UNIVERSITY OF TURKU

# ABSTRACT

| | |
|---|---|
| | Bachelor's thesis |
| x | Master's thesis |
| | Licentiate's thesis |
| | Doctoral dissertation |

| | | | |
|---|---|---|---|
| Subject | Economics | Date | 1.6.2020 |
| Author(s) | Joonas Pajukoski | Student number | 508083 |
| | | Number of pages | 84 |
| Title | Predicting credit rating change using machine learning and natural language processing | | |
| Supervisor(s) | Ph.D. Janne Tukiainen | | |

Abstract

Corporate credit ratings provide standardized third-party information for market participants. They offer many benefits for issuers, intermediaries and investors and generally increase trust and efficiency in the market. Credit ratings are provided by credit rating agencies. In addition to quantitative information of companies (e.g. financial statements), the qualitative information in company-related textual documents is known to be a determinant in the credit rating process. However, the way in which the credit rating agencies interpret this data is not public information.

The purpose of this thesis is to develop a supervised machine learning model that predicts credit rating changes as a binary classification problem, based on form 10-k annual reports of public U.S. companies. Before using in the classification task, the form 10-k reports are preprocessed using natural language processing methods. More generally, this thesis aims to answer, to what extent a change in a company's credit rating can be predicted based on the form 10-k reports, and whether the use of topic modeling can improve the results. A total of five different machine learning algorithms are used for the binary classification of this thesis and their performances are compared. These algorithms are support vector machine, logistic regression, decision tree, random forest and naïve Bayes classifier. Topic modeling is implemented using latent semantic analysis.

The studies of Hajek et al. (2016) and Chen et al. (2017) are the main sources of inspiration for this thesis. The methods used in this thesis are for the most part similar as in these studies. This thesis adds value to the findings of these studies by finding out how credit rating prediction methods in Hajek et al. (2016), binary classification methods in Chen et al. (2017) and utilization of form 10-k annual reports (used in both Hajek et al. (2016) and Chen et al. (2017) can be combined as a binary credit rating classifier.

The results of the study show that credit rating change can be predicted using 10-k data, but the predictions are not very accurate. The best classification results were obtained using a support vector machine, with an accuracy of 69.4% and an AUC of 0.6744. No significant improvement on classification performance was obtained using topic modeling.

| | |
|---|---|
| Key words | Machine learning, natural language processing, credit rating prediction |
| Further information | - |

TURUN YLIOPISTO

# TIIVISTELMÄ

| | |
|---|---|
| | Kandidaatintutkielma |
| x | Pro gradu -tutkielma |
| | Lisensiaatintutkielma |
| | Väitöskirja |

| | | | |
|---|---|---|---|
| Oppiaine | Taloustiede | Päivämäärä | 1.6.2020 |
| Tekijä(t) | Joonas Pajukoski | Matrikkelinumero | 508083 |
| | | Sivumäärä | 84 |
| Otsikko | Predicting credit rating change using machine learning and natural language processing | | |
| Ohjaaja(t) | VTT Janne Tukiainen | | |


Tiivistelmä

Yritysten luottoluokitukset antavat standardoitua kolmannen osapuolen tietoa markkinaosapuolille. Ne tarjoavat monia etuja liikkeellelaskijoille, välittäjille ja sijoittajille ja lisäävät yleistä luottamusta ja tehokkuutta markkinoilla. Luottoluokituksia myöntävät luottoluokituslaitokset. Kvantitatiivisten yritystä koskevien tietojen (esim. Tilinpäätöstietojen) lisäksi yrityksen julkaiseman tekstimuotoisen datan sisältävien laadullisten tietojen tiedetään vaikuttavan luottoluokitusprosessiin. Tapa, jolla luottoluokituslaitokset tulkitsevat tätä tietoa, ei kuitenkaan ole julkisesti tiedossa.

Tämän tutkielman tarkoituksena on kehittää ohjattu koneoppimismalli, joka ennustaa luottoluokitusmuutoksia binäärisenä luokitteluongelmana Yhdysvalloissa toimivien pörssiyhtiöiden 10-k -muotoisten vuosikertomuksien perusteella. 10-k vuosikertomukset esikäsitellään luonnollisen kielen käsittelyn menetelmillä, ennen kuin niitä käytetään luokittelutehtävässä. Yleisemmin tämän tutkielman tavoitteena on selvittää, missä määrin yrityksen luottoluokituksen muutosta voidaan ennustaa 10-k vuosikertomuksen perusteella ja voidaanko aihemallinnuksen avulla parantaa tuloksia. Tutkielmassa käytetään binääriseen luokitteluun yhteensä viittä erilaista koneoppimisalgoritmia ja verrataan niiden suorituskykyjä. Nämä algoritmit ovat tukivektorikone, logistinen regressio, päätöspuu, satunnainen metsä ja naïve Bayes-luokitin. Aihemallinnus toteutetaan latentin semanttisen analyysin avulla.

Hajek ym. (2016) ja Chen ym. (2017) tutkimukset ovat toimineet pääasiallisena inspiraation lähteenä tälle tutkielmalle. Tässä tutkielmassa käytetyt metodit ovat pitkälti samoja kuin näissä tutkimuksissa. Tämä tutkielma tuo lisäarvoa näiden tutkimusten tuloksiin selvittämällä, kuinka Hajek ym. (2016) käyttämiä luottoluokituksen ennustusmetodeja, Chen ym. (2017) käyttämiä binäärisen luokittelun metodeja ja 10-k vuosikertomusten hyödyntämistä (käytetty sekä Hajek ym. (2016) että Chen ym. (2017)) voidaan yhdistää binääriseksi luottoluokitusennustimeksi.

Tutkielman tulokset osoittavat, että luottoluokituksen muutosta voidaan ennustaa käyttämällä 10-k vuosikertomuksia, mutta ennusteet eivät ole kovin tarkkoja. Paras luokittelutulos saatiin tukivektorikoneella, tarkkuudella 69,4% ja AUC-arvolla 0,6744. Aihemallinnuksella ei saavutettu merkittävää parannusta luokittelutuloksiin.


| | |
|---|---|
| Avainsanat | Koneoppiminen, luonnollisen kielen käsittely, luottoluokitusmuutoksen ennustaminen |
| Muita tietoja | - |

UNIVERSITY
OF TURKU
Turku School of
Economics

# PREDICTING CREDIT RATING CHANGE USING MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING

Master's Thesis
in economics

Author(s):
Joonas Pajukoski

Supervisor(s):
Ph.D. Janne Tukiainen

1.6.2020
Turku

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| **ML** | **Machine learning** |
| **NLP** | **Natural language processing** |
| **BOW** | **Bag-of-words** |
| **SMOTE** | **Synthetic Minority Over-sampling Technique** |
| **ROC** | **Receiver operator characteristics** |
| **AUC** | **Area under the curve** |
| **POS** | **Part-of-speech** |
| **TD** | **Term-document** |
| **TF-IDF** | **Term Frequency – Inverse Document Frequency** |
| **LSA** | **Latent semantic analysis** |
| **SVD** | **Singular value decomposition** |
| **SVM** | **Support vector machine** |
| **KNN** | **K-nearest neighbors** |
| **ANN** | **Artificial neural network** |
| **SEC** | **U.S. Securities and Exchange Commission** |
| **IBDR** | **Integrated binary discriminant rule** |
| **MNB** | **Multinomial naïve Bayes** |

# 1 INTRODUCTION

Over the last few years, there has been a growing interest in implementing machine learning (ML) methods for financial forecasting purposes. Numerous studies have been conducted on the prediction capabilities of ML methods, for example, in stock price movements and corporate bankruptcies. Another popular prediction task concerning the financial sector is corporate credit ratings. Credit rating represents the creditworthiness[1] of a company. Thus, it provides crucial information to the company's stakeholders. The ability to predict credit ratings makes it possible to detect the problems of the company of interest at an early stage.

Credit ratings are issued by credit rating agencies. These agencies openly report on their websites the factors on the basis of which they issue the ratings. However, the reported factors include only those that can be measured quantitatively. Among the variety of information, financial ratios available in financial statements are the most important factors determining a company's credit rating. Therefore, credit ratings can be predicted quite accurately based on them alone. Nevertheless, it is known that qualitative factors are also considered in the process. The exact methods used by credit rating agencies to utilize qualitative information from textual data published by companies are not publicly known.

Hajek et al. (2016) developed a methodology to address this issue. They extracted topical content from form 10-k annual reports and examined how it could be utilized to predict credit ratings, using some popular ML algorithms. Therefore, the task at hand was solved as a multi-class prediction problem. The obtained predictions were combined with the more traditional approach, where financial ratios are used as predictor variables. The study suggested that in the 10-k reports, there might be some hidden information useful for credit rating prediction. Chen et al. (2017), in turn, used topical content of 10-k reports to predict bank failures. They implemented the prediction as a binary classification task to try to determine whether the bank will default in the next period or not.

Utilizing ideas from both of these studies, the aim of this thesis is to develop a methodology to predict credit rating changes as a binary classification problem. First, using 10-k reports and credit ratings from corresponding years (from Standard & Poor's), the

---

[1] Creditworthiness means the company's ability to meet its payable commitments (Hajek et al., 2016).

ability to predict a change in credit rating is examined using a bag-of-words (BOW) approach. The BOW is a widely used natural language processing (NLP) method, in which individual words and combinations of words in text documents are used as predictor variables. After the examination of the BOW approach, it is examined whether the use of topic modeling improves the prediction performance or not. Both Hajek et al. (2016) and Chen et al. (2017) state that the BOW approach tends to overfit the data, yielding therefore a limited prediction performance. Thus, one could assume that the prediction results can be improved utilizing the topic modeling methods used by Hajek et al. (2016) and Chen et al. (2017).

A total of five widely used ML algorithms are trained for the prediction task, and their performances are compared. This way, a predictive model, designed to add value to another model outside of this work, is construct. Ultimately, the constructed model is able to output the probability that the credit rating of the company under consideration will either decrease or increase. The prediction model constructed in this study can be applied to any 10-k report that has not been used to train the model. In this way, the possible credit rating change of a company of interest can be assessed on the basis of a 10-k report. Thus, the constructed model provides an estimate of the development of the company's financial condition in the near future.

At a more general level, this thesis answers the following research questions:

1. To what extent can a company's credit rating change be predicted utilizing the textual data of the previous period's form 10-k annual report?
2. Can the prediction performance be improved using topic modeling?

The theoretical framework of this study provides first a quick overview of corporate credit ratings; what they are and why they are important. After that, a more detailed review of ML is given in section 2.2. It includes a theoretical definition of ML in general, a breakdown of the similarities and differences between ML and statistics, an overview of the more practical features of ML and some metrics to evaluate the learning process. In addition to ML, another important theoretical entity regarding this thesis is natural language processing (NLP). A few essential NLP methods are described in section 2.3. After these a bit more technical sections, some previous studies related to this thesis are reviewed in section 3.

Section 4 gives a description of the data and methods that were utilized in this thesis. The section explains what kind of data was used, and how it was processed before use. A total of five different ML algorithms are used in this thesis to predict corporate credit ratings. Their basic characteristics are described in section 4.7.

The results, obtained using the theoretical framework and methods described in sections 2 and 4, are presented in section 5. The last section concludes the findings of the study and proposes recommendations for future research on this topic.

# 2 THEORETICAL BACKGROUND

## 2.1 Corporate credit rating

The purpose of credit rating is to provide standardized, forward-looking third-party information for market participants. They comprehensively increase trust and thus efficiency in the market. Credit ratings are used by issuers, intermediaries and investors. For issuers, for example companies, credit ratings allow the company to provide reliable information about its own creditworthiness and financial strength to the market. This is an effective way to increase operational transparency and reliability to current investors, and also to widen the pool of potential future investors, which may reduce the cost of funding. Generally, for a company's stakeholders, credit rating indicates the financial condition of the company. Credit ratings play an important role in, for example, the decisions concerning a company's capital structure. Credit ratings also allow companies to obtain more favorable terms of financing. Similarly, the financiers of a company can monitor a company's solvency through loan covenants in which a credit rating acts as a trigger. In addition to companies, credit ratings are assigned also for other issuers of specific types debt, for example nations and local governments. (Spglobal.com, Shin et al. 2012.)

For intermediaries, for example investment banks, credit ratings improve the flow of money from investors to issuers. They can be used to benchmark the relative credit risk of different debt issues; to compare the creditworthiness of companies in an objective fashion. The initial pricing for individual debt issues and the interest rate they pay can be set on the basis of credit ratings. (Spglobal.com.)

Credit ratings provide a third-party opinion of credit quality for investors. Investors, for example pension funds, can use credit ratings to assess credit risk. Credit ratings allow comparing different issuers and debt issues, when making investment decisions and managing portfolios. Credit ratings also provide information and metrics to make informed decisions. For investors, this may be supplementing their own credit analysis or establishing thresholds for credit risk and investment guideline. (Spglobal.com.)

Credit ratings vary usually on a scale from Aaa/AAA to D, where Aaa/AAA represents the highest possible creditworthiness and D (default) is assigned in the case of bankruptcy. Figure 1. shows the credit rating classes of Standard & Poor's. The figure demonstrates how the creditworthiness of a company decreases, as the rating class approaches the bottom of the table. Rating classes from AAA to BBB- are often considered as the

investment classes and rating classes from BB+ to D as the non-investment classes, indicating how trustworthy a company is as an investment. (Spglobal.com, Hajek et al. 2016.)

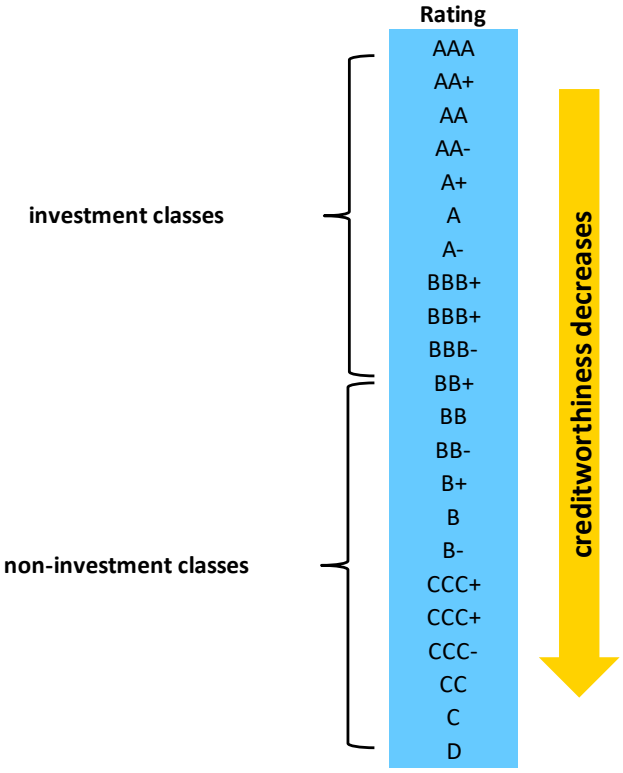


**Figure 1. Standard & Poor's credit rating classes (Adapted from Spglobal.com).**

Credit ratings are issued mainly in two types, short-term and long-term ratings. As the names imply, short-term ratings describe short-term solvency and long-term ratings describe long-term solvency. The credit ratings used in this study are long-term ratings. (Moodys.com.)

Credit ratings are provided by credit rating agencies (eg. Standard & Poor's, Moody's or Fitch), who require a variety of company-related information to complete the rating process. Among the factors affecting the credit rating are, for example, financial statements, conference calls, management interviews, corporate annual & quarterly reports etc. As stated in the introduction, quantitatively measurable credit rating determinants are openly reported by credit rating agencies, and they are usually the most important factors, when determining a credit rating. For example, certain values of financial ratios are always associated with certain probabilities of bankruptcy, and thus certain credit ratings. However, a variety of textual data is also considered in the process. This company-related qualitative information is evaluated carefully by the credit rating agencies, for it is assumed to contain important information about the financial condition of the company and

its future. All in all, the corporate credit rating process is very expertise and time-demanding, which for its part explains the growing interest in simulating the process through ML methods. (Hajek et al. 2016.)

## 2.2 Machine learning

Machine learning is a subfield of artificial intelligence examining intelligent systems that learn. A specific feature of ML, just like of human intelligence, is the ability to learn independently based on experience. As learning enables an artificial agent to become independent of its creator, it can be considered the most fundamental aspect of intelligence. In many occasions, the designer of the agent does not have complete knowledge of the environment where a desirable task is performed. Therefore, the autonomy provided by learning is essential, as it releases the agent from the limited knowledge of the designer and the assumptions built into its original settings. This is one of the relevant differences between traditional programming and ML; an ML model is not dependent on built-in logic. In many implementations, the most successful systems are not constructed by traditional programming, but by a learning process. (Boden 1996.)

Successful actions are what make an agent intelligent (Boden 1996). Thus, an agent cannot be considered intelligent, if it is not able to learn from its mistakes, without being explicitly programmed[2]. Learning from mistakes decreases both the probability of corresponding occasions and the severity of the consequences caused by these mistakes. This is the essence of ML; the ability of an artificial agent to perform tasks successfully in a changing environment. (Russell & Norvig 2009.)

More practically speaking, ML can be defined as a field, which aims to develop algorithms that adapt to exogenous alterations in a desirable way by means of empirical data, experience and training (Kapitanova et al. 2012) or according to Alpaydin (2014), the use of example data or past experience to program a computer to optimize performance criteria. The basic idea of ML is that the machine gradually learns from input data. As the learning process progresses, the ML model improves its performance and automatically learns to make more accurate predictions (if the model is predictive) or to gain better knowledge from data (if the model is descriptive) or both. In an ML model, some

---

[2] Artificial Intelligence pioneer Arthur Samuel defined in 1959 machine learning as a field of study that gives computers the ability to learn without explicit programming (Samuel 1959).

parameters are predefined, and the actual learning process is in practical terms the execution of a computer program and the optimization of the predefined parameters, using past experience or training data. (Alpaydin 2014.)

ML has a lot in common with optimization. In fact, ML problems are often formed as an optimization problem of a loss function on training data. In this case, a loss function points out the difference between the final predicted values of the chosen ML model and the actual problem instances. Classic optimization theory is however utilized only to train the algorithm; to find the parameters that minimize the loss function on training data, whereas in ML, the main focus is in the performance of the model on unseen test data. This is the crucial difference between optimization and ML. (Le Roux et al. 2012.)

### 2.2.1 The relationship between ML and statistics

ML and statistics are nowadays somewhat overlapping terms, yet they are not the same thing. However, there would be no ML without statistics, because the models used in ML are based on statistical theories. (Alpaydin, 2014). The principal goal is what practically makes the difference between ML and statistics. Statistics is traditionally used to make inferences from a sample and to detect causal relationships, while ML is heavily focused on recognizing patterns for forecasting purposes and to achieve forecasting results as good as possible. Generally, statistical methods can be utilized in prediction tasks as well, but predictive accuracy is not their strength. Likewise, some ML models may be interpretable, but the best predictions are obtained with non-interpretable models. (Bzdok et al. 2018.)

To detect some latent mechanism or to find causal connections in data usually requires statistical methods, and these cases tend to be the domain of statistics. However, ML methods enable making accurate predictions, without the need to understand the underlying mechanisms. Hence, the model can be learned from data and the explicit structure of the learning process does not need to be known[3]; the exact way to transform the known input into the desired output may remain hidden. In ML, the real shape of the function is often thought to be so complex that it cannot be defined beforehand. (Bzdok et al. 2018, Breiman 2001.)

---

[3] For this reason, some ML algorithms are often referred as “black box” -algorithms

Although their purpose is what makes the major difference, ML and statistics are separated also by other factors. The use of statistical models in ML tasks often causes confusion. For example, why is a simple linear regression sometimes perceived as ML? In ML, a linear regressor can be trained to obtain the same outcome as a statistical regression model, minimizing the squared error between data points, would yield. The answer lies in the way the performance of the model is evaluated. In ML, the performance is evaluated using a separate subset of data as a "test set[4]", whereas the performance of a statistical model would be determined by the significance and robustness of the model parameters. (Boelaert & Ollion 2018.)

The use of statistical models usually has a strong theoretical basis, whereas the reason why one chooses to use a particular ML method lies primarily in the empirical results. The mechanics of ML algorithms are based on statistical theories, but in practice, algorithms that produce the best prediction results are used. Therefore, ML can be said to be very result-oriented. (Boelaert & Ollion 2018.)

The structure and quantity of the data in use has an influence on which one, statistical or ML method, suits best for the concerning case. When the data is considered "wide" (more input variables than subjects), ML methods are usually effective, whereas statistical methods tend to be more useful, when one is dealing with "long" data (more subjects than input variables). (Bzdok et al. 2018.)

### 2.2.2 Types of learning

There are many types of ML algorithms. They have different approaches and are intended to solve different kinds of tasks or problems. They also use different input data and output different results. These learning methods can be roughly divided into two main groups based on their goal and the data available: supervised learning and unsupervised learning. (James et al. 2013.) Their principal characteristics are described in this section.

The purpose in supervised learning is to create a model that predicts the value of a response variable (output) using explanatory variables (input). In ML vocabulary, explanatory variables are often called features or predictors whereas response variables are called targets or responses. In supervised learning, the required training data is referred

---

4 Section 2.2.2. explains, why a test set is not always used in ML.

as labeled data. This means that for each predictor in the data, a known response is provided. Supervised learning algorithms can predict the labels of new, unseen observations, based on the information learned from the labeled training data. (James et al. 2013.)

Generalization is what makes supervised ML methods useful. Because of it, the trained algorithm provides reasonable responses for inputs that the algorithm has not encountered before. This also makes the algorithm capable of dealing with noise (minor discrepancies in the data). (Marsland 2014.)

Supervised ML methods are most commonly used in classification (categorical predicted value) and regression (continuous predicted value) tasks (James et al. 2013).

In unsupervised learning tasks, there are no labeled training and test data sets. Instead, the learner identifies similarities between the input variables without any training data. The goal is to find structure and provide a summarized or compressed version of the input data. A typical unsupervised learning task is clustering data into smaller subsets consisting of similar objects. (Marsland 2014.)

The results obtained from unsupervised learning tasks are often utilized in increasing the interpretability of the data for further analysis. Two examples of widely used unsupervised learning tasks are dimensionality reduction and grouping. Summaries or compressed versions of the data can be obtained with dimensionality reduction techniques. Feature grouping allows labeling the groups based on their content and then the use of supervised learning methods, or the detection of new relationships between features. (James et al. 2013.)

Supervised and unsupervised learning methods are only the most common types of learning and the ones that are relevant to this thesis. Besides them, there are several other types of ML, for example semi-supervised and reinforcement learning. These methods are mixtures of both supervised and unsupervised learning, and are also widely used, but they are not examined in this thesis. (Marsland 2014.)

### 2.2.3 Train-test -split

In supervised learning, to evaluate the quality of the learning process, the predictions provided by the trained algorithm need to be compared with the known target labels. However, the algorithm needs to generalize to unseen examples, which the algorithm has not encountered in the training data. A separate test set is therefore required to success-

fully evaluate the performance of the algorithm. Practically, this is carried out by detaching some predictor-target pairs from the data and thus forming a new separate dataset. The predicted outputs, that are based on unseen predictors, can thereafter be compared with the known targets. If the algorithm would be tested on the training data, the results would likely be very (positively) biased. The train-test split unfortunately reduces the amount of training data in use, but one just has to accept it, in order to successfully evaluate the learning process. (Marsland 2014.)

### 2.2.4 Overfitting and validation

Overfitting occurs, when an ML model fits the training set "too well". Besides learning the actual function, an overfitting model also learns the noise and inaccuracies from the data. As a result, the model's ability to generalize to unseen data decreases and therefore the prediction accuracy of an overfitting model is usually poor. An overfitting model has high statistical variance. (Kuhn & Johnson 2013.) Boelaert & Ollion (2018) parallel an overfitting model to a student, who learns perfectly all the exact answers given in class but does not understand the actual reasoning behind them, when preparing for a math exam. This strategy would be successful only in the unlikely case that the exam consists of exactly the same exercises seen already in class. If the teacher changes any numbers in the exercises, the student would fail.

Figure 2. shows two different stages of a simplified learning process. The left-hand side of the figure presents a model that fits the training data well. The training error is small, but not zero, since the curve (learned function) goes near the training data points, but not through them. The right-hand side presents, in turn, a situation, where the model becomes too complex as the learning process continues. The model is now matching the training data perfectly, including the inaccuracies and noise in it, instead of only finding the underlying function. The training error of this kind of model is very close to zero; it is overfitting and not able to generalize to unseen examples. (Marsland 2014.)

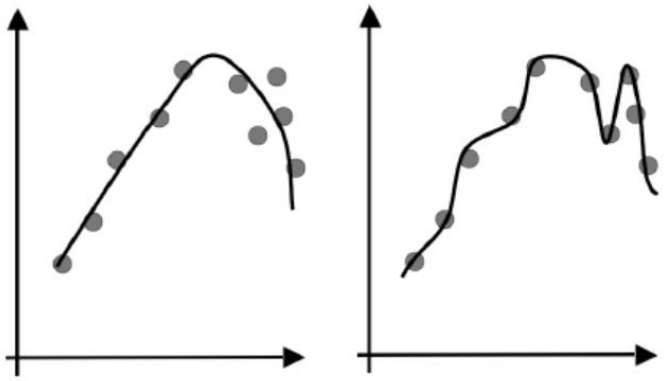

**Figure 2. An example of overfitting (Marsland 2014).**

Along with the training and test sets, a third data set, called the validation set, is required to detect overfitting. As its name suggests, the validation set is used to validate the learning process so far. In other words, validation set is utilized to find out how well the model is generalizing at each moment and to stop the learning process early enough to avoid overfitting. (Marsland 2014.)

If the available data were large enough, it could be divided randomly into k parts and each part could be further divided randomly into two parts. Then, half of each k part could be used for training and half for the model validation. In real world tasks, the data is however rarely sufficient for this. In order to successfully validate the learning process with smaller datasets, a process called cross-validation can be implemented. When using it, the insufficiency of data is compensated by repeatedly splitting the data in a different way. (Alpaydin 2014.)

More specifically, the cross-validation process is commonly carried out using a k-fold cross-validation. In a k-fold cross-validation, the training data is divided into k roughly equal-sized subsets (k is usually 5, 10 or 30). After that, the model is fitted and validated k times, using all the subsets alternately for fitting and validation. For example, if a 30-fold cross-validation was performed, the data would first be divided into 30 subsets. The model would then be fitted using 29 subsets for learning, and the excluded subset would be used as the validation set (to calculate the prediction error). In the next step, the same would be repeated using the second subset as the validation set. This is continued until all 30 subsets have been used as validation sets. This way, the whole data is used for both training and validating the model; to understand how well the model performs the task of learning from some data and predicting some new data. The final performance estimate is obtained from the average of these 30 validations. The validation process is

actually just a way to ensure that the model performs well enough. After the model has been validated, it is trained using all the available training data. An overfitting model typically performs well on the training data but fails when using the test data. (Alpaydin 2014.)

When implementing the k-fold cross-validation technique, in vital importance is to keep the class representations balanced. For example, if class A represents 20 percent of the whole dataset, the subsets drawn from the data as validation sets must also contain approximately 20 percent class A. This is called stratification in ML literature. (Alpaydin 2014, Kuhn & Johnson 2013.)

### 2.2.5 Bias-variance (complexity) trade-off

As stated above, ML models with high complexity may result in overfitting. But what about a model that is too simple for a given task? According to Marsland (2014), too simple model that is not accurate and does not match the training data well, is said to be biased. For example, a simple linear line fitted to the data in Figure 2. would have a very high bias. This kind of model would have no variance at all, but due to the bad fit to the data, bias would in contrast be very high. A highly biased model does not therefore detect the important connections between the input and output variables (the connections are too complex for the model). A model with a high bias is said to underfit the data (Alpaydin 2014). Overfitting and underfitting are hence the two main reasons, why an ML model may in some cases be unsuccessful.

Generally speaking, bias can be decreased by increasing the complexity of the model, and vice versa. Thus, it is easy to train a very biased, or a very complex model. A well-functioning ML model would however have both low enough bias and low enough complexity. This forms a fundamental problem in supervised ML tasks; finding the optimal level of complexity for the model to obtain good results. (James et al. 2013.) The problem is often referred to as the bias-complexity or the bias-variance trade-off.

Hyperparameters are tools that are used to tune the complexity of the model. They are used in almost all modern supervised learning models, and thus they are used also in the empirical part of this thesis. (Probst et al. 2019.)

### 2.2.6 Prediction accuracy and model interpretability trade-off

The level of complexity is closely linked to the prediction capabilities and interpretability of the model. As noted in chapter 2.2.1., ML algorithms are often unable to detect the exact causal connections between input and output variables. Instead, they tend to be so complex, that the explicit learning process is practically unknown. James et al. (2013) state that there is a trade-off between the prediction accuracy and the interpretability of the model. That is, if the prediction accuracy of a model is preferable in a given task, the interpretability in such a case decreases (e.g. support vector machines (SVM), random forests), and vice versa. Of course, besides "black box" algorithms, there are also simpler ML methods that are easier to interpret (e.g. regression models), but as stated, their prediction capabilities tend to be more restricted. Therefore, the right type of algorithm must be chosen based on the intended use.

### 2.2.7 Handling imbalanced classes

A case, where the (training) data consists of precisely balanced classes (e.g. the targets are either positive or negative, and appear in even proportions in the data), is quite rare. This may possibly cause problems in the learning process. (Marsland 2014.) An efficient predictive model would predict both classes[5] accurately, but data imbalance often causes the model to excessively predict the majority class. This results in a relatively large number of minority class misclassifications. (Vluymans 2019.) Therefore, the results obtained using an imbalanced training data tend to be very skewed (Kuhn & Johnson 2013).

There are many ways to handle the class imbalance. These include for example tuning the model to predict the minority class more sensitively; to lower the model's minority class predicting threshold. In addition, sampling methods are widely used and effective (e.g. Van Hulse et al. 2007, Burez & Van den Poel 2009, Jeatrakul et al. 2010) for handling this problem. Instead of tuning the model to deal with imbalance, the idea of sampling methods is to fix the imbalance in the training data. After the training data is balanced, the learning process can be performed normally. (Kuhn & Johnson 2013.)

Sampling methods can be further categorized into oversampling and undersampling techniques and hybrid versions of them. Figure 3. illustrates their fundamental principles. On the left-hand side in Figure 3., the light blue bar represents the majority class, and the

---

[5] Assuming that there are two classes in the data.

dark blue the minority class. Undersampling techniques reduce the size of the majority class to the level of the minority class to fix the imbalance. Oversampling techniques take the opposite approach; the size of the minority class is increased. In oversampling techniques, new elements are added to the minority class based either on duplication or interpolation. Hybrid solutions combine features from both undersampling and oversampling techniques. (Vluymans 2019.) If the training set is resampled to fix the imbalance, the test set should however be more nature-like to obtain reliable results. This means that if there is naturally a class imbalance in the data, the imbalance should not be fixed in the test data. (Kuhn & Johnson 2013.)

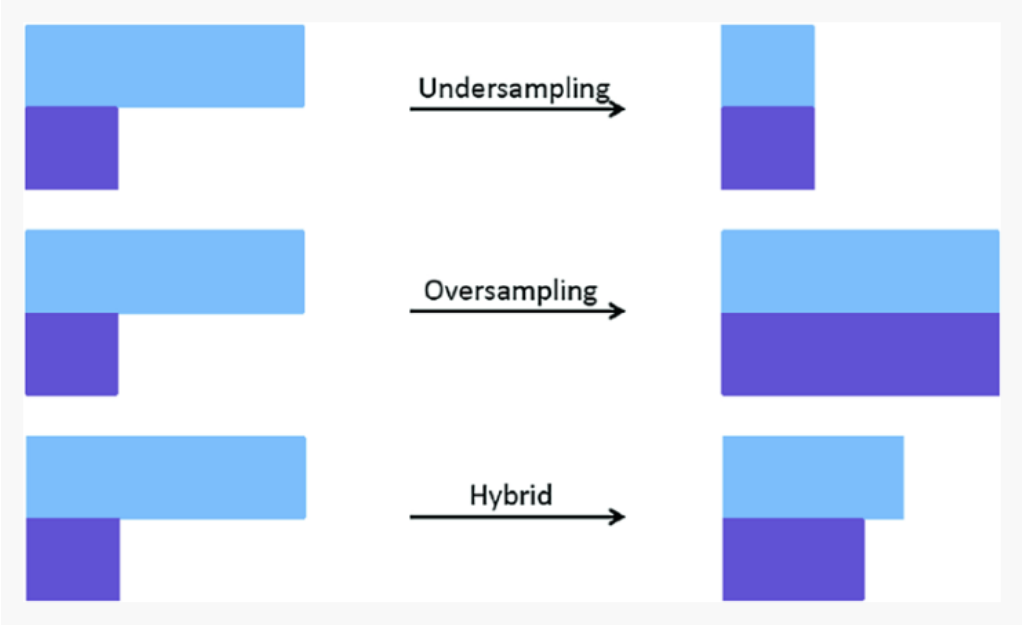


**Figure 3. Sampling techniques (Vluymans 2019).**

Synthetic Minority Oversampling Technique (SMOTE) is the most popular oversampling technique. It reinforces the minority class by generating new synthetic instances using linear interpolation between existing minority class instances and their nearest neighbors. Nearest neighbors mean in this case the data points, that are most similar with a given data point, measured with given features. (Vluymans 2019, Chawla et al. 2002.)

As stated above, sampling methods have been proven to be an effective way to handle the class imbalance. However, according to Kuhn & Johnson (2013), there is no one-above-others approach when it comes to sampling techniques. Being the most popular and easy to implement and interpret, SMOTE was chosen to be used in this thesis.

### 2.2.8 Evaluation metrics

There is a wide range of different ML models, designed to solve a variety of problems. Their functioning and characteristics often differ significantly, making it sometimes hard to evaluate, which one is performing best. In order to objectively evaluate the performance of a model, a consistent set of evaluation tools is required. In this section, some widely used tools for evaluating the performance of binary classification models are introduced[6].

Accuracy is the simplest and perhaps the most used performance metrics. To calculate the accuracy score, the labels predicted by the trained model are compared to the known labels in the test data. Accuracy is thus the percentage of correct predictions made. The simplicity however brings some problems with it. Firstly, accuracy does not tell anything about the type or errors made during the process. In some tasks, this would be an important attribute, for some errors may be more severe than others. Secondly, accuracy does not take class imbalance into account. For example, if there are 50 instances of negative class and 5000 instances of positive class in the data, the model would perform almost perfectly, if measuring only with accuracy. This happens, because almost all instances represent the positive class in the data. The model would then predict almost all of them to be positive, and therefore predict almost all of them correctly. Accuracy score would be high in this case, but little would be known about the performance on predicting the negative class. (Kuhn & Johnson 2013.) Thus, more versatile metrics are needed.

| | Predicted class | | |
|---|---|---|---|
| True class | Positive | Negative | Total |
| Positive | TP: true positive | FN: false negative | p |
| Negative | FP: false positive | TN: true negative | n |
| Total | p´ | n´ | N |

**Figure 4. Confusion matrix (Adapted from Alpaydin 2014).**

Confusion matrix (illustrated in Figure 4.) is a simple chart, where the observed classes can be arranged. It gives an overall picture of the performance of the model. In the matrix, a true positive (TP) is an observation correctly classified into the positive class, a false

[6] The empirical part of this thesis focuses on binary classification. Therefore, it is reasonable to introduce only the evaluation metrics concerning it.

positive (FP) an observation incorrectly classified into the positive class, a true negative (TN) an observation correctly classified into the negative class and a false negative (FN) an observation incorrectly classified into the negative class. P and n represent the total quantities of the true class instances, while p′ and n′ stand for the total quantities of corresponding predictions. N is the total quantity of data. (Marsland 2014.)

Based on the results in the confusion matrix, following measurements can further be determined to help interpreting the performance of the model. Recall (also known as sensitivity and the true positive rate, equation 1) shows the amount of true positive instances, divided by all the positive instances in the data, while precision (equation 2) is the ratio of true positive instances divided by the quantity of all positive predictions made. Together, precision and recall give more information about the performance than accuracy. F1-score (equation 3) can be calculated as the harmonic mean of precision and recall. It gives a more holistic view on the performance of the model; such as accuracy, it does not make a distinction between the error types. (Marsland 2014.)

$$Recall = \frac{TP}{TP+FN} \tag{1}$$

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

$$F1 = 2 \times \frac{precision \times recall}{precision+recall} \tag{3}$$

Using the knowledge from the confusion matrix enables also a visual performance analysis. ROC (receiver operator characteristics) curve, illustrated in Figure 5., consists of true positive rate – false positive rate combinations. It describes the types of errors the classifier makes using different probability thresholds. The probability threshold means how sure the classifier needs to be to predict a certain class. Using different probability thresholds yields data points in the ROC space, and the ROC curve can then be specified through these points. Using the ROC curve analysis, it is possible to detect the types of errors the classifier is prone to make, and hence to tune the classifier to avoid the more severe types of errors. (Kuhn & Johnson 2013). The ideal classification model would draw a ROC curve, that goes through the point (0,1) (0% false positives, 100% true positives). In contrast, the worst possible classifier would have a linear ROC-curve going through point (0,0) and (1,1). (James et al. 2013.) In order to wrap up the ROC curve

analysis to a single number, the AUC (area under the curve) can be calculated. An ideal classifier would have an AUC of 1.0, while the worst possible classifier would have an AUC of 0.5. Generally speaking, the closer the ROC curve is to the upper left corner of the plot in Figure 5., the better the performance of the model. (Alpaydin 2014.)

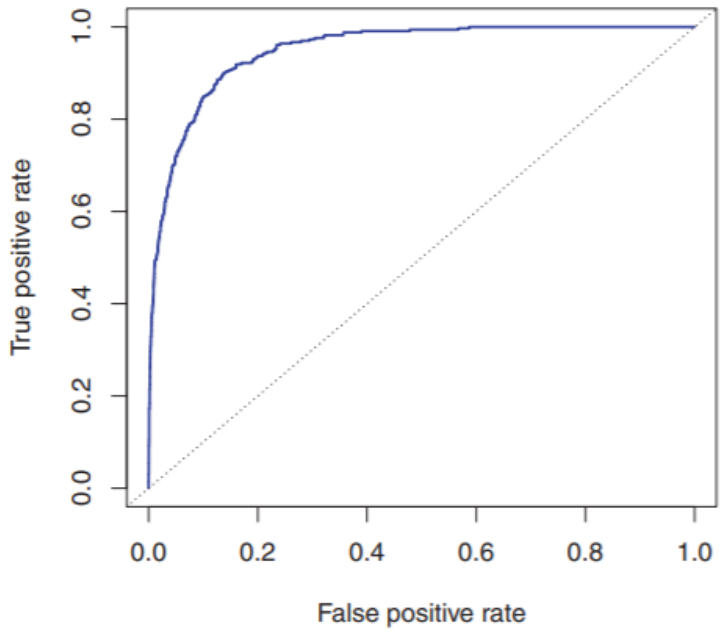


**Figure 5. ROC-curve (James et al. 2013).**

## 2.3 Natural language processing

NLP is a field concerned with technologies that enable computers to work efficiently with human language data. NLP is an interdisciplinary field; it combines techniques of linguistics, artificial intelligence, information engineering and computer science. NLP tasks include enabling computers to understand human language, improving communication between humans and other useful human speech or text processing tasks. The main objective in NLP applications is usually either understanding or producing human language. The focus of this thesis is in the former. (Hirschberg & Manning 2015, Cady 2017, Jurafsky & Martin 2018.)

To enable a computer to understand and analyze human language, the task needs to be split into smaller subtasks, that can be performed using special techniques. Various techniques can be implemented in NLP tasks. The relevant ones to this thesis are introduced in the following sections. An NLP analysis usually starts with text preprocessing.

### 2.3.1 Text preprocessing techniques

Tokenization is usually among the very first steps in most NLP tasks. Basically, tokenization is separating a piece of text into smaller pieces (tokens), that are usually individual words. The separation is carried out using some predefined rules concerning for example spaces and certain punctuation marks. (Cady 2017.) Typically, in English language, a space separates two words from each other. There are however many exceptions (e.g. New York), where two consecutive words need to be treated as one entity (one token), in order the capture the correct meaning. In turn, sometimes an entity containing no spaces, needs to be separated into two tokens (e.g. I'm → I am). It is therefore crucial and sometimes challenging to develop suitable rules for tokenization. (Jurafsky & Martin 2018.)

After the input text has been tokenized, a process called part-of-speech tagging (POS tagging) can be performed. In this process, a POS marker is assigned for each tokenized word in the corpus[7]. Since for most words the POS depends on the situation (e.g. "book" can be a noun or a verb), the POS tagging process can be seen as a disambiguation process; the goal is to find the correct POS in a given context. Depending on the task, POS tagging is yet not always needed. It can be computationally expensive and thus its unnecessary use should be avoided. (Jurafsky & Martin 2018.)

Techniques called stemming and lemmatization can be applied in order to unite different variations of the same word. Their principal goal is similar, but they provide both different approaches and results. In lemmatization, the "lemma" of the word is its base form. For example, "running", "ran" and "runs" are all variations of the lemma "run". Thus, if the text in question is lemmatized, all the aforementioned words are turned into "run". The downside of lemmatization is its computational expense. In order to extract the lemma of the word, one also needs to define the POS for the word. This, as stated above, requires knowledge of the whole sentence surrounding the word. (Cady 2017.)

Stemming, in turn, is a more straightforward technique. It is computationally cheaper and easier to perform, but as a consequence, is usually less precise. The "stem" of the word is simply a part of the word, obtained using predefined rules. Usually these rules strip off certain endings from words (e.g. "er" or "ing"). For example, the stem for words "producer", "producing", "product" and "production", could be for example "produc-".

---

7 A corpus is a collection of texts.

In the “run” example used above, the words “running” and “ran” would be treated as separate words, if stemming was used. (Cady 2017.)

Along with stemming and lemmatization, the words are usually converted into lowercase letters. For example, a word in the beginning of the sentence has its first letter written in uppercase. In further NLP analysis, this word, and the same word with all letters written lowercase, are treated as completely different words, regardless of the fact that they have the exact same meaning. (Uysal & Gunal 2014.) It is also common to remove certain punctuation marks from the text, sometimes all of them (Igual & Seguí 2017).

The removal of “stop words” is often a part of the preprocessing stage. One cannot unambiguously define which words are considered stop words, but generally words that do not contain much information themselves, are considered stop words. In English language, for example, words “and”, “a”, “the”, “it” and “as” are usually treated as stop words, when preprocessing a piece of text. In many implementations, stop words are considered as unnecessary noise only complicating the task, which justifies their removal. The implementer of an NLP analysis can define a list of stop words manually, or use a more general, pre-assembled list. In some occasions, certain words that are often considered stop words, can be very meaningful. The removal of this kind of words will have a negative impact on the analysis. (Cady 2017.)

One preprocessing method worth consideration is the detection of synonyms and similar words. In the data, there usually appears several different words, that have a very similar meaning. Sometimes the analysis can be improved by collapsing all these words into a single identifier, a synset. This method is however problematic, because the meaning of the word often depends on the context, and therefore it can belong to many synsets. (Cady 2017.)

The goal of the above-mentioned techniques is, in essence, to convert the text into a more standard form; to normalize it. These techniques help to get rid of unnecessary and duplicate information. Appropriately used, they simplify the analysis and improve the results. (Jurafsky & Martin 2018). Their use always depends on the final objective of the analysis and the available data. Therefore, there is no exact guideline specifying when and how to use these techniques; usually the optimal solution must be found by experimentation. (Igual & Seguí 2017.)

### 2.3.2 Bag-of-words (BOW) and vectorization

After the textual data has been normalized using the above-introduced techniques, it can be turned into numerical vectors. In a vector form, the textual data is finally ready to be used as an input for an ML algorithm. In most NLP tasks, text documents are represented in a BOW format. This means that the positions of the words are ignored; only their frequency within the text is of interest. Basically, in the vectorization process, the dimension of the document vector is equal to the amount of individual words (or tokens) occurring in the entire corpus. Depending on their frequency, a weight is assigned for each word in the document. Hence, in the BOW concept, the text document can be represented as a vector in a (usually) high-dimensional word-space. Similarly, each word can be seen as a combination of relations to each document. (Jurafsky & Martin 2018, Cady 2017.)

The vectorization process can be visualized as a term-document (TD) matrix (Figure 13.). In Figure 6., the rows represent individual words, and how often they appear in certain documents (columns). For instance, in the document "Julius Caesar", the word "battle" appears 7 times. These vectors are said to be count vectors; the vectors are created simply by counting the word occurrences. Therefore, the weight for each word is obtained by counting its occurrences in the document. (Jurafsky & Martin 2018.)

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 0 | 7 | 13 |
| good | 114 | 80 | 62 | 89 |
| fool | 36 | 58 | 1 | 4 |
| wit | 20 | 15 | 2 | 3 |

**Figure 6. A simple TD matrix (Jurafsky & Martin 2018).**

This approach is however very limited. In most cases, some words are considered more important than others, and usually the most important words are not the most common ones. The count vector approach considers the most common words as the most important, as their weights are the highest. A widely-used approach called TF-IDF (Term Frequency – Inverse Document Frequency) provides a solution for this problem. It basically highlights rare words but lowers the weights for common words. (Cady 2017.) TF-IDF score $w$ for word $t$ in document $d$ is calculated the following way:

$$w_{t,d} = tf_{t,d} \times idf_t \tag{24}$$

Where $tf_{t,d}$ is simply the frequency of the word in the document. It is usually weighted down by taking $log_{10}$ of the frequency. Formally, $tf_{t,d}$ is given by:

$$tf_{t,d} = \begin{cases} 1 + log_{10} count(t,d) & if\ count(t,d) > 0 \\ 0 & otherwise \end{cases} \quad (25)$$

$idf_t$, the inverse document frequency, is given by:

$$idf_t = log_{10}(\frac{N}{df_t}) \quad (26)$$

Where N is the total number of documents in the corpus and $df_t$ tells how many documents the word t occurs in. Thus, words that occur only in a few documents are given a higher weight. To conclude, TF-IDF method assigns high importance for words that occur frequently in one document, but rarely in others. (Jurafsky & Martin 2018.)

Usually in a TF-IDF matrix, a minimum frequency constraint is applied. If a word occurs only very few times in a large document, its importance (TF-IDF score) gets inappropriately high. Thus, only the words that occur at least, for example, five times in a document are included in the matrix. Even a minimum frequency constraint of two usually excludes a lot of noise, for example typos, from the matrix. (Cady 2017.)

Instead of using only individual words, the above-mentioned techniques can be applied also for so-called n-grams. An n-gram is a sequence of words, containing n words (n is the number of words). Often, words together construct a completely different meaning comparing to individual words. The use of n-grams enables the capture of these meanings. In an NLP analysis, an n-gram is treated in the same way as an individual word. (Cady 2017.) Usually only two (bigram) or three (trigram) words long n-grams are used, because longer n-grams would form too large a TD matrix (Jurafsky & Martin 2018).

More generally, the text vectorization process is called feature extraction. That is, because the main goal of the vectorization process is to produce the features, that can be utilized as input data for a selected ML algorithm in the classification process. Usually, the vectors obtained in the vectorization process tend to be very sparse. The sparsity derives from the fact that usually a large number of words are involved in the corpus, but only a small fraction of the words appear in a single document. Therefore, most of the words in a document vector are assigned a weight of zero. A word, that is assigned a

weight of zero in many documents, is however not considered redundant or irrelevant, often quite the contrary. The goal of the above-introduced text preprocessing techniques can be seen as the removal of the redundant and irrelevant features from the analysis; to reduce the dimensionality of the TD matrix. (Elakiya & Rajkumar 2017.)

Even after careful preprocessing, lots of useless features typically remain in the TD matrix. A process called feature selection is implemented to select only the most informative features for the classification stage. A basic approach to feature selection is to define some informativeness measure, rank all the features according to it, and eliminate some fixed amount of the most useless features. (Jurafsky & Martin 2018.)

### 2.3.3 Topic modeling

The dimensionality reduction of the TD matrix can be taken even further using a method called topic modeling. Topic modeling methods are a good example of unsupervised ML. The goal of topic modeling is to group all the words in the corpus into clusters of words, that occur frequently together. The formed clusters are intended to represent semantic or meaningful topics, giving an insight into the topical content of the corpus. In topic modeling, under the BOW assumption, the documents can be seen as a mixture of topics, where a topic is defined as a probability distribution over words. A topic model hence detects similar words in a document, groups them together into topics and outputs a distribution over topics for the document. The topics can be further used as features in an ML model. (Steyvers & Griffiths 2007.)

According to Hajek et al. (2016) there are two general approaches to perform the dimensionality reduction of the TD matrix: latent semantic analysis (LSA) and probabilistic topic models, for example latent dirichlet allocation. Both of these reduce the dimensionality of the TD matrix significantly, and therefore the number of features in the ML model. LSA is not officially an actual topic modeling technique, but it yields a functional and easy-to-interpret topical content analysis (Steyvers & Griffiths 2007, Hajek et al. 2016).

LSA constructs a semantic space of the TD matrix using singular value decomposition (SVD). This allows the detection of the documents that contain similar topics, but different terms[8]. The SVD decomposes the TD matrix $A$ (m x n dimension) into a matrix

[8] The above-introduced text preprocessing technique of removing synonyms and polysemy is partly an overlapping technique with LSA.

of document vectors $V^T$, a matrix of word vectors $U$ and Σ, a matrix of the singular values of $A$. The matrix factorization can be expressed by:

$$A = U\,\Sigma\,V^T \tag{27}$$

where $U$ (m x m dimension, m is the number of terms) and $V^T$ (n x n dimension, n is the number of documents) are orthogonal matrices and Σ (m x n dimension) is a diagonal matrix, where the singular values of the matrix $A$ are the diagonal values. The first r columns of the matrix $U$ (here r is the rank of the matrix) consist of r orthonormal eigenvectors determined by the r nonzero eigenvalues of $AA^T$. The first r columns of the matrix V consist of r orthonormal eigenvectors determined by the r nonzero eigenvalues of $A^TA$. So, in order to compute the SVD, the eigenvalues and eigenvectors of $AA^T$ and $A^TA$ have to be found. Here, the columns of $U$ are the eigenvectors of $AA^T$ and the columns of $V$ are the eigenvectors of $A^TA$. The common positive eigenvalues of $AA^T$ and $A^TA$ are the diagonal values of matrix Σ, and thus the singular values of $A$. Selecting the k largest singular values, and their corresponding singular vectors from $U$ and V yields a best rank k approximation to $A$. As a result, the dimensionality reduction from r to k removes the noise from the original matrix $A$ and captures the latent semantic structures in the data. The topics can be now determined by comparing the cosine similarities between the terms. Terms, that are near each other (have a high similarity) in the reduced k-dimensional space, have a similar meaning. Accordingly, documents, that are near each other share common topics. (Martin & Berry 2007, Hajek et al. 2016.)

# 3 RELATED LITERATURE

Machine learning has received a lot of attention during the last years, when it comes to forecasting purposes in finance. Numerous studies have been conducted on the prediction capabilities of ML methods, for example, in stock price movements and corporate bankruptcies. A state-of-art approach is to pick a few ML algorithms, implement them in the prediction task and compare the obtained results.

For instance, Shynkevich et al. (2017) used SVMs, artificial neural networks (ANN) and K-nearest neighbors (KNN) to predict stock price movements in the short term. The SVM model performed best with an accuracy of 75%, while accuracy obtained with the ANN model was slightly worse, 73%. Clearly the worst-performing model in the study was the KNN, with an accuracy of 60%. In turn, Barboza et al. (2017) achieved an accuracy of 87% in predicting corporate bankruptcies, using a random forest model. It outperformed a more "traditional" logistic regression model, which achieved 69% accuracy.

Another popular prediction task in the financial sector is corporate credit ratings. The effect of financial ratios on credit ratings has been studied quite thoroughly. For example, Wu et al. (2014) used bagging combined with decision trees to achieve a credit rate classification accuracy of 83%, using corporate financial statements as data. In addition, popular methods for this kind of prediction task are, for example, SVMs (Kim & Ahn 2012, Chen & Li 2014), random forests (Yeh et al. 2012) and ANNs (Huang et al. 2004, Wallis et al. 2019).

In contrast to the above-mentioned approaches, the use of hidden information in company-related textual content as predictor variables for credit rating classification is yet a relatively new and little researched concept. Hajek et al. (2016) combined the more traditional approach, where only the financial ratios are used as predictor variables, with textual, topical content extracted from annual reports of U.S. companies (10-k form), for a credit rating classification task. They used TF-IDF to create a TD matrix and LSA to extract the hidden topics from it. Classification was performed using naïve Bayesian network, decision tree, random forest, SVM, multilayer perceptron, logistic regression and KNN. All of these classifiers were implemented on both the financial and the textual indicators, after which their results were combined. Without the financial indicators, the best-performing classifiers were logistic regression (AUC 0.907), SVM (AUC 0.862), multilayer perceptron (AUC 0.761) and naïve Bayesian network (AUC 0.728). Without the textual indicators, the best-performing classifiers were naïve Bayesian network (AUC

0.922), logistic regression (AUC 0.920), random forest (AUC 0.919) and multilayer perceptron (AUC 0.911). When the financial and textual indicators were combined, best classifiers were random forest (AUC 0.925), naïve Bayesian network (AUC 0.924), multilayer perceptron (AUC 0.875) and SVM (AUC 0.870).

Without the financial indicators, the performance of the best-performing classifiers decreased significantly. This demonstrates their importance on credit rating classification tasks. The performance on only the financial indicators is almost as good as with all indicators. However, random forest and naïve Bayesian network, the best-performing classifiers on all indicators decreased in performance, when the textual indicators were not used. This suggests that in the form 10-k annual reports, there might be some latent information, useful for credit rating prediction.

Chen et al. (2017) compared the performance of different topic models and their abilities to predict bank failures. They used annual reports of U.S. companies in 8-k and 10-k forms as data. Bank failure prediction performance on 10-k data was significantly worse than on 8-k data. They state that this derives from the high similarity of companies' reports between consecutive years; the content within a 10-k report is typically mostly copied from the previous years' corresponding document. A two-layer neural network was constructed for classification, achieving its best performance with latent dirichlet allocation and 8-k data.

# 4 DATA AND METHODS USED IN THE STUDY

## 4.1 Form 10-k annual reports

The textual data used in this thesis is form 10-k annual reports of public U.S. companies from years 2000-2018. The U.S. Securities and Exchange Commission (SEC) requires all public U.S. companies to submit annual reports in form 10-k, after the company's fiscal year has ended. The deadline varies depending on the amount of the company's public float (60-90 days after the end of the fiscal year). Form 10-k reports contain plenty of detailed information about the company's latest fiscal year, allowing investors to be aware of the company's financial condition, before investing in it. (Sec.gov.)

The report is standardized and consists of four main sections, divided further into 20 subsections. Being a comprehensive report, a form 10-k is often very large, containing possibly hundreds of pages. (Sec.gov). Because of this, only three subsections, Item 1A (Risk factors), Item 7 (Management's discussion and analysis of financial condition and results of operations) and Item 7A (Quantitative and qualitative disclosures about market risks) were included in the data. These subsections are assumed to contain crucial information about the near future of the company. Therefore, it is assumed that one could predict a possible change in the company's credit rating in the near future, on the basis of this information. In addition, according to Cohen et al. (2020), the content of 10-k annual reports generally varies very little between consecutive years. Chen et al. (2017) state this too; the cosine similarities between 10-k reports of consecutive years were very high. In Cohen et al. (2020), items 1A, 7 and 7A were found to have the most variation between consecutive years. Thus, they could be expected to contain the most predictive value, justifying their selection for this study. Chen et al. (2017) found also that another type of mandatory SEC filing, the form 8-k report, varies much more from year to year. As a consequence, predictions using 8-k data were also better. The use of 8-k data was considered, but in a more standard format, 10-k data was found to be more suitable for this study.

## 4.2 Data preparation

The chosen subsections of form 10-k reports were downloaded from SEC's EDGAR database. Some companies submitted the desired subsections in a particular exhibit file. These files were also included in the data. The data was transferred to a Jupyter notebook,

where the data preparation and ML processes were convenient to perform, using Python programming language. The data preparation was started by deleting pure duplicate rows. Then, if there were two or more instances of the same subsection for a given company for a given year, the shorter one was deleted. Cases, where the content of the subsection referred to another document, was in a form of a data table or was non-informative in some other way, were also deleted. All the subsections for a given company for a given year were then merged, so that for a given company on a given year, the text data included all the available subsections.

The next step was to download public U.S. companies' credit ratings for all those years, when the credit ratings had changed. First, if a company had received two or more ratings during one year, only the last rating of the year was included in the data, assuming that it represents the true creditworthiness of the company on that given year. After that, the rating values written in letters (e.g. AA+, CCC-), were replaced with numbers. Now, the direction of the credit rating change for a company could be determined with a simple subtraction. The credit rating change was then turned into a dummy variable, 1 representing an increase in credit rating and 0 representing a decrease.

After both tables had been prepared, one containing the text documents and other the credit ratings, the two tables were merged. The resulting table consisted of document-rating pairs for a given company for a given year (e.g. Microsoft's documents for year 2004 and Microsoft's credit rating for year 2004). For those documents that could not be "matched" this way, a match was sought from the next years' ratings, assuming that the information in the 10-k report could possible implicate a change in credit rating also one year after the corresponding 10-k report had been submitted. The total number of document-credit rating pairs construct this way was 1337.

### 4.3 Data limitations

After the above-mentioned preparations, the final quantities of the form 10-k subsections were: Item 1A: 1651, Item 7: 13406, Item 7A: 17163 and Exhibits: 2484. The number of Item 1A's is remarkably low, because only the biggest companies are required to submit it. Thus, there are very few rows in the data, where all the subsections are included for a given company for a given year. The study could have been conducted using data including, for example, only item 7A's, but that would have shrunk the already limited data even smaller. In contrast, the amount of textual data was maximized, however, keeping

the data relevant. Although the company-year rows contain slightly different subsections, the textual data is still assumed to contain predictive information.

Generally, in ML tasks, more and better data implicate better results. 1337 is quite a limited number of document-credit rating pairs in this kind of complex prediction task, but given the dataset, it is the best that could be achieved.

### 4.4 Data preprocessing and feature extraction

The textual data was preprocessed utilizing the NLP methods introduced in section 2.4.1. First step was to remove all punctuation marks and to stem the documents. After this, the documents were vectorized using a TF-IDF vectorizer. (TF-IDF took care of the tokenization). TF-IDF was performed also with bigrams and trigrams, so that each individual word, bigram and trigram was treated as a feature. The vectorization yielded 18 581 individual words, 660 980 bigrams and 1 856 305 trigrams. According to the TF-IDF vectorization score, the features were ranked, so that only the most informative features could be included in the final data. In this manner, the performance of different sized feature sets could be considered. Table 1. shows how 12 datasets were formed using combinations of different sized feature sets and n-grams. N-gram range 1 means that only individual words were used as features. N-gram range 1-2 means, that both individual words and bigrams were used as features. N-gram range 1-3 in turn means, that individual words, bigrams and trigrams were all used as features. These datasets therefore consist of preprocessed form 10-k reports that are represented as high-dimensional vectors, where each dimension represents the frequency of a word or an n-gram in the text.

**Table 1. Used datasets**

| | | |
|---|---|---|
| **1000 features n-gram range 1** | **1000 features n-gram range 1-2** | **1000 features n-gram range 1-3** |
| **2000 features n-gram range 1** | **2000 features n-gram range 1-2** | **2000 features n-gram range 1-3** |
| **3000 features n-gram range 1** | **3000 features n-gram range 1-2** | **3000 features n-gram range 1-3** |
| **unlimited features n-gram range 1** | **unlimited features n-gram range 1-2** | **unlimited features n-gram range 1-3** |

## 4.5 Train-test split and validation

In order to evaluate the performance of the construct classification models, the data was split into separate training and test sets. If the data was sufficient, it would have been possible to do, for example, a 60-20-20 split into training, test and validation sets. But quite the opposite, the data was very limited. Due to this, the amount of training data was maximized to train the model as well as possible. However, a decent amount of data was still spared to evaluate the models, so that reliable results could be obtained. Thus, 80% of the data was chosen for the training set and 20% for the test set. No separate validation set was used due to the lack of data, and therefore a 10-fold cross validation was implemented. Stratification was used in the cross-validation process to keep the class ratios unchanged, when picking the validation sets. The classes were imbalanced, as there were 517 rows, where the credit rating had increased, and 820 rows where it had decreased. The class imbalance in the training data was fixed using SMOTE, yielding finally 656 instances of both cases in the training data. These datasets were then ready to be used as inputs in the actual ML process.

## 4.6 Latent semantic analysis (LSA)

In order to evaluate the effect of topic modeling on the classification performance, the 12 different BOW datasets in Table 1., in other words the 12 different TD matrices, were transformed into topic-document matrices. Following the methodology in section 2.3.3., an LSA was implemented using an SVD.

In topic modeling, it is crucial to find the optimal number of topics. The number of topics has an effect on how meaningful the extracted topics are and how well they can be further utilized as input vectors in the ML process. Stevens et al. (2012) find that LSA is not the best possible approach among many topic modeling techniques for extracting meaningful topics, but it yields the best performance in classification tasks. Values of the singular matrix in the SVD, also known as the singular values, act as coherence scores for the extracted topics. Therefore, in LSA, the n most meaningful topics can be extracted by choosing n topics with the highest singular values. However, Stevens et al. (2012) also state that topics with the highest coherence scores are not necessarily best for classification purposes. For this reason, the truly optimal number of topics for classification should be searched exhaustively. Unfortunately, this optimization method requires a lot of computing power, and could thus not be implemented in this study. Hajek et al. (2016) picked

topics with a singular value over 1, when choosing the number of topics for classification. This method is also easy and fast to implement and was therefore a suitable way to optimize the number of topics in this study.

As the name implies, topic modeling helps to understand the main topics in the corpus. This can provide valuable information for some purposes, but in this study, topic modeling is really only used as a dimensionality reduction method. Thus, although the topics used in the classification phase are selected using coherence scoring, the coherence of the topics per se is not really relevant here.

For all 12 datasets in Table 1., words and n-grams were clustered into optimal number of topics, using SVD. Table 2. shows the optimal number of topics in each case.

**Table 2. Optimal number of topics for each dataset.**

| | | |
|---|---|---|
| **1000 features**<br>**n-gram range 1**<br>**167 topics** | **1000 features**<br>**n-gram range 1-2**<br>**156 topics** | **1000 features**<br>**n-gram range 1-3**<br>**154 topics** |
| **2000 features**<br>**n-gram range 1**<br>**192 topics** | **2000 features**<br>**n-gram range 1-2**<br>**184 topics** | **2000 features**<br>**n-gram range 1-3**<br>**184 topics** |
| **3000 features**<br>**n-gram range 1**<br>**201 topics** | **3000 features**<br>**n-gram range 1-2**<br>**202 topics** | **3000 features**<br>**n-gram range 1-3**<br>**198 topics** |
| **unlimited features**<br>**n-gram range 1**<br>**216 topics** | **unlimited features**<br>**n-gram range 1-2**<br>**300 topics** | **unlimited features**<br>**n-gram range 1-3**<br>**350 topics** |

Quite intuitively, as the number of considered features increases, the number of optimal topics, topics with a singular value over 1, increases. This is due to the fact that with limited features, only the most important words and n-grams (highest TF-IDF score) are considered. Therefore, they also form the most meaningful topics with high singular values. However, a very limited number of topics tend to achieve a high singular value; usually only a few meaningful topics can be extracted from a large corpus. Because of this, the singular values tend to pile up on the key topics, and the less meaningful topics are assigned a low singular value. Correspondingly, as the number of considered features increases, the formed key topics lose their coherence, and therefore the singular values are distributed more evenly among all topics.

## 4.7 Machine learning algorithms

As stated in section 2.2.1., the use of ML algorithms is usually not justified by theories. Rather than that, the best-performing algorithm on a given dataset is selected. Therefore, since the most suitable algorithm for the classification task in this thesis cannot be known beforehand, a few of the most popular ML classification algorithms have been chosen for comparison. Their characteristics and the way they are implemented in this thesis are described in the following sections.

### 4.7.1 Support vector machine (SVM)

SVMs are a class of supervised learning techniques that perform efficiently on both linear and non-linear classification tasks. SVMs were originally introduced to solve binary classification problems, though they have later been utilized in multiclass classification, regression and other tasks as well. SVM literature can be divided into two main categories: SVM classification and SVM regression. (Ali, 2008.) Only the former is used in this thesis.

An SVM categorizes the observations based on their mutual distances. In a simple classification task, such as the example in Figure 7., the categorization can be carried out by finding an optimal separating hyperplane between the observations. Figure 7. demonstrates geometrically, that if there exists an unambiguous optimal hyperplane, there exists an infinite number of hyperplanes that divide the observations into two categories. The optimal hyperplane is p-1 dimensional in a p-dimensional space (e.g. in a 4-dimensional space the hyperplane is 3-dimensional[9] and in a 2-dimensional space the hyperplane is a 1-dimensional line). (James et al. 2013.)

[9] In dimensions $p > 3$, the hyperplane is hard to visualize.

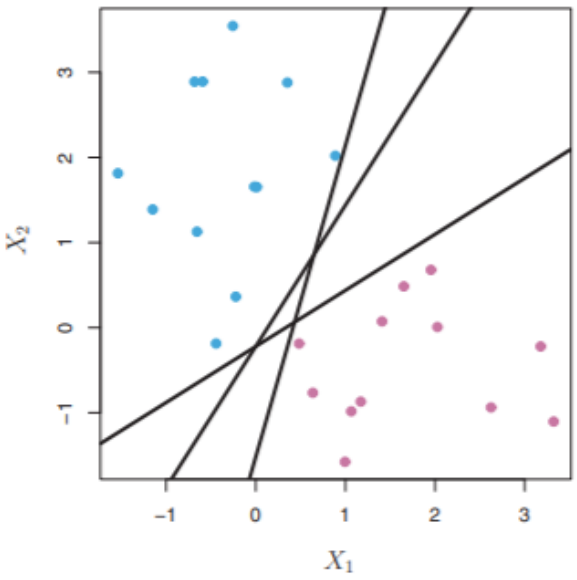


**Figure 7. Separating hyperplanes (James et al. 2013).**

Generally, the hyperplane can be defined by equation 4:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = 0 \quad (4)$$

where $\beta_p$ represents the parameters and $x_p$ are the points on the hyperplane. If a point on the hyperplane, $x_p$, does not satisfy equation 4, the point lies on either side of the hyperplane. Thus, the hyperplane splits the p-dimensional space into two halves and can be considered as a classifier, determining the class of the observation by calculating the sign of the left-hand side of equation 4. (James et al. 2013, Steinwart & Christmann 2008.)

The optimal separating hyperplane, also known as the maximal margin hyperplane, is defined by the following maximization problem:

$$\max_{\beta_0, \beta_1, \ldots, \beta_p} M \quad (5)$$

$$subject\ to\ \sum_{j=1}^{p} \beta_j^2 = 1, \quad (6)$$

$$y_i\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p\right) \geq M \ \ \forall\, i = 1, \ldots, n \quad (7)$$

The maximal margin hyperplane is hence calculated by maximizing the distance between the training observations and the hyperplane; the maximal margin hyperplane is the furthest hyperplane from observations. (James et al. 2013.)

Figure 8. illustrates geometrically the maximal margin hyperplane. The hyperplane splits the space into two halves, classifying the data points into purple and blue dots. The solid line represents the hyperplane and the maximized margin is the distance between the solid line and the dashed lines. The dots that lie on the dashed lines (two blue dots and one purple dot) "supporting" the hyperplane are the support vectors. Their distance from the maximal margin hyperplane is the maximized margin (indicated by the three small arrows). The support vectors of the model are therefore the data points, that are closest to the maximal margin hyperplane and share an equal distance to it. They determine the maximal margin hyperplane exclusively; other observations further from the hyperplane do not affect the shape or location of it. The maximal margin hyperplane acts here as a classification rule; blue grid indicating one class and purple another. Once the classification rule has been decided, the performance of the classifier can be evaluated using the test data set. (James et al. 2013.)

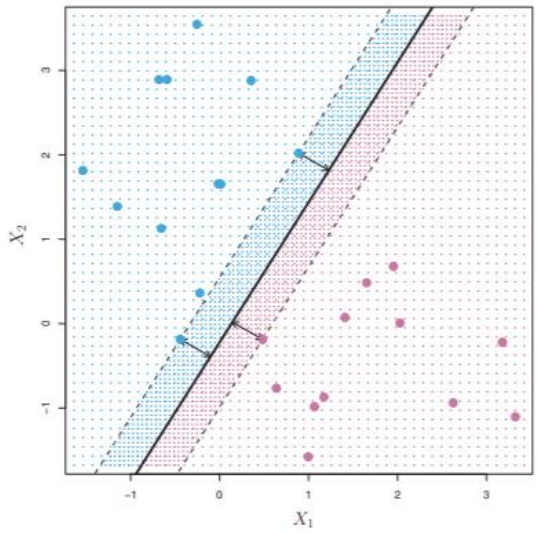


**Figure 8. The maximal margin hyperplane (James et al. 2013).**

The classification tasks illustrated in Figures 7. and 8. are very simple. Real-world problems are often much more complex, and thus the classification model needs to be too. Since in complex classification tasks the observations are not linearly separable, the optimization problem in equations 5-7 does not have any solutions, for $M > 0$. Therefore,

the maximal margin hyperplane does not exist. In the examples in Figures 7. and 8., the data is perfectly linearly separable, in which cases, the classifier can be referred to as a hard margin classifier, because it makes zero classification errors. In order to classify linearly nonseparable observations, a technique called soft margin classifier can be implemented. Soft margin classifier is not able to classify observations with zero errors, but in turn it has a better generalization performance. The property of generalization provides benefits in the sense that it decreases the model's sensitivity to changes in data. As a result, it reduces overfitting and makes the classifier more efficient in general. (James et al. 2013, Ali 2008.)

Figure 9. demonstrates the function of a soft margin classifier. Now, as the data points are clearly linearly nonseparable, the soft margin classifier allows some observations to be classified incorrectly. One purple and one blue dot lie on the wrong side of the soft margin hyperplane, but the hyperplane still fits the data well; the model is able to generalize. (James et al. 2013, Ali 2008.)

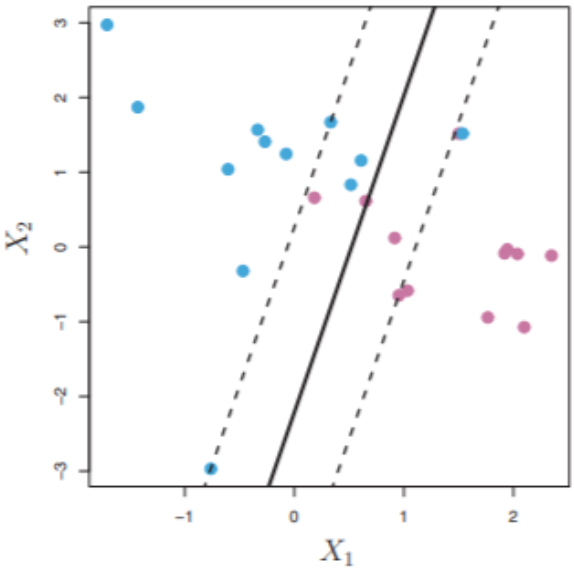


**Figure 9. A soft margin classifier (James et al. 2013).**

The above-described classification methods are referred to as support vector classifiers in ML literature. Their classification capabilities are limited to constructing linear classification margins. Since the real-world classification tasks are often nonlinear, an SVM constructing nonlinear decision boundaries, is required. The actual SVM is therefore an extension of the support vector classifier. (Steinwart & Christmann 2008.)

The idea of a nonlinear SVM is, that it is able to define a nonlinear classification margin in a nonlinear data space. The classification of nonlinear patterns can be done by mapping the input vectors into a higher dimensional feature space using kernels. In a higher dimension, a linear classification margin can be constructed. Figure 10. gives a simplified demonstration of this. (Steinwart & Christmann 2008.)

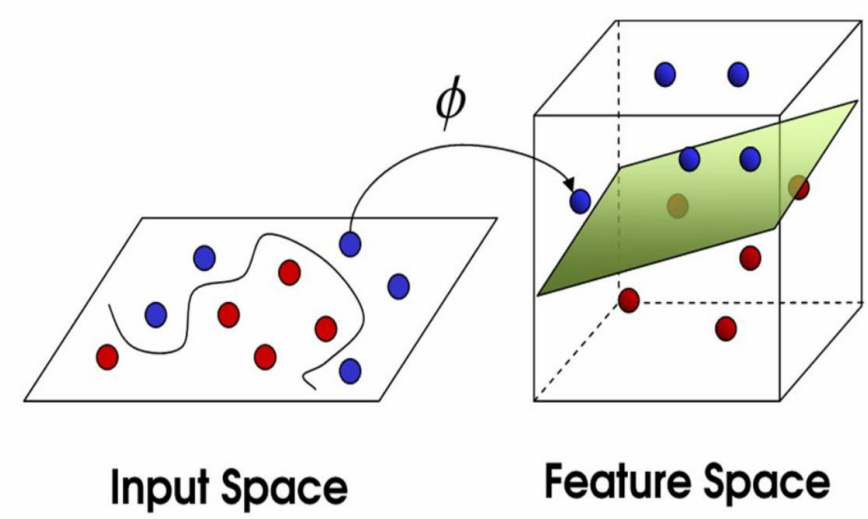


**Figure 10. Mapping input vectors into feature space using kernel (Kaundal et al. 2006).**

Support vector classifiers could be modified to be able to construct nonlinear classification boundaries. This could be done by enlarging the feature space using for example quadratic, cubic, and even higher-order polynomial functions of the predictors. Kernels are actually just a computationally efficient way to execute this. (James et al. 2013.)

To return to the maximization problem in equations 5-7, it can be expressed simply as the inner product of the observations. The inner product of r-vectors a and b is:

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i \, b_i \tag{8}$$

The inner product of two observations $x_i$, $x_{i'}$ is thus defined as:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} \, x_{i'j} \tag{9}$$

Kernel K is a function, that generalizes the inner product as $K(x_i, x_{i'})$. Generally, kernel is a function that defines the similarity of two observations. For example:

$$\mathrm{K}(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij}\, x_{i'j} \qquad (10)$$

represents simply the support vector classifier. Thus, Equation 10 is referred to as a linear kernel. It quantifies the similarity between two observations using basic Pearson correlation. To obtain a more flexible classification margin, for example, a polynomial kernel can be used. Polynomial kernel of degree d has the form:

$$\mathrm{K}(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij}\, x_{i'j})^{d} \qquad (11)$$

where $d$ is a positive integer. Combining a linear support vector classifier with a non-linear kernel, such as a polynomial kernel, results a classifier called SVM.

Another nonlinear alternative to the polynomial kernel is the radial kernel:

$$\mathrm{K}(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \qquad (11)$$

where γ is a positive constant controlling the complexity of the model. A high γ-value increases the complexity of the model, but if the value gets too high, the model gets prone to overfitting as a result. In contrast, a low γ-value simplifies the model, but in the case of too low γ, the bias of the model gets too high. The radial kernel is said to act locally, because in the classification process, a test observation is labeled only in accordance with the nearby training observations. Figure 11. demonstrates the performance of a nonlinear SVM with both a polynomial (on the left-hand side) and a radial kernel (on the right-hand side). The model succeeded to classify the observations with high accuracy, even though the classification margin is not linear in either of the cases. (James et al. 2013.)

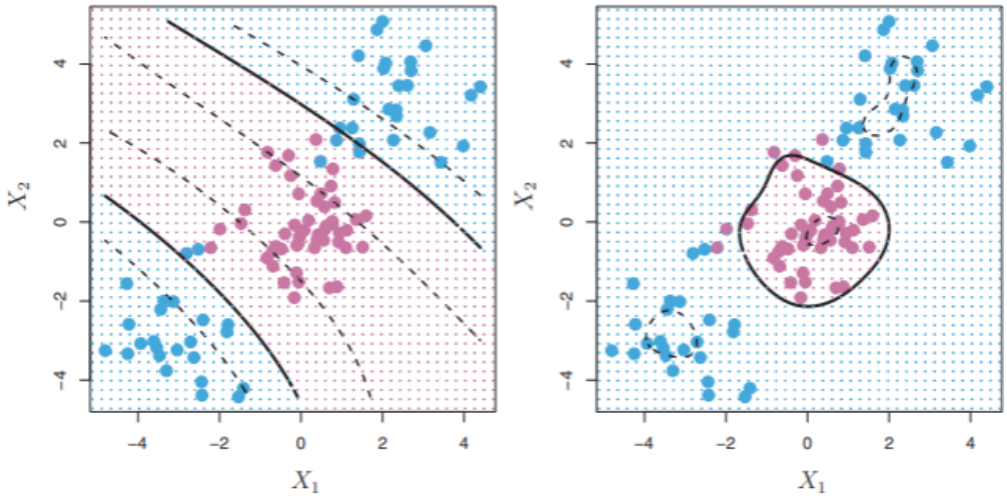


**Figure 11. Binary classification performed with polynomial and radial kernels (James et al. 2013).**

An SVM can be an efficient classification tool. The SVM process itself classifies the observations into two classes, yet often a probability estimate for an observation is desired. According to Madevska-Bogdanova et al. (2004), the SVM outputs can be interpreted as distances between the observations and the separating hyperplane. This allows the calculation of probability for each observation for belonging to either of the classes. If an observation is close to the hyperplane, the probability of the observation being misclassified is high and vice versa. The integrated binary discriminant rule (IBDR) helps to improve the classification performance of an SVM by utilizing a logistic regression model. To put it simply, if the result obtained from a logistic regression is consistent with the result obtained from an SVM (with a probability large enough), the IBDR accepts the SVM's result. Otherwise, if the logistic regression supports the SVM's result with a probability low enough, the result is modified by IBDR.

As noted in section 4.4., the data points to be classified in this study are preprocessed form 10-k annual reports. These reports are represented as high-dimensional vectors in a space consisting of individual words and n-grams. Using a kernel, the vectors in this space are mapped to a higher dimensional feature space, where a decision boundary can be fitted to classify the data points into two classes (credit rating increase and decrease).

### 4.7.2 Logistic regression

Despite its name, logistic regression is used for classification problems rather than regression. It is a classic, straightforward and easy to interpret technique, that is still widely used and effective in classification problems. (Kuhn & Johnson 2013.) According to James et al. (2013), SVM and logistic regression often give very similar results. Generally speaking, SVMs tend to perform better when the classes are well separated, while logistic regression usually deals better with overlapping classes. This is however only a guideline; the results vary significantly depending on the task at hand.

A logistic regression model predicts the probability for an observation to belong to either of the classes in a binary classification problem. It models the logarithmic odds of an event as a linear function:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_P x_P \tag{12}$$

where p is the probability of an event, P the number of predictors, $X_1,\ldots,X_P$ are the predictor variables and $\beta_0,\ldots,\beta_P$ are the coefficients ($\beta_0$ is the bias/intercept term). The right-hand side of the equation 12 is referred to as the linear predictor and the left-hand side as the log-odds or logit. This equation can be further modified into:

$$p = \frac{1}{1+\exp[-(\beta_0+\beta_1 x_1+\cdots+\beta_P x_P)]} \tag{13}$$

Which is now a nonlinear, sigmoidal function that constraints the estimated probabilities between 0 and 1. The coefficients in equation 13 are estimated using the maximum likelihood estimation method. This method defines the coefficient estimates so, that it yields a p value close to 1 for all observations belonging to class 1 and a p value close to 0 for all observations belonging to class 0. Formally, the likelihood function can be stated as:

$$\ell(\beta_0, \beta_1, \ldots, \beta_p) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \tag{14}$$

where the estimates for $\beta_0$, $\beta_1,\ldots$, $\beta_p$ are calculated to maximize the function. (James et al. 2013.)

A simple illustrated example in Figure 12. shows how the sigmoidal function tells the probability for an observation (a company, the orange ticks lying on the black dashed lines) to belong to either of the classes (default or not) depending on the balance value. The sigmoidal function (the blue curve in the figure) acts here as a classifier, classifying unseen instances to either of the classes. If the probability of an instance to belong to class 1 is > 0.5, based on the balance value, the instance is classified into that class. If, in turn, the probability is < 0.5, the instance is classified into the class 0. However, 0.5 is just a default decision boundary and it can be tuned depending on the classification task at hand. (Kuhn & Johnson 2013, James et al. 2013.)

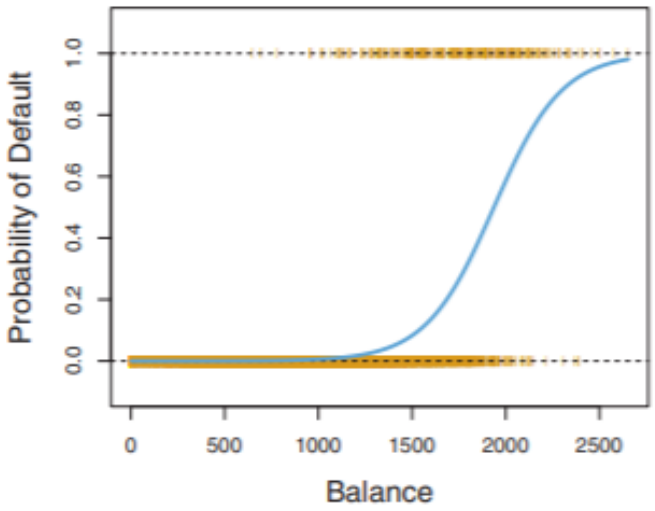


**Figure 12. An example of a sigmoidal function (James et al. 2013).**

In this study, logistic regression is implemented so that the coefficients ($\beta_0$, $\beta_1$,…, $\beta_p$ in equation 13) for predictor variables (TF-IDF scores for words and n-grams) in the training data are optimized in a way that each training data instance corresponds to the true class (increase or decrease) as well as possible. In other words, $\beta_0$, $\beta_1$,…, $\beta_p$ are calculated to maximize the maximum likelihood function, such that plugging the coefficient estimates in equation 13, yields a p (the probability of an event) value close to 1 for training data instances (documents), for which the corresponding class is 1 (rating increase) and close to zero for instances, for which the corresponding class is 0 (rating decrease). Now, the sigmoidal function can be used to classify the test data instances. The $X_1$,…,$X_P$ values in the sigmoidal function fitted to training data are replaced with corresponding values of the test data instances, one instance at a time. This way, a probability estimate for each test data instance to belong in either of the classes, is obtained.

### 4.7.3 Decision trees

Decision trees are supervised ML algorithms, that can be utilized in both classification (classification trees) and regression tasks (regression trees). Regression and classification trees have a lot in common, but the special features of regression trees will not be reviewed in this thesis, since the primary focus is in classification. The structure of classification trees can either be designed by human experts or be automatically generated. The implementation of the latter approach works well with labeled data, and hence it is chosen for this thesis. (Hassan & Verma 2009.)

The principle of classification trees is quite simple. The tree recursively splits the data (the "successor child" being chosen to continue in the process), until the purity of the decision nodes cannot be improved anymore, or a certain predefined stopping criterion is reached. Figure 13. demonstrates the principle of a simple binary classification tree.

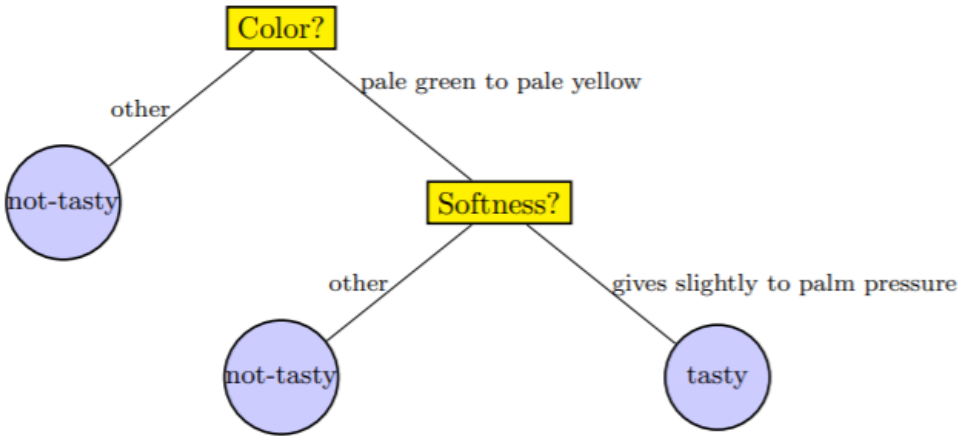


**Figure 13. Binary classification tree model (Shai & Shai 2014).**

In Figure 13., the classification tree is used to determine whether a papaya is tasty or not. In the first decision node the color of the papaya is examined. If the color is something else than pale green to pale yellow, the papaya is predicted to be not-tasty. But if this criterion is met, the tree continues examining the papaya, this time its softness. If the softness level satisfies criterion "gives slightly to palm pressure", the model predicts the papaya to be tasty, and vice versa. In this example the decision tree is very simple and

consequently very small. As the classification task gets more complex, the size of the tree increases accordingly. (Shai & Shai 2014.)

The data splits are performed using the predictor variables (in Figure 13. the color and softness). The first decision node, where the whole data set is split, is called the root node. Branches connect the root node, the inner nodes and the leaf nodes. The leaf nodes represent the final output labels; no more splits are made in them. (Shai & Shai 2014.)

The above-mentioned purity of a decision node is measured using either the Gini index or cross-entropy. Gini index is defined by:

$$G = \sum_{k=1}^{K} \hat{p}_{mk} (1 - \hat{p}_{mk}) \quad (15)$$

Which represents the total variance across the K classes (K=2 in binary classification). Here, $\hat{p}_{mk}$ is the share of training instances of class k in the node m. As $\hat{p}_{mk}$ can only take a value between 0 and 1, the value of G varies between 0 and 0.5. When all training instances in a node belong to the same class (all $\hat{p}_{mk}$ are close to zero or one), the node is pure, and the Gini index takes a value of 0. Accordingly, a Gini index of 0.5 indicates that the node is impure, and both classes are equally represented in the node. An alternative way to express the purity of a node is cross-entropy, which is given by:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \quad (16)$$

The formula of cross-entropy results somewhat similar value that Gini index does. (James et al. 2013.)

Either Gini index or cross-entropy is used as a split criterion to grow a classification tree. The value of either of these measures for a node defines the structure of the tree in a way, that the split performed in the root node yields the greatest possible reduction in the value of either of them. In contrast, the leaf nodes have the lowest possible decreases in Gini index or cross-entropy values. Thus, these criteria determine, how the predictor variables are arranged in the tree structure. (James et al. 2013.)

If the tree is allowed to grow until all the leaves achieve perfect purity, there is a considerable risk of overfitting. In a case of complex classification problem, an unlimited tree becomes very large, and has a high variance. As stated in section 2.2.5., this leads to a perfect fit on the training data, but poor classification performance on the test data. To

avoid overfitting, complexity parameters can be defined for the model. These parameters limit, for example, the depth of the tree, and thus the overall size of it. (Alpaydin 2014.)

Another way to avoid overfitting and to enhance the performance of the model is a method called pruning. To put it simply, pruning eliminates non-informative branches from the tree. If a branch is not improving the classification performance, it is pruned. This simplifies the structure of the tree, avoids overfitting and makes it more interpretable. (Kuhn & Johnson 2013.)

A tree-like structure and the ability to display them graphically makes classification trees very simple to understand and interpret. Consisting of nested if-then statements, decision trees reflect the decision-making process of humans. These attributes make decision trees a popular classification technique. However, more complex models tend to achieve more precise classification results. A more complex ML model combining decision trees and bootstrap aggregating, called Random forests, is introduced in the next section. (Hassan & Verma 2009, James et al. 2013.)

Figure 14. shows a simplified demonstration of a decision tree's classification process in this study. In the figure, as the process starts, all the instances (10-k documents) to be classified are in the first node (samples = 1312, gini = 0.5). In each node of the tree, "value" tells how many instances are classified to each class so far. X[N] tells which one of the predictor variables makes an optimal split in the corresponding node. N refers here to the ordinal number of each variable (variables are ranked on the basis of their TF-IDF score). Corresponding TF-IDF score is given next to X[N]. This tree is just an example, and thus very simplified. If this tree was used in the classification task of this thesis, the classification performance would be very weak as a consequence. The decision trees grown for classification in this thesis are much deeper and more complex.

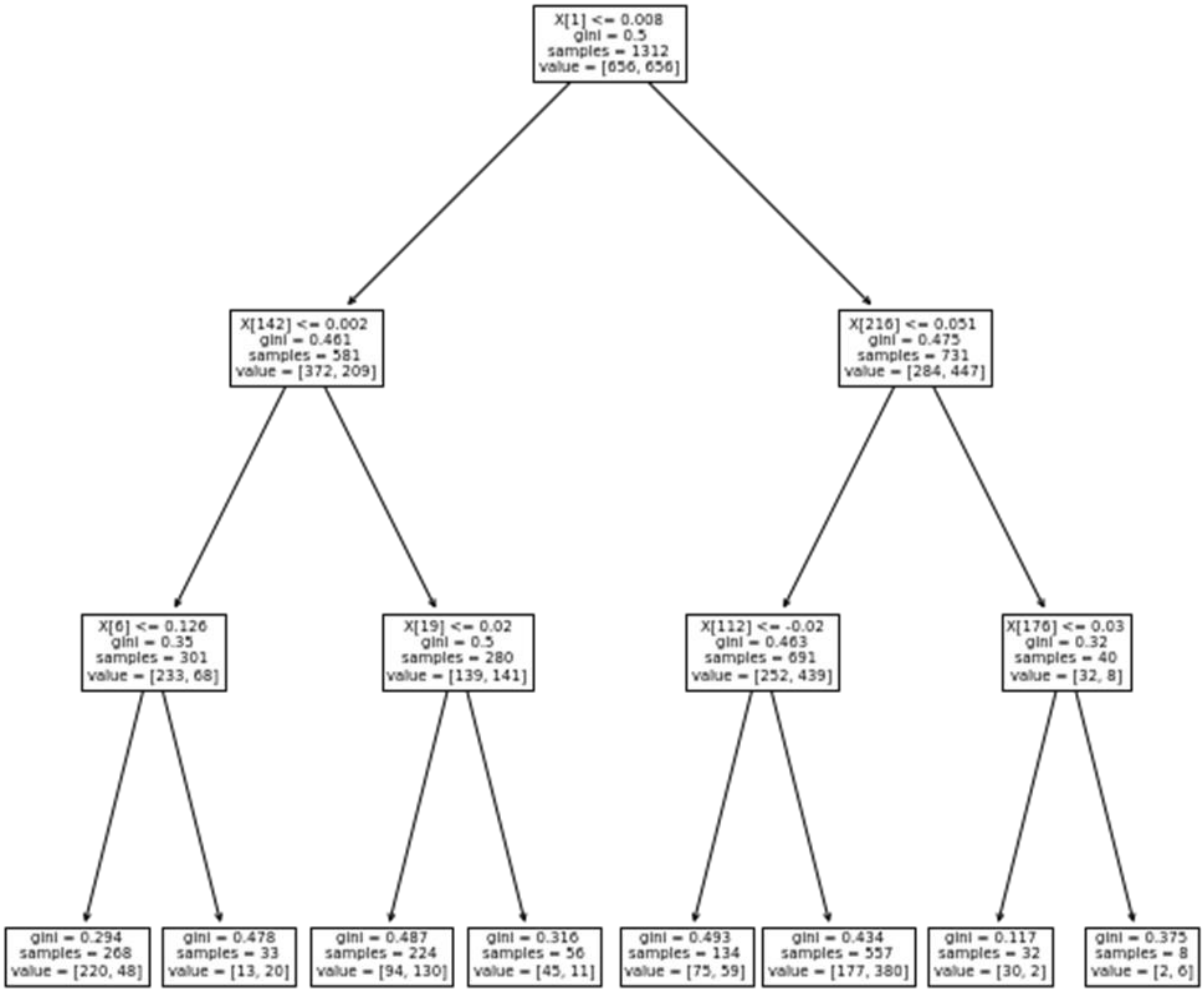


**Figure 14. An example of a decision tree.**

### 4.7.4 Random forest

As stated above, decision trees tend to have a high variance. A method called bootstrap aggregating (bagging) can be utilized to reduce the variance of an ML model. It is widely used, especially when handling decision trees. The main idea behind bagging is that variance can be reduced, when averaging a set of observations. Thus, in a classification task, an easy way to implement this would be to pick multiple training sets from the data, con-

struct an individual classifier using each of them, and finally take an average of the obtained results. In this manner, the variance of the final model is reduced. (James et al. 2013.)

Since the data is limited, it is not possible to pick many non-overlapping training sets from the data. Therefore, in bagging, the picked training sets are so called bootstrap samples. This means that when a single training set is chosen from the data, several bootstrap samples are randomly picked from it. After each data instance is chosen for a bootstrap sample, the data instance is put back into the original training data set. A bootstrap sample is thus picked with replacement; some data instances may be chosen several times and others not even once. Each bootstrap sample is used to build an individual classifier. Each individual classifier provides slightly different predictions, and the final result is obtained by averaging all the predictions made by these individual classifiers. (James et al. 2013.)

In the case of classification trees, bagging process is carried out by first picking n bootstrap samples, and then n classification trees are grown without any restrictions, also without pruning. After that, the class prediction of each tree is observed, and the final prediction is the majority class. Each individual tree is known to have low bias and high variance. Utilizing the bagging method, variance can be reduced, without increasing the bias. Usually the number of trees grown using bagging is some hundreds or thousands, depending on the task. (James et al. 2013.)

Random forest is an ensemble model, that utilizes both bagging and classification trees[10]. As its name suggests, in random forest several decision trees are grown, randomly. The final classification result is determined by the majority class predicted by the individual trees. Part of the randomness is provided by the bagging method, but that alone is not enough. The rest of the required randomness is achieved through limiting the number of predictors, that the tree can use for dividing the data in the decision nodes. Hence, at each node of each tree, rather than selecting the overall optimal predictor to divide the data (as explained in the previous section, 4.7.3.), the optimal predictor is chosen from a random subset of predictors. As a rule of thumb, the size of this subset is the square root of the total number of predictors. The number of trees is usually increased until no decrease in the error rate is achieved anymore. (Marsland 2014.)

---

[10] Both classification and regression trees can be utilized in a random forest model. For the sake of relevancy, only the former case is reviewed in this thesis.

Randomly choosing the available predictors yields, that the trees grown are not dependent on each other. Thus, it increases the randomness in the growing process of each tree. Alike bagging alone, the combination of these randomness factors results in reduced variance, without increasing the bias. One benefit provided by random forest is also that the pruning of the trees can be omitted entirely. One more feature worth mentioning is the so-called out-of-bootstrap data instances. When choosing the bootstrap sample from the training data, the not-chosen data instances can be used for validating the classification model built from the bootstrap sample. Utilizing these out-of-bootstrap instances eliminates the need for cross-validation. (Marsland 2014.)

### 4.7.5 Naïve Bayes classifier

According to Kuhn & Johnson (2013), Bayes' rule determines the probability for an observation to belong in a class, given the observed predictors. Formally, this can be presented as follows:

$$\Pr[Y = C_l|X] = \frac{\Pr[Y]\Pr[X|Y=C_l]}{\Pr[X]} \quad (17)$$

where Y represents the class variables and X the predictor variables. Thus, $\Pr[Y = C_l|X]$ can be interpreted as the probability that the result is the l:th class, given the X values. The left-hand side of the equation is usually called the posterior probability of the class. On the right-hand side, $\Pr[Y]$ is the prior probability of the result, $\Pr[X]$ is the probability of the predictor values and $\Pr[X|Y = C_l]$ is the conditional probability.

In the naïve Bayes approach, the probability calculation is simplified significantly by assuming, that all of the predictor variables are independent of each other. In the classification task of this thesis this means that each word position is generated independently of every other. This is usually an unquestionably unrealistic assumption, hence the name naïve Bayes. Using this assumption however reduces the calculation complexity greatly. Under the assumption of independency, in equation 17 the conditional probability $\Pr[X|Y = C_l]$ can be now calculated as the product of the probability densities for each predictor variable:

$$\Pr[X|Y = C_l] = \prod_{j=1}^{P} \Pr[X_j|Y = C_l] \quad (18)$$

$\Pr[X]$, the probability of the predictor values can be calculated in a similar manner, under the assumption of independency. $\Pr[Y]$ is simply the frequency of the class in the training data. The naïve Bayes classifier predicts the class, which has the highest posterior probability, hence maximizing equation 17. (Kuhn & Johnson 2013.)

One significant advantage of naïve Bayes classifier is, that the reduced complexity in probability calculations makes the model quick to compute. (Kuhn & Johnson, 2013). The inevitable violation of the naïve independence assumption weakens the accuracy of probability estimations. However, this does not necessarily mean that the classification itself would be unsuccessful. Naïve Bayes classifier is relatively efficient, despite its "naivety". (Kubat 2017.)

A special kind of naïve Bayes classifier, multinomial naïve Bayes (MNB), was chosen for the classification task of this study. This was done, because according to Sklearn, it is particularly suitable for text classification tasks (Scikit-learn.org). The word "multinomial" refers here to the structure of the features. In this thesis, the features (document vectors) can be seen to represent the frequencies with which events (occurrence of a word or an n-gram in a document) have been generated by a multinomial $p_1,\ldots,p_n$, where $p_i$ is the probability that event i occurs. A document vector $\mathbf{x} = (x_1,\ldots,x_n)$ is then a histogram, with $x_i$ counting the number of times event i was observed in a particular instance. (Rennie et al. 2003.)

Practically, using the training data, a probability to belong to a class can be determined for each word and n-gram. Thus, the probability for each word and n-gram in a test data instance to belong to a class can be calculated accordingly. This way, using the MNB classifier, the instance is simply classified in the class that achieves the higher probability.

# 5 RESULTS

In this section, the performance of the ML models introduced in section 4.7, implemented to the binary credit rating classification task, is evaluated one by one. First, the performance of all models is evaluated using a BOW approach, followed by an identical evaluation using topic modeling. An objective assessment is carried out using the evaluation metrics described in section 2.2.8. Thus, accuracy, precision, recall, F1 and AUC scores were calculated for each model, using all datasets. Precision, recall and F1 scores were calculated with respect to both classes. A confusion matrix was also calculated in each case to help interpreting the results. For each constructed model, the best result for each n-gram range is reported. Thus, three different cases are tabulated for each model, which are evaluated in more detail.

Each ML algorithm used in this study has specific hyperparameters (except naïve Bayes) that were tuned for optimal results. For SVM, logistic regression and decision tree, the parameters were tuned using GridSearchCV of the Scikit learn machine learning library (an exhaustive search). RandomizedSearchCV (randomized search, all the options are not calculated) was used to find optimal parameters for random forest, because GridSearchCV was computationally too expensive for it (very long calculation times). All models were trained and tested using the Scikit learn ML library. A summary of the performance of all models is compiled at the end of this section.

## 5.1 Bag-of-words

### 5.1.1 Support vector machine

For SVM, the tuned hyperparameters were a penalty parameter C and the type of kernel. C controls the variance/complexity of the model; higher C values allow the model to become more complex, and thus makes the model more prone to overfitting. Considered kernel types were the ones introduced in section 4.7.1.; linear, polynomial and radial kernel. The type of kernel determines, which kind of decision boundary is fitted to the data. Table 3. wraps up the highest performance scores in each n-gram case. In each n-gram case, the optimal C value was 10, and the optimal kernel type was polynomial. The best classification result was achieved using the n-gram range 1-3 with 2000 features. This dataset yielded an accuracy of 69.4 % and an area under curve (AUC) of 0.6744, which was the overall best classification score in the whole study.

**Table 3. SVM classification results (BOW).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, 3000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6683 | 0.6866 | decrease | 0.74 | 0.75 | 0.75 |
| | | | increase | 0.60 | 0.59 | 0.59 |
| | | | | | | |
| 2 | n-gram range = 1-2, 1000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6425 | 0.6679 | decrease | 0.72 | 0.76 | 0.74 |
| | | | increase | 0.58 | 0.53 | 0.55 |
| | | | | | | |
| 3 | n-gram range = 1-3, 2000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6744 | 0.6940 | decrease | 0.74 | 0.76 | 0.75 |
| | | | increase | 0.61 | 0.59 | 0.60 |

**Table 4. SVM confusion matrices (BOW).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, 3000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **123** | **41** | 164 |
| | | increase | **43** | **61** | 104 |
| | | | 166 | 102 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, 1000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **124** | **40** | 164 |
| | | increase | **49** | **55** | 104 |
| | | | 173 | 95 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, 2000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **125** | **39** | 164 |
| | | increase | **43** | **61** | 104 |
| | | | 168 | 100 | N 268 |

In Table 4., the corresponding confusion matrices are presented. The matrices are quite similar; the decrease class was classified significantly better, which can also be seen from the corresponding precision, recall and F1 scores. In cases 1 and 3, the increase class was predicted a little bit better than in case 2. The difference in prediction performance between cases 1 and 3 derives from the case 3's slightly better ability to predict the decrease class.

### 5.1.2 Logistic regression

In logistic regression, the only hyperparameter tuned was the penalty parameter C, which acts similarly in logistic regression, as in SVM. Table 5. shows the results of logistic regression. In cases 1 and 2, the optimal C value was 1000. In case 3, the optimal value was 100. The best performance was achieved using the n-gram range 1-3 with 2000 features. The AUC and accuracy scores in this case were 0.6617 and 68.29 %. Thus, the performance of logistic regression was almost as good as of SVM.

**Table 5. Logistic regression classification results (BOW).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, 2000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6521 | 0.6754 | decrease | 0.73 | 0.76 | 0.74 |
| | | | increase | 0.59 | 0.55 | 0.57 |
| | | | | | | |
| 2 | n-gram range = 1-2, 3000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6333 | 0.6567 | decrease | 0.71 | 0.74 | 0.72 |
| | | | increase | 0.56 | 0.53 | 0.54 |
| | | | | | | |
| 3 | n-gram range = 1-3, 2000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6617 | 0.6829 | decrease | 0.73 | 0.76 | 0.74 |
| | | | increase | 0.60 | 0.57 | 0.58 |

Table 6. shows the corresponding confusion matrices. In cases 1 and 3, the ability to predict the decrease class was equal, but in case 3, the increase class was predicted slightly better. In case 2, the predictions were the worst for both classes.

**Table 6. Logistic regression confusion matrices (BOW).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, 2000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **124** | **40** | 164 |
| | | increase | **47** | **57** | 104 |
| | | | 171 | 97 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, 3000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **121** | **43** | 164 |
| | | increase | **49** | **55** | 104 |
| | | | 170 | 98 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, 2000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **124** | **40** | 164 |
| | | increase | **45** | **59** | 104 |
| | | | 169 | 99 | N 268 |

### 5.1.3 Decision tree

For decision tree, the tuned hyperparameters were a little different; maximum depth of tree (max_depth in Scikit learn), minimum number of samples required to be at a leaf node (min_samples_leaf) and minimum number of samples required to split an internal node (min_samples_split). All of these parameters determine the size and structure of the decision tree to control the bias-variance trade-off. In addition, Gini impurity was used to measure the purity of the leaves. The classification results of the decision tree are presented in Table 8. The n-gram range 1 with 1000 features yielded the best classification performance, with an accuracy of 63.81 % and an AUC of 0.6251. Optimal hyperparameter values for decision tree are given in Table 7. Clearly, the best setup for the decision tree was to strictly limit the size of the tree, the size of the TD matrix (only individual words are considered) and the number of features.

**Table 7. Optimal hyperparameter values for decision tree (BOW).**

| | n-gram range = 1, 1000 features | n-gram range = 1-2, 1000 features | n-gram range = 1-3, unlimited features |
|---|---|---|---|
| min_samples_split | 2 | 2 | 5 |
| min_samples_leaf | 2 | 1 | 1 |
| max_depth | 10 | 15 | 30 |

**Table 8. Decision tree classification results (BOW).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, 1000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6251 | 0.6381 | decrease | 0.71 | 0.68 | 0.7 |
| | | | increase | 0.53 | 0.57 | 0.55 |
| | | | | | | |
| 2 | n-gram range = 1-2, 1000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6060 | 0.6082 | decrease | 0.71 | 0.62 | 0.66 |
| | | | increase | 0.50 | 0.60 | 0.54 |
| | | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6072 | 0.6269 | decrease | 0.70 | 0.70 | 0.70 |
| | | | increase | 0.52 | 0.52 | 0.52 |

The confusion matrices show in Table 9., that in case 2 the predictions were clearly the worst for the decrease class. In cases 1 and 3 the predictions for the decrease class were quite similar, but for the increase class, case 1 achieved a better recall score, and thus was the optimal decision tree classifier.

**Table 9. Decision tree confusion matrices (BOW).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, 1000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **112** | **52** | 164 |
| | | increase | **45** | **59** | 104 |
| | | | 157 | 111 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, 1000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **101** | **63** | 164 |
| | | increase | **42** | **62** | 104 |
| | | | 143 | 125 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **114** | **50** | 164 |
| | | increase | **50** | **54** | 104 |
| | | | 164 | 104 | N 268 |

### 5.1.4 Random forest

For random forest, tuned hyperparameters were the same as for decision tree, except the number of trees (n_estimators). The number of trees parameter defines, how many decision trees are grown in the forest. Optimal hyperparameter values for random forest are given in Table 10. Table 11. shows the classification results of random forest. The results were somewhat similar for all n-gram ranges, yet case 2, where n-gram range was 1-2 and number of features was restricted to 2000, performed best. The AUC and accuracy scores in this case were 0.6548 and 66.79 %.

**Table 10. Optimal hyperparameter values for random forest (BOW).**

| | n-gram range = 1, 1000 features | n-gram range = 1-2, 2000 features | n-gram range = 1-3, 1000 features |
|---|---|---|---|
| min_samples_split | 2 | 5 | 5 |
| min_samples_leaf | 1 | 1 | 1 |
| max_depth | 30 | 15 | 15 |
| n_estimators | 1000 | 2000 | 2000 |

**Table 11. Random forest classification results (BOW).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, 1000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6341 | 0.6642 | decrease | 0.71 | 0.77 | 0.74 |
| | | | increase | 0.58 | 0.50 | 0.54 |
| | | | | | | |
| 2 | n-gram range = 1-2, 2000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6548 | 0.6679 | decrease | 0.74 | 0.71 | 0.72 |
| | | | increase | 0.57 | 0.60 | 0.58 |
| | | | | | | |
| 3 | n-gram range = 1-3, 1000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6316 | 0.6567 | decrease | 0.71 | 0.74 | 0.73 |
| | | | increase | 0.56 | 0.52 | 0.54 |

Corresponding confusion matrices are given in Table 12. In case 2, the prediction performance for the decrease class was the worst, but the ability to predict the increase class most accurately made it the optimal random forest classifier.

**Table 12. Random forest confusion matrices (BOW).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, 1000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **126** | **38** | 164 |
| | | increase | **52** | **52** | 104 |
| | | | 178 | 90 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, 2000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **117** | **47** | 164 |
| | | increase | **42** | **62** | 104 |
| | | | 159 | 109 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, 1000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **122** | **42** | 164 |
| | | increase | **50** | **54** | 104 |
| | | | 172 | 96 | N 268 |

### 5.1.5 Naïve Bayes classifier

For naïve Bayes classifier, there were no hyperparameters to tune. Among many types of naïve Bayes classifiers, MNB was chosen for the task, because according to Scikit-learn, it is suitable for text classification. The results are represented in Table 13. The best classification result was achieved using the n-gram range 1-3 with unlimited features. With this dataset, the AUC and accuracy scores were 0.6177 and 62,7 %. Clearly with MNB, limiting the number of features is not an efficient choice. MNB classifier was the worst, when using the BOW approach.

**Table 13. MNB classification results (BOW).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.5939 | 0.5784 | decrease | 0.71 | 0.52 | 0.60 |
| | | | increase | 0.47 | 0.66 | 0.55 |
| | | | | | | |
| 2 | n-gram range = 1-2, 3000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.5926 | 0.5746 | decrease | 0.71 | 0.51 | 0.60 |
| | | | increase | 0.47 | 0.67 | 0.55 |
| | | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6177 | 0.6269 | decrease | 0.71 | 0.66 | 0.68 |
| | | | increase | 0.52 | 0.58 | 0.55 |

**Table 14. MNB confusion matrices (BOW).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **86** | **78** | 164 |
| | | increase | **35** | **69** | 104 |
| | | | 121 | 147 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, 3000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **84** | **80** | 164 |
| | | increase | **34** | **70** | 104 |
| | | | 118 | 150 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **108** | **56** | 164 |
| | | increase | **44** | **60** | 104 |
| | | | 152 | 116 | N 268 |

Corresponding confusion matrices are shown in Table 14. Interestingly, the prediction performance for the minor class (increase) was quite good, hence in each case relatively high F1 score was achieved for it. Comparing to the other classifiers, competitive results for predicting the increase class were obtained using the MNB classifier. However, the poor ability to predict the major class (decrease) deteriorated the overall performance of the MNB, in each case.

## 5.2 Topic modeling

Like all classification models constructed in this study, the LSA was also implemented using the Scikit learn library. Using the 12 LSA-modified datasets in Table 2., all five ML algorithms were trained and tested again. Their optimal hyperparameters and classification results are presented in this section.

### 5.2.1 Support vector machine

Using the topical content as input, optimal hyperparameters for SVM were C=100 in case 1, C=20 in case 2 and C=30 in case 3. 2000 was the optimal number of features in each case. The best result in each case was obtained using a polynomial kernel. Classification results were very similar in each case, case 3 achieving narrowly the best AUC and accuracy scores (0.6551 and 67.91 %). Comparing to the results obtained using the BOW approach, topic modeling gave slightly worse results. The classification results are given in Table 15.

**Table 15. SVM classification results (topic modeling).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, 2000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6455 | 0.6716 | decrease | 0.72 | 0.76 | 0.74 |
| | | | increase | 0.59 | 0.53 | 0.56 |
| | | | | | | |
| 2 | n-gram range = 1-2, 2000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6438 | 0.6716 | decrease | 0.72 | 0.77 | 0.74 |
| | | | increase | 0.59 | 0.52 | 0.55 |
| | | | | | | |
| 3 | n-gram range = 1-3, 2000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6551 | 0.6791 | decrease | 0.73 | 0.76 | 0.74 |
| | | | increase | 0.59 | 0.55 | 0.57 |

**Table 16. SVM confusion matrices (topic modeling).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, 2000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **125** | **39** | 164 |
| | | increase | **49** | **55** | 104 |
| | | | 174 | 94 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, 2000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **126** | **38** | 164 |
| | | increase | **50** | **54** | 104 |
| | | | 176 | 92 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, 2000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **125** | **39** | 164 |
| | | increase | **47** | **57** | 104 |
| | | | 172 | 96 | N 268 |

The confusion matrices for SVM are given in Table 16. Like the overall performance of the classification, the classification errors made were quite similar between the three cases. In case 3, the ability to predict the increase class slightly better than in cases 1 and 2, yielded the best overall classification scores. With the BOW approach, the prediction performance was better for both classes.

### 5.2.2 Logistic regression

Using logistic regression, the best result in each case was obtained with the C value of 1000 and unlimited features. Clearly, using logistic regression with topic modeling, a high level of complexity yielded the best results. Using the BOW approach, in turn, the optimal setup was to limit the number of features and to lower the C value. The n-gram range 1-

2 yielded the best result with topic modeling, an AUC of 0.6299 and an accuracy of 64.18 %. The classification performance of logistic regression is presented in Table 17.

**Table 17. Logistic regression classification results (topic modeling).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6177 | 0.6269 | decrease | 0.71 | 0.66 | 0.68 |
| | | | increase | 0.52 | 0.58 | 0.55 |
| | | | | | | |
| 2 | n-gram range = 1-2, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6299 | 0.6418 | decrease | 0.72 | 0.68 | 0.70 |
| | | | increase | 0.54 | 0.58 | 0.56 |
| | | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6173 | 0.6306 | decrease | 0.71 | 0.68 | 0.69 |
| | | | increase | 0.52 | 0.56 | 0.54 |

The corresponding confusion matrices in Table 18. show that the classification performance was quite similar in all cases. In case 2, the ability to predict both classes most precisely yielded narrowly the best overall classification scores. With the BOW approach, the prediction performance was better for both classes.

**Table 18. Logistic regression confusion matrices (topic modeling).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **108** | **56** | 164 |
| | | increase | **44** | **60** | 104 |
| | | | 152 | 116 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **112** | **52** | 164 |
| | | increase | **44** | **60** | 104 |
| | | | 156 | 112 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **111** | **53** | 164 |
| | | increase | **46** | **58** | 104 |
| | | | 157 | 111 | N 268 |

### 5.2.3 Decision tree

For decision tree, the optimal hyperparameter values are given in Table 19. and the classification results of each case in Table 20. The best AUC score (0.6352) was achieved in case 2, and the best accuracy score (65,3 %) was achieved in case 3. In case 3, more test data instances were predicted correctly, but in case 2, the difference between error types was a lot smaller. Therefore, case 2 was the overall best classifier (higher AUC score). Best results were obtained, when the tree was let to grow quite deep (with respect to the BOW approach) and the number of features considered was unlimited. Thus, the optimal decision tree was relatively complex.

**Table 19. Optimal hyperparameters for decision tree (topic modeling).**

| | n-gram range = 1, unlimited features | n-gram range = 1-2, unlimited features | n-gram range = 1-3, 3000 features |
|---|---|---|---|
| min_samples_split | 2 | 15 | 5 |
| min_samples_leaf | 1 | 1 | 1 |
| max_depth | 30 | 30 | 20 |

**Table 20. Decision tree classification results (topic modeling).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.5844 | 0.6120 | decrease | 0.67 | 0.71 | 0.69 |
| | | | increase | 0.50 | 0.46 | 0.48 |
| | | | | | | |
| 2 | n-gram range = 1-2, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6352 | 0.6418 | decrease | 0.73 | 0.66 | 0.69 |
| | | | increase | 0.53 | 0.61 | 0.57 |
| | | | | | | |
| 3 | n-gram range = 1-3, 3000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6338 | 0.6530 | decrease | 0.72 | 0.72 | 0.72 |
| | | | increase | 0.55 | 0.55 | 0.55 |

Corresponding confusion matrices in Table 21. show that in case 2, the ability to classify the increase class was the best. In case 1, the increase class predictions were not better than a pure guess. With the BOW approach, the prediction performance was better for both classes.

**Table 21. Decision tree confusion matrices (topic modeling).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **116** | **48** | 164 |
| | | increase | **56** | **48** | 104 |
| | | | 172 | 96 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **109** | **55** | 164 |
| | | increase | **41** | **63** | 104 |
| | | | 150 | 118 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, 3000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **118** | **46** | 164 |
| | | increase | **47** | **57** | 104 |
| | | | 165 | 103 | N 268 |

### 5.2.4 Random forest

The optimal hyperparameters for random forest are given in Table 22. and the classification results are given in Table 23. As in the BOW approach with random forest, the best classification performance was achieved by limiting the features. Using topic modeling, 1000 features and the n-gram range 1-3 yielded the best results; an AUC of 0.6446 and an accuracy of 67.91%. Comparing to the BOW approach, in this case the optimal depth of trees was significantly greater, meaning that the optimal classifier was more complex.

**Table 22. Optimal hyperparameters for random forest (topic modeling).**

| | n-gram range = 1, unlimited features | n-gram range = 1-2, 1000 features | n-gram range = 1-3, 1000 features |
|---|---|---|---|
| min_samples_split | 10 | 2 | 2 |
| min_samples_leaf | 2 | 1 | 1 |
| max_depth | 20 | 20 | 50 |
| n_estimators | 200 | 1000 | 500 |

**Table 23. Random forest classification results (topic modeling).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6228 | 0.6567 | decrease | 0.70 | 0.77 | 0.73 |
| | | | increase | 0.57 | 0.47 | 0.52 |
| | | | | | | |
| 2 | n-gram range = 1-2, 1000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6345 | 0.6754 | decrease | 0.70 | 0.82 | 0.75 |
| | | | increase | 0.61 | 0.45 | 0.52 |
| | | | | | | |
| 3 | n-gram range = 1-3, 1000 features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6446 | 0.6791 | decrease | 0.71 | 0.80 | 0.75 |
| | | | increase | 0.61 | 0.49 | 0.54 |

Corresponding confusion matrices are given in Table 24. The ability to predict the increase class correctly was quite poor. Actually, in all three cases a majority of the increase class instances were classified incorrectly (low recall scores). Thus, the increase class predictions were not better than a pure guess. In case 3, the increase class predictions were the least bad, making it the optimal classifier. Using topic modeling, the decrease class was predicted better than when using the BOW approach. However, when using the BOW, the increase class was predicted remarkably better. Therefore, the BOW approach yielded better overall performance.

**Table 24. Random forest confusion matrices (topic modeling).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **127** | **37** | 164 |
| | | increase | **55** | **49** | 104 |
| | | | 182 | 86 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, 1000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **134** | **30** | 164 |
| | | increase | **57** | **47** | 104 |
| | | | 191 | 77 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, 1000 features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **131** | **33** | 164 |
| | | increase | **53** | **51** | 104 |
| | | | 184 | 84 | N 268 |

### 5.2.5 Naïve Bayes classifier

MNB gave the worst results using the BOW approach, and using topic modeling did not help with this. Table 25. shows the classification results for MNB classifier. The best results were obtained using the n-gram range 1-2 (AUC=0.6095, accuracy=60.82). Limiting the features was clearly not an efficient choice when using MNB, because in each n-gram case, the best results were obtained using unlimited features.

**Table 25. MNB classification results (topic modeling).**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.5835 | 0.5634 | decrease | 0.70 | 0.49 | 0.58 |
| | | | increase | 0.46 | 0.67 | 0.54 |
| | | | | | | |
| 2 | n-gram range = 1-2, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.6095 | 0.6082 | decrease | 0.71 | 0.60 | 0.65 |
| | | | increase | 0.50 | 0.62 | 0.55 |
| | | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | | |
| | AUC | ACC | class | PRECISION | RECALL | F1 |
| | 0.5902 | 0.6082 | decrease | 0.68 | 0.67 | 0.68 |
| | | | increase | 0.5 | 0.51 | 0.50 |

**Table 26. MNB confusion matrices (topic modeling).**

| | | | | | |
|---|---|---|---|---|---|
| 1 | n-gram range = 1, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **81** | **83** | 164 |
| | | increase | **34** | **70** | 104 |
| | | | 115 | 153 | N 268 |
| | | | | | |
| 2 | n-gram range = 1-2, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **99** | **65** | 164 |
| | | increase | **40** | **64** | 104 |
| | | | 139 | 129 | N 268 |
| | | | | | |
| 3 | n-gram range = 1-3, unlimited features | | | | |
| | | | | | |
| | | | predicted class | | |
| | | | decrease | increase | |
| | True class | decrease | **110** | **54** | 164 |
| | | increase | **51** | **53** | 104 |
| | | | 161 | 107 | N 268 |

Corresponding confusion matrices are given in Table 26. As with the BOW approach, the ability to predict the decrease class was the worst also with topic modeling. With the BOW approach, the prediction performance was better for the decrease class and equal for the increase class.

## 5.3 Summary of the results

To conclude, SVM was the best classification model used in this study. The best results, using both the BOW approach and topic modeling, were obtained using it. The highest AUC and accuracy scores, 0.6744 and 69,4%, were obtained using the BOW approach, the n-gram range 1-3 and 2000 features. The second-best results were obtained using logistic regression, the BOW approach, the n-gram range 1-3 and 2000 features. The AUC and accuracy scores with this setup were 0.6617 and 68.29%, thus very close to SVM. Random forest with the BOW approach, the n-gram range 1-2 and 2000 features yielded the third best results. The AUC and accuracy scores with this setup were 0.6548 and 66.79%. The top 3 classification models in this thesis, using both the BOW approach and topic modeling, achieved their best results when the number of features was limited, and the n-gram range was either 1-2 or 1-3. Hence, limiting the number of features and using n-grams can be said to be effective methods in this kind of tasks.

Based on the results, it is clear that no significant improvement in classification performance was achieved using topic modeling. Previous similar studies that used topic modeling showed promising results, but improving the BOW-based credit rating change prediction model with topic modeling clearly failed in this study. The BOW approach provided quite modest results, probably due to overfitting (the BOW approach is known to do that) or the high similarity between the 10-k reports. Hajek et al. (2016) and Chen et al. (2017) did not compare the topic modeling classification results to other approaches, but the results in both of these studies were much better than in this study. The classification tasks in these studies were a little different, so the results are not directly comparable. However, these studies were the best available benchmarks.

Generally, the decrease class (a decrease in credit rating), was predicted significantly better than the increase class (an increase in credit rating). This result is in line with the study of Cohen et al. (2020). In the study, they find that even a small change (of any kind) in item 7 of the company's 10-k annual report is a sign of the company's financial diffi-

culties. Because 10-k reports are typically very boilerplate (vary very little between consecutive years), texts that stand out, can intuitively be assumed to have better predicting capabilities. Thus, the stand out cases are likely to be the ones that correspond to a decrease in credit rating. It seems that if a company's financial situation is good, then very little changes are made to the 10-k annual report. Correspondingly, increases in credit rating are harder to predict based on the 10-k reports.

# 6 CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER RESEARCH

The purpose of this study was to determine how well a change in a company's credit rating can be predicted using the text data in the previous year's form 10-k annual report and whether topic modeling can be used to improve the prediction performance. Different ML models were trained to perform a binary credit rating classification task, and their classification results were compared. The methods used in this thesis were for the most part similar as in the studies of Hajek et al. (2016) and Chen et al. (2017).

To conclude, SVM was the best classification model used in this study. The best results, using both the BOW approach and topic modeling, were obtained using it. The highest AUC and accuracy scores, 0.6744 and 69,4%, were obtained using the BOW approach, the n-gram range 1-3 and 2000 features. The second-best results were obtained using logistic regression, the BOW approach, the n-gram range 1-3 and 2000 features. The AUC and accuracy scores with this setup were 0.6617 and 68.29%, thus very close to SVM. Random forest with the BOW approach, the n-gram range 1-2 and 2000 features yielded the third best results. The AUC and accuracy scores with this setup were 0.6548 and 66.79%. The top 3 classification models in this thesis, using both the BOW approach and topic modeling, achieved their best results when the number of features was limited, and the n-gram range was either 1-2 or 1-3. Hence, limiting the number of features and using n-grams can be said to be effective methods in this kind of tasks.

Based on the results, it is clear that no significant improvement in classification performance was achieved using topic modeling. The BOW approach provided quite modest results, probably due to overfitting (the BOW approach is known to do that) or the high similarity between the 10-k reports. Hajek et al. (2016) and Chen et al. (2017) did not compare the topic modeling classification results to other approaches, but the results in both these studies were much better than in this study. The classification tasks in these studies were a little different, so the results are not directly comparable.

The classification results are quite modest, but still promising. When developing the classification model, numerous choices were made that could be made differently. For example, different over/undersampling techniques could be tried out. In addition, some of the methods chosen were quite straightforward and could certainly be developed to obtain better results. For example, the NLP techniques used for text preprocessing could be improved manually to be more efficient. This means, for example, more advanced

logic for tokenizing and stemming rules and selecting the stop-words. Also, the form 8-k data used by Chen et al. (2017) would certainly be worth trying in a similar task.

The method used in this study to optimize hyperparameters is somewhat naïve. It is quite unlikely that a global optimum can be found among discrete predetermined hyperparameter values. In addition, this method is computationally expensive. Thus, a more advanced solution could be used to replace it. However, this method certainly produced good, indicative results on how accurate predictions can be produced under these given conditions. The same applies to the topic modeling in this study; the search for the optimal number of topics was quite rudimentary. Other topic modeling approaches could also be tried.

In the future, the methods used in this thesis could also be used for other research purposes. One interesting area of research could be how the presence of certain topics in a company's sustainability report, integrated in an annual report, corresponds the company's credit rating. In this way, it would be possible to determine whether a company's sustainability risks are reflected in the company's creditworthiness. In addition, it would be interesting to implement the methods used in Cohen et al. (2020) in a binary credit rating classification task. In this way, it could be found out, whether a small change to the language and structure of a company's 10-k report would manifest itself in a future credit rating change. In this thesis, topic modeling was used only as a dimensionality reduction method. A profound, qualitative analysis of the topics extracted from 10-k reports could also be a study worth conducting.

# REFERENCES


Ali, Shawkat. (2008) Support Vector Machine: Itself an Intelligent Systems. In: *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications,* eds. Kristin Klinger – Kristin Roth – Jennifer Neidig – Jamie Snavely – Carole Coulson, 501-522. IGI Global, London.

Alpaydin, Ethem. (2014) *Introduction to machine learning*. The MIT Press, London.

Barboza, F. – Kimura, H. – Altman, E. (2017) Machine learning models and bankruptcy prediction. *Expert Systems with Applications,* Vol. 83, 405–417.

Boelaert, J. – Ollion, E. (2018) The Great Regression. Machine Learning, Econometrics, and the and the Future of Quantitative Social Sciences. *Revue française de sociologie*, Vol. 59 (3), 475-506.

Boden, M. A. (1996) *Artificial intelligence*. Academic Press, San Diego

Breiman, L. (2001) Statistical modeling: The two cultures. *Statistical science*, Vol. 16 (3), 199–231.

Bzdok, D. – Altman, N. – Krzywinski, M. (2018) Statistics versus Machine Learning. *Nature Methods*, Vol. 15 (4), 233–234.

Cady, F. (2017). *The data science handbook*. John Wiley & Sons Inc., Hoboken, New Jersey.

Chen, Y. – Gupta, A. – Rabbani, G. (2017) Comparative text analytics via topic modeling in banking. *Paper presented at the IEEE Symposium Series on Computational Intelligence (SSCI),* Honolulu, Hawaii, November 27 – December 1, 2004.

Chen C. C. – Li, S. T. (2014) Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications,* Vol. 41 (16), 7235–7247.

Cohen, L. – Malloy, C. – Nguyen Q. (2020) Lazy prices. *The Journal of Finance,* Vol 75 (3), 1371-1415.

Elakiya, E. – Rajkumar, N. (2017) Designing preprocessing framework (ERT) for text mining application. *Paper presented at the International Conference on IoT and Application (ICIOT),* Nagapattinam, India, May 19-20, 2017.

Hajek, P. – Olej, V. – Prochazka, O. (2016) Predicting Corporate Credit Ratings using Content Analysis of Annual Reports – A Naïve Bayesian Network Approach. In: *Enterprise Applications, Markets and Services in the Finance Industry,* eds. Stefan Feuerriegel – Dirk Neumann, 47-61. Springer.

Hassan, S. Z. – Verma, B. (2009) Hybrid Data Mining for Medical Applications. In: *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications,* eds. Kristin Klinger – Kristin Roth – Jennifer Neidig – Jamie Snavely – Carole Coulson, 523-543. IGI Global, London.

Hirschberg, J. – Manning, C. D. (2015) Advances in natural language processing. *Science,* Vol. 349 (6245), 261-266.

Huang, Z. – Chen, H. – Hsu, C. J. – Chen, W. H. – Wu, S. (2004) Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision support systems,* Vol. 37 (4), 543–558.

James, G. – Witten, D. – Hastie, T. – Tibshirani R. (2013) *An introduction to statistical learning: with applications in R.* Springer.

Jurafsky, D. – Martin, J. H. (2018) *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Draft of September 23, 2018.

Kapitanova, K. – Sang, H. S. (2012) Machine Learning Basics. In: *Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning*, eds. Qi Hao – Fei Hu, 3–29. CRC Press.

Kim, K. J. – Ahn, H. (2012) Corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach. *Computers & Operations Research* Vol 39 (8), 1800-1811.

Kuhn, M. – Johnson, K. (2013) *Applied predictive modeling (Vol. 26).* Springer, New York.

Le Roux, N. – Bengio, Y. – Fitzgibbon, A. (2012) Improving first and second order methods by modeling uncertainty. In: *Optimization for Machine Learning,* eds. Suvrit Sra – Sebastian Nowozin – Stephen J. Wright, 403-429. MIT Press, London.

Marsland, S. (2014) *Machine Learning: An algorithmic perspective, Second Edition.* CRC Press.

Martin, D. I. – Berry, M. I. (2007) Mathematical Foundations Behind Latent Semantic Analysis. In: *Handbook of latent Semantic analysis,* eds. Thomas K. Landauer – Danielle S. McNamara – Simon Dennis – Walter Kintsch, 35-55. Psychology Press.

Moody's, Moody's Rating Symbols and Definitions. <https://www.moodys.com/sites/products/AboutMoodysRatingsAttachments/MoodysRatingSymbolsandDefinitions.pdf>, retrieved 25.5.2020.

Chawla N. V. – Bowyer K. W. – Hall L. O. – Kegelmeyer, W. P. (2002) SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research,* Vol. 16 (1), 321–357.

Probst, P. – Boulesteix, A. – Bischl, B. (2019) Tunability: Importance of Hyperparameters of Machine Learning algorithms. *Journal of Machine Learning Research* Vol 20, 1-32.

Kaundal R. – Kapoor, A. S. – Raghava, G. P. S. (2006). Machine learning techniques in disease forecasting: a case study on rice blast prediction. *BMC Bioinformatics,* Vol 7, 485.

Rennie, J. D. M. – Shih, L. – Teevan, J. – Karger, D. R. (2003) Tackling the poor assumptions of Naïve Bayes text classifiers. *Proceedings of the twentieth international conference on machine learning,* 616-623.

Russell, S. J. – Norvig, P. (2009) *Artificial intelligence: a modern approach. 3. edition.* Pearson.

Samuel A. L. (1959) Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, Vol 3 (3), 210-229.

Scikit-learn, Multinomial Naïve Bayes. <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html>, retrieved 28.5.2020.

SEC, Form 10-k. < https://www.sec.gov/fast-answers/answers-form10khtm.html>, retrieved 26.5.2020.

Shin, M. S. – Kim, S. E. – Shin, J. H. (2012) The effects of credit ratings on capital structure: Evidence from Korea. *Journal of Finance & Accountancy*, Vol. 11, p12.

Shynkevich, Y. – McGinnity, T. M. – Coleman, S. – Belatreche, A. – Li, Y. (2017) Forecasting price movements using technical indicators: investigating the impact of varying input window length. *Neurocomputing*, Vol. 264, 71–88.

Standard & Poor's, Understanding ratings. <https://www.spglobal.com/ratings/en/about/understanding-ratings>, retrieved 26.5.2020.

Steinwart, I. – Christmann, A. (2008) *Support vector machines*. Springer.

Stevens, K. – Kegelmeyer, P. – Andrzejewski, D. – Buttler, D. (2012) Exploring Topic Coherence over many models and many topics. *Paper presented at the Conference on Empirical Methods in Natural Language Processing,* Jeju, Korea, July 12-14, 2012.

Steyvers, M. – Griffiths, T. (2007) Probabilistic Topic Models. In: *Handbook of latent Semantic analysis,* eds. Thomas K. Landauer – Danielle S. McNamara – Simon Dennis – Walter Kintsch. Psychology Press.

Uysal, A. K. – Gunal, S. (2014) The impact of preprocessing on text classification. *Information Processing & Management,* Vol 50 (1), 104-112.

Vluymans, S. (2019) *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods.* Springer.

Wallis, M. – Kuldeep, K. – Gepp, A. (2019) Credit Rating Forecasting Using Machine Learning Techniques. In: *Managerial Perspectives on Intelligent Big Data Analytics,* eds. Zhaohao Sun, 180-198. IGI Global.

Wu H. C. – Hu Y. H. – Huang, Y. H. (2014) Two-stage credit rating prediction using machine learning techniques. *Kybernetes* Vol. 43 (7), 1098-1113.

Yeh C. C. – Lin, F. – Hsu, C. Y. (2012) A hybrid KMV model, random forests and rough set theory approach for credit rating. *Knowledge-Based Systems,* Vol. 33, 166–172.