A COMPERATIVE STUDY OF TEXT CLASSIFICATION MODELS ON INVOICES
The feasibility of different machine learning algorithms and their accuracy

Bachelor Degree Project in Information Technology
Basic level 30 ECTS
Spring term 2018

Linus Ekström & Andreas Augustsson

Supervisor: Niclas Ståhl
Examinator: Alan Said
Abstract

Text classification for companies is becoming more important in a world where an increasing amount of digital data are made available. The aim is to research whether five different machine learning algorithms can be used to automate the process of classification of invoice data and see which one gets the highest accuracy. Algorithms are in a later stage combined for an attempt to achieve higher results.

N-grams are used, and results are compared in form of total accuracy of classification for each algorithm. A library in Python, called scikit-learn, implementing the chosen algorithms, was used. Data is collected and generated to represent data present on a real invoice where data has been extracted.

Results from this thesis show that it is possible to use machine learning for this type of problem. The highest scoring algorithm (LinearSVC from scikit-learn) classifies 86% of all samples correctly. This is a margin of 16% above the acceptable level of 70%.

Keywords: Machine Learning, Text Classification, Invoices, Supervised Learning, Information Retrieval, Ensemble learning
Acknowledgement
We would like to thank our supervisor, Niclas Ståhl, at the University of Skövde for the help and support given during the completion of this thesis. Without him, the making of this work would not have been possible. We would also like to thank the company Asitis AB, where the thesis has been written, for their support.
Contents

1 Introduction ............................................................................................................. 1

2 Background ............................................................................................................. 3
   2.1 Invoice Handling and the Current System ....................................................... 3
   2.2 Text Classification .............................................................................................. 4
       2.2.1 N-Gram-Based text classification on character level ............................. 5
   2.3 Machine Learning ............................................................................................... 6
       2.3.1 Decision Tree .............................................................................................. 7
       2.3.2 K-Nearest Neighbors (k-NN) ..................................................................... 8
       2.3.3 Bayesian Approach (Naïve Bayes) ........................................................... 9
       2.3.4 Support Vector Machines (SVM) ............................................................. 10
       2.3.5 Neural Networks ......................................................................................... 10
       2.3.6 Ensemble Learning ..................................................................................... 11
   2.4 Related work ........................................................................................................ 12

3 Problem .................................................................................................................... 14
   3.1 Aim ..................................................................................................................... 14
   3.2 Motivation .......................................................................................................... 14
   3.3 Research questions ............................................................................................ 14
   3.4 Hypothesis .......................................................................................................... 14
   3.5 Objectives ........................................................................................................... 15
       3.5.1 Work contribution ....................................................................................... 15
   3.6 Method ............................................................................................................... 15
       3.6.1 Case study .................................................................................................... 15
       3.6.2 Alternative methods .................................................................................... 16
       3.6.3 Selection of Algorithms ............................................................................. 16
       3.6.4 Data Collection ............................................................................................ 17
       3.6.5 Implementation ........................................................................................... 18
       3.6.6 Training and Testing .................................................................................. 19
       3.6.7 Comparison of Results ............................................................................... 19
       3.6.8 Validity Threats ......................................................................................... 19

4 Implementation ....................................................................................................... 21
   4.1 Data pre-processing. ......................................................................................... 21
   4.2 Setting up the Algorithms (Objective 2) ......................................................... 21
       4.2.1 Decision Tree .............................................................................................. 21
       4.2.2 K-Nearest Neighbor (k-NN) ..................................................................... 21
       4.2.3 Naïve Bayes ............................................................................................... 21
       4.2.4 Support Vector Machine (SVM) ............................................................. 22
       4.2.5 Neural Network .......................................................................................... 22
   4.3 Separated Tests (Objective 3) ........................................................................... 22
   4.4 Combined Tests (Objective 4) .......................................................................... 22

5 Results ....................................................................................................................... 24
   5.1 Presentation ....................................................................................................... 24
       5.1.1 Objective 3 ................................................................................................. 24
       5.1.2 Objective 4 ................................................................................................. 28
   Analysis ..................................................................................................................... 31
       5.1.3 Objective 3 - Andreas ................................................................................. 31
1 Introduction

The uses of text classification for companies and organizations is becoming more and more important in a world where an ever-increasing amount of digital data are made available. When handling different kinds of digital documents, one difficulty is the presence of errors, like spelling errors or grammatical errors of various kind. The handling of these errors to some extent is crucial for any type of Text Classification (TC). One way of meeting this is the use of N-gram, to provide a tolerant element to the TC.

The concept of machine learning in the use of text classification refers to the approach of automatically labeling documents or text by learning from a set of pre-classified documents. In this study the supervised learning process will be used through five different machine learning algorithms: Decision Tree, K-Nearest Neighbors (k-NN), Naïve Bayes, Support Vector Machine (SVM) and Neural Networks. Ensemble based systems, which will be used in this thesis for combination of models, are sometimes used to achieve a better result when predicting results with machine learning. By using additional opinions to make a decision results can improve.

The aim of this thesis is to see if machine learning can be used to make handling of Swedish invoices in digital form easier and also to research whether algorithms can be combined to yield a better result, a result with higher accuracy. The motivation for this is to help companies, in general, in handling large number of invoices and Asitis AB in particular, with customers handling of invoices on their platform.

To this end, there are three different research questions this study aims to answer:

1. Can machine learning be used to automatically categorize information on an invoice within the acceptable range of accuracy decided by the company?
2. Which one out of five different common machine learning algorithms can be used to solve this task with the highest accuracy?
3. Can the five different algorithms be combined to yield a better result, seen to accuracy?

For the thesis, case study has been chosen as the most appropriate methodology. In case studies, data is collected for certain purpose. Based on the gathered data, a statistical analysis can be made. The case to be examined can be any type of unit and the aim is to fathom something in that unit. The unit in this case is the classification of text from invoices.

Data will be collected from different sources and some of the data will be generated to fit the purpose. The implementation will be done using scikit-learn, a library found in Python. The algorithms will be trained on 80% of the data and tested on the remaining 20%. The results will then be compared in form of an average of total accuracy in percent, for each algorithm, over 10 seeds each.

Asitis AB develops system solutions within the financial industry, mainly in debt collection and factoring. The company was founded 2002, with the aim of improving old, unwieldy systems, developing new, internet-based systems that would revolutionize the business. Further references to Asitis AB in this thesis will be solely as the Company.
The results acquired from all the different algorithms showed that a majority of the models made it above the acceptable level of 70%. Support Vector Machine proved to be the most accurate algorithm with 86% which also was predicted in the hypothesis of the study. The ensemble learning models using voting gave promising results, not far behind the SVM. This shows that machine learning, with a 16% margin over the acceptance level, can provide a decrease in effort needed to classify information on invoices.
2 Background

This chapter contains and handles the theoretical background needed to solve and understand the problem area. The current system that needs improvement will be explained with details covering what handling of invoice data means in the case for this research. The concept of text classification is thereafter explained. Lastly, a more in-depth explanation of the different machine learning algorithms will be presented together with the theory behind every method that will be used in this paper.

2.1 Invoice Handling and the Current System

Invoices contain, in most cases, important information, which makes the handling of said information an important task and the need to store this information grows. In order to minimize the administrative task of handling this, an easy way to automate the retrieval and storing is desired. To make matters more complex, most invoices differ from company to company.

An invoice contains different types of data, ranging from name of the invoice sender and receiver to VAT amounts, invoice rows and organization number. Moreover, it also contains dates and addresses. All these different kinds of data poses challenges when extracting and classifying the information contained in an invoice. In order to simplify the administrative effort surrounding the transfer of data from the invoice, whether in PDF- or XPS-form, into a database for storage, text classification might be a solution. Using text classification could provide the administrator with an automated tool to prefill the necessary fields (Figure 1) before storing the data in the database. With this help, the time spent on each invoice before storing its data could be decreased and thus enable the administrator to work more efficient and process more invoices in a shorter time.

The current system used in Asitis Financial System (AFS) does not use text classification but utilizes a text extraction feature. Hence the information in the PDF representation on the invoice is extracted but no classification is being done after the extraction. This makes the administrative tool blunt and the need to manually process each field of data necessary at the moment. If text classification could be implemented on this data, the manual process would decrease, and the appeal of the administrative application would very much increase.

In a perfect world, all fields shown in Figure 1 would be prefilled thanks to the use of text classification. This would leave the administrator with the sole task of inspecting the fields to check for accuracy and then register the invoice, which would save a lot of time. But, even a scenario where most fields are correctly prefilled would make an impact in the overall time spent on each invoice and therefore be an improvement to the current system.
2.2 Text Classification

Text classification (TC) is also known as text categorization or topic spotting and its uses in information retrieval has grown in large quantity, due to an ever-increasing number of documents in digital form (Sebastiani, 2002). Sebastiani, 2002, pp. 2–3 defines TC as:

Text categorization is the task of assigning a Boolean value to each pair \((d_i, c)\) \(\in D \times C\), where \(D\) is a domain of documents and \(C = \{c_1, \ldots, c|c|\}\) is a set of predefined categories. A value of \(T\) assigned to \((d_i, c)\) indicates a decision to file \(d_i\) under \(c\), while a value of \(F\) indicates a decision not to file \(d_i\) under \(c\).

Up until the late 1980s, TC was most often approached using domain experts or knowledge engineers, at least in real-world applications, where manually created rules were used to classify documents. This approach was considered costly and time consuming, hence the approach lost in popularity to machine learning approaches, where pre-classified documents
are used to automatically build an automatic text classifier. The accuracy from these automatic processes were corresponding to the accuracies achieved by human experts. This was a noticeable gain, using machine learning strategies, since using expert labor to intervene in constructing the classifier was not needed.

There are different incidences regarding text classifications. One of them are the use of either Single-Label or Multilabel text classification. The single-label classification aims to assign exactly one label to each document in a domain of documents whilst the multilabel classification means that more than one label can be assigned to the same document. Moreover, there is a special case of single-label called binary text classification, where each document must be assigned to either one category or its complement. The repeated binary text classification is more general than the multilabel TC, which also is true for the single-labeled, since binary TC also can be used for multilabel TC.

Another incidence to consider in TC is the Category-Pivoted versus the Document-Pivoted TC, which is two different approaches in using a TC. Category-Pivoted looks at all the different categories in a set of categories and tries to find all documents in a document set to be filed under each category. Document-Pivoted, on the other hand, starts from the other side and looks at all the documents in a set of documents and aims to find all categories in a set of categories to file it under. The Document-Pivoted is more suitable when documents becomes accessible at different times, for instance, when TC is used in an e-mail filter. The Category-Pivoted TC is a more suitable choice when a new category is to be added to the current set of categories, after documents already been classified using the set of categories and these documents need re-classifications.

A final incidence to consider is the “Hard” Categorization versus the Ranking Categorization. Using Ranking is a good way to assist a human expert to take the final decisions in categorization in a system with partial automation of TC. By ranking the categories in order of appropriateness the human expert can look at the top choices of categories before making a decision, which saves time and effort, not having to browse all categories in order to find the most appropriate one. Another way to assist the human expert would be the “Hard” categorization, where the ranking is done on the documents in the set of documents in regard to their appropriate fit to each category in the set of categories. This kind of semiautomated classification is very useful in applications where a fully automated system would yield worse result than the result of a domain expert or another human expert, especially if the application is critical or if the quality of the training set might be low or not complete (Sebastiani, 2002).

In this thesis single-label text classification will be used, as well as hard categorization.

2.2.1 N-Gram-Based text classification on character level
When handling different kinds of digital documents, one difficulty is the presence of errors, like spelling errors or grammatical errors of various kind. The handling of these errors to some extent is crucial for any type of TC. One way of meeting this is the use of N-gram, to provide a tolerant element to the TC.

A lot of the digital documents that are handled in various systems have the benefit of being controlled and checked in an automated way but also manually. Other documents do not have this kind of scrutiny, which put them at risk of containing different kinds of errors and using an N-Gram-based TC can benefit greatly and reduce the time and money spent on manual inspection and processing (Cavnar & Trenkle, 1994).
Another use of N-Gram-based TC is when there is a need for automated processing of digital documents. By applying this approach in TC, the expectation is that a more accurate result can be achieved, improving the performance of the TC in that regard.

A key problem in TC is feature representation, which often is based around a model called Bag-of-Words (BoW), where N-grams often are used as features. A challenge when using N-grams is the fact that they often ignore conceptual information. This can be a problem and yield different results, depending on the value of N (Lai, et al., 2015). For instance, in an address line with the street name “Per Anders Gata”, if N is set to one, a unigram, and it analyzes the different parts in the street name one by one, the model will most likely classify “Per” and “Anders” as two surnames. If instead a trigram was used, taking all three parts of the street address into account, it would more likely be able to identify the string as a street address. The same principle goes when N-gram is applied to letters (characters), instead of words.

<table>
<thead>
<tr>
<th>N-gram Type</th>
<th>Sample Sequence</th>
<th>N-gram Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-gram</td>
<td>Asitis</td>
<td>A, s, i, t, i, s</td>
</tr>
<tr>
<td>2-gram</td>
<td>Asitis</td>
<td><em>A, As, si, it, ti, is, s</em></td>
</tr>
<tr>
<td>3-gram</td>
<td>Asitis</td>
<td>_ <em>A, <em>As, Asi, sit, iti, tis, is</em>, s</em> _</td>
</tr>
</tbody>
</table>

Table 1 Table showing three types of character-based N-grams where N is set to be 1, 2 or 3.

The table shown (Table 1) gives three examples of character-based N-gram sequences of the word “Asitis”. When N is set to the value 1 the text will be split up into sequences containing only one character, when N is 2 the sequences contains two characters and so on. If N is increased the possibility of fitting full words into one sequence rises and therefore approaches the word-based method. At the same time, a smaller value of N increases the chance of finding smaller similarities in the sample sequence.

For a more comprehensible understanding of N-Gram-Based text classification, refer to Cavnar & Trenkle (1994).

2.3 Machine Learning

The concept of machine learning (ML), through supervised learning (explained later in this chapter), in the use of text classification refers to the approach of automatically labeling documents or text by learning from a set of pre-classified documents (Sebastiani, 2002). This is done by selecting a few characteristics or features (the latter term will be used in the rest of this paper) that should be investigated, find some correlation or relationship between them and from this predict a new outcome (an existing classification in this case) when new data are presented to the model. The method can be compared to other methods, like rule-based learning or knowledge engineering, explained in the previous chapter (2.2). Sebastiani (2002) writes that ML has become a more used approach for solving TC since the ‘90s but that it still is used the most in the research community. Although, this may not be the case at the time of this thesis being written, 16 years later. This transit from rule-based to ML has led to effort moving from classification of documents to the engineering of systems that will learn from pre-classified data and therefore making the process more effective. This has disadvantages in the form of the need for existing data to learn from. Sebastiani (2002) does
not see this as a problem in most cases because of companies already having access to previously classified documents that can be used in the new process. However, this is a problem for new companies where data have not yet been acquired and classified.

When using ML through supervised learning to solve a problem, the existing data (which are needed) is split into two parts: one for training and one for testing. The former set is used to “teach” the classifier by looking at the existing characteristics and the latter is used to test the accuracy of the final model. Because of the already classified documents, new predictions can be compared to these and therefore be used to see how effective the results are. It is important to know that the documents in the test set cannot, in any way, take part in the construction and training of the classifier (Sebastiani, 2002).

When machine learning is used to classify text, or data in general, where the desired output is known, it can be categorized as supervised learning. This is explained by Raju, et al. (2017) as “[...] the learning process is supervised by the knowledge of categories and of the training instances belongs to them.” which can be seen in contrast to unsupervised learning where the categories are unknown and not shown to the model. In this study the former learning process will be used through five different machine learning algorithms. Each one of those have different approaches to solving the classification problem and will be explained in the following parts more detailed. The algorithms have been chosen from the comparative analysis by Raju, et al. (2017) of these specific methods on text classification.

2.3.1 Decision Tree

The decision tree (DT) used for text classification is a tree where internal nodes are labeled by terms and leaves are labeled by the categories that will be used (Sebastiani, 2002). The branches in the tree are determined by the weight the term has in the test data. The classifier categorizes text by recursively going through labels and their weight until a leaf node is reached, and therefore reaching a classification that can be predicted.

**Figure 2** Illustration showing how a decision tree decides whether a text can be classified as being about wheat or not. It is represented as a binary tree where underlining means negation of the term (“WHEAT” = Not classified as being
about “wheat”). The illustration is a simplification of Fig. 2 (Sebastiani, 2002) pp. 22.

Sebastiani (2002) states that most of these trees are built as binary trees and can therefore be illustrated as in Figure 2. The algorithm tests each weight of the words in the text (in this case frequency of words are used as feature) and recursively tests if it is present or not until a leaf node is reached. As Figure 2 shows the text can be classified as being about either the term “wheat” or not depending on the words and their frequency in the data. For example, the sentence “The wheat that grows in the field weighs several tonnes”, would be classified as a text about wheat. The sentence contains the word wheat but not farm. It does not contain the word agriculture but it does contain tonnes, leading it to the correct classification. By using decision trees, it can easily be comprehensible by humans where a visualization of decisions can be presented. This can be of great value where it can give insight in many practical problems (Johnson, et al., 2002).

There are three clear benefits of using decision trees, in addition to the comprehensibility by humans (Raju, et al., 2017): 1) It is able to handle many kinds of data; there is support for classification of nominal, numeric and textual data. 2) It can process datasets containing errors and missing data. 3) Decision trees are available for many different platforms for data mining and text classification.

2.3.2 K-Nearest Neighbors (k-NN)

K-nearest neighbors or k-NN is a form of example-based classifier. These do not build or “learn” a representation of each category; they simply rely on the already existing data from the training set and classify new data from looking at data points (already known by the model from training) with similar features (Sebastiani, 2002). The number of existing data points that will be looked at when predicting a new outcome is decided by the developer, therefore the “k” in k-NN where it represents the number of “neighbors” (data points with similar features) that should be used to classify a new data point.

![Figure 3](image-url) Graph showing how a new data point (shown as an “X”) can be classified using k-NN. White dots are showing a data point classified as true and black points are classified as false. The area surrounding the new data point marks the area the model should “look at” (k = 3).
The graph in Figure 3 can be used of the same problem as Figure 2 illustrates for decision trees. The white points represent text classified as being about “Wheat” and the black points are not. When the new text (“X” in the graph”) is to be predicted the model looks at the closest “neighbors”, which in this example is decided to be three. The majority of these points are classified as true (classified as “wheat”) in the graph and therefore the new data point will be categorized as a text about “wheat” as well. This can be seen as a voting process where every chosen neighbor votes on a classification (its own category), weighted by similarities to the new data point (Bijalwan, et al., 2014). In this example Euclidean distance is used to decide similarity between points because of its simplicity in deciding nearest “neighbors” (Raju, et al., 2017). As a guideline, an uneven number of neighbors should be used, in order to avoid a draw.

Raju, et al. (2017) describes the method as “[...] non-parametric, effective, easy for implementation” but that the key for it to work effectively is the availability of a similarity measure to identify close neighbors.

2.3.3 Bayesian Approach (Naïve Bayes)

The bayesian approach is a probabilistic approach where a classification is decided from the probability that the new data point is a part of category “C”. To compute the probability Bayes’ Theorem is used, given by (Sebastiani, 2002)

$$P(C|D) = \frac{P(C)P(D|C)}{P(D)}$$

The theorem can be interpreted as $P(C|D)$ being the probability of a document being classified as $C$ given the features of the text $D$. To solve the equation different probabilities have to be solved. Both $P(D)$ (probability that the text will have the specific features of $D$) and $P(D|C)$ (probability of having specific features given being categorized as $C$) are difficult because of the many combinations of features in $D$, though this can be solved if random variables in $D$ are seen as statistically independent (Sebastiani, 2002).

A machine learning algorithm using this theorem is the Naïve Bayesian approach. The algorithm uses the Bayes’ Theorem to predict, through probabilities, a classification for new text. It is naïve because of the assumptions of independence of variables. The result of this assumption is that order of features does not matter, and one feature does not affect other features in any way (Raju, et al., 2017). These assumptions of the algorithm have made it one of the worst performing methods in many tests (Rennie, et al., 2003). It is though, still used frequently because of its simplicity and easy implementation.

Rennie, et al. (2003) have researched the poor performance of the algorithm and have shown that transformations of the method can be applied to make it perform as good as other state-of-the-art classifiers. All this without making the algorithm slower, which from the start is one of Naïve Bayes strong features. One of the solutions presented by Rennie, et al., 2003 is to introduce “complements classes” to get around a bias effect where some classes have more training examples than others. Their solution also makes the assumptions of independent features in the algorithm fewer. Because of these solutions, the Naïve Bayes algorithm can still be seen as a relevant method to classify text. This can be seen in other recent studies (Larsson & Segerås, 2016). According to this paper, “[...] Naïve Bayes was able to automate the process of invoice handling”. Although this only categorized into one of two categories and the authors state that there is a need for big amounts of training data for it to be accurate.
2.3.4 Support Vector Machines (SVM)

The method can be described as organizing data, correlated with each other, into linearly separable categories (Raju, et al., 2017). Linear in the sense of SVMs can be seen as a linear method in a high-dimensional feature space (Hearst, et al., 1998). Hearst, et al (1998) explains the special properties of SVMs as being able to handle complex algorithms for nonlinear data by seeing it as a linear algorithm. The potentially nonlinear input space (meaning the space of possible input values to the model) is mapped to features which can be put in linearly separated hyperplanes (Khan, et al., 2010).

![Figure 4](image)

**Figure 4** Mapping of nonlinear input data from the input space to the high dimensional feature space where they can be split linearly (Khan, et al., 2010) pp. 12.

The SVM tries to maximize the margin or the optimal separating hyperplane (OSH) (Khan, et al., 2010) between the different classifications. The optimal separation is achieved by finding a hyperplane that separates the two classes and has the largest distance to the closest data points of both classes in the space. However, this linear version of the SVM can be switched out for other, so called, kernels to change the behavior of the algorithm (Hearst, et al., 1998). A different kernel can be used, for instance a polynomial kernel, to split the different features nonlinear. This can be very useful in cases where data cannot be separated by a linear hyperplane. When a kernel is used the data is first taken to the kernel before it gets presented to the SVM, making the data filtered in a different way.

Khan, et al. (2010) states from their comparative study that SVM in the most cases achieves the highest classification precision but that the method is very time consuming because of many parameters and a demand for computation time. This result is from a comparison with k-NN and Naïve Bayes' on binary classification tasks and according to the authors the performances of the different methods are comparable; this makes it interesting to study how it will perform in a comparison on invoice data.

2.3.5 Neural Networks

Neural networks (NN’s) can be seen as networks of different units split up into input and output units which are connected with edges representing relations and weights of terms in text classification (Sebastiani, 2002). The process of categorizing a document being used as input for the network and its weights are loaded into the input units. These units propagate the features forwards through the layers taking different edges depending on the values and their weights. A final output layer is, at the end, reached and a classification is chosen.

10
Different hidden layers can be used between the input and output layers to handle different assigned tasks, for example handling noise and blur in image recognition or spelling errors in text classification. These hidden layers can filter information sent through the network and result in a more precise classification by output layers. Figure 5 illustrates how input can flow through the NN.

![Figure 5](image.png)

**Figure 5** Illustration showing the flow of decisions in a neural network. Input goes to the input layer, propagates further on the edges to the hidden layers where edges are chosen depending on weight of the features in input. This is taken to the output layer to get a final classification.

The algorithm can be categorized as a self-adaptive method, meaning the model being able to modify and adjust the weights by itself without any given specification (Raju, et al., 2017). A common way of “teaching” the model is by using a method called error back propagation where documents are given to the input layers. If an incorrect classification occurs the error is “backpropagated” to change parameters in the network and therefore minimize faults in the future (Sebastiani, 2002).

An advantage with NN’s is the ability to handle data containing high-dimensional features and data containing faults. The disadvantages, on the other hand, are the high computing cost and the complicated structures and theories behind it which makes it hard to understand for the average user (Khan, et al., 2010).

There are many different approaches to using neural networks, for many different tasks (not only text classification), as explained by Lai, et al. (2015) where a model called Recurrent Convolutional Neural Network (RCNN) is used. The results from the study shows that this model outperformed all of the tradition methods, such as SVM’s. This method, however, leaves the bag of words (BoW) -features which involves the use of n-grams. Different layers are instead used to understand each word and their context.

### 2.3.6 Ensemble Learning

Ensemble based systems, which will be used in this thesis for combination of models, are sometimes used to achieve a better result when predicting results with machine learning. By using additional opinions to make a decision results can improve, just like in the real world when asking several doctors for opinions before surgery or reading reviews before purchasing a product (Polkar, 2006). In ensemble learning each ML algorithm can be seen as an expert
where a hypothesis is made for each data to be classified. Several experts are then put together and a final agreed hypothesis (result) is made.

There are many types of ensemble learning. In this thesis voting and stacking will be used where the former meaning exactly what the name implies; simply letting the algorithms vote on their “choice” with highest probability. The latter uses a kind of meta-classifier to determine the result of the used algorithms. According to Polkar (2006), this method lets the algorithms first decide their output, a second layer containing an additional classifier thereafter uses the output to decide a final decision.

According to Khan, et al. (2010), ensemble learning techniques (or Hybrid techniques as they call it) can be used to improve the performance of individual classifiers. Some mechanisms are explained for building such models (beyond the use of several different methods, explained earlier in this subchapter) where different subsets of training data are used within single learning methods and different parameters are used for training.

Khan, et al. (2010) describes a specific case of ensemble learning for text classification where Naïve Bayes is used at the front end to vectorize the data combined with a Support Vector Machine in the back end to classify the text document to the right category. This has been proven to increase the accuracy over using only the Naïve Bayes model. Overall, the authors claim that ensemble learning has, from earlier research, been proven to outperform individual models in most cases.

### 2.4 Related work

This sub-chapter surveys previous work in text classification and machine learning. There has been much work done in these two respective fields, although little work has been done in regard to its uses in invoices specifically.

In the thesis-paper Automated invoice handling with Machine learning and OCR (Larsson & Segerås, 2016) two OCR-engines where evaluated. Text matching was applied on raw text and the possibilities of using machine learning to automatically process invoices, where ML was used to validate invoices, was examined. The conclusion of their thesis shows that the prototype using machine learning with Naïve Bayes was able to automate the handling of invoices in a satisfying way and it was able to determine if an invoice was correct or not. The prototype in the thesis examined if the invoice as a whole was correct. This is something that this thesis aims to examine deeper, by trying to classify each part of the invoice correctly.

Earlier work has been done comparing different machine learning algorithms. Khan, el at. (2010) did a comparison of different methods and analyzed different selections of features and classification algorithms. They also explore the possibilities of combining different algorithms as hybrid approaches. The conclusion of the research shows that different techniques are better in different cases. According to Khan, et al. (2010) naïve bayes performs well on spam filtering and email categorization while SVM has shown promising results on most of the data sets. Though, it becomes clear that parameter tuning, and kernel selection makes it hard to get state-of-the-art results using SVM’s. The study concludes that k-NN performs well but that classification time might be a problem and that the value of k has to be decided.

Raju, et al. (2017) have compared the specific ML algorithms that will be used in this thesis. Conclusions made by this paper states that SVM outperforms all other evaluated supervised
algorithms for text classification, it has a higher accuracy and can adjust parameter settings. Table 2 shows the conclusions (generalized) made by Raju, et al. (2017).

<table>
<thead>
<tr>
<th>ALGORITHM USED</th>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>It learns very fast compared to Neural Networks. Easy to code.</td>
<td>It has trouble dealing with noise. It is very expensive.</td>
</tr>
<tr>
<td></td>
<td>Reduce problem complexity.</td>
<td></td>
</tr>
<tr>
<td>K- Nearest Neighbor</td>
<td>It achieves very good results and scales up well with the number of</td>
<td>It requires more time for classification.</td>
</tr>
<tr>
<td></td>
<td>documents.</td>
<td></td>
</tr>
<tr>
<td>Bayesian Approach</td>
<td>It is simple Classifier which works very well on numerical and textual</td>
<td>Low classification performance. Performs very poorly when features</td>
</tr>
<tr>
<td></td>
<td>data.</td>
<td>are highly correlated.</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>High dimensional input space. Many of the text categorization</td>
<td>Is It very time consuming because of more parameters and requires</td>
</tr>
<tr>
<td></td>
<td>problems are linearly separable. Performance is very high.</td>
<td>more computation time.</td>
</tr>
<tr>
<td>Neural Network</td>
<td>It is used in recognizing complex patterns and performing nontrivial</td>
<td>It is very hard to understand. Slow classification technique.</td>
</tr>
<tr>
<td></td>
<td>mapping functions. It is used in statistical modeling.</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Table showing conclusions, in the form of pros and cons, made about the five different machine learning algorithms (Raju, et al., 2017), pp. 1616.

Conclusions has been done on comparisons of different techniques, and the ones presented in Table 2 in particular. The methods have never been tested and compared on invoice data, therefore this is an interesting area of research where new results can be acquired.
3 Problem

This chapter provides details regarding this thesis aim and motivation. It also provides the research questions to be answered and the hypothesis, before the different objectives are listed and the chosen method is presented.

3.1 Aim

The aim of this study is to see if machine learning (ML) can be used to make the handling of invoices in digital format easier. In order to narrow the scope, the invoice data used will be in Swedish. Five different commonly used methods of ML will be used on already extracted text to compare their accuracy and investigate if they can be seen as feasible for the task. The thesis also aims to research whether algorithms can be combined and yield a better result with a higher accuracy. An acceptable result (accuracy) which the study aims to reach is where the automatic classification makes the work more effective; an accuracy higher than 70%, decided together with the Company, where the research is being conducted.

3.2 Motivation

The motivation for this thesis is to help companies handling large amounts of invoices (and other similar documents) in general, and in particular the Company with their customers handling and registration of invoices on the platform. Today information has to manually be registered into the system from information on invoices. This can be a very time-consuming task. Therefore, the use of machine learning could transform this into a much more effective process where data is classified into fields required for registration automatically, which can lead to large cuts in cost and time spent on administrative tasks.

3.3 Research questions

There are three different questions this study aims to answer:

1. Can machine learning be used to automatically categorize information on an invoice within the acceptable range of accuracy decided by the company?
2. Which one out of five different common machine learning algorithms can be used to solve this task with the highest accuracy?
3. Can the five different algorithms be combined to yield a better result, seen to accuracy?

3.4 Hypothesis

The hypothesis for this study is that machine learning will simplify, that is, lessen the manual efforts required in the handling and registration of invoices. This means that the results gained from at least one model in the case study will achieve an accuracy of at least 70%, which has been discussed with the Company as an improvement over the current system. From the background, presented in chapter 2, it is expected that either SVM or neural networks with use of the methods presented by Lai, et al. (2015) will achieve the highest accuracy based on earlier results when comparing the chosen methods. An ensemble of several algorithms is thought to increase the accuracy even more, based on the findings of Khan, et al. (2010). The combination of the highest performing algorithms in the ensemble methods should perform
with higher accuracy than the ones using all five together due to faulty prediction of the worst performing algorithms being left out.

Because of the spread of different data types on invoices all fields such as amounts of money and dates might be difficult for a ML algorithm to classify correctly because of their non-correlational nature.

### 3.5 Objectives

To complete the study, different objectives have to be completed:

1. Research the problem through the literature written on the area.
2. Build the models using the five different machine learning algorithms in Python.
3. Run the different algorithms; train and test them on a dataset containing data present on invoices.
4. Combine the different algorithms and run the same tests on the same dataset.
5. Analyze and present the results from the different algorithms (both separated and combined).

#### 3.5.1 Work contribution

<table>
<thead>
<tr>
<th>Objective</th>
<th>Contributor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Andreas &amp; Linus</td>
</tr>
<tr>
<td>2</td>
<td>Andreas &amp; Linus</td>
</tr>
<tr>
<td>3</td>
<td>Andreas</td>
</tr>
<tr>
<td>4</td>
<td>Linus</td>
</tr>
<tr>
<td>5</td>
<td>Andreas &amp; Linus</td>
</tr>
</tbody>
</table>

**Table 3** Contributions to the different objectives done by the participants of this thesis.

### 3.6 Method

This subchapter presents the method used in the thesis. First, the chosen method, *Case study* will be detailed, then the grounds for the selection of the algorithms will be presented. After that, the specifics regarding data collection, as well as the training and testing of the data will presented. Later, the specific details on how the results will be compared are described and finally different validity threats and their relevance will be discussed.

#### 3.6.1 Case study

The chosen method for this thesis is case study. A case study project in software engineering is:

> an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software
engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified.

Wohlin, et al., 2012, pp. 10

In case studies, data is collected for certain purpose and based on that data, a statistical analysis can be made (Wohlin, et al., 2012). The case to be examined can be any type of unit and the aim is to fathom something in that unit (Berndtsson, et al., 2008). The unit in this case is the classification of text from invoices.

In a software engineering setting, case studies can be used to evaluate in what way a certain phenomenon occurs, but it can also be used to evaluate differences between different methods. For the relevance of this thesis, it means that a case study can be used to examine which algorithm or algorithms is best suited to classify text from invoices.

3.6.2 Alternative methods

In many ways, a case study bears resemblance to Action research, however, where a case study is purely observational, action research actively involved in trying to change a process (Reason & Bradbury, 2001). If researchers are active in improvements made, the method could be characterized as action research but when researcher simply study the results of changes, the methodology is considered to be a case study. Since this study does not aim to actively change any process, simply observe the results, action research was discarded as a potential methodological approach.

The differences between a case study and an Experiment might seem small but if the study is more of a controlled nature, the methodology is to be considered experiment, since the case study is observational (Wohlin, et al., 2012) and that observational factor is something considered more suitable in this case. The aim of the research conducted in this thesis is to feed data to the different machine learning algorithms and just observe their performance, in the form of accuracy.

A survey is most often used before a new technique has been introduced or after said technique has been applied in a certain area, in order to get the status and perception of its assets and liabilities (Wohlin, et al., 2012). This methodology was considered to miss key aspects in this study, yielding it difficult, if not impossible, to draw any real conclusions from it and therefore making a survey not feasible to use in the scope of this study.

The case study has been preceded by a literature search, in order to identify suitable machine learning algorithms.

3.6.3 Selection of Algorithms

There have been five different machine learning algorithms selected for comparison in this case study:

- Decision Tree
- K-Nearest Neighbor (k-NN)
- Naïve Bayes
- Support Vector Machine (SVM)
- Neural Network
The reason for choosing these five algorithms is based in their frequent occurrences in earlier research (Khan, et al., 2010) (Raju, et al., 2017) (Sebastiani, 2002). Comparisons have been made on these specific methods throughout different tests and on different data sets. Their different properties, explained by Raju, et al. (2017), makes them interesting for comparison where strong and weak sides of each algorithm can be found when invoice data is used. In earlier research text classification have been done with these algorithms on different kinds of data, but never specifically on data fields present on invoices.

When results have been collected from the algorithms these will be combined (objective 4). The combination will be selected from the algorithms with the highest accuracy. If it is possible to see that one algorithm has a high accuracy for some classifications and another for different classifications a combination of these can be made to see if (total) accuracy increases. The three highest scoring algorithms from objective 3 will be combined and a combination of all methods will be tested. These two combinations of algorithms will be combined using both voting and stacking classification (Polkar, 2006) to see if the different techniques get different results. Soft voting will be used – meaning the probabilities from each algorithm will be used to decide the outcome in the voting case. As meta classifier for the stacking method logistic regression will be used for simplicity.

Earlier research (Khan, et al., 2010) (Raju, et al., 2017) (Sebastiani, 2002) have explained and compared other algorithms, besides the five selected for this thesis. By looking at results done by these researchers conclusions can be made that the five selected methods are a selection of the most popular algorithms and have shown the most promising results in many classification tasks. Therefore, other algorithms could be excluded from this thesis.

### 3.6.4 Data Collection

Data for the case study will be collected from three different sources to build the dataset to use. As a guideline for what data to use template data of invoices will be used, taken from the Company. This template data consists of rows on Australian invoices. This research aims to study how data on Swedish invoices are handled, therefore data from Statistiska Centralbyråns (SCB) will be added for cities and common Swedish names. For street names in Sweden, data will be collected from OpenAddresses. Because the invoices at the Company often are read directly from digital format where every field/row is read, titles for different fields have to be added to the dataset. This can include, for example, the text string “Street Name” which is used as a title before the actual street name for the invoice receiver. A single dataset with rows from all different sources will be built to form a set that can be split for training and testing.

When data is added to the set a manual classification will be done. This is done to be able to compare to actual classifications when testing of the algorithms are being done, but also to teach the models the correlation between data and actual classifications. The classifications that will be used and tested on for this research are 17 different and can be seen in Appendix A. (numbers represent the number that will be used as category during implementation).

The classification for 17 (other) will be used for data that does not need to be classified as a specific category, for example the titles for fields on the invoice.

---

3.6.5 Implementation

To test the selected algorithms the programming language Python will be used together with the open source library scikit-learn, presented by Pedregosa, et al. (2011). The tool was selected because of the simplicity in testing the chosen algorithms, and in handling the data set for training, testing and splitting data in a correct way (explained further in 3.6.6). Pedregosa, et al. (2011) explains scikit-learn as a library which “[...] exposes a wide variety of machine learning algorithms, both supervised and unsupervised, using a consistent, task-oriented interface, thus enabling easy comparison of methods for a given application.” The authors also claim that the library, easily, can be used as building blocks for many different use cases. Algorithms chosen for this research will be tested using the following settings in scikit-learn (Internal settings for each algorithm has been tested and the highest performing settings, seen to accuracy, has been chosen. This was in the most cases the default parameters):

- **Decision tree** – sklearn.tree.DecisionTreeClassifier
- **K-Nearest Neighbor (k-NN)** – sklearn.neighbors.KNeighborsClassifier
  - The k for this algorithm will use the standard value provided by the library which is five.
- **Naïve Bayes** – sklearn.naive_bayes.MultinomialNB
  - The Multinomial Naïve Bayes will be used because of its good performance in text classification and the use in earlier studies (Rennie, et al., 2003).
- **Support Vector Machine (SVM)** – sklearn.svm.LinearSVC
  - The Support Vector Classifier with a linear kernel will be used because of its simplicity. It has also shown promising results earlier (Hearst, et al., 1998). Parameter dual optimization will be set to false due to fewer features than samples used (Scikit-learn, 2017), penalty will be set to ‘l1’ instead of default ‘l2’ because of higher accuracy in this case. The rest of parameters will be used with default values.
- **Neural Network** – sklearn.neural_network.MLPClassifier
  - A multi-layer perception classifier which is available in scikit-learn. The method uses backpropagation to learn. Default parameters from the library will be used.

To implement the combinations of algorithm for objective 4 a different library for Python will be used, called mlxtend, containing functions to implement both voting and stacking:

- **Voting** – mlxtend.classifier.EnsembleVoteClassifier
  - Soft voting will be used as parameter setting beyond the default settings. The meaning of this is that the probability of each prediction will be used to vote, not the actual hard result.
- **Stacking** – mlxtend.classifier.StackingClassifier
  - Default parameters will be used. The meta-classifier used for this algorithm will be a logistic regression model because of its simplicity.

N-grams has been selected as the feature for the data based on the background theories. In this case the n-grams will be selected on character-level, meaning different combinations of letters in text will be used. The value of n for this study will be set to 1-4, meaning unigrams (“bag of characters”) up to four-grams will be used. This size is reasonable based on the length of text on invoices.
3.6.6 Training and Testing
The dataset will be randomly split into two parts: one set for training and one for testing. Scikit-learn will be used to make these splits. The training set will contain 80% of the data and the testing set 20%. Ten different seeds for randomizing the splits will be used for every algorithm to minimize the risks of validity threats against the study in the form of bias in the data and outliers. Even if the ten splits are randomized, the same exact splits will be used for each used algorithm to make sure that the same data is being used in training and testing.

The algorithms will be trained on the training set with n-grams as selected feature to correlate with classification. When training has been completed predictions will be done by the model on data from the test set. This can thereafter be compared to the actual classifications in the test set.

The same process will be done when combinations of algorithms have been chosen; the ensemble models will be trained and tested on the same, ten different splits.

3.6.7 Comparison of Results
From each test done with all the algorithms, both separated and combined, ten results will be acquired. A mean of total accuracy will be taken from these ten results which will be seen as the “score” for each algorithm. These scores will be compared between them and results will be presented, showing if the problem has been solved or not.

Results showing how well algorithms work for different, isolated classifications will only be used when selecting which algorithms to combined for Objective 4. When doing this the results will be analyzed more in depth to see which parts to use and not to use. If no clear patterns in accuracy for different categories (Appendix B - Confusion Matrices) can be found the total accuracy will be used as a selector for the combined algorithms. This thesis aims to test the total accuracy for the used algorithms. Conclusions may be drawn from results for specific categories but the detailed results about each category will not be used as a measurement for final comparisons. Although, these results may be interesting in the future by the Company when selecting algorithms for specific data types and categories and should therefore be included as results.

3.6.8 Validity Threats
The value of a study and the result it presents needs to have a certain degree of validity in order to be accepted as a contribution to the research field in which it resides, or to be accepted by the organization or company for whom the study is conducted.

There are four different types of validity threats, as identified by Wohlin, et al. (2008); internal, external, construct and conclusion.

One threat to Conclusion Validity to be aware of is the Reliability of treatment implementation, which means that there is a risk of differing implementation between different researchers in their applying the treatment or between different times. Therefore, it is important to use the same implementation, or as similar as possible, for different treatments or at different times (Wohlin, et al., 2012). One threat against this validity is, if the equipment used to perform the tests differs. Should one computer fail to perform one or multiple tests and another is to be used, one with more memory for instance, then the training and testing time might not be reliable, even if the accuracy score still is not compromised. Also, parts of the data used in this thesis, are data other researcher do not have access to. In order to
replicate the performed tests, which means that they would have to generate their own set of data represented on an invoice.

When performing a case study, one must be aware of confounding factors and lessen the effect from these. One confounding factor to take into consideration in the execution of this specific implementation is the factor that can make it difficult to determine effects between different factors (Wohlin, et al., 2012). If one of the algorithms used in this thesis implementation yield low mean accuracy, and conclusions are drawn based on that, there is a risk that the researchers draw misleading conclusions based on this. It might be one poor performing factor that is responsible for the overall mean accuracy of an otherwise very accurate algorithms. It might also be the other way around. Nevertheless, being aware of confounding factors are paramount when performing a case study.

One threat against validity in this study, is the way the used data gets split. In order to rely on the results, it is important that the splitting of data is done in a balanced and measured way. If there is a skew between the different classes or labels, in, for example, the total amount of data, the result might not be reliable. To avoid this, it is important to balance the amount of data for each category. If one category were to contain very large number of data, for instance, if the total amount of data fields is 150 000 and the category Name represents 100 000 of these fields, then the model will learn that category very well and present an overall mean accuracy that is high. This would be misleading, since the actual performance might be much poorer. To handle this threat, it is important that the amount of data fields per category does not greatly exceed any other, and if they to, to be aware of this. Related to this threat is the use of common names present in the training data. Presenting only common names to the model could pose a problem when more unusual names are in the data.

Another threat against the validity in this study is the use of libraries from Python. We are reliant on scikit-learn, the library implementing the algorithms. The simplicity of this open source library is appealing but it also poses a threat against validity since we have no control over it, nor its implementation. It is kind of like a car. The car gets the driver where he or she wants to go, but the driver has to trust the manufacturer that the components of the car are made and mounted correctly.

The level of accepted accuracy of 70% is decided together with the company and might by non-generalizable to a larger population since that level could differ between different companies. This means that if the hypothesis of this thesis proves correct, this might not be true in other cases.
4 Implementation

This chapter explains the steps taken to complete the case study and acquire results from the objectives. The different parts go through the progression and describes the design decisions made.

4.1 Data pre-processing

A data pre-processor was built to import, classify and split the data. Where data was collected from the different categories can be seen in Appendix A - Classifications (Table 1). To do this the libraries pandas and scikit-learn (sklearn) was used in python. All data that was going to be used was placed in different sets with comma-separated variables. By doing this each set could be read separately and pre-classify this data automatically. This was done for all different categories of data. When classification of each dataset was done this was placed in one single dataset with the data collected as independent variables and the classifications (1-17) as the dependent ones.

With the use of sklearn the dataset was split into two sets; one set containing 80% of the data, representing the training data and one set containing 20%, representing the test data. This resulted in a total of two datasets containing both dependent (categories) and independent variables (text to be classified).

When the initial tests were conducted a realization was made that the categories with a small amount of records were not always put in both the training and the test set. Therefore, each category was split separately for each seed before being put together for the final training and test sets. This resulted in a guaranteed 80 against 20 percent split of each category for training and testing.

After data had been collected and split into training and testing sets the features for the data was created using n-grams. Functionalities from sklearn was used here also to vectorize the data into combinations of characters.

4.2 Setting up the Algorithms (Objective 2)

4.2.1 Decision Tree

The Decision Tree algorithm was implemented using DecisionTreeClassifier from sklearn.tree in Python. The algorithm was run using the standard parameters.

4.2.2 K-Nearest Neighbor (k-NN)

The algorithm was implemented using KNeighborsClassifier from sklearn.neighbors. After testing parameter settings with a different number of neighbors a decision was made to stay with the default parameters. Default parameter value for neighbors are five.

4.2.3 Naïve Bayes

When implementing the Naïve Bayes algorithm MultinomialNB from sklearn.naive_bayes was used. As explained in 3.6 this type of Bayesian approach was used due to its performance in earlier work.
4.2.4 Support Vector Machine (SVM)
SVM was implemented using LinearSVC from sklearn.svm. The parameters explained in 3.6 was used to implement the classifier. Penalty was set to ‘l1’ and dual optimization was set to false. Apart from these, default parameters were used.

4.2.5 Neural Network
The neural network algorithm was implemented using MLPClassifier from sklearn.neural_network. The default parameters were used. The default number of neurons in the hidden layer are 100.

4.3 Separated Tests (Objective 3)
With the data pre-processing in place and the parameter settings for the different algorithms decided, the separated tests could begin. To visualize the results, a confusion matrix was used. This made it possible to display the results from each run, and thereby show the amount of correct classifications for each class or label. The confusion matrix can be seen as a visualization of the algorithms performance (see 5.1 for results). In the confusion matrix, each row is the true label and each column the predicted label. This makes it simple to see if the classifier confuses two labels or classes. An obvious example of this was the classification of Invoice Date and Due Date, both a source for confusion apparent in the results.

Data of training and testing time together with accuracy score was saved in a text file. To preserve all data, separately and in its original form, all the data was saved in text files as well. This raw data is presented in the confusion matrix but for full transparency, it is also saved in this form.

To give the algorithms different datasets for training and testing, each algorithm was run using different random seeds. In total ten seeds were used per algorithm, where the data were split with the same ten different seeds. The implementation itself was rather candid after preparations was done. Each run started with the algorithm currently testing and training getting the datasets, then implementing n-gram as a feature before using it specific classifier to train and then test on the different datasets. After training and testing, all results were saved, as described above.

4.4 Combined Tests (Objective 4)
When the separated tests from Objective 3 were finished results from the different algorithms were collected. From these results it was possible to see which ones got the highest results, both total and in specific categories from the created confusion matrices (5.1). From the decided method for the case study three of the algorithms were to be selected for two separated tests using different ensemble learning techniques. Also, all used algorithms were to be combined using the same two methods.

From the tests done in Objective 3 the results showed that SVM, Neural Network and Decision Tree yielded the best total accuracy (5.1.1, Figure 6). Specific results for different classifications varied between the algorithms but did not show one method being vastly superior over the three achieving the highest total result. Initially there was a thought to pick the algorithms that gained the best results for specific classifications to “help each other” in areas where there was a lack of accuracy. Because of the minor differences and of the fact that the two worst
performing algorithms got barely acceptable total results, (k-NN not even reaching the acceptable score of 70%) these could not be seen as candidates for the three algorithms to be used when combining.

Two functions implemented in Python were created for each ensemble learning technique (stacking and voting): one for three (the selected ones) algorithms and one for all five. The different algorithms were put together using EnsembleVoteClassifier for voting and StackingClassifier for stacking, both from the python library mlxtend.classifier. The same parameter settings as for the separated tests were used for the different algorithms. LogisticRegression from sklearn.linear_model was implemented (with default parameters) as the meta-classifier for the stacking method. As for the separated tests, ten different seeds were used for data-splits for each method, resulting in a total of 40 tests for the different combinations. The exact same data sets and splits as in Objective 3 were used to conduct the combined tests.

The combination of all five algorithms for the stacking technique had to be tested on a different machine than all other tests. This was due to the memory usage of k-nearest neighbors in combinations with all other algorithms which exceeded the memory of 16GB on the machine used for all other tests. A virtual machine, allocated 64GB of memory and with a different processor, was used instead to perform this specific training and testing. This could result in better performance in form of faster training- and testing times but will not affect the results in any way considering accuracy of classifications due to the same splits of data (same seeds). The tests conducted with this algorithm still used up to 90% of the 64GB memory.

When collecting the results from the combined tests the same methods as for Objective 3 was used. All data about predictions and actual classifications together with training- and testing time, and total accuracy were saved. To complement this, confusion matrices for the tests were saved to show accuracy for specific classifications.
5 Results

This chapter presents the results acquired from the different used algorithms in the case study. The chapter is divided into parts referring to the different objectives conducted by the two different researchers. Firstly, the data will be presented and secondly, these results will be analyzed.

5.1 Presentation

5.1.1 Objective 3

Objective 3 was aimed at running the different algorithms; train and test them on a dataset containing data presented on invoices. The results from the implementation of the five algorithms will be presented in this section, using a bar-chart to display their respective total mean accuracy, from ten different seeds, in percent (Figure 6) and bar-charts showing category mean accuracy, from ten different seeds, per algorithm for each text category. The bar-charts is produced using data from confusion matrices (see Appendix B - for details). This will demonstrate the accuracy performance between the different algorithms, as well as show what categories each algorithm performed well, or lacked, in. The acceptable result regarding accuracy set at 70% was, as Figure 6 shows, reached by four of the five algorithms.

![Figure 6 Bar-chart showing the mean of the total accuracy for the separate algorithms](image)

As shown in Figure 6, the SVM algorithm achieved the highest accuracy, with a total mean accuracy of 86.86%. The Neural Network algorithm had the second highest mean accuracy with 84.39%. Ranked third, with a mean accuracy of 79.88%, is the Decision Tree algorithm. The Naïve Bayes’ algorithm achieved a mean accuracy of 71.3%, barely climbing over the acceptable accuracy level of 70%. The k-NN algorithm finished with the lowest mean accuracy, achieving 47.63%.
As shown in Figure 7, SVM performed well in many categories, like Name, SSN-/ORG.nr, Address, Post Number, Phone, Email and Agreement, whilst struggling in a few other, such as Country, Invoice and Due Date and Other. As seen in the confusion matrix in Appendix B - Figure 4, the SVM classified half of the dates as Invoice dates and half of the dates as Due date, which support the hypothesis that the models would struggle with dates. It is also apparent that the SVM had some difficulties when classifying labels with numbers, such as Invoice Nr, Customer Nr and the amount categories, like Total and VAT, classifying these as other types of labels, similar to the correct one. 1346 data rows were correctly classified as Customer Nr but the model also wrongly classified 410 data rows as Invoice Nr.

As Figure 8 shows, Naïve Bayes performed well in some categories and poorly in others. It did well in Name, Address, Post Number, Email, Agreement and Reference but failed to accurately
classify Country, Invoice Nr and Due Date. The accuracy for classifying Other were almost none and most samples were labeled as Name. As seen in Appendix B - Figure 3, Naïve Bayes struggled classifying City, labeling many as Address or Name. Phone was another category of struggle, where the model wrongly categorized Phone as Reference or SSN-/ORG.nr. In the same figure, it is discernable that the model wrongly predicted most Invoice Nr as Reference. Overall, the categories containing numbers were problematic for the model.

![k-NN Category Accuracy](image)

**Figure 9** Bar-chart showing the mean category accuracy for k-NN

As shown in Figure 9, k-NN had troubles correctly classifying several labels. Whilst performing well for Name, Address and Post Number and decently for Invoice Nr and Agreement, it had problems with most labels, in many cases not even reaching 50% accuracy. When looking at the confusion matrix in Appendix B - Figure 2, it is important to note how many miss-classifications that has been made for Invoice Nr. Even though the accuracy for said category in Seed 0 is 85.13%, many other categories wrongly has been classified as Invoice Nr as well. Only about 21% of all predictions labelling the category as Invoice Nr, was actually Invoice Nr. One thing to notice in the confusion matrix is the models predictions for Email. Most of the predictions for Email are wrongly classified as Name. The reasons for this is further analyzed in 5.1.3.
As Figure 10 shows, Decision Tree performed well for many categories, for instance Name, Address, Post Number, Email and Agreement, whilst struggling with others, such as Country, Total, VAT and Other and failing to classify Due Date completely. In the confusion matrix in Appendix B - Figure 1 these difficulties correctly classifying number-based categories are apparent. The categories where Decision Tree has problem, in many cases has been wrongly classified as other labels, similar to the label it failed to classify, such as wrongly classifying Customer Nr as Invoice Nr or wrongly classify Reference as Invoice Nr, Phone or SSN/ORG.nr.

As shown in Figure 11, the Neural Network algorithm achieved high accuracy for roughly half of the categories. It struggled more with most other categories, reaching around, or little higher than, 50%. The algorithms biggest struggle was classifying the label Country. When looking at the confusion matrix for this algorithm (seed 0) in Appendix B - Figure 5, all dates
was classified as due date. Because the mean of all results is used in the chart, the results look different, showing 50% for Invoice Date and 50% for Due Date. It is apparent that the models classification for dates are random. The same goes for Total and VAT, as well as for the other labels barely reaching over 50 percent. The model can predict the general label, such as dates or amounts, but it cannot distinguish what “sub-label” it is, i.e., if it is a Due Date or an Invoice Date, Total or VAT.

5.1.2 Objective 4

Objective 4 aimed to combine the different algorithms from Objective 3 and thus make four different algorithms. Two ensemble models were built from the three best performing algorithms tested in Objective 3. These three algorithms were chosen from the highest total accuracy shown in Figure 6. The remaining two ensemble models were built from all of the separated algorithms used in Objective 3. The names used for the ensemble algorithms will from now on be referred to as Stack Three, Vote Three, Stack All and Vote All. The first part of the name describing which ensemble learning method is used, the second the number of algorithms used for combination.

![Total Accuracy in percent - Ensemble Algorithms](image)

**Figure 12**  Bar-chart showing the mean of the total accuracy in percent from ten different seeds achieved by the different ensemble algorithms.

As Figure 12 shows, only two of the ensemble algorithms achieved a higher result in percentage than the acceptable accuracy of 70% and both used the voting-technique. The algorithm using the three highest performing algorithms from Objective 3 (Vote Three) got the highest total mean result of 86.55% with the other voting algorithm (Vote All) not far behind with a result of 85.26%.
As shown in Figure 13 the accuracy of the Stack Three algorithms lacks in a majority of the categories. Only four (Name, SSN-/ORG.Nr, Phone and Customer Nr.) of the 17 classifications can be seen as acceptable (if the total acceptable limit of 70% is used) which shows why the total accuracy of the algorithm in Figure 12 is gaining such low total performance. As seen in the confusion matrix in Appendix B - Figure 6 this algorithm predicts almost all addresses and post numbers as SSN-/ORG.nr. Many types of data are classified as phone numbers and customer numbers. The percentage of accuracy shown in Figure 13 is high in these categories but as seen in the confusion matrix, many types of data are also classified to these incorrectly. With SSN as an example; approximately 75% of social security numbers were classified correctly, but only approximately 64% of all predictions for this category actually were SSN-/organization numbers.
The best performing ensemble learning algorithm; Vote Three, gains high results in the majority of the categories (Figure 14); the classifications lacking accuracy being Country, Dates (Invoice-Due dates), Amounts (Total, VAT) and Other. The algorithm gained an accuracy of 100% in three of the categories: Post number, Email and Agreement. This was the only ensemble method to predict more than zero countries accurately. The chart in Figure 14 shows a higher accuracy for invoice date than for due date, though, this is completely random. If the confusion matrix for this algorithm (for seed 0) is studied (Appendix B - Figure 8), all dates was classified as due date. Because the mean of all results is used in the chart the results look different and thus proves the random behavior of these fields.
The second stacking algorithm using all separated algorithms combined shows (Figure 15) a small improvement over the stacking model using only three (Figure 13). This version of the algorithm succeeded in getting acceptable results for the Email category but fails, on the other hand, to classify phone numbers acceptably. There are overall better results than for the Stack Three algorithms which also was shown in the total accuracy (Figure 12). Still, only four categories gain acceptable results if the 70% limit is used. The confusion matrix for this algorithm (Appendix B - Figure 7) shows similar results as for the Stack Three method. The spread of predictions is big and the categories gaining the highest accuracy are actually the once containing predictions from most of the text fields. One interesting result is that email addresses gained an increase of approximately 76% in accuracy when algorithms used for stacking was increased from three to five. Especially when k-NN was added as one of the algorithms which by itself had the worst performing accuracy for email-addresses of all separated algorithms (Figure 9).

![Vote All Category Accuracy](image)

**Figure 16**  
Bar-chart showing accuracy for each category achieved by the Vote All algorithm. Results are shown in percentage of accuracy (mean of ten seeds).

The second voting method, using all algorithms shows similar results (Figure 16) as for the Vote Three algorithm (Figure 14). The classification for Country still lacks in performance, the same for dates, amounts (Total and VAT) and Other. The algorithm achieves an accuracy of 100% on the same categories as Vote Three; Post Number, Email and Agreement. The results from the confusion matrices (Appendix B - Figure 9) gained from the tests done with this algorithm shows similar results as for Vote Three. There is a completely random behavior of dates and amounts.

**Analysis**

**5.1.3 Objective 3 - Andreas**

In general, the majority of the five trained and tested algorithms achieved good or very good accuracy results in the implementation. Since the accepted accuracy level of 70% is considered the measurement for when automatic classification yields more effectivity, the results
presented in 5.1.1, shows that four of the five algorithms are able to correctly classify the different labels with an accuracy that exceeds the accepted accuracy level. The only algorithm that cannot be considered as acceptable is the k-NN, with the low accuracy of 47.63%, which, by the definition of the aim, fails to meet the given standard. One probable reason for this is likely due to the internal structure of k-NN. Since it looks at all parts of the string that composes an email, and many emails contains names, i.e. name@email.com, the model most likely sees more similarities with names and puts more weight to that fact than the fact that the string contains a “@”, a strong indicator that it in fact is an email. This factor, together with its general difficulties classifying labels containing numbers, is likely the reason for its poor accuracy.

The algorithm with the highest accuracy was the SVM, supporting previous work (Lai, et al., 2015) showing SVM’s performance. Its total mean accuracy of 86.86% was very high and outperformed all the other algorithms. Although Neural Network came in as a close second, with its total mean accuracy of 84.39%, the average training and testing time of the Neural Network algorithm falls short of that of the SVM algorithm, as shown in Table 4.

The training time for Neural Network greatly exceeds that of SVM. The mean training time for the SVM is 50.99 seconds and for Neural Network it is 3782.63 seconds, or roughly 63 minutes on the selected machine. However, training time is not that important, when evaluating the algorithms for potential use in a real-world application. The general training of the model is something that would be done one time, so the time it takes to train the model plays a small part in the overall evaluation of suitability. Looking at the training times for the other algorithms (Table 4), the training time is very low for both Naïve Bayes and k-NN, where k-NN has the shortest training time of all five algorithms. The difference between Naïve Bayes and k-NN is and the other three algorithms are noticeable.

Since the underlying purpose of the acceptable 70% accuracy, is to see if any of the algorithms is suitable to be implemented in a real-world application, it is important to consider the test time for the different algorithms, considering this is what the algorithm would do in an application – receive data from an invoice and then classify it.

Looking at testing times, there appear to be no noticeable difference between any of the algorithms, besides k-NN, who’s testing time greatly exceeds the others. However, the difference in testing time between SVM and Neural Network, 0.01 seconds and 0.05 seconds respectively, are five times greater. In a system with a lot of users and traffic, this difference is huge. So, albeit the accuracy score for SVM and Neural Network doesn’t differ much, the difference in testing time gives SVM an even larger advantage. Even with similar accuracy, choosing an algorithm to be implemented in a real-world application, when the testing time differs that much, would not be so difficult.

The fastness of training and testing and the high accuracy displayed by SVM, makes the algorithm the overall best choice when comparing the separated algorithms. The long testing time for k-NN makes the algorithm unfeasible for use in an application. Even if k-NN would yield an accuracy of 100%, the testing time would make it impossible to implement.
Table 4 Training and testing times for the separated algorithms, shown in seconds taken to complete. The values are the mean of all tests for each algorithm on the ten different seeds.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Training Time (sec)</th>
<th>Avg. Test Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>50.99</td>
<td>0.01</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.09</td>
<td>0.02</td>
</tr>
<tr>
<td>k-NN</td>
<td>0.01</td>
<td>37.26</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>19.1</td>
<td>0.01</td>
</tr>
<tr>
<td>Neural Network</td>
<td>3782.63</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Neither of the separate algorithms showed acceptable results on dates (Invoice Date and Due Date), amounts (Total and VAT) or Countries. This follows what was predicted in the hypothesis (3.4). The algorithms difficulties to accurately classify dates and amounts is given. There is no discernable difference between Invoice Date and Due Date, nor between Total amount and VAT, so for the algorithms to make a distinction between these is nearly impossible without some sort of rule-based approach. Their difficulties to classify countries is probably caused by the small amount of countries in the data.

The hypothesis (3.4) was that SVM or Neural Network would yield the highest accuracy of the five separate algorithms. This hypothesis was based on earlier work by other researchers (Lai, et al., 2015) and proved to be correct.

5.1.4 Objective 4 - Linus

The results presented in 5.1.2 shows the voting methods being advantageous over the stacking algorithms with none of the latter performing over the accepted accuracy level of 70%. By the definition of the aim in this thesis, this means that the algorithms using stacking for classification not can be seen as acceptable to automatically classify invoice data.

An interesting result when analyzing the performance of the two stacking algorithms is the fact that the method using all separated algorithms outperforms the method using the three best (seen to total accuracy) ones from Objective 3. One could think that the result from all five algorithms combined would perform worse than the algorithms using only the three best ones, as stated in 3.4. The results from the two stacking algorithms tested offers evidence in disproof to this hypothesis. This could potentially be a result of the decisions made by the meta classifier used for this research. The use of more predictions for the stacking may improve the accuracy of the results, as a bigger “audience” is used for opinions about classifications.

Both voting algorithms performs with similar, high performing results. This could be the effect of the soft voting used, where probabilities from each used algorithm is used. The highest performing algorithms, shown in Figure 6, may have a higher percentage of predictions for the correct classifications. This would lead to their “vote” having more weight in predicting the final outcome. The possible uncertainty of the worst performing algorithms will thus have a minor impact on the prediction made by the ensemble model. Seen to the figures showing the results from Vote Three (Figure 14) and Vote All (Figure 16), there is a better distribution of correct prediction for dates by the latter one, making this algorithm look better for these types
of categories. Though, this could be completely random due to the fact of lack in knowledge for different dates in both algorithms.

An interesting result is the big difference between the voting and the stacking methods. With the use of logistical regression as meta classifier for stacking the outcome of this method should not differ much from the voting technique. The result should be similar to voting because the meta classifier should predict similar classifications. Due to the use of soft voting where voting was used outcomes could differ. The stacking method should also, theoretically, learn which internal classifier that most of the time gets the right predictions and therefore “mimic” the best performing method, which in this case is the SVM. The use of n-grams which results in a big amount of features might affect the meta classifier where a clear result or correlation never is found between the different classifiers and therefore affect the final predictions.

None of the ensemble algorithms performed with acceptable results on dates (Invoice- and Due date), amounts (Total and VAT) or Countries. This falls in line with the predicted outcome in the given hypothesis and follows the same pattern as for the separated tests. It implies that the ensemble algorithms fail on these categories due to most of the separated methods used not being able to classify them from the start.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Training Time (sec)</th>
<th>Avg. Test Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack Three</td>
<td>4608.24</td>
<td>54.98</td>
</tr>
<tr>
<td>Vote Three</td>
<td>4446.19</td>
<td>55.36</td>
</tr>
<tr>
<td>Stack All</td>
<td>6922.22</td>
<td>114.65</td>
</tr>
<tr>
<td>Vote All</td>
<td>4484.32</td>
<td>94.24</td>
</tr>
</tbody>
</table>

Table 5 Training- and testing times for the ensemble algorithms shown in seconds taken to complete. The values are the mean of all tests for each algorithm on the ten different seeds. Green coloring shows the best time and red shows the worst.

When looking at feasibility of using the different ensemble methods all of them take a significantly long time to train, and also to test compared to the separated algorithms (Table 5). The training time in this case depends a lot on the use of neural networks in every case. As Table 4 shows, neural networks take a significantly longer time to train compared to the other algorithms. Because NN performed as one of the better methods it was included in all of the ensemble algorithms. The training time of Stack All, shown in Table 5 sticks out as being larger than the other three although this specific algorithm was tested on a more powerful computer to be able to handle the memory usage.

This is a notable result due to the time being significantly longer for the ensemble methods than for the separated test. Theoretically the test time for Vote Three should be \((\text{SVM test time}) + (\text{DT test time}) + (\text{NN test time})\) which equals 0.07, according to test times shown in Table 4. There might take some extra time to decide the outcome of the vote but not as much as times in Table 5 are showing. This might be a result of hardware used to run these algorithms or the implementation of the models in the mlxtend library used.

Considering the results shown from the four ensemble methods it is possible to say, both from an accuracy as a time- and memory usage point of view, that the stacking methods can be seen as unfeasible and unable to perform over the set limit. The Stack Three did the testing with
the fastest average time, but it is not a significant decrease from Vote Three which got 27.3% higher total average accuracy in classification in almost the same time.

5.1.5 **Objective 3 & 4 – Final comparison**

From the results acquired and analyzed from Objective 3 and 4 it is possible to see which ones that possibly could be used to classify data effectively on invoices in general and especially by the Company in their work. There are six different algorithms that can be seen as acceptable and help the handling. These are SVM, Naïve Bayes (NB), Decision Tree (DT), Neural Network (NN), voting ensemble consisting of SVM, NN and DT, and voting ensemble consisting of all the five separated algorithms. This can be seen in the chart illustrated in Figure 17.

![Image](Image)

**Figure 17** Complete comparison of all algorithms used in the case study (Both from Objective 3 and 4). Shown in total average percent of accuracy over ten different seeds.

It is possible to conclude from the results that SVM outperforms the other methods. The voting methods are not far behind seen to total accuracy. Time taken to train and especially test the ensemble algorithms (Table 5) compared to the SVM (Table 4) makes them inferior and not as feasible in a real setting outside this research.

When analyzing the results for each category achieved by the best algorithms (SVM and the voting methods), it is possible to conclude that the results could be even higher if fields like date, countries and social security numbers were left out. This would be a simple task because of the fact that these field easily can be classified by rule based methods. It is the more complex fields such as names, cities and addresses that would be hard for rule based method to decide. The best performing algorithms in this study achieves between 75-100% accuracy for these categories which makes the results even better. The reason why number-based fields (like SSN and invoice numbers) got worse accuracy compared to other fields may depend on the fact that it is hard to find definitive patterns where numbers in practice can be formed in any way.
possible. Social security numbers has somewhat more consistency than invoice- and customer numbers because of the decided format (with e.g. a date format in Sweden with an added dash in most cases). There is although hard for the classifiers to see clear patterns in the n-gram features of random sequences of numbers, whereas there might be a larger amount of reoccurring sequences of letters in, for example, names. Most of the classifiers can classify dates correctly, although, this is not shown in the results because of the two different date categories. The classifiers has no way of deciding which one out of two dates is invoice- and due date. A rule based method can easily use standard date patterns and do a comparison between the dates to decide the order of dates. To achieve this with the ML algorithms they need to do the same internal comparison between found dates to make a decision.

A problem that occurred during early testing was that the models always had a 100 percent accuracy for addresses which seemed abnormal. This happened due to a skew in the data where more than half of the total data contained rows about addresses, including duplicates of street names with different numbers. This made the models overfitted to these exact addresses. Because of this, all street name duplicates were removed from the data and some addresses were removed, to become a smaller part of the total data set. The final results are therefore not as bias against any category and overfitting will not occur.
6 Discussion

6.1 Summary
This thesis aimed to research whether machine learning was suited to classify text fields on invoices with an acceptable accuracy or not. The motivation for this was to help companies with large amounts of digital invoices in general and Asitis AB (The Company) in particular to automate processes where classification of data was being made. The acceptance level of 70% was decided together with the Company as it would speed up processes with great significance.

Five different machine learning algorithms were first trained and tested on data containing text found on invoices (see appendix A): Decision Tree, K-Nearest Neighbor, Naïve Bayes, Support Vector Machine and Neural Network. Models were built in python in a library used for machine learning called scikit-learn. Every algorithm was trained and tested on the same ten different randomized splits from the dataset.

When the five different algorithms had been trained and tested, and results had been acquired, combinations of these models were implemented. Ensemble learning was used in the style of both voting and stacking to combine, first the three highest performing (with the highest total accuracy) and lastly all five algorithms. These newly created models were trained and tested on the same exact splits of the dataset and comparable results were collected.

The results acquired from all the different algorithms showed that a majority of the models made it above the acceptable level of 70%. Support Vector Machine proved to be the most accurate algorithm with 86% which also was predicted in the hypothesis of the study. The ensemble learning models using voting gave promising results, not far behind the SVM. This shows that machine learning, with a 16% margin over the acceptance level, can provide a decrease in effort needed to classify information on invoices.

6.2 Conclusion

6.2.1 Comparison to Previous Work
Results found in this thesis follows the same patterns as earlier work done by other researchers such as Raju, et al. (2017) and Khan, et al. (2010). According to these authors SVM is one of the best performing algorithms for text classifications. This was clearly shown in this study as it performed with the highest accuracy even on invoice data where only n-grams were used. Raju, et al. (2017) states that SVM primarily have been used for e-mail classification and spam filtering where good results have been acquired. In these cases, n-grams have been used on word-level as in text being classified based on words present in the text. This thesis proves SVM works just as well using n-grams on character-level where single words, in most cases, are being classified. The table (Table 2) provided in Raju, et al. (2017) shows some inconsistencies with the results from this thesis. In the table, Raju, et al. (2017) describes scalability as a pro for K-nearest neighbor (k-NN), which is said to scale up well with the number of documents. However, it is clear from the tests performed during implementation in this thesis that k-NN does not. The mean accuracy is lower than 50 percent and its test time is a lot higher than the other algorithms. This is no surprise, since each K have to make a comparison with its neighbors, which in turn has to do the same with its neighbors. This means that the more documents, the longer k-NN takes to complete its classification.
When looking at the worst performing algorithms the found results differs from earlier work. Bijalwan, et al. (2014) concludes that K-nearest neighbor performs well for text classification, outperforming Naïve Bayes (NB). This is not shown in the results of this thesis. While both k-NN and NB are the worst performing methods k-NN is still far behind; not even close to the acceptance level of 70%. This might be an effect of the different methods of feature selection where Bijalwan, et al. (2017) are using n-grams in form of words as well as Raju, et al (2017) has explained for SVM.

The results seen from ensemble learning techniques show promising results (at least for the voting method) but does not outperform the standalone version of SVM. These findings differ from the work done by Polkar, et al. (2006) where it is claimed that ensemble techniques often outperform single algorithms. The authors although conclude that there is no single algorithm with universally higher accuracy among the ensemble algorithms. Both the voting and stacking methods can be effective in specific areas. This thesis showed that the stacking method was not suited to this particular area, with the parameters and settings used. The voting algorithm was found to be much simpler because of the use of a simple vote among the different algorithms in use. Polkar, et al. (2017) claims that this often is the case, where the simplest algorithms tend to be the best ones.

6.2.2 Validity

The validity of the thesis is dependent on the handling of the validity threats identified in section 3.6.8. One threat identified is related to the Reliability of treatment implementation, where there is a risk of differing implementation between different researchers in their applying the treatment or between different times. The threat is handled by using the same implementation for both Objective 3 and Objective 4 (see 3.5), with the same seeds and the same split of data. Another threat, linked to the same validity regards the risk of differing of time. As described in Implementation, section Combined Tests (see 4.4) the implementation using all five algorithms using stacking needed more memory to run, which means that the training and testing times from this test have been affected in a different way than all other test runs. Even though this change of hardware has no effect on the accuracy score, it could potentially affect the algorithms running time and this needs to be taken into consideration when evaluating this part of the result.

When performing a case study, one must be aware of confounding factors, such as the difficulty to determine effects between different factors and their potential effects on the results. To counteract the risks of the confounding factors, an average accuracy score for each category was calculated and visualized using a confusion matrix. That allowed for individual examination and evaluation, it gave the means to find potential confounding factors and attribute their proper impact to the total average accuracy.

In order to avoid a skew between classes or labels, when splitting the data, efforts were made to balance the amount of data per label, to not get an overload of data to certain labels. Even though this was something considered, there still is differences in the amount of data that was used. Name has much more data fields than most of the other labels. However, the greatest impact on how well an algorithm performs seems to depend more on the format of the data, than the number of fields. All algorithms performed very well on Payment Agreement, despite having no more than 35 data fields, but the format of this data was more distinct able, than for example dates or amounts. Yet, this skew between labels, is something one needs to keep in mind when observing the results, as well as the selected N in N-grams.
The use of the *scikit-learn* library from Python and our lack of control over it is something one needs to accept, and the library’s implementation of the algorithms cannot be affected. It is important to note that the results presented in this thesis only are applicable to the use of said library and therefore might not be generally applicable but it also makes it possible for others to reproduce the tests.

Even though the level of accepted accuracy might not be completely generalizable to all companies handling invoices or other digital data, 70% could be seen as somewhat of a benchmark for when automatization could be an improvement to manual handling. Even if a different company sets a higher level of accepted accuracy, the results are still present and valid and the final results should be the area of interest, if one aims to implement automatization solutions.

6.2.3 Social Aspects

The social benefit of the results found in this thesis is that time and money can be saved in many areas where invoices are being handled in digital form. These areas are, by each day, increasing in numbers. Similar industries with similar documents as invoices will also draw benefit from this research as the data exists in many forms of documents. Names, addresses, social security numbers and other categories that was used as data in this study appears in a big part of the digital world and may impact industries of all sorts when handling has to be done manually. Future developments built on the findings in this thesis may fully automate categorization of fields on invoices which in turn will lead to a much more efficient process that may affect the society positively.

Even if fully automated categorization is a possibility in the future, there will most likely always be a need for some sort of supervised handling. If the classification fails or categorizes data improperly, there is a risk that personal or financial data gets stored incorrectly. The implications of faulty handling can potentially be vast and there is ethical drawback from the potential automatization of classification of data from invoices. There are a number of people working with manually classifying data right now and if their task where to be automated, there is an impending risk that their employment type will be considered superfluous. Even if a small number of supervisors where to be considered necessary, many might not and this could have a negative impact on said subordinates in specific, and society in general.

6.3 Future Work

To take the research done in this thesis further and make it work in a setting outside of this study, it has to be combined with other forms of learning and methods to decide outcomes. One big problem that the case study showed was that the models did not know about dates and amounts of money enough to separate them and classify them into the right category. Future work will include combining the SVM (having the highest accuracy) with some sort of rule-based method to decide the final outcome of uncertain elements. For example, could this rule-based algorithm include rules telling dates that fall behind another date to be the due date, because of the probability that this is the case. This could increase to accuracy even more and reach values close to 100% if done right.

A big step to get closer to perfect accuracy when classifying would be to include positions of text fields in the learning process. If the machine learning model would be combined with an OCR-engine (Optical Character Recognition) where positions of text were known this could
give variables to classify one more feature to look at by the ML algorithm. It would be interesting to see how this would affect the outcome and results of the highest performing algorithms. This can, however, be risky since it may be possible to trick the algorithm. It could lead to the model putting more trust in the position retrieved, rather than the actual data, when classifying an invoice and if the layout of a new invoice differs from the ones used for training, these invoices could potentially have their data classified wrong.

The ensemble learning technique using stacking did not perform as well as expected. This was a big surprise and the cause for this is still unknown. Future work could look at how to classify the same data using different meta-classifiers and parameters for the stacking classifier. Logistic regression was used as the meta classifier in this study but could be compared to many different classifiers to make the best algorithms found in this thesis perform more accurately when stacked.

Another area for future work could be looking at the level of acceptance, to try and find out where the actual level for when automatization improves the performance of handling digital data over manual handling lies. The level of accepted accuracy in this thesis is based on discussions with one company. To scientifically ground the level of accepted accuracy in research, this could be interesting for the future.

The algorithms left out by this thesis which other researchers have used in earlier work could be tested in this problem area (on invoice data). There is a possibility that this study have missed one algorithm that potentially could out-perform the SVM. There is also a possibility to research other parameter settings of the used algorithm and for example investigate further the effects of changing the $N$ in $N$-grams which in this thesis was set to a range between one to four. To use a larger value of $N$ or even use $N$-grams on word-level could also potentially make the results even better, with regards to accuracy.
7 Bibliography


# Appendix A - Classifications

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>CLASSIFICATION</th>
<th>DATA SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Name (person or company)</td>
<td>Data collected from SCB</td>
</tr>
<tr>
<td>2</td>
<td>Social security number (SSN) / Organization number</td>
<td>Data generated to match Swedish SSNs and organization numbers</td>
</tr>
<tr>
<td>3</td>
<td>Address</td>
<td>Data collected from openaddresses.io</td>
</tr>
<tr>
<td>4</td>
<td>Post number</td>
<td>Data collected from openaddresses.io</td>
</tr>
<tr>
<td>5</td>
<td>City</td>
<td>Data collected from SCB</td>
</tr>
<tr>
<td>6</td>
<td>Country</td>
<td>Nordic countries</td>
</tr>
<tr>
<td>7</td>
<td>Phone</td>
<td>Generated data from common Swedish phone formats</td>
</tr>
<tr>
<td>8</td>
<td>Email</td>
<td>Data from Asitis AB</td>
</tr>
<tr>
<td>9</td>
<td>Invoice Number</td>
<td>Generated data matching invoice number formats</td>
</tr>
<tr>
<td>10</td>
<td>Invoice Date</td>
<td>Randomly generated dates (in different formats) between year 2000-2020</td>
</tr>
<tr>
<td>11</td>
<td>Due Date</td>
<td>Randomly generated dates (in different formats) between year 2000-2020</td>
</tr>
<tr>
<td>12</td>
<td>Customer Number</td>
<td>Generated data matching customer number formats</td>
</tr>
<tr>
<td>13</td>
<td>Payment Agreement</td>
<td>Data created containing common Swedish payment agreements</td>
</tr>
<tr>
<td>14</td>
<td>Reference</td>
<td>Data generated following OCR standard for reference numbers</td>
</tr>
<tr>
<td>15</td>
<td>Total Amount</td>
<td>Generated data containing decimal numbers between 1-50000</td>
</tr>
<tr>
<td>16</td>
<td>Value-added tax (VAT)</td>
<td>Generated data containing decimal numbers between 1-12000</td>
</tr>
<tr>
<td>17</td>
<td>Other</td>
<td>Created data containing common titles on Swedish invoices</td>
</tr>
</tbody>
</table>

Table 1 A table listing all classifications with a number and information describing where data for training and testing was acquired.
Appendix B - Confusion Matrices

Appendix showing one confusion matrix from each used algorithm. The results are from seed 0 for every algorithm.

Figure 1 Confusion matrix showing results from Decision Tree seed 0.
**Figure 2** Confusion matrix showing results from K-Nearest Neighbor seed 0.
Figure 3 Confusion matrix showing results from Naïve Bayes seed 0.
**Figure 4** Confusion matrix showing results from Support Vector Machine seed 0.
**Figure 5** Confusion matrix showing results from Neural Network seed 0.
Figure 6 Confusion matrix showing results from Stack Three seed 0.
**Figure 7** Confusion matrix showing results from Stack all seed 0.
Figure 8 Confusion matrix showing results from Vote Three seed 0.
Figure 9 Confusion matrix showing results from Vote All seed 0.