum of trees

$$f_k(x) = \sum_{m=1}^{M} T_{km}(x)[8]. \tag{21}$$

The relative importance measure for a predictor variable x_l in an additive tree expansion is averaged over the trees

$$I_l^2 = \frac{1}{M} \sum_{m=1}^{M} I_l^2(T_m) \tag{22}$$

which in the case of K-class classification generalizes to

$$I_{lk}^2 = \frac{1}{M} \sum_{m=1}^{M} I_l^2(T_{km}) \tag{23}$$

where I_{lk}^2 is the relevance of the predictor variable x_l in separating the class k observations from all the other classes. Averaging over all the classes results in the overall relevance

$$I_l^2 = \frac{1}{K} \sum_{k=1}^{K} I_{lk}^2$$

of x_l for separating any class [8].

2.4.2 Random Forest

The relative importance of the variables can be constructed in the same way as for *GBM*. With the random selection among the m variables at each split of a tree in the random forest, the importance of this variable increases

for that particular tree, while no such selection occurs with boosting. The importance measure of a predictor variable is accumulated over all trees in the forest and is determined by the improvement in the split-criterion. In the analysis such a plot is presented in section 4.2.2 and it bases the predictor variable importance on the Gini splitting index, the same as for gradient boosting. The Gini index is given by

$$\sum_{k \neq k'} \hat{p}_{jk}\hat{p}_{jk'} = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk}). \tag{24}$$

where in a node j, representing a region R_j with N_j observations we let

$$\hat{p}_{jk} = \frac{1}{N_j} \sum_{x_i \in R_j} I(y_i = k) \tag{25}$$

be the proportion of class k observations in node j. In the case of two classes the Gini index measurement is given by $2p(1-p)$ where p is the probability of an object being classified to a particular class [8].

2.5 Model Accuracy

2.5.1 Misclassification Error rate

The performance of the random forest model and the gradient boosting model will be evaluated using the misclassification error rate, which following (25) is given by

$$\frac{1}{N_j} \sum_{i \in R_j} I(y_i \neq k(j)) = 1 - \hat{p}_{jk(j)}. \tag{26}$$

In the case of two classes, as is the case in this paper, the misclassification measurement is $\min(p, 1-p)$ if p is the proportion in the second class.

2.5.2 ROC

Generally in classification problems, misclassifying a true observation as false might have a bigger consequence than misclassifying a false observation as true. The terms *specificity* and *sensitivity* can be used to characterize predictions. To exemplify, in a medical situation the probability of predicting disease given true state of disease would be the *sensitivity* and *specificity* would be the probability of predicting non-disease given a true state of non-disease. The receiver operating characteristic curve (ROC) summarizes the tradeoff between *specificity* and *sensitivity* [8]. It plots the true positive rate (sensitivity) against the false positive rate (one minus the specificity) for all possible thresholds [7]. The area under the ROC curve is used as a single measure of the accuracy of the GBM-model when tuning the parameters. The Area Under the Curve (AUC) measures the ability of a classifier

to distinguish between classes, where values close to one indicates that the model is classifying with very high prediction. If the value of AUC is 0.5, the classifier performs equally well as random guessing, whereas if AUC were 0, it would mean that the model classifies all positives as negatives and regardless of the false positive rate [4].

2.5.3 Zero rule

We will also evaluate the performance of the two models using the Zero Rule as a benchmark value. The Zero Rule can be used as a benchmark value and it simply represents the most frequently occurring classification in a test set of data by classifying all of the data points to that class [6].

2.6 Package in R

2.6.1 Random Forest

In the analysis the R-package *randomForest* will be used which implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification [1]. The function contains a parameter *mtry* which represents the number of variables randomly sampled as candidates at each split. The default values for classification is $mtry = \sqrt{p}$ where p is the number of predictor variables in the data. The parameter *ntree* determines the number of trees to grow and in order to ensure that every input row gets predicted at least a few times we initially set the parameter value of $ntree = 5000$. The function returns a random forest model with parameters such as *err.rate*, which is a vector with the error rates of the prediction on the input training data. Its i-th element is the OOB-error rate for all trees up to the i-th tree [2] [1].

2.6.2 Gradient Boosting Machine (GBM)

The gradient boosting procedure is implemented in the analysis using the R-package *gbm*. The Gradient Boosting Machine, GBM, is an extension of the work of Friedman, Hastie, and Tibshirani (2000). Based on additional input from the companion papers Friedman (2001) and Friedman (2002) the method has been generated using the connection between boosting and optimization [15]. Boosting as implemented in the R-package *gbm* requires the tuning of some important parameters.

First, a loss function must be selected which by default for binary classification problems is the Bernoulli loss function given by

$$-2\frac{1}{\sum_{i=1}^{N} w_i}\sum_{i=1}^{N} w_i(y_i f(x_i) - \log(1+\exp(f(x_i))))[15]. \tag{27}$$

This is a weighted generalization of the K-class multinomial deviance loss function described in section 2.2.5, where $w_1, \ldots, w_N$ are weights assigned to all observations of the training dataset and K equals 2 since the response variable in our data is of binary nature. In section 2.2.7. we mentioned that other parameters can be tuned to enhance the *gbm* model performance. Different values of the shrinkage parameter, number of boosting iterations and the minimum number of observations per node will be evaluated. Another parameter that can be tuned is the *interaction.depth* parameter that determines the depth of the tree and it can be used to control the order of approximation. Different values of these parameters create the hyperparameter space for tuning the *gbm* model. The function *Grid Search* from the R-package *caret* was used to tune the function *gbm* using different combinations of the hyperparameters in Table 1 [12].

Table 1: Hyperparameter values of the Gradient Boosting Machine (GBM), used in grid search.

Parameter	Values
interaction.depth	{ 1,2,3,4 }
n.trees	{ 200,500,800,1000,1500,1800,2000,5000 }
shrinkage	{ 0.001, 0.003, 0.005, 0.01, 0.01 }
n.minobsinnode	{ 5, 10, 20 }

The values of the shrinkage parameter and the number of trees were chosen in part with regards to computational cost. The smaller the shrinkage parameter the more iterations are needed. It is suggested in Hastie et al. (2017) that a number $4 \leq J \leq 8$ of terminal nodes of each tree works fine in the context of boosting, whereas different values within that range have a smaller impact on the performance of the model [8].This is tuned by the *interaction.depth* parameter, where an interaction depth of D generates $J = (2 \times D) + 1$ terminal nodes [15] [8].

As mentioned in section 2.2.7 we will find the optimal number of boosting iterations M using cross validation via the R-function *gbm.perf* from the R-package *gbm*. During boosting, simple base-learners are iteratively combined to produce the final estimate. The graph 4 of $CV(\hat{f}_M)$ shows the performance metric's evolution as the gradient boosting algorithm combines a progressively larger number M of base learners. If the cross validation error does not improve with more boosting iterations the model might overfit the training data [15] [8].

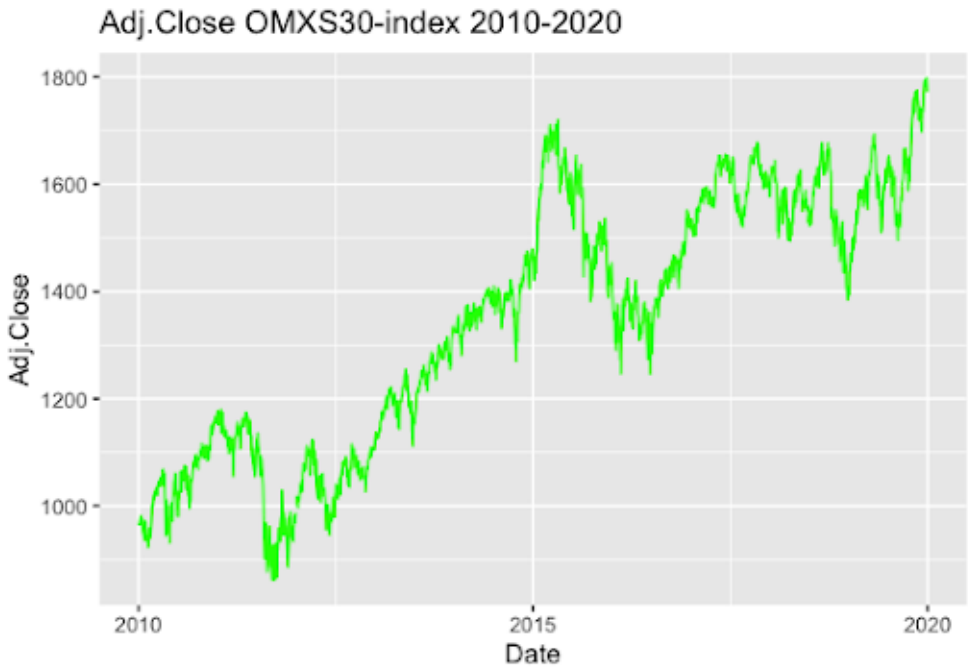

Figure 1: OMXS30 Adj.Close, the adjusted closing price of OMXS30, during ten years (2010-01-01 to 2020-01-01).

3 Data

3.1 OMXS30

OMXS30 is a stock market index listed on the Stockholm stock market and it consists of the 30 most traded stocks on Nasdaq Stockholm. It is a market weighted index meaning that the stocks included in the index have a weight in proportion to the respective stocks total market value. The 30 stocks are revised every sixth months and redefined depending on their historical trading volumes. There are however some rules to prevent the variation of stocks in the index from changing too much between periods. The OMXS30 index is the most traded index out of all that are listed by Nasdaq Nordiq [14].

The overnight log return of the OMXS30 index will act as our response variable in this analysis. In total daily OHLCV (Open, High, Low, Close and Volume [18]) values for the period 2010-01-01 to 2020-01-01 have been retrieved. The daily return of a stock can be divided into two parts, the overnight return and the intraday return. The intraday return is the return, i.e. closing price in relation to opening price, during the trading day when

the stock market is open. The overnight return is the return when the stock market is closed, i.e. the opening price in relation to the closing price the day before [11]. The closing prices for both the response as well as the predictor variables have been adjusted for corporate actions (Adjusted Closing Price). The Swedish stock market is open from 09:00 - 17:30 CET. At 08:00 CET orders can be placed and at 08:45 CET the stock market starts to collect and summarize all the buy and sell orders resulting in an equilibrium price for each stock which can be thought of as a potential opening price of the stock. At 09:00 the stock market opens and the price is set [17]. The Adjusted Closing Price is calculated at 17:30 the same day. In order to get the overnight log return of the OMXS30 stock index day t the formula is

$$\text{Overnight}(\text{OMXS30}, t) = \log(\text{Open}, t) - \log(\text{Adjusted closing price}, t-1). \tag{28}$$

In order to turn this in to a binary classification task the overnight log return of the response variable OMXS30 will be divided into two groups. If the overnight log return is negative the value of that observation will be 0 and otherwise it is set to 1. The dataset does not take into account the magnitude of the stock price movement but only if it is negative or not.

3.2 Predictor variables

The daily OHLCV values for the same time period 2010-01-01 to 2020-01-01 as the response variable were gathered for twelve different parameters. Six of them are the log intraday or overnight returns of different stock market indices around the world, where the overnight/intraday log return depends on their respective opening hours. The stock market indices in North America (New York City and Toronto) are open from 15:30 - 22:00 CET and hence "yesterdays" intraday log return will be used as a predictive value for "todays" OMXS30 opening value. The stock market indices in Shanghai and Hong Kong both open at 03:30 CET and close at the same time or after the Swedish stock market opens. Since we want all of the predictor variables to have values set before our response variable value is determined, the overnight log return will be used for both the index in Shanghai and the index in Hong Kong. The Tokyo stock market opens at 02:00 CET and closes at 08:00 CET (an hour before the Swedish stock market opens) and as of such the intraday log return value will be used for the Nikkei 225 Index (Tokyo) [5]. The six stock indices with their corresponding Intraday/Overnight returns used as predictor values of Overnight(OMSX30,t) are summarized in Table 2.

Table 2: Six stock indices, used as predictor values of Overnight(OMSX30,t)

City	Variable	Intraday/Overnight	Day
Hong Kong	Hang Seng Index	Overnight	t
Shanghai	SSE Composite Index	Overnight	t
Toronto	SP/TSX Composite Index	Intraday	(t-1)
New York City	S&P 500	Intraday	(t-1)
New York City	Nasdaq Composite Index	Intraday	(t-1)
Tokyo	Nikkei 225 Index	Intraday	t

Figure 2: Opening hours of certain stock markets (Dag = Day).

The intraday/overnight variables, defined in Table 2, are calculated by:

$$\begin{aligned}
\text{Intraday, day}(t) &= log(Adj.Close, t) - log(Open, t) \\
\text{Intraday, day}(t-1) &= log(Adj.Close, t-1) - log(Open, t-1) \\
\text{Overnight, day}(t) &= log(Open, t) - log(Adj.Close, t-1)
\end{aligned}$$

Also included as explanatory variables were some commodities and currencies, represented in Table 3. Since the opening and closing prices for these variables are not as well defined, the important thing was to make sure that no value would be used that is the result of trading activity after the Swedish stock market has opened (by that time the value of our response OMXS30 has already been set).

Table 3: Variables, whose log returns are used as predictor variables of Overnight(OMXS30,t)

Variable	log return day
Gold	Adj.Close(t-1) - Adj.close(t-2)
Silver	Adj.Close(t-1) - Adj.Close(t-2)
EURSEK	Adj.Close(t-1) - Open(t-1)
USDSEK	Adj.Close(t-1) - Open(t-1)
Crude oil	Adj.Close(t) - Open(t)
Natural gas	Adj.Close(t) - Open(t)

3.3 Total dataset

Since the different stock markets have different deviating opening hours, some NA values were present in the data. After removing all NA values and creating a lag (t+1) for some parameter values, we obtained a total data set of 2083 observations between the period of 2010-01-01 to 2020-01-01. The rows were randomized, before splitting the observations into training data and test data, in order to eliminate any time trends. After re-sampling the rows to randomize the order the data was listed, we used a 80/20 percent split, where the training data and test data represent 80% and 20% respectively of the total data set. The distribution of each dataset's response variable values are represented in Table 4.

Table 4: Each dataset's distribution of the response variable values

OMXS30	0 (stock price down)	1 (stock price up)
Total data	877	1206
Train data	697	969
Test data	180	237

3.4 Correlation variables in training data

The correlation matrix is represented in the appendix and shows that some of the explanatory variables are correlated. The S&P 500 Index and Nasdaq Composite Index are very correlated (correlation coefficient 0.95612904) which is adequate since they are both indices of the US stock market, have the same opening hours and mainly because the indices actually contain many of the same stocks. The gold and silver variables show signs of correlation and both of the currencies USDSEK and EURSEK are somewhat correlated (correlation coefficient 0.59). We also notice that the Nasdaq

Composite Index is more correlated with Nikkei 225 Index (correlation coefficient 0.47) than with SSE Composite Index (correlation coefficient 0.22).

4 Modeling

4.1 GBM

We tuned the GBM-model using the values in the hyperparameter space presented in section 2.5.2. The ROC was used to select the optimal model using the largest value of the area under the curve (AUC). The largest value of AUC = 0.7430 was with the corresponding parameter values:

Table 5: Optimal values of the hyperparameters of GBM.

interaction.depth	2
n.trees	1800
shrinkage	0.003
n.minobsinnode	20

4.1.1 Relative importance variables

We run the GBM-model using 10-fold cross validation on the training data with the parameters presented in Table 5. Figure 3 visualizes the relative importance, described in section 2.4.1, of the parameters in the model built on the training data. An interesting observation in the relative influence plot of Figure 3 is that the overnight return of the Hang Seng Index has more relative influence than the intraday return of the Nikkei 225 Index. The overnight log return of the Hang Seng Index is calculated when the Hong Kong stock market opens at 03:30 CET, while the intraday log return of the Nikkei 225 Index is calculated at 08:00 CET, only one hour before the Swedish stock market opens and the value of the response variable OMXS30 is set. We see that the Shanghai SSE Composite Index and the Toronto SP/TSX Composite Index have the least relative influence among the predictor variables when classifying an observation.

4.1.2 Optimal number of boosting trees

When we built the model we used 10-fold cross validation which in practice means that we fit 10 different GBM models before computing the cross validation error estimate $CV(\hat{f})$ and then fitting the 11th and final model with $n.trees$ iterations using all of the data [15]. The $gbm.perf$ graph of Figure 4 shows the cross validation error together with the number of iterations. We see from the graph that the the cross validation error stops improving after $n.trees = 1727$. When using our fitted GBM model to predict the

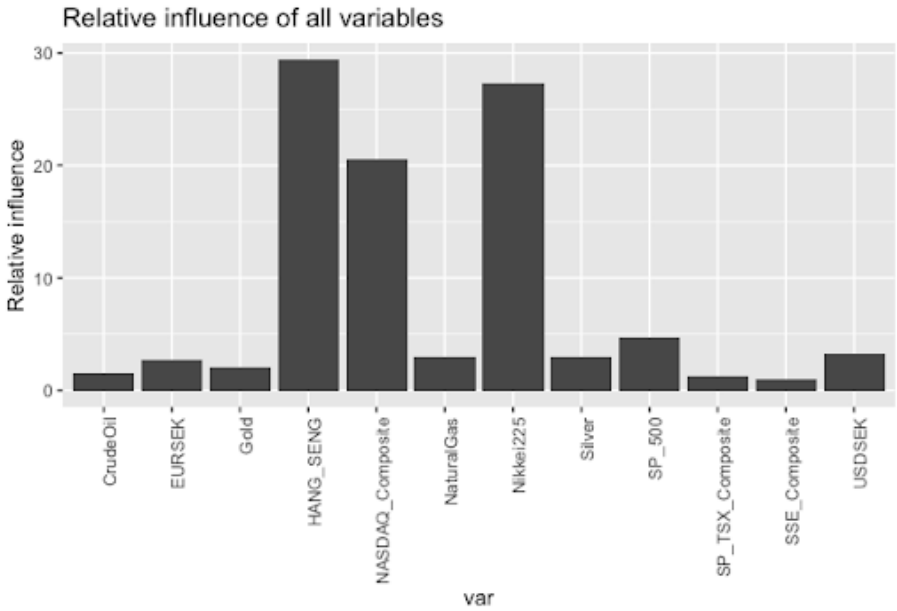

Figure 3: Relative influence of explanatory variables for GBM

new unseen test data we will choose this value 1727 of the hyperparameter *n.trees*.

4.2 Random Forest

4.2.1 Parameter Tuning

We set the number of trees to 5000 and evaluate how the out-of-bag estimate of the error rate changes with every added tree, see Figure 5 . The red line shows the error rate when classifying negative log returns and the green line shows the error rate for positive log returns. The blue line is the out of bag error rate. We see that the error rates have stabilized and thus we do not need to try a model with more trees than 5000 to obtain lower OOB-error rates. In fact, it seems that even 1500 trees would be sufficient.

We continue by evaluating the optimal number of variables tried at each split to ensure that the best possible value is chosen within our parameter range. We evaluate the parameter value space $mtry = 1, ...11$ and see in Figure 6 that the lowest OOB-rate is obtained when then number of variables each split equals 2. Thus we choose the value 2 of the hyperparameter *mtry* in our final model.

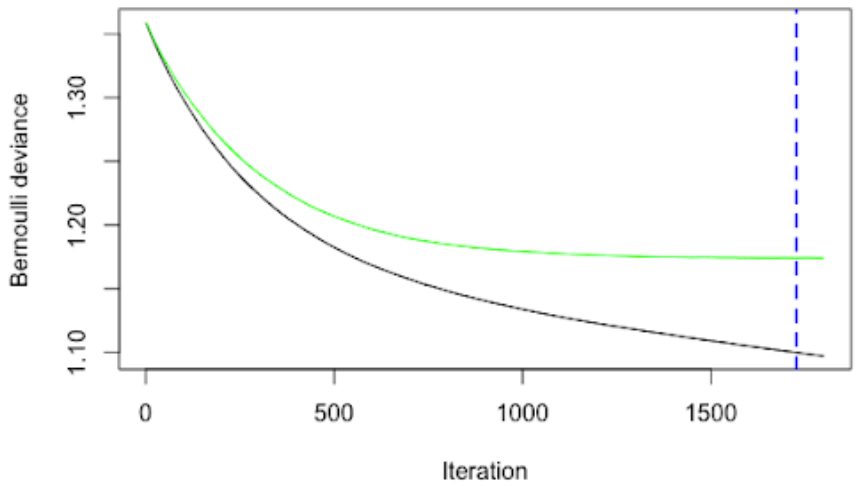

Figure 4: Cross-validation error (green) and prediction error (black) on training data as a function of the number of boosting iterations M. The optimal choice of M is illustrated with the vertical dashed line.

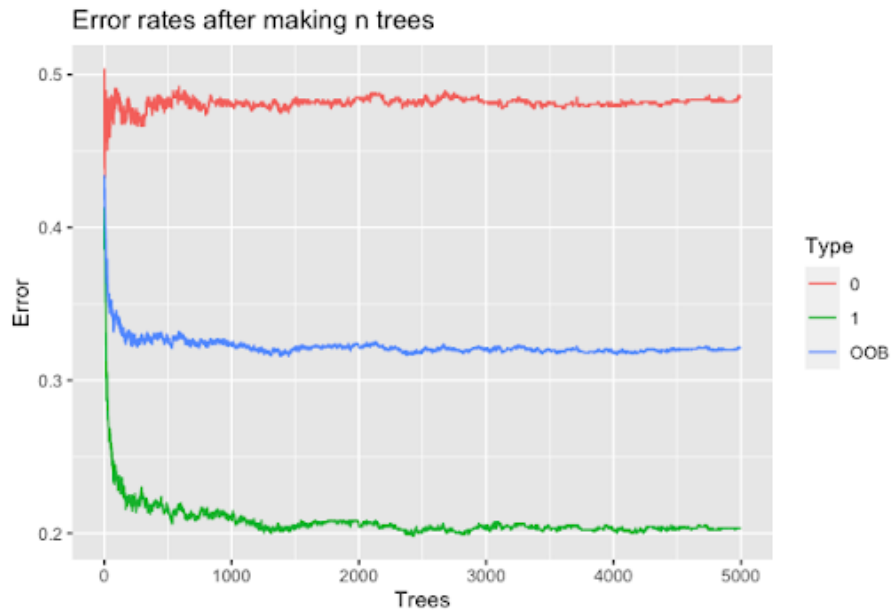

Figure 5: Error rates of random forest as a function of the number of resampled trees B, for negative returns (red), positive returns (green) and the whole out-of-bag (OOB) sample (blue).

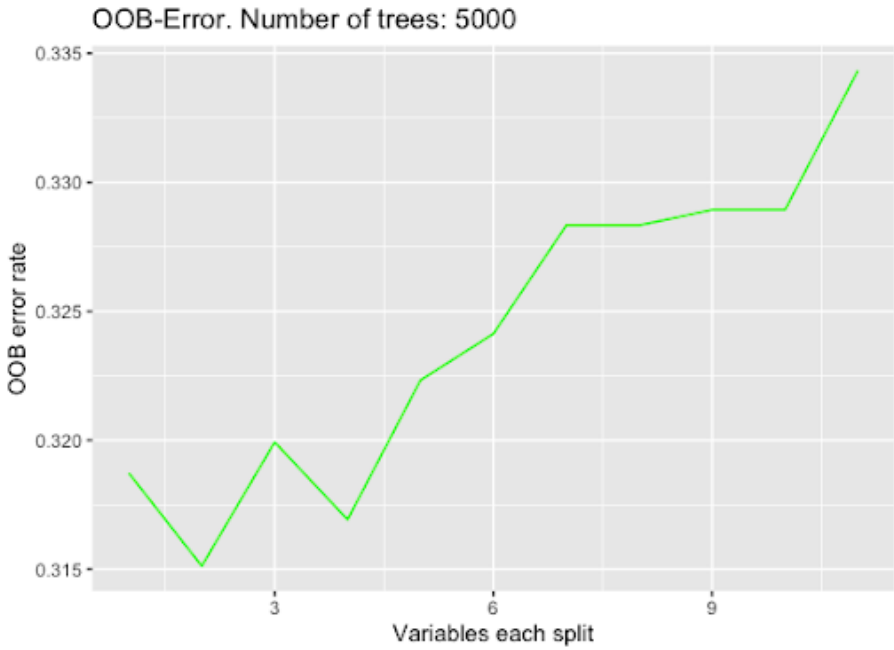

Figure 6: The out-of-bag (OOB) error of random forest, as a function of the number of predictor variables used in each split of the resampled trees.

4.2.2 Variable Importance

In the same way as for GBM, the relative influence of the variables in predicting the class of an observation is visualized in Figure 7.

5 Result

5.1 GBM Prediction

When using using our built GBM model on the new unseen test data the model generated a prediction accuracy of 0.6954 in successful classifications of the observations in the test data. The confusion matrix of Figure 8 shows the number of true positives (TP, lower right), false positives (FP, upper right), false negatives (FN, lower left), and true negatives (TN, upper left).

5.2 Random Forest Prediction

The Random Forest model generated a prediction accuracy of 0.6906 in successful classifications of observations in the test data. See the confusion matrix of Figure 9.

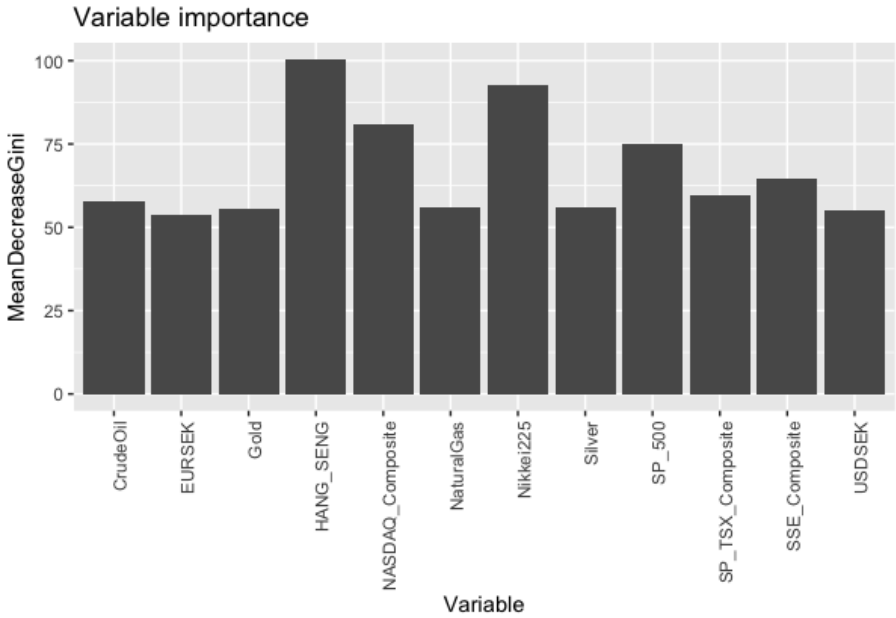

Figure 7: Variable importance of the predictor variables of random forest.

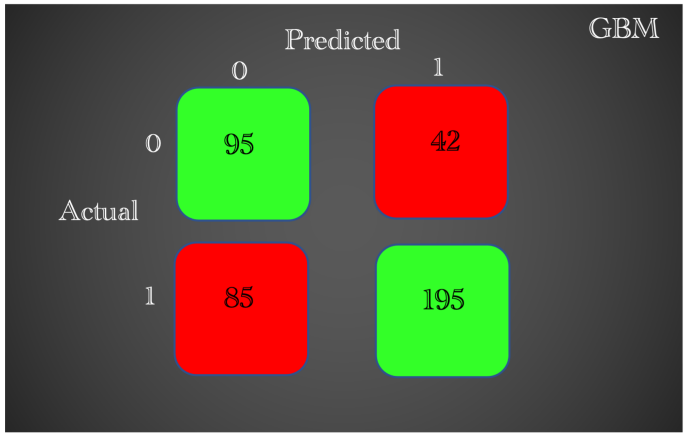

Figure 8: The confusion matrix of Gradient Boosting Machine (GBM).

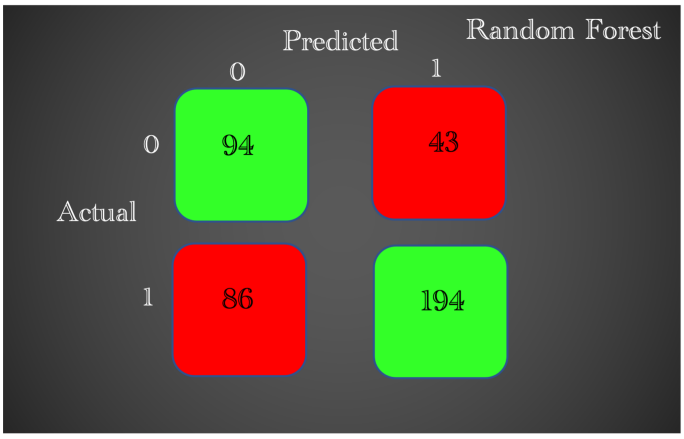

Figure 9: The confusion matrix of random forest.

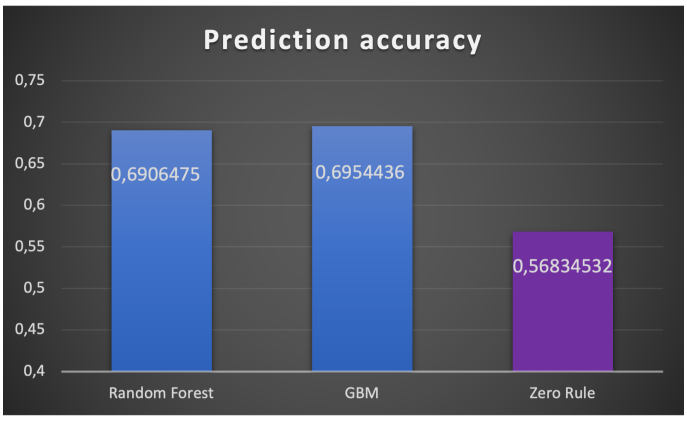

Figure 10: Prediction accuracies on test data of random forest, Gradient Boosting Machine (GBM) and the Zero Rule.

5.3 Accuracy

The histogram of Figure 10 shows the prediction accuracies of GBM, random forest and the Zero Rule, that was introduced in section 2.5.3.

We see that the difference in performance between the two machine learning methods is minimal, where the GBM model succeeded in predicting one more TP and one more TN than the random forest model, i.e. two more successful classifications out of a sample of 417 observations. Both of these methods outperformed the Zero Rule.

6 Discussion

We can see from the result in section 5.3 that the GBM-model outperformed, even if only by a little, the random forest model in number of successful classifications. There are however other factors that must be mentioned when evaluating the overall model performance. The computational cost is one of them, i.e. how much time it takes to train the model. Training the GBM-model was far more time consuming and even when we restricted the parameter space for GBM the difference in computational cost between tuning the GBM model compared to the random forest model was clear. The choice of which model to use should be data- and task specific. Data of more sensitive nature should strive for the highest prediction accuracy and could hence justify training a model with more computational cost.

It should be mentioned that there exists a newer method for GBM in R that is called XGBoost. XGboost is short for eXtreme Gradient Boosting, and is an scalable and efficient implementation of the GBM-model described in section 2.6.2. This package includes efficient linear model solver, tree learning algorithm, it supports classification (amongst others) and is generally over 10 times faster than GBM [3]. It would be interesting, in addition to the two models compared in this paper, to include XGBoost in the analysis and compare it to the other models in both speed and accuracy. Another addition to the analysis would be working on a similar dataset but containing more observations. Extending the time-period of the different variable returns could possibly lead to better predictions and more difference between the prediction accuracies of the models. It would however also lead to more computational time especially for GBM.

When converting the predictions generated by the gradient boosting model to binary values, we used a threshold of 0.5 where values over 0.5 were turned to 1 and 0 otherwise. In a test data set of 417 observations 107 of them were in the range 0.4-0.6, 27 of them were between 0.45 and 0.5 and 22 were between 0.5 and 0.55. This shows the sensitivity in setting the value of the threshold and small changes could cause a dramatically different result.

The result shows that both of the models were more successful in predicting true positives than true negatives. A possible explanation could be that the models were not sufficiently trained in classifying negative response values, since the training data consisted of approximately 58% positives and the rest negatives. The response variable, before being turned to a binary value, had both negative and positive log returns. It would be interesting to do a similar analysis but without converting the response variable into a binary type and thus leaving it as numerical. Since the relationship between log returns of different stocks, indices and currencies naturally is of

linear nature we lose information about the data when transforming the log returns of the response to binary variables. A possible analysis would be to fit a ordinary linear regression for the log return data before converting the fitted estimators to 0 or 1 depending on whether the log return prediction is negative or non-negative. It would be interesting to compare the prediction accuracy using that approach with the method performed in this paper. Furthermore, since the stock movement can be both large and small, it would be interesting to compare the models and how well they perform on a multiclass classification problem, i.e. assign a stock movement to a class depending on the size of the positive/negative movement.

7 Appendix

This appendix consists of a plot referenced in section 3.1.3 as well as a link to the code used in the analysis.

Correlation plot of explanatory variables 11.

Code used in Section 3, 4 and 5: https://github.com/Filip-Bergkvist/Code/blob/main/BachelorThesis.Rmd

8 References

References

[1] Breiman, L. (2001). *Random Forests*, *Machine Learning* 45(1), 5-32. https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

[2] Breiman, L. (2002). Manual On Setting Up, Using, And Understanding Random Forests V3.1. https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf

[3] Chen, T. & He, T. (2022). xgboost: eXtreme Gradient Boosting. https://cran.r-project.org/web/packages/xgboost/vignettes/xgboost.pdf

[4] FAWCETT, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters. 27(8), 861-874*

[5] Finansportalen. Börsöppettider. https://www.finansportalen.se/borsoppettider/ Accessed on 2022-02-10

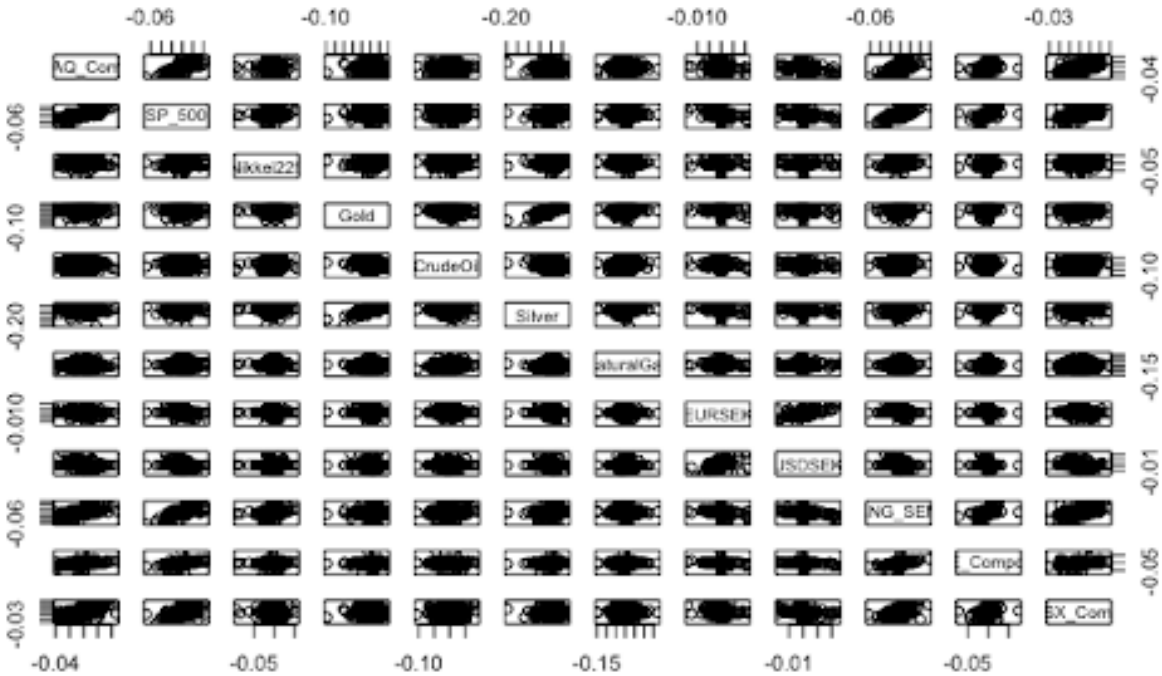

Figure 11: Correlation plots between the explanatory variables in the training data.

[6] Gutzeit, C. & Lubkowitz, M. Zero Rule.
https://machinelearningcatalogue.com/algorithm/alg_zero-rule.html
Accessed on 2022-05-10

[7] Hanley J.A. & McNeil, B.J. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve, *Radiology 143: 29-36, April 1982.* https://pubs.rsna.org/doi/epdf/10.1148/radiology.143.1.7063747

[8] Hastie, T., Tibshirani R. & Friedman, J. (2017). *The Elements of Statistical Learning*. Second edition, Springer Series in Statistics Springer New York Inc., New York, NY.

[9] IG Markets. Vad är CFD-trading och hur fungerar den?
https://www.ig.com/se/cfd-trading/vad-ar-cfd-trading-hur-fungerar-den
Accessed on 2022-05-10

[10] IG Markets. Helghandel.
https://www.ig.com/se/helghandel
Accessed on 2022-05-10

[11] Kenton, W.(2022). Intraday Return.
https://www.investopedia.com/terms/i/intraday-return.asp
Accessed on 2022-04-10

[12] Kuhn, M. et al. (2022). Classification and Regression Training.
https://cran.r-project.org/web/packages/caret/caret.pdf

[13] Li, K. (2022). Predicting Stock Prices Using Machine Learning.
https://neptune.ai/blog/predicting-stock-prices-using-machine-learning
Accessed on 2022-05-15

[14] Nasdaq, Inc. Vad är OMX Stockholm 30 index?
http://www.nasdaqomxnordic.com/utbildning/optionerochterminer/vadaromxstockholm30index
Accessed on 2022-05-10

[15] Ridgeway, G. (2020). Generalized Boosted Models: A guide to the gbm package. https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf

[16] Rokach, L. & Maimon, O. Decision Trees. Department of Industrial Engineering, Tel-Aviv University, https://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf

[17] Swedbank AB
https://www.swedbank.se/privat/spara-och-placera/aktier/aktier-tips-aktieskolan/aktier-nyborjare.html
Accessed on 2022-04-20

[18] yahoo finance
https://finance.yahoo.com
Accessed on 2022-02-20