Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Markus Skagemo

Stacking classifiers for improved order execution

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Adil Rasheed
Co-supervisor: Per Christian Moan, Ph.D.
June 2022

NTNU
Norwegian University of
Science and Technology

Markus Skagemo

Stacking classifiers for improved order execution

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Adil Rasheed
Co-supervisor: Per Christian Moan, Ph.D.
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

Long time horizon trading strategies do not necessarily consider short-term price movements in the period shortly before and after orders are placed. In this thesis, we introduce a framework for finding the most profitable time to execute an order, allowing a time delay in the order of days. We consider the opportunity cost of delaying orders and estimate execution parameters by modeling the total increase in risk-adjusted return. The order execution model is based on a novel adaptation of the stacking classifier ensemble method that incorporates purged K-fold cross-validation. As regular cross-validation causes data leakage when financial time series are used, our approach allows a stacking classifier to be trained without over-exaggerated validation fold performances. Our stacking classifier implementation uses the prediction probabilities of a LightGBM and a random forest classifier as inputs into a logistic regression model. We show that the execution strategy substantially outperforms the original trading strategy when accounting for trading costs, and that it reduces the total number of orders. This implies that the execution strategy can compress trading strategies by lowering trading costs while sustaining or increasing their risk-adjusted return.

Sammendrag

Langsiktige handelsstrategier tar ikke nødvendigvis hensyn til kortsiktige prisbevegelser kort tid før og etter markedsordre er plassert. I denne oppgaven introduserer vi et rammeverk for å finne det mest lønnsomme tidspunktet for ordreutførelse, som tillater en tidsforsinkelse i ordreutførelse. Vi tar hensyn til tap i profitt assosiert ved å forsinke ordrer, og estimerer strategiparametere ved å modellere den totale økningen i risikojustert avkastning. Ordreutførelsesmodellen er basert på en ny tilpasning av stablingsklassifiserings-metoden inkludert renset K-fold kryssvalidering. Siden vanlig kryssvalidering forårsaker datalekkasje når finansielle tidsserier brukes, tillater vår tilnærming at en stablingsklassifiserer kan trenes uten overvurdert valideringsfold-ytelse. Vår stablingsklassifiseringsimplementering bruker prediksjonssannsynlighetene til en LightGBM og en tilfeldig skogklassifiserer som input til en logistisk regresjonsmodell. Vi viser at utførelsesstrategien overpresterer den opprinnelige handelsstrategien vesentlig når det tas hensyn til handelskostnader, og at den reduserer det totale antallet ordre. Dette antyder at utførelsesstrategien er i stand til å komprimere handelsstrategier, slik at mindre handelskostnader påløper, samtidig som den opprettholder eller øker den risikojusterte avkastningen til handelsstrategien.

Preface

This thesis is a result of my work in the last six months in collaboration with *Intelligent Trading AS*, aiming to improve the execution of long-term time horizon trading strategies. I would like to thank my supervisor, professor Adil Rasheed, and my co-supervisor, Per Christian Moan, Ph.D., for invaluable guidance in this process.

Note that my master's thesis is a continuation of progress made in my project thesis, Skagemo et al. (2021). Let this paragraph be a disclaimer that theory subsections 2.1, 2.2.1, 2.2.3, 2.4.1, 2.4.2 and 2.5.1 are direct para-phrasings of Skagemo et al. (2021). This is also true for the first paragraph of Section 3.1.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

Given a list of market orders comprising a long time horizon trading strategy, we wish to find the optimal market execution time within an allowed period of E days. Orders of the trading strategy have targeted holding durations of over a month and are the result of concrete trading rules resulting from a transparent, non-machine-learning-based trading model. The trading strategy might act on recent or real-time information but does not consider price drift in the period directly after order execution. We augment the trading strategy with a short-term pricing model such that the price is less likely to move in a direction detrimental to orders shortly after they are executed. Consider an example where a long-term valuation estimates the price of an asset to increase substantially within a year. However, a recent news event forces the entire market to fall briefly. A short-term execution model might avoid buying the asset until the market fall has subsided, resulting in increased total profit after the trade is liquidated.

1.2 Literature study

Many papers consider the problem of optimal execution of market orders, focusing on approaches that utilize market microstructure for execution of block trades, where all block trades compose the market order (Almgren and Chriss (2001) and Bertsimas and Lo (1998)).
In Bertsimas and Lo (1998), the authors list the different sources of trading costs, including commissions, market impact, bid/ask spreads, and the opportunity cost of waiting. This thesis measures this opportunity cost and devises a framework that estimates the profit-maximizing waiting duration over a period of days. Analogous to Bertsimas and Lo, who specify that $\mathcal{S}$ shares are to be executed within a time period $\mathcal{T}$, we execute an arbitrary amount of shares within E days. Note that our papers operate with very different time frames, as $\mathcal{T}$ represents a number of 30-minute chunks, and a sequence of block trades are executed optimally with

respect to an intra-day market impact model. In contrast, our setting is market data at daily frequencies, and the market impact is assumed to be negligible for the portfolios on which the models are meant.

Optimal execution papers also do not assume an end time frame of the orders they execute (Almgren and Chriss (2001) and Leal et al. (2020)). We specify a long time horizon on trades, expecting most price movement to not occur within the first days of execution, which allows for a longer execution window. Therefore, this thesis should not be considered an addition to traditional optimal order execution models but rather a suggestion of how short-term information can augment long-term investments.

De Prado (2018) describes how black-box machine learning algorithms (ML) have massive learning potential but suffer from being opaque, causing human practitioners to fail to fulfill their generalization abilities while being overconfident in their resulting badly specified models. He argues that ML models should work as an additional layer to other theoretically sound models, governing only a single task in the investment process, such as order sizing. Labels are then generated from the underlying model with respect to this task, and an ML model is trained on these labels. De Prado coins this approach "meta-labeling". In our case, we let the learning task be to determine the best time of execution of a theoretically sound trading strategy, generating so-called meta-labels from future short-term price movements.

Various machine learning approaches have been attempted in finance, with varying levels of success. Arguably the staple of modern machine learning research, neural networks are documented to be effective in a variety of real-life applications (Brown et al. (2020), Flagel et al. (2019) and Jumper et al. (2021)), and are proven to be universal function approximators, having the ability to learn any learning set distribution (Sonoda and Murata (2017)). Naturally, many finance researchers find it tempting to use neural networks to predict asset returns, and some have been successful (Gu et al. (2020) and Krauss et al. (2017)). However, the decisions of neural networks suffer from being very complex. Eschenbach (2021) finds that understanding the inner workings of such networks is beyond human comprehension, which causes distrust in their effectiveness. Tree-based models, however, output results that are easier for humans to understand, as they rely on concrete rule-based tree structures. Gu et al. (2020) performs a comparative analysis of a variety of ML algorithms, showing that decision tree ensembles are the most effective models for asset pricing. The combination of several machine learning models also shows promise, with Jiang et al. (2020) employing a stacking classifier based on the predictions of tree based ensemble methods (XGBoost and LightGBM) and deep learning techniques (LSTM and GRU), showing that a stacked combination of these classifiers display a higher accuracy, AUC and F1-score than each of the classifiers by themselves. Stacking, of Wolpert (1992) is effective for combining the outputs of several different classifiers into a single strong classifier and uses a cross-validation approach to maximize the generalization of the resulting model. Standard K-fold cross-validation, used in the stack-

ing classifier algorithm, assumes that samples in training and validation folds are independent. Finance data, or features of data sets generated with rolling windows, are self-similar over time; therefore, this assumption can not be guaranteed. De Prado (2018) modifies regular K-fold cross-validation to ensure no overlap between train- and validation-folds, using methods he names purging and embargoing. In this thesis, we combine the stacking ensemble classifier with purged K-fold cross-validation, resulting in a stacking classifier that is applicable to finance data. This approach is, to our knowledge, a novel contribution to the field of empirical finance.

1.3 Thesis outline

The thesis is structured as follows. The background chapter, Chapter 2, introduces techniques that are prerequisites for the methods described in Chapter 3 for constructing an execution strategy. The strategy is implemented, and empirical results are listed and displayed in Chapter 4. In Chapter 5 we discuss a few of our major findings from the empirical study, then conclude by summarizing the advancements made in the thesis, as well as discussing future work in Chapter 6.

1.4 Research objectives and research questions

1.4.1 Primary and secondary objectives

The primary objective of the thesis is to develop a framework for improving execution of long-term time horizon trading strategies. The secondary objective is to verify that the execution strategy framework is effective on several different trading strategies.

1.4.2 Research questions

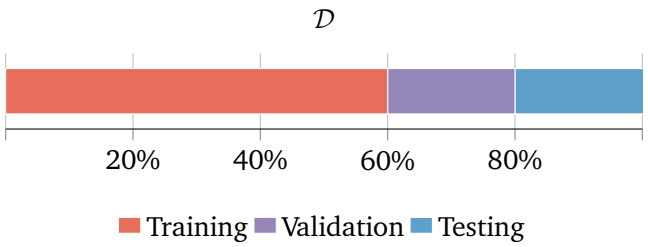
- Can we adapt a stacking ensemble method for financial time-series, and use it to train a classifier that predicts future price direction?
- Can several of the key risk factors of the execution strategy be identified?

Chapter 2

Background

2.1 Data set splitting

When learning the parameters of a model, one should never expose the model to the entire data set. This is due to many models' tendencies to learn the noise of the data when given enough time. Instead, we split the data set, $\mathcal{D}$, into three different independent chunks, called the training-set, validation-set, and test-set, visualized as



The training set is used for evaluating different model parameters. After using the train set to train the model on in-sample data, we use the validation set to estimate how well the model performs on out-of-sample data. We differentiate *estimating-* from *validating-* how well the model performs out of sample because we cannot regard the validation-set as an unseen data set, as the model is continuously evaluated against it, causing implicit data leakage, as described in Kaufman et al. (2011). Data leakage is when data intended to be unseen is revealed as a result of methodological error. After experiments are run and model parameters are found, the test-set is used once to check if the model is functioning. Train, test, and validation splits are experiment-dependent, but train sets generally comprise most of the full data set.

2.1.1 Cross validation

Cross-validation is a widely used method of validating the performance of models with overfitting risk. Instead of training a model on the entire training set (see

Section 2.1), a cross-validation approach would separate a validation set from the training set so that model performance may be estimated without exposing the test set. Performing the separation procedure multiple times, when aggregated, tends to estimate how well the model has generalized to unseen data. The separation procedure can be random sampling or chronologically extracting parts of the training set. The number of times this is done is typically determined by the parameter K, which explains the naming of the most frequently used cross-validation approach; K-fold cross-validation (Stone, 1974). Figure 2.1a illustrates how a cross-validation with $K = 4$ splits the data set.

2.1.2 Purged K-fold cross validation

By using self-similar data features (auto- or cross-correlation), such as ones computed on a rolling basis, one risks introducing future information into a model if typical K-fold cross-validation is used. De Prado (2018) details the concept of purging, forcing the most recent sample in a training set to be T samples previous to the first validation set sample. This measure removes data set leakage if the amount of self-similarity is correctly estimated.
Notice how a purging period of $T = 20$ units creates a margin between the validation set and the training set in Figure 2.1b, compared to normal K-fold CV in Figure 2.1a.

2.2 Data transformation

Data transformation is used to ensure that data conforms to a given structure. Many predictive models assume that input data is normally distributed, of a specific type, or within a pre-set range. Rigorous statistical theory operates with assumptions of properties such as homoskedasticity and stationarity. Using data transformation, we can standardize a wide variety of data sources and even compare data with entirely different structures.

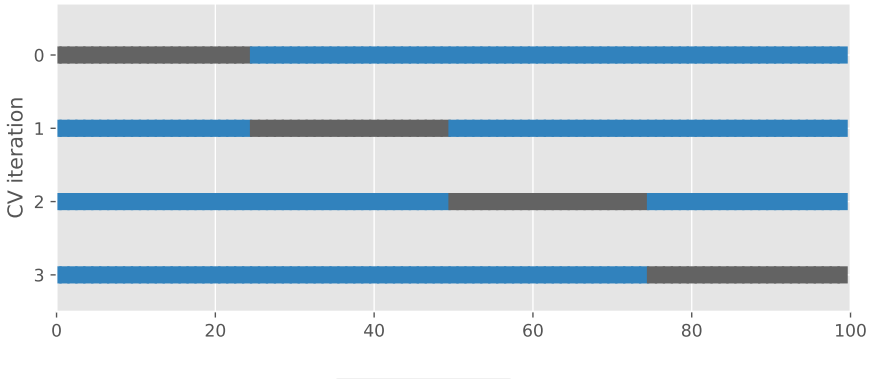
2.2.1 Standard scaling

Standard scaling transforms data to so that its mean and standard deviation is equal to 0 and 1, respectively. Letting $\mathcal{Z}(w)$ denote the standard scaling function, a standard scaling of a set of values $Y \in \mathbb{R}^w$ can be expressed as

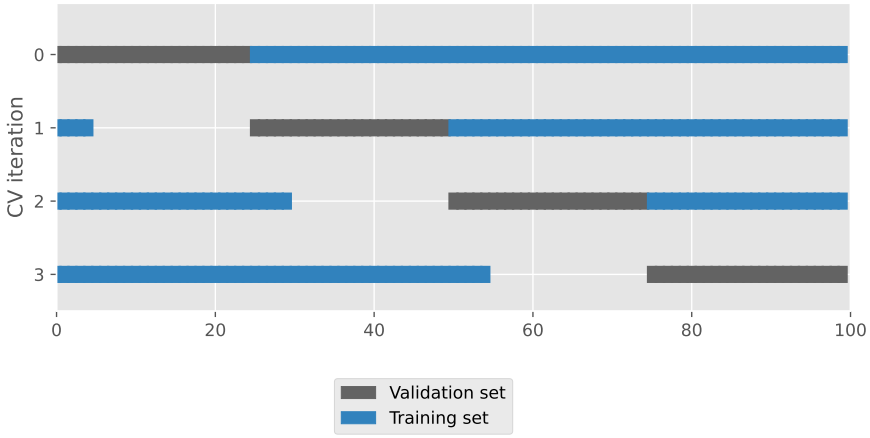
$$Y^z = \frac{Y - \mu_Y}{\sigma_Y}, \text{ where } \mu_Y = \frac{1}{w}\sum_{i=1}^{w} Y_i \text{ and} \tag{2.1}$$

$$\sigma_Y = \sqrt{\frac{1}{w}\sum_{i=1}^{w}(Y_i - \mu_Y)^2} \tag{2.2}$$

and $\mu_Y = [\mu_Y, \ldots, \mu_Y]$ in $\mathbb{R}^w$. Then Y^z is the standard scaling of Y.



(a) K-fold cross validation with K=4.



(b) Purged K-fold cross validation with $K = 4$ and $T = 20$.

Figure 2.1: Comparison between regular and purged CV.

2.2.2 Winsorizing

Winsorizing is the process of truncating extreme values on both tails of a data distribution. In contrast to clipping, where extreme values are dropped from the data set, winsorizing truncates them with the limit value consistent with the side of the distribution they lie on. A percentile $0 \leq p \leq 1$ is defined such that $1-p$ of the values of the data set are truncated (Tukey (1962)). See Figure 2.2 for an illustration of the effect of winsorization on a standard normal distributed data set, where $p = 0.95$ such that 5% of values are truncated.

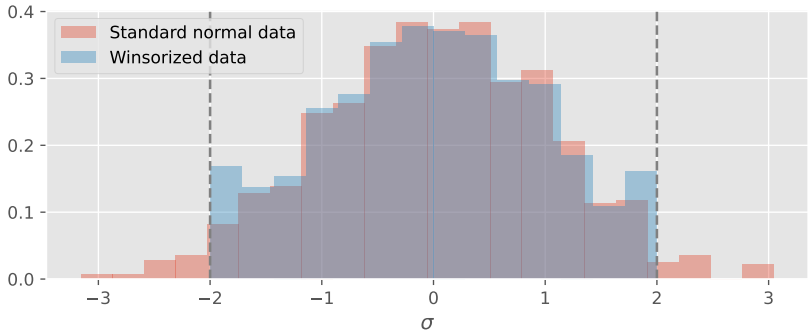


Figure 2.2: Distribution of standard normal data in comparison to the same data after winsorization. $N = 1000$, $p = 0.95$.

2.2.3 Exponential moving average

The exponentially weighted moving average (EWMA) is a way of computing a moving average over a series of values Y where the most recent data points have a much stronger influence over the output. EWMAs are widely used in finance as they are simple and computationally efficient, as well as have the ability to take new information into account.

Definition 1 *Given a real valued vector Y, where the rightmost values in the vector corresponds to the most recent observation, an exponentially weighted moving average for t, S_t, is defined as*

$$S_t = \begin{cases} Y_1, & t = 0 \\ \alpha Y_t + (1-\alpha)S_{t-1}, & t > 0 \end{cases} \tag{2.3}$$

where $0 < \alpha < 1$ is a smoothing factor that governs the decay of the weights in the EWMA.

Even though Definition 1 is in bracket style, Hunter (1986) shows that the current S also can be simply stated as

$$\begin{aligned} S_t = \alpha \big[& Y_t + (1-\alpha)Y_{t-1} + (1-\alpha)^2 Y_{t-2} + \cdots \\ & \cdots + (1-\alpha)^k Y_{t-k} \big] + (1-\alpha)^{k+1} S_{t-(k+1)} \end{aligned} \tag{2.4}$$

We can specify the decay factor $\alpha = 1 - \exp\left[\frac{ln(0.5)}{\lambda}\right]$, where λ is the halflife of the exponential decay, which can be a more interpretable way of quantifying decay. One can illustrate the exponential behaviour of the EWMA by expressing its weights as

$$\alpha \sum_{i=0}^{t-1} (1-\alpha)^i = \alpha \left[\frac{1-(1-\alpha)^t}{1-(1-\alpha)} \right] = 1-(1-\alpha)^t \tag{2.5}$$

showing that weights monotonously decrease. Notice that the weights in Equation (2.5) in the limit sum to

$$\lim_{t \to \infty} [1-(1-\alpha)^t] = 1 \tag{2.6}$$

2.3 Triple-barrier labelling

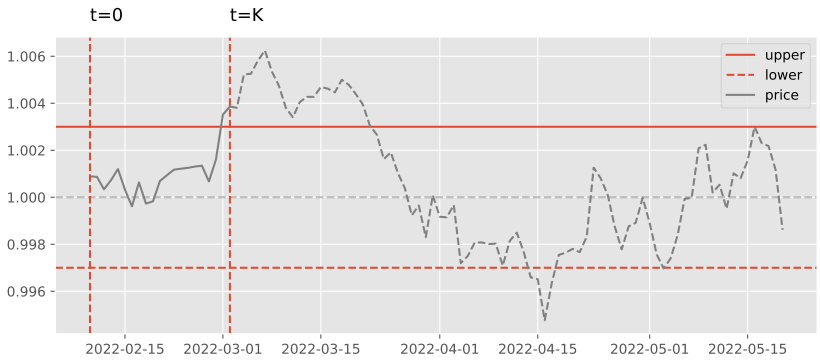


Figure 2.3: Example of positive label with triple barrier method, with $\pm 3\sigma$ as take-profit/stop-loss levels and $R_{\text{triple}} = 20$ as the right-barrier limit.

The most typical approach for labeling the direction of future price movements is picking a time in the future with respect to the labeling timestamp, then calculating the sign of the price change. This is called the fixed-time horizon method. De Prado (2018) explains how this approach likely fails to maximize profits, as it tends to generate plenty of buy/sell signals, each with very thin profit margins. De Prado introduces a labeling method with dynamic barriers, where labels are set if the price exits a range determined by the volatility of the asset and a predetermined maximum time duration, R_{triple}. Consider the synthetic example in Figure 2.3, where price change after $t = 0$ initially was negative, then subsequently reached the dynamic barrier three standard deviations above the starting point. This example would be more profitable when using triple-barrier labeling than fixed-time horizon labeling.

2.4 Regression

Regression is the task of predicting dependent numerical target values y given data points X (the regressors or the independent variables). The learning algorithm produces a function f, which maps $y = f(X) + \varepsilon$, put as a category as

$$X \xrightarrow{f} y$$

The prediction is associated with an error ε, which we aim to minimize with methods appropriate for the covariance structure of the regressors.

2.4.1 Linear regression

Linear regression is a method of parameter estimation where, given a set of endogenous variables $X \in \mathbb{R}^{p \times n}$, we wish to find the parameter vector β that is the solution to the optimization problem

$$\hat{\beta} = \arg\min_{\beta} ||y - X^\top \beta|| \tag{2.7}$$

where $y \in \mathbb{R}^n$, the dependent variable, is the target values of our optimization.

Definition 2 *The minimization in 2.7 can be rewritten in the general case to*

$$\hat{\beta} = \arg\min_{\beta} (y - X^\top \beta)^\top \Omega^{-1} (y - X^\top \beta) \tag{2.8}$$

where Ω is the covariance matrix of $y - X^\intercal \beta$, and is referred to as the error covariance matrix. Nelder and Wedderburn (1972) defines this formulation of the optimization problem as the generalized least squares (GLS).

For the linear regression solution β to be BLUE, the regression problem must satisfy the Gauss-Markov theorem. When Gauss-Markov is satisfied we can ensure that the error covariance is i.i.d with equal error variance σ^2 along the diagonal, such that $\Omega = \Omega^{-1} = \sigma^2 I_n$, causing the GLS to be equal to the ordinary least squares problem in 2.7.

Definition 3 *Letting Ω from Equation (2.8) be diagonal such that the errors are assumed i.i.d, we define this new error covariance matrix as W. Minimization of the reformulated objective function*

$$\begin{aligned} \hat{\beta} &= \arg\min_{\beta} \|W^{\frac{1}{2}} \left(y - X^\top \beta\right)\| \\ &= (X^\top W X)^{-1} X^\top W y \end{aligned} \tag{2.9}$$

yields the solution to the weighted least squares (WLS), a special case of the GLS from Definition 2.

2.4.2 Cross-sectional regression

Table 2.1: Structure of a design matrix, X, in finance. The matrix defines relationships between factors, β, and asset names a, quantified by their weights.

	β_1	$\dots$	β_p
a_1	X_{11}	$\cdots$	X_{1p}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_n	X_{n1}	$\cdots$	X_{np}

Where regular regression applications typically consider data across time, cross-sectional regression describes how the dependent variable, y_t, for a single time-step distributes into the p factors describing y_t. In practice, each column of X describes how a row of the dependent variable is influenced by one or several factors. See Table 2.1 for an example of a typical structure of X in finance, first described by Rosenberg (1974), which we can use to model the structure of values of assets a. A the cross-sectional regression models a_1 as

$$a_1 = \sum_{i=1}^{p} \beta_{1i} X_{1i} + \alpha_1 \tag{2.10}$$

As an example, let $x_{ij} \in \{0, 1\}$, $i = 1, 2, \dots, p$, $j = 1, 2, \dots, n$ and

$$X_1 = [x_{11}, x_{12}, ..., x_{1p}]^\top = \begin{cases} x_{1i} = 1, & i = 1, 3 \\ x_{1i} = 0, & otherwise \end{cases}$$

Then we can write out 2.10 as $a_1 = \beta_{11} + \beta_{13} + \alpha_1$, and the different βs can represent memberships of categories such as country and sector, and α will represent the market factor that is common in all assets. The chosen value of a_0 can then be simply deconstructed into $a_0 = \text{country}(a_0) + \text{sector}(a_0) + \text{market}$. This kind of reasoning is very prevalent in finance, - perhaps most famously used by Eugene Fama and Kenneth French in Fama and French (1992).

2.4.3 Logistic regression

Logistic regression is a binary classification method that seeks to estimate a set of parameters $\beta \in \mathbb{R}^p$ such that the logistic function

$$p(x) = \frac{1}{1 + e^{-\beta x}} \tag{2.11}$$

best estimates the probability that a sample $x \in \mathbb{R}^p$ belongs to a class of the Bernoulli distributed variable y taking the value of either 0 or 1. The class of a

prediction $\hat{y}$ is determined using the prediction probability $0 \leq p(x) \leq 1$ such that

$$\hat{y} = \begin{cases} 1, & p(x) > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

Let $L(\beta; Y, X)$ be the likelihood of the logistic regression parameters β with respect to the target data set $Y = \{y_0, \ldots, y_N\}$ and the feature data set $X = \{x_0, \ldots, x_N\}$. The expression of $L(\beta; Y, X)$ is as described in Kleinbaum et al. (2002), and expresses the probability of observing a constant X given varying parameters β. Since β is the argument of L, a maximization of L yields the parameters that best fits X. Then we introduce the parameter C, which corresponds to the reciprocal of the L^2-regularization strength, such that the parameters are the solution of

$$\min_{\beta} \log L(\beta; Y, X) + \frac{1}{C}\|\beta\|_2 \tag{2.13}$$

2.5 Constrained optimization

Constrained optimization is the problem of optimizing an objective function (equivalent to the energy function in control theory) with respect to a set of constraints, often used for imposing real-life limitations on the arguments of the objective function. Generally, a constrained optimization problem can be posed as

$$\begin{array}{rlll} \min & f(x) & & \\ \text{subject to} & g_i(x) = c_i & \text{for } i = 1, \ldots, m & \text{Equality constraints} \\ & h_j(x) \geq d_j & \text{for } j = 1, \ldots, \ell & \text{Inequality constraints} \end{array} \tag{2.14}$$

2.5.1 Karush-Kuhn-Tucker

For a general problem in constrained optimization as stated in Equation (2.14), where $c_i = 0 \ \forall\, i$ and $d_j = 0 \ \forall\, j$ and constraints are formulated as

$$g(x) = [g_1(x), \ldots, g_m(x)]^\top, \quad h(x) = [h_1(x), \ldots, h_\ell(x)]^\top \tag{2.15}$$

The Lagrangian function $\mathcal{L}(x, \mu, \lambda)$, a weighted sum of the objective function and its constraints, is constructed in the form

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \mu^\top g(x) + \lambda^\top h(x) \tag{2.16}$$

Theorem 1 *If f, g, $h \in C^1(\{x^\star\})$, i.e. continuously differentiable at a point $x^\star \in \mathbb{R}^n$, and $x^\star$ is a local optimum, then there exists KKT-multipliers $\mu^\star \in \mathbb{R}^m$ and $\lambda^\star \in \mathbb{R}^\ell$ such that $J_f(x^\star) + J_g(x^\star)\mu^\star + J_g(x^\star)\lambda^\star = 0$, which minimizes Equation (2.14) if the following conditions are held.*

1. *$g(x^\star) \leq 0$ and $h(x^\star) = 0$ (Primal feasibility)*

2. $\mu^\star \geq 0$ *(Dual feasibility)*
3. $\mu^{\star\top} g(x^\star) = 0$ *(Complementary slackness)*

These conditions are not exclusive in producing an objective minimizing solution that satisfies the KKT conditions (Kuhn and Tucker (2014) and Nocedal and Wright (2006)). There are a variety of other constraint qualifications that ensures a constrained minimizer also satisfies the KKT conditions. Since this thesis only concerns convex objective functions and constraints, *Slater's condition* is the only additional constraint qualification that will be included in this section, stating the existence of a point $x^\star$ which satisfies Item 1, primal feasibility (Slater (2013)).

2.6 Decision tree classification

Decision tree classification involves methods that distinguish between two or more classes by using classification trees. Ensemble methods rely on multiple classification trees that are combined by either bootstrap aggregating or boosting to minimize classification model error.

2.6.1 Classification trees

A classification tree is a supervised learning algorithm that, when visualized, has the appearance of a tree with a root node, internal nodes, and leaf nodes (see Figure 2.4).

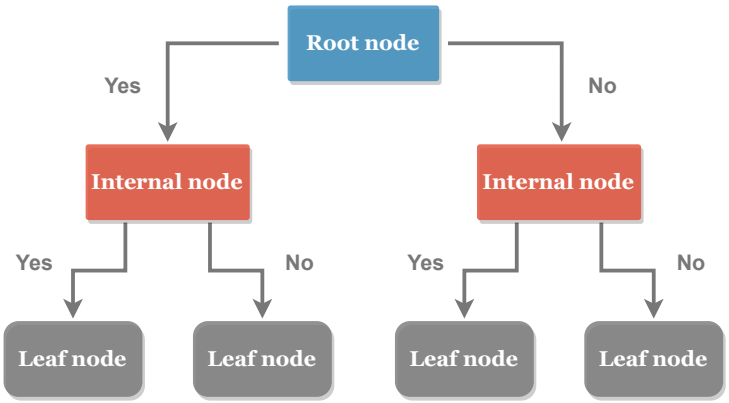


Figure 2.4: Structure of a decision tree with two levels. Decision trees can have an arbitrary number of levels of internal nodes before reaching leaf nodes.

Consider a binary classification problem where leaf nodes show classification probabilities $p = P(y = 1 \mid X)$. Then $P(y = 0 \mid X) = 1 - p$. Decision trees, of which classification trees are a special case, are constructed by splitting nodes at a feature value that best divides the data set. In other words, a split aims to maximize the difference between the sides of the split with respect to the two classes. Trees can be split based on a variety of criteria, including Gini impurity, estimate of positive

correctness or information gain (Rokach and Maimon (2005)). In this thesis we employ Gini impurity, which is described by

$$\mathrm{I}_G(p) = \sum_{i=1}^{P}\left(p_i \sum_{j\neq i} p_j\right) = \sum_{i=1}^{P} p_i(1-p_i), \tag{2.17}$$

for a data set containing P classes. For binary classification it is simply expressed as

$$\mathrm{I}_G(p) = 2p(1-p) \tag{2.18}$$

Intuitively, Gini impurity can be described as the probability of misclassifying a sample if given a random class label according to the distribution of the classes. A Gini impurity of zero would indicate that within a split, there would be a zero-probability of sampling a member of the wrong class. Therefore, each node is split at the point of the data feature that minimizes I_G. This means that a candidate's minimum Gini impurity has to be computed for each feature, and the tree is ultimately split on the argument of the minimum of all candidates.

2.6.2 Ensemble learning techniques

Ensemble learning techniques are used to increase the accuracy of predictions compared to the outputs of e.g., a single classification tree, which would have a high bias and low variance. In this thesis, we improve on this by employing the three following ensemble techniques.

Bagging

Bootstrap aggregating, or bagging, increases classification accuracy by taking the equal-weighted mean of classifications from a set of predictors trained on bootstrapped subsets of a training set $\mathcal{L} = \{(y_n, x_n)\}_{n=1}^{N}$. Following Breiman (1996), let y_n be a Bernoulli distributed target class and x_n be an input sample. A bagged predictor is created by random sampling M equal sized training sets $\{\mathcal{L}_m\}_{m=1}^{M}$ from $\mathcal{L}$ and growing a classification tree $\phi_m(x, \mathcal{L}_m)$ for each. The bagged classifier is then expressed as

$$\phi(x)_A = \frac{1}{M}\sum_{m=1}^{M}\phi_m(x) \tag{2.19}$$

The random forest classifier is essentially a result of this approach in combination with a random sampling of a subset of features of a randomly sampled training data set. This is done to reduce the emphasis on the most predictive features, such that correlation across trees is reduced (Ho (1995)).

Stacking

Wolpert (1992) introduces stacked classifiers as a method of reducing generalization error by using the outputs of a set of classifiers B_i, $i = 1, \ldots, M$ as the training

of for an additional classifier C_m, which we refer to as the meta-classifier. Then a classification of a data sample x is given by the composite function

$$\hat{y} = C_m(B_1(x), B_2(x), \ldots, B_M(x)) \tag{2.20}$$

Boosting

In the same manner as bagging, multiple weak predictors ϕ_m are formed from the original data set $\mathcal{L}$, but instead of using bootstrapped subsamples of $\mathcal{L}$, weak predictors are trained sequentially on the entirety of $\mathcal{L}$. Samples with high misclassification rates are then weighted higher than their counterparts, and a new weak classifier is trained by minimizing error with respect to the new weighting. This process is continued until the desired number of weak classifiers is reached. Then, the final strong classifier $\phi(x)_B$ is a linear combination of $\phi_1, \ldots, \phi_m$ (Bühlmann (2012)). The weighting method and loss function vary across boosting implementations, but in this thesis, we use the efficient gradient boosted decision tree algorithm *LightGBM* of Ke et al. (2017). *LightGBM*, or LGBM, lets each weak predictor learn the residuals of the former weak predictors and builds decision trees very efficiently by approximating the split criterion instead of computing it on all data instances. The authors show that LGBM trains much more efficiently than its high-performing peers, including *XGBoost* of Chen and Guestrin (2016) while only suffering a slight performance decrease. This makes LGBM suitable for large data sets with a large amount of features.

Chapter 3

Method

The execution strategy can be described as the sum of three components. The first is a multi-factor model, which we use for generating data features that we train our second component, a stacking classifier, with. The stacking classifier predicts the future direction of asset prices for all assets daily. These classifications are used in a final execution rule, the algorithm that we consider our third and final component of the execution strategy.

3.1 Pure factor model

For this thesis we consider a pure factor model with a structure similar to the Barra Global Equity Model (GEM2), implementing a range of style factors. A pure factor model is defined as a result of multivariate regressions that simultaneously consider all factors, as opposed to simple factor models that compute all factors isolated in a univariate fashion. Note that we wish to estimate the factors, which are unobserved. Recall the cross-sectional decomposition of an asset's returns from Equation 2.10, and consider the standard formulation of GEM2 as published in Menchero, Morozov et al. (2010), decomposing the returns r_k of an asset k for a point in time t into

$$r_k = 1 \cdot \beta_m + \sum_c X_{c,k}\beta_c + \sum_i X_{i,k}\beta_i + \sum_s X_{s,k}\beta_s + u_k, \quad k = 1, \ldots, M \tag{3.1}$$

where β_m is the market factor to which all assets are exposed to, β_c is a country factor, β_i is an industry factor, and β_s is a style factor. u_k is the regression error, which Menchero, Morozov et al. (2010) refers to as the idiosyncratic return, which is the return that can not be attributed by the factor model. We have a design matrix X in the form of Table 2.1. Following Heston and Rouwenhorst (1994) we impose the following equality constraints on the regression parameters.

$$g_{\text{countries}} = \sum_c w_c\beta_c = 0 \quad \text{and} \quad g_{\text{sectors}} = \sum_i w_i\beta_i = 0 \tag{3.2}$$

Heston and Rouwenhorst specifies that the weights w_c must sum to one and constrains the weighted sums of sectors and countries to sum to zero. The weighted sums are calculated by taking the sums of market capitalizations of all assets in a factor,

$$M_c = \sum_{k}^{\mathcal{A}_c} m_k, \tag{3.3}$$

where $\mathcal{A}_c$ is the set of all assets in factor c and m_k is the market cap of asset k. We then normalize these sums, yielding country constraint weights

$$w_c = \frac{M_c}{\sum_{\mathcal{C}} M_c}, \tag{3.4}$$

where $\mathcal{C}$ is the set of all country factors. Equivalently, we perform the same normalization as Equation (3.4) over the set $\mathcal{I}$, containing all sector factors.
Note that these constraints are not enforced on each sector or country by themselves but on sectors or countries as groups of factors. Therefore we cannot be sure that the constraint holds for a single country in isolation since there is no guarantee that assets in all countries are distributed over the same sectors with the same market cap distribution as the whole market.
Finally, the model imposes an indirect constraint on the style factor loadings, such that the market cap-weighted average factor exposure of each style is equal to zero. The specifics of how this standardization is performed are described in Section 3.1.2.

3.1.1 Factor loadings

Factor loadings, which are the scalar values contained within the exposure matrix X, are, in this implementation, the exponentially weighted beta values of each asset against the market cap-weighted average returns of the assets contained in the factor. This approach is similar to Shepard (2008), where for an arbitrary factor, the loading for an asset k on a factor i is given by

$$X_{k,i} = \frac{\mathrm{Cov}(r_k, R_i)}{\mathrm{Var}(R_i)} \tag{3.5}$$

where $R_i \in \mathbb{R}^T$ is the market cap weighted averages of the returns of the members of factor i, and $r_k \in \mathbb{R}^T$ is the T most recent log-returns of asset k.
As described by Shepard, the intuition behind this approach is based on the tendency of assets to be affected by the movements of their industries to varying degrees. When unit exposing every asset to a factor, the factor return will have an equal impact on the returns of each of the assets. In real life, one would expect this impact to vary based on, e.g., the company supply chain and sources of cash flow, which do not necessarily source from the same industry as the asset.
We consider a slightly modified version of the factor loading method of Shepard, utilizing the definition of financial beta in the form of the regression

$$r_k = \beta R_i + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{3.6}$$

which is computed as an exponentially weighted WLS regression. The solution to the WLS regression is given by

$$X_{k,i} = \beta_{k,i} = \left(R_i^\top A R_i\right)^{-1} R_i^\top A r_k \tag{3.7}$$

where $A \in \mathbb{R}^{T \times T}$ is a square weight matrix with an exponential weighting scheme on the diagonal given by

$$A_{t,t} = \frac{\exp\left[-\frac{t}{L}\ln\frac{1}{2}\right]}{\sum_{j=1}^{T}\exp\left[-\frac{j}{L}\ln\frac{1}{2}\right]}, \quad t = 1, 2, \ldots, T \tag{3.8}$$

where L is the halflife of the exponential weight decay. This expression of β places more emphasis on recent observations, as opposed to Equation (3.6), which would consider values at the end of the calculation window equally relevant as the observation of the current day. We show in Section 4.3 that using this factor loading method results in increased explanatory power of the factor model compared to unit exposures.

3.1.2 Style factors

We include style factors in our multi-factor model to increase the explained variance of the model, as well as using their exposures as features in the stacking meta-classifier. Style factors are different from sector and country factors by being exposed to all assets in the loadings matrix, enabling practitioners to see how characteristics such as return momentum and company size reflect on asset return performance on a global scale.
Style factor weights are processed in the same manner as in Menchero, Morozov et al. (2010), where the raw values of the style factor, $d_{s,k}$, are standardized with respect to the cap-weighted mean of the style factor as follows

$$X_{s,k} = \text{winsorize}\left(\frac{d_{s,k} - \mu_s}{\sigma_s}, p\right), \tag{3.9}$$

where the truncation percentile p is determined by maximization of model explained variance using K-fold cross validation on a subset of the asset return data. From this point on, we refer to raw style factors as descriptors, in line with the language used in the MSCI-literature (Menchero, Morozov et al., 2010). In Equation (3.9), μ_s is the market cap weighted mean of all descriptors, and σ_s is the equal weighted standard deviation of all descriptors.

In this thesis, we consider a selection of the most well-documented style factors in the literature. Feng et al. (2020) shows that the production of factors in academic finance has exploded due to academic practitioners failing to rigorously test

whether factors contribute to increasing the explanatory power of factor models. They introduce a novel method of evaluating explanatory power using a hybrid of the double selection LASSO of Belloni et al. (2014) and Fama-Macbeth regression of Fama and MacBeth (1973), then evaluate a selection of 150 factors introduced in the literature in the last 30 years. In addition to the small-minus-big (SMB) factor of Fama and French (1992) and the momentum factor of Jegadeesh and Titman (1993), we include the following significant factors in the model:

Industry-adjusted size

Asness et al. (2000) models the returns of an asset k with respect to its log market cap value m_k in comparison to its mean market cap within-industry, as follows

$$r_k = \gamma_0 + \gamma_1 \overline{m}_k + \gamma_2 (m_k - \overline{m}_k) + \varepsilon_k, \tag{3.10}$$

where $\overline{m}_k$ is the equal weighted average log market cap of all assets in the industry of asset k, in effect breaking traditional explanatory variables into an across-industry component and a within-industry component. The authors explain that only the factor $\gamma_1 - \gamma_2$ is statistically significant ($t = 2.43$), showing that Equation (3.10) is equal to

$$r_k = \gamma_0 + \underbrace{(\gamma_1 - \gamma_2)}_{\gamma_i} \overline{m}_k + \gamma_2 m_k + \varepsilon_k. \tag{3.11}$$

In this implementation we add the style factor γ_i to the model, where γ_i is an instrumental variable (Bowden and Turkington, 1990).

Volatility of dollar trading volume

The notion that liquidity can affect asset returns is widely accepted, shepherded by Amihud and Mendelson (1986). Chordia et al. (2001) successfully verifies that the second moment of liquidity also has a similar effect and uses dollar trading volume as a proxy for liquidity. The variability of dollar trading volume is then computed as the standard deviation of the natural logarithm of dollar trading volume, computed on a window of a trading year. We use NOK as the standard currency, which closely approximates the weights resulting from dollar values after standardization using the methodology in Equation (3.9).

Betting against beta

Frazzini and Pedersen (2014) uses an ordinal rank scale of market beta values as part of a weighting factor, sorting low beta assets and high beta assets into separate portfolios, L_{pf} and H_{pf}, respectively. Beta ranks are given by a vector $z \in \mathbb{R}^N$, containing elements $z_i = \text{rank}_\beta(\beta_i)$, $i = 1, 2, \ldots, N$, where each beta is ranked in comparison to the betas of all other assets in the investment universe. Portfolios are rebalanced every calendar month, following the portfolio weights

$$\begin{aligned} w_H &= k(z - \bar{z})^+ \\ w_L &= k(z - \bar{z})^- \end{aligned} \tag{3.12}$$

where k is a normalizing constant expressed as $k = \frac{1}{2}\sum_{i=1}^{N} |z_i - \bar{z}|$. The $+$ or $-$ superscripts indicate the positive and negative elements of the vector $z - \bar{z}$, putting all assets with lower than average beta rank into portfolio L_{pf} and vice versa into H_{pf}. We calculate β-values as described in Equation (3.5), setting $R_i = R_M$ (see **??**). Note that Frazzini and Pedersen employs a shrinkage measure on the market β values as described in Vasicek (1973) before ranking them in order to compute a theoretical zero-beta portfolio. However, the shrinkage measure shrinks the β values linearly towards the cross-sectional mean, which does not change their ordinal ranks. Therefore, we skip this processing step when exclusively computing factor weights and rank the exposures to the market factor.

3.1.3 Regression weights

The type of weights w is a design choice dependent on how one wants each asset in the factor to influence the latent factor value. Menchero and Nagy (2013) shows in *The Impact of Regression Weighting* that using the square root of current log market capitalization as weights acts as a proxy for inverse specific variance weighting, which minimizes sampling error in factor realizations. Therefore we let $w_k = \sqrt{m_k}$, $k = 1, \ldots, N$, be the weights on the diagonal of $W \in \mathbb{R}^{N \times N}$ in Equation (2.9).

3.1.4 Panel regression

We let the equality constraint weights from Equation (3.2) be put into vector form, defining a constraint matrix $C \in \mathbb{R}^{2 \times K}$ containing factor-wise constraint weights

$$C = \begin{bmatrix} w_{\mathcal{C}_1} & \cdots & w_{\mathcal{C}_n} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & w_{\mathcal{I}_1} & \cdots & w_{\mathcal{I}_m} \end{bmatrix} \tag{3.13}$$

such that primal feasibility is satisfied if $C\beta = 0$ (see Theorem 1). This approach is dependent on the ordering of factors within β, meaning that the n-th column of C corresponds to the n-th factor, β_n.

Theorem 2 *Given the optimization problem*

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|W^{\frac{1}{2}}\left(r - X^\top \beta\right)\|_2^2 \\ \text{subject to} \quad & C\beta = 0 \end{aligned} \tag{3.14}$$

where $W \in \mathbb{R}^{N \times N}$ is a diagonal error covariance matrix, $r \in \mathbb{R}^N$ is a return vector, and $C \in \mathbb{R}^{2 \times K}$ is a linear equality constraint matrix as described in Equation (3.13), a solution, $\beta \in \mathbb{R}^K$, is given by

$$\begin{bmatrix} \beta \\ \lambda \end{bmatrix} = \begin{bmatrix} X^\top W X & \mathrm{C}^\top \\ \mathrm{C} & 0 \end{bmatrix}^\dagger \begin{bmatrix} X^\top W r \\ 0 \end{bmatrix} \tag{3.15}$$

and is valid if the equality constraints are satisfied.

Proof 1 (Proof of Theorem 2) *Taking the objective function of the WLS from Equation (2.9), we define the Lagrangian of the system*

$$\begin{aligned}\mathcal{L}(\beta,\lambda) :&= \frac{1}{2}\|W^{\frac{1}{2}}(r - X^\top\beta)\|_2^2 + \lambda^\top C\beta \\ &= \frac{1}{2}(r - X^\top\beta)^\top W(r - X^\top\beta) + \lambda^\top C\beta\end{aligned} \tag{3.16}$$

Taking the partial derivatives of the Lagrangian and finding where they vanish, we obtain the linear system

$$\begin{bmatrix} X^\top WX & \mathrm{C}^\top \\ \mathrm{C} & 0 \end{bmatrix}\begin{bmatrix} \beta \\ \lambda \end{bmatrix} = \begin{bmatrix} X^\top Wr \\ 0 \end{bmatrix} \tag{3.17}$$

Equation (3.17) is then solved using a left pseudoinverse, yielding proposed factor returns and Lagrange multipliers on the form

$$\begin{bmatrix} \beta \\ \lambda \end{bmatrix} = \begin{bmatrix} X^\top WX & \mathrm{C}^\top \\ \mathrm{C} & 0 \end{bmatrix}^\dagger \begin{bmatrix} X^\top Wr \\ 0 \end{bmatrix} \tag{3.18}$$

Since we have time-varying returns, regression weights, and loadings matrices, the panel regression is solved by chronologically solving the cross-sectional regression described in Theorem 2. This is equivalent to solving a fixed-effects panel regression with pooled least squares. Asset return values are winsorized at $\pm 3\sigma$ for any given asset to reduce extreme values of factor returns, especially in factors with few participants.

3.1.5 Model explanatory power

We measure the explanatory power of the factor model by computing the R^2 of the cross sectional regressions

$$R^2 = 1 - \frac{\sum_k w_k u_k^2}{\sum_k w_k r_k^2} \tag{3.19}$$

where w_k is the regression weight and u_k is the regression residual for asset k, following the definition of Guerard Jr (2009).

3.2 Order execution model

In this section, we detail the construction of an order execution model based on ensemble learning, where output probabilities of a set of tree-based classification models are used as inputs in a stacking meta-classifier. The tree-based classifiers are primarily trained on data features of deviations from asset return predictions over time, where return predictions are directly computed from factor realizations and loadings detailed in Section 3.1.

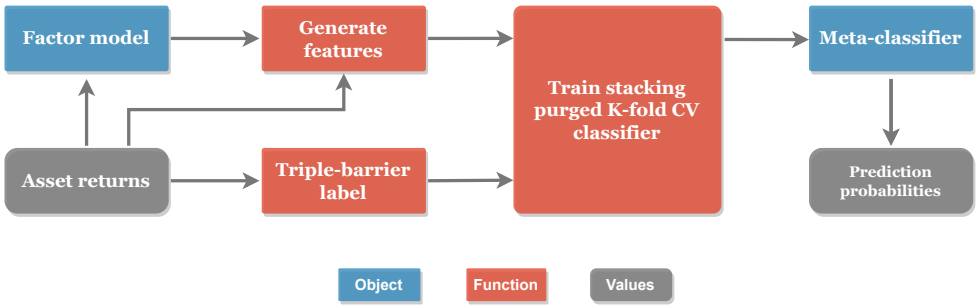


Figure 3.1: Overview of the order execution model.

Consider a rough overview of the order execution model in Figure 3.1, showing how factor model loadings and asset returns are used to compute training data for a meta-classifier model, which predicts the sign of the next asset return. The meta-classifier is a logistic regression model trained on the prediction probabilities of four light gradient boosting models (LGBM), each of which is trained on classifier features detailed in Section 3.2.2. The training procedure is based on a slightly modified version of stacked generalization in Tang et al. (2014), using purged K-fold cross-validation of De Prado (2018) as its cross-validation splitter. The process is described in detail in Section 3.2.3.

3.2.1 Execution decisions from meta-classifier predictions

Letting trading signals (pending orders) be contained in vectors d for each asset, each signal $d_t \in d$ is the signal for time t with values $d_t = -1 \vee 1$. For each point in time we take the meta-classifier prediction probability, $-1 < p_m < 1$, and checks if the sign of the signal d_t matches the sign of p_m, and that the prediction probability is greater than $p_{\text{threshold}}$. Let $\{p_m \mid p_m \leq 0\}$ correspond to negative return predictions within the prediction horizon. Vice versa, we let the rest of the classifications be positive return predictions. Algorithm 1 shows how a signal is executed only if the meta-classifier model agrees with the trading direction of the signal. Since orders have a long time horizon, they do not have to be executed instantly. If the meta-classifier model (which is short-term based) disagrees with the direction of the long-term signal, the signal is placed in embargo. Signals are executed regardless after a specified maximum embargo period E. If the dates of

two orders in the same asset coincide due to Algorithm 1, the most recent one is chosen.

Algorithm 1 Make trade execution decision from classifier predictions.

procedure DECIDE_EXECUTION(d, E, p_m, $p_{\text{threshold}}$)
  $j \leftarrow 1$
  while $j \leq E$ **do**
    if $d_j = \frac{|p_m|}{p_m}$ **and** $|p_m| \geq p_{\text{threshold}}$ **then**
      Execute trade d, j days after original signal date.
    end if
    $j \leftarrow j + 1$
  end while
  Execute trade d, E days after original signal date.
end procedure

We use a threshold on prediction probabilities, $p_{\text{threshold}}$, to adjust the confidence in classifications. Some trading signals, such as signals based on news releases and quarterly reports, tend to decay in profitability as a function of time after publication. This means that a delay in execution caused by a high-conviction meta-classifier signal can result in a less profitable trade if the trading strategy we want to improve execution on is associated with a high degree of time-based profit decay. On the other hand, if a strategy is robust to time delays, $p_{\text{threshold}}$ can be set to a value closer to one. Section 3.2.4 details how $p_{\text{threshold}}$ is determined with respect to the distribution of prediction probabilities and time based profit decay.

3.2.2 Base classifier features

We include a total of $q = 20$ features for the base classifiers, as follows.
The vector $\hat{r}_t \in \mathbb{R}^N$ of asset return now-casts are given by $\hat{r}_t = X_t \beta_t$, the loadings and factor realizations of the factor model for time t. It is important to emphasize that today's asset prices, and therefore returns, cannot be used when predicting the same returns.
The close prices of the current and previous dates are used for computing factor realizations which are then used to compute $\hat{r}$. When residuals from actual returns $r - \hat{r}$ are calculated, processed, and used as inputs into classification models, it is based on the assumption that they have prediction value about the direction of future price changes.
Table 3.1 shows the features of the base classifiers. All features with a temporal aspect, such as cumulative percentage deviations, are calculated on a rolling basis, reducing the possibility of data set leakage. Features are based on the hypothesis that deviations over time of true asset returns from predicted returns tend to resolve themselves and that the magnitude of deviation, as well as consensus across time-frames, is predictive in estimating future price direction. We also include the

Table 3.1: Input features $\mathcal{D}_b$ of base classifiers. Note that although seven different types of features are included, some of them have variations in parameters, resulting in 20 features in total.

Name	Definition	Variations
Cumulative percent deviation (nowcast - true)	$\exp\left[\sum_{i=1}^{w}(\hat{r}_{-i} - r_{-i})\right] - 1, \quad w \in W$	$W = \{4, 8, 16, 32\}$
Cumulative percent deviation (equal weighted - true)	$\exp\left[\sum_{i=1}^{w}(\tilde{r}_{-i} - r_{-i})\right] - 1, \quad w \in W$	$W = \{4, 8, 16, 32\}$
Cumulative percent deviation (equal weighted - nowcast)	$\exp\left[\sum_{i=1}^{w}(\tilde{r}_{-i} - \hat{r}_{-i})\right] - 1, \quad w \in W$	$W = \{4, 8, 16, 32\}$
Style factor loading of asset k	$X_{s,k}$	All style factors (Section 3.1.2)
R^2 (nowcast, true)	Exponentially weighted R^2 with a 25-day halflife	-
Most recent asset return	r_{-1}	-
Most recent nowcast	$\hat{r}_{-1}$	-

R^2-value between nowcasts and true returns to adjust the confidence of classifications, such that worse-fitting assets have prediction probabilities close to 50%. Having classification probabilities that reflect real-world values is paramount for our base classifiers, as their outputs are used as inputs in the final meta-classifier. In addition to utilizing residuals from nowcasts, we introduce an equal-weighted asset equivalent, meaning that for any given asset participating in several factors, its equivalent would be the estimated return mean of all assets inhabiting the same characteristics. For an asset participating in sector i, country c and style factor s, we express the equivalent as

$$\begin{aligned} \tilde{r}_k^t &= \frac{\beta_c^t}{|\mathcal{A}_c|}\sum_{j=1}^{|\mathcal{A}_c|} X_{jc} + \frac{\beta_i^t}{|\mathcal{A}_i|}\sum_{j=1}^{|\mathcal{A}_i|} X_{ji} + \underbrace{\frac{\beta_s^t}{N}\sum_{j=1}^{N} X_{js}}_{=0} \\ &= \beta_c^t + \beta_i^t \end{aligned} \tag{3.20}$$

where $|\mathcal{S}|$ is the size of a set $\mathcal{S}$. Unit exposures of industry and country factors follow from their loadings being beta-exposures towards their cap-weighted return averages, as described in Section 3.1.1. Style factors vanish as a result of style factor constraints of Section 3.1.2. The decision of only including one style factor in Equation (3.20) is irrelevant, as additional style factors also would vanish in the aggregate. We use $\tilde{r}$ in the same manner as $\hat{r}$, where we compute cumulative percent deviations from true asset returns. These features are designed to give a reference of how well an asset is performing in comparison to similar assets, in effect calculating specific indices for each asset. This added reference point is informative, as it explains how well assets are performing compared to how well their peers are expected to perform, and how well assets are expected to perform compared to how well their peers are expected to perform. This enables a well trained model to identify and remove recent biases in the factor model return estimates.

3.2.3 Stacking with purged K-fold Cross Validation

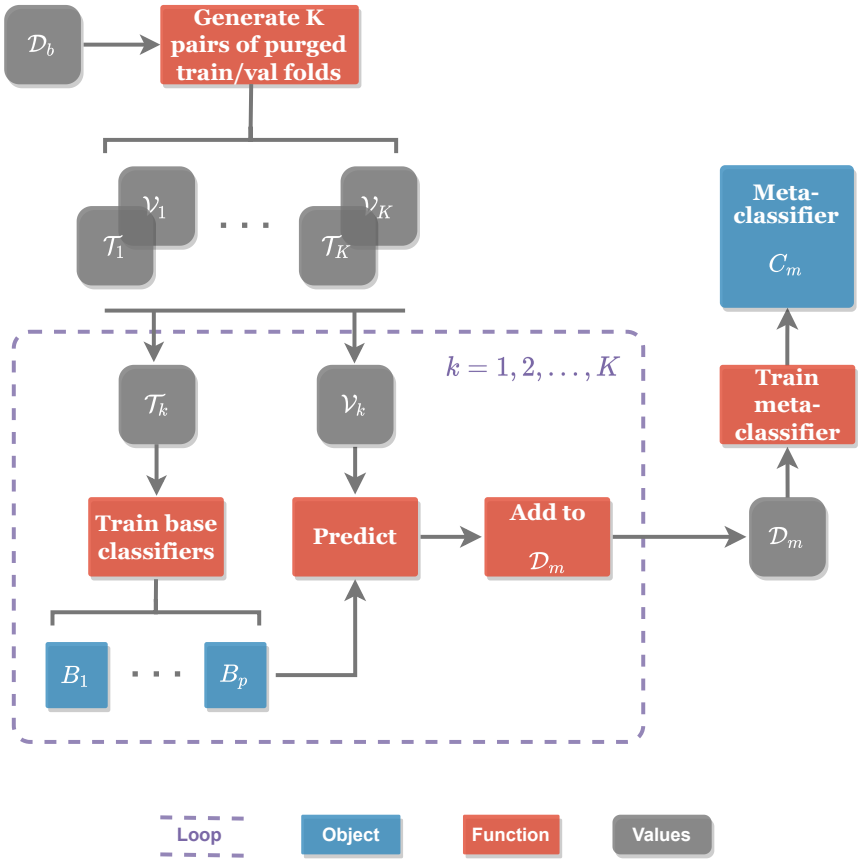


Figure 3.2: Purged K-fold cross validation stacking.

Using triple-barrier labeled prices as target variables and data features as listed in Table 3.1, a selection of tree-based classifiers are trained as bases[1] to a stacked ensemble learner we refer to as the meta-classifier. In this thesis we introduce a modified version of stacking with K-fold cross-validation, where instead of randomly splitting a data set $\mathcal{D}$ into K even validation folds $\mathcal{D} = \{\mathcal{D}_k \mid |\mathcal{D}_k| = c\}_{k=1}^{K} (c \in \mathbb{Z}^+)$, we generate K pairs of purged training and validation pairs, $\{\mathcal{T}_k, \mathcal{V}_k\}_{k=1}^{K}$. In the original approach described in Tang et al. (2014), training folds are simply expressed as $\mathcal{T}_k = \mathcal{D} \setminus \mathcal{D}_k$. In the case of overlapping time-series data, such as financial data, this approach would cause data leakage and, therefore, over-optimistic results. Stacking with purged K-fold cross-validation reduces data leakage by negating data samples in the training set if they are closer to the first validation set sample than a set purging duration. Despite purging, the meta-classifier is exposed to each sample in $\mathcal{D}$ exactly once, as $\{\mathcal{V}_k\}_{k=1}^{K} = \mathcal{D}$. This is a property of regular stacking

[1] We refer to base classifiers and meta-classifiers synonymously to what Wolpert respectively would refer to as level 0 and level 1 learners in Wolpert (1992).

with K-fold cross-validation as well, meaning that only the size of training sets is decreased with purged data set splitting.
Let $\mathcal{D}_b = \{(x_t, y_t)\}_{t=1}^{T} (x_t \in \mathbb{R}^q, y_t \in \mathbb{R})$ denote the training data set for base classifiers, $B_i, i = 1, 2, \ldots, p$, containing the features of Table 3.1 as well as triple-barrier labels as targets. Also, let $\mathcal{D}_m = \{(b_t, y_t)\}_{t=1}^{T} (b_t = [B_1(x_t), \ldots, B_p(x_t)]^\top, y_t \in \mathbb{R})$ be the training data set of the meta-classifier, C_m, containing prediction probabilities of the base classifiers as features and the same targets as in $\mathcal{D}_b$. The meta-classifier is then trained on the full training data set $\mathcal{D}_b$ using purged K-fold cross-validation. As in regular stacking with cross-validation, we construct the training set for the meta-classifier with the probabilistic outputs of the base classifiers. This method differs in how training and validation folds are computed. See Figure 3.2 for an overview illustration of the method. A specific explanation is documented in Algorithm 2.

Algorithm 2 Stacking with purged K-fold cross validation.

procedure STACK_PURGED($\mathcal{D}_b$)
  Split $\mathcal{D}_b$ into K tuples of $\{\mathcal{T}_k, \mathcal{V}_k\}_{k=1}^{K}$ using purged K-fold CV
  $\mathcal{D}_m \leftarrow \emptyset$
  for $k \leftarrow 1$ to K **do**
    for $i \leftarrow 1$ to p **do**
      Train base classifier B_i on $\mathcal{T}_k$ (Section 3.2.5)
    end for
    for $x_i, y_i \in \mathcal{V}_k$ **do** ▷ Add outputs of base classifiers to $\mathcal{D}_m$
      $\mathcal{D}_m \leftarrow \mathcal{D}_m \cup \{\hat{x_i}, y_i\}$, where $\hat{x_i} = \{B_1(x_i), B_2(x_i), \ldots, B_p(x_i)\}$
    end for
  end for
  Train meta-classifier C_m on $\mathcal{D}_m$
  for $i \leftarrow 1$ to p **do** ▷ Retrain base classifiers on entire training set
    Train base classifier B_i on $\mathcal{D}_b$
  end for
  return $C_m(B_1(x), B_1(x), \ldots, B_p(x))$
end procedure

3.2.4 Determining key execution parameters

Key parameters are defined as the parameters that are thought to have the greatest influence on the performance of the execution strategy while being independent of any classifier parameters discussed in the following subsections. We consider these parameters to be the prediction probability threshold $p_{\text{threshold}}$, the maximum embargo period E, and the right triple barrier limit R_{triple}. The final parameters are thought to maximize the Sharpe ratio of the execution strategy. De Prado (2018) emphasizes the importance of not using historical backtesting as a research tool, and the adage of Goodhart's law states that "*When a measure be-*

comes a target, it ceases to be a good measure" (Goodhart (1984)). Therefore, we choose the value that maximizes the signal-to-noise ratio of excess return on the validation set of the meta-classifier, which we use as a proxy for the Sharpe ratio. The measured Sharpe ratio is only calculated when backtesting, which is performed after the completion of all modeling. Notice the similarity of the Sharpe ratio, S_a, of Sharpe (1998) and the square root of the signal to noise ratio of a random variable, $\sqrt{\text{SNR}_a}$, defined in Butler and Sherman (2016) as

$$S_a = \frac{\mathbb{E}[r_a - r_b]}{\sigma_a} \tag{3.21}$$

$$\sqrt{\text{SNR}_a} = \sqrt{\frac{\mathbb{E}[r_a]^2}{\sigma_a^2}} = \frac{\mathbb{E}[r_a]}{\sigma_a} \tag{3.22}$$

where r_a is the return of an investment, r_b is the risk-free return, and σ_a is the standard deviation of r_a. If we let $r_b = 0$, then the two definitions are equal. Regardless, the ranks of the different values are preserved between the two definitions, as the risk-free return can be seen as a linear bias[2]. Since we cannot simulate the actual excess profit from the execution strategy before backtesting, we model the distribution of $\sqrt{\text{SNR}_a}$ and extract the parameters that maximize its expected value. Assume that for any profitable trade by the execution strategy, we receive a constant payoff per day. Since profits are positive, we estimate the per day profit to be the mean absolute return over all assets in the investment universe, that is

$$\mu_{|r|} = \frac{1}{NT} \sum_{k=1}^{N} \left(\sum_{t=1}^{T} |r_{k,t}^x| \right) \tag{3.23}$$

where $r^x = e^r - 1$ is a linearly scaled asset return[3]. The mean probability of the meta-classifier correctly classifying future return direction, ε, can be estimated empirically from classification probabilities predicted from a validation data set, and represents $\varepsilon = Pr(C_m(x) = y \mid |p_m| \geq p_{\text{threshold}})$ for any sample x and target return direction y. Then mean profit the day after an order execution is modelled as

$$\begin{aligned} \pi_d &= \mu_{|r|}\varepsilon - \mu_{|r|}(1-\varepsilon) \\ &= \mu_{|r|}(2\varepsilon - 1) \end{aligned} \tag{3.24}$$

Following Section 3.2.1, we use Algorithm 1 to execute an order within E days. Let β be the probability[4] of the meta-classifier outputting a prediction probability greater than $p_{\text{threshold}}$. Then $p_c = 1 - (1 - \alpha\beta)^D$ is the probability of encountering at least one probability value $|p_m|$ greater than $p_{\text{threshold}}$ within the period, and that the order direction is equal to the predicted direction. Here α is defined

[2] Consider that $a + b < c + b$ if $a < c$ for all $b \in \mathbb{R}$.

[3] Recall that r is log-scaled.

[4] This can be estimated from the empirical cumulative distribution of prediction probabilities from a validation data set.

as the probability that the meta-classifier predicts that the price will move in the same direction as the trade side, e.g. when predicted positive price change and a long order coincides. $D \in \mathbb{Z}^+$ is the expected embargo duration $D = \lceil \min\{\frac{1}{\beta}, E\} \rceil$. Since $\frac{1}{\beta}$ quickly exceeds E when β decreases, we have to enforce the maximum embargo duration E by taking the minimum of the two at each time. Finally, $\pi_d p_c$ represents the mean gross profit over an embargo period. However, as a side effect of delaying the original trading strategy by D days on average, we can expect a profit reduction in the original trading strategy as a function of the time delay D. This loss in profitability is dependent on the extent to which the strategy acts on time-sensitive information, such as real-time news updates or quarterly financial reports of companies. We define θ_t as the per day profit percentage difference caused by delaying the original trading strategy by t days. θ_t is estimated empirically by backtesting the original trading strategy on a validation data set using orders that are shifted by t days, then estimating the mean and standard deviation of the difference in profit compared to the performance of the un-shifted strategy. If we crudely assume that $\theta_t \sim \mathcal{N}(\mu_{\theta_t}, \sigma^2_{\theta_t})$, then profit decay over a period of t days is sampled from $t\theta_t \sim \mathcal{N}(t\mu_{\theta_t}, t^2\sigma^2_{\theta_t})$. Therefore, the estimated net execution strategy profit per order is sampled from

$$\pi_E \sim \mathcal{N}\left(\pi_d p_c - D\mu_{\theta_D}, D^2\sigma^2_{\theta_D}\right) \tag{3.25}$$

which we can use to model our proxy for Sharpe ratio, $\sqrt{SNR}$, as defined in Equation (3.22), as

$$\begin{aligned} \sqrt{SNR}_E &\sim \mathcal{N}\left(\frac{\pi_d p_c - D\mu_{\theta_D}}{D\sigma_{\theta_D}}, 1\right) \\ &\sim \mathcal{N}\left(\frac{\pi_d p_c}{D\sigma_{\theta_D}} - \frac{\mu_{\theta_D}}{\sigma_{\theta_D}}, 1\right) \end{aligned} \tag{3.26}$$

when we recognize the analogs $r_a := \pi_d p_c - D\mu_{\theta_D}$ and $\sigma_a := D^2\sigma^2_{\theta_D}$. Then the maximization of its expected value, $\mathbb{E}\left[\sqrt{SNR}_E\right]$, yields E and $p_{\text{threshold}}$. We formalize this optimization problem as

$$E^\star, p^\star_{\text{threshold}} = \underset{E, p_{\text{threshold}}}{\arg\max} \frac{\pi_d p_c}{D\sigma_{\theta_D}} - \frac{\mu_{\theta_D}}{\sigma_{\theta_D}} \tag{3.27}$$

where $E^\star$ and $p^\star_{\text{threshold}}$ represents the parameters that are modelled to result in an execution strategy that maximizes the Sharpe ratio of the original trading strategy. The right triple barrier label length $R^\star_{\text{triple}}$ is chosen by training the meta-classifier as described in Section 3.2.3 for different values of R_{triple} in the training data set and optimizing Equation (3.27) for each of them. The final value of R_{triple} is the one that maximizes $\mathbb{E}\left[\sqrt{SNR}_E\right]$ over all candidate triple barrier lengths.

3.2.5 Randomized parameter search for base classifiers

Each base classifier B_i is optimized K times (see Figure 3.2), and each optimization step is performed by using each $\mathcal{T}_k$ as a standalone data set for purged cross-validation. It should be emphasized that this data set splitting is additional to when $\mathcal{D}_b$ is split K times. Letting F be the number of folds when optimizing any B_i, the total number of samples to be learned is $TK \cdot \frac{T}{K}F = T^2F$. This number is then multiplied by the number of attempted parameter combinations, n, which can make the process very computationally intensive. To simplify the complexity of each base classifier optimization, we choose random parameter search over, e.g., a stochastic gradient descent of log-loss. Instead of iterating until an optimum is reached, we specify a performance quantile $q \in [0, 1]$ that we find acceptable for each base classifier. The optimal parameter combination yields performance in $q = 1$.
For each parameter we wish to tune in the base classifier, we specify a range of values from which the random search can choose. Assuming that an optimal parameter combination lies within the specified ranges, we know that over a number of trials, we can place each trial's performance within a quantile. Setting a power p, we can estimate how many iterations of random search are needed to reach a parameter combination with a performance above the qth quantile, with $100p$ % certainty. Assuming that trials are independent, the probability that all n of them are beneath q is q^n. Therefore, the probability that at least one of the trials is above q is $1-q^n$. Hence, we can express the sufficient number of trials as

$$1-q^n \geq p \quad \Longrightarrow \quad n \geq \frac{\log(1-p)}{\log(q)} \tag{3.28}$$

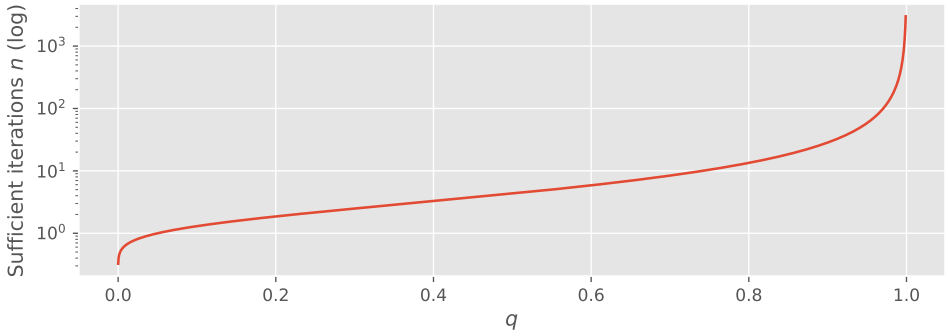


Figure 3.3: The amount of iterations sufficient for encountering a model performance within the quantile q, with p set to 95% confidence (log scale).

Figure 3.3 shows the relationship between n and q if $p = 0.95$. The iteration number can then be chosen based on time- and computational constraints. As an example, a result in the top 5% with 95% confidence requires $n = 59$ iterations, following Equation (3.28).

3.2.6 Feature selection for base classifiers

Since base classifiers are tree-based, we utilize feature permutation importance of Altmann et al. (2010), and leave out data features with less than zero importance. Permutation importance is used, as the most frequently used feature importance methods are based on impurity reduction of splits (MDI), which are proven to be biased (Breiman et al. (1984)). This bias can be illustrated by adding a noise column as a data feature, then applying the various feature importance methods to a trained model. For impurity reduction-based methods, the noise feature will show a non-zero and, at times, substantial importance, as displayed in *Permutation Importance vs Random Forest Feature Importance (MDI)* (2022). MDI, although fast, only considers the mean prevalence of a feature within all trees. Permutation importance purposely corrupts a feature and calculates its effect on model performance. This approach is less biased but comes at the cost of being more computationally demanding.

Chapter 4

Experimental results

The approach described in Chapter 3 discusses a general framework for constructing an execution model based on stacking classifiers using information from a multi-factor model. Except for a description of the features of the base classifiers, the type and amount of classifiers are arbitrary such that the method can be extended to trading strategies of different types and time horizons. In this chapter, we test the performance of the execution strategy by training a meta-classifier based on real market information, factor realizations, and exposures and then testing it out-of-sample by improving trading signals from two real trading strategies, *Strategy 1* and *Strategy 2*. We benchmark the performance of the execution strategy against the performance of the original trading strategy and establish whether market conditions are influential on the accuracy of the meta-classifier. We also test the explanatory power of the factor model, investigating whether or not the factor loading method detailed in Section 3.1.1 shows promise over traditional unit weighted loadings.

4.1 Measures against data leakage

When evaluating the effectiveness of the execution strategy, minimizing the risk of data leakage is of the highest priority. Data leakage occurs when a model accesses information outside its training data set. This can happen directly, as a result of a logical error, or indirectly, in the form of "p-hacking". P-hacking is a human error made when only positive results are reported after multiple testing. For instance, if several models are trained on a training set, but only the top-performing model on the test set is chosen, then the results have been p-hacked. In a scenario like this, the test set loses its purpose and becomes a part of the training set. In this thesis, we reduce the risk of this happening by exposing the model to the test set precisely once per trading strategy, meaning that all parameters are determined before historical backtesting begins. Another measure against p-hacking was receiving the out-of-sample trading signals of *Strategy 2* after the completion of all modeling, making it impossible to p-hack the performance of the execution strategy on that trading strategy.

Throughout the development of the execution model, we assume that we do not necessarily get to execute orders instantly after market data is received. Therefore we increase model robustness by computing classifier features on market open prices and labeling targets from market close prices (starting the next day).

4.2 Experimental setup

This section describes the specifics of testing the execution strategy and its underlying components, such as the factor model, the base classifiers, and the meta-classifier. This includes specifying the trading universe, data sources, data processing, and parameter choices.

4.2.1 Data sources

Throughout the thesis, we use data from three different sources. Firstly, market data such as daily price quotes, currencies, market capitalization, and volume are used to compute factor realizations as well as getting the differences between predicted and actual price returns. We use a selection of 1150 European stocks and the currency pairs EURNOK, SEKNOK, and DKKNOK, which we retrieve from Refinitiv Eikon through a subscription managed by NTNU. Secondly, asset characteristics such as industry, currency, and country of origin are used to construct the daily updating dynamic factor matrices. Lastly, we estimate the performance of the execution strategy by simulating signals from real trading strategies supplied by *Intelligent Trading*, where the signals with unmodified execution dates give us a benchmark equity curve used to statistically test whether the execution strategy is significantly more profitable. See Table 4.1 for a sample of how trading signals from *Intelligent Trading* are listed. Note that the numbers in the table have been fictionalized, as the company does not wish to disclose the signals to the public. We test execution model performance on two different trading strategies. One that is mostly based on momentum, meaning in short terms that recently overperforming stocks are thought to continue to over-perform, and vice versa. We refer to this strategy as *Strategy 1*. The other strategy is primarily based on fundamental information such as quarterly financial reports. We refer to this strategy as *Strategy 2*. One can assume that the former is more sensitive to delays in execution time, as other market participants act on the same information as quickly as it is released. The momentum effect of Jegadeesh and Titman (1993) assumes monthly rebalancing, which indicates that it is resistant to substantial time delays. See Table 4.4 for empirical estimations of θ_t (Section 3.2.4), which support this assumption. For the factor model, we use market data from 2004-2022, and factors are computed on market open prices[1]. Classifiers use factor model exposures and factor realizations in the features described in Section 3.2.2, which cover trading dates between 2006 and April 2022. Triple-barrier labels use market close prices in the same date range.

[1]The O in OHLC bar data.

Table 4.1: Fictionalized sample of trading signals. Column explanations: *Security ID*: asset identifier. *Side*: Buy (1) or sell (-1) order. When *Previous position* is zero or negative, -1 is a short-order. *Previous position*: The amount of shares of this asset previously held in the portfolio. An increase or decrease of position that does not result in a zero exposure to the stock is a rebalancing trade. *Order volume*: Amount of shares. *Currency*: Currency of *Security ID*.

Schema	d_{first}	...	d_{last}
Date	2007-01-02	···	2022-04-25
Security ID	DHER GY	···	MAERSKB DC
Side	-1	···	1
Previous position	-3764	···	334
Order volume	263	···	22
Currency	EUR	···	DKK
Fill price	117.3218	···	15553.9602

4.2.2 Data processing

When used in regression constraints, currency pairs are used to convert all market capitalization values into NOK, ensuring that we are not over-emphasizing assets where the currency value has a large scalar value. There is no specific reason NOK is chosen as the currency other than that values are ensured to have the same unit. When the market capitalizations are used as weights, the scale is insignificant as the weights are normalized.

4.2.3 Factor model specifications

We implement a factor model as developed in Section 3.1, with the factors listed in Table A.1. The halflife of the exponentially weighted beta values (Equation (3.8)) is set to - as a result of maximizing explanatory power from 2004-2006. Equality constraints of Equation (3.2) are weighted by market cap, and regression weights of WLS-regressions are set to square roots of market capitalization in line with Section 3.1.3. Asset returns are winsorized at $\pm 3\sigma_k$, where σ_k is an asset standard deviation of log returns computed from rolling sample variance over a trading year. We compute factors using each trading day's opening prices to counteract implicit data leakage caused by assuming that traders can execute on closing prices. Since style factor exposures and return now-casts are used as data features in the meta-classifier, mistakenly classifying price direction based on data only retrievable later in the day would be a large methodical error. Figure A.1 displays the result of implementing the specified factor model. Other information is also stored, such as regression residuals and time-varying exposure matrices.

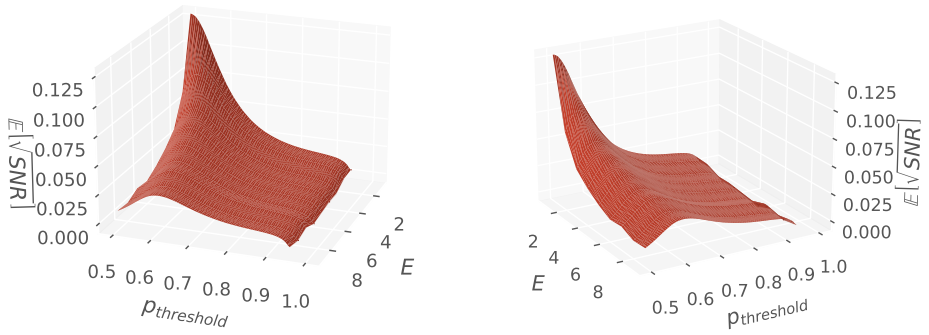


Figure 4.1: Model of $\mathbb{E}\left[\sqrt{SNR}\right]$ of excess profit net the original trading strategy. The figure includes two different views of the surface. Arguments at maximum: $p^{\star}_{\text{threshold}} = 0.5$ and $E^{\star} = 2$.

4.2.4 Meta-classifier specifications

Using the data set described in Section 3.2.2 we train a meta-classifier using prediction probabilities from two base classifiers, from 2006 until the end of 2019. In this thesis, we let the base classifiers be a random forest classifier and a light gradient boosting machine trained on the same features and targets. The meta-classifier is a logistic regression model. Using stacking with purged K-fold cross-validation as detailed in Section 3.2.3, we optimize the model parameters in Table 4.2 using random search, where the sufficient number of iterations is calculated by specifying a tolerance of results in the top 5% quantile with a 95% confidence. Application of Equation (3.28) yields 59 iterations. Purged K-fold cross-validation is set to $K = 4$ and a purging period of 175 days for the stacking process as well as in randomized parameter search. The purging period is set to 175 days, as it is the smallest period that allows exponential weights outside the purging period to have less than 1% of their original size, given an exponential window halflife of 25 days. Out of the 20 data features in Table 4.2, the only feature affected is the rolling correlation, meaning that we implicitly allow a minimal degree of data leakage into the validation folds when training the stacking meta-classifier. However, this does not affect backtesting simulations, as they are calculated on a testing data set from 2020 to May 2022. After generating the classifier features of Table 3.1 and generating triple-barrier labels with lengths 1, 3, and 5, we train three separate meta-classifiers using the different triple-barrier labels as targets. We estimate the parameters $R^{\star}_{\text{right}} = 1$, $p^{\star}_{\text{threshold}} = 0.5$ and $E^{\star} = 2$ by solving Equation (3.27) on the training set from 2006-2019, meaning that values of β, α and θ are estimated from meta-classifier predictions and backtest performance of the original trading strategy from all data up to and including 2019. Figure 4.1 shows the surface of $\mathbb{E}\left[\sqrt{SNR}\right]$ as a function of E and $p_{\text{threshold}}$. The final meta-classifier

Table 4.2: Search space for random parameter search of the models involved in the execution classifier.

Model	Parameter name	Search space	Final value
Logistic regression	C	0.1, 1, 10	0.1
Random forest classifier	n estimators	50, 100, 200, 500	200
	max depth	3, 4, 5, 6	4
LGBM	n estimators	50, 100, 200, 500	200
	max depth	3, 4, 5, 6	4
	learning rate	0.01, 0.1, 0.2, 0.3	0.1

parameters are listed in Table 4.2 after a randomized parameter search on both base classifiers within the meta-classifier using permutations of the parameters described in the same table. The meta-classifier achieves an accuracy of 52.6% on validation folds, which is higher than the class bias of 50.2%[2]. We measure an F1 score of 52.3%.

4.3 Factor model explanatory power

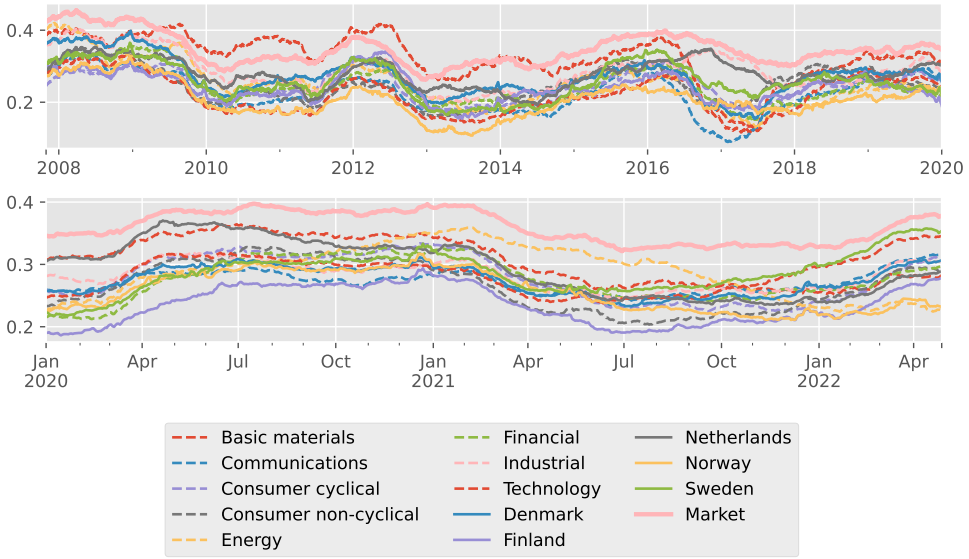


Figure 4.2: Rolling explanatory power (R^2) of the beta weighted factor model. Top view: years 2008-2020. Bottom view: years 2020-2022.

Recall the definition of Equation (3.19), where the explanatory power of the factor

[2]Markets tend to rise over time, which explains how returns are slightly more likely to be positive.

model is given by

$$R^2 = 1 - \frac{\sum_k w_k u_k^2}{\sum_k w_k r_k^2} \tag{4.1}$$

where weights w are market capitalizations and u are regression residuals. As discussed in Section 3.1.1, factor exposure matrices are typically unit weighted, meaning that for an example where asset A is a member of factor F, then $X_{A,F} = 1$, otherwise 0. Section 3.1.1 also describes the method used in this thesis, where the exposure of $X_{A,F}$ is the exponentially weighted beta value of the return r_A against the market cap-weighted return average of all assets that are members of factor F. We compute the factor realizations from 2007-2022 using both weighting methods and compare the explanatory powers of each method over rolling periods of a trading year. Explanatory power values are listed in Table 4.3, showing that beta weighting over-performs unit weights in all factors on average. Note that the explanatory power of each factor is not based on the performance of a single-factor model using that factor exclusively. Instead, explanatory power is calculated from returns and regression residuals of assets that are members of a factor. This explains why R^2-values do not sum to the R^2-value of the common market factor. See Figure 4.2 for a plot of rolling explanatory power of the beta weighted factor model.

Table 4.3: Comparison of explanatory power (R^2) of factor models using exposure matrices either with unit weights or with weights of beta against market cap weighted return of each factor.

		Unit weights		Beta weights	
Factor type	Name	μ	σ	μ	σ
Country	Denmark	0.247	0.041	0.272	0.049
	Finland	0.215	0.037	0.240	0.039
	Netherlands	0.240	0.042	0.281	0.041
	Norway	0.186	0.040	0.216	0.049
	Sweden	0.234	0.043	0.263	0.048
Sector	Basic materials	0.293	0.051	0.320	0.063
	Communications	0.197	0.035	0.232	0.054
	Consumer cyclical	0.224	0.042	0.251	0.042
	Consumer non-cyclical	0.218	0.035	0.251	0.048
	Energy	0.230	0.044	0.258	0.059
	Financial	0.226	0.043	0.251	0.049
	Industrial	0.252	0.043	0.288	0.047
	Technology	0.207	0.042	0.232	0.055
Common	Market	0.318	0.039	0.349	0.041

4.4 Simulations

Using an event-driven backtesting engine, we compute historical simulations of trading strategy performances pre- and post-application of the execution strategy. Distribution parameters of θ_t are also estimated in order to solve the optimization problem of Equation (3.27).

4.4.1 Delayed execution strategy returns

Optimizing Equation (3.27) requires estimates of θ_t for relevant time delay values t. The results of Table 4.4 are found by shifting market orders of the original trading strategies by t trading days, which is dependent on the trading calendar of each asset, as trading calendars might differ based on the exchange an asset is traded on. This is among the considerations a systematic asset manager has to make for portfolios that span multiple countries. Simulations using the different amounts of shift are performed by backtesting the signals over a period from 2007-2020. Whether or not backtests for this purpose include trading costs is irrelevant, as the performance difference of a strategy when shifted cancels out incurred costs. This is valid when we know that order sizes are not recalculated prior to execution, implying that the currency cost incurred on a single order is constant when shifted.

Table 4.4: Parameter estimations of θ_t for $t = 0, 1, \ldots, 5$ for both trading strategies supplied by *Intelligent Trading*. Values are loss of return per day compared to the original, un-shifted strategy.

		Time delay (days)					
		0	1	2	3	4	5
Strategy 1	μ_θ	0	-2.0e-6	1.9e-5	8.0e-6	3.7e-5	5.9e-5
(Momentum)	σ_θ	8.8e-3	8.8e-3	8.8e-3	8.8e-3	8.7e-3	8.8e-3
Strategy 2	μ_θ	0	-9.4e-6	4.6e-7	7.2e-5	7.3e-5	9.8e-5
(Financial reports)	σ_θ	1.8e-4	1.8e-4	1.8e-4	1.8e-4	1.8e-4	1.9e-4

4.4.2 Execution strategy performance

We compute historical backtests of both *Strategy 1* and *Strategy 2* using either original trading signals or signals from the execution strategy, which result from Algorithm 1. Backtests are run with and without trading fees, which are set to a constant of 0.2% of each order size. Let the in-sample period be defined as data from 2006 until the end of 2019, and 2020 until May 2022 be defined as the out-of-sample period. Recall that the in-sample period is used for training all models and that the out-of-sample period is previously unseen. Cumulative returns are shown in Figure 4.4 and Figure 4.3, for in-sample and out-of-sample tests, respectively. Backtest statistics for *Strategy 1* are listed in Table 4.5, and statistics for *Strategy 2* are listed in Table 4.6. Historical testing shows that signals using

the execution strategy have higher out-of-sample Sharpe ratios on all counts compared to the original signals. Higher Sharpe ratios are also observed in-sample on all counts except for in-sample on *Strategy 1* (momentum-based) when simulated without trading fees (Exec. strategy Sharpe: 0.96. Original strategy Sharpe: 1.07). Overall, we measure a mean increase in Sharpe ratio of 0.02 when simulating without fees and an increase of 0.75 with fees. Welch's t-tests[3] show that the execution strategy produces significant excess profitability over the original strategy only when fees are included in simulations. *Strategy 1* generates significant excess profit when simulating with fees ($p = 0.0187$), but not without fees ($p = 0.988$). The same goes for *Strategy 2* (financial reports-based), getting p-values of $p = 0.006$ with fees and $p = 0.713$ without fees.

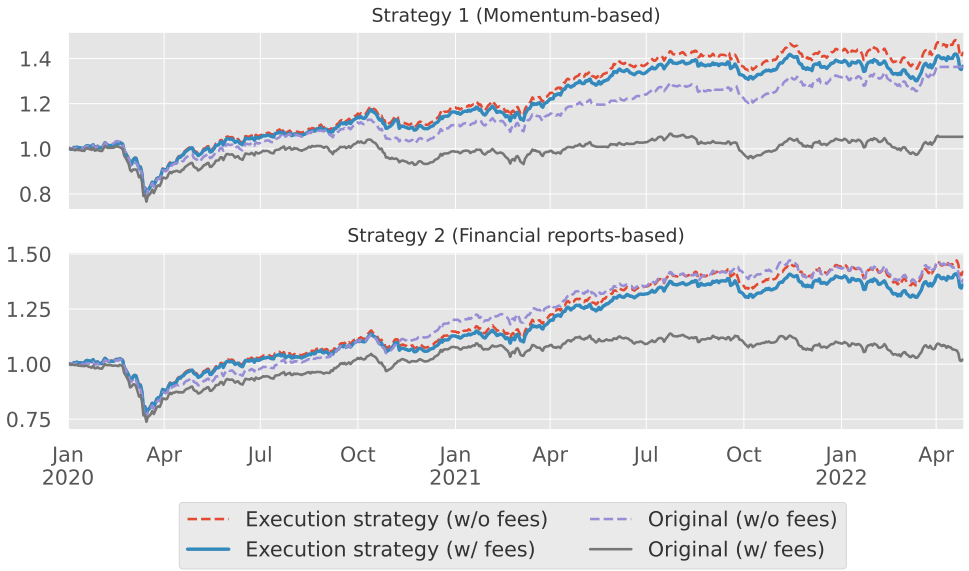


Figure 4.3: Out-of-sample cumulative performance of the execution strategy with and without commission fees, which are set to 0.2% per order.

Although simulations without trading fees are likely to be as profitable with the execution strategy as without, the simulations using the execution strategy generate the same profit with much fewer trades. In *Strategy 1*, the original trading strategy in-sample includes a total of 8260 trades, while the same strategy with improved execution includes 7716 trades - a reduction of 4.9%. From Table 4.5 we can calculate a reduction out-of-sample of 6.6%. In *Strategy 2* we observe reductions of 6.5% and 4.9% in-sample and out-of-sample, respectively.

[3] An adapted version of the Student's t-test where equal population variances are not assumed (Welch (1947))

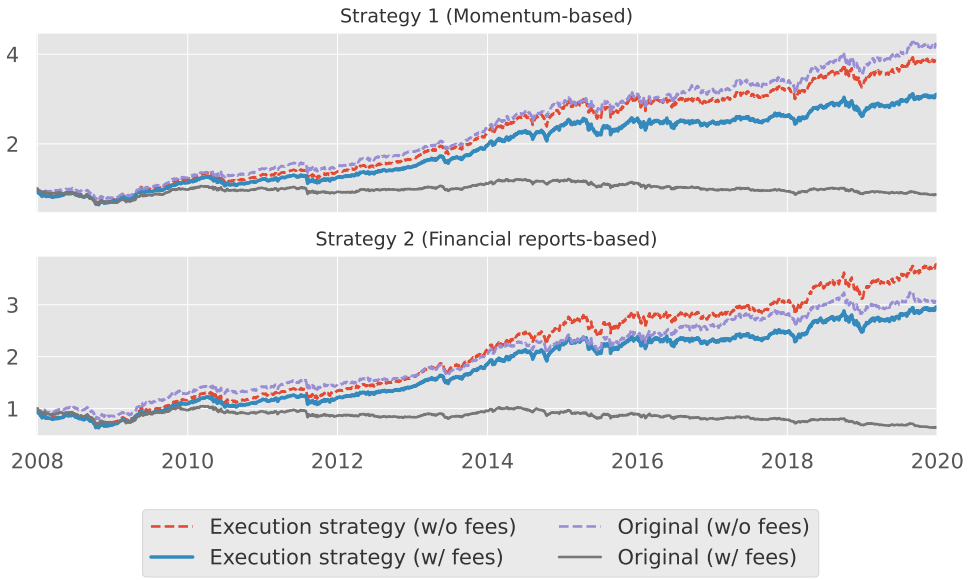


Figure 4.4: In-sample cumulative performance of the execution strategy with and without commission fees, which are set to 0.2% per order.

Table 4.6: Performance of *Strategy 2*, the financial reports-based strategy, with or without using the execution strategy and with or without trading fees. *OoS*: Out-of-sample, *IS*: In-sample.

	With fees				Without fees			
	Execution strategy		Original		Execution strategy		Original	
	IS	OoS	IS	OoS	IS	OoS	IS	OoS
Cumulative return	194.76%	35.97%	-36.41%	2.08%	276.82%	41.91%	208.81%	38.23%
CAGR%	9.4%	14.14%	-3.69%	0.89%	11.66%	16.26%	9.83%	14.95%
Sharpe	0.77	0.99	-0.26	0.13	0.93	1.12	0.85	1.09
Max drawdown	-37.24%	-24.67%	-39.94%	-26.37%	-34.52%	-24.61%	-24.18%	-25.67%
Days longest drawdown	643	155	3576	276	509	139	329	179
Mean drawdown	-2.07%	-2.04%	-10.57%	-3.42%	-1.99%	-1.84%	-2.04%	-1.84%
Days mean drawdown	29	17	546	44	26	15	28	17
Number of trades	7770	1484	8308	1560	7770	1484	8308	1560

The simulated returns from both strategies display high degrees of correlation before and after the execution strategy is applied. *Strategy 1* is 84.0% correlated with itself after application, and for *Strategy 2* $R^2_{\%} = 84.6\%$. However, higher correlations are measured between *Strategy 1* and *Strategy 2* after both have applied the execution strategy, with $R^2_{\%} = 98.6\%$. The original strategies are 87.5% correlated. Excess return over the original strategies displays autocorrelation over a day. It is strongest with fees applied, where in *Strategy 1* excess profits are 15.0% autocorrelated ($p < 10^{-5}$) and in *Strategy 2*, they are 18.5% autocorrelated ($p < 10^{-5}$). In other words, the execution strategy is more likely to continue outperforming the original trading strategy if it also outperformed the previous day. Autocorrelation is also found in the classification accuracy of the meta-classifier, where

Table 4.5: Performance of *Strategy 1*, the momentum-based strategy, with or without using the execution strategy and with or without trading fees. *OoS*: Out-of-sample, *IS*: In-sample.

	With fees				Without fees			
	Execution strategy		Original		Execution strategy		Original	
	IS	OoS	IS	OoS	IS	OoS	IS	OoS
Cumulative return	209%	35.59%	-13.33%	5.33%	291.79%	42.48%	321.38%	36.3%
CAGR%	10.57%	14.36%	-1.18%	2.26%	12.02%	16.46%	12.7%	14.26%
Sharpe	0.8	1.03	-0.04	0.23	0.96	1.16	1.07	1.08
Max drawdown	-35.95%	-22.71%	-35.7%	-24.44%	-33.73%	-22.76%	-27.54%	-23.51%
Days longest drawdown	624	155	1794	276	508	139	333	136
Mean drawdown	-2.01%	-2.12%	-5.93%	-4.86%	-1.97%	-1.99%	-1.82%	-2.39%
Days mean drawdown	29	17	196	75	26	16	26	21
Number of trades	7716	1443	8260	1517	7716	1443	8260	1517

correctly- or incorrectly classified future price directions are likely to follow other correct or incorrect classifications. However, at 5.5% at lag one, this effect is not as strong as the autocorrelation found in excess returns resulting from backtesting simulations.

We also check whether market conditions affect the over-performance of the execution strategy, as well as the prediction ability of the meta-classifier. Regressing mean correlations of assets towards the cap-weighted market with execution strategy excess return does not display any linear dependence. We also perform logistic regression, mapping mean market correlation to whether or not predictions of the meta-classifier are correct, finding no significant coefficients. When mean market correlation is replaced with the rolling explanatory power of the factor model and the same regressions are computed, we again find no statistically significant links.

Chapter 5

Discussion

5.1 Attribution of execution strategy over-performance

Table 4.5 shows that the total number of trading signals from *Strategy 1* is decreased after the application of the execution strategy. The reductions are by 4.9% and 6.6%, in- and out-of-sample, respectively. Similarly, in Table 4.6, we show that *Strategy 2* experiences reductions of trading signals by 6.5% in-sample and 4.9% out-of-sample. This suggests that the execution strategy compresses the original trading strategy while retaining the same performance level. This effect can partly explain why the execution strategy vastly over-performs the original strategy under trading costs: The total number of trades is significantly reduced, as is the total sum of trading costs that chips away at the strategy's profitability. Note that our out-of-sample period coincides with a period of market prosperity, which may positively skew the performance indicators in Table 4.5 and Table 4.6. When testing a machine learning model, one typically expects the out-of-sample performance to decrease due to the model over-fitting to the training data set. The extent of this problem is hard to measure, but it is likely to be improved in our case as a result of adapting purged cross-validation to the stacking algorithm. De Prado (2018) discusses how purging training folds improves data leakage in self-similar data, which leads to less biased validation fold performances.

5.2 Impact of market distress on excess profitability

It is well known in the finance literature that periods of market distress cause average stock correlations to increase. Preis et al. (2012) document that correlations within DJIA[1] scale linearly with the returns of the index over various time scales. Baig and Goldfajn (1999) study correlations in currency- and stock markets during and after the Asian financial crisis of 1997, showing that correlations increased

[1]The Dow Jones Industrial Average, a market index with 30 constituents intended to reflect the state of the US economy. Whether or not it accomplishes that is controversial (Haensly et al. (2001), Jr. and Allen (1979) and Shoven and Sialm (2000)).

substantially between markets, suggesting that market distress is not restricted to single countries but also propagates to similar markets. Our execution strategy does not directly consider market conditions through any set rule. Considering that our experimental implementation covers a multi-country asset universe, it is natural to question whether market distress worsens its performance compared to the original trading strategy. In Section 4.4.2 we show that there is no evidence of this when we use a similar approach as Preis et al. (2012) when estimating average correlations (A slight difference can be argued, considering that DJIA is a price-weighted average and our index is market cap-weighted). Additionally, no connection is found between incorrect meta-classifier predictions and market distress. This evidence implies that our execution strategy does not under- or over-perform the original trading strategy as a function of market distress, indicating that it is robust against adverse market conditions.

5.3 Explanatory power of exposure matrix weighting methods

In the results section we evaluate the explanatory power of the factor model using either unit weighted or exponentially weighted beta loadings in the factor exposure matrices. Guerard Jr (2009) employs unit weighted loadings and documents overall explanatory power of $R^2_{\%} = 30\%$. Using the specifications of our factor model implementation, its explanatory power with traditional unit weights as well as beta weights described in Section 3.1.1, we see from the resulting R^2-values in Table 4.3 that our implementation yields a total explanatory power of 31.8%, which is similar to Guerard Jr (2009) when unit weighted loadings are used. However, we observe increased explanatory power when beta weightings are used, measuring 34.9%. When measured with respect to sectors and countries, explanatory powers are also higher for beta weights in all counts in Table 4.3. Note that Guerard Jr (2009) implements a global multi-factor model, which differs from our European-based model. Connor (1995) reports explanatory powers of 42.6% for a US-based multi-factor model calculated using cross-sectional regressions. However, the model of Connor includes twelve fundamental style factors, and our model only includes seven, which can explain some of the differences in explanatory power. It is also possible that the idiosyncratic return components (see Equation (3.1)) of European stocks are larger than in their North-American counterparts. However, we do not have quantitative evidence to support this conjecture at the time of writing.

5.4 Risks

Even though we observe an increase in risk-adjusted return in trading strategies using our execution framework, this does not imply that the additional profits are

not risk-less. This section identifies several possible risk sources associated with our approach.

5.4.1 Autocorrelation of meta-classifier accuracy

Since over-performance is autocorrelated, we can expect the execution strategy to have consecutive days of over- or under-performance. We also show in Section 4.4.2 that meta-classifier prediction accuracy is autocorrelated for one lag. This leads to increased risk after the day after a misclassified price direction, where the consecutive day also is more likely to be misclassified. Since classification features are calculated from market open information, and the labels are calculated from market close prices of the next day, the true value of the previous classification is not known until after the next classification is made. In cases where significant autocorrelation can be found in longer time lags, we can address this problem, e.g., by using an autoregressive method, such that future losses can be reduced (Shumway et al. (2000)). However, this is not implemented in the current iteration of the execution strategy, meaning that this downside risk currently persists for meta-classifiers with longer autocorrelation.

5.4.2 Increased correlation across trading strategies

Systematic fund managers may simultaneously deploy multiple trading strategies to diversify risk. This is contingent on the assumption that strategies are independent, such that the variance of the resulting portfolio is reduced. Sorensen et al. (2004) details the combination of multiple sources of alpha by computing optimal weights for each of these alpha sources, such that the linear combination is referred to as the composite alpha source. It is reasonable to think that such an approach could also be applied to the trading strategies we use to benchmark our execution strategy, to construct a portfolio with a Sharpe ratio higher than each of its parts. A problem can arise, however, if we combine highly correlated strategies. Sorensen et al. expresses the expected information ratio of the composite alpha as a multivariate function that decreases when correlation coefficients increase. As we showed in Section 4.4.2, the application of the execution strategy on *Strategy 1* and *Strategy 2* increases across-strategy correlation from 87.5% to 98.6%. Therefore, an increase in profitability of different strategies after applying the execution strategy might lead to a composite alpha that is worse than the composite alpha resulting from combining the original trading strategies.

Chapter 6

Conclusion

In this thesis, we describe a framework for optimizing the execution time for orders of a long time horizon trading strategy, allowing orders to be delayed by E days. The framework, referred to as the execution strategy, is based on a novel adaptation of a stacking classifier, replacing regular K-fold cross-validation with purged K-fold cross-validation. This enables the use of self-similar classifier features, which are often found in finance. Using a LightGBM and a random forest classifier as base classifiers in the stacking ensemble, we achieve a prediction accuracy of 52.6%, and an F1-score of 52.3%, when predicting the direction of asset returns of the next trading day.
Features of the base classifiers are based on information from a BARRA-style multi-factor model, which we use for estimating asset returns and style factor exposures. Rolling deviations between the return estimates from the factor model and true returns are also included as data features. We show that using exponentially weighted factor-beta values as loadings of the factor model produce an overall explanatory power of 34.9%, which is higher than with unit weights (31.8%), which are traditionally used in the literature.
Different trading strategies respond differently to a delay in market execution. We use the estimated loss associated with delayed execution and the distribution of probabilities of meta-classifier predictions to estimate the optimal parameters of the execution strategy. To test the performance of the execution model, we simulate two different trading strategies, finding significant profit increases in both strategies when simulated with trading fees of 0.2% per trade ($p = 0.0187$ and $p = 0.006$), and a mean increase in Sharpe ratios of 0.75. We find a mean improvement in Sharpe ratios of 0.02 when simulating without trading fees but find insignificant changes in profitability. However, we observe a mean decrease in total amounts of trades of 4.9%, out-of-sample, which suggests that the execution strategy can compress trading strategies without loss of profitability. Our results imply that the execution strategy is effective for several different trading strategies, being a viable approach for augmenting human reasoning with machine intelligence.

Chapter 7

Future work

We can think of a number of ways this work can be developed.
It is not essential that the execution model uses a factor model for parts of its features. The execution framework supports any binary classification model that predicts future asset returns. Therefore, a way of addressing the problem of increased correlation between strategies, as discussed in Section 5.4.2, could be to let different, independent classifiers determine the execution of each strategy. Alternatively, one could swap the factor model for another predictive model, such that the structure and feature names would be identical but with different values for model-true deviations (see Table 3.1).
In the implementation discussed in this thesis we use daily data frequency. We think it is reasonable to assume that the execution strategy also can be effective for other time-frames. The amount of orders of a trading strategy typically increases as its data frequency increases, as high-frequency trading is an example of. If the execution strategy retains its ability to lower the number of orders while performing as well, it might enable the use of strategies that are ordinarily unprofitable under trading costs.

Bibliography

Almgren, Robert and Neil Chriss (2001). 'Optimal execution of portfolio transactions'. In: *Journal of Risk* 3, pp. 5–40.

Altmann, André, Laura Toloşi, Oliver Sander and Thomas Lengauer (2010). 'Permutation importance: a corrected feature importance measure'. In: *Bioinformatics* 26.10, pp. 1340–1347.

Amihud, Yakov and Haim Mendelson (1986). 'Asset pricing and the bid-ask spread'. In: *Journal of financial Economics* 17.2, pp. 223–249.

Asness, Clifford S, R Burt Porter and Ross L Stevens (2000). 'Predicting stock returns using industry-relative firm characteristics'. In: *Available at SSRN 213872*.

Baig, Taimur and Ilan Goldfajn (1999). 'Financial market contagion in the Asian crisis'. In: *IMF staff papers* 46.2, pp. 167–195.

Belloni, Alexandre, Victor Chernozhukov and Christian Hansen (May 2014). 'High-Dimensional Methods and Inference on Structural and Treatment Effects'. In: *Journal of Economic Perspectives* 28.2, pp. 29–50. DOI: 10.1257/jep.28.2.29. URL: https://www.aeaweb.org/articles?id=10.1257/jep.28.2.29.

Bertsimas, Dimitris and Andrew W Lo (1998). 'Optimal control of execution costs'. In: *Journal of financial markets* 1.1, pp. 1–50.

Bowden, Roger J and Darrell A Turkington (1990). *Instrumental variables*. 8. Cambridge university press.

Breiman, Leo (1996). 'Bagging predictors'. In: *Machine learning* 24.2, pp. 123–140.

Breiman, Leo, Jerome H Friedman, Richard A Olshen and Charles J Stone (1984). *Classification and regression trees*. Routledge.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al. (2020). 'Language models are few-shot learners'. In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Bühlmann, Peter (2012). 'Bagging, boosting and ensemble methods'. In: *Handbook of computational statistics*. Springer, pp. 985–1022.

Butler, John L and Charles H Sherman (2016). *Transducers and arrays for underwater sound*. Springer.

Chen, Tianqi and Carlos Guestrin (2016). 'Xgboost: A scalable tree boosting system'. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Chordia, Tarun, Avanidhar Subrahmanyam and V Ravi Anshuman (2001). 'Trading activity and expected stock returns'. In: *Journal of financial Economics* 59.1, pp. 3–32.

Connor, Gregory (1995). 'The three types of factor models: A comparison of their explanatory power'. In: *Financial Analysts Journal* 51.3, pp. 42–46.

De Prado, Marcos Lopez (2018). *Advances in financial machine learning*. John Wiley & Sons.

Eschenbach, Warren J von (2021). 'Transparency and the black box problem: Why we do not trust AI'. In: *Philosophy & Technology* 34.4, pp. 1607–1622.

Fama, Eugene F. and Kenneth R. French (1992). 'The Cross-Section of Expected Stock Returns'. In: *The Journal of Finance* 47.2, pp. 427–465. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2329112.

Fama, Eugene F. and James D. MacBeth (1973). 'Risk, Return, and Equilibrium: Empirical Tests'. In: *Journal of Political Economy* 81.3, pp. 607–636. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/1831028 (visited on 13th May 2022).

Feng, Guanhao, Stefano Giglio and Dacheng Xiu (2020). 'Taming the factor zoo: A test of new factors'. In: *The Journal of Finance* 75.3, pp. 1327–1370.

Flagel, Lex, Yaniv Brandvain and Daniel R Schrider (2019). 'The unreasonable effectiveness of convolutional neural networks in population genetic inference'. In: *Molecular biology and evolution* 36.2, pp. 220–238.

Frazzini, Andrea and Lasse Heje Pedersen (2014). 'Betting against beta'. In: *Journal of Financial Economics* 111.1, pp. 1–25.

Goodhart, Charles AE (1984). 'Problems of monetary management: the UK experience'. In: *Monetary theory and practice*. Springer, pp. 91–121.

Gu, Shihao, Bryan Kelly and Dacheng Xiu (Feb. 2020). 'Empirical Asset Pricing via Machine Learning'. In: *The Review of Financial Studies* 33.5, pp. 2223–2273. ISSN: 0893-9454. DOI: 10.1093/rfs/hhaa009. eprint: https://academic.oup.com/rfs/article-pdf/33/5/2223/33209812/hhaa009.pdf. URL: https://doi.org/10.1093/rfs/hhaa009.

Guerard Jr, John B (2009). *Handbook of portfolio construction: contemporary applications of Markowitz techniques*. Springer Science & Business Media.

Haensly, Paul J, Niranjan Tripathy and Daniel Peak (2001). 'Tracking error in the Dow Jones Industrial Average versus alternative market indices: New evidence'. In: *Quarterly Journal of Business and Economics*, pp. 101–116.

Heston, Steven L and K Geert Rouwenhorst (1994). 'Does industrial structure explain the benefits of international diversification?' In: *Journal of financial economics* 36.1, pp. 3–27.

Ho, Tin Kam (1995). 'Random decision forests'. In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE, pp. 278–282.

Hunter (1986). *Single Exponential Smoothing*. [Cited 2021-11-15]. URL: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc431.htm.

Jegadeesh, Narasimhan and Sheridan Titman (1993). 'Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency'. In: *The Journal*

of Finance 48.1, pp. 65–91. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2328882 (visited on 14th May 2022).

Jiang, Minqi, Jiapeng Liu, Lu Zhang and Chunyu Liu (2020). 'An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms'. In: *Physica A: Statistical Mechanics and its Applications* 541, p. 122272.

Jr., Hartman L. Butler and J. Devon Allen (1979). 'The Dow Jones Industrial Average Re-Reexamined'. In: *Financial Analysts Journal* 35.6, pp. 23–30. DOI: 10.2469/faj.v35.n6.23. eprint: https://doi.org/10.2469/faj.v35.n6.23. URL: https://doi.org/10.2469/faj.v35.n6.23.

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žıdek, Anna Potapenko et al. (2021). 'Highly accurate protein structure prediction with AlphaFold'. In: *Nature* 596.7873, pp. 583–589.

Kaufman, Shachar, Saharon Rosset and Claudia Perlich (Jan. 2011). 'Leakage in Data Mining: Formulation, Detection, and Avoidance'. In: vol. 6, pp. 556–563. DOI: 10.1145/2020408.2020496.

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye and Tie-Yan Liu (2017). 'LightGBM: A Highly Efficient Gradient Boosting Decision Tree'. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Kleinbaum, David G, K Dietz, M Gail, Mitchel Klein and Mitchell Klein (2002). *Logistic regression*. Springer.

Krauss, Christopher, Xuan Anh Do and Nicolas Huck (2017). 'Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500'. In: *European Journal of Operational Research* 259.2, pp. 689–702.

Kuhn, Harold W. and Albert W. Tucker (2014). 'Nonlinear programming'. In: *Traces and emergence of nonlinear programming*. Springer, pp. 247–258.

Leal, Laura, Mathieu Laurière and Charles-Albert Lehalle (2020). 'Learning a functional control for high-frequency finance'. In: *arXiv preprint arXiv:2006.09611*.

Menchero, Jose, Andrei Morozov and Peter Shepard (2010). 'Global Equity Risk Modeling'. In: *Handbook of Portfolio Construction*. Ed. by John B. Guerard. Boston, MA: Springer US, pp. 439–480. ISBN: 978-0-387-77439-8. DOI: 10.1007/978-0-387-77439-8_15. URL: https://doi.org/10.1007/978-0-387-77439-8_15.

Menchero, Jose and Zoltán Nagy (2013). *Risk and Return of Factor Portfolios: The Impact of Regression Weighting*. URL: https://www.msci.com/documents/10199/cb704602-b417-4077-a277-c170501a70d7.

Nelder, John Ashworth and Robert WM Wedderburn (1972). 'Generalized linear models'. In: *Journal of the Royal Statistical Society: Series A (General)* 135.3, pp. 370–384.

Nocedal, J and S J Wright (2006). *Numerical optimization*. Springer.

Permutation Importance vs Random Forest Feature Importance (MDI) (2022). https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html.

Preis, Tobias, Dror Y Kenett, H Eugene Stanley, Dirk Helbing and Eshel Ben-Jacob (2012). 'Quantifying the behavior of stock correlations under market stress'. In: *Scientific reports* 2.1, pp. 1–5.

Rokach, Lior and Oded Maimon (2005). 'Decision trees'. In: *Data mining and knowledge discovery handbook*. Springer, pp. 165–192.

Rosenberg, Barr (1974). 'Extra-Market Components of Covariance in Security Returns'. In: *Journal of Financial and Quantitative Analysis* 9.2, pp. 263–274. DOI: 10.2307/2330104.

Sharpe, William F (1998). 'The Sharpe Ratio'. In: *Streetwise–the Best of the Journal of Portfolio Management*, pp. 169–185.

Shepard, Peter G (2008). 'Integrating multi-market risk models'. In: *Journal of Risk* 10.2, p. 25.

Shoven, John B and Clemens Sialm (2000). 'The dow jones industrial average: the impact of fixing its flaws'. In: *The Journal of Wealth Management* 3.3, pp. 9–18.

Shumway, Robert H, David S Stoffer and David S Stoffer (2000). *Time series analysis and its applications*. Vol. 3. Springer.

Skagemo, Markus, Per Christian Moan and Adil Rasheed (2021). 'Improved market execution of long-term time horizon trading signals using short-term residual reversal'. In: *Unpublished*.

Slater, Morton (2013). 'Lagrange Multipliers Revisited'. In: *Traces and Emergence of Nonlinear Programming*, pp. 293–306. DOI: 10.1007/978-3-0348-0439-4_14.

Sonoda, Sho and Noboru Murata (2017). 'Neural network with unbounded activation functions is universal approximator'. In: *Applied and Computational Harmonic Analysis* 43.2, pp. 233–268.

Sorensen, Eric H, Edward Qian, Robert Schoen and Ronald Hua (2004). 'Multiple alpha sources and active management'. In: *The Journal of Portfolio Management* 30.2, pp. 39–45.

Stone, Mervyn (1974). 'Cross-validatory choice and assessment of statistical predictions'. In: *Journal of the royal statistical society: Series B (Methodological)* 36.2, pp. 111–133.

Tang, Jiliang, Salem Alelyani and Huang Liu (2014). 'Data classification: algorithms and applications'. In: *Data Mining and Knowledge Discovery Series*, pp. 37–64.

Tukey, John W. (1962). 'The Future of Data Analysis'. In: *The Annals of Mathematical Statistics* 33.1, pp. 1–67. DOI: 10.1214/aoms/1177704711. URL: https://doi.org/10.1214/aoms/1177704711.

Vasicek, Oldrich A (1973). 'A note on using cross-sectional information in Bayesian estimation of security betas'. In: *The Journal of Finance* 28.5, pp. 1233–1239.

Welch, Bernard L (1947). 'The generalization of 'STUDENT'S'problem when several different population varlances are involved'. In: *Biometrika* 34.1-2, pp. 28–35.

Wolpert, David H (1992). 'Stacked generalization'. In: *Neural networks* 5.2, pp. 241–259.

Appendix A

Additional experiment content

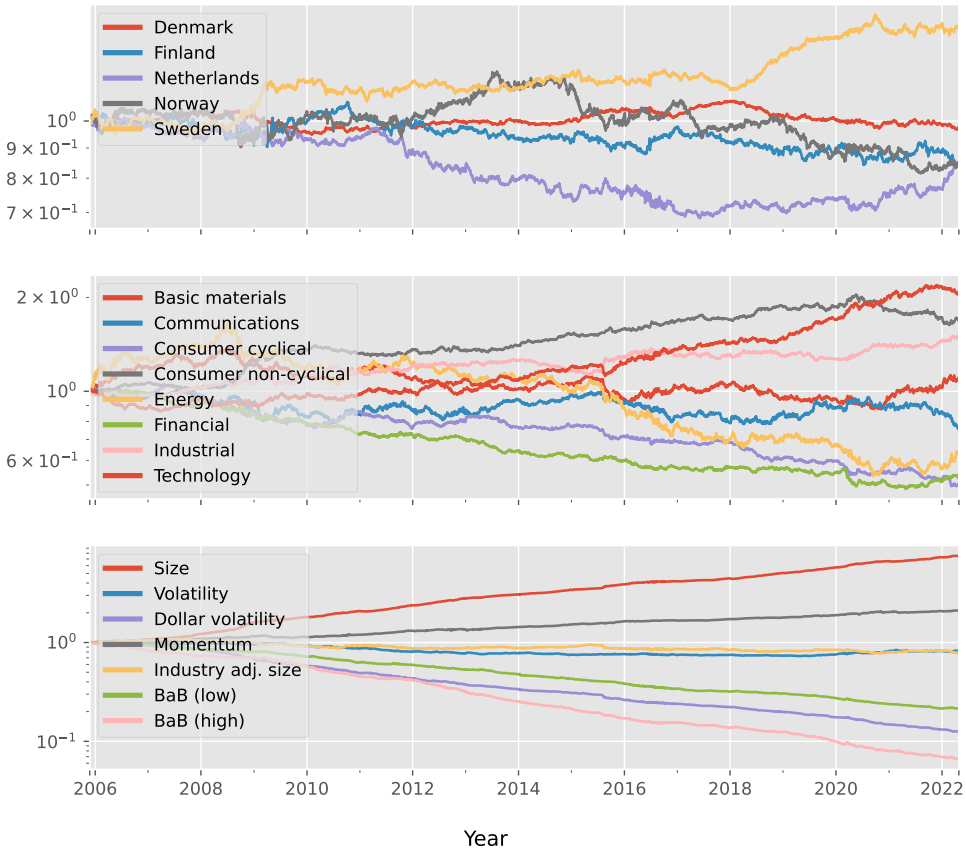
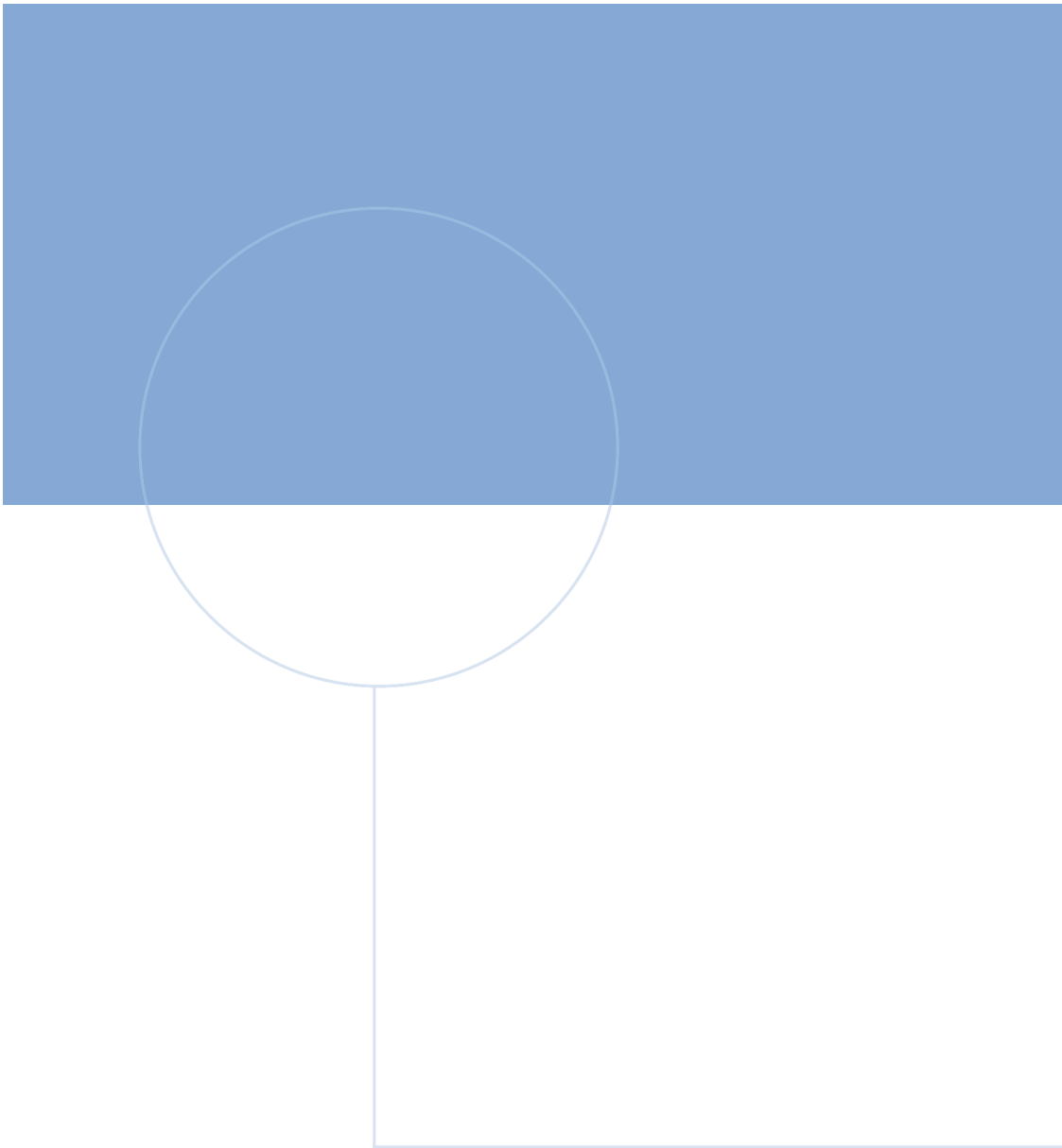


Figure A.1: Plot of cumulative factor realizations using the factor model specifications described in Section 4.2.3. Y-axis scale as well as factor scales are logarithmic.

Table A.1: Sectors, countries and style factors of the assets used in the experiment.

Factor type	Name
Country	Denmark
	Finland
	Netherlands
	Norway
	Sweden
Sector	Basic materials
	Communications
	Consumer cyclical
	Consumer non-cyclical
	Energy
	Financial
	Industrial
	Technology
Style	Described in Section 3.1.2



NTNU

Norwegian University of
Science and Technology