# Examination of AI-based ESG-scores as a valid source of alpha in the Swedish investing landscape

Marcus Haevaker

Master's thesis
2022:E38

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

CENTRUM SCIENTIARUM MATHEMATICARUM

# Acknowledgements

I would like to express my gratitude to my assistant supervisor Carl Hvarfner for providing continuous support during the writing of this thesis. Furthermore, I would like to deeply thank Oscar Dahlblom and Gustav Johnsson Henningsson at Sanctify Financial Technologies for their support, feedback and for providing data used in the thesis.

# Contents

## Abstract

Investing based on environmental, social, and governmental (ESG) criteria has grown rapidly in recent years. The trend has been driven by both an increased interest in sustainability, and the fact that ESG related corporate events have been shown to influence stock prices. In tandem with this development, a number of companies has pioneered methods to quantify a firm's ESG performance. One of these companies is Sanctify Financial Technologies.

In this thesis, numerous machine learning models are used to try and predict stock prices. ESG scores from Sanctify are then incorporated in some of the models, these models are then compared to identical models without access to these scores. The predicted stock prices are then inserted into a custom-made trading algorithm that creates daily investment portfolios that maximises the expected Sharpe ratio. The benchmarks used for evaluating the models are the Sharpe ratio, the Sortino ratio and gross returns.

A random forest regressor using various moving averages of the ESG scores ends up performing the best. With a Sharpe ratio of 1.185, a Sortino ratio 1.658 and gross returns of 60.5%, it outperforms the OMXS30GI index on all three benchmarks during 2020-2021.

In general, the performance of the models varies widely and indicates somewhat low predictive power. Excess returns are still achieved however, and the results indicate predictive performance for the Sancify scores on stock prices.

# 1

# Introduction

Since the stock market was created as an institution centuries ago the race to achieve larger and more predictable returns was on. As time passed various qualitative and quantitative methods were conceived and debates regarding the efficacy of these various techniques is not a new phenomena.

Regarding the quantitative side, econometrics, i.e. the application of statistical methods and tools on financial data was developed and the classical multivariate linear regression model has been popular for the past 100 years [36]. This simple predictor remained popular as economic datasets in the 20th century were mostly small and simple to analyse. This has changed in recent decades and tools such as machine learning has been applied in the field to handle this new surge of data.

In parallel to the developments in quantitative finance, societal expectations on corporations were changing. In 1970, famous economist Milton Friedman argued that "The Social Responsibility of Business Is to Increase Its Profits" [19]. In recent years, this view has largely fallen out of favour and corporations with both financial and non-financial goals are very much the standard today [39]. Why this happened is multi-faceted but clearly the motivation for managers and CEO's to pursue social and environmental aims includes an insight that responsible business could help to achieve financial goals [27]. For example, a company with ESG-conscious customers could lose through poor sustainability practises [27].

This rise in companies engaged in corporate social responsibility (CSR) has been matched by increased interest to invest in said companies and a company's perceived performance in Environmental, Social and Governance (ESG) metrics is now a driver for stock prices [3, 5]. A challenge when trying to use ESG performance in investment decisions is that ESG impact of a company is hard to measure in objective numerical terms [24]. A new set of fintech companies are trying to change this though, by leveraging advances in machine learning and the enormous amount of sentimental data available in our age, they aim to provide numerical measurements of company ESG performance.

## 1.1 Related work

### 1.1.1 Stock price prediction

Stock market prediction is the employment of various methods to try and predict the future prices of stocks. By doing this, the person doing the prediction hopes to achieve better risk-adjusted returns than the market can offer. Research during the 1960s-1970s [16, 17] led to the efficient market hypothesis (EHM). This hypothesis states that stock market prices are inherently unpredictable. According the EHM, stock price changes due to sudden events would quickly get priced in, and regular, long-term excess returns would not be possible. The hypothesis was not universally accepted however. Dreman and Berry [14] for example, found that stocks with low price to earning (P/E) ratios outperform those with high P/E ratios. Doubt regarding the EHM has lead to stock price prediction methods such as technical analysis. Technical analysts believe that historical stock price and stock trading volume data can provide information regarding future stock prices [6].

### 1.1.2 Stock price prediction using machine learning

As data is abundant and the potential gains enticing, a large amount of research on machine learning as a tool in stock prediction has been published in the latest decade. These machine learners generally use traditional technical analysis indicators, but apply them on larger data sets and with more powerful models. Selvin et al. [44] used various deep learning techniques to predict prices of stocks listed on the New York stock exchange. Convolutional Neural Networks (CNN) [31], Recurrent Neural Networks (RNN) [15] and Long Short Term Memory (LSTM) [22] were evaluated and CNNs was deemed the best model for stock prediction. Vijh et al. [51] compared the predictive power of Artificial Neural Networks (ANN) and Random forest regression on the close prices of the stocks of five large American companies. The ANN performed better in most cases but not uniformly so. Patel et al. [40] took another approach and compared four prediction models for classification, ANN, support vector machine (SVM) [7], Random Forest [9] and naive Bayes, in their respective ability to predict the direction of stock prices for Indian companies and indices. Chen and Liu [10] used scholar big data on ESG, and machine learning to create an investment strategy. Linear regression, Lasso regression [49], Support vector regression (SVR), Ridge regression [23], random forest regression and LSTM were all used in an ensemble to create the strategy. This strategy outperformed an identical approach which only relied on traditional financial indicators.

### 1.1.3 Stock price prediction using sentimental data

With the goal of finding data other than traditional technical indicators to incorporate in stock price prediction, plenty of research on using sentimental data from various sources to predict stock prices has been published. Sul et al. [48] found that the sentiment in tweets sent by users with less than 171 followers on Twitter, was significant in predicting the mentioned firm's stock returns for up to 20 trading days after the tweet was made. Li et al. [33] created a LSTM based classification predictor which incorporated both technical indicators based on historical stock data and news sentiments to predict the direction of stock prices for 12 Hong Kong listed stocks. The paper concluded that models using both technical indicators and news sentiment outperformed models that used only one of the data sources. News sentiment was in turn confirmed as viable data to use in stock market prediction. Sousa et al. [47] used the natural language processing (NLP) technique called BERT on 582 data-sources to create sentimental data. This sentiment was used to predict the direction of the Dow Jones index and was successful in 69% of cases. Kordonis et al. [28] processed tweets with NLP and sentiment analysis. Thereafter, they used both naive Bayes and SVM to predict the sentiment of tweets. The conclusions of their work included that the Twitter sentiment could affect stock prices. Therefore, to some extent, the stock market can be predicted.

## 1.2 Sanctify Financial Technologies

This thesis has been produced in collaboration with Sanctify Financial Technologies, a Lund based company engaged in providing numerical ESG-performance scores for companies worldwide. The company uses an API to receive information from a large number of sources. NLP is then applied to the information to convert it into sentimental data which in turn is used to create numerical scores. The scores are intended to be used by fund managers of various types to incorporate ESG in their investment decisions.

## 1.3 Objective

The objective of the thesis is to evaluate whether Sanctify's AI-based ESG-scores are a valid source of *alpha* (positive risk adjusted returns) in the Swedish investing landscape. Several predictors which incorporate this data will be created and compared to both a reference index and identical predictors which lack access to Sanctify's data. The viability of the predictors will be examined through three benchmarking metrics. The Sharpe ratio, the Sortino ratio and gross returns.

## 1.4 Scope

This objective, at its extreme, could encompass every single stock that Sanctify keeps scores on, and every single method to incorporate these scores in a stock-price predictor. As this is unfeasible, a more narrowly defined scope has been decided upon. The first delimitation is regarding the investment universe, only Swedish stocks included in the OMXS30 index will be traded by the predictors and conversely, the reference index for comparison will be the OMXS30GI index, a variant of OMXS30 adjusted for dividends.

As the data that Sanctify uses to derive its ESG-scores are scarcely available before 2016, we will only consider the period from 2016-2021. 2020-2021 has been chosen as the test period and the 4 previous years will be used to train the predictors.

Another important delimitation is regarding investment horizon. The predictors will be allowed to recompose their investment portfolios on a daily basis. With the exception of some high-frequency-traders, this is not a realistic strategy for fund managers due to both administrative hurdles and transaction fees. However, these hurdles and fees are not present in an academic environment, and the short investment horizon will still allow for examination of the predictive power of Sanctify generated ESG-scores. Finally, the predictors have been prohibited from utilizing short positions in any stock, a concept explained in section 2.2.

When selecting attributes to use for stock price prediction, one has many potential data sources to utilise. Factors that influence the pricing mechanisms of stocks range from broad macroeconomic data, to firm-specific data disclosed in financial statements, to posts on social media [48]. Attribute selection is therefore a very complex and time consuming task when attempting to predict stock prices. As a result, the scope of the thesis has been limited by largely adopting the same attributes used by Vijh et. al. [51] in their 2019 paper on stock prediction. This is further explained in section 4.3.

# 2 Background

## 2.1 Long positions

When an investor believes that the price of a certain stock will increase, the investor can buy the stock, or in financial terms, take a long position in the stock. When holding such a position, the returns for the investor corresponds to the change in the stock price. A surge from a price at purchase from \$100 to \$150 at sales means a profit of \$50 for the investor. Normally, an investor can also take leveraged long positions by borrowing money to invest more than 100% of its funds, or by purchasing derivatives with innate leverage, such as call options. In this thesis however, leverage will not be utilised.

## 2.2 Short positions

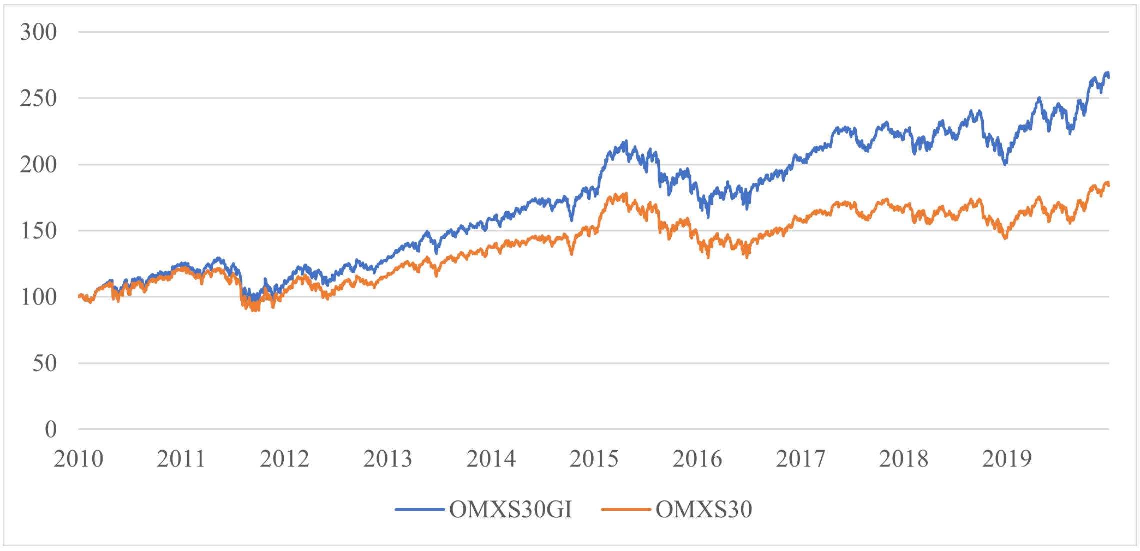
If the investor instead believes that the price of the stock will fall, it can instead take a so-called short position in the stock. A short position is essentially the opposite of a long one, and the investor makes a positive return if the price of the shorted stock falls. The investor borrows a stock from an existing owner (lender) and instantly sells it for the current market price. At a later time the investor buys back the stock from the market and returns it to the lender. The lender receives interest for the loan and the investor receives a negative exposure towards the stock price for the duration of the loan. If the price of the stock in question falls by \$50 during the duration of the loan, the investor takes home a profit of \$50 minus interest. If the price instead surges by \$50, the investor instead loses \$50 plus interest. The concept of short positions is important for modern financial markets. Studies have shown that access to short selling improves market efficiency through increased liquidity and price discovery [4]. In this thesis however, short selling has been prohibited for a variety of reasons. The main two being:

1. Modelling purposes.
2. A short-selling ban was requested by thesis collaborator Sanctify Financial Technologies as their clients mainly composes of long-only investors.

## 2.3 Stock indices

Two stock indices will be mentioned in this thesis. The OMXS30 and the OMXS30GI. The OMXS30 is a famous index frequently quoted by various analysts as a measurement on how the Swedish stock market is performing. The index is composed of the 30 stocks listed on the Swedish stock exchange with the highest turnover measured in Swedish Krona (SEK). The index is also reconstituted and rebalanced every six months, i.e. the 30 stocks and their respective weights are chosen every six months. [37]. Only the stocks present in the OMXS30 at the time of evaluation can be traded by the predictors in this thesis. If a stock is added to the index 2020-01-01 and removed 2020-07-01, it can only be traded between those two dates. The reason for this is so that the predictors can be compared to the reference index on a fair basis. If the members of the OMXS30 in 2022 could be traded for the entire test period, we would essentially only trade "winners", i.e. stocks that have either remained in, or advanced to the OMXS30, while not trading "losers", stocks that previously have been part in the index but was excluded due to low turnover. It should be obvious that winners on average would provide larger returns so only trading members of the index at the time of evaluation safeguards against an introduction of bias. The OMXS30GI is a total return variant of the OMXS30. This means that cash dividends from the stock included in the index are reinvested in the index, creating larger index returns.

*Figure 2.1: Index performance for OMXS30 and OMXS30GI 2010-2019. The inclusion of dividends in OMXS30GI leads to higher returns*



As the predictors will trade *Adjusted Close prices* from Yahoo Finance, explained in section 2.4, which also includes reinvested dividends, OMXS30GI will be used for comparison with the predictors.

## 2.4 Adjusted close prices

All predictors in this thesis use stock price data from Yahoo Finance. In calculating portfolio returns $R_t$ from one market day to the next, the following formula is used:

$$R_t = \frac{\sum_{i \in S}(w_{i,t} p_{i,t})}{\sum_{i \in S}(w_{i,t-1} p_{i,t-1})} - 1 \tag{2.1}$$

Where $S$ is the set of stocks in the investment universe, $w_{i,t}$ is the portfolio weight of stock $i$ at time $t$, and $p_{i,t}$ is the price of stock $i$ at time $t$. For this formula to provide an accurate return measurement, stock splits and dividends will have to be accounted for. This is achieved by using the Yahoo Finance provided Adjusted Close prices, which adjust for these events by using multipliers for historical stock prices [53]. Section 2.4.2 explains this in detail:

### 2.4.1 Ex-dates

To understand the adjusted close, one must also understand the function of the ex-date. The ex-date is the first trading day on which ownership of a stock does not entitle the owner to the next dividend. Let us consider a \$1 dividend at July 14 with the corresponding ex-date is July 11. In this scenario, all owners of the stock at the end of trading at the trading day before July 11, is entitled to \$1 on July 14.

### 2.4.2 Fictional example

In the trading week from February 10 to February 14, a stock has the following splits and dividends

- A $2:1$ split on February 11, represented by a 0.5 multiplier.
- A \$1 dividend on with an ex-date at February 13, represented by a $(1-\frac{1}{20}) = 0.95$ multiplier.

*Table 2.1: Fictional example of the calculation of Adjusted close prices*

| Date | Close price | Adjusted close price |
|---|---|---|
| 10/2 | \$43.00 | $43.00 * 0.5 * 0.95 =$ \$20.425 |
| 11/2 | \$21.10 | $21.10 * 0.95 =$ \$20.045 |
| 12/2 | \$21.20 | $21.20 * 0.95 =$ \$20.14 |
| 13/2 | \$20.00 | \$20.00 |
| 14/2 | \$20.40 | \$20.40 |

## 2.5 Sanctify ESG-scores

The Sanctify ESG database provides ESG insights for tens of thousands of companies on numerous exchanges worldwide. The insights are presented in the form of numerical scores based on NLP analysed news media. The scores are derived from a very large amount of publications of various sorts and are updated daily. Every piece of news media deemed relevant for the sustainability profile of a company is categorised according to the 26 sustainability categories defined by the non-profit organisation Sustainability Accounting Standards Board (SASB). Sentiment analysis is then used to score the media from [-1,1], where positive scores are assigned for positive news and negative scores for negative news. The score is then added to either the E-score (environmental), the S-score (social) or the G-score (governmental) of the company depending on the particular SASB category. The total ESG-score is calculated as a simple average of the E, S and G-score. There are two main variants of the scores, absolute and relative. Absolute scores are not comparable between companies and instead reflects the current media sentiment for a particular company. Scores are calculated as a sum of newly added news, and old news multiplied by a decay factor. The decay factors are available in three variations: short, medium and long. Relative scores instead are comparable between companies. This is achieved by incorporating historical ESG performance in the media for long periods combined with current trends.

In this thesis, only the short-term absolute ESG-scores will be utilised.

## 2.6 Predictor benchmarking and evaluation

### 2.6.1 Sharpe ratio

The Sharpe ratio (SR) [45] is commonly used to measure the risk-adjusted returns of investment portfolios [2] and is therefore a suitable benchmark for our predictors. The SR is defined by the following formula:

$$\text{SR} = \frac{r_p - r_f}{\sigma_p} \tag{2.2}$$

where $r_p$ is the average yearly return, $r_f$ the risk-free rate and $\sigma_p$ the yearly volatility The formula applies only for returns and volatility calculated on an annual basis. Traditionally, the ratio is multiplied by the square root of the number of periods in a year for shorter periods, such as $\sqrt{12}$ for monthly returns and volatility. Lo [34] has shown that this approach contains faulty assumptions and leads to incorrect estimations of annual SR. In this thesis, returns are calculated on a daily basis and the traditional annualising method will be used, even with its flaws, for simplicity.

### 2.6.2 Sortino ratio

The Sortino ratio [46] is similar to the Sharpe ratio but varies in one crucial way: it only accounts for downside risk. The rationale behind the ratio is that an investor would not mind if the return of a portfolio displays large variance only on the positive side. A portfolio that yields 1% in January, 100% in February and 30% in March would satisfy almost any investor even though it displays large variance. What investors fear is instead volatility on the downside. The formula for the Sortino Ratio is:

$$\text{Sortino Ratio} = \frac{r_p - r_f}{\sigma_{p-negative}} \tag{2.3}$$

where $r_p$ is the portfolio return, $r_f$ the risk-free return and:

$$\sigma_{p-negative} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\min(0, r_n)^2)} \tag{2.4}$$

where $N$ is the number of periods and $r_n$ the return at period $n$

### 2.6.3 Gross return

Gross return, while a sub-optimal measurement from a scientific point of view, is still used for marketing of financial products due to its simplicity. The gross return is defined as:

$$\text{Gross return(t)} = \frac{y_t - y_0}{y_0} \tag{2.5}$$

where $y_t$ is the value of the portfolio at time $t$ and $y_0$ is the initial portfolio value.

### 2.6.4 Technical analysis tools

#### 2.6.4.1 Simple moving average (SMA)

The simple moving average (SMA) is a popular tool in technical analysis. The SMA is defined as:

$$\text{SMA\_n}_t = \frac{x_{t-n} + x_{t-(n-1)} + ... + x_t}{n} \tag{2.6}$$

where SMA_n$_t$ is the n-day SMA at time $t$ and $x_t$ is the underlying (such as a stock price) at time $t$.

#### 2.6.4.2 Momentum (MOM)

Another common tool in technical analysis is the momentum (MOM). The MOM is defined as:

$$\text{MOM\_n}_t = x_t - x_{t-n} \tag{2.7}$$

where MOM_n$_t$ is the n-day MOM at time $t$ and $x_t$ is the underlying (such as a stock price) at time $t$.

### 2.6.5 Mean squared error (MSE)

When performing hyperparameter optimisation, the mean squared error (MSE) will be used as the measurement to minimise. The MSE is defined by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.8}$$

where $y_i$ are the true values, $\hat{y}_i$ are the predicted values and $n$ is the number of predictions.

# 3

# Mathematical and machine learning theory

## 3.1 Introduction to machine learning

Distinguished machine learning experts Tom M. Mitchell and Michael I. Jordan described machine learning with the following words in a 2015 issue of *Science*. "Machine learning addresses the question of how to build computers that improve automatically through experience. It is one of today's most rapidly growing technical fields, lying at the intersection of computer science and statistics, and at the core of artificial intelligence and data science" [26]. The experts made this observation in 2015 and it is fair to say that the field has been getting ever more attention since [41].

### 3.1.1 Supervised learning

The focus of this thesis are models that all fit under the umbrella-term called supervised learning. In supervised learning, we are given labeled data, i.e. pairs $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)$ where $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in X, \quad y_1, y_2, ..., y_n \in Y$, and the goal is to learn the relationship between $X$ and $Y$ [13]. Each observation $\mathbf{x}_i$ is a vector of one or more attributes, or inputs. These attributes are, in an ideal case, correlated with the corresponding label, or output $y_i$. For example, one could imagine $y_i$ as the power consumption in Lund on day $i$, and $\mathbf{x}_i$ being a $24 \times 1$ vector of projected temperature in Lund for each hour of day $i$. In its essence, supervised learning is a tool to address a fundamental prediction problem. Let $\hat{y}(\mathbf{x})$ be a predictor of an 1-dimensional output $y$, given an $n$-dimensional input vector $\mathbf{x} = (x_1, x_2, ..., x_n)$. Machine learning can, in the supervised learning case, be viewed as the study and construction of an input-output map of the form

$$y = F(\mathbf{x}) \tag{3.1}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$ [13]. $y$ can be either continuous as for the temperature in Lund, or discrete, as in what species of tree a certain photographed leaf belongs to. This division between discrete and continuous gives rise to another dividing line in machine learning, classification and regression.

#### 3.1.1.1 Classification

Classification is the process of producing a model that is capable of distinguishing between a fixed set of classes. In the terminology from equation (3.1), $y \in (1, 2, ..., K)$ where $1, 2, ..., K$ are a set of $K$ discrete classes, for example the species of trees mentioned above. Classification is not inside the scope of this thesis but an understanding of the concept is required for several of the model derivations in this chapter.

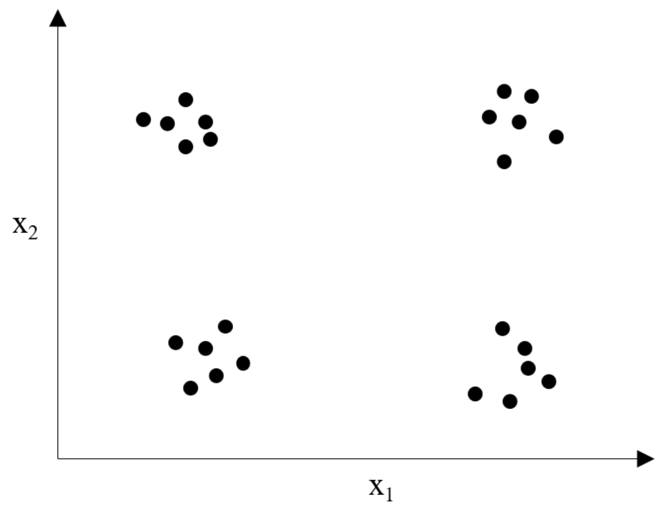
#### 3.1.1.2 Regression

In regression, one tries to produce a model with the capacity to produce real-valued, continuous outputs as a response to one or more inputs. Using equation (3.1) as a foundation, we have a regression problem if $y \in \mathbb{R}^m$. For example, $y$ could be the previously mentioned temperature in Lund on a certain day.

### 3.1.2 Unsupervised learning

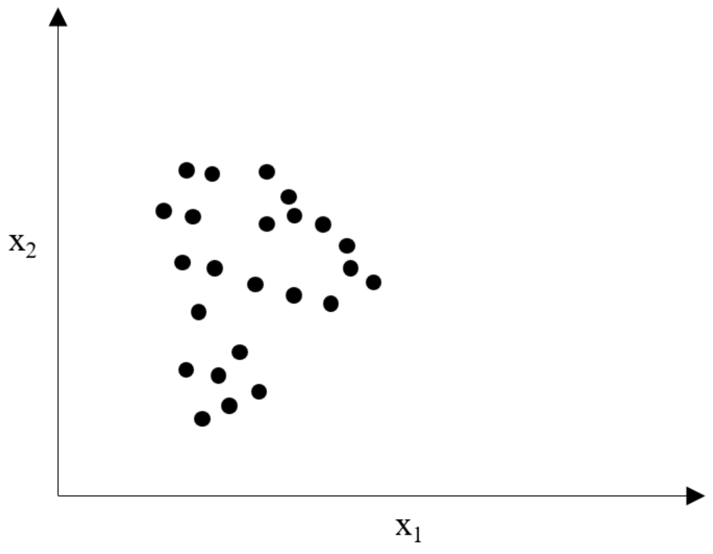
In supervised learning, the dataset, as mentioned above, consists of pairs of labeled data and its associated features. In unsupervised learning, the data is not labeled with classes and the machine learning task is instead to extract information from this ambiguous set of data. One of the more popular methods in unsupervised learning is looking for groups of similar examples, called clusters, in the data [29]. By detecting such clusters, one can then use these for predicting the values of unknown attributes.

*Figure 3.1: Clear clustering visible in 2-dimensional data*



In the two-attribute example seen in Figure 3.1, one can clearly see the four clusters and there is certainly no risk of overlap between the clusters. In other cases, the data is more ambiguously clustered, as can be seen in Figure 3.2. In this scenario, the human eye will find it hard to determine both the number of clusters, and where one should place the border between different clusters.

Figure 3.2: *Ambiguous clustering in 2-dimensional data*



This represents a problem, as each data-point must belong to one and only one cluster. Of course, one could let every data-point be its own cluster but that would turn the cluster analysis meaningless. To solve the "number of clusters" issue, implementations of cluster analysis are often constructed in such a way that the user supply the number of clusters via an input parameter. It is also possible for a machine learning algorithm to determine the number on its own [29]. Regarding the issue of which cluster a data-point should belong to, the simplest approach is to use so called *centroids* and Euclidian distance. Centroids can be though of as dots in the middle of an n-th dimensional cluster. The centroid is identified by the averages of the attributes of the members of the cluster. If the attributes are non-numerical this can be resolved by converting them to numerals. With centroids defined in this way, new data-points can simply be identified with the closest (in Euclidian distance) centroid and in turn, the cluster associated with the centroid. The Euclidian distance from centroid $k$, $d_k$ is measured with the following equation:

$$d_k(\mathbf{x}, \mathbf{x}_{cent}) = \sqrt{\sum_{i=1}^{n}(x_i - x_{cent,i})^2} \tag{3.2}$$

where $\mathbf{x}$ are the attributes of the data-point, $\mathbf{x}_{cent}$ are the attributes of centroid $k$ and $n$ the number of attributes. When performing clustering analysis in this sense, the relative sizes and the chosen unit of the attributes influences the result. To remove this influence, normalization of all attributes to the unit interval $x_i \in [0, 1]$ can be performed.

## 3.2 Classical Autoregressive predictors

Autoregressive predictors are predictors that model processes where the value of the process at time $t$ is dependent on the value of the process at some time $t - k$ where $k$ is a positive integer. These predictors have generally been around for a long time, for example, the central Yule-Walker equations were formulated in the 1920s and

30s [50, 52]. Further developments were regularly made during the last century. The underlying autoregressive process and extensions to it are introduced in this section. Furthermore, one method to model and predict the process, the Kalman filter, is introduced [25].

### 3.2.1 Autoregressive (AR) processes

A process $y_t$ is an autoregressive (AR) process of order $p$ (an AR($p$)-process) if [25]:

$$A(z)y_t = y_t + a_1 y_{t-1} + ... + a_p y_{t-p} = e_t \tag{3.3}$$

where $A(z)$ is a monic polynomial of order p:

$$A(z) = 1 + a_1 z^{-1} + ... + a_p z^{-p} \tag{3.4}$$

where $a_p \neq 0$ and $e_t$ is a zero mean white noise process uncorrelated with previous measurements of $y$ ($y_{t-k}$ where $k > 0$). $z^{-1}$ is the unit delay operator, defined as $z^{-1}y_t = y_{t-1}$. As $E[e_t] = 0$ and in turn $E[A(z)y_t] = 0$, the process is stationary if all zeros of $A(z)$ are within the unit circle.

### 3.2.2 Autoregressive moving average (ARMA) processes

Building on the previous section, $y_t$ is an autoregressive moving average (ARMA) processes of order $p, q$ (an ARMA($p, q$)-process) if [25]:

$$A(z)y_t = C(z)e_t \tag{3.5}$$

where $A(z)$ is defined by equation (3.4) and:

$$C(z) = 1 + c_1 z^{-1} + ... + c_q z^{-q} \tag{3.6}$$

### 3.2.3 Trends and ARIMA processes

If the process $y_t$ has clear trends, either in mean or variance, it is not possible to model it as a stationary process. If the trend is deterministic, it can be modelled as a deterministic function which later can be subtracted from the process. If the trend is stochastic, the process could instead be modelled as an autoregressive integrated moving average (ARIMA) process [25]. The process $y_t$ is an ARIMA process of order $p, d, q$ (an ARIMA($p, d, q$) process) if:

$$A(z)(1 - z^{-1})^d y_t = C(z)e_t \tag{3.7}$$

where $A(z)$ and $C(z)$ is given by equations (3.4) and (3.6)

### 3.2.4 Exogenous inputs

If additional time series with information correlated to the process $y_t$ is available, it is sensible to expand the model to incorporate this. If only one input variable is available, an ARMA model can be expanded in the following way, creating an ARMAX process:

$$A(z)y_t = B(z)x_t + C(z)e_t \tag{3.8}$$

where $A(z)$ and $C(z)$ is given by equations (3.4) and (3.6), $x_t$ is the value of the input variable at time $t$ and:

$$B(z) = b_0 + b_1 z^{-1} + ... + b_s z^{-s} \tag{3.9}$$

The ARMAX process can also be modelled to incorporate more than one input variable:

$$A(z)y_t = B_1(z)x_{1t} + B_2(z)x_{2t} + ... + B_n(z)x_{nt} + C(z)e_t \tag{3.10}$$

where $x_{nt}$ is the value of the $n$:th input variable at time $t$. The aforementioned ARIMA process can also be expanded to incorporate input variables in the same way, resulting in an ARIMAX process.

## 3.3 Parameter estimation and prediction

When modelling a process such as an AR, ARMA, ARIMA or ARIMAX, one estimates the parameters of the polynomials ($A$, $C$, $B$) with a method such as least squares, maximum likelihood, or the prediction error method [25]. This will result in a model that can be used for prediction by inserting the exogenous inputs $(x_t, ..., x_{t-i})$, the previously observed errors $(e_{t-1}, ..., e_{t-1-k})$, and the previously observed outputs $(y_{t-1}, ..., y_{t-1-j})$. Here $i$, $k$ and $j$ are positive integers that corresponds to the order of the chosen model.

### 3.3.1 Dynamic systems

When estimating parameters as explained in section 3.3, the predictor will assume that the model polynomials are constant. In turn, it will yield non-optimal results if the characteristics of the system varies with time. If one for example tries to model the price of the Dow Jones stock market index, the polynomials of the process will probably differ in 1970 compared to 2020. This issue of non-constant polynomials can be tackled by different methods but here we will focus on a famous one, the Kalman filter.

#### 3.3.1.1 Linear state space representation

Consider a linear state space representation of the form:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{e}_t \tag{3.11}$$

$$y_t = \mathbf{C}_{t|t-1}\mathbf{x}_{t|t-1} + w_t \tag{3.12}$$

Where $y_t$ is a measurable output variable and $\mathbf{x_t}$ is an $n$:dimensional non-observable internal state vector. $\mathbf{e}_t$ is the white noise error of the process and $w_t$ is the white noise measurement error. If we assume that $y_t$ follows an ARMAX$(2, 2, 2)$ process with unknown parameters, one could express the process as a state space representation in the following way, we have:

$$A(z)y_t = B(z)x_t + C(z)e_t \tag{3.13}$$

With $A(z)$, $B(z)$ and $C(z)$ defined as in equations (3.4), (3.9) and (3.6). This can be written as:

$$y_t = B(z)x_t + C(z)e_t - a_1 y_{t-1} - a_2 y_{t-2} \tag{3.14}$$

So: $y_t = \mathbf{C}_{t|t-1}\mathbf{x}_{t|t-1} + w_t$ with $\mathbf{C}_{t|t-1} = [x_t, x_{t-1}, x_{t-2}, e_{-1}, e_{-2}, -y_{-1}, -y_{-2}]$ and $\mathbf{x_t} = [b_0, b_1, b_2, c_1, c_2, a_1, a_2]$

#### 3.3.1.2 The Kalman filter

Consider the state space model of equations (3.11) and (3.12). We now introduce

$$\mathbf{Y}_t = [y_1, y_2, ..., y_t]^T \tag{3.15}$$

Which is a vector containing all outputs up to and including time t. The optimal linear predictor of the hidden state vector $\mathbf{x}$ at time $t$ can then be defined as:

$$\hat{\mathbf{x}}_{t|t} = E[\mathbf{x}_t \mid \mathbf{Y}_t] \tag{3.16}$$

If we split $\mathbf{Y}_t$ into its current $y_t$ and past $\mathbf{Y}_{t-1}$ measurements we get:

$$\hat{\mathbf{x}}_{t|t} = E[\mathbf{x}_t \mid y_t, \mathbf{Y}_{t-1}] \tag{3.17}$$

Equations (3.18) and (3.19) holds for a vector $\mathbf{z}$ that is correlated with both $\mathbf{x}$ and $\mathbf{y}$ [25] :

$$E\{\mathbf{x} \mid \mathbf{y}, \mathbf{z}\} = E\{\mathbf{x} \mid \mathbf{z}\} + C\{\mathbf{x}, \mathbf{y} \mid \mathbf{z}\}V\{\mathbf{y} \mid \mathbf{z}\}^{-1}(\mathbf{y} - E\{\mathbf{y} \mid \mathbf{z}\}) \tag{3.18}$$

$$V\{\mathbf{x} \mid \mathbf{y}, \mathbf{z}\} = V\{\mathbf{x} \mid \mathbf{z}\} - C\{\mathbf{x}, \mathbf{y} \mid \mathbf{z}\}V\{\mathbf{y} \mid \mathbf{z}\}^{-1}C\{\mathbf{x}, \mathbf{y} \mid \mathbf{z}\}^T \tag{3.19}$$

Using this on the expectation in equation (3.17) yields:

$$E[\mathbf{x}_t \mid y_t, \mathbf{Y}_{t-1}] = \hat{\mathbf{x}}_{t|t-1} + C[\mathbf{x}_t, y_t \mid \mathbf{Y}_{t-1}]V[y_t \mid \mathbf{Y}_{t-1}]^{-1}(y_t - \hat{y}_{t|t-1}) \tag{3.20}$$

Where $\hat{y}_{t|t-1}$ is the prediction of $\mathbf{y}_t$ formed at time $t-1$. This imply that $y_t - \hat{y}_{t|t-1}$ is the 1-step prediction error. Inserting (3.20) into (3.17) now yields:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + C[\mathbf{x}_t, y_t \mid \mathbf{Y}_{t-1}]V[y_t \mid \mathbf{Y}_{t-1}]^{-1}(y_t - \hat{y}_{t|t-1}) \tag{3.21}$$

Where $C[\mathbf{x}_t, y_t \mid \mathbf{Y}_{t-1}]V[y_t \mid \mathbf{Y}_{t-1}]^{-1}$ is referred to as the Kalman gain $\mathbf{K_t}$

We can therefore conclude that the optimal reconstruction of the state vector $\mathbf{x}_t$ is given by the following equation:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K_t}(y_t - \hat{y}_{t|t-1}) \tag{3.22}$$

If we assume that the prediction error $y_t - \hat{y}_{t|t-1}$ is random, the best linear prediction of $\mathbf{x}_{t+1|t}$ is simply given by:

$$\hat{\mathbf{x}}_{t+1|t} = \hat{\mathbf{x}}_{t|t} \tag{3.23}$$

Using equations (3.23) and (3.12) and the fact that $w_t$ is white-noise, the best linear prediction of $y_{t+1|t}$ is be given by:

$$\hat{y}_{t+1|t} = \mathbf{C}_{t+1|t}\hat{\mathbf{x}}_{t|t} \tag{3.24}$$

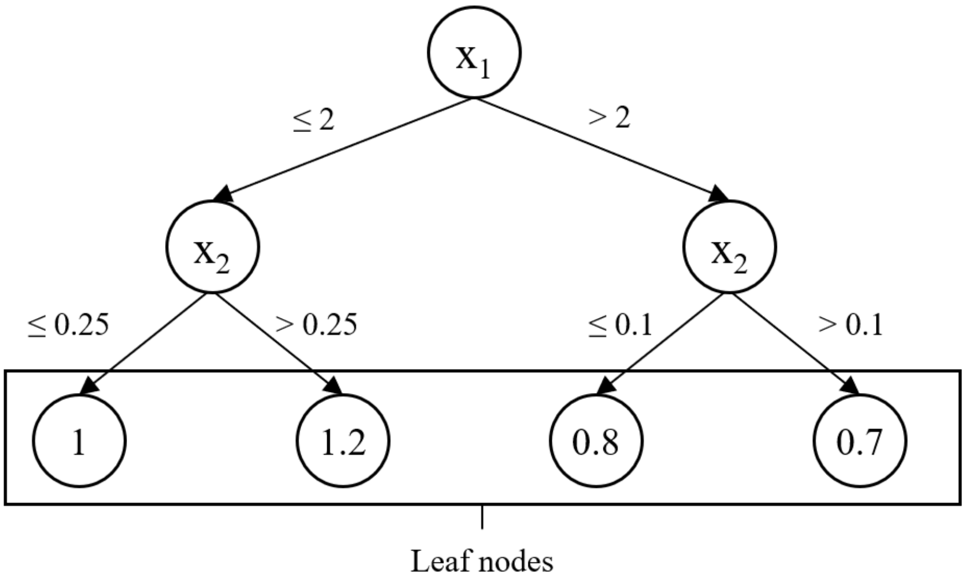
The Kalman filter is implemented by recursively predicting $\hat{y}_{t+1|t}$ and updating $\hat{\mathbf{x}}_{t|t}$ according to equation (3.22)

## 3.4 Decision trees and Random forests

### 3.4.1 Decision trees

Decision trees are obtained by recursively partitioning the available data based on its attributes and in turn creating a predictor [35]. The recursive partitioning can be graphically represented as a tree which gives the algorithm its name. A tree consists of nodes, of which there are two types of: internal nodes and leaf nodes. Internal nodes correspond to an attribute test where the value of the attribute determines which branch of the tree to follow. Leaves on the other hand are end nodes and correspond to an output. If the output is a discrete class, we have a classification tree, if the output is a continuous numeric value, we have a regression tree. A simple example of a regression tree can be seen in Figure 3.3

*Figure 3.3: Simple example of 2-attribute regression tree*



Decision trees are grown by using an algorithm for the partitioning, there are various such to choose from, but we will focus on the CART-algorithm used in the Random Forest ensemble. CART recursively partitions data into a tree by utilizing the Gini index for classification and sum of squared deviations from the mean (SS) for regression [35, 20]. We refer to [20] for further information about the Gini index.

The SS is defined as:

$$\text{SS} = \sum_{D} (y_i - \bar{y})^2 \tag{3.25}$$

Where $D$ is a set of $(\mathbf{x}_i, y_i)$ – tuples and $\bar{y}$ is the mean of all $y \in D$ . CART uses the SS by evaluating binary splits for each attribute and choosing the split with the lowest SS-value. A binary split is a partition that divides the dataset into two sets. For a continuous attribute this means splitting $D$ into $x_i > z$ and $x_i \leq z$ where $x_i$ is the value of the attribute $i$ and $z$ is the dividing value used for the binary split. For a discrete attribute one instead defines a split by creating a subset of all present values of $x_i$, this subset is called $S_i$. One then splits into $x_i \in S_i$ and $x_i \notin S_i$.

When evaluating a possible binary split, the combined SS of the two new sets are computed with the following formula:

$$\text{SS} = \sum_{D_1}(y_i - \bar{y_1})^2 + \sum_{D_2}(y_i - \bar{y_2})^2 \tag{3.26}$$

Where $D_1$ and $D_2$ are the sets created by the binary split, i.e. $D = D_1 \cup D_2$, $\bar{y_1}$ is the mean of all $y \in D_1$ and $\bar{y_2}$ is the mean of all $y \in D_2$. The value used for the binary split $z$ for the attribute $x$ is then chosen as the split that results in the lowest SS-value. This process is repeated for all attributes $X$ and the attributes with the lowest possible SS-value is selected as the attribute for the split. This attribute and its corresponding split-value $z$ together constitute the criterion for the partition.

By implementing this algorithm on the whole set $D$, we label the root node with this criterion and grow two new branches from the node. These two sets are then split with the same method. It is worth mentioning that attributes are allowed to be split multiple times. This process is repeated recursively for all created nodes until at least one of the following is true:

1. Every $y \in D$ are equal.
2. No more attributes to split are available. This can only occur if all attributes are discrete.
3. The best possible split for $D$, which results in $D_1$ and $D_2$, leaves either $D_1$ or $D_2$ empty.

If 1 is true, the node is converted into a leaf node with the value/class $y$. If 2 and/or 3 is true, the node is also converted into a leaf node, but the value of the leaf becomes the average of all $y \in D$ (regression) or the majority class of all $y \in D$ (classification).

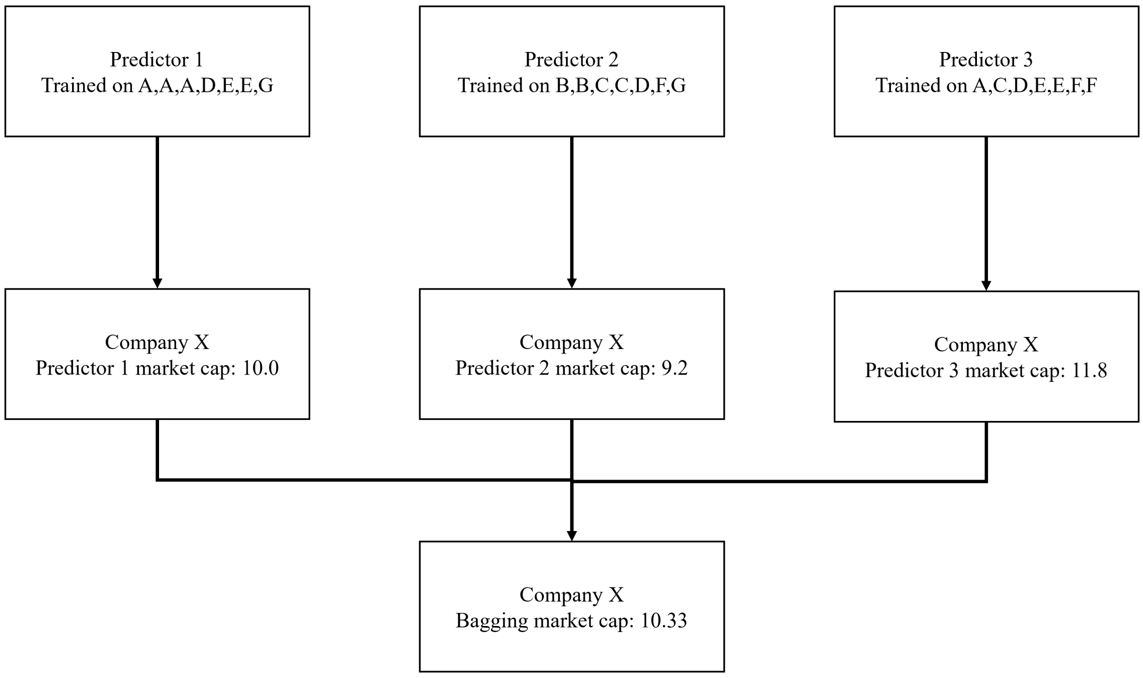
### 3.4.2 Bagging

Bagging, or bootstrap aggregation, [8] is an ensemble method for improving accuracy of various predictors. (An ensemble method is to employ more than one predictor and use their combined predictions to improve performance). For a predictor with $N$ tuples of training data $(\mathbf{x}_i, y_i)$ where $y$ is the target label or number, and $\mathbf{x}$ is an array of input attributes, $N$ tuples of data are randomly chosen with replacement (after each individual selection). These $N$ tuples together constitute a set called $S_1$. This implies that each $(\mathbf{x}_i, y_i)$ tuple can appear between 0 and $N$ number of times in $S_1$. Several of these sets $S_1, S_2, S_3$... are made and predictors are trained on these sets individually, generating numerous predictors trained on unique sequences of training data. For classification, a majority vote system where the individual predictors vote on classes are used for prediction while standard averaging is used for regression. A fictional example of applying bagging with 7 data points of test data and three predictors for regression is presented in table 3.1 and Figure 3.4:

*Table 3.1: Fictional data on seven companies*

| Company name | Profit ($x_1$) | Revenue growth, % ($x_2$) | Market cap ($y$) |
|---|---|---|---|
| A | 5 | 3 | 50 |
| B | 3 | 20 | 65 |
| C | 6 | 3 | 63 |
| D | 4 | 4 | 50 |
| E | 4 | -10 | 20 |
| F | 10 | 0 | 90 |
| G | 2 | 1 | 28 |

*Figure 3.4: Three predictor fictional example of bagging. The target to be predicted is the market cap of Company X. The training data is from table 3.1*



### 3.4.3 Random forests

Random Forests [9], is an ensemble method that combines bagging, decisions trees and random attribute selection. $K$ sets of training data $S_i$ are generated by using bagging on the original test data $S$. Each set $S_i$ is used to train a decision tree predictor and together these trees form a so called "forest" [20]. When calculating the optimal split for a node, a subset of size $j$ attributes are selected from the full set of all attributes of size $n$. This selection is done without replacement so if we set $j = n$, we consider all attributes at each split.

## 3.5 Support vector machines (SVM)

The Support Vector Machine is a machine learning algorithm originally developed for classification. In essence, it uses hyperplanes to divide data-sets into different classes. Through extensions, the algorithm can also be used for regression [18].

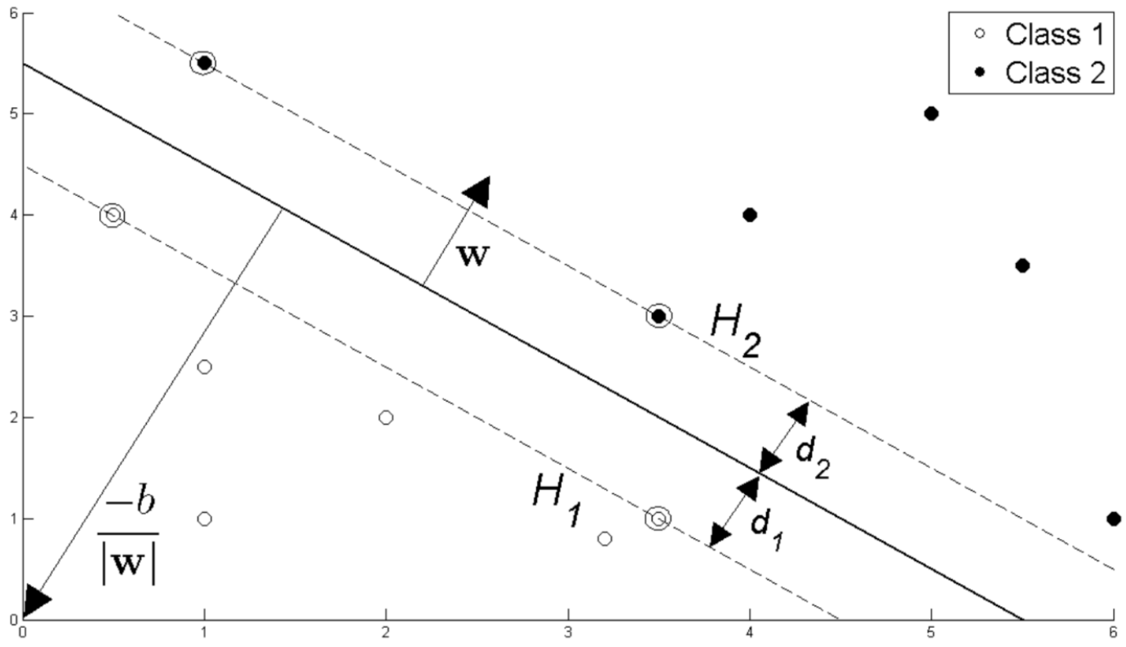
### 3.5.1 Linear SVM

In the most basic case, training data for classification consist of N pairs of data on the form: $(\mathbf{x}_i, y_i)$ where $i = 1, 2 \ldots, N$, $y \in -1, 1$ and all $\mathbf{x}$ are of dimension $D$ To further simplify, we assume that the data is linearly separable. With linearly separable we mean that we can separate the two classes by drawing a hyperplane on the $x_1, x_2, x_3, ..., x_D$-space. A hyperplane is simply a plane generalised to any dimension so for $D = 2$ the hyperplane is simply a line on the $x_1, x_2$ space. This hyperplane can be described by:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{3.27}$$

where $\mathbf{w}$ is orthogonal to the hyperplane and $\frac{|b|}{||\mathbf{w}||}$ is the perpendicular distance from the hyperplane to the origin. The examples located the shortest distance from the separating hyperplane are called support vectors and the Support Vector Machine's task is to orientate the hyperplane in a way that maximises the distance between the support vectors and the hyperplane.

*Figure 3.5: Hyperplane that divides two linearly separable classes. $d_1 = d_2$ is the margin while $H_1$ and $H_2$ are support vectors [18]*



When we implement the SVM, we therefore strive to select $\mathbf{w}$ and $b$ so that the training data fulfil the following inequalities:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \text{ for } y_i = 1 \tag{3.28}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq 1 \text{ for } y_i = -1 \tag{3.29}$$

These two inequalities (3.28) and (3.29) can be combined into:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \; \forall i \tag{3.30}$$

We call the distance between the support vectors and the hyperplane, the *margin*. The SVM is defined in such a way that the margin between the hyperplane and the first support vector, is equal to the distance between the hyperplane and the second support vector. Now, in order to maximise distance between the support vectors and the hyperplane, we need to maximise this margin. Linear algebra shows that the margin is $\frac{1}{||\mathbf{w}||}$ so we minimize the l2-norm $||\mathbf{w}||$ which is equal to minimizing $\frac{1}{2}||\mathbf{w}||^2$. We get the following optimisation problem:

$$\min \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \tag{3.31}$$

As we have $N$ constrains, we introduce an $n$-dimensional vector of Lagrange multipliers $\boldsymbol{\alpha}$, where $\alpha_i \geq 0\ \forall i$, to incorporate the constrains in the minimization:

$$\begin{aligned} L_p &= \frac{1}{2}||\mathbf{w}||^2 - \boldsymbol{\alpha}(y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\forall_i) \\ &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i(y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1) \\ &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{N} \alpha_i \end{aligned} \tag{3.32}$$

Differentiating $L_p$ with regards to $\mathbf{w}$ and $b$ yields:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{3.33}$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.34}$$

The so called dual form of the Lagrange objective function, $L_D$, can now be calculated by inserting (3.33) and (3.34) into (3.32).

$$\begin{aligned} L_D &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \text{ s.t. } \alpha_i \geq 0\forall_i, \sum_{i=1}^{N} \alpha_i y_i = 0 \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i H_{ij} \alpha_j \text{ where } H_{ij} \equiv y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \text{ s.t. } \alpha_i \geq 0 \quad \forall i \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i y_i = 0 \end{aligned} \tag{3.35}$$

Which is a maximisation problem of convex quadratic form, it can be solved by utilizing a quadratic programming solver.

$$\max_{\alpha} \left[ \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \right] \quad \text{s.t.} \quad \alpha_i \geq 0 \quad \forall i \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.36}$$

The solver will give us the vector $\alpha$ which in turn can be inserted into equation (3.33) to give us $\mathbf{w}$. We now study equation (3.30) and note that any data point that is a support vector will satisfy the following equation:

$$y_s(\mathbf{x}_s \cdot \mathbf{w} + b) = 1 \tag{3.37}$$

where $y_s$ and $\mathbf{x}_s$ are the output and attribute-vector of a support vector. Inserting $\mathbf{w}$ from (3.33) into (3.37) yields:

$$y_s \sum_{m \in S} (\alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s) + b = 1 \tag{3.38}$$

Where $S$ denote the complete set of support vectors. $S$ can be determined by locating all indices $i$ where $\alpha_i > 0$. Furthermore, as $y_s \in -1, 1$ we have $y_s^2 = 1$ and in turn:

$$y_s^2 \sum_{m \in S} (\alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s) + b = y_s \tag{3.39}$$

$$b = y_s - \sum_{m \in S} (\alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s) \tag{3.40}$$

To further improve our measurement of $b$, one can use an average of all support vectors instead of one specific $\mathbf{x}_s$

$$b = \frac{1}{N_s} \sum_{s \in S} \left( y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s \right)$$

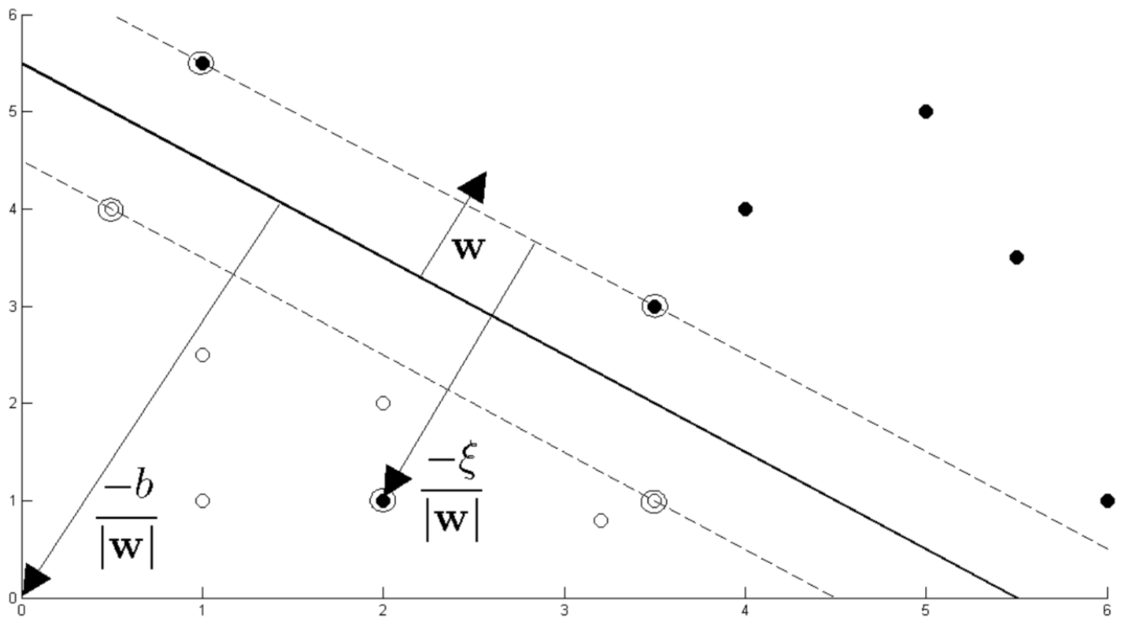
With $\mathbf{w}$ and $b$ determined, new vectors of attributes $\bar{\mathbf{x}}$ is classified by by the equation:

$$\bar{y} = sgn(\mathbf{w}\bar{\mathbf{x}} + b) \tag{3.41}$$

#### 3.5.1.1 Non-linearly separable data

To extend the model mentioned above to cases where the data is not linearly separable.

*Figure 3.6: Hyperplane through two non-linearly separable classes [18]*



The constraints from inequalities (3.28) and (3.29) are modified with a dynamic slack variable $\xi_i$. As $\xi_i$ can vary for different indices $i$, it is now possible for every single data point to fulfil its related constraint.

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i \text{ for } y_i = +1 \tag{3.42}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i \text{ for } y_i = -1 \tag{3.43}$$

$$\xi_i \geq 0 \quad \forall i \tag{3.44}$$

These three inequalities (3.42), (3.43) and (3.44) can be combined into:

$$y_i \left( \mathbf{x}_i \cdot \mathbf{w} + b \right) - 1 + \xi_i \geq 0 \quad \text{where} \quad \xi_i \geq 0 \quad \forall i \tag{3.45}$$

Even though we can fit all data points, it is desirable to have as few points on the wrong side of the dividing hyperplane as possible, so we modify our objective function to incorporate a cost-hyperparameter $C$ for these misclassifications. $C$ can be chosen freely and represents the trade off between maximising the margin and minimising misclassifications. A smaller $C$ implies a larger emphasis on maximising the margin and vice versa for a larger $C$.

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i \quad \text{s.t.} \quad y_i \left( \mathbf{x}_i \cdot \mathbf{w} + b \right) - 1 + \xi_i \geq 0 \quad \forall_i \tag{3.46}$$

This minimization is performed with a similar method as used on (3.31) in the previous section. With $\mathbf{w}$ and $b$ determined with this method, new vectors of attributes $\bar{\mathbf{x}}$ is once again classified by equation (3.41)

### 3.5.2 Support vector regression (SVR)

As mentioned earlier in section 3.1.1.2, regression is the task of predicting a continuous, real valued output $y$ with the help of one or more attributes found in an input vector $\mathbf{x}$. If we call the predicted output from the regression $f(\mathbf{x}_i)$ and the true value $y_i$, the absolute error of the prediction can be formulated as $|f(\mathbf{x}_i) - y_i|$. When performing regression, there are essentially an infinite number of classes, which turns the concept of a margin meaningless. The SVR-algorithm still relies on minimising the L2 norm of the coefficient vector $\mathbf{w}$ but with some modifications.

*Figure 3.7: One-dimensional example of Support Vector Regression. Note the tube formed by the error toleration $\epsilon$ [18]*

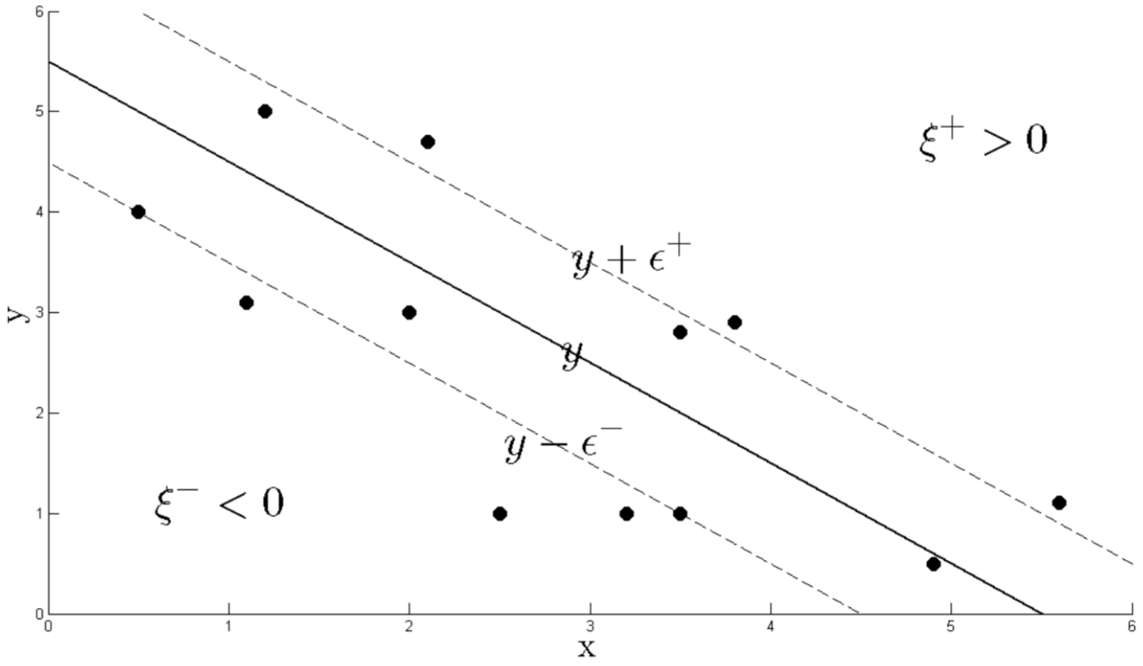


Figure 3.7 depicts regression where $x$ is one-dimensional. When implementing an SVR, two hyperparameters are needed, $\epsilon$ and $C$. $\epsilon$ corresponds to a specific amount of error toleration. If $|f(\mathbf{x}_i) - y_i| \leq \epsilon$, the model will not penalise this misclassification. Missclassifications outside of the $\epsilon$-tube are handled in a similar way as for the support vector classification (SVC), with flexible slack variables $\xi_i^+$ for $f(\mathbf{x}_i) < y_i$ and $\xi_i^-$ for $f(\mathbf{x}_i) > y_i$ These are determined by:

$$y_i \leq f(\mathbf{x}_i) + \epsilon + \xi_i^+ \text{ if } f(\mathbf{x}_i) < y_i \tag{3.47}$$

$$y_i \geq f(\mathbf{x}_i) - \epsilon - \xi_i^- \text{ if } f(\mathbf{x}_i) > y_i \tag{3.48}$$

The penalty function to be minimized can then be written as:

$$C \sum_{i=1}^{N} (\xi_i^+ + \xi_i^-) + \frac{1}{2} \|\mathbf{w}\|^2 \tag{3.49}$$

This needs to be minimized subject to the constraints $\xi_i^+ \geq 0, \quad \xi_i^- \geq 0 \quad \forall i$ in addition to equations (3.47) and (3.48).

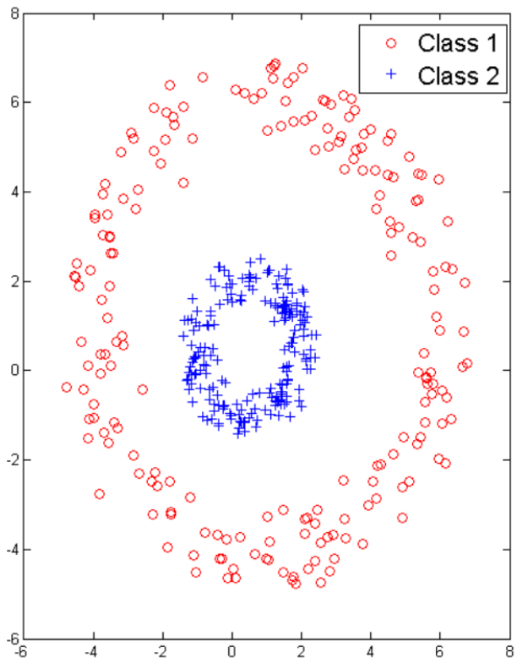
Using similar techniques as in section 3.5.1 yields $\mathbf{w}$ and $b$. New vectors of attributes $\bar{\mathbf{x}}$ are regressed by:

$$\bar{y} = \mathbf{w} \cdot \bar{\mathbf{x}} + b \tag{3.50}$$

### 3.5.3 Non linear SVMs and the Kernel trick

The SVC and SVR explained above have limited usefulness since it operates on an assumption that the data is (at least approximately) linearly separable or regressable.

*Figure 3.8: Data where no linear hyperplane would provide useful classification [18]*



In Figure 3.8, we see a 2-dimensional classification problem where this assumption does not hold. No line can even approximately divide the two classes. To solve this classification problem, we need to extend our method of creating linear SVMs to nonlinear cases. This is done by mapping the attributes onto a higher dimensional space [21]. A 2-D attribute vector $\mathbf{x} = (x_1, x_2)$ could for example be mapped onto 3-D, using the mappings $\mathbf{z} = (z_1, z_2, z_3) = \phi(\mathbf{z}) = (x_1, x_2, x_1 x_2)$. In this 3-D space we can then find the optimal dividing hyperplane as explained in 3.5.1 and 3.5.2. This method does have some issues, however. First, one needs to find a mapping function $\phi$ that performs well. Second, utilising this method involves many computationally heavy dot-products [21]. These issues can be resolved by using a so called Kernel function on the attribute data. A Kernel function $K$ is defined by:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \tag{3.51}$$

This means that all higher dimension dot-products can be calculated in the lower original dimension. Furthermore, we actually do not need to determine the mapping-function $\phi$ when employing a Kernel function. Three common Kernel functions are:

$$\text{Polynomial kernel of degree h:} \quad K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^h \tag{3.52}$$

$$\text{Gaussian radial basis function kernel:} \quad K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|x_i - x_j\|^2}{2\gamma^2}} \tag{3.53}$$

$$\text{Sigmoid kernel:} \quad K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \cdot \mathbf{x}_j - \delta) \tag{3.54}$$

Where $h$, $\gamma$, $\kappa$ and $\delta$ are free parameters determining the behavior of the Kernel functions.

*Figure 3.9: Data from Figure 3.8 re-mapped to 3-D [18]*

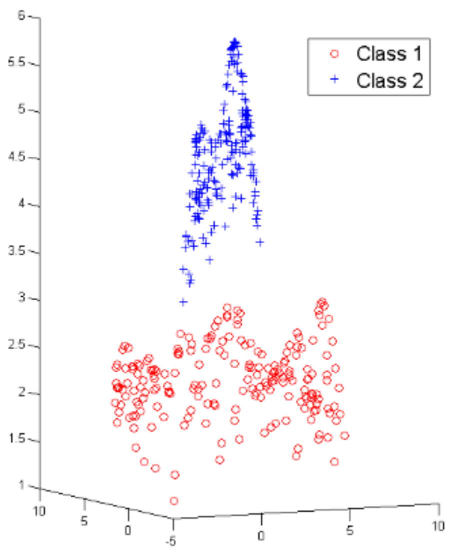


Figure 3.9 displays attribute space that the Gaussian radian basis function kernel defines implicitly when it is used on the data from Figure 3.8 [18]. Unlike in Figure 3.8, in Figure 3.9, we can clearly find a separating hyperplane.

When using one of the functions (3.52), (3.53), (3.54) or any other Kernel function, the parameters of the nonlinear SVM that lead to the largest margin can be calculated with similar methods as used for the various linear SVMs described in sections 3.5.1 and 3.5.2.

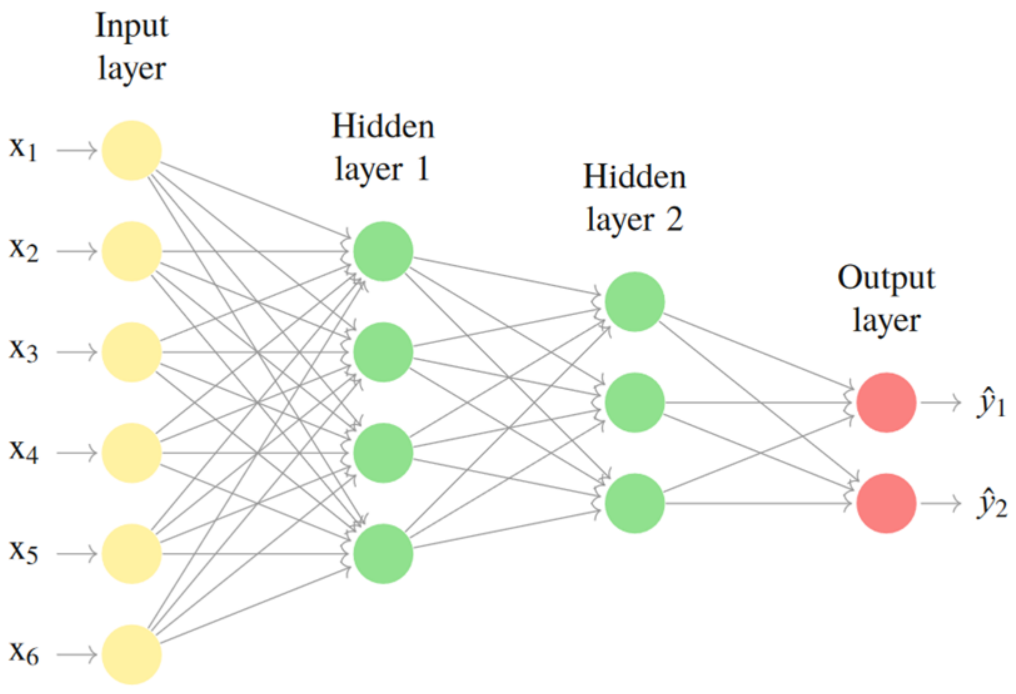
## 3.6 Artificial Neural Networks

Artificial Neural Networks (ANNs) are created by connecting many small units called neurons into large network capable of solving prediction problems. ANNs get their name from their resemblance of the interconnected simple neurons that, through strength in number and interconnection, emulates our human brains. The behaviour of the network can be determined by the weights of the links that connect the neurons [29]. The machine learning related task when implementing ANNs can essentially be reduced to provide algorithms that can determine weights that result in acceptable classification/regression performance.

### 3.6.1 Multilayer perceptrons (MLPs)

An MLP is defined by an input layer, one or more hidden layers and an output layer. These layers in turn consist of one or more of the mentioned neurons. With two hidden layers, the architecture can look like Figure 3.10.

*Figure 3.10: An MLP with 2 hidden layers, 6 attributes and 2 outputs [13]*



The input layer consists of neurons that only receive data from one of the attributes in the input vector $\mathbf{x}$, the number of input neurons is therefore equal to the number of attributes. Trainable weights connect the neurons in the input layer to the neurons in hidden layer 1. Additional weights in turn connect these neurons, to neurons in hidden layer 2. Lastly another set of weights connects to the neurons in the output layer. Data propagates through the network until the output nodes are reached. In regression, one output node is generally used unless multi-output regression is to be performed. In classification, the number of output nodes can vary depending on how many classes one must classify to. The number of hidden layers and nodes in every hidden layer on the other hand, can only be determined empirically as there exists a trade-off. Too many hidden neurons produce overfitting and too few results in a lack of generalisation for the network [1]. If the inputs to a node are $x_1, x_2, \ldots, x_n$ and the weights are $w_1, w_2, \ldots, w_n$, the output $y$ of the node is determined by the following equation:

$$y = f(x_1w_1, x_2w_2, \ldots, x_nw_n) \tag{3.55}$$

Where $f$ is referred to as the activation function. In principle, any function can be used and a common one is the sigmoid function defined as:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3.56}$$

The output from the node is then propagated forward to the next layer, this process is then repeated for all nodes and all layers until the output nodes are determined.

### 3.6.2 Recurrent neural networks (RNNs)

An MLP is an example of a feedforward neural network. These networks are simple in the sense that information flows forward and only forward in the network. Looking at Figure 3.10 we see that all ”arrows” with information flow to the right and do not form any cycles. For each new input-vector $\mathbf{x}$, one receives an output vector $\mathbf{y}$. If the data is in the form of a time-series, no regard is given to specific time $t$ that the input vector represents. In reality, we often want to capture the time-dynamics of the input data and its effect on the output and to achieve this we can implement a RNN. In a general sense, the only difference between an MLP and an RNN can be described with the following equations [13]:

$$\hat{y}_t = f\left(\mathcal{X}_t\right) \text{ where } \mathcal{X}_t := \text{seq}_{T,t}(\mathbf{X}) = (\mathbf{x}_{t-T+1}, \ldots, \mathbf{x}_t) \tag{3.57}$$

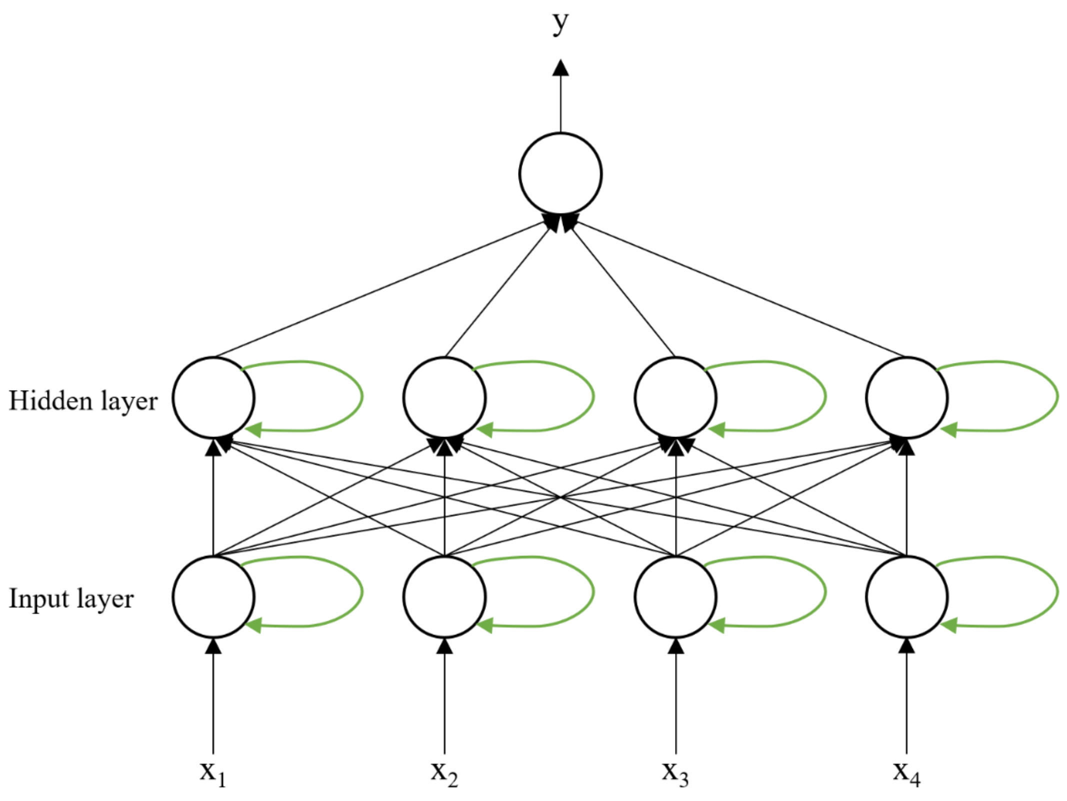
$$\hat{y}_t = f(\mathbf{x}_t) \tag{3.58}$$

where $\mathbf{x}_{t-j}$ is a $j$th lagged observation of $\mathbf{x}_t$. Equation (3.57) here represents the workings of an RNN and equation (3.58) an MLP.

#### 3.6.2.1 Vanilla RNNs

By modifying a feedforward neural network, such as an MLP with so called recurrent links, we can create a simple, or vanilla RNN.

*Figure 3.11: A RNN with 1 hidden layer, 4 attributes and 1 output*



In Figure 3.11 we see such an RNN. The recurrent links are highlighted in green and symbolise using node-outputs from previous time-steps as inputs in future time-steps. As a result, the network now reacts not only to the input vector, but also to various node-outputs from previous steps. We see that the recurrent links can be used in various ways, one can use the previous outputs to influence the current input node or let an internal node be influenced by its previous output. As every new input-output relation in an RNN is affected by the previous relation and in turn, the relation before that, the network implicitly contains the reaction from all previous input-output relations [29]. For an internal node with a recurrent link with itself, the output of the node can be described as [12]:

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \tag{3.59}$$

Where $\mathbf{h}_t$ is the internal node output at time $t$, $f$ the activation function and $\mathbf{W}_{hh}$ a vector of weights given to the given to the recurrent link and $\mathbf{W}_{xh}$ the vector of weights given to outputs from the previous layer. $\mathbf{b}_h$ is a vector of bias terms. Simple RNNs were introduced by Elman in 1990 [15] and are only capable of learning short and simple patterns. More complicated patterns remained unsolved even with their introduction [29].

#### 3.6.2.2 Long short-term memory (LSTMs)

The long short-term memory (LSTM) [22] is an attempt to solve some of the issues with the RNN. As mentioned, the RNN can only learn short-term pattern, i.e. short term memory. The LSTM derives its name from the fact that it can learn both long and short-term patterns, i.e. long short-term memory. LSTMs maintain this long-term memory by using memory cells as nodes. These state of these memory cells, the cell states, are then regulated with associated gates. Every cell state $\mathbf{C}_t$ is altered by a forget gate $\mathbf{f}_t$, an input gate $\mathbf{i}_t$ and an output gate $\mathbf{o}_t$ Mathematically, this relation can be explained by the following equations [32]:

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f\left[\mathbf{h}_{t-1};\mathbf{x}_t\right] + \mathbf{b}_f\right) \tag{3.60}$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i\left[\mathbf{h}_{t-1};\mathbf{x}_t\right] + \mathbf{b}_i\right) \tag{3.61}$$

$$\tilde{\mathbf{C}}_t = \tanh\left(\mathbf{W}_c\left[\mathbf{h}_{t-1};\mathbf{x}_t\right] + \mathbf{b}_c\right) \tag{3.62}$$

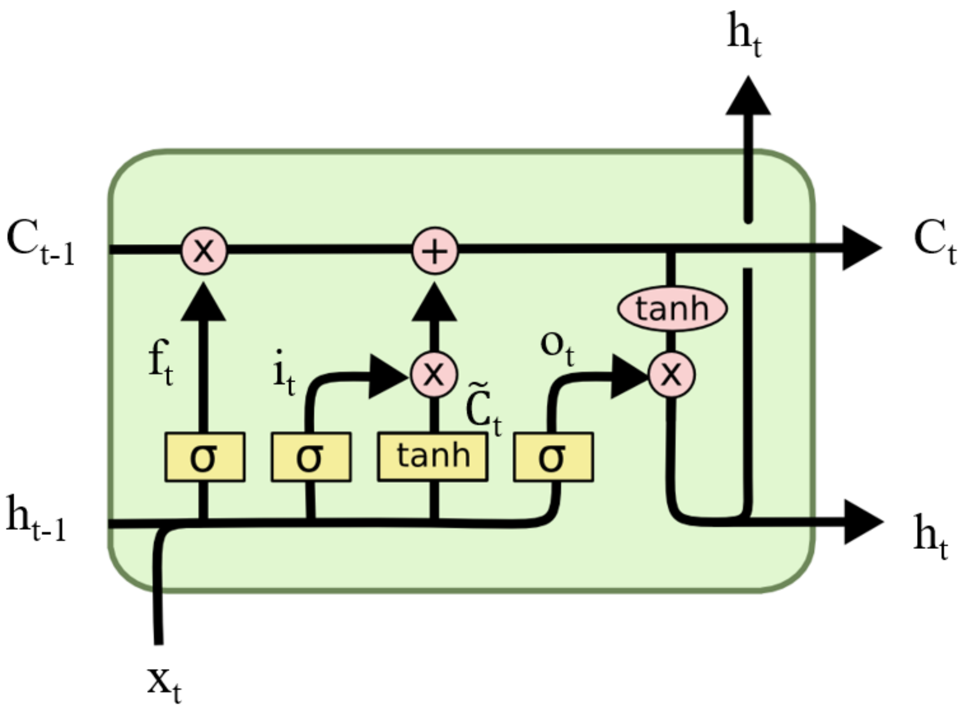
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\left[\mathbf{h}_{t-1};\mathbf{x}_t\right] + \mathbf{b}_o\right) \tag{3.63}$$

$$\mathbf{C}_t = \mathbf{C}_{t-1} * \mathbf{f}_t + \tilde{\mathbf{C}}_t * \mathbf{i}_t \tag{3.64}$$

$$\mathbf{h}_t = \tanh\left(\mathbf{C}_t\right) * \mathbf{o}_t \tag{3.65}$$

Here $\sigma$ is the sigmoid function from equation (3.56), $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_C$ and $\mathbf{W}_o$ are weight vectors, $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_C$ and $\mathbf{b}_o$ are bias-vectors, $\tilde{\mathbf{C}}_t$ the cell state candidate, $[x;y]$ is vector concatenation of vector $x$ and $y$ and $*$ is element-wise multiplication. $\mathbf{x}_t$ is the input to the cell at time $t$ this could be either an attribute or the output from a neuron on a previous layer. Lastly, $\mathbf{h_t}$ is the hidden state output.

*Figure 3.12: A LSTM-cell [11]*



The process of updating the cell state can be explained by Figure 3.12. The steps are:

1. $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ are concatenated, multiplied by a weight vector $\mathbf{W}_f$ and inserted into the sigmoid function. $\mathbf{C}_{t-1}$ is multiplied by this output $\mathbf{f}_t \in [0, 1]$ so $(1 - \mathbf{f}_t) \in [0, 1]$ is the forget factor used on the previous cell state $\mathbf{C}_{t-1}$.

2. $\tilde{\mathbf{C}}_t$ and $\mathbf{i}_t$ are calculated and multiplied element-wise. The result is added to the cell state. The cell state $\mathbf{C}_t$ has now been calculated.

3. $\mathbf{C}_t$ is passed forward to be used in calculating $\mathbf{C}_{t+1}$.

4. $\mathbf{o}_t$ is calculated and multiplied with $\tanh(\mathbf{C}_t)$. The result of this is the hidden state $\mathbf{h}_t$. This is passed forward to the next layer of the neural network and to itself for the next time-step where it will constitute $\mathbf{h}_{t-1}$.

# 4

# Method

## 4.1 Software

The analysis of this thesis was performed in Python 3.9.7. The Numpy and Pandas libraries were used for basic mathematical functions and data structuring. For machine learning, pre-made predictors from Scikit-learn were utilised for SVM and Random forest while PyTorch was used to create neural networks.

## 4.2 Data

Stock market data on the 34 companies that were part of the OMXS30 index for the entire or part of the analysed time period, (see section 4.2.1) were downloaded from Yahoo finance through the Python library yfinance. The data includes adjusted close prices, close prices, open prices, day-high prices, and day-low prices. The frequency of the data is daily on a trading day (c. 252 days per year) basis.

Table 4.1: *The composition of OMXS30 from 2016-2021. A stock is present in the investment universe of the models if it is present in OMXS30*

| Stock ticker | Included in OMXS30 | Excluded from OMXS30 |
|---|---|---|
| EVO.ST | 2021-01-01 | n.a. |
| SCA-B.ST | Prior to 2016 | n.a. |
| TELIA.ST | Prior to 2016 | n.a. |
| SWED-A.ST | Prior to 2016 | n.a. |
| TEL2-B.ST | Prior to 2016 | n.a. |
| NDA-SE.ST | Prior to 2016 | n.a. |
| INVE-B.ST | Prior to 2016 | n.a. |
| ABB.ST | Prior to 2016 | n.a. |
| AZN.ST | Prior to 2016 | n.a. |
| ATCO-B.ST | Prior to 2016 | n.a. |
| HEXA-B.ST | 2018-07-01 | n.a. |
| GETI-B.ST | Prior to 2016 | n.a. |
| ERIC-B.ST | Prior to 2016 | n.a. |
| BOL.ST | Prior to 2016 | n.a. |
| ATCO-A.ST | Prior to 2016 | n.a. |
| ELUX-B.ST | Prior to 2016 | n.a. |
| SWMA.ST | Prior to 2016 | n.a. |
| ESSITY-B.ST | Prior to 2016 | n.a. |
| VOLV-B.ST | Prior to 2016 | n.a. |
| HM-B.ST | Prior to 2016 | n.a. |
| SEB-A.ST | Prior to 2016 | n.a. |
| ALFA.ST | Prior to 2016 | n.a. |
| SHB-A.ST | Prior to 2016 | n.a. |
| ALIV-SDB.ST | 2017-01-01 | n.a. |
| SKF-B.ST | Prior to 2016 | n.a. |
| SKA-B.ST | Prior to 2016 | n.a. |
| KINV-B.ST | Prior to 2016 | n.a. |
| SINCH.ST | 2021-07-01 | n.a. |
| SAND.ST | Prior to 2016 | n.a. |
| ASSA-B.ST | Prior to 2016 | n.a. |
| SECU-B.ST | Prior to 2016 | 2021-06-30 |
| SSAB-A.ST | Prior to 2016 | 2020-12-31 |
| FING-B.ST | 2016-01-01 | 2018-06-30 |
| LUNE.ST | Prior to 2016 | 2017-12-31 |

The Sanctify ESG-scores for the 34 companies were downloaded for the same period through an API provided by Sanctify.

### 4.2.1 Time span

The analysed time period is 2016-01-01 – 2021-12-31. Stock market data for the relevant companies, of course extend much further back than 2016, and it is possible that the predictors would have performed better with access to longer time-series of data. As explained in section 1.4 however, the data that Sanctify derive their scores from are scarcely available before 2016. As the main objective (section 1.3) is to examine the predictive power of Sanctify ESG-scores, a shorter time span was deemed better suited for the task.

### 4.2.2 Train-test split

When creating machine learning predictors based on supervised learning, a split of the available data into training and test data is necessary. The training data is used to train the predictors which later are evaluated on the test data. The data was split in train data: 2016-01-01 – 2019-12-31 and test data 2020-01-01 – 2021-12-31, a $\frac{2}{3}$ split. This split was selected to include the covid induced market crash of March 2020 in the test data. This market crash represented a unique market environment not seen since the last century and therefore challenges the predictors which are trained in a more stable market environment.

## 4.3 Attribute selection

To evaluate the effect that ESG-data can have on stock price prediction, several models with or without access to this data are created and compared. To further evaluate the general effectiveness of the more advanced models, they are compared against a dynamic ARIMAX-model implemented via a Kalman filter. This model will be called the "Naive model". The attribute selection can therefore divided into two categories. Attribute selection for the Naive model and attribute selection for the all other models.

### 4.3.1 ARIMAX-model

With an ARIMAX model, the attributes can be said to be the exogenous inputs mentioned in section 3.2.4. As the ARIMAX-model was created as a simple benchmark, it also uses a simple version of this exogenous input. The only exogenous input used in this model is the latest available Sanctify short-term absolute ESG-score. This correspond to the attribute named 'ESG' in table 4.3.

### 4.3.2 Other models

The selection of attributes, was as explained in section 1.4, inspired by Vijh et. al. [51]. For the base models without access to ESG-data, the following attributes were used.

*Table 4.2: General attributes*

| Attribute | Definition | Formula |
|---|---|---|
| H-L | Previous day-high less day-low | $h_{t-1} - l_{t-1}$ |
| C-O | Previous day-close less day-open | $c_{t-1} - o_{t-1}$ |
| SMA_7 | 7 day moving average of adj. close | $\frac{1}{7}\sum_{i=1}^{7} y_{t-i}$ |
| SMA_14 | 14 day moving average of adj. close | $\frac{1}{14}\sum_{i=1}^{14} y_{t-i}$ |
| SMA_21 | 21 day moving average of adj. close. | $\frac{1}{21}\sum_{i=1}^{21} y_{t-i}$ |
| STD_7 | 7 day standard deviation of adj. close | $\sqrt{\frac{\sum_{i=1}^{7}(y_{t-i}-\bar{y})}{7}}$ where $\bar{y} = \frac{1}{7}\sum_{i=1}^{7} y_{t-i}$ |

$h_t$ is the day-high price, $l_t$ the day-low price, $c_t$ the closing price, $o_t$ the open price and $y_t$ the adjusted close price at time $t$

With the ESG-data from Sanctify, an additional seven attributes were created.

*Table 4.3: ESG-derived attributes*

| Attribute | Definition | Formula |
|---|---|---|
| ESG | Last available ESG-score | $E_{t-2}$ |
| ESG_SMA_7 | 7 day moving average of ESG | $\frac{1}{7}\sum_{i=2}^{8} E_{t-i}$ |
| ESG_SMA_14 | 14 day moving average of ESG | $\frac{1}{14}\sum_{i=2}^{15} E_{t-i}$ |
| ESG_SMA_21 | 21 day moving average of ESG | $\frac{1}{21}\sum_{i=2}^{22} E_{t-i}$ |
| ESG_MOM_7 | 7 day momentum of ESG | $E_{t-2} - E_{t-8}$ |
| ESG_MOM_14 | 14 day momentum of ESG | $E_{t-2} - E_{t-15}$ |
| ESG_MOM_21 | 21 day momentum of ESG | $E_{t-2} - E_{t-22}$ |

$E_t$ is the Sanctify short-term absolute ESG-score at time $t$. Scores for the day $t$ are published on day $t + 1$ and can therefore be used when predicting prices on $t + 2$.

To deduce the effect of the ESG-derived attributes, four versions of each model with access to different sets of attributes are created. We call the versions *attribute profiles* and name the profiles ESG_BASE, ESG_SMA, ESG_MOM and ESG_ALL. Lastly we add the profile NO_ESG, a profile with no access to ESG-data to be used as comparison. The attribute composition of each profile is explained in Table 4.4.

*Table 4.4: Attribute profiles*

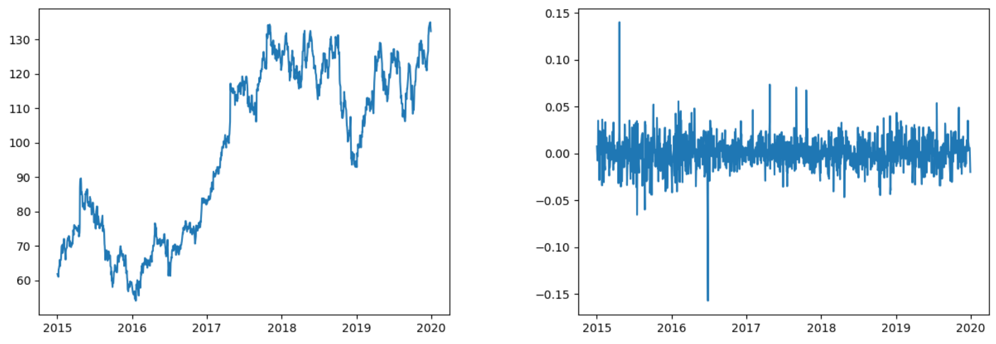
| Attribute | NO_ESG | ESG_BASE | ESG_SMA | ESG_MOM | ESG_ALL |
|---|---|---|---|---|---|
| H-L | x | x | x | x | x |
| C-O | x | x | x | x | x |
| SMA_7 | x | x | x | x | x |
| SMA_14 | x | x | x | x | x |
| SMA_21 | x | x | x | x | x |
| STD_7 | x | x | x | x | x |
| ESG | | x | | | x |
| ESG_SMA_7 | | | x | | x |
| ESG_SMA_14 | | | x | | x |
| ESG_SMA_21 | | | x | | x |
| ESG_MOM_7 | | | | x | x |
| ESG_MOM_14 | | | | x | x |
| ESG_MOM_21 | | | | x | x |

## 4.4 Pre-processing

Machine learning operates under the assumption that test and train-data originates from the same probability distribution. If $X$ is a random variable, both $\mathbf{x}_{test}$ and $\mathbf{x}_{train}$ should be vectors of outcomes from $X$. With time-series data, this is complicated by the fact that processes can change in structure with time. Such *non-stationary* processes need processing to more closely resemble stationary data. Stock prices are a classical example of non-stationary data. They generally display both increases in mean and in variance. To remove this structure the data is pre-processed. Instead of using the absolute price of the stock at time $t$. We instead set

$y_t$ as the daily log-return of the stock, i.e. $y_t = ln(p_t/p_{t-1})$ where $p_t$ is the adjusted close price (section 2.4) of the stock. This removes much of the increase in mean and variance as can be seen below in Figure 4.1

*Figure 4.1: The adjusted close price of of Volvo AB (VOLV-B.ST) 2015-2019. On the left we see the unaltered price and on the right the daily log-returns*



The attributes were also processed with a similar method. For some attributes the log difference was extracted, i.e.

$$x'_t = ln(x_t/x_{t-1}) \tag{4.1}$$

where $x$ is the original attribute and $x'_t$ is the log difference of the attribute. For other attributes, the standard difference was extracted, i.e.

$$x'_t = x_t - x_{t-1} \tag{4.2}$$

where $x$ is the original attribute and $x'_t$ is the standard difference of the attribute. The reason for this discrepancy is the fact that some attributes can have instances of sign-switching, i.e. they can display both positive and negative values. Extracting the log difference from these types of attributes results in unwanted major swings when a attribute switches from positive to negative or vise versa.

*Table 4.5: Processing methods for the attributes*

| Attribute | Log difference | Standard difference |
|---|---|---|
| H-L | | x |
| C-O | | x |
| SMA_7 | x | |
| SMA_14 | x | |
| SMA_21 | x | |
| STD_7 | x | |
| ESG | | x |
| ESG_SMA_7 | | x |
| ESG_SMA_14 | | x |
| ESG_SMA_21 | | x |
| ESG_MOM_7 | | x |
| ESG_MOM_14 | | x |
| ESG_MOM_21 | | x |

To finish pre-processing, standardisation of both attributes and outputs, i.e. the $(\mathbf{x_t}, y_t)$ tuples is performed with a min-max scaler. The scaler normalises attributes/outputs according to the formula:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.3}$$

where $x$ is the original value $x_{min} = \min(\mathbf{x}_{train})$ and $x_{max} = \max(\mathbf{x}_{train})$. This normalises all attributes and outputs in the train data to $\mathbf{x}_{train} \in [0, 1]$. Test data can be scaled to both $> 1$ if $x > x_{max}$ and to $< 0$ if $x < x_{min}$.

## 4.5 Model creation

A general principle for all models in this thesis is the use of individual models for each stocks, i.e. no multi-output models are used. Instead individual models for all stocks are created by for-looping the model creation process. This same principle is also used regarding the five attribute profiles mentioned in Table 4.4. The process of stock prediction can generally for all models therefore be described as follows.

**Algorithm 1** Model procedure

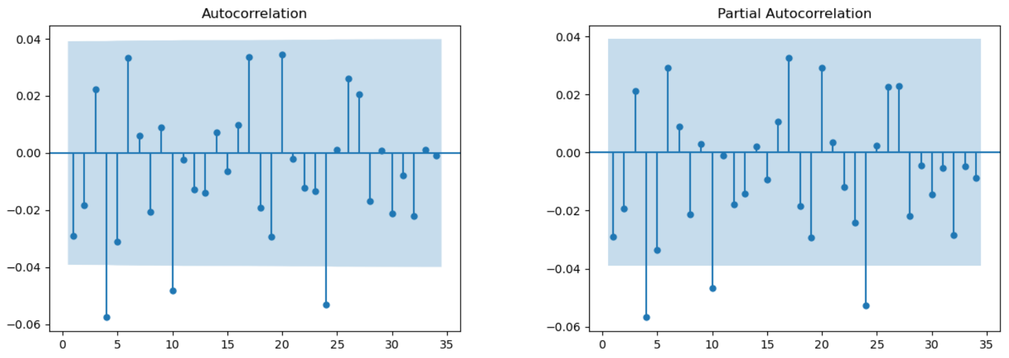
```
1: for <every stock> do
2:     Extract attributes
3:     Pre-processing of data
4:     Optimise hyperparameters
5:     Train model with train data
6:     Predict using test data
```

Technically, we therefore create $34 \times 5 = 170$ models of all model-types, (with model-type we here refer to the machine learning algorithm used, such as SVM or LSTM). These models will not be evaluated on a individual basis though. Instead every combination of attribute profile and model-type will be evaluated on its cumulative performance on predicting the entire investment universe. For example, the 34 individual SVM-regressors based on the "ESG_SMA" attribute profile will have their predictions merged into one dataframe. This dataframe will then be used to evaluate the (SVM, ESG_SMA) tuple. Each of these tuples will be referred to as a "model" from now on. We differentiate between the naive models (section 4.5.1) and the primary models (sections 4.5.2, 4.5.3 and 4.5.4). Two naive modes, one with access to ESG-data and one without such access, for comparison purposes. For each primary model-type (SVM, Random forest, LSTM), five models, one for each attribute profile, are created.

### 4.5.1 Naive-model

The Naive models are modelled as ARIMA-models with exogenous inputs (ARIMAX-models) implemented via Kalman filter. As explained in section 4.4, the output data was differentiated once, so we have ARIMA-models of order (p,1,q) with an exogenous input. The order of the AR-term $p$ and of the MA-term $q$ was determined through analysis of the auto-correlation function and the partial auto-correlation function.

*Figure 4.2: The autocorrelation function (left) and partial autocorrelation function (right) of the OMXS30GI index 2010-2019. The shaded areas are 95%-confidence intervals*



In Figure 4.2, we see moderate indication for correlation at time-lags 4,10,24. Since this model is meant to operate as a naive comparison, the order of the AR and MA polynomials are simply assumed to be 4. Since every stock is modeled individually, it is possible that the order of the AR and MA polynomials vary for different stocks. Since this model is naive, we simplify this by assuming that all stocks have the same polynomial order as the OMXS30GI index. We therefore arrive at ARIMA(2,1,2) models with exogenous inputs.

A naive approach is used for the exogenous input. We solely use the attribute named "ESG" in table 4.3.

With this, we set up the Linear state space representations as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{e}_t \tag{4.4}$$

$$y_t = \mathbf{C}_{t|t-1}\mathbf{x}_{t|t-1} + w_t \tag{4.5}$$

With $y_t$ defined as the stock-price differentiated according to equation (4.2), $\mathbf{e}_t$ is additive noise allowing the system parameters to change over time and $w_t$ is the white noise measurement error and:

$$\mathbf{C}_{t|t-1} = [-y_{t-1}, -y_{t-2}, -y_{t-3}, -y_{t-4}, e_{t-1}, e_{t-2}, e_{t-3}, e_{t-4}, E_{t-2}] \tag{4.6}$$

Where $E_t$ is the ESG-score at time $t$ and: $e_t = \hat{y}_t - y_t$ where $\hat{y}_t$ is the prediction of $y_t$. We also have: $\mathbf{x_t} = [a_1, a_2, a_3, a_4, c_1, c_2, c_3, c_4, b_2]$

We can now recursively estimate stock-prices using the method described in section 3.3.1.2

### 4.5.2 Support Vector Machine

The SVM-regressor was implemented through the Scikit-learn class *sklearn.svm.SVR* [43]. Hyperparameter optimisation was performed via random search. 50 random combinations of parameters $C$, (equation (3.49) $\epsilon$ (equations (3.47) and (3.48)) and $\gamma$ (equation (3.53) were tested and evaluated through minimisation of the MSE. The evaluation was performed through cross-validation.

*Table 4.6: Hyperparameter optimisation, SVM*

| Hyperparameter | Parameter range | Scikit learn default |
|---|---|---|
| $C$ | $[0.5, 2.0]$ | 1.0 |
| $\gamma$ | $[0.05, 0.50]$ | see [43] |
| $\epsilon$ | $[0.05, 0.20]$ | 0.10 |

50 random hyperparameter combinations from the allowed ranges in table 4.6 were tested for each individual stock predictor. No requirement for uniformity in each model was set so a model could in theory have different values for $C$ for different stocks. All remaining hyperparameters retained the Scikit-learn default values [43].

### 4.5.3 Random forest

The Random forest regressor was implemented through the Scikit-learn class *sklearn.ensemble.RandomForestRegressor* [42]. The hyperparameters chosen for optimisation were:

- The number of trees used in the forest $n\_estimators$.
- The amount of features randomly chosen for comparison when splitting a node: $max\_features$.
- The minimum number of required samples in a node for it to be split instead of being turned into a leaf $min\_samples\_split$.

*Table 4.7: Hyperparameter optimisation, Random forest*

| Hyperparameter | Parameter range | Scikit learn default |
|---|---|---|
| $n\_estimators$ | $[25, 400]$ | 100 |
| $max\_features$ | $dim(\mathbf{x}) * [0.2, 1.0]$ | $dim(\mathbf{x})$ |
| $min\_samples\_split$ | $[2, 50]$ | 2 |

Where $dim(\mathbf{x})$ is the number of attributes

As for the SVM, different stocks can have different values on their respective hyperparameters and the hyperparameters that were not optimised retained their Scikit-learn default values.

### 4.5.4 Long short-term memory (LSTM)

The LSTM-architecture which is utilised, is a modified version of Kuguoglu's Pytorch-based architecture [30]. Unlike all other models, the data was not simply split into test and train data as explained in section 4.2.2. Instead a 3-part split into train, validation and test data was performed. The split was formed by first performing the $\frac{2}{3}$ train/test split as detailed in section 4.2.2 and then splitting the train data into train data (80%) and validation data (20%). Hyperparameter optimisation was performed via a self-constructed random search algorithm.

**Algorithm 2** LSTM hyperparameter optimisation

```
1: for <every stock> do
2:     for i in range(50) do
3:         Randomly select a dropout $\in [0.0, 0.5]$ (uniformly spaced)
4:         Randomly select a learning rate $\in [10^{-5}, 10^{-1}]$ (logarithmically spaced)
5:         Train with 50 epochs
6:         Predict on validation data
7:         Calculate the MSE of the prediction
8:     Select the dropout, learning rate combination that gave the lowest MSE
```

In addition to the dropout and the learning rate. The chosen value for other hyperparameters can be found in table 4.8

*Table 4.8: Remaining hyperparameters, LSTM*

| Hyperparameter | Chosen value | Description |
|---|---|---|
| *batch_size* | 64 | Number of samples processed before the weights are updated |
| *weight_decay* | $10^{-6}$ | Regularisation term with reduces overfitting |
| *hidden_dim* | 64 | Nodes per hidden layer |
| *layer_dim* | 2 | Number of hidden LSTM-layers |
| *n_epochs* | 50 | Number of epochs |

## 4.6 Trading algorithm

The predictors predict daily log-returns for all stocks allowed to trade. These log-returns are converted to simple returns with the formula $r_t = e^{r_{log,t}} - 1$ were $r_t$ is the simple returns and $r_{log,t}$ the log-returns at time $t$. The following question now arises: *How shall one use these predicted returns to construct a trading strategy?* A naive approach would be to simply fully invest in the stock with the largest predicted next-day return. This approach would be very sensitive to prediction errors and also would lack in providing risk-adjusted returns, a more sophisticated approach is therefore used. Utilising one of the benchmarks for the predictors, the investment portfolio is constructed by maximising the Sharpe ratio (section 2.6.1), with two additional constraint. No short-selling is allowed and no individual stock is allowed to account for $> 10\%$ of the portfolio. The motive for the first constraint is explained in section 2.2 and the second is to reduce the impact of large prediction errors. 10% was deemed as a fair limit as the largest index weight of a single stock in the OMXS30, as of May 2022, was 7.37% [38] .The maximisation was performed with a self-made algorithm explained in pseudo code below:

**Algorithm 3** Trading algorithm

```
1: procedure
2:     stocklist ← List of stocks allowed to trade, this varies from date to date
3:     predictedreturns ← Daily predicted returns for all stocks $\in$ stocklist
4:     returns ← Daily true returns for all stocks $\in$ stocklist
5:     daterange ← start and end dates for the algorithm
6:     date ← start date for the algorithm
7:     while date $\in$ daterange do
8:         $\mathbf{\Omega}_t \leftarrow$ Cov. matrix for the historical returns of the stocks $\in$ stocklist
9:         $\hat{\mathbf{R}}_t \leftarrow$ Predicted next-day returns for the stocks
10:        $\mathbf{w}_t \leftarrow \max_{\mathbf{w}_t} \frac{\mathbf{w}_t^T \cdot \hat{\mathbf{R}}_t}{\mathbf{w}_t^T \cdot \mathbf{\Omega}_t \cdot \mathbf{w}_t}$ where $\mathbf{w}_t^T = (w_{t,1}, ..., w_{t,N}), \quad 0 \leq w_{t,i} < \frac{1}{10} \forall i$
```

Where $w_{t,i}$ is the portfolio weight of stock $i$ at time $t$. Daily portfolio returns from time $t$ to $t+1$ are now given by $\mathbf{w}_t^T \cdot \mathbf{R}_t$ where $\mathbf{R}_t$ is the vector with true stock-returns from time $t$ to $t+1$. *stocklist* correspond to the stocks present in OMXS30 at *date* and $N$ is the length of *stocklist*, table 4.1 explain this in detail.

## 4.7 Calculation of benchmarks

When the models have been used to predict stock prices and the trading algorithm has transformed these predictions into financial performance, benchmarking can be performed. The Sharpe-ratios (section 2.6.1), the Sortino-ratios (section 2.6.2) and the gross returns (section 2.6.3) are calculated on the outputs of the trading algorithm.

The naive models and the SVRs are all deterministic models, meaning that they produce the same results if they are re-fitted to the test data and later used for prediction one more time. Due to this, the calculated benchmarks for these models are simply the single observed figure from running these models once.

This is not the case for the LSTM-models and the random forest regressors though. As explained in sections 3.4.2 and 3.4.3, during construction of a random forest model, randomness is involved when fitting the model to test data. For an LSTM, randomness is instead introduced during the fitting procedure as a result of the weight vectors mentioned in 3.6.2.2. The initial values of these are set by randomness and these initial values maintain influence even over the epochs.

To partially correct for the randomness in the random forest and the LSTM, the fitting procedure is repeated 10 times. Prediction on test data is then performed for every one of these instances. The outputs from the predictions are inserted into the trading algorithm (section 4.6) and benchmarks are calculated for each instance. The worst observation, best observation and the mean of all 10 observation is presented in section 5.1.

# 5

# Results

## 5.1 Benchmarking

In table 5.1 we present the results from the benchmarking:

*Table 5.1: Benchmarking results*

| Model | SR | | | Sortino ratio | | | Gross returns | | |
|---|---|---|---|---|---|---|---|---|---|
| | Min | Mean | Max | Min | Mean | Max | Min | Mean | Max |
| SVM_NO_ESG | n.a. | 1.103 | n.a. | n.a. | 1.535 | n.a. | n.a. | 53.4% | n.a. |
| SVM_ESG_BASE | n.a. | 0.897 | n.a. | n.a. | 1.232 | n.a. | n.a. | 42.1% | n.a. |
| SVM_ESG_SMA | n.a. | 1.046 | n.a. | n.a. | 1.451 | n.a. | n.a. | 49.9% | n.a. |
| SVM_ESG_MOM | n.a. | 1.095 | n.a. | n.a. | 1.553 | n.a. | n.a. | 54.4% | n.a. |
| SVM_ESG_ALL | n.a. | 0.700 | n.a. | n.a. | 0.970 | n.a. | n.a. | 29.1% | n.a. |
| RF_NO_ESG | 0.647 | 0.766 | 0.830 | 0.870 | 1.047 | 1.162 | 26.9% | 33.6% | 37.1% |
| RF_ESG_BASE | 0.798 | 0.897 | 1.011 | 1.072 | 1.222 | 1.386 | 35.4% | 41.3% | 47.5% |
| RF_ESG_SMA | 1.120 | 1.185 | 1.280 | 1.564 | 1.658 | 1.793 | 55.8% | 60.5% | 68.0% |
| RF_ESG_MOM | 0.837 | 1.056 | 1.174 | 1.137 | 1.459 | 1.643 | 38.1% | 51.6% | 60.6% |
| RF_ESG_ALL | 0.972 | 1.116 | 1.317 | 1.337 | 1.549 | 1.850 | 47.1% | 55.6% | 70.9% |
| LSTM_NO_ESG | 0.934 | 1.033 | 1.181 | 1.253 | 1.395 | 1.630 | 45.2% | 52.6% | 63.8% |
| LSTM_ESG_BASE | 0.794 | 1.099 | 1.406 | 1.064 | 1.508 | 1.964 | 34.3% | 52.4% | 73.2% |
| LSTM_ESG_SMA | 1.064 | 1.142 | 1.206 | 1.446 | 1.553 | 1.655 | 52.7% | 59.2% | 65.0% |
| LSTM_ESG_MOM | 0.735 | 0.826 | 0.890 | 0.986 | 1.112 | 1.201 | 33.2% | 38.2% | 42.7% |
| LSTM_ESG_ALL | 0.629 | 0.747 | 0.852 | 0.855 | 1.018 | 1.155 | 26.3% | 33.0% | 39.8% |
| Naive_NO_ESG | n.a. | 0.890 | n.a. | n.a. | 1.235 | n.a. | n.a. | 42.7% | n.a. |
| Naive_ESG | n.a. | 0.923 | n.a. | n.a. | 1.285 | n.a. | n.a. | 44.8% | n.a. |
| OMXS30GI | n.a. | 0.896 | n.a. | n.a. | 1.229 | n.a. | n.a. | 42.6% | n.a. |

- As a first observation, the naive models performed very similar to the underlying OMXS30GI. The Naive_ESG model slightly outperformed the Naive_NO_ESG model on every metric.

- Regarding the SVM-models, the NO_ESG and ESG_MOM attribute profiles performed the best with NO_ESG having the best SR and ESG_MOM having the best Sortino ratio and gross returns. ESG_ALL performed the worst, underperforming both OMXS30GI and the naive-models.

- Regarding the random forests-models, the ESG_SMA profile outperformed every other model, including SVM and LSTM models, on all metrics. We furthermore observe a low min-max spread for the ESG_SMA profile compared to, for example, the ESG_ALL profile, indicating less variance between instances. The NO_ESG and ESG_BASE profiles of the random forests did not perform well and underperformed both OMXS30GI and their respective naive comparisons.

- Looking at the LSTM-models, the ESG_SMA profile once again dominates. It outperforms the other LSTM-models on every metric. Similar to the random forest case, it also display a low min-max spread. The ESG_MOM and ESG_ALL profiles of the LSTM did not perform well and underperformed both OMXS30GI and their respective naive comparisons.

## 5.2 Cumulative gross returns

To provide some intuitive information regarding the performance of the models, charts documenting the development of the cumulative gross returns are shown in this section. The development of the OMXS30GI is also shown for comparison. For the random forests- and the LSTM-models, only the final instance is presented.

*Figure 5.1: Development of the cumulative gross returns: SVM-models*

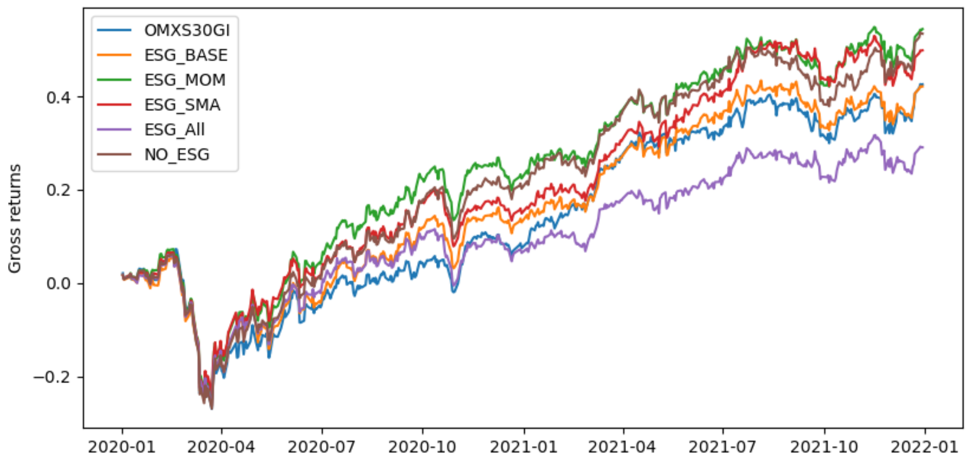


*Figure 5.2: Development of the cumulative gross returns: Instance 10/10 of the random forests-models*

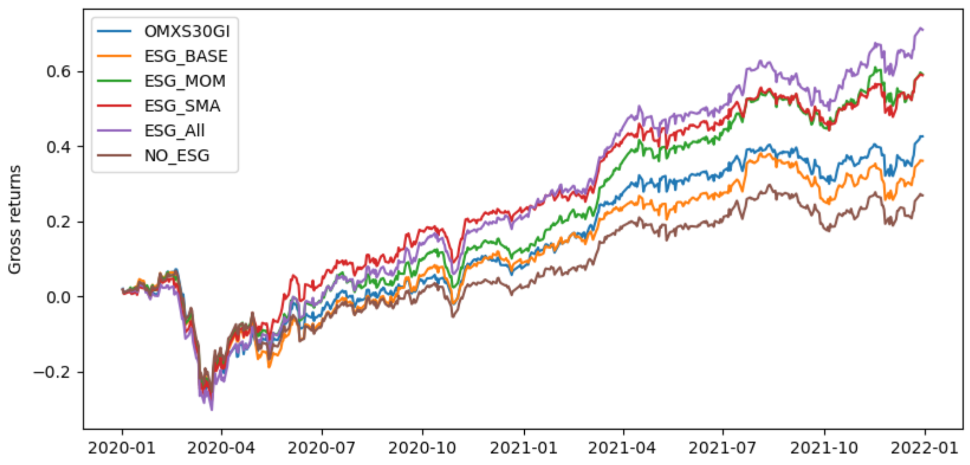


*Figure 5.3: Development of the cumulative gross returns: Instance 10/10 of the LSTM-models*

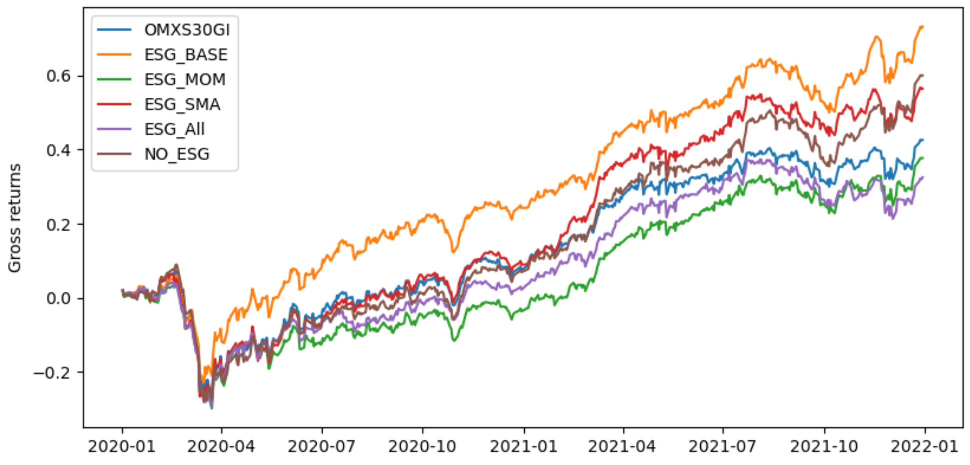
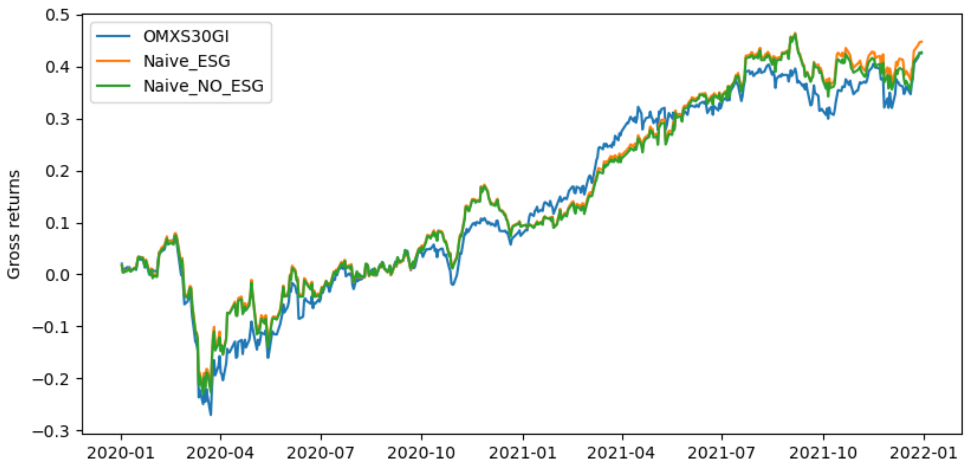


*Figure 5.4: Development of the cumulative gross returns: Naive-models*



# 6 Discussion

## 6.1 Discussion of results

When observing the results from section 5.1 one can draw some conclusions:

1. *The ESG_SMA attribute profile was the best performing profile*: This profile was used in the best performing of all models (RF_ESG_SMA) and was also the profile that led to the best results among the LSTM-models. For the SVM, the profile was not the best but it still outperformed both the naive predictors and OMXS30GI on every benchmark metric. Lastly, the ESG_SMA LSTM and random forest models displayed low min-max spreads on the benchmarks, indicating better prediction potential. The good performance of the ESG_SMA profile compared to the NO_ESG profile also indicate (but does not prove) predictive power of the Sanctify ESG-scores on stock price movements.

2. *Stock prediction is very complicated*: The difference in performance among the models was very high, and some attribute profiles performed well with certain model types and bad with other ones (ESG_MOM for example). Furthermore, the min-max spreads of some models were alarmingly high. As an example, the max benchmarks of the LSTM_ESG_BASE model are around twice as high as the min ones. This implies that the outcome of the random assignment of weights in the neural network carries significant influence on the benchmark results. This should not be the case if the model has high predictive power so one can probably conclude that the predictive power is low. It is possible that a higher number of epochs while training could alleviate this issue but when studying the loss functions we found very little reduction in loss when using more than 50 epochs. Also, the fact that the high min-max spread is present even in some random forest models suggest that the issue is larger than the LSTM-models epoch choice. A more likely explanation is that stock prediction is very complicated and that most models have quite low predictive power. This would give the random assignment of LSTM weights and the bagging procedure of the random forests larger influence compared to models with high predictive power. This is to be expected however. If stock prediction through machine learning was easy, every data scientist would be rich.

3. *Excess returns were achieved*: Somewhat contradictionary to the last conclusion, excess returns compared to the OMXS30GI was still achieved in most

cases. 9/12 of the non-naive models with access to ESG-data had larger mean SRs compared to the OMXS30GI and 8/12 had larger mean Sortino ratios. If we look at the two best performing models, the RF_ESG_SMA and the LSTM_ESG_SMA they outperform the index and provide excess return in every one of their 10 instances.

## 6.2 Future work

### 6.2.1 Investment horizons and holding periods

The choice to rebalance the model portfolios every trading day is explained in section 1.4. While this is optimal in a theoretical environment with zero transaction cost nor any legal or administrative hurdles, it is not a realistic trading strategy. To analyse the effect of more realistic strategies, new trading algorithms that utilizes holding periods could be developed. Ideally the model should weigh the incurred financial and non-financial costs of rebalancing the portfolio with the potential gains from doing so.

### 6.2.2 New investment universe

As mentioned in section 2.3, the only stocks modelled and traded in this thesis was stock stocks present in the OMXS30 index during the chosen time period. In future work, expanding this investment universe could be of significant interest. Testing the models on a larger set of stocks could provide larger confidence in their predictive power and would also allow for exploring if the pricing dynamics varies between exchanges in different countries. Furthermore, it would be interesting to compare the predictive power of Sanctify scores on large versus small stocks. Small companies naturally induce less media coverage to analyse but the coverage that exist could very well generate larger swings in stock prices (at least if the author's intuition is correct). A start could be extending the market universe to the entire Swedish market, instead of limiting it to OMXS30 companies. Later one could try and expand the analysis to new exchanges to eventually generate more statistically significant conclusion that applies worldwide. Further research could also choose other train-test splits and evaluate if the results in this thesis can be replicated in different market environments.

### 6.2.3 Sanctify ESG-scores

As mentioned in section 2.5, Sanctify ESG scores exist in many types, and this thesis has only explored the use of one of these. In further analysis, other score types could be used in prediction and various types could even be used at the same time. For example, one could try and model only on environmental scores or combine short term governmental scores with long term social scores, the possibilities are endless.

### 6.2.4 Other model-types

Four different models were utilised and evaluated in this thesis. Performance was generally sufficient for the task and excess returns were achieved in many cases.

Further research could expand this by utilizing other predictor-models, for example Convolutional Neural Networks (CNN). Further research into investment horizons and holding periods (section 6.2.1) could also include classification rather than regression models. Another method to explore would be to create multi-output models that model many stocks in one model instead of predicting each stock price in a separate model.

### 6.2.5 Correlation between Sanctify ESG-scores and stock price movement

Sanctify ESG-scores have been utilized as attributes in most of the models presented in this thesis and it has been shown that these scores can improve the prediction of stock prices in 2020-2021. What has not been studied however, is the relation between ESG-scores and stock price movement. In future work, one could try and and quantify this relation and search for patterns between, for example:

- Movements in ESG-scores and the effect on $n$-day stock returns.
- Correlation between relative ESG-scores, i.e. the ESG-scores comparable between companies, and stock performance.
- Analysis of the effect of ESG-scores on stock returns in different market environments. For example, the market environment in late 2020 was generally favorable for companies with a high focus on ESG, while late 2021 and early 2022 present an opposite environment.

# Bibliography

[1] Pattern Recognition and Signal Analysis in Medical Imaging. In Anke Meyer-Baese and Volker Schmid, editors, *Pattern Recognition and Signal Analysis in Medical Imaging (Second Edition)*, pages 197–243. Academic Press, Oxford, January 2014.

[2] George O Aragon and Wayne E Ferson. *Portfolio performance evaluation*. Now Publishers Inc, 2007.

[3] Stevan Bajic and Burcin Yurtoglu. Which aspects of csr predict firm market value? *Journal of Capital Markets Studies*, 2018.

[4] Alessandro Beber and Marco Pagano. Short-Selling Bans Around the World: Evidence from the 2007–09 Crisis. *The Journal of Finance*, 68(1):343–381, 2013.

[5] Bloomberg. ESG by the Numbers: Sustainable Investing Set Records in 2021 . https://www.bloomberg.com/news/articles/2022-02-03/esg-by-the-numbers-sustainable-investing-set-records-in-2021/, 2022. Accessed: 2022-06-04.

[6] Lawrence Blume, David Easley, and Maureen O'hara. Market statistics and technical analysis: The role of volume. *The journal of finance*, 49(1):153–181, 1994.

[7] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[8] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[9] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[10] Qian Chen and Xiao-Yang Liu. Quantifying esg alpha using scholar big data: an automated machine learning approach. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

[11] Christopher Olah. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2022-05-24.

[12] Robert DiPietro and Gregory D. Hager. Chapter 21 - Deep learning: RNNs and LSTM. In S. Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger, editors, *Handbook of Medical Image Computing and Computer Assisted Intervention*,

The Elsevier and MICCAI Society Book Series, pages 503–519. Academic Press, January 2020.

[13] Matthew F. Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance: From Theory to Practice*. Springer International Publishing, Cham, 2020.

[14] David N Dreman and Michael A Berry. Overreaction, underreaction, and the low-p/e effect. *Financial Analysts Journal*, 51(4):21–30, 1995.

[15] Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

[16] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

[17] Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

[18] Tristan Fletcher. Support Vector Machines Explained.

[19] Milton Friedman. The Social Responsibility of Business Is to Increase Its Profits. In Walther Ch Zimmerli, Markus Holzinger, and Klaus Richter, editors, *Corporate Ethics and Corporate Governance*, pages 173–178. Springer, Berlin, Heidelberg, 2007.

[20] Jiawei Han, Micheline Kamber, and Jian Pei. 8 - Classification: Basic Concepts. In Jiawei Han, Micheline Kamber, and Jian Pei, editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 327–391. Morgan Kaufmann, Boston, January 2012.

[21] Jiawei Han, Micheline Kamber, and Jian Pei. 9 - Classification: Advanced Methods. In Jiawei Han, Micheline Kamber, and Jian Pei, editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 393–442. Morgan Kaufmann, Boston, January 2012.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. Conference Name: Neural Computation.

[23] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[24] Jennifer Howard-Grenville. Esg impact is hard to measure-but it's not impossible. 2021.

[25] Andreas Jakobsson. *An introduction to time series modeling*. Studentlitteratur, 2019.

[26] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, July 2015.

[27] Tim Koller, Robin Nuttall, and Witold Henisz. Five ways that esg creates value. *The McKinsey Quarterly*, 2019.

[28] John Kordonis, Symeon Symeonidis, and Avi Arampatzis. Stock price forecasting via sentiment analysis on twitter. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, pages 1–6, 2016.

[29] Miroslav Kubat. *An Introduction to Machine Learning*. Springer International Publishing, Cham, 2021.

[30] Kaan Kuguoglu. Building rnn, lstm, and gru for time series using pytorch. https://towardsdatascience.com/building-rnn-lstm-and-gru-for-time-series-using-pytorch-a46e5b094e7b. Accessed: 2022-04-06.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[32] Hao Li, Yanyan Shen, and Yanmin Zhu. Stock Price Prediction Using Attention-based Multi-Input LSTM. In *Proceedings of The 10th Asian Conference on Machine Learning*, pages 454–469. PMLR, November 2018. ISSN: 2640-3498.

[33] Xiaodong Li, Pangjing Wu, and Wenpeng Wang. Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. *Information Processing & Management*, 57(5):102212, September 2020.

[34] Andrew Lo. The Statistics of Sharpe Ratios. *Financial Analysts Journal*, 58, February 2003.

[35] Wei-Yin Loh. Classification and regression trees. *WIREs Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[36] Marcos Lopez de Prado. Beyond Econometrics: A Roadmap Towards Financial Machine Learning. SSRN Scholarly Paper 3365282, Social Science Research Network, Rochester, NY, September 2019.

[37] Nasdaq. NASDAQ OMX STOCKHOLM 30 INDEX. https://indexes.nasdaqomx.com/docs/methodology_OMXS30.pdf. Accessed: 2022-04-13.

[38] Nasdaq. Weightings values for DRBG . https://www.nasdaq.com/docs/2022/05/05/OMXS30_(1).pdf, 2022. Accessed: 2022-06-18.

[39] Andreas Nilsson and David T. Robinson. What Is the Business of Business? *Innovation Policy and the Economy*, 18:79–106, January 2018. Publisher: The University of Chicago Press.

[40] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, January 2015.

[41] Raffaele Pugliese, Stefano Regondi, and Riccardo Marini. Machine learning-based approach: Global trends, research directions, and regulatory standpoints. *Data Science and Management*, 2021.

[42] Scikit learn. sklearn.ensemble.RandomForestRegressor. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Accessed: 2022-05-19.

[43] Scikit learn. sklearn.svm.SVR. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html. Accessed: 2022-05-19.

[44] Sreelekshmy Selvin, R Vinayakumar, E. A Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, September 2017.

[45] William F. Sharpe. Mutual Fund Performance. *The Journal of Business*, 39(1):119–138, 1966. Publisher: University of Chicago Press.

[46] Frank A Sortino and Robert Van Der Meer. Downside risk. *Journal of portfolio Management*, 17(4):27, 1991.

[47] Matheus Gomes Sousa, Kenzo Sakiyama, Lucas de Souza Rodrigues, Pedro Henrique Moraes, Eraldo Rezende Fernandes, and Edson Takashi Matsubara. Bert for stock market sentiment analysis. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1597–1601. IEEE, 2019.

[48] Hong Sul, Alan Dennis, and Lingyao Yuan. Trading on Twitter: Using Social Media Sentiment to Predict Stock Returns: Trading on Twitter. *Decision Sciences*, 48, June 2016.

[49] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[50] George Udny Yule. On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London Series A*, 226:267–298, 1927.

[51] Mehar Vijh, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar. Stock Closing Price Prediction using Machine Learning Techniques. *Procedia Computer Science*, 167:599–606, January 2020.

[52] Gilbert Thomas Walker. On periodicity in series of related terms. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 131(818):518–532, 1931.

[53] Yahoo. What is the adjusted close? https://help.yahoo.com/kb/SLN28256.html. Accessed: 2021-04-15.

Master's Theses in Mathematical Sciences 2022:E38
ISSN 1404-6342

LUTFMS-3443-2022

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/