Using Random Forests in conintegrated pairs trading

By: Reimer Meulenbeek

Supervisor Radboud University:
Prof. dr. E.A. Cator

Supervisors FRIJT BV:
Dr. O. de Mirleau
Drs. M. Meuwissen

November 5, 2017
Abstract

In this thesis we will use Random Forests to define a trading strategy. Using this powerful machine learning technique, we will try to predict the daily price changes of financial products that move similarly over the long term, so-called cointegrated pairs. We propose a way to adjust our portfolio based on these predictions, while limiting our risk. Firstly, we test our strategy on data generated from a model that mimics these kinds of financial products. After promising results, we test our strategy on the Dutch AEX index and the German DAX index. From our backtests we see that our strategy outperforms both indices in terms of Sharpe ratio. Using a backtesting period of 10 years up to mid 2017 we find an annualized Sharpe ratio of about 0.7, before transaction costs and ignoring the risk-free return rate.
Acknowledgements

I would like to thank my supervisor Eric Cator for his time and very insightful input. I would also like to thank Olivier and Marc from FRIJT BV. I had a wonderful time writing my master thesis at their company.
## Contents

1 **Introduction** 8  
   1.1 Cointegrated pairs ........................................... 9  
   1.2 Sharpe ratio .................................................. 10  

2 **Random Forest** 12  
   2.1 Classification Tree ........................................... 12  
   2.2 Regression Tree ............................................... 15  
   2.3 Cross-validation ............................................... 18  
   2.4 Random Forest ................................................. 22  

3 **The model** 24  
   3.1 Introduction of the model .................................... 24  
   3.2 Input variable selection ...................................... 29  
   3.3 More data points ................................................ 30  
   3.4 Variable importance ............................................ 31  

4 **Strategy** 32  
   4.1 General Framework ............................................ 32  
   4.2 Determining the optimal weights ............................... 36  
   4.3 Hedging .......................................................... 37  
   4.4 First test of our strategy ..................................... 39  
   4.5 Introducing more constraints ................................... 40  
   4.6 Predicting the covariance matrix ............................... 44  
   4.7 More data ......................................................... 48  
   4.8 A closer look at parameter selection ............................ 49  
   4.9 The ARCH model ................................................ 52  
   4.10 More input variables ............................................ 56  
   4.11 Importance of random projections ............................. 58  
   4.12 Summary ........................................................ 59  

5 **Real market data** 62  
   5.1 Backtesting setup .............................................. 62  
   5.2 The data .......................................................... 63  
   5.3 First Results ..................................................... 64  
   5.4 Performance of $\hat{\Sigma}$ ..................................... 67  
   5.5 A closer look ...................................................... 70  
   5.6 Comparison with simple predictor ............................... 74  
   5.7 Empirical estimate of $\Sigma$ .................................... 75  
   5.8 More indices ....................................................... 78  
   5.9 Omitting the random projections ................................. 80  
   5.10 Implementing a real trading strategy and estimating the associated costs .................................................. 82  
   5.11 Literature review ................................................ 85  
   5.12 Improvements to our strategy .................................. 86  

6 **Conclusion** 88
1 Introduction

In the early 1980's American bank Morgan Stanley brought together a group of computer scientists, mathematicians and traders to work on a secret project. A couple of years later, the group had finished the first algorithm based on (what we would now call) pairs trading. The strategy was very profitable during the first year of its implementation, but profits declined shortly afterwards. Nevertheless, it marked the beginning of a new era in trading. Instead of traders shouting their buy and sell orders on the trading floor, more and more trading decisions where autonomously made by algorithms. Most algorithmic trading strategies are based on statistical arbitrage. This is an umbrella term of strategies, with pairs trading as the first kind of statistical arbitrage. Statistical arbitrage is often based on mean reversion. It makes use of processes that will revert to some mean in the long term, but can divert from it on the short term. In this thesis, we will mostly look at the Dutch AEX index and the German DAX index pair. These are examples of financial products that are used in statistical arbitrage, since they are heavily correlated and possibly cointegrated. We will explain the latter notion in the next subsection. Traditional techniques in pairs trading are often based on technical indicators. These are metrics derived of the historical prices of the product.

In this thesis we will use machine learning to find a profitable trading strategy on the AEX/DAX pair. More specifically, we will make use of Random Forests. Random Forests are a powerful machine learning technique designed to do classification or regression based on very many input variables. We hope that at day \( t \), the Random Forest can make a useful prediction of the prices at day \( t + 1 \) using all historical data as input.

In order to do so, we need to understand how Random Forests work. This is explained in section 2. In section 3 we introduce a model in which we try to mimic financial product pairs like the AEX and the DAX. We will test the ability of the Random Forest to predict the next day prices in this model. In section 4 we define our trading strategy, which is not only based on the prediction of the next day prices, but also on a prediction of the covariance between our prediction and the actual data. Finally, in section 5 we will test our strategy on real market data.
1.1 Cointegrated pairs

Cointegration is often explained by the following example. Suppose a man leaves a pub with his dog. Because the man drank too much, his path is a random walk. If the dog does not feel the pull of the leash he will also wander around aimlessly, so his path is also a random walk. While the paths of the man and his dog might be independent random walks in the short term, the man and the dog can never get more apart than the length of the leash. In this example the man and the dog are a cointegrated pair.

Formally, we will call two time series variables $X_t, Y_t$ cointegrated if there is a certain linear combination $Z_t = X_t + cY_t$ such that this linear combination is a weakly stationary process. In turn, $Z_t$ is a weakly stationary process if $\mathbb{E}(Z_t)$ and $\mathbb{E}(Z_tZ_{t+h})$ exist and are finite and do not depend on $t$ for every integer $h$. Note that cointegrated time series do not need to be correlated. Indeed, in our example the path of the man and his dog where independent random walks. Moreover, correlation is often measured between daily differences $X_t - X_{t-1}$ of two time series while cointegration is about linear combinations of those time series. We will primarily use the AEX and DAX index, which are correlated in terms of daily differences $X_t - X_{t-1}$.

Suppose stocks $A$ and $B$ are cointegrated, denoting their prices as $S_A, S_B$ there thus is some constant $c$ such that $S_A - cS_B$ is a stationary process with some mean $\mu$. Suppose now that the value of $S_A - cS_B$ is smaller than $\mu$ at some time $t$. If we buy the linear combination we will certainly make a profit, since the mean $\mu$ is constant in time. Note that buying or selling the linear combination in this context means that we buy one product and short sell another product. Short selling is the act of selling a borrowed product and buying it back at a later time. If the price of the product went down, the short sell was profitable.
1.2 Sharpe ratio

In this text we will often use the Sharpe ratio. The Sharpe ratio was introduced by American economist William Sharpe in 1994 (see [1]) and is widely used in measuring the performance of trading strategies. Let $R_a$ denote the daily return for a strategy $a$ and let $R_b$ denote the daily return for some riskless asset $b$. For this asset $b$, risk-free bonds issued by governments are usually chosen. The daily Sharpe ratio measures the excess return compared to $b$ per unit of deviation and is defined as

$$\text{daily Sharpe ratio} = \frac{\mathbb{E}(R_a - R_b)}{\sqrt{\text{var}(R_a - R_b)}}.$$  \hspace{1cm} (1)

When we refer the Sharpe ratio in this thesis, we are always talking about the annualized Sharpe ratio. For this definition, we assume that the standard deviation of the yearly returns is the square root of the number of trading days times the standard deviation of the daily returns. Naturally, the expected yearly return is just the expected daily return multiplied by the same number. Since the number of trading days in most years is 252, we define the Sharpe ratio as

$$\text{Sharpe ratio} = \sqrt{252} \cdot \frac{\mathbb{E}(R_a - R_b)}{\sqrt{\text{var}(R_a - R_b)}}.$$ \hspace{1cm} (2)

In this thesis we will often test strategies on historical data. We call this backtesting. Suppose we run a backtest from time $t = 1$ to time $t = T$. From this backtest, we can calculate our daily profits and losses $p = (p_1, p_2, \ldots, p_T)$. In calculating the Sharpe ratio we always omit the riskless asset $b$. We do this because it is often hard to calculate, and since it is not a big factor anyway. Given our daily profits and losses $p$, we can thus calculate the Sharpe ratio as

$$\text{Sharpe ratio} = \sqrt{252} \cdot \frac{\bar{p}}{\sigma(p)},$$ \hspace{1cm} (3)

where $\bar{p}$ denotes the sample mean and $\sigma(p)$ denotes the sample standard deviation. Note that the Sharpe ratio is independent on the amount of money invested. Indeed, if a strategy would invest a factor $\lambda$ more money, both the daily profits and the standard deviation in daily profits would scale with $\lambda$. This leads to the same Sharpe ratio.
2 Random Forest

In this section we will introduce the Random Forest. Random Forests are a machine learning technique that can be used for doing either classification or regression. A Random Forest averages predictions of many tree predictors into a single prediction. In order to understand the Random Forest, we will thus first need to define these tree predictors. We will do this in the next two subsections, for both the classification case and the regression case. We will illustrate these techniques on data sets we generate from a certain model. After that we will explain how to properly select optimal parameters in constructing a tree predictor. Finally we will introduce the Random Forest.

2.1 Classification Tree

A Classification Tree is a method to classify based on several input variables. Formally, a classification tree $T$ maps $N$ input variables $X = (x_1, \ldots, x_N)$ to a prediction $\hat{y}$, where $\hat{y}$ belongs to a discrete set of labels of size $M$, i.e.

$$T : X \mapsto \hat{y},$$

where $\hat{y} \in \{y_1, \ldots, y_M\}$. The input variables $x_i$ can be both categorical or continuous.

A classification tree can be viewed as a directed graph. A tree has multiple layers, with one starting node in the top layer. Each node creates either two child nodes one layer lower, or is an end node. Given a classification tree $T$ along with input variables $X = (x_1, \ldots, x_N)$, the prediction $\hat{y}$ can be found by starting at the top node of the graph and traveling a certain path until arriving at an end node. At each node that splits into two child nodes, the direction is determined by the value of some input variable $x_i$. The prediction $\hat{y}$ is given by the end node of the graph.

A classification tree is built (or ‘grown’) in a top-down fashion. Initially, all data points belong to the same parent node. The tree then grows by repeating the following steps

- For each node, find the best split. Do this by finding the best attribute $x_i$ and corresponding value $c$ to make the split. Which split is best depends on which purity measure is used. The data points in the particular node are then split using the criterion $x_i \leq c$ for continuous variables $x_i$, or $x_i = c$ for categorical variables. Each split creates two new nodes.
- Repeat until each node is pure, i.e. all nodes only consist of data points belonging to the same class. This class will be the prediction of our tree. Growing the tree can also be stopped earlier, for example at a certain depth, when a certain value of the impurity measure is reached, etc.. In that case, the prediction will be the majority vote of all classes at a leaf.

The impurity at each node can be defined in multiple ways. A popular impurity
measure is the Gini impurity $I$, defined by

$$I = \sum_{j=1}^{M} p_j (1 - p_j),$$  \hfill (5)$$

where $p_j$ denotes the fraction of data points belonging to class $y_j$. The lower the measure, the more data points belong to the same class. If the measure equals zero, the node is pure.

The Gini index of a split $I_{\text{split}}$ into two nodes is computed by

$$I_{\text{split}} = \frac{n_1}{n} I(1) + \frac{n_2}{n} I(2),$$  \hfill (6)$$

where $n_i$ represents the number of data points in each resulting node, $n$ the total number of data points before the split and $I(i)$ is the Gini index for each resulting node. The best split is the split with the lowest value of $I_{\text{split}}$.

Note that a Classification Tree, just like all other methods we will encounter in this chapter, can also handle higher dimensional target values. In (4), $\hat{y}$ is then a multidimensional vector. In such case, the best split is based on the average of the impurity measure of all target variables $y_i$.

We will now fit a Classification Tree on a generated data set. The data set is inspired by the Exclusive OR (XOR) Boolean operator. The operator requires two inputs and outputs 1 if either one of the inputs is 1, but not both.

The data set consists of 100 data points with attributes $x_1, x_2 \in \mathbb{R}$ and class $y \in \{0, 1\}$ and is generated as follows:

- The $(x_1, x_2)$ attributes for the first 25 points are independently drawn from a $N_2((0, 0), I_2)$ distribution and are labeled $y = 0$.
- The $(x_1, x_2)$ attributes for the next 25 points are independently drawn from a $N_2((5, 5), I_2)$ distribution and are also labeled $y = 0$.
- The $(x_1, x_2)$ attributes for the next 25 points are independently drawn from a $N_2((0, 5), I_2)$ distribution and are labeled $y = 1$.
- The $(x_1, x_2)$ attributes for the last 25 points are independently drawn from a $N_2((5, 0), I_2)$ distribution and are also labeled $y = 1$.

The data set is plotted in figure 1.

Before we fit the tree, we randomly divide the data in a training set $X_{\text{train}}$ and a test set $X_{\text{test}}$ with there corresponding labels $y_{\text{train}}$ and $y_{\text{test}}$. The training set consists of 80 data points, the test set consists of the remaining 20 data points.

The tree is now fitted on the training set. We do not restrict the tree in any way, this will result in a tree where the end nodes (also called leaves) are always pure. The tree is depicted in figure 2.

Starting at the first node, we see that the training set consists of 40 data points labeled 0 and 40 data points labeled 1. This is a coincidence, since we randomly
Figure 1: Scatter plot of the XOR data set. Points belonging to class 0 are labeled red, points belonging to class 1 are labeled blue.

divided the data into the training and test set. It is however recommended to have approximately the same ratio of the two classes in the training set as in the whole data set. This might otherwise lead to a biased tree.

We furthermore note that the tree makes different splits than one might expect. A human would probably only do two splits to make a predictor, namely the split $x_1 \leq 2.5$ followed by $x_2 \leq 2.5$. The tree would however never make the split $x_1 \leq 2.5$, since $I_{\text{split}}$ would be nearly maximal, since the two resulting nodes would contain both classes in a roughly 1 : 1 ratio. Note that we want the value of $I_{\text{split}}$ to be as low as possible.

To evaluate the performance of our tree, we define the accuracy for a set $X$ consisting of multiple instances $X_i$ of input variables as

$$\text{accuracy}(X) = \frac{\#\{X_i \in X \mid T(X_i) = y_i\}}{\#X},$$

i.e. the fraction of data points in the set $X$ of which the label is correctly predicted.

Since all the leaf nodes are pure, we must have $\text{accuracy}(X_{\text{train}}) = 1$. Note that this does not need to be the case if we put some restrictions on our tree. If we set the maximum depth (i.e. the maximum number of splits before reaching a leaf) of our tree to 3, the last split in figure 2 could not be made. We would thus end up with a leaf that contains 8 data points labeled 0 and one data point labeled 1. In such a case, the prediction would be the class that occurs most at the node, in this case class 0. If we thus restrict our maximum depth of the tree
to 3, we would have an accuracy on the training set of $\text{accuracy}(X_{\text{train}}) = \frac{79}{80}$. It is usually not a good idea to allow the tree to grow until all leaves are pure, since it would mostly always lead to overfitting on the training set.

On the test set, we find

$$\text{accuracy}(X_{\text{test}}) = \frac{18}{20} = 0.9.$$  \hfill (8)

Our Classification Tree thus performs quite well in predicting the class labels of the test set.

2.2 Regression Tree

A Regression Tree is a method to predict a continuous variable $y \in \mathbb{R}$ based on several input variables. The definition is the same as that of a classification tree
(see (4)), except for the fact that $y$ can now take any value.

For the impurity measure, the mean squared error (MSE) is often used and can be written as

$$\text{MSE}(S) = \frac{1}{|S|^2} \sum_{i \in S} \sum_{j \in S} \frac{1}{2}(y_i - y_j)^2,$$  \hspace{1cm} (9)

where $S$ is the index set of the data points at a node.

To determine the quality of a split, we simply add the MSE’s of the resulting nodes weighted by the number of data points per node and subtract that from the MSE at the node before the split

$$I_{\text{MSE}} = \text{MSE}(S) - \left( \frac{n_t}{n} \text{MSE}(S_t) + \frac{n_f}{n} \text{MSE}(S_f) \right),$$  \hspace{1cm} (10)

where $n$ denotes the number of data points before the split, $S_t$ the index set for which the split criterion $x_i \leq c$ is true, $n_t$ the number of data points in $S_t$, $S_f$ the index set for which the split criterion is false and $n_f$ the number of corresponding data points.

The best split is determined by the highest value of $I_{\text{MSE}}$, since it represents the biggest reduction in terms of MSE.

We modify our previous example to demonstrate the regression tree. We generate 100 data points in the following manner.

- The $(x_1, x_2)$ attributes for the first 25 points are independently drawn from a $\mathcal{N}_2((0, 0), I_2)$ distribution, with their target value $y$ independently drawn from a $\mathcal{N}(2, 1)$ distribution.
- The $(x_1, x_2)$ attributes for the next 25 points are independently drawn from a $\mathcal{N}_2((5, 5), I_2)$ distribution, with their target value $y$ also independently drawn from a $\mathcal{N}(2, 1)$ distribution.
- The $(x_1, x_2)$ attributes for the next 25 points are independently drawn from a $\mathcal{N}_2((0, 5), I_2)$ distribution, with their target value $y$ independently drawn from a $\mathcal{N}(-2, 1)$ distribution.
- The $(x_1, x_2)$ attributes for the last 25 points are independently drawn from a $\mathcal{N}_2((5, 0), I_2)$ distribution, with their target value $y$ also independently drawn from a $\mathcal{N}(-2, 1)$ distribution.

The data set is plotted in figure 3.

To demonstrate the importance of the input variables $X$, we are going to fit two decision trees. For the first one, we are going to use the same inputs as we did with the classification trees, i.e. $X = (x_1, x_2)$. For the second tree, we are going to include several other input variables, namely the distances to the four centers the points where generated around, i.e. $X = (x_1, x_2, r_{(0,0)}, r_{(0,5)}, r_{(5,0)}, r_{(5,5)})$, where $r_{(a,b)}$ represents the Euclidean distance to the point $(a, b)$, i.e $r_{(a,b)} = \sqrt{(x_1 - a)^2 + (x_2 - b)^2}$. The idea behind this is, of course, that for low values of for example $r_{(0,0)}$ the forest should predict higher values of $y$. This can be
considered cheating, since in practice we often don’t know how the data set is generated. We will see however that the performance improves by adding the new input variables.

For our particular data set, we find an MSE of 5.30 on the test set when using first tree, and an MSE of 3.11 when using the second tree. We are interested in how good these values are, so we look for other predictors to compare them with. We look at the predictor that does not even consider the variables in $X$ and just always predicts 0. Using this predictor leads to an MSE of 4.20, which is better than using the first tree! This can have multiples reasons, e.g. unfortunate choice of $X_{\text{train}}$ and $X_{\text{test}}$ or overfitting on the training set. To demonstrate this, we use the exact same data set, but now with a different (randomly generated) division of the data into the training set and the test set. We now find an MSE for the first tree of 2.72! The MSE for the second tree now is 2.67. This difference is much smaller. This demonstrates the importance of the choice of the training and test set.

The final predictor we consider is a predictor who knows how the data is generated, and it thus guesses $y = 2$ when points are drawn around (0, 0) and (5, 5), and $y = -2$ for points generated around (0, 5) and (5, 0). Using this predictor leads to an MSE of 1.08.

Note that the above results vary hugely when the data set is regenerated. To find the expected value of all MSE’s, we simulate 100 different data sets and apply our four different predictors. The result of this simulation study is shown...
in table 1.

<table>
<thead>
<tr>
<th>predictor</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree with $X = (x_1, x_2)$</td>
<td>3.21</td>
</tr>
<tr>
<td>Tree with $X = (x_1, x_2, r_{(0,0)}, r_{(0,5)}, r_{(5,0)}, r_{(5,5)})$</td>
<td>2.20</td>
</tr>
<tr>
<td>Predictor that always predicts $y = 0$</td>
<td>5.06</td>
</tr>
<tr>
<td>Predictor that knows the distribution</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Table 1: Results of 100 simulations from our XOR model with continuous $y$ values. The MSE is computed on the test set.

Note that for $Y \sim N(\mu, \sigma^2)$ we have that $\mathbb{E}(Y^2) = \mu^2 + \sigma^2$, so we expect $\text{MSE} = (\pm 2)^2 = 5$ for our predictor that always predicts $y = 0$. The MSE for the predictor that knows the distribution also coincides with our expectations, since $\mathbb{E}((Y-\mu)^2) = \sigma^2 = 1^2 = 1$.

From these simulations, we may indeed conclude that including the distances to the various centers improves the performance of our tree predictor.

When we generate a data set of 1000 data points, we expect our trees to perform better. Furthermore, we expect that the values of the MSE’s are less dependent on the choice of the training and test set. In table 2 the results are shown for a single data set with 1000 data points, where we used 10 randomly chosen pairs of training sets and test sets.

<table>
<thead>
<tr>
<th>predictor</th>
<th>mean MSE</th>
<th>standard deviation in MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree with $X = (x_1, x_2)$</td>
<td>1.41</td>
<td>0.28</td>
</tr>
<tr>
<td>Tree with $X = (x_1, x_2, r_{(0,0)}, r_{(0,5)}, r_{(5,0)}, r_{(5,5)})$</td>
<td>1.32</td>
<td>0.12</td>
</tr>
<tr>
<td>Predictor that always predicts $y = 0$</td>
<td>5.12</td>
<td>0.22</td>
</tr>
<tr>
<td>Predictor that knows the distribution</td>
<td>1.02</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Table 2: Performance of the different predictors on a single data set consisting of 1000 data points. The mean and standard deviation of the MSE is calculated by using 10 different training and test set pairs.

We indeed conclude that the performance of our predictors are less dependent of the choice of the training and test set when number of data points grows. Moreover, the value of the MSE reduces as the size of the data set gets bigger.

In practise, it is unusual to use different pairs of training and test set. It is more common that the test set is fixed. Sometimes the choice of the test set is natural, for example when the data is a time series. In such a case the training set is usually all the data up to some time $t$ and the test set all data points in the future.

### 2.3 Cross-validation

When fitting a classification or regression tree, we have to input certain parameters. Besides the choice of splitting criterion, the amount of pruning is very
important. Pruning a tree can be done in two ways: setting a maximum depth of the tree (i.e. a maximum number of splits before reaching a leaf), or a minimum number of data points below which a node does not get split anymore (and thus becomes a leaf). We will primarily use the latter, since it is easier to interpret.

Exploring which parameters lead to a good performance of the tree is called validation. Suppose we have a data set which is split into a training set which contains 80% of the data, and a test set containing the other 20%. We want to fit a tree to this data, but we don’t know what parameters we should use. It would be unfair to try out various values of the parameters and test the performance on the test set, since this would lead to overfitting on the test set. A common way to prevent this is \( k \)-fold cross-validation. In cross-validation, the training set is split into \( k \) parts. For certain values of our parameters, we fit \( k \) trees, each time using \( k - 1 \) parts of the training set. Each tree’s performance is measured on the remaining part of the training set that was not used in growing the tree. All \( k \) performance scores get averaged into a single performance score for each choice of the parameters. We can thus pick the parameter configuration that has the best performance score. After we picked our parameters, we can now test the performance of our tree on the test set. In this way, we do not overfit on the test set.

For fitting a regression tree to our generated data set of 1000 points, we want to control overfitting. Instead of setting a max depth parameter, we want to use a parameter that sets a minimum amount of data points in each leaf node. We will call this parameter \( \lambda_{\text{leaf}} \). The way the tree is constructed with this parameter is straightforward. Firstly, a tree is grown until all the leaves are pure. Secondly, the tree is pruned until all leaves contain at least \( \lambda_{\text{leaf}} \) data points.

Using a random 80%/20% training/test set split and 10-fold cross-validation, we compute the MSE for the \( \lambda_{\text{leaf}} \) ranging from 1 to 800. The results are shown in figure 4.

We see that the MSE is minimal when the \( \lambda_{\text{leaf}} \) parameter is around 150. We see that the tree starts performing very poorly for \( \lambda_{\text{leaf}} > 180 \). Since we use \( 9/10 \times 80\% \times 1000 = 720 \) data points for growing each tree, it starts performing bad once it must put more than \( 180 = 720/4 \) data points in each leaf node, since then data points around different centers would be included in the same leaf node.

By choosing our value of \( \lambda_{\text{leaf}} = 150 \) we can now fit a tree on the whole training set and check its performance on the test set. Note that we have not used the test set in the cross-validation procedure. We find an MSE of 1.28, which is in line with what we expected from the cross-validation. The corresponding tree is depicted in figure 5. Note how the tree makes smart use of the distance attributes we added. The tree only consists of three splits: asking if the data point is close to \((0, 5), (5, 0)\) and \((0, 0)\). Furthermore, we indeed see that all leaf nodes contain at least 150 data points.
Figure 4: The MSE using 10-fold cross-validation for different values of the $\lambda_{\text{leaf}}$ parameter.
Figure 5: The resulting tree using the $\lambda_{\text{leaf}} = 150$ parameter, where $\text{dist}(a,b)$ stands for our input variables $r_{(a,b)}$. 
2.4 Random Forest

When the set of input variables is large, overfitting on the training set becomes a big problem. The idea of a Random Forest is to overcome this problem by fitting many trees, as uncorrelated as possible. The final prediction of a Random Forest will be the average of all tree predictors (in case of a regression problem) or the majority vote (in case of a classification problem).

To ensure the trees in a forest are not highly correlated, Random Forests use a technique called 'bagging'. Each tree is fitted on a data set that is sampled from the training set, with replacement. This data set is the same size as the training set. Furthermore, for each tree, only a random selection of the input variables is used to determine the splits.

Given a training set $X_{\text{train}}$ and a corresponding target values $y_{\text{train}}$, the algorithm to construct a Random Forest with $B$ trees is as follows.

For $b = 1, \ldots, B$
- Sample with replacement a data set $X_b$ from the training set $X_{\text{train}}$ with corresponding target values $y_b$.
- From this data set, randomly select $p$ input variables.
- Fit a tree $T_b$ using the $p$ selected input variables from $X_b, y_b$.

Common choices for $p$ are $p = \sqrt{N}$ in case of classification, or $p = N/3$ for regression, where $N$ is the number of input variables.

The prediction for input values $X_i$ is in case of a regression forest the average of all the tree predictors

$$RF(X_i) = \frac{1}{B} \sum_{b=1}^{B} T_b(X_i), \quad (11)$$

or in case of a classification forest the majority vote

$$RF(X_i) = \arg\max_{y_k} \# \{ b \mid T_b(X_i) = y_k \}. \quad (12)$$
To illustrate the power of the Random Forest, we consider another example. In this example we use the same model as we introduced in section 2.2. This time, we take 10 data sets consisting of 1000 points each. Our data sets thus consist of four times 250 points generated around the points (0, 0), (0, 5), (5, 0), (5, 5) with target values \( y \) drawn from \( N(2, 1) \) for the points around (0, 0) and (5, 5) and target values \( y \) drawn from \( N(-2, 1) \) for the points around (0, 5) and (5, 0).

We will use the same input variables \( X = (x_1, x_2, r_{(0,0)}, r_{(0,5)}, r_{(5,0)}, r_{(5,5)}) \), but we will also add 1000 random input variables for each data point. These input variables are drawn i.i.d. from a standard normal distribution. By adding these input variables that are independent from our target variable \( y \), we add noise to our input. We expect that a Random Forest will be better at finding the relevant input variables and ignoring the noise, compared to a simpler Regression Tree. In training our Random Forest, we will use \( B = 100 \) trees each using \( p = N/3 = (6 + 1000)/3 \approx 335 \) input variables per tree. For our 10 data sets, we have averaged the performance over 10 random 80%/20% training and test set pairs. We also fitted a regular Regression Tree on each data set. Results are shown in table 3.

<table>
<thead>
<tr>
<th>predictor</th>
<th>mean MSE</th>
<th>standard deviation in MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>2.13</td>
<td>0.16</td>
</tr>
<tr>
<td>Random Forest</td>
<td>1.26</td>
<td>0.09</td>
</tr>
<tr>
<td>Predictor that always predicts ( y = 0 )</td>
<td>5.06</td>
<td>0.14</td>
</tr>
<tr>
<td>Predictor that knows the distribution</td>
<td>1.02</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 3: Average performance of the different predictors on 10 data sets consisting of 1000 data points.

From table 3 we indeed see that the Random Forest performs better than a single Regression Tree in terms of MSE. Compared to not adding the noisy input variables, we see that the performance of the Regression Tree clearly suffers. From table 2 we see that the average MSE increased from 1.32 to 2.13. The Random Forest, however, performs quite well with the noisy input. It’s MSE of 1.26 is only 25% higher than the MSE of the predictor that knows the distribution.
3 The model

The objective of this thesis is to investigate using Random Forests in predicting the stock market and incorporating this in a trading strategy. Before using real market data, we will use data that we generate ourselves. This data is generated to mimic the financial markets, in particular pairs of financial products that are highly correlated. The advantage of using generated data is that we can compare our predictor with a predictor that knows how the data is generated, and thus can compare the performance of our Random Forest to an optimal predictor.

In the first subsection we will introduce our model and compare the Random Forest predictor with the optimal predictor on data generated from our model. In the second subsection we will try to improve our Random Forest predictor by adding more input variables to our Random Forest. In the third subsection we will investigate whether the Random Forest performs better on larger data sets. Lastly, we will look at which input variables are the most important for the prediction of the Random Forest.

3.1 Introduction of the model

We begin with the simplest version of our model. We want our generated data to look like pairs of products that are highly correlated, such as the Dutch AEX index and the German DAX index. Our generated data thus is a two dimensional time series \( Y_t \in \mathbb{R}^2 \). The first product \( Y_{1,t} \) is a pure random walk. The second product \( Y_{2,t} \) is generated in such a way that a certain linear combination of \( Y_{1,t} \) and \( Y_{2,t} \) is a mean reverting process, more specifically an AR(1) process. We thus have

\[
Y_{1,t+1} = Y_{1,t} + \varepsilon_{t+1} \tag{13}
\]

and

\[
Y_{2,t+1} - aY_{1,t+1} = b(Y_{2,t} - aY_{1,t}) + \eta_{t+1}, \tag{14}
\]

where \( a > 0 \), together with \( 0 < b < 1 \) and \( \varepsilon_{t+1}, \eta_{t+1} \) are independent white noise with variances \( \sigma_1^2 \) and \( \sigma_2^2 \) respectively. We can rewrite \( Y_{2,t+1} \) as

\[
Y_{2,t+1} = bY_{2,t} + a(1 - b)Y_{1,t} + a\varepsilon_{t+1} + \eta_{t+1}, \tag{15}
\]

and our process can now be written as

\[
Y_{t+1} = AY_t + V_{t+1} \tag{16}
\]

with

\[
A = \begin{pmatrix}
1 & 0 \\
a(1-b) & b
\end{pmatrix} \tag{17}
\]

and \( V_{t+1} \sim N_2(0, \Lambda) \), where

\[
\Lambda = \begin{pmatrix}
\sigma_1^2 & a\sigma_1^2 \\
a^2\sigma_1^2 & \sigma_2^2 + a^2\sigma_1^2
\end{pmatrix} \tag{18}
\]

It is known that for \(|b| < 1\) the time series \( Y_{2,t} - aY_{1,t} \) is stationary with expectation 0. Using the definition in section 1.1, the time series \( Y_{1,t} \) and \( Y_{2,t} \) are indeed cointegrated. We make three remarks about the dissimilarities between our model and real market data.
• Usually, the one day log returns \( \log(Y_{i,t+1}/Y_{i,t}) \) are modeled as being normally distributed, since this fits the real market data better. We chose not to do this, since we want to keep our model simple. More importantly, it will be easier to define an optimal strategy, as we will see in section 4.

• The multivariate normal distributed renewals \( V_{t+1} \) in our model are constant in time. This is also quite unrealistic. In real market data, we clearly see periods in which there is abnormally high or low volatility.

• In our model, the values of our products \( Y_{i,t} \) can become negative. While this obviously not happens in the real market, it is not a big problem, since we only use daily price differences \( Y_{t+1} - Y_t \) in calculating the daily profits of our strategies.

For \( a = 0.5, \ b = 0.8, \ \sigma_1^2 = \sigma_2^2 = 4 \) and initial values \( Y_{1,0} = 100, \ Y_{2,0} = 50 \) a realization of our model for \( t \in \{0, 1, \ldots, 1000\} \) is plotted in figure 6.

![Figure 6: Plot of our two generated time series \( Y_{1,t} \), \( Y_{2,t} \). In red, the linear combination \( Y_{2,t} - 0.5Y_{1,t} \) that obeys an AR(1) process is shown.](image)

Given all information up to time \( t \), the goal of our Random Forest is to predict the next day prices \( Y_{t+1} \). Since the predictions of a Random Forest will always be an average of values seen in the training set, we let our Random Forest predict the daily price differences \( Y_{t+1} - Y_t \). If we would not do this, the Random Forest would never predict values higher or lower than the maximum or minimum value of \( Y_t \) in the training set, this is clearly not desirable. Our training set thus consist of target values \( y_t = Y_{t+1} - Y_t \). For our input variables \( X_t \) we use

• The prices from today and the last 9 days, i.e.

\[
Y_{j,t-h}
\]  

\[ \text{(19)} \]
for $j = 1, 2$ and $h = 0, 1, \ldots, 9$.

- The difference between today and the last 9 days, i.e.
  \[ Y_{j,t} - Y_{j,t-h} \]  
  for $j = 1, 2$ and $h = 1, \ldots, 9$.

- The difference between the two different prices for today and the last 9 days, i.e.
  \[ Y_{1,t-h} - Y_{2,t-h} \]  
  for $h = 0, 1, \ldots, 9$.

At each time $t$ our input variables $X_t$ is thus 48 dimensional.

For fitting the Random Forest, we use 1000 trees and a $\lambda_{leaf}$ parameter set to 10. Furthermore, we use all of the input variables for each tree. We do this because the number of input variables is fairly small. If we include less than 48 input variables, we might lose input variables that have the most predictive power. For the training data we use the first 800 data points, for the test set we use the remaining 200 future data points. To compare the performance of our Random Forest we introduce two other predictors. The first predictor $P_{\text{previous}}$ always predicts the values of today, i.e. $P_{\text{previous}}(X_t) = Y_t$. The second predictor $P_{\text{optimal}}$ knows the distribution and predicts the expected value of $Y_{t+1}$ according to our model: $P_{\text{optimal}}(X_t) = AY_t$, see (16).

The resulting MSE’s on the test set are shown in table 4.

<table>
<thead>
<tr>
<th>Predictor</th>
<th>MSE $Y_1$</th>
<th>MSE $Y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>3.93</td>
<td>4.91</td>
</tr>
<tr>
<td>$P_{\text{optimal}}$</td>
<td>3.95</td>
<td>4.76</td>
</tr>
<tr>
<td>$P_{\text{previous}}$</td>
<td>3.95</td>
<td>5.22</td>
</tr>
</tbody>
</table>

Table 4: Performance of the different predictors on the test set.

We first note that all three predictors perform equally well in predicting $Y_{1,t+1}$. This is quite natural, since $Y_1$ is a random walk and thus cannot be predicted. The values for the MSE lie around the expected value $\mathbb{E}[(Y_{1,t+1} - Y_{1,t})^2] = \sigma_1^2 = 4$.

Regarding predicting $Y_2$, we see that our Random Forest outperforms the $P_{\text{previous}}$ predictor in terms of MSE. To determine how well the Random Forest predictor performs in our model, we do a simulation study. We simulate 250 data sets from our model. For each data set we fit a Random Forest and test its performance in terms of MSE on the test set (the last 20% of the data). In table 5 its performance is shown, as well as that of our predictors $P_{\text{previous}}$ and $P_{\text{optimal}}$.

The results do not look very promising at first sight: it seems that our Random Forest can not outperform the predictor that predicts yesterdays prices. However, one may argue that MSE is not a good measure to consider in this case. In our model, the daily price change $\Delta Y_2$ in $Y_{2,t}$ at time $t$ has a normal distribution

\[ \Delta Y_2 = Y_{2,t+1} - Y_{2,t} \sim N(\mu, \sigma_2^2), \]
<table>
<thead>
<tr>
<th>Predictor</th>
<th>MSE $Y_1$</th>
<th>MSE $Y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>4.30</td>
<td>5.42</td>
</tr>
<tr>
<td>$P_{\text{optimal}}$</td>
<td>4.00</td>
<td>4.97</td>
</tr>
<tr>
<td>$P_{\text{previous}}$</td>
<td>4.00</td>
<td>5.41</td>
</tr>
</tbody>
</table>

Table 5: Average performance of the different predictors on the test set, for 250 simulated data sets of size $N = 1000$.

where $\mu \neq 0$ almost always. The expected value of the MSE of our $P_{\text{previous}}$ predictor thus equals $\mu^2 + \sigma^2$. Consider a predictor $P_{\mu}$ that predicts $-\mu$ 40% of the time, and $\mu$ 60% of the time. The expected MSE of this predictor would be

$$E[\text{MSE}(P_{\mu})] = 0.6E[(\mu - \Delta Y_2)^2] + 0.4E[(-\mu - \Delta Y_2)^2]$$

$$= 0.6\sigma_2^2 + 0.4((2\mu)^2 + \sigma_2^2)$$

$$= 1.6\mu^2 + \sigma_2^2.$$

This is higher than the expected MSE of our $P_{\text{previous}}$ predictor. Consider the following strategy based on $P_{\mu}$: buy if $P_{\mu} = \mu$, sell if $P_{\mu} = -\mu$. We assumed here that $\mu > 0$. For negative $\mu$, we reverse our strategy. The expected profit of this strategy would be

$$0.2|\mu|.$$

We have thus shown our predictor $P_{\mu}$ could be used in defining a profitable trading strategy, while it has a worse expected MSE than predicting no price change at all.

This gives rise to a different performance measure of our estimators. For a predictor $P$, input variables $X_i \in X$ and corresponding target values $y_i$, we define

$$\text{accuracy}(P, X) = \frac{\# \{X_i \mid \text{sgn}(\Delta P(X_i)) = \text{sgn}(\Delta y_i)\}}{\# X},$$

(22)

where $\Delta P(X_i) = P(X_i) - Y_i$, $\Delta y_i = Y_{i+1} - Y_i$ and sgn($x$) the usual sign function. The accuracy thus measures how often the predictor predicts correctly if a price goes up or down, analogous to a classification problem.

In table 6 the performance of our predictors are shown once again, but now also according to our new accuracy measure (22). We see that both our optimal predictor $P_{\text{optimal}}$ and the Random Forest predictor $P_{\text{RF}}$ have an accuracy of about 50% in predicting the daily price change in $Y_1$. This is to be expected, since $Y_{1,t}$ is a random walk. The accuracy for predicting the daily price change in $Y_{2,t}$ is considerably better for both predictors. Note that the previous predictor $P_{\text{previous}}$ always predict a price change of 0, so we can’t calculate its accuracy.
<table>
<thead>
<tr>
<th>Predictor</th>
<th>MSE Y₁</th>
<th>MSE Y₂</th>
<th>Accuracy Y₁</th>
<th>Accuracy Y₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>4.30</td>
<td>5.42</td>
<td>0.5044</td>
<td>0.5546</td>
</tr>
<tr>
<td>$P_{optimal}$</td>
<td>4.00</td>
<td>4.97</td>
<td>0.5015</td>
<td>0.5916</td>
</tr>
<tr>
<td>$P_{previous}$</td>
<td>4.00</td>
<td>5.41</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6: Average performance of the different predictors on the test set, for 250 simulated data sets of size $N = 1000$. 
3.2 Input variable selection

In our previous analysis we only used 48 input variables. This is quite a low number, since Random Forests are designed to detect structures in much higher dimensional input data. The goal of this subsection is to look for relevant input variables, and see if the Random Forest performs better if we include them.

Consider figure 6 once more. By the way we generated our data, the linear combination \( Z_t := Y_{2,t} - 0.5Y_{1,t} \) is a mean reverting process. A profitable trading strategy would look like a classic mean reversion strategy: buy \( Z_t \) if \( Z_t < 0 \) and sell it if \( Z_t > 0 \). Buying \( Z_t \) in this context means buying a certain amount of \( Y_2 \), while short selling half of that amount \( Y_1 \). Since \( Z_t \) is a AR(1) process, we know for sure that \( Z_t \) will be positive at some time \( t \) in the future and we would thus always make a profit.

Looking at linear combinations could also help in predicting the daily price changes. Using equation (16) we can write the distribution of our daily price changes \( Y_{t+1} - Y_t \) as

\[
Y_{t+1} - Y_t = (A - I)Y_t + V_{t+1}
\]

\[
= \begin{pmatrix} 0 & 0 \\ a(1-b) & b-1 \end{pmatrix} Y_t + V_{t+1}.
\]  

(23)

Since the expected daily price change is a linear combination of today’s prices, it might be useful to include it in the input variables. Since it would be cheating to include only the linear combination from equation (23), we include many random linear combinations. These random projections \( Z_i^t \) are generated in the following way

\[
Z_i^t = r_{i1}^t Y_{1,t} + r_{i2}^t Y_{2,t},
\]  

(24)

where \( r_{i1}^t, r_{i2}^t \) i.i.d. standard normal.

At each time \( t \), we include the following values in our input data

\[
Z_{t-h}
\]  

(25)

and

\[
Z_{t-h} - \frac{1}{50} \sum_{i=t-50}^{t} Z_{i-h}
\]  

(26)

for \( h = 0, \ldots, 5 \). We thus add 10 input variables for each random projection. The second value we input is based on the following idea. Suppose \( Z_t \) is a mean reverting process around some changing mean \( \mu(t) \). We would want to determine if \( Z_t \) is above or below this value, since we would maybe be interested in selling/buying \( Z_t \). The second term in equation (26) tries to estimate this \( \mu(t) \). If equation (26) if negative, it might be a good opportunity to buy. If it’s positive, it may be wise to sell.

In table 7 we updated our performance table, now including a Random Forests fitted on the same data sets and using 100 random projections. Note that where we previously used all 48 input variables in fitting our Random Forest, we now
take the square root of the number of input variables in fitting our Random Forest with the random projections. We do this to save time in constructing our Random Forest and because of the correlation between the input variables we are less likely to lose much predictive power. Moreover, the idea behind the Random Forest is to include not all input variables in each tree. Since we add 10 input variables for each random projection, we use $\sqrt{48 + 10 \cdot 100} \approx 32$ input variables in each tree.

$$\begin{array}{|c|c|c|c|c|}
\hline
\text{Predictor} & \text{MSE } Y_1 & \text{MSE } Y_2 & \text{Accuracy } Y_1 & \text{Accuracy } Y_2 \\
\hline
\text{Random Forest} & 4.30 & 5.42 & 0.5044 & 0.5546 \\
\text{Random Forest with } Z_1^t & 4.33 & 5.44 & 0.5058 & 0.5692 \\
\text{P}_{\text{optimal}} & 4.00 & 4.97 & - & 0.5916 \\
\text{P}_{\text{previous}} & 4.00 & 5.41 & - & - \\
\hline
\end{array}$$

Table 7: Average performance of the different predictors on the test set, for 250 simulated data sets of size $N = 1000$. We used all input variables in fitting the Random Forest, while we use the square root of the number of input variables in fitting the Random Forest with random projections $Z_1^t$.

We see that adding the random projections to the input variables does not reduce the MSE of either $Y_1$ or $Y_2$. We see, however, that the accuracy in predicting $Y_2$ is 1.46% higher than the Random Forest that only made use of our initial 48 input variables. This is quite an improvement, concerning the gap in accuracy of predicting $Y_2$ between the latter predictor and the optimal predictor is only 3.70%.

### 3.3 More data points

We expect our Random Forest to perform better if we input more data points. To test this hypothesis we perform another simulation study, this time generating 25 data sets of size $N = 10000$. We fit our Random Forest on the first 80% of the data, and check its performance on the remaining 20%. The results are shown in table 8.

$$\begin{array}{|c|c|c|c|c|}
\hline
\text{Predictor} & \text{MSE } Y_1 & \text{MSE } Y_2 & \text{Accuracy } Y_1 & \text{Accuracy } Y_2 \\
\hline
\text{Random Forest} & 4.16 & 5.35 & 0.5005 & 0.5583 \\
\text{Random Forest with } Z_1^t & 4.21 & 5.22 & 0.4956 & 0.5816 \\
\text{P}_{\text{optimal}} & 4.00 & 5.01 & - & 0.5923 \\
\text{P}_{\text{previous}} & 4.00 & 5.45 & - & - \\
\hline
\end{array}$$

Table 8: Average performance of the different predictors on the test set, for 25 simulated data sets of size $N = 10000$. We used all input variables in fitting the Random Forest, while we use the square root of the number of input variables in fitting the Random Forest with random projections $Z_1^t$.

We conclude that the Random Forest that uses the random projections performs much better, considering both MSE and accuracy, than the Random Forest that uses our initial 48 input variables. We also note that the difference in accuracy in
predicting \( Y_2 \) only is about 1% between our optimal predictor and the Random Forest with the random projections.

### 3.4 Variable importance

For determining the importance of an input variable \( x_i \), the ‘mean decrease impurity’ measure is often used. This measure is calculated as follows: given a tree, we determine all splits involving \( x_i \). For each split, we calculate the total decrease in node impurity (in case of using MSE as the impurity measure this is precisely (10)). This total decrease in node purity is then multiplied by the probability of arriving at the particular node, which is estimated by the number of data points that reach the node divided by the total number of data points. The sum of these products is calculated for each tree. Finally, the ‘mean decrease impurity’ is the mean of these values over all trees in the Random Forest. Note that this measure depends on the data set. So one can generally not compare this measure between several data sets.

As an example, we pick the last data set in our last simulation study. The five most important input variables are shown in table 9, where we used the notation for \( Z_t \) as in equations (24) and (25).

<table>
<thead>
<tr>
<th>Rank</th>
<th>Input variable</th>
<th>Importance score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Z_t ) with ( r_1 = -0.821, r_2 = 1.682, h = 0 )</td>
<td>0.034</td>
</tr>
<tr>
<td>2</td>
<td>( Z_t ) with ( r_1 = -0.821, r_2 = 1.682 ), minus rolling mean, ( h = 0 )</td>
<td>0.023</td>
</tr>
<tr>
<td>3</td>
<td>( Z_t ) with ( r_1 = 0.500, r_2 = -1.057, h = 0 )</td>
<td>0.016</td>
</tr>
<tr>
<td>4</td>
<td>( Y_{1,t} - Y_{1,t-1} )</td>
<td>0.016</td>
</tr>
<tr>
<td>5</td>
<td>( Y_{2,t} - Y_{2,t-7} )</td>
<td>0.015</td>
</tr>
</tbody>
</table>

Table 9: Most important input variables for a Random Forest fitted on a particular data set of size \( N = 10000 \) in our simulation study.

Using equation (23) and recalling that we used \( a = 0.5 \) and \( b = 0.8 \) we have at time \( t \)

\[
E(Y_{2,t+1} - Y_{2,t}) = 0.1Y_{1,t} - 0.2Y_{2,t}. \tag{27}
\]

The three most important input variables in table 9 all are random projections with \( r_1/r_2 \approx -1/2 = 0.1/(-0.2) \). This indicates that our Random Forest recognizes that those random projections have predictive power. It is strange that the fourth most important input variable is \( Y_{1,t} - Y_{1,t-1} \), since it does not have any prediction power in our model. The fifth most important input variable \( Y_{2,t} - Y_{2,t-7} \) also does not seem relevant.

The reason for this is most likely overfitting: on the training set the Random Forest saw relations involving \( Y_{1,t} - Y_{1,t-1} \) and \( Y_{2,t} - Y_{2,t-7} \) and made splits accordingly. We however know that both these input variables are not relevant.

Note that these importances are from fitting one Random Forest on a particular data set. Fitting another forest using the same parameters might lead to different most important input variables due to the random nature of our Random Forest.
4 Strategy

In this section we will define our trading strategy. This means that for each time \( t \), we need to determine the optimal weights for each product \( Y_{i,t} \) in our portfolio. In the first subsection we make certain assumptions about our predictor for the next day prices \( Y_{t+1} \). Furthermore we impose certain restrictions on magnitude of our daily losses. These lead to restrictions on our weights. In the next subsection we determine the optimal weights by optimizing the expected profit under these restrictions. In the third subsection we investigate a univariate strategy and illustrate the concept of hedging. After that we will test our strategy on data from our model. We find that we need to make an adjustment to our strategy, this is done in the following subsection. After that, we discuss predicting the covariance in our model by another Random Forest and test this once again on data from our model. In the next subsection we investigate the profits of our strategy on a larger data set. Following this, we will look closer into parameter selection in our strategy. After that, we will introduce a new model, based on the ARCH model. Then, we will discuss the importance of our input variables in the Random Forest. In the penultimate subsection, we will add more input variables trying to better the performance of our strategy. Finally, we will summarize our findings and look ahead at applying our strategy on real market data.

4.1 General Framework

Suppose we are looking at a \( k \)-dimensional time series of prices \( Y_t \in \mathbb{R}^k \). Suppose we are at time \( t \) and we want to determine the weights \( w \) that maximize our expected profit. Our weights vector \( w = (w_1, \ldots, w_k) \) will thus be a \( k \)-dimensional vector, where \( w_i \) represents the amount of product \( Y_{i,t} \) to buy. If \( w_i < 0 \), we short sell \( Y_{i,t} \). We will assume that we can buy any amount of \( Y_{i,t} \), so \( w_i \) does not need to be an integer. While maximizing our expected profit, we also want to limit our risk. We therefore impose that the probability of losing more than \( L \) is at most \( \alpha \).

We thus want to maximize

\[
E(w^T(Y_{t+1} - Y_t)|F_t),
\]

under the constraint

\[
P(w^T(Y_{t+1} - Y_t) < -L|F_t) \leq \alpha,
\]

where \( F_t \) denotes all information up to time \( t \). Suppose we have a predictor \( P(X_t) \) of \( Y_{t+1} \) and let the difference between the predictor and today’s value be denoted by \( \Delta = P(X_t) - Y_t \), where \( X_t \) denotes all input variables for our predictor. We will assume that at time \( t \) the difference between the target value and our predictor is normally distributed with mean 0 and covariance matrix \( \Sigma \), i.e.

\[
Y_{t+1} - P(X_t)|F_t \sim N_k(0, \Sigma).
\]

Under this assumption we have

\[
Y_{t+1} - Y_t|F_t \sim N_k(\Delta, \Sigma).
\]
The probability in equation (29) is now equal to
\[ P \left( N(0, 1) < \frac{-L - w^T \Delta}{\sqrt{w^T \Sigma w}} \right). \tag{32} \]

In order to bound this probability by \( \alpha \), we need
\[ -L - w^T \Delta \leq z_\alpha, \tag{33} \]
where \( z_\alpha = \Phi^{-1}(\alpha) \) the inverse cumulative distribution function for a standard normal random variable. Since generally \( z_\alpha < 0 \), this gives
\[ L^2 + (w^T \Delta)^2 + 2Lw^T \Delta \geq z_\alpha^2 w^T \Sigma w. \]

We rewrite this to
\[ w^T (z_\alpha^2 \Sigma - \Delta \Delta^T) w - 2Lw^T \Delta - L^2 \leq 0. \]

This inequality can be written as
\[ (w^T - c^T) A (w - c) \leq K, \tag{34} \]
where
\[ A = z_\alpha^2 \Sigma - \Delta \Delta^T \]
\[ c = LA^{-1} \Delta \]
\[ K = L^2 (1 + \Delta A^{-1} \Delta). \]

Note that we assumed that \( A \) is invertible. The inequality in (34) describes either a ellipsoid, a hyperboloid or a paraboloid, depending on the eigenvalues of \( A \).

Since we will often encounter the value \( \Delta^T A^{-1} \Delta \), we will calculate it. Let \( A^{-1} \Delta = \nu \), we can then write
\[ A \nu = \Delta = z_\alpha^2 \Sigma \nu - \Delta \Delta^T \nu. \]

Writing \( \mu = \Delta^T A^{-1} \Delta \) we can rewrite this to
\[ z_\alpha^2 \Sigma \nu = \Delta (1 + \mu), \]
and thus
\[ \nu = \frac{1}{z_\alpha^2} \Sigma^{-1} \Delta (1 + \mu). \]

Since \( \Delta^T \nu = \mu \) we have
\[ \mu = \frac{1}{z_\alpha^2} \Delta^T \Sigma^{-1} \Delta (1 + \mu). \]

Solving this for \( \mu \) leads to
\[ \Delta^T A^{-1} \Delta = \mu = \frac{\Delta^T \Sigma^{-1} \Delta}{z_\alpha^2 - \Delta^T \Sigma^{-1} \Delta}. \tag{35} \]
Our quadratic form thus reads
\[(w^T - c^T)A(w - c) \leq K,\] (36)
with parameters
\[A = z_a^2 \Sigma - \Delta \Delta^T\] (37)
\[c = L \left( \frac{1}{z_a^2 - \Delta \Delta^T \Sigma^{-1} \Delta} \right) \Sigma^{-1} \Delta\] (38)
\[K = L^2 \left( \frac{z_a^2}{z_a^2 - \Delta \Delta^T \Sigma^{-1} \Delta} \right).\] (39)

We want the inequality in equation (36) to describe the interior of an ellipsoid, since a hyperboloid or a paraboloid doesn’t have a bounded interior and the inequality would allow for infinite values of \(w\). The constraint (36) describes an ellipsoid if and only if \(A\) is positive definite. In order to obtain a condition for this, we make some observations.

Note that \(A\) is symmetric, since \(\Sigma\) and \(\Delta \Delta^T\) are as well. We use the following observation: let \(B\) be a \(k \times k\) matrix that is symmetric and of full rank, then \(A\) is positive definite if and only if \(BAB\) is positive definite. This can be seen as follows. Let \(A\) be positive definite and let \(v \in \mathbb{R}^k\) non-zero, we have \(v^T BABv = (Bv)^T A (Bv) > 0\). So \(BAB\) is positive definite.

If we assume \(BAB\) is positive definite, we have \(v^T A v = v^T B^{-1} BAB B^{-1} v = (B^{-1} v)^T BAB (B^{-1} v) > 0\). This concludes our observation.

Since the covariance matrix \(\Sigma\) is positive definite, there exists a unique square root \(\Sigma^{1/2}\) such that \(\Sigma = \Sigma^{1/2} \Sigma^{1/2}\). Note that \(\Sigma^{1/2}\) is also positive definite. Let \(\Sigma^{-1/2}\) denote the inverse of \(\Sigma^{1/2}\). We can now write
\[\Sigma^{-1/2} A \Sigma^{-1/2} = z_a^2 I_k - \Sigma^{-1/2} \Delta \Delta^T \Sigma^{-1/2}\]
\[= z_a^2 I_k - (\Sigma^{-1/2} \Delta)(\Sigma^{-1/2} \Delta)^T,\] (40)
where \(I_k\) is the identity matrix. We can easily determine the eigenvectors. We see that \((\Sigma^{-1/2} \Delta)(\Sigma^{-1/2} \Delta)^T\) has rank 1, with eigenvector \(\Sigma^{-1/2} \Delta\). We can calculate the corresponding eigenvalue by writing
\[(\Sigma^{-1/2} A \Sigma^{-1/2}) \Sigma^{-1/2} \Delta = z_a^2 \Sigma^{-1/2} \Delta - \Sigma^{-1/2} \Delta \| \Sigma^{-1/2} \Delta \|^2 \]
\[= (z_a^2 - \| \Sigma^{-1/2} \Delta \|^2) \Sigma^{-1/2} \Delta \]
\[= (z_a^2 - \Delta \Delta^T \Sigma^{-1} \Delta) \Sigma^{-1/2} \Delta,\]
and concluding that \(z_a^2 - \Delta \Delta^T \Sigma^{-1} \Delta\) is an eigenvalue of \(\Sigma^{-1/2} A \Sigma^{-1/2}\). From equation (40) we can easily see that the other eigenvectors are the \(n - 1\) vectors perpendicular to \(\Sigma^{-1/2} \Delta\). All these eigenvectors have an eigenvalue equal to \(z_a^2\).

Since having only positive eigenvalues is equivalent to positive definiteness, we can conclude that \(\Sigma^{-1/2} A \Sigma^{-1/2}\) is positive definite if and only if
\[\Delta \Delta^T \Sigma^{-1} \Delta < z_a^2.\] (41)
By our earlier remark we can conclude that \(A\) also always has at least \(n - 1\) positive eigenvalues, and that it is positive definite if the same inequality holds.
Figure 7: Boundaries of ellipses (36) using $L = 2$ and $\alpha = 10\%$ for several values of the predicted price change $\Delta = (0, \Delta_2)$: $\Delta_2 = 0.5$ (blue), $\Delta_2 = 1$ (red), $\Delta_2 = 2$ (yellow) and $\Delta_2 = -1.5$ (black). The centers (38) of the ellipses are shown by a dot.

For $\Sigma = \Lambda$ from (18) using $a = 0.5$, $b = 0.8$, $\sigma_1^2 = \sigma_2^2 = 4$ together with $L = 2$ and $\alpha = 10\%$, the boundaries of the ellipse (36) constraining our weights $w$ is shown for several predicted price changes $\Delta = (0, \Delta_2)$ in figure 7. Note that our optimal predictor will always predict $\Delta_1 = 0$, since $Y_{1,t}$ is a random walk. We see indeed that for large values of the predicted price change $\Delta_2$ in $Y_2$ the ellipses become bigger. Moreover, we see that the center of all the ellipses lie on the same line. Looking back at (38) we indeed see that all centers are a multiple of $\Sigma^{-1}\Delta$. 

35
4.2 Determining the optimal weights

We have rewritten our constraint (29) defined by a maximum probability $\alpha$ of losing at most $-L$ to a quadratic form (36). Under condition (41) this quadratic form defines an ellipsoid. We want to maximize the daily profit $w^T \Delta$ under the constraint that our weight do not lie outside the ellipsoid. For this kind of optimization problems, the maximum is always attained at the boundary of the ellipsoid. Our optimization problem thus reads

$$\text{maximize} \quad w^T \Delta$$

such that

$$(w^T - c^T)A(w - c) = K.$$ (43)

We calculate the Lagrangian

$$\mathcal{L}(w, \lambda) = w^T \Delta + \lambda((w^T - c^T)A(w - c) - K),$$ (44)

and want to set the derivatives to zero. We see

$$\nabla_w \mathcal{L}(w, \lambda) = \Delta + 2 \lambda A(w - c) = 0$$

$$\nabla_{\lambda} \mathcal{L}(w, \lambda) = (w^T - c^T)A(w - c) - K = 0.$$ (45)

The two solutions to this problem read

$$\hat{w} = c + A^{-1} \frac{1}{2\lambda}(-\Delta)$$

$$\hat{\lambda} = \pm \frac{1}{2} \sqrt{\frac{\Delta^T A^{-1} \Delta}{K}},$$

leading to

$$\hat{w} = L \left( 1 \pm \sqrt{\frac{1 + \Delta^T A^{-1} \Delta}{\Delta^T A^{-1} \Delta}} \right) A^{-1} \Delta.$$ (46)

Note that under condition (41) we have that $\Delta^T A^{-1} \Delta > 0$. So the value of the square root is greater than 1. Since also $L > 0$, we conclude that we have to take the plus sign in the above equation, since otherwise our expected profit $w^T \Delta$ would be negative. We thus conclude that the equation for our weights $\hat{w}$ that optimize our expected profit whilst satisfying the constraint implied by our risk managing condition (29) reads

$$\hat{w} = L \left( 1 + \sqrt{\frac{z^2_\alpha}{\Delta^T \Sigma^{-1} \Delta}} \right) \left( \frac{1}{z^2_\alpha - \Delta^T \Sigma^{-1} \Delta} \right) \Sigma^{-1} \Delta,$$ (46)

where we used expression (35) to rewrite (45) to (46).
4.3 Hedging

Before we are going to test our strategy on data generated from our model, we first take a little sidestep. We will discuss whether it is important to both include \(Y_{1,t}\) and \(Y_{2,t}\) in the portfolio for the optimal strategy. Since in our model \(Y_{1,t}\) is just a random walk, we cannot find a winning (or losing) strategy that only consist of buying and/or selling \(Y_{1,t}\). We wonder if a strategy where we only consider \(Y_{2,t}\) will be as profitable as considering both \(Y_{1,t}\) and \(Y_{2,t}\). This turns out not to be the case. This illustrate the concept of hedging. Because \(Y_{1,t}\) and \(Y_{2,t}\) are correlated, short selling \(Y_{1,t}\) will allow us to have a bigger long position in \(Y_{2,t}\) and vice versa. Since \(Y_{1,t+1}\) has expectation \(Y_{1,t}\) our expected profit will not be affected by including \(Y_{1,t}\) in our portfolio. This however allows to buy/sell more of \(Y_{2,t}\), leading to higher profits.

Let \(\hat{w} = (\hat{w}_1, \hat{w}_2)\) be the optimal solution where we consider both stocks, see equation (45). Let \(w' = (0, w'_2)\) be the solution of the optimal strategy where we only consider stock \(Y_2\) and let \(\Delta = (\Delta_1, \Delta_2)\) be the predicted price change for the next day. From equation (45) we conclude

\[
w'_2 = L \left(1 + \frac{1 + \Delta_2^2 A_{22}^{-1}}{\Delta_2^2 A_{22}^{-1}}\right) A_{22}^{-1} \Delta_2,
\]

where \((A_{22})^{-1} = (z_a^2 \Sigma_{22} - \Delta_2^2)^{-1}\), the inverse of the element in the right corner of \(A\). This can also be obtained by solving equation (34), which is now a standard quadratic expression.

Let the real day-to-day differences be denoted by \(\Delta Y_{t+1} = (\Delta Y_{1,t+1}, \Delta Y_{2,t+1}) = Y_{t+1} - Y_t\). Note that under assumption (30), we have

\[
\Delta Y_{t+1} | F_t \sim \mathcal{N}(\Delta, \Lambda)
\]

and thus \(\mathbb{E}(\Delta Y_{t+1} | F_t) = \Delta\). We are interested in the ratio of the expectations of the two profits \(\hat{w}^T \Delta Y_{t+1}\) and \(w'_2 \Delta Y_{2,t}\). The expected theoretic profits can be written as

\[
\mathbb{E}(\hat{w}^T \Delta Y_{t+1} | F_t) = \hat{w}^T \Delta = LQ(1 + \sqrt{1 + 1/Q})
\]

\[
\mathbb{E}(w'_2 \Delta Y_{2,t+1} | F_t) = w'_2 \Delta_2 = LQ_2(1 + \sqrt{1 + 1/Q_2}),
\]

where \(Q = \Delta^T A^{-1} \Delta\) and \(Q_2 = \Delta_2^2 (A_{22})^{-1}\). After some calculation we find that

\[
\frac{Q}{Q_2} = \frac{\Delta \Sigma^{-1} \Delta (z_a^2 \Sigma_{22} - \Delta_2^2)}{\Delta_2^2 (z_a^2 - \Delta \Sigma^{-1} \Delta)}.
\]

We also find

\[
\sqrt{1 + 1/Q} = \frac{|z_a|}{\sqrt{\Delta^T \Sigma^{-1} \Delta}}.
\]

\[
\sqrt{1 + 1/Q_2} = \frac{|z_a| \sqrt{\Sigma_{22}}}{|\Delta_2|}.
\]
Which gives us for the ratio between the two expected profits

\[
\frac{\hat{w}^T \Delta}{w'_2 \Delta_2} = \frac{\Delta^T \Sigma^{-1} \Delta (z^2_0 \Sigma_{22} - \Delta_2^2)}{\Delta_2^2 (z^2_0 - \Delta^T \Sigma^{-1} \Delta)} \cdot \frac{1 + \frac{|z_0|}{\sqrt{\Delta^T \Sigma^{-1} \Delta}}}{1 + \frac{|z_0|}{|\alpha|}}.
\]

In our model, the theoretic predictor is of the form \( \Delta = (0, \Delta_2) \). We are interested in how the theoretic profit ratio behaves as a function of \( \Delta_2 \). Using \( \Delta^T \Sigma^{-1} \Delta = \Delta_2^2 \Sigma_{11} / \det \Sigma \), we find

\[
\frac{\hat{w}^T \Delta}{w'_2 \Delta_2} = \frac{(z^2_0 \Sigma_{22} - \Delta_2^2) \Sigma_{11}}{z^2_0 \det \Sigma - \Delta_2^2 \Sigma_{11}} \cdot \frac{|\Delta_2| + |z_0| \sqrt{\det \Sigma / \Sigma_{11}}}{|\Delta_2| + |z_0| \sqrt{\Sigma_{22}}}. \tag{47}
\]

In figure 8, the ratio (47) is depicted as a function of the predicted price change \( \Delta_2 \) in \( Y_{2,t} \). We see that for all values of \( \Delta_2 \) this ratio is bigger than 1. Even if \( \Delta_2 \) approaches 0, the ratio is still strictly bigger than 1. This thus shows that including \( Y_{1,t} \) in our portfolio does yield a higher expected profit. We can calculate this limit as \( \Delta_2 \) approaches zero by writing

\[
\lim_{\Delta_2 \to 0} \frac{\hat{w}^T \Delta}{w'_2 \Delta_2} = \frac{\Sigma_{11} \Sigma_{22}}{\det \Sigma} = \sqrt{\frac{\Sigma_{11} \Sigma_{22}}{\Sigma_{11} \Sigma_{22} - \Sigma_{12}^2}} > 1, \tag{48}
\]

and see that it is indeed strictly bigger than 1.

Figure 8: Ratio of the expected profits of using a strategy where we can use both stocks \( Y_{1,t}, Y_{2,t} \) versus a strategy where we only use the predictable (i.e. not random walk) stock \( Y_{2,t} \) as a function of the predicted price change \( \Delta_2 \) in \( Y_{2,t} \), see equation (47).
4.4 First test of our strategy

Now we have defined our strategy, we are going to test it on data generated from our model. Firstly, we are testing the strategy based on the optimal predictor \( P_{\text{optimal}} \). In our model, the difference between the target value and the predictor \( P_{\text{optimal}} \) is indeed normally distributed, see (16). Our assumption (30) is thus indeed true, and we have

\[
Y_{t+1} - P_{\text{optimal}}(X_t) | F_t \sim N(0, \Sigma),
\]

where \( \Sigma = \Lambda \), see (18). By using the optimal predictor \( P_{\text{optimal}} \) and corresponding matrix \( \Lambda \), we know that the expected profit of this strategy is higher than any other strategy that also satisfies the constraint. If we use our Random Forest predictor in our strategy later on, we can compare its profits to the optimal strategy, i.e. the strategy that knows the distribution of our generated data.

At each time \( t \) we use the \( P_{\text{optimal}} \) predictor to find the optimal weights \( \hat{w} \). Our daily profit will thus be \( \hat{w}^T (Y_{t+1} - Y_t) \).

For two different data sets generated from our model using our usual parameters, the cumulative profits from our optimal strategy are shown in figure 9. The data sets consist of a 1000 data points, of which we used the first 80% to fit the Random Forest predictor. On the last 20% we applied our strategy. We used the condition that we would lose more than \( L = -2 \) less than \( \alpha = 10\% \) of the time.

![Figure 9: Cumulative daily profits for our optimal strategy on two different data sets.](image)

The profits on the left in figure 9 look very good, it looks like a straight line with little fluctuations. This can also be said of the profits on the right of figure 9, except for the peak just before \( t = 900 \). During the peak our strategy led to very large weights \( \hat{w} = (\hat{w}_1, \hat{w}_2) \). This can be explained by the definition of \( \hat{w} \) in (46). During the peak our predictor predicted a large price change \( \Delta \). Because of this the term \( \Delta^T \Sigma^{-1} \Delta \) became very close to \( z^2_\alpha \). The term \( (z^2_\alpha - \Delta^T \Sigma^{-1} \Delta)^{-1} \) in equation (46) thus became very large and so did our optimal weights \( \hat{w}_1, \hat{w}_2 \).

Note that if \( \Delta^T \Sigma^{-1} \Delta > z^2_\alpha \), the restriction on \( \hat{w}_1 \) and \( \hat{w}_2 \) in \((\hat{w}_1, \hat{w}_2)\)-space is not an ellipse anymore, but a hyperbola. This means that the optimal weights can actually take infinite values. But in both scenario’s, the probability of losing
more than $L$ still is less than $\alpha$. The constraint is thus satisfied, but both scenario’s are clearly not desirable in a real trading strategy. A possible solution is presented in the next subsection.

4.5 Introducing more constraints

We propose a way to ensure finite weights (i.e. the space of allowed weights $w$ is not an ellipsoid), while also having more control over the distribution of the losses.

The idea is to expand the constraint in equation (29) to various values of $L$ and $\alpha$. In fact, we will let $\alpha$ be a function of $L$, so we impose a constraint for all level of losses $L$. We will see that this is possible for functions $\alpha(L)$ that decline less fast than $\exp(-L^2)$.

Let $\alpha(L)$ be a function of $L$. We now impose

$$P(w^T(Y_{t+1} - Y_t) < -L|F_t) \leq \alpha(L)$$

(50)

for all $L > 0$. As calculated before, we may write the optimal weight $\hat{w}$ for certain $L$, $\alpha$ and $\Delta, \Sigma$ as in equation (46)

$$\hat{w}(\alpha, L; \Delta, \Sigma) = L \left( 1 + \sqrt{\frac{z^2_{\alpha(L)}}{\Delta^T \Sigma^{-1} \Delta}} \right) \left( \frac{1}{z^2_{\alpha(L)} - \Delta^T \Sigma^{-1} \Delta} \right) \Sigma^{-1} \Delta.$$  (51)

Since all optimal weights $\hat{w}$ stand in the same direction $\Sigma^{-1} \Delta$, we can write

$$\hat{w}(\alpha, L; \Delta, \Sigma) = c(\alpha, L; \Delta, \Sigma) \cdot \Sigma^{-1} \Delta,$$  (52)

with

$$c(\alpha, L; \Delta, \Sigma) := L \left( 1 + \sqrt{\frac{z^2_{\alpha(L)}}{\Delta^T \Sigma^{-1} \Delta}} \right) \left( \frac{1}{z^2_{\alpha(L)} - \Delta^T \Sigma^{-1} \Delta} \right).$$  (53)

Because $\alpha(L)$ is a function of $L$, we only have to minimize $c(\alpha, L; \Delta, \Sigma)$ to ensure all constraints in (50) are satisfied. Denoting $\tilde{w}$ as the optimal weight under our newly defined constraints, we thus have

$$\tilde{w} = \min_{L>L_{\min}} c(\alpha(L), L; \Delta, \Sigma) \cdot \Sigma^{-1} \Delta.$$  (54)

Note that we minimize $c(\alpha(L), L; \Delta, \Sigma)$ for values of $L$ greater than some minimum value $L_{\min}$, since (51) is only valid under condition $\Delta^T \Sigma^{-1} \Delta < z^2_{\alpha(L)}$. The value of $L_{\min}$ will depend on the function $\alpha(L)$.

A reasonable dependency for $\alpha$ and $L$ is

$$\alpha(L) = e^{-rL},$$  (55)

for some rate parameter $r$. Note that we are only allowed to have $\alpha(L)$ decline less fast than $e^{-L^2}$. Since in our model the profits are normally distributed, we
can thus not impose tails that decline faster than the tail of a normal distribution.

Since we used the constraints $L = 2$ and $\alpha = 0.1$ in our previous subsection, we will choose $r = -\frac{1}{2} \log(0.1)$, so we have $\alpha(2) = 0.1$.

Since the definition of $c(\alpha, L; \Delta, \Sigma)$ is only valid in case of $\Delta^T \Sigma^{-1} \Delta < z_\alpha^2$ we need to solve

$$z_\alpha^2 = \Delta^T \Sigma^{-1} \Delta.$$  

This leads to

$$z_\alpha = \pm \sqrt{\Delta^T \Sigma^{-1} \Delta},$$

(56)

Using $z_\alpha = \Phi^{-1}(\alpha)$ and $\alpha(L) = \exp(-rL)$ we have

$$\Phi^{-1}(\exp(-rL)) = \pm \sqrt{\Delta^T \Sigma^{-1} \Delta},$$

which leads to

$$L = -\frac{1}{r} \log \left(\Phi(\pm \sqrt{\Delta^T \Sigma^{-1} \Delta})\right).$$

Note that

$$L_1 := -\frac{1}{r} \log \left(\Phi(\sqrt{\Delta^T \Sigma^{-1} \Delta})\right) < -\frac{1}{r} \log \left(\Phi(-\sqrt{\Delta^T \Sigma^{-1} \Delta})\right) := L_2,$$

so we have to set $L_{\text{min}} = L_2$ in equation (54).

Since in our theoretic model $Y_{1,t+1} - Y_{1,t}$ is just white noise, our optimal predictor $P_{\text{opt}}$ will always predict a price change of 0. We thus have $\Delta = (0, \Delta_2)$. For $\Delta_2 = 1$, we find $L_1 = 0.32$ and $L_2 = 1.02$. We also find $\tilde{w} = 4.37\Sigma^{-1} \Delta$ corresponding to a critical value $L_{\text{critical}} = \arg\min_{L > L_{\text{min}}} c(\alpha(L), L; \Delta, \Sigma) = 3.62$ and $\alpha_{\text{critical}} = \exp(-rL_{\text{critical}}) = 0.016$, see figure 10.

In figure 11 we plotted the ellipse shaped restrictions of the weights in $(w_1, w_2)$-space for $\Delta = (0, 1)$ and $\alpha = \exp(-rL)$. As expected, the ellipses start out big for low values of $L$ and get smaller as $L$ grows. Around the critical value $L_{\text{critical}} = 3.62$ the ellipses stop getting smaller and we are left with a non empty intersection for possible values of $w$. In the same figure two vectors are drawn. One stands in the direction of the predicted price change $\Delta = (0, 1)$ and the other vector is the optimal weights vector $\tilde{w} = 4.37\Sigma^{-1} \Delta$. This illustrates that generally, these two vectors do not stand in the same direction. In fact, the optimal weight $\tilde{w}$ is the vector on the boundary of the ellipse that has the largest inner product with $\Delta$.

In figure 12 the critical values for $L$ and $\alpha(L)$ are plotted as a function of the predicted price change $\Delta = (0, \Delta_2)$ of our optimal predictor $P_{\text{opt}}$. We see that the critical values for $L$ are fairly large, and the corresponding values $\alpha(L)$ very small. This thus shows that using the function $\alpha(L) = \exp(-rL)$ leads to a strategy that is restricted most by the condition that having a very big loss needs to be very rare.
Figure 10: Plot of $c(\alpha(L), L; \Delta, \Sigma)$ as a function of $L$ with $\alpha(L) = \exp(-rL)$ and $\Delta = (0, 1)$ for $L > L_{\text{min}}$.

Figure 11: Ellipses in $(w_1, w_2)$-space indicating allowed values for $w$ corresponding to $L$ and $\alpha(L) = \exp(-rL)$ for values of $L > L_{\text{min}}$ and $\Delta = (0, 1)$. In black an arrow is drawn in the same direction as $\Delta$. In green, the optimal weights vector $\tilde{w}$ is drawn.
Figure 12: The critical values of $L$ and $\alpha(L)$, i.e. the values for which the constraint of losing more than $L$ is precisely $\alpha(L)$, are plotted as a function of the predicted price difference $\Delta = (0, \Delta_2)$ by our optimal predictor $P_{\text{opt}}$. 

![Graph showing the critical values of $L$ and $\alpha(L)$ plotted against $\Delta_2$]
4.6 Predicting the covariance matrix

Up to now our strategy is as follows. At each time \( t \) we use our predictor \( P(X_t) \) to predict a price change \( \Delta = P(X_t) - Y_t \) for the following day. We calculate the optimal weights of each product \( Y_{i,t} \) in our portfolio according to equation (54):

\[
\hat{w} = \min_{L > L_{\text{min}}} \ c(\alpha(L), L; \Delta, \Sigma) \cdot \Sigma^{-1} \Delta.
\]  

(57)

In our simulation model (16) we have used multiple predictors for \( \Delta \), such as the optimal predictor \( P_{\text{optimal}} \) and our Random Forest predictor \( P_{RF} \). Before we can determine the optimal weights \( \hat{w} \) we also need a predictor for the covariance matrix \( \Sigma_t = \text{Cov}(Y_{t+1} - P(X_t)|\mathcal{F}_t) \). Note that in our model \( \Sigma \) is a constant matrix and we can thus define an optimal predictor

\[
\hat{\Sigma}_{\text{opt}} := \Sigma.
\]  

(58)

In case we use our Random Forest predictor \( P_{RF} \) we need another way to come up with a prediction of \( \Sigma_t \). We propose a method that uses another Random Forest to estimate \( \Sigma_t \).

The idea of this method is to apply our predictor \( P_{RF} \) on a data set, calculate the empirical squared differences \( \delta_{i,j}^t = (Y_{i,t+1} - P_{RF}(X_t)_j)^2 \) between the target value and our predictor at each time \( t \) and use \( \delta_{i,j}^t \) as a target variable to fit a new Random Forest, where \( i \geq j \) and \( i,j = 1, \ldots k \) with \( k \) the number of products. We will do this in two ways:

(a) Fit a Random Forest predictor \( P_{RF} \) on the whole training set. Use this predictor to calculate the squared differences \( \delta_{i,j}^t \) for all points in the training set. Use \( \delta_{i,j}^t \) as target values to fit a new Random Forest on the whole training set.

(b) Randomly equally divide the training set \( K \) times into two sets. Each time, fit a Random Forest predictor \( P_{RF}^l \) on one of the sets and compute the squared differences \( \delta_{i,j}^t \) on the other set. On this set, fit a Random Forest predictor \( \hat{\Sigma}_{RF} \) to predict \( \delta_{i,j}^t \). Finally, combine these \( K \) predictors into one by averaging, i.e. \( P_{RF} = \frac{1}{K} \sum_l P_{RF}^l \) and \( \hat{\Sigma}_{RF} = \frac{1}{K} \sum_l \hat{\Sigma}_{RF}^l \).

Note that we use a Random Forest with multi outputs and target values \( \delta_{i,j}^t \) for \( i \geq j \) and \( i,j = 0, \ldots k \). The output is thus \((k^2 + k)/2\) dimensional. For the final estimate of our covariance matrix we convert these output values to a \( k \times k \) symmetric covariance matrix. Moreover, our predicted covariance matrix \( \hat{\Sigma} \) might not be positive definite. This will lead to big problems, since the term \( \Delta^T \Sigma \Delta \) in equation (51) might not be positive and we will have a negative value in the square root. To avoid this we take \( \hat{\Sigma} \) to be \((\hat{\Sigma}^T \hat{\Sigma})^{1/2}\), where \( M^{1/2} \) denotes the matrix square root of a matrix \( M \). This is equivalent with replacing all eigenvalues \( \lambda_i \) by their absolute values \(|\lambda_i|\) in the eigenvalue decomposition \( \hat{\Sigma} = QAQ^{-1} \), where \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_k) \) is the diagonal matrix with entries \( \lambda_i \). This way we thus ensure that our predicted covariance matrix \( \hat{\Sigma} \) is positive definite.

Both of these methods have drawbacks. With method (a), it is very likely that our Random Forest will be slightly overfitted on the training data. This will
lead to smaller values of $\delta_t$ and thus an underestimate of the risk. Using method (b) we are more likely to overestimate the risks. Since every predictor $P^1_{RF}$ is fitted on only half of the training data, it is likely to perform worse than the combined predictor $P_{RF} = \frac{1}{K} \sum P^1_{RF}$. We expect bigger values of $\delta_t$ and thus an overestimate of the risk.

In order to test our methods (a) and (b) we first do a simulation study in order to find suitable parameters for fitting our Random Forest that predicts the covariance matrix. We generate 25 times a two dimensional time series of $N = 1000$ points from our model. For four different values of the $\lambda_{leaf}$ parameter, namely $\lambda_{leaf} = 1, 5, 25, 100$ we apply methods (a) and (b). Each simulation we evaluate both methods on average daily profit and Sharpe ratio, where we use the first 80% of the data as the training set and the last 20% as the test set. Recall that the $\lambda_{leaf}$ parameter stands for the minimum number of data points that may be contained in a leaf node. In the next subsection, we will take a closer look at the performance of the predictor of the covariance matrix $\hat{\Sigma}$ for the different values of $\lambda_{leaf}$. For now, we show in table 10 the mean profit and Sharpe ratio for the different strategies, all averaged over the 25 data sets and the four different values of the $\lambda_{leaf}$ parameter.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean profit</th>
<th>Mean Sharpe ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>0.56</td>
<td>4.54</td>
</tr>
<tr>
<td>(a)</td>
<td>0.64</td>
<td>2.82</td>
</tr>
<tr>
<td>(b)</td>
<td>0.37</td>
<td>2.92</td>
</tr>
</tbody>
</table>

Table 10: Comparison between our three different strategies. Results are obtained from 25 data sets of size $N = 1000$ generated from our model. We also averaged over the four values of the parameter $\lambda_{leaf}$.

In figure 13 we compare the combined empirical fraction of losses bigger than $-L$ to our constraint $\alpha(L) = \exp(-rL)$. We clearly see that method (a) takes too much risk. Method (b) however, seems to do very well in satisfying the constraint. Moreover, its mean profit is about $\frac{2}{3}$ of the mean profit of the optimal strategy, we conclude from table 10.

![Figure 13: Empirical fractions of losses bigger than $L$ are compared to our constraint $\alpha(L)$. On the left the empirical fractions are shown in red for method (a), on the right for method (b). In blue, the function $\alpha(L)$ is drawn.](image)
One might think, however, that method (a) is a better method if we could adjust the weights in such a way that the distribution of the daily losses would be in line with our constraint defined by $\alpha(L)$. The argument for showing that is not the case goes as follows. Note that the Sharpe ratios are roughly the same, while the mean profit of strategy (a) is a factor $0.64/0.37 \approx 1.7$ higher. It is reasonable to assume that the Random Forest predictors in strategy (a) and (b) are heavily correlated. The reason that the profit of strategy (a) is factor 1.7 bigger is thus very likely to be that its weights are simply a factor 1.7 bigger, since the Sharpe ratio is the same. Denoting $\alpha_a(L)$ as the empirical fraction of losses bigger than $L$ for strategy (a) and $\alpha_b(L)$ the empirical fraction of losses bigger than $L$ for strategy (b) (right in figure 13), we would then have the relation

$$\alpha_a(1.7L) = \alpha_b(L). \quad (59)$$

Judging from figure 13, we see that the curve for (b) would look very similar to that of (a) if we would multiply all weights in strategy (a) with a factor 1.7, i.e. moving all $L$-coordinates a factor 1.7 closer to the $\alpha$-axis in the left image in figure 13.

To compare the predictions of the Random Forest with that of the optimal predictor, we make a scatter plot of the optimal predictions for $Y_{2,t+1} - Y_{2,t}$ versus the predictions of the Random Forest, see figure 14. We do not do this for $Y_{1,t+1} - Y_{1,t}$, since the optimal predictor for $Y_{1,t+1} - Y_{1,t}$ is always zero. Instead, we make a histogram of the distribution of $P_{RF}(X_t)_1 - Y_{1,t}$. We see that the predictions for the difference in $Y_2$ are indeed correlated, while there is still quite a big spread. We also observe that the estimates for the daily difference for $Y_1$ according to our Random Forest predictor are symmetrically distributed around 0, again with quite a big spread.

![Figure 14: On the left a scatter plot of the optimal predicted daily differences $P_{opt}(X_t)_2 - Y_{t,2}$ versus the predicted daily differences of the Random Forest predictor $P_{RF}(X_t)_2 - Y_{t,2}$. Since the optimal predictor always predict a daily difference of 0 for $Y_{1,t+1} - Y_{1,t}$, we show a histogram of the predicted daily differences $P_{RF}(X_t)_1 - Y_{1,t}$ on the left. Data is obtained over 25 data sets of size $N = 1000$, where we used 80% to fit the Random Forest and 20% for the test set.](image-url)
Figure 15: Scatter plot of the optimal weight $\tilde{w}_2$ of $Y_2$ versus the predicted daily difference $\Delta_2$ of $Y_2$ using $\alpha(L) = \exp(-rL)$, based on our 25 data sets.

Figure 16: Scatter plot of the optimal weight $\tilde{w}_2$ of $Y_2$ versus the predicted daily difference $\Delta_2$ of $Y_2$ based on our 25 data sets for various risk managing functions $\alpha_k(L)$, where $k = 1, 2, 3, 4, 5$.

the predicted daily difference $\Delta_2 = P_{\text{opt}}(X_t)_2 - Y_{2,t}$. It seems that our optimal strategy has a minimal optimal weight, no matter how small the predicted daily difference is. Furthermore, the maximum weight corresponding to the biggest predicted daily price change seems to be only around 1.5 times this minimal weight. One might find this undesirable, since for very small values of the predicted price change we are basically gambling. Furthermore, the factor 1.5 between the minimum and maximum weight seems very small. To broaden this, we might try to set a different risk managing function $\alpha(L)$ that declines less fast than $\alpha(L) = \exp(-rL)$. In figure 16 we have plotted the optimal weights $\tilde{w}_2$ versus the predicted daily price change $\Delta_2$ for functions of the form

$$\alpha_k(L) = c(k) L^{-k}$$  \hspace{1cm} (60)

for $k = 1, 2, 3, 4, 5$ and $c(k)$ such that $\alpha_k(2) = 0.1$ for all $k$. We indeed see that the factor between the weights for large values of $\Delta_2$ becomes bigger for smaller values of $k$. For $k = 1$ and $\Delta_2 > 2$, the factor is about 5 or bigger. Using $k = 1$ however might now be a good idea, since the expectation of our daily losses becomes infinite for $\mathbb{P}(\text{daily losses} > L) = \alpha_1(L) \sim L^{-1}$. For $k = 2$ the resulting weights look already very similar to those when using $\alpha(L) = \exp(-rL)$. For these reasons, we will stick to using $\alpha(L) = \exp(-rL)$ in the rest of this text.
4.7 More data

In our previous simulation study, we only used $N = 1000$ data points. We expect better results if we let the number of data points $N$ grow. To test this, we perform a simulation study similar to the previous one, now with $N = 3000$ data points and 10 data sets. Results are shown in table 11. We conclude that both strategy (a) and (b) perform better. Strategy (a) however is still taking too much risk. For strategy (b), we find a mean profit of roughly $\frac{3}{4}$ of the mean profit of the optimal strategy. To see the correlation between the two predictors, we look at figure 17. We see indeed that the predictions for the daily differences in $Y_2$ are more correlated. Furthermore, the histogram for the daily differences using our Random Forest predictor has a smaller spread.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean profit</th>
<th>Mean Sharpe ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>0.54</td>
<td>4.48</td>
</tr>
<tr>
<td>(a)</td>
<td>0.68</td>
<td>3.31</td>
</tr>
<tr>
<td>(b)</td>
<td>0.41</td>
<td>3.24</td>
</tr>
</tbody>
</table>

Table 11: Comparison between our three different strategies. Results are obtained from 10 data sets of size $N = 3000$ generated from our model.

Figure 17: On the left a scatter plot of the optimal predicted daily differences $P_{opt}(X_{t})_2 - Y_{t,2}$ versus the predicted daily differences of the Random Forest predictor $P_{RF}(X_{t})_2 - Y_{t,2}$. Since the optimal predictor always predict a daily difference of 0 for $Y_{1,t+1} - Y_{1,t}$, we show a histogram of the predicted daily differences $P_{RF}(X_{t})_1 - Y_{1,t}$ on the left. Data is obtained over 10 data sets of size $N = 3000$. 
4.8 A closer look at parameter selection

We have finished building our strategy and have presented some results in the previous subsections. Now it’s time to take a closer look at these results and discuss the parameters in our strategy. Since strategy (a) turned out to take too much risk, we will always use strategy (b). Our strategy consists of roughly 6 parameters:

- the number of trees $N_{\text{tree}}$ used in our fitting the Random Forest for our predictor $P_{\text{RF}}$.
  the $\lambda_{\text{leaf}}$ parameter used when fitting this Random Forest.
- the number of trees $N_{\text{tree}}$ used in our fitting the Random Forest for our predictor of the covariance matrix $\hat{\Sigma}$.
  the $\lambda_{\text{leaf}}$ parameter used when fitting this Random Forest.
- the number of divisions of our training set $K$.
- the number of random projection $Z_t$ to add to our input variables.

We will fix most of the parameters. We will use $K = 10$ divisions of our training set. Combined with $N_{\text{tree}} = 100$ trees in each forest and 100 random projections, we found consistent results when running our strategy several times on the same data. Furthermore, we choose $\lambda_{\text{leaf}} = 10$ for the Random Forests involved in making our predictor $P_{\text{RF}}$. We don’t want to choose this value too low, since it might lead to overfitting, but we also don’t want it to be too high since our predictor won’t be able to make very specific predictions.

We will discuss the choice of the $\lambda_{\text{leaf}}$ parameter for the Random Forests that make up the predictor for the covariance matrix $\Sigma$. Since in our model the covariance matrix $\Sigma$ constant, the best prediction would of course be to average over all the squared differences $\delta_{ij}^t$ in the training set. We expect better results if we take a large value of $\lambda_{\text{leaf}}$. By doing this, the Random Forest is forced to use an average of at least 100 points in each tree in predicting $\Sigma$, and thus come closer to the average of the whole training set. We expect the Random Forest to perform worse for small values of $\lambda_{\text{leaf}}$, since it would probably overfit on the training data.

In table 12 we compare the strategies and their prediction of the covariance matrix $\Sigma$ when using $\lambda_{\text{leaf}} = 1, 5, 25, 100$. For $\lambda_{\text{leaf}} = 1$ we find much higher values for the means of the prediction of $\Sigma_{11}, \Sigma_{12}$ and $\Sigma_{22}$. This leads to a strategy that takes less risk, thus leading to a lower mean profit. Since the standard deviation in profit is also lower, we see that the Sharpe ratios between the four parameters are comparable. This is quite natural, since they all use the same predictor $P_{\text{RF}}$ in predicting the daily price changes $\Delta$. Furthermore, we indeed see that for bigger values of $\lambda_{\text{leaf}}$ the mean of the predictions of all entries of the covariance matrix $\Sigma$ decline. For $\lambda_{\text{leaf}} = 100$ we find mean values of our predictor that are in line with the empirical values of $\Sigma_{11} = 4.30$ and $\Sigma_{22} = 5.46$, where we averaged over the whole training set.

We want to further invest the performance of our estimator $\hat{\Sigma}$ for the different parameter values of $\lambda_{\text{leaf}}$. Firstly, we are going to test our assumption about the normality of the distribution of the difference between our predictor and the
Table 12: Comparison the performance of our strategy when using different values of $\lambda_{\text{leaf}}$ parameter, where we used the same 25 data sets as we used in section 4.6 consisting of $N = 1000$ data points each. Note that the theoretical values when using the optimal predictor are $\Sigma_{11} = 4$, $\Sigma_{12} = 2$ and $\Sigma_{22} = 5$.

<table>
<thead>
<tr>
<th>$\lambda_{\text{leaf}}$</th>
<th>Mean profit</th>
<th>Std. dev. profit</th>
<th>Mean Sharp</th>
<th>Mean $\hat{\Sigma}_{11}$</th>
<th>Mean $\hat{\Sigma}_{12}$</th>
<th>Mean $\hat{\Sigma}_{22}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.32</td>
<td>1.83</td>
<td>2.81</td>
<td>6.2</td>
<td>3.99</td>
<td>9.38</td>
</tr>
<tr>
<td>5</td>
<td>0.37</td>
<td>1.95</td>
<td>3.05</td>
<td>4.95</td>
<td>2.71</td>
<td>6.52</td>
</tr>
<tr>
<td>25</td>
<td>0.36</td>
<td>1.99</td>
<td>2.91</td>
<td>4.57</td>
<td>2.29</td>
<td>5.68</td>
</tr>
<tr>
<td>100</td>
<td>0.38</td>
<td>2.08</td>
<td>2.92</td>
<td>4.31</td>
<td>2.09</td>
<td>5.40</td>
</tr>
</tbody>
</table>

target value (30). Secondly, we investigate if there is a relation between the predicted daily price change $\Delta$ and the estimated covariance matrix $\Sigma$. Concerning the former, we note that under assumption (30) we have at time $t$

$$\frac{\hat{Y}_{i,t} - Y_{i,t}}{\sqrt{\hat{\Sigma}_{ii,t}}} \sim N(0,1),$$

(61)

where $\hat{Y}$ denotes our Random Forest prediction $P_{RF}(X_t)$. In figure 18, histograms of the empirical distribution are shown for the various values of the $\lambda_{\text{leaf}}$ parameter, for $i = 2$. Alongside these histograms, we show a scatter plot of the predicted daily price changes $\Delta_2$ and the corresponding diagonal element $\Sigma_{22}$ of the predicted covariance matrix. We choose to only display the figures for $Y_2$, since it’s the only predictable variable. Moreover, the figures for $Y_1$ look very similar.

We note that the differences (61) indeed look normally distributed. This is not very strange, since the difference $Y_{2,t+1} - Y_{2,t}$ is also normally distributed. For $\lambda_{\text{leaf}} = 25, 100$ we find that the variance of the empirical distribution is indeed around 1. For the two smaller parameter values we find a variance less than 1. This is in line with our earlier observation that the predicted variances in $\Sigma$ were too big, see table 12.

Looking at the scatter plots in figure 18, we confirm again that for low values of $\lambda_{\text{leaf}}$, the estimates of $\Sigma_{22}$ are way too big. For the lowest value of $\lambda_{\text{leaf}}$ we see many instances in which the predicted value of $\Sigma_{22}$ was higher than 20, compared to a historical empirical mean value of $\Sigma_{22} = 5.46$. Overfitting might be an explanation for this. We also see that for $\lambda_{\text{leaf}} = 100$, the values of $\Sigma_{22}$ lie around the real value of $\Sigma_{22} = 5.46$. Furthermore, we do not see a particular correlation between $\Delta_2$ and $\Sigma_{22}$. This is good to see, since the variance is constant and thus independent of the predicted price change $\Delta_2$, at least for the optimal predictor $P_{\text{opt}}$. Since the optimal predictor and the Random Forest predictor are positively correlated, we do not expect a dependency of $\Sigma_{22}$ on $\Delta_2$ when using our Random Forest predictors.
Figure 18: For $\lambda_{\text{leaf}} = 1, 5, 25, 100$ we display in ascending order a histogram of the distribution of the difference between the predicted and the real value value of $Y_2$ divided by the predicted standard deviation $\sqrt{\hat{\Sigma}_{22}}$ (61), as well as a scatter plot of the predicted daily price change $\Delta_2$ and the estimated variance $\hat{\Sigma}_{22}$ of $Y_2$. On the left, the best normal fit is also plotted, along with its mean and standard deviation.
4.9 The ARCH model

Up to this point, we generated data from our model (16)

$$Y_{t+1} = AY_t + V_{t+1},$$

(62)

where $V_{t+1}$ was multivariate normally distributed with some constant covariance matrix $\Lambda$. This is not a very realistic way to model the real financial markets, in which we observe periods of low or high volatility. To incorporate this, we use a model inspired by the Autoregressive conditional heteroskedasticity (ARCH) model. We define

$$Y_{t+1} = AY_t + \varepsilon_{t+1},$$

(63)

with

$$\varepsilon_{t+1} = \sigma_{t+1}Z_{t+1},$$

(64)

where $Z_t$ is i.i.d. multivariate normal with mean 0 and the same covariance matrix $\Lambda$ we used in our previous model, and

$$\sigma_{t+1}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \|\varepsilon_{t+1-i}\|^2,$$

(65)

for some constants $\alpha_0, \ldots, \alpha_q$. We thus have that $\varepsilon_{t+1} | \mathcal{F}_t \sim N_k(0, \Sigma_t)$, with $\Sigma_t := \sigma_{t+1}^2 \Lambda$. In our further analysis, we will use $q = 10$ together with $\alpha_0 = 0.1$ and $\alpha_i = 0.01$ for $i = 1, \ldots, q$. We choose quite a large value of $q$, because it leads to very clear patterns of high or low volatility, see figure 19. For lower values of $q$, this was not very visible.

![Figure 19: An example of generated data of size $N = 1000$ from our ARCH model (63) on the left. On the right the absolute values of the daily price changes are plotted in blue for $Y_1$ and in green for $Y_2$.](image)

Note that in our new model we can still use our optimal strategy. At time $t$, we know that

$$\varepsilon_{t-i} = Y_{t-i} - AY_{t-1-i},$$

(66)

for $i \geq 0$. From this we can calculate $\sigma_{t+1}$ and we thus know the distribution of $Y_{t+1} - P_{\text{opt}}(X_t) = Y_{t+1} - AY_t$ at time $t$. 

52
We perform yet another simulation study to test the performance of our strategies. This time, we will only consider the optimal strategy and strategy (b) from section 4.6. We will again use different values of the $\lambda_{\text{leaf}}$ parameter in fitting the Random Forests that predict $\Sigma$. Contrary to our previous model, the variance in the ARCH model is time dependent. When choosing a large value of $\lambda_{\text{leaf}}$, we have to average over many points in our training set, and we will not be able to make very specific predictions. We thus expect smaller values of the $\lambda_{\text{leaf}}$ parameter to perform better in our ARCH model.

In table 13 the performance of our strategy is shown on 25 data sets of size $N = 1000$ generated from our ARCH model (63). We see that for a $\lambda_{\text{leaf}}$ parameter of 5, our strategy performs best in terms of Sharpe ratio. The corresponding profit is about 58% of the profit of the optimal strategy, with a slightly lower standard deviation. Furthermore, we indeed see that for larger values of the $\lambda_{\text{leaf}}$ parameter the Sharpe ratio declines. For $\lambda_{\text{leaf}} = 100$, we find a standard deviation of 8.25, indicating that our strategy takes way too much risk.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean profit</th>
<th>Std. dev. profit</th>
<th>Mean Sharpe ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>0.60</td>
<td>2.59</td>
<td>3.73</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 1$</td>
<td>0.28</td>
<td>2.35</td>
<td>1.87</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 5$</td>
<td>0.35</td>
<td>2.34</td>
<td>2.29</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 25$</td>
<td>0.45</td>
<td>3.58</td>
<td>1.98</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 100$</td>
<td>0.61</td>
<td>8.25</td>
<td>1.19</td>
</tr>
</tbody>
</table>

Table 13: Comparison between our two different strategies. Results are obtained from 25 data sets of size $N = 1000$ generated from our ARCH model (63).

In figure 20 we display for each value of the $\lambda_{\text{leaf}}$ parameter a histogram of the differences between the predictor and the target value scaled by the predicted standard deviation

$$\frac{Y_{2,t} - \hat{Y}_{2,t}}{\sqrt{\hat{\Sigma}_{22,t}}}.$$ (67)

Moreover, we make a scatter plot of the real values of the variance $\Sigma_{22,t}$ and the predicted variance $\hat{\Sigma}_{22,t}$. It is important to note that the variance $\Sigma_{22}$ is the variance of the distribution of the differences between $Y_2$ and the optimal predictor. Since the Random Forest predictor and the optimal predictor are correlated, comparing the variances might still give some insights.

The histograms in figure 20 look different compared to our model with constant covariance, see figure 18. In our non-ARCH model, we found that the distribution of the scaled differences (67) looked very much like a normal distribution. In our ARCH model, however, the distribution is more dense around 0, for all values of $\lambda_{\text{leaf}}$. This indicates that our Random Forest often overestimates $\Sigma_{22}$, leading to smaller values of (67).

Just like in our non-ARCH model, we see that the standard deviation of the scaled differences (67) becomes bigger for larger values of $\lambda_{\text{leaf}}$. We, however, still observe a peak in the empirical density around 0. This indicates that our Random Forest still overestimates often, but overall predicts too low values of
Σ_{22}. This is clearly apparent in table 13, where we see the standard deviation in the daily profits growing for larger values of \( \lambda_{\text{leaf}} \).

Regarding the scatter plots in figure 20, we see some strange patterns. For \( \lambda_{\text{leaf}} = 1, 5 \) we see a vague positive correlation, but we also see very big overestimates of \( \Sigma_{22} \). For the larger values of \( \lambda_{\text{leaf}} \) this correlation seems to be gone, but most of the very big overestimates seem to have disappeared as well. For \( \lambda_{\text{leaf}} = 100 \), we see strange horizontal lines. An explanation for this is that the trees are very shallow for a big value of \( \lambda_{\text{leaf}} \), leading to only few possible predictions of \( \Sigma_{22} \).
Figure 20: For our ARCH-model, we display for $\lambda$ in ascending order a histogram of the distribution of the difference between the predicted and the real value divided by the estimated variance $\hat{\Sigma}_2^2$(61), as well as a scatter plot of the estimated standard deviation $\sqrt{\hat{\Sigma}_2^2}$ and the variance of the difference between $Y_2$ and its optimal predictor. On the left, the best normal fit is also plotted, along with it’s mean and standard deviation.
4.10 More input variables

In this subsection we will try to improve our predictor of $\Sigma_t$ in our ARCH-model by adding more input variables. We add rolling windows looking back 10 days of the squared and absolute values of the $j$-day differences, i.e.

$$\sum_{i=t-9}^{t} (Y_{k,i} - Y_{k,i-j})^2$$ (68)

and

$$\sum_{i=t-9}^{t} |Y_{k,i} - Y_{k,i-j}|$$ (69)

for $k = 1, 2$ and $j = 1, \ldots, 9$.

We expect the performance of our predictor $\hat{\Sigma}$ to increase, since the above input variables are a good indication of the variance at time $t$. In table 14 the results are shown on the same data sets we used before. Compared to not adding the input variables (68) and (69), we see a decrease in mean profit for all values of $\lambda_{\text{leaf}}$. However, for 3 of the 4 values of $\lambda_{\text{leaf}}$ the Sharpe ratio did increase.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean profit</th>
<th>Std. dev. profit</th>
<th>Mean Sharpe ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>0.60</td>
<td>2.59</td>
<td>3.73</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 1$</td>
<td>0.26</td>
<td>2.99</td>
<td>2.12</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 5$</td>
<td>0.29</td>
<td>2.20</td>
<td>2.11</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 25$</td>
<td>0.38</td>
<td>2.96</td>
<td>2.08</td>
</tr>
<tr>
<td>$\lambda_{\text{leaf}} = 100$</td>
<td>0.42</td>
<td>4.40</td>
<td>1.54</td>
</tr>
</tbody>
</table>

Table 14: Comparison between the optimal strategy and our strategy that uses the extra input variables (68) and (69). Results are obtained from the same 25 data sets of size $N = 1000$ generated from our ARCH model which we used in making table 13.

In figure 21 we replicate figure 20 for our strategy with the added input variables. We see that for all values of $\lambda_{\text{leaf}}$ the distribution of scaled differences are less peaked around 0, this indicates that the Random Forests are less often overestimating $\Sigma_{22}$. We also see stronger positive correlation between the predicted values of $\Sigma_{22}$ and the real values for our optimal predictor. From this, we conclude that adding the input variables (68),(69) improves the performance of our predictor $\hat{\Sigma}$. 

56
Figure 21: We replicate figure 20, this time using extra input variables $(68),(69)$ in fitting our Forests. For $\lambda_{\text{leaf}} = 1, 5, 25, 100$ we display in ascending order a histogram of the distribution of the difference between the predicted and the real value value divided by the estimated standard deviation $\sqrt{\hat{\Sigma}_{22}}$ (61), as well as a scatter plot of the predicted daily price change $\Delta_2$ and the estimated variance $\hat{\Sigma}_{22}$ of $Y_2$. On the left, the best normal fit is also plotted, along with it’s mean and standard deviation.
4.11 Importance of random projections

Concluding this section we test the importance of the random projections \( Z_i \) introduced in section 3.2. We concluded earlier that using the random projections in our Random Forest input variables improved the accuracy in predicting \( Y_2 \), with a accuracy of 0.5492 in case of a data set of size \( N = 1000 \) and 0.5816 in case of \( N = 10000 \) data points, see tables 7 and 8. We will now test if we also see a improvement in the profit of our strategy. We will test this on 25 data sets of size \( N = 1000 \) from our non-ARCH model. In table 15 we show the results for a strategy that does not use the random projections \( Z_i \) and for a strategy that uses 100 random projections \( Z_i \). We have again used all input variables when using no random projections, and the square root of the number of input variables when using 100 random projections. Moreover, we used \( N_{\text{tree}} = 1000 \) trees in each forest.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy ( Y_2 )</th>
<th>Mean profit</th>
<th>Mean Sharpe ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>no random projections ( Z_i )</td>
<td>0.5561</td>
<td>0.26</td>
<td>2.36</td>
</tr>
<tr>
<td>100 random projections ( Z_i )</td>
<td>0.5653</td>
<td>0.33</td>
<td>2.80</td>
</tr>
</tbody>
</table>

Table 15: Comparison between our strategy that does not use random projections \( Z_i \) as input and our strategy that uses 100 random projections. Results are obtained from 25 data sets of size \( N = 1000 \) generated from our non-ARCH model.

From table 15 we see that using the random projections indeed leads to a bigger profit. Compared to table 7 the values for the accuracy in \( Y_2 \) look similar. Compared to our earlier test of our strategy we see a slightly lower profit of 0.33 to 0.37 as well as a Sharpe ratio of 2.80 compared to 2.92, see table 10. This is likely to be the effect of random fluctuations in our data and in fitting our Random Forest. Indeed, we see that the standard deviation of the total profits in each simulation is about 0.12, so the 0.04 gap seems not significant.

Note that our strategy that uses no random projections uses a total of 66 input variables. These are the 48 original input variables plus the 2·9 input variables we added in section 4.10. The strategy that uses the random projections uses only \( \sqrt{66 + 10 \cdot 100} \approx 33 \) input variables. We multiplied the number of projections by a factor 10, since we also include the 10 lagged random projections \( Z_{i-h} \) for \( h = 0, \ldots, 9 \). So the Random Forest with the random projections uses only half the amount of inputs compared to using no random projections and still it performs better. A possible explanation for this goes as follows. Note that in our model the expectation of \( Y_{2,t+1} - Y_{2,t} \) equals \( 0.1Y_{1,t} - 0.2Y_{2,t} \). The most relevant input variables are thus of the form \( aY_{1,t} - bY_{2,t} \). Note that when we use no random projections the only input variable of that form is \( Y_{1,t} - Y_{2,t} \), this represents only 1/66 of the input variables used. When using 100 random projections, however, we have for each random projection an input variable of the form \( aY_{1,t} - bY_{2,t} \). We thus have 100/1066 relevant input variables. This corresponds to an average of about 3 input variables of the relevant form in a total of 33 input variables. This ratio is significantly higher than the 1/66 ratio when using no random projections. One can imagine that the Random Forest is
more likely to be affected by noise in the non-relevant input variables if this ratio is lower. This might thus lead to a worse performance of the Random Forest.

4.12 Summary

In this chapter, we have developed a strategy and tested it on data generated from our model. We will briefly summarize our findings and discuss the application of our strategy on real market data.

In the first two sections, we determined the weights of the products in our portfolio by maximizing the expected profit while also limiting our risk. The expected profit was calculated by a predicted daily price change $\Delta$ given by a Random Forest, while we at first considered the covariance matrix $\Sigma$ used in determining the risk known. We initially tried to limit our risk by imposing that we would lose more than $L$ per day with a probability of at most $\alpha$. When we tested this on the data generated from our model, we found that this was not suitable for a real trading strategy since our strategy would take very large (sometimes even infinite) weights in case of very large values of the predicted price change $\Delta$. To always ensure finite weights, we introduced the idea to impose the restriction for more values of $L$ and $\alpha$. In fact, we let $\alpha(L)$ be a function of $L$. For suitable functions, such as $\alpha(L) \sim \exp(-L)$ we ensure that our weights are always non-zero and finite.

Since the covariance matrix $\Sigma$ is usually not known, we propose a way to predict it as well. We do this by dividing our training set $K$ times in two parts. On the first part, we fit our Random Forest that predicts the daily price changes $\Delta$. On the second part, we compute the differences between the prediction of the Random Forest and the actual data. We use this in fitting another Random Forest, that tries to predict $\Sigma$.

To mimic the financial market, we introduced a more sophisticated model, namely a higher dimensional version of the ARCH model. In this model, the covariance matrix $\Sigma_t$ is time dependent. In this way, we were able to simulate markets with periods of high or low volatility.

Because we use a Random Forest in predicting the daily price changes and another Random Forest in predicting the covariance matrix $\Sigma$, our strategy has a lot of parameters. A particular parameter we looked into, is $\lambda_{\text{leaf}}$ for the Random Forests that predict $\Sigma$. This parameter is the minimum amount of data points in each leaf node of any tree in the Random Forest. We found that in our initial model, the strategy performed better for high values of $\lambda_{\text{leaf}}$. In our ARCH model, however, we found that our strategy performed better for lower values of $\lambda_{\text{leaf}}$.

Apart from the parameters of our Random Forests, the input variables are even more important. For example, we found that adding variables that are relevant for the prediction of the volatility, such as a rolling average of the squared daily price changes, improved our strategy.

Finally, we investigated the importance of including the random projections $Z'$ (see section 3.2) in our input variables. We indeed found that our strategy performed better if we included them.
In the next chapter, we will test our strategy on real market data. In order to do this, we have to choose all the parameters in our model, as well as the input variables. Since we suspect that the real market is closer to our ARCH model than to our model with constant volatility, we will use the parameters that worked well in our ARCH model. This will be small values for $\lambda_{\text{leaf}}$ for both the Random Forests that predicts the daily price change, as well as the Random Forests that predict the covariance matrix. Furthermore, we will take as the number of input variables in each tree the square root of the total number of input variables, unless otherwise stated.

We will also use all input variables we have introduced up to this point. We do this because we found good results in our ARCH model. The input variables thus are

- The prices from today and the last 9 days, i.e.
  
  $$Y_{j,t-h}$$
  
  for $j = 1, 2$ and $h = 0, 1, \ldots 9$.

- The difference between today and the last 9 days, i.e.
  
  $$Y_{j,t} - Y_{j,t-h}$$
  
  for $j = 1, 2$ and $h = 1, \ldots 9$.

- The difference between the two different prices for today and the last 9 days, i.e.
  
  $$Y_{1,t-h} - Y_{2,t-h}$$
  
  for $h = 0, 1, \ldots 9$.

- Values of the random projections $Z^i_t$ from today and the past 5 days
  
  $$Z^i_{t-h}$$
  
  as well as the random projections minus a moving average
  
  $$Z^i_{t-h} - \frac{1}{50} \sum_{i=t-90}^{t} Z^i_{t-h}$$
  
  for $h = 0, \ldots, 5$. The random projections $Z^i_t$ are defined by
  
  $$Z^i_t = r^1_i Y_{1,t} + r^2_i Y_{2,t}$$
  
  where $r^1_i, r^2_i$ are i.d.d. standard normal.

- Values that help to predict the volatility:
  
  $$\sum_{i=t-9}^{t} (Y_{k,i} - Y_{k,i-j})^2$$
  
  and
  
  $$\sum_{i=t-9}^{t} |Y_{k,i} - Y_{k,i-j}|$$
  
  for $k = 1, 2$ and $j = 1, \ldots, 9$. 

60
5 Real market data

In this section we will test our strategy on real market data. Initially, we will look at the Dutch AEX index and the German DAX index. Later on, we will also look at more indices. We run backtests to evaluate the performance of our strategy on historical data. Unless otherwise stated, we will use the 2500 days up to the July 19th 2017. Since every year has around 250 trading days, this corresponds roughly to a 10 year period. More specifically, our backtests will run from September 11th 2007 to July 19th 2017, unless stated otherwise. In the first subsection we will explain our backtesting setup. After that, we will shortly examine the data we are going to use. In the third subsection we will present the first results of our strategy on the AEX/DAX data. Ather that, we will take a closer look at the performance of our predictor of the covariance matrix $\hat{\Sigma}$. Next we will analyze our results more in depth and we propose a strategy that is always based on mean reversion. After that, we test our strategy when using a much simpler, non machine learning, predictor. Following that, we discuss the implementation of our strategy and consider transaction costs. Finally, we do a brief literature review and we discuss possible improvements to our strategy.

5.1 Backtesting setup

Up to this points, we always used 80% of our data to fit the model, and tested it on the remaining 20%. For a real market strategy, we want to use all information up to day $t$ to make a prediction for next day $t+1$. Since very old data is most likely not very relevant, we only want to look back a certain time period when fitting our forest. For this, we introduce a lookback parameter. Let $\lambda_{lb}$ be the number of previous days we use in fitting our forest. Our trading strategy can be summarized as follows

- At time $t$, use all data from the interval $[t - \lambda_{lb}, t]$ to fit a predictor $P_{RF}$ for the next day prices $Y_{t+1}$ and a predictor $\hat{\Sigma}$ for the covariance matrix, according to method (b) described in section 4.6. We use all input variables we have introduced up to this point, see the list at the end of section 4.12.

- Use these predictors to calculate the optimal weights $\tilde{w}_t$ according to (54).

Our daily profit will then be $\tilde{w}_t^T(Y_{t+1} - Y_t)$.

Since the backtest would take too much time if we would fit new predictors at every time $t$, we fit new predictors every 10 days. At time $t$, we fit $P_{RF}$ and $\hat{\Sigma}$ and use these to execute the above trading strategy for the next 10 days.

We will use the risk managing constraint $\alpha(L) = \exp(-rL)$, with $r$ such that $\alpha(2) = 0.1$. Our constraint for $L = 2$ will thus mean that we impose that the probability of losing more than 2 units (e.g. euro) is less than 10%. Note that the corresponding optimal weights are arbitrarily scalable, meaning that if we would allow all level of losses to be 1000 bigger we would only need to multiply the optimal weights by a factor 1000.
5.2 The data

The data we use is downloaded from Yahoo Finance [3]. Each day, we want to use the closing prices for each product. However, we cannot do this since these prices do not account for events like stock splits or dividend payments. To illustrate this in the case of a dividend payment, suppose that the closing price of a stock is €100 at day $t$, and the stock will pay a €10 dividend at day $t + 1$. Suppose the closing price at day $t + 1$ is €95. If we bought the stock at day $t$ it looks like we lost €5, while we actually made a profit of €5 since we received €10 dividend. Yahoo Finance offers adjusted closing prices, which adjust for dividend payments by multiplying the closing prices of all days before day $t + 1$ by $(1 - \text{dividend}/\text{price}_t)$. In our example, the adjusted closing price of the stock at time $t$ would be $(1 - 10/100) \cdot £100 = £90$. Using this adjusted closing price, we would indeed have £95 − £90 = €5 profit.

In figure 22 the adjusted closing prices for the AEX index and the DAX index are plotted from 2002 to mid 2017, divided by their total mean over the whole period. In the same figure, a scatter plot of the daily differences of the AEX and DAX is shown. In both plots, we clearly see that the two indices are very heavily correlated.

![Figure 22: The AEX and DAX index are plotted from 2002 to mid 2017, divided by their means over the same period, on the left. On the right, a scatter plot of the daily difference from 2002 to mid 2017 of the AEX index versus the DAX index is shown.](image-url)
5.3 First Results

Using the backtesting method described in section 5.1, we present the first results. In figure 23 the results of the backtest on the AEX/DAX data is shown for various values of $\lambda_{lb}$. For these backtests, we used $\lambda_{leaf} = 10$ for the Random Forests that predict the daily price changes and $\lambda_{leaf} = 25$ in predicting $\Sigma_t$. Moreover, we used $N_{tree} = 100$ trees in each Random Forest. Looking at figure 23 we see profits ranging from a little below zero to around 235 over the 10 year period. The Sharpe ratios range from $-0.02$ to 0.76.

By the random nature of the Random Forests, their predictions may vary as we fit them multiple times on the same data set. By increasing the number of trees $N_{tree}$ in a Random Forest the fluctuation in the predictions is reduced. To test if the number of trees $N_{tree} = 100$ we used in making figure 23 is enough to generate consistent results, we try to replicate figure 23 by running the same backtest. Unfortunately, we were not able to replicate the results, see figure 24. Although the Sharpe ratios are similar, the graphs of the profits look different, especially for $\lambda_{lb} = 500$. In both figures, however, using $\lambda_{lb} = 500$ seems to result in the highest profit and Sharpe ratio. We use this value backtest on the same data set twice more, now using $N_{tree} = 1000$ trees in each Random Forest. The results are shown in figure 25. We see that the cumulative daily profits look very similar and thus conclude that using $N_{tree} = 1000$ is sufficient to generate consistent results.

Using $N_{tree} = 1000$ and $\lambda_{lb} = 500$, we display the performance per year of the first backtest in table 16. Note that we only display absolute profits and losses and not % returns. We do this because the % returns will depend on the implementation of our strategy, which we will discuss in section 5.10. Looking at table 16, we see that our strategy performs very well in the last quarter of 2007, with a Sharpe ratio higher than 6. We also see that the Sharpe ratios per year of the our Strategy and that of the AEX or DAX index do not look very correlated. Furthermore, we see that our strategy has three losing years. In 2013 and 2015 our strategy had considerable losses, both of around 20. In 2011 we lost an insignificant amount. Our strategy performed best in the (last quarter of) 2007, 2012 and 2016, with combined profit of around 70% of the total profit over the 10 year period. Note that our strategy outperforms both the AEX and DAX index in terms of Sharpe ratio over the 10 year period. Note, however, that we have not taken transaction costs into account. We will do this in section 5.10.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AEX Sharpe</td>
<td>0.25</td>
<td>-1.93</td>
<td>1.22</td>
<td>0.32</td>
<td>-0.62</td>
<td>0.58</td>
<td>1.24</td>
<td>0.39</td>
<td>0.25</td>
<td>0.44</td>
<td>1.56</td>
<td>0.01</td>
</tr>
<tr>
<td>DAX Sharpe</td>
<td>1.27</td>
<td>-1.57</td>
<td>0.85</td>
<td>0.85</td>
<td>-0.57</td>
<td>1.36</td>
<td>1.63</td>
<td>0.16</td>
<td>0.37</td>
<td>0.35</td>
<td>1.33</td>
<td>0.30</td>
</tr>
<tr>
<td>RF Sharpe</td>
<td>6.39</td>
<td>0.38</td>
<td>0.93</td>
<td>0.35</td>
<td>-0.09</td>
<td>2.22</td>
<td>-0.68</td>
<td>1.71</td>
<td>-0.61</td>
<td>1.59</td>
<td>0.08</td>
<td>0.69</td>
</tr>
<tr>
<td>RF profit</td>
<td>55.3</td>
<td>18.5</td>
<td>20.9</td>
<td>8.2</td>
<td>-3.2</td>
<td>61.0</td>
<td>-20.6</td>
<td>62.8</td>
<td>-20.7</td>
<td>40.8</td>
<td>1.1</td>
<td>226.6</td>
</tr>
</tbody>
</table>

Table 16: Sharpe value and profit of our Random Forest predictor are shown for each year on the AEX/DAX data set using $N_{tree} = 1000$. Furthermore, the Sharpe ratios of the indices themselves are shown. Note that for 2007 and 2017, we only used the last quarter and first half of the year, respectively. Also note that we did not take transaction costs into account.
Figure 23: Cumulative daily profits for our strategy tested on AEX/DAX from mid 2007 to mid 2017, for different values of $\lambda_b$. We furthermore used $\lambda_{\text{leaf}} = 10$ for $P_{RF}$ and $\lambda_{\text{leaf}} = 25$ for $\Sigma$.

Figure 24: Cumulative daily profits for our strategy tested on AEX/DAX for a different realization of our strategy, using the same parameters as in figure 23.
Figure 25: Cumulative daily profits in two backtests the AEX/DAX data. We used the same parameters as in figure 23, but now with $N_{\text{tree}} = 1000$ trees in each forest.
5.4 Performance of $\hat{\Sigma}$

In this section we will take a closer look at the results of our backtest. To see if the strategy satisfied the risk management constraint, we plot the empirical fraction of losses greater than $-L$ in figure 26. We see that our strategy satisfies the constraints quite well, with the empirical fraction of losses lying only a bit above our constraint curve $\alpha(L)$.

To see the performance of our predictor $\hat{\Sigma}$, we plot the distribution of the scaled differences

$$\frac{\hat{Y}_{i,t} - Y_{i,t}}{\sqrt{\hat{\Sigma}_{ii,t}}}$$

(70)

in figure 27. We see that the distribution for both AEX an DAX look very much like normal distribution with mean 0 and standard deviation 1. Similar to the results in our ARCH-model, we see that the distribution is more peaked around 0, compared to a normal distribution. In figure 28, we plotted the predicted standard deviation $\hat{\sigma}_{i,t} = \sqrt{\hat{\Sigma}_{ii,t}}$ for the AEX for each time $t$. Alongside, we plotted an empirical estimate of $\sigma$, which is the sample standard deviation of $\hat{Y}_t - Y_t$ using a rolling window of size 25. The plots for the DAX index look similar, so we will not display them. We see that our predictor $\hat{\Sigma}$ does quite well: it seems to predict the spikes in late 2008, mid 2010 and late 2011 well. It also predicts the period of low volatility from 2012 to 2015, as well as the increase starting in 2015. From 2009 to mid 2010 our predictor seems to overestimate, which coincides with the empirical value of $\sigma = 0.97 < 1$ in figure 27. In figure 29 we display a scatter plot of empirical values of $\sigma$ using a rolling window of size 25 versus our predicted values. We find a correlation coefficient of 0.87, which is a very good result. Note, however, that the rolling window estimate only measures the volatility in the market. Ideally, we would like our Random Forest predictor $\hat{\Sigma}$ to give a better prediction based on the input variables. After all, we would otherwise be better of by using the empirical estimate in our strategy instead of the Random Forest predictor. In section 5.7 we will further investigate using the empirical estimate for $\Sigma$. 

67
Figure 26: Empirical fractions of losses bigger than $L$ are compared to our constraint $\alpha(L) = \exp(-rL)$. In red, the fraction of losses bigger than $L$ are plotted for the AEX/DAX data set using $\lambda_B = 500$ and $N_{tree} = 1000$. In blue, the function $\alpha(L)$ is drawn.

Figure 27: Histograms of the differences $\hat{Y}_{i,t} - Y_{i,t}$ between the predicted prices and the real prices, scaled by the inverse of the square root of the estimated variance $\hat{\Sigma}^{-1/2}$ for $i = \text{AEX (left), DAX (right)}$. The best normal fit is also drawn.
Figure 28: For $\lambda_{ib} = 500$ and $N_{\text{tree}} = 1000$, the predicted standard deviations $\hat{\sigma}_t = \sqrt{\hat{\Sigma}_t}$ for the AEX are plotted for each time $t$ in green. The empirical standard deviation is also plotted, which is the sample standard deviation of the previous 25 data points.

Figure 29: For $\lambda_{ib} = 500$ and $N_{\text{tree}} = 1000$, the empirical standard deviations $\sigma$ (horizontal axis) are plotted versus the predicted standard deviations $\hat{\sigma}_t = \sqrt{\Sigma_t}$ (vertical axis).
5.5 A closer look

Regarding the first backtest in figure 25, we find that 51.3% of the trading were profitable, compared to 51.8% for the second backtest. In figure 30 we show the distribution of the daily profits $\tilde{w}^T(Y_{t+1} - Y_t)$ for the first backtest using a histogram. On average, we make a profit of $\mathbf{226.6}/2500 = \mathbf{0.091}$ per day, with a standard deviation of $\mathbf{2.09}$. At first sight this may seem insignificant, but if we model our daily profits to be independent our total profit has mean $\mathbf{226.6}$ and standard deviation $\sqrt{2500 \cdot \mathbf{2.09}} = \mathbf{104.5}$. With this reasoning, the total profit is quite significant.

![Histogram of the daily profit of our strategy on the AEX/DAX data set using $N_{\text{lookback}} = 500$ and $N_{\text{tree}} = 1000$.](image)

To get insight into how our strategy makes money, we want to look at the optimal weights $\tilde{w}_t$. In figure 31 we show the sum of the absolute values of our positions, i.e.

$$|\tilde{w}_{1,t}Y_{1,t}| + |\tilde{w}_{2,t}Y_{2,t}|,$$

where the 1 index stands for the AEX and the 2 index for the DAX. Note that this represents a value in euro’s. The higher this value, the more risk we take. For example, a position where we are €300 long in the AEX and €300 short in the DAX is more risky than a position where we are only €100 long and short. In figure 31 we clearly see lower values of (71) for higher predicted values of $\Sigma$, see figure 28. For example, during the end 2008 until the beginning of 2009 we see a sudden dip in figure 31. Looking at figure 28, we see that this dip corresponds precisely with a sudden period of very large predictions of $\Sigma$.

It is even more interesting to look at the previous sum without the absolute
Figure 31: The sum of the absolute values of our positions (71) is shown for each time $t$.

values, i.e.

$$\tilde{w}_{1,t}Y_{1,t} + \tilde{w}_{2,t}Y_{2,t}.$$  

This could potentially tell us how our strategy makes the most money. Roughly speaking, we have two kinds of strategies: trending and mean reverting. The trending strategy is based on predicting the movement of the market as a whole and act accordingly. For example, if we think the AEX and the DAX will both rise, we can buy $\mathbf{E}100$ in AEX and $\mathbf{E}100$ in DAX. The mean reverting strategy is based on predicting the AEX relative to the DAX. For example, if we predict that the AEX will rise relative to the DAX, we can buy $\mathbf{E}100$ in AEX and sell $\mathbf{E}100$ in DAX. We make money if $Y_{1,t} - Y_{2,t}$ rises, irrespective of the absolute movements of $Y_{1,t}$ or $Y_{2,t}$.

Note that in our two examples the value of (71) was in both cases $\mathbf{E}200$, while the value of (72) was $\mathbf{E}200$ and $\mathbf{E}0$, respectively. We thus see that for trending strategies we will see high values of (72), while for mean reverting strategies we will see values of (72) around 0.

To investigate what strategy is mostly used by our algorithm, we count the number of instances where both $\tilde{w}_1,\tilde{w}_2 > 0$, $\tilde{w}_1,\tilde{w}_2 < 0$ and where $\tilde{w}_1,\tilde{w}_2$ have different signs. We also calculate the mean profit for these three configurations of the signs of $\tilde{w}_1,\tilde{w}_2$. The results are shown in table 17.

Looking at table 17 we see that we rarely have both positive or negative weights, only in about 12% of the cases. Furthermore, we see that the mean profit for both positive weights is much higher compared to the case where the weights are both negative. The average for these two cases weighted by their occurrence
Table 17: The percentage of occurrence and the mean profit are shown for different configurations of the signs of $\tilde{w}_1, \tilde{w}_2$.

<table>
<thead>
<tr>
<th>Occurrence</th>
<th>$\tilde{w}_1, \tilde{w}_2 &lt; 0$</th>
<th>$\tilde{w}_1, \tilde{w}_2 &gt; 0$</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean profit</td>
<td>€0.049</td>
<td>€0.184</td>
<td>€0.090</td>
</tr>
</tbody>
</table>

is however almost equal to the mean profit of the third case where $\tilde{w}_i < 0, \tilde{w}_j > 0$.

Only looking at the signs of $\tilde{w}_i$ is a little bit unsophisticated. Consider buying €10 worth of AEX, while selling €300 worth of DAX. This is clearly a trending strategy but it would not be recognized as such in table 17. In figure 32 we display scatter plot of the sum of the absolute values (71) versus the the sum of the values (72), as well as a histogram of the values (72).

![Figure 32](image)

Regarding the left image in figure 32, we first notice the two straight lines with slope ±1. These lines represent cases where $\text{sgn}(\tilde{w}_1) = \text{sgn}(\tilde{w}_2)$. Furthermore, we see that there does not seem to be a correlation between the sum of the absolute values and the regular sum. From the histogram on the right, we conclude that there are indeed instances where the sum of the positions (72) is around 0, indicating a mean reverting strategy. We also conclude, however, that the sum of the positions are mostly significantly larger or smaller than 0, indicating a trending strategy. Lastly, we note that the histogram is roughly symmetric around 0, this coincides with our earlier observation from table 16 that our profits do not seem to be correlated with the movements of the whole market.

We wonder how our strategy performs if we enforce it to use a mean reverting strategy, i.e. have the sum of all positions (72) equal to zero. We can accomplish this in the following way. Suppose we are at time $t$, and we are looking at $k$ products. Just like before, we use (54) to calculate the optimal weights $\tilde{w}_t$. The idea is to take the projection of these weights on the plane $\{w \in \mathbb{R}^k \mid w^T Y_t = 0\}$. Let $\tilde{w}_t^0$ denote this projection. Since $Y_t/\|Y_t\|$ is the normal vector to this plane, we have

$$\tilde{w}_t^0 = \tilde{w}_t - \frac{Y_t^T \tilde{w}_t}{\|Y_t\|^2} \cdot Y_t.$$  (73)
In figure 33 the first backtest on the AEX/DAX data set (see figure 25) is once again shown. We used the weights \( \tilde{w}_t \) of this backtest to calculate our projected weights \( \tilde{w}_0^t \). The profits using the projected weights are also shown. Quite surprisingly, we find that our strategy still performs well. We find a slightly lower total profit, but the same Sharpe ratio, which implies a slight decrease in standard deviation. The reason for this is that the projected weights \( \tilde{w}_0^t \) can now take arbitrarily small values, in contrast to our previous strategy, see figure 15. Suppose our optimal weights \( \tilde{w}_t \) stand roughly in the direction of \( Y_t \), say \( \tilde{w}_t = \lambda Y_t \). This coincides with a prediction that the market as a whole will rise (in case of \( \lambda > 0 \)) or will fall (in case of \( \lambda < 0 \)). Our projected weights \( \tilde{w}_0^t \) will now be

\[
\tilde{w}_0^t = \lambda Y_t - \lambda \frac{\|Y_t\|^2}{\|Y_t\|^2} \cdot Y_t = 0.
\]

We thus see that we indeed take no position if we predict an equal rise of all products in the market. This is precisely what we want if we want to use a mean reversion strategy and not be exposed to market movements.

Figure 33: In blue, the cumulative profit using our usual weights \( \tilde{w}_t \) is shown on the AEX/DAX data. In green, we display the profits using the projected weights \( \tilde{w}_0^t \).
5.6 Comparison with simple predictor

In this section, we will compare the performance of our strategy using Random Forests with another method. We will fit a Vector Autoregressive VAR(1) model on the AEX/DAX data, in which we assume

\[ Y_{t+1} = AY_t + \varepsilon_t, \]

(75)

where \( A \) is a \( 2 \times 2 \) matrix and \( \varepsilon_t \) i.i.d. multivariate normal with mean 0 and covariance matrix \( \Gamma \). We will fit \( A \) and \( \Gamma \) using the Python package [2]. We will not discuss the fitting methods in this text. Just as we did in fitting our Random Forests predictors, we will have a lookback window of \( \lambda_{lb} = 500 \). One might argue that using a smaller lookback window would lead to better performances of the VAR predictors. However, we found that for \( \lambda_{lb} = 50 \) the predictor of \( \Gamma \) performed very badly. This very often led to very large values of the optimal weights and thus an unusable strategy.

Since the predictors are very simple, we expect the performance of these predictors to be very poor. We want to make this comparison, since it also involves a predictor for the next day values and estimation of the variance between the actual values and the predicted value. Because of this, we can simply use our strategy to calculate the optimal weights \( \tilde{w} \) using equation (54).

In figure 34 the profits are shown using the VAR(1) model (75). Moreover, we display the empirical fraction of losses bigger than \( L \). Quite surprisingly, the strategy seems to make a profit. However, we also see big fluctuations in the graph of the profits, resulting in a Sharpe ratio of only 0.21. Concerning the image on the right, we see that the strategy takes way too much risk and does not satisfy our risk managing constraint at all.

Figure 34: On the left, the profits are displayed using the predictors from our VAR(1) model. On the right, the fractions of losses bigger than \( -L \) are plotted in red. In blue, our risk managing curve \( \alpha(L) = \exp(-rL) \) is drawn.
5.7 Empirical estimate of $\Sigma$

We will investigate the use of an empirical estimate of $\Sigma$, instead of using a Random Forest predictor. As we already mentioned at the end of section 5.4, we ideally want our Random Forest predictor $\hat{\Sigma}_t$ to predict more than just the volatility in the market. In figure 35 we display a histogram of the predicted price changes $\Delta = P_{RF}(X_t) - Y_t$ for the AEX as well as a histogram for the daily price changes $Y_{t+1} - Y_t$ for the AEX. The figures for the DAX look the same, so we will not display them.

![Histograms](image)

Figure 35: On the left a histogram of the distribution of the predicted daily price changes $\Delta = P_{RF}(X_t) - Y_t$ is shown for the AEX. On the right we display a histogram of the distribution of the daily price changes $Y_{t+1} - Y_t$ for the AEX. On the left we find a standard deviation of 5.08, on the right we find a standard deviation of 1.38.

From figure 35 we conclude that the standard deviation in the predicted price changes is almost a factor 4 smaller than the standard deviation of the actual price changes. We therefore suspect that the covariance of $Y_{t+1} - P_{RF}(X_t)$ is not very dependent of the prediction $P_{RF}(X_t)$. If this is the case, we could also just use a empirical estimate of the covariance matrix of $Y_{t+1} - P_{RF}(X_t)$ using a rolling window. This would save a lot of time. The estimation of $\Sigma$ takes a lot of time, since we need to fit it on various partitions of the training set as described in section 4.6.

Using a lookback window of $\lambda_{lb} = 100$ for estimating the covariance matrix by taking the empirical covariance matrix, the profits on the AEX/DAX data are shown in figure 36.

From figure 36 we see that the profits using the two methods are very correlated. Note that the empirical method starts 100 days later than the Random Forest methods, and thus misses the very profitable end of 2007. Calculating from 2008 and onwards we find a total profit of 200.6 for our Random Forest method and a corresponding Sharpe ratio of 0.64. For our empirical method, we find a total profit of 205.8 with a Sharpe ratio of 0.63. We thus conclude that using an empirical estimate of the covariance matrix $\Sigma$ yields the same results as using the Random Forest method from section 4.6.
Figure 36: The cumulative daily profits on the AEX/DAX data are shown in green for using an empirical estimation of the covariance matrix $\Sigma$ with a look-back window of size $\lambda_{lb} = 100$. In blue, the cumulative daily profits using Random Forests to predict $\Sigma$. 
In figure 37 we ran more backtest for different lookback windows for calculating the empirical covariance matrix. Note that we used the same Random Forest predictor $P_{RF}$ for the prediction of the daily differences.

![Cumulative daily profits on the AEX/DAX](image)

Figure 37: Cumulative daily profits on the AEX/DAX/ are shown for different lookback windows for calculating the empirical covariance matrix. We used the same Random Forest predictor $P_{RF}$ for each of the different lookback windows.

From figure 37 we conclude that for lookback windows of size 25, 50, 100, 250 the profits, the Sharpe ratio and the standard deviation in daily profit are very similar, once we correct for the very profitable period late 2007 which is only seen with a window size of 25.
5.8 More indices

In this section, we will test our strategy on four indices. Apart from the AEX and DAX index, we add the French CAC index as the Euro Stoxx 50 index. The latter consists of the 50 most important stocks in the eurozone. These four indices are still heavily correlated since there is a lot of overlap between them. Our data source [3] seems to miss a couple of data points for the Euro Stoxx 50. If this was the case at day $t$ we deleted this data point. In our backtest, day $t-1$ and $t+1$ will thus be viewed as consecutive days. This only occurred about 80 times in the data set, and we will assume that this will not influence our results greatly. Since we still used 2500 days in our backtest our backtest will now start mid 2007 instead of in the last quarter of that year.

As discussed in the previous section, we will use an empirical estimate of the covariance matrix, looking back 50 days. For two instances of our strategy on the same data set and using the same parameters, the results are shown in figure 38.

![Figure 38: The cumulative daily profits on the AEX/DAX/CAC/Euro Stoxx 50 are shown, for two instances of our backtest. We used an empirical estimate for the covariance matrix looking back 50 days.](image)

In figure 38, we see that our strategy makes more profit than on the AEX/DAX data set, see figure 25. The Sharpe ratios, however, are similar for the two data sets. From the discussion in 4.6 we thus expect that the strategy on the AEX/-DAX/CAC/Euro Stoxx 50 data set takes too much risk. Indeed, in figure 39 we see that the empirical fraction of losses lie well above our constraint curve $\alpha(L)$.

From this we conclude that our strategy performs worse when using more prod-
Figure 39: Empirical fractions of losses bigger than $L$ on the AEX/DAX/-CAC/Euro Stoxx 50 are plotted in red. In blue, the curve $\alpha(L)$ is drawn.

This is quite natural, since the covariance matrix has now size $4 \times 4$, which is a lot harder to estimate than in the 2-dimensional case. The fact that the two instances of the same backtest in figure 38 seem to differ quite a bit can also be explained in the same way.
5.9 Omitting the random projections

Just like we did for the data in our model in section 4.11, we are interested if adding the random projections $Z_t^i$ to our Random Forest as defined in section 3.2 is beneficial. We saw earlier that our strategy indeed performed better when adding these as input variables to our Random Forest. We run a backtest using $N_{\text{tree}} = 1000$ looking back $\lambda_{\text{lb}} = 500$ days and using our usual predictors $P_{\text{RF}}$ and $\hat{\Sigma}$ from section 4.6. The result is shown in figure 40.

![Figure 40: Backtest on the AEX/DAX data without using random projections $Z_t^i$ as input variables for our Random Forests.](image)

To our surprise, the strategy performs better compared to our strategy that made use of the random projections, see section 5.3. We find a bigger total profit and a bigger Sharpe ratio of 0.93.

Note that this can be the result of noise in our backtesting caused by the random nature of Random Forests. We argue however that this is unlikely. Judging from figure 25 we see that the Sharpe ratio can differ between different runs of our backtest, since we found Sharpe values of 0.69 and 0.75 between the same two backtests. However, the graphs of the cumulative daily profits look very similar. If we compare them with the graph in figure 40, we see that when using no random projections, the profits look different. For example, the big drop between late 2010 and the beginning of 2011 is now much smaller. We thus conclude that it is likely that the better performance of the strategy is not due to randomness in our backtest.

However, it is very important to note that this better performance in our backtests does not necessarily indicate that omitting the random projections is indeed
better. For example, we see that the profits when using no random projection start to diminish after 2012. In the period 2012 – 2017 the strategy only makes a profit of around 80. The profit of our strategy with the random projections makes an average profit of around 150 in the same period, see figure 25. If we would have ran our backtests on that period, we would have concluded that using the random projections is indeed very beneficial.
5.10 Implementing a real trading strategy and estimating the associated costs

Up to this point, we always assumed that we could buy or sell any amount of an index at the exact closing price. Moreover, we ignored transaction costs. In this section we will discuss the implementation of our algorithm in a real world trading setting and investigate the effects of transaction costs on our profits.

First of all we will discuss buying at the closing price. One might say this will lead to look-ahead bias, since the closing price is not known before a trading day has ended. We will however assume that buying or selling a couple of seconds before the closing price will not influence our resulting profit. Moreover, some brokers offer a special order type called Market-on-Close. This order type will ensure that one can buy/sell as close to the closing price as possible.

To make an estimate of the transaction costs, we will first discuss two ways to implement our algorithm. Note that the AEX and DAX are indices, which we cannot buy or sell directly. We can however buy a so-called Exchange-Traded Fund (ETF). This product is designed to closely mimic another product, such as the AEX index. Another way of implementing our algorithm is through futures contracts. A futures contract (or future for short) is an agreement between two parties to buy or sell something at a predetermined price \( K \) and specified time \( T \) in the future. This contract itself can be traded up to any time before the expiry \( T \). Note that there is no value transferred when buying or selling a future. This is because at time \( t < T \), the price of the future will be such that the expected value of the future is precisely 0, under some assumptions that we will not discuss. Futures are settled on a daily basis, meaning that if you bought a future at day \( t < T \) and the underlying value of the future ended up \( €10 \) higher at day \( t + 1 < T \), you would roughly receive \( €10 \) that day. For the indices, futures usually come with leverage. This is a factor 200 for the AEX and a factor 25 for the DAX. Considering the AEX future trades at around \( €530 \) at the time of writing, one AEX future represents \( 200 \times €530 = €106,000 \) in underlying value. For the DAX this is even higher, with one future representing about \( €300,000 \) of underlying value. In order to buy or sell a future, brokers require that you have enough funds in your account to cover your losses. Usually this is not more than 10% of the underlying value, meaning that we only need \( €30,000 \) to buy a DAX future with an underlying value of \( €300,000 \). Note that the price of a future can differ from that of its underlying value. This is because of events, like dividend payments and interest, are taken into account. We have always used closing prices of the index, and not the closing prices of the corresponding future, but we will assume that our profits will approximately be the same. The transaction costs for futures are almost non-existent. For one DAX future the transactions costs are only a couple of euro’s. Futures are thus an extremely cheap way to have a big exposure \( (€300,000 \) in case of the DAX) with very low transaction costs. For ETF’s, these costs are much higher. Moreover, ETF’s don’t offer leverage. We thus conclude that our algorithm is best implemented using futures. It must, however, be noted that a big disadvantage of using futures is that we cannot buy or sell an arbitrary amount, since our exposure in for example the DAX index must be a multiple of €300,000.
In this text, we never evaluated the performance of strategies in terms of % returns. From the above discussion we see that the % depends on the implementation of our strategy. If we use ETF’s we need the full amount of the investment if we buy. If we short sell, however, we also need some kind of margin to ensure that we will be able to buy back the borrowed ETF. For futures we only need the required margin, which is usually around 10% of invested amount. Note that this margin requirement is even lower if we have long and short positions in correlated products. Note that the margin requirements can also change over time, for example if the market is more volatile or if the broker decided change it for an other reason. Margin requirements are thus often hard to estimate and differ between different brokers. For this reasons we will have not calculated the % returns in our backtest.

While transaction costs are thus very small, we also have to account for the bid-ask spread. The highest price that someone offers for a product is called the best bid. The lowest price that someone wants to sell the product for is called the best ask. The difference between these two is called the bid-ask spread. A higher trading volume in a product usually corresponds with a lower bid-ask spread. For example, the AEX future has at the time of writing a spread of only €0.10. Suppose we could buy one future and instantaneously sell it, we would lose $200 \times \€0.10 = \€20$. The closing prices that we use in our backtest are the closing prices of the indices. Suppose for now that these are equal to the prices of the corresponding futures. Note that the closing price represents the last trade. We do, however, not know if this price was the best bid or the best ask price just before the last trade. Assuming we could also have traded at the same price, we have to consider two scenario’s. If the closing price was the best bid price before the last trade, we could have traded at that price if we wanted to sell. If we wanted to buy, we would have had to pay more, namely the bid-ask spread. If the closing price was the best ask price before the last trade, we could have traded at that price if we wanted to buy. If we wanted to sell, we would have received a little bit less, namely the bid-ask spread. With this argument, we would have to pay the bid-ask spread in 50% of the cases. To model this, we will add half of the bid-ask spread in each trade alongside the transaction costs of the broker.

Consider our backtest from section 5.3. Where we previously used $\alpha(L = 2) = 10\%$ in our strategy, we define $\alpha(L) = \exp(-rL)$ such that we want to lose more than $L = \€2000$ in less than 10% of the cases. This will just scale all optimal weights $\tilde{w}_t$ by a factor 1000. Recalling the mean of €0.091 from section 5.5, our mean profit now is €91 per day, with a total profit of around €226,000 over the 10 year period. To give a very rough estimate of the total costs, we round the optimal weights $\tilde{w}_t$ to the nearest multiple of the underlying value of a single future. For AEX this will thus be 200 and for the DAX 25. Let $[\tilde{w}_t]$ denote the rounded optimal weights. We take €2.50 transaction costs per future. For some brokers, this fee is even lower. Using a bid-ask spread of around €0.10 for the AEX and €1 for the DAX, our total costs would be

$$€2.50 + \frac{1}{2} \cdot 200 \cdot €0.10 = €12.50$$

(76)
per AEX future, and

$$\varepsilon 2.50 + \frac{1}{2} \cdot 25 \cdot \varepsilon 1 = \varepsilon 15.00$$

(77)

per DAX future. For day $t$, our total costs will thus be

$$([\tilde{w}_{1,t}] - [\tilde{w}_{1,t-1}]) \cdot \varepsilon 12.50 + ([\tilde{w}_{2,t}] - [\tilde{w}_{2,t-1}]) \cdot \varepsilon 15.00.$$

(78)

For each time $t$ we calculate these values and find an average total costs of €27.71 per day, which is a hefty 30% of our average daily profit of €91. Since the standard deviation in daily profit will roughly be the same after subtracting the total costs for each day, we conclude that the corresponding Sharpe ratio will also be 30% smaller, namely around 0.50. Note that these are very rough estimates. To get a better view of the involved costs we would need data with both bid and ask prices.
5.11 Literature review

In the literature, trading strategies involving machine learning predictors like Random Forests often use classification instead of regression. For instance, in [4] classification is used to predict a probability that a stock in the S&P 500 index will go up the next day. For this, the machine learning techniques Deep Neural Networks, Gradient-boosted Trees and Random Forests are used. The S&P 500 index consist of stocks of the 500 biggest American companies. From these stocks, the 10 stocks with the highest predicted probability get bought, while the 10 stocks with the lowest probability get short sold. Backtests show that this method worked very well in the period 1992 – 2001. From there, however, the performance declined. Eventually even leading to negative returns with a quite significant average Sharpe ratio of $-1$ in the period 2010 – 2015, after transaction costs.

In [5] classification using Classification Trees is used in determining stocks that are likely to outperform the market. For constructing these trees, many technical indicators are used as input variables. Technical indicators are derived from historical prices and volume data. A simple example is a moving average of the stock price. More complicated technical indicators include Williams %R defined by

\[
\frac{\max(S_t, \ldots, S_{t-N}) - S_t}{\max(S_t, \ldots, S_{t-N}) - \min(S_t, \ldots, S_{t-N})} \times -100,
\]

indicating how much today’s price differs from the highest price in the last $N$ days.

The strategy is based on buying stocks that get classified as being likely to outperform the market. The strategy performs quite well, with yearly Sharpe ratios between 0.4 and 1.0 after estimated transaction costs. The strategy traded stocks from the Stockholm Stock Exchange.

In [6] classification is used to predict the direction of certain stocks using Random Forests. Among other things, the authors tried to predict whether stocks would be higher priced or lower priced after 3 days. They tested this for stocks of Apple, Microsoft and Amazon and concluded that they could correctly predict this 85% of the time averaged over the three stocks, in a period from 2010 to 2014.

We see from these examples that machine learning is often used for selecting profitable products to buy or sell, instead of determining a trading strategy given a small number of products. The latter is precisely what we wanted to obtain in this thesis. If machine learning is used for forecasting the stock market, a binary output of up or down is often predicted. Moreover, the analysis in mostly focused on the accuracy of the predictions and not about implementing the predictions into a trading strategy.
5.12 Improvements to our strategy

In this subsection we briefly discuss some potential improvements to our strategy.

- **Pair selection.** In this thesis we mostly looked at the AEX index and the DAX index. It is likely to be profitable on other pairs as well, or maybe a basket of multiple correlated/conintegrated products.

- **Input variable selection.** It might be worthwhile to look closer at preprocessing our input data. Our Random Forests might perform better if we normalize the data in some way, or take the log. Moreover, all our input data was based on the historical prices. Prediction might be improved if we also use other input data, for example trading volume. Another example is to input some sentiment score about a certain stock or about the market as a whole. These scores can be based machine learning techniques that analyze news articles, tweets or Google searches about specific topics. We might also try to include technical indicators in our input variables, see the discussion in section 5.11.

- **Trading signals.** In our current strategy we are forced to trade everyday. One might try to implement a strategy that makes use of trading signals, i.e. triggers that occur when a certain condition is met. We might also try to use for frequent data such as minutely prices, and generates triggers based on the the price evolution throughout the trading day.

- **Classification** Instead of regression we might try to use classification. For example, we could try to predict whether the ratio $Y_{1,t+1}/Y_{2,t+1}$ will be greater or smaller than $Y_{1,t}/Y_{2,t}$ based on all information up to time $t$. 

6 Conclusion

In this thesis we have investigated the use of Random Forests in defining a strategy for trading cointegrated pairs. In this section, we will summarize and discuss our findings.

We first introduced a simple model to mimic cointegrated pairs such as the AEX/DAX pair. The prices of the two products is a two dimensional time series \( Y_t \) and was defined as a VAR(1) process by

\[
Y_{t+1} = AY_t + V_{t+1},
\]

where \( A \) is a constant matrix and \( V_t \) are independent multivariate normal distributed random variables with expectation 0 and constant covariance matrix \( \Lambda \). The matrices \( A \) and \( \Lambda \) are chosen in such a way, that \( Y_{1,t} \) is a pure random walk and the linear combination \( Y_{2,t} - 0.5Y_{1,t} \) is an AR(1) model. Moreover, this linear combination is a stationary process that reverts to a certain mean over time.

We first tested the ability of a Random Forest to predict the daily price changes in this model. Because the random noise has mean 0, we know that the expectation of the next day prices is actually \( AY_t \). Since we can’t predict the random noise, this value would be the optimal prediction. Naturally, we found that the Random Forest predicted worse than the optimal prediction in terms of MSE. We however also found that the Random Forest predicted worse than a predictor that would predict no price change at all! At first sight, this was very worrisome. However, we found that our Random Forest managed to predict correctly whether \( Y_{2,t} \) would go up or down more than 55% of the time.

At first, we only used very basic input variables in our Random Forest, namely the prices of the last 10 days, the difference between today’s prices and the last 10 days and the difference between the two prices \( Y_1 \) and \( Y_2 \). We proposed new input variables, namely random projections \( Z_t^i \) of \( Y_{1,t} \) and \( Y_{2,t} \) defined by

\[
Z_t^i = r_{1,t}^i Y_{1,t} + r_{2,t}^i Y_{2,t},
\]

where \( r_{1,t}^i, r_{2,t}^i \) are i.d.d. standard normal. We found that when using this input variables, the aforementioned accuracy increased to about 57%. Note that the optimal predictor, i.e. the predictor that predicts the expectation of \( Y_{t+1} \) given all information at time \( t \), had an accuracy of about 59%. We concluded that in this simple model, Random Forests perform well in predicting the next day prices.

We want to use these Random Forest predictions to define a trading strategy. In order to do so, we have to define the weights \( w_{i,t} \) of each product \( Y_{i,t} \) in our portfolio. We proposed a way to daily adjust the portfolio by making use of the predicted daily price change of the Random Forest. The weights are chosen in such a way that the expected profit given by the predicted daily price changes is maximized. While maximizing our expected profits, we also want to limit our risk. We proposed a method that imposes that the probability of losing more than \( L \) is at most \( \alpha \). We did this by making an assumption about the distribution of our predicted next day prices. We namely assumed that the distribution
of the next day prices $Y_{t+1}$ is normally distributed with mean $Y_t + \Delta$, where $\Delta$ denotes the predicted daily price change by our Random Forest, and covariance matrix $\Sigma$. With this assumption, the restriction on our weights manifests as an ellipse shaped restriction in $w_1, w_2$-space. We found that this method was not suitable for a real trading strategy, since the weights could take very large values when the predicted price change was very big. Sometimes, the predicted daily price change was so big that the restriction if $w_1, w_2$-space became a hyperbola, leading to infinite weights. To tackle this problem, we limited the probability of losing more than $L$ all values of $L$. To achieve this, we imposed that losing more than $L$ would happen with a probability of at most $\alpha(L) \sim \exp(-L)$ using the same method.

To test our strategy, we need a way to estimate the covariance matrix $\Sigma$ of the distribution of the next day prices $Y_{t+1}$. We proposed a way that also make use of Random Forests. We proposed a method that uses the predicted daily price changes on the training set to train another Random Forests to predict the covariance matrix $\Sigma$. By dividing the training set in two multiple times, we prevented overfitting on the training set.

Since the optimal prediction for the daily price changes is known in our model, as well as the covariance matrix $\Lambda$, we can compare the performance of the strategy when using Random Forests to predict the daily price changes and covariance matrix with an ‘optimal’ strategy that knows the expected daily price change and the actual covariance matrix $\Lambda$. We found that the profit of the strategy when using the Random Forest was $\frac{2}{3}$ of the profit when using the optimal strategy. We also saw roughly the same ratio for the corresponding Sharpe ratios. We thus concluded that the Random Forest strategy works well in this simple model.

We also introduced a more sophisticated model. Based on the ARCH model, we modified our model such that the covariance matrix $V_{t+1}$ in equation (80) is time dependent. With this model we where able to simulate markets more realistically, with periods of high or low volatility. In this model we found that our Random Forest strategy also performed well, with a ratio slightly less than $\frac{2}{3}$ in terms of both profit and Sharpe ratio compared to the optimal strategy.

After showing that our strategy performed well on the data generated from our models, we tested it on real market data. We looked at the Dutch AEX index and the German DAX index, from late 2007 to mid 2017. We used the closing prices of these indices and adjusted the weights in our portfolio once per day, where we used the exact same strategy as for the data from our models.

We found a Sharpe ratio of 0.69 over this 10 year period. Note that we did not account for transactions costs and ignored the risk-free return rate in the definition of the Sharpe ratio. This performance is quite good, considering that buying and holding one of the indices would lead to a Sharpe ratio of 0.01 in case of the AEX and 0.30 in case of the DAX. Using our strategy, we where thus able to outperform the individual indices.

Apart from the good Sharpe ratio, we also find that our strategy obeys our restrictions very well, meaning that the actual fractions of losses bigger than $L$ are only slightly above our imposed probability of $\alpha(L)$. We see that our strategy takes less risk (i.e. the weights $w_i$ are smaller) when the market is more volatile.
We also tested our strategy on more indices at once. Once we added the French CAC index and the European Euro Stoxx 50 index, we found a Sharpe ratio of about 0.74. While this higher than the Sharpe ratio we found when applying our strategy on only the AEX and DAX, it is not a significant improvement. Moreover, we found that the strategy did not obey our restrictions on the level of losses, since the empirical fraction of losses were often much bigger than the probabilities $\alpha(L)$ we imposed. An explanation for this is that the covariance matrix now has size $4 \times 4$, which is a lot harder to estimate than in the two dimensional case.

As mentioned before, the Sharpe ratio of 0.69 was calculated without accounting for transaction costs. We made a very rough estimate of the costs involved in trading our strategy, and concluded that these costs are very likely to have impact on the Sharpe ratio. Using our estimates, we found a Sharpe ratio of about 0.50 after transaction costs.

Finally, we conclude from this discussion that Random Forests can indeed be used in defining a profitable trading strategy. As said before, our strategy outperformed the two individual indices over a long period of time. This demonstrates that a Random Forest can indeed make useful predictions. The profitability of our strategy in a real life trading setting is less clear, since the costs involving in running the strategy seem quite high. One would have to do more sophisticated backtests in order to get a better idea of the actual costs. Our strategy could be improved in multiple ways. First of all, it might perform better on other pairs of products. Since we only tested our strategy on one pair, it is likely that it might perform better on some other pair. Secondly, we could improve our strategy by adding more input variables. One might think of including more variables directly related to the market, such as trading volume. However, input variables such as a sentiment score derived from Twitter messages might also improve the performance of our strategy.
References


[5] Sundén. *Trading based on classification and regression trees*