Product categorisation using machine learning

Produktkategorisering med hjälp av maskininlärning

STEFAN VASIC
NICKLAS LINDGREN
Product categorisation using machine learning
Produktkategorisering med hjälp av maskininlärning

STEFAN VASIC
NICKLAS LINDGREN

Degree project in
Computer Engineering
First cycle, 15 credits
Supervisor at KTH: Jonas Wåhslén
Examiner: Ibrahim Orhan
TRITA-STH 2017:39

KTH
School of Technology and health
141 52 Huddinge, Sweden
Abstract

Machine learning is a method in data science for analysing large data sets and extracting hidden patterns and common characteristics in the data. Corporations often have access to databases containing great amounts of data that could contain valuable information.

Navetti AB wants to investigate the possibility to automate their product categorisation by evaluating different types of machine learning algorithms. This could increase both time- and cost efficiency.

This work resulted in three different prototypes, each using different machine learning algorithms with the ability to categorise products automatically. The prototypes were tested and evaluated based on their ability to categorise products and their performance in terms of speed. Different techniques used for preprocessing data is also evaluated and tested.

An analysis of the tests shows that when providing a suitable algorithm with enough data it is possible to automate the manual categorisation.

Keywords
Machine learning, product categorisation, support vector machine, naive Bayes, DBSCAN
Sammanfattning

Maskininlärning är en metod inom datavetenskap vars uppgift är att analysera stora mängder data och hitta dolda mönster och gemensamma karaktärsdrag. Företag har idag ofta tillgång till stora mängder data som i sin tur kan innehålla värdefull information.

Navetti AB vill undersöka möjligheten att automatisera sin produktkategorisering genom att utvärdera olika typer av maskininlärnings-algoritmer. Detta skulle dramatiskt öka effektiviteten både tidsmässigt och ekonomiskt.

Resultatet blev tre prototyper som implementerar tre olika maskininlärnings-algoritmer som automatiserat kategoriserar produkter. Prototyperna testades och utvärderades utifrån dess förmåga att kategorisera och dess prestanda i form av hastighet. Olika tekniker som används för att förbereda data analyseras och utvärderas.

En analys av testerna visar att med tillräckligt mycket data och en passande algoritm så är det möjligt att automatiska den manuella kategoriseringen.

Nyckelord
Maskininlärning, produktkategorisering, support vector machine, naive Bayes, DBSCAN
# Table of contents

1. Introduction 1
   1.1 Background 1
   1.2 Problem definition 1
   1.3 Objective 1
      1.3.1 Research objectives 2
      1.3.2 Development objectives 2
      1.3.3 Analytic objectives 2
   1.4 Delimitations 2
   1.5 Authors contribution 3
2. Theory and Background 5
   2.1 Machine Learning 5
   2.2 SciKit 5
   2.3 Previous Work 6
   2.4 Social effects of Artificial Intelligence and Machine Learning. 6
   2.5 Supervised & Unsupervised Learning 6
   2.6 Machine Learning Algorithms 7
      2.6.1 Naive Bayes Algorithm 7
      2.6.2 Support Vector Machine 7
      2.6.3 DBSCAN 8
3. Methods of solution 9
   3.1 Initial literature studies 9
   3.2 Data preparation 9
   3.3 Prototype development 10
   3.4 Prototype testing and optimising 10
      3.4.1 Result gathering 12
4. Result 13
   4.1 Literary Study Result 13
      4.1.1 Social aspects 13
      4.1.2 Algorithms and libraries 13
   4.2 Prototypes 14
      4.2.1 Prototypes using supervised machine learning 15
      4.2.2 Prototype using unsupervised machine learning 17
4.3 Test results

5. Analysis and discussion
   5.1 Analysis of implemented prototypes
      5.1.1 Supervised machine learning
      5.1.2 Unsupervised machine learning
   5.2 Analysis of data preparation
   5.3 Analysis of automated categorisation
   5.4 Analysis of alternative frameworks
   5.5 Social and economic aspects

6. Conclusion and future recommendations
   6.1 Future recommendations

7. References
1. Introduction

This chapter describes the problem definition that set the foundation for this thesis. The objectives and delimitations set during the start of this thesis are also described.

1.1 Background

There can sometimes be a need to automate work and let a computer perform the task rather than a human. Setting up clear, pre-decided rules for how a computer should execute the task can be a simple solution to this problem. But if you want a machine that learns over time, exploring the possibilities within the field of machine learning could provide the solution.

Navetti AB is a company that works with pricing and it have expressed a need for exploring machine learning techniques that could be applied to their product categorisation. They wanted a comprehensive review of the possible solutions currently available, as well as a prototype application.

1.2 Problem definition

At the moment of writing, Navetti categorises products manually. The products can be anything from air filters to drill heads and there can be up to a million products per client. For every new client they have to categorise the products and apply pricing methods on the categories. Applying rules regarding how a computer could perform the categorisation is not an option because the products are different for every new client. The problem could be solved with the help of advanced machine learning that could learn from previous categorisation and then make decisions on new products and keep learning over time.

This work evaluates the available solutions and approaches that can provide a machine that learns and uses that knowledge to automate the categorisation of products. This work also examines how data can be prepared for the machine and what effect the preparation has.

1.3 Objective

The aim of this thesis is to produce a proposal for further development of automated categorisation using machine learning. The work is divided into three stages: research, prototype development and analysis. Each stage has individual objectives.
1.3.1 Research objectives
1. Obtain information on different techniques and methods in the field of artificial intelligence and machine learning that can be used for product categorisation and document their differences.
2. Document what previous work has been done in this field and what the results have showed.
3. Document how artificial intelligence and machine learning affects society with regard to economic, ecological and social aspects.

1.3.2 Development objectives
Based on the result from the research on suitable techniques, two or three prototypes will be developed and tested. The objectives with the development stage are:
1. To implement prototypes that use machine learning techniques for automating categorisation. The prototypes are testable and comparable.
2. To test the prototypes using different data sets and evaluate the accuracy. The tests will show if the prototypes can categorise a product and they will be compared based on speed and accuracy.

1.3.3 Analytic objectives
With the data collected from tests conducted during the development phase an analysis will be performed with these objectives:
1. Compare the prototypes based on speed and accuracy.
2. Analyse if manual categorisation could be replaced with one of the machine learning techniques.
3. Recommend a future approach on how to achieve automated categorisation.

1.4 Delimitations
This work has delimitations and those are:
1. Only prototypes were developed and nothing was implemented in the company’s system.
2. The algorithms used by the different techniques are already written. New algorithms will not be implemented during this work.
3. Tests are made under the limitation of the computers prestanda. Speed will not be optimised but a comparison of the speeds will be a valid result.
4. The focus is on Navetti’s need and not product categorisation in general.

1.5 Authors contribution

All the work presented in this paper is done by the authors. Our supervisors, one from Navetti and one from KTH is guiding us throughout this work. The research and analysis is done by the authors and the development is done by the authors with input and guidance from our supervisor at Navetti. Both authors worked together throughout the whole work but author number one focused more on supervised machine learning whilst author number two focused on unsupervised machine learning.
2. Theory and Background

The summary of the research that was done in order to understand the problem is treated in this chapter. Section 2.1 introduces the field machine learning. Section 2.2 describes the tool SciKit. Section 2.3 presents previous work that has been done in this field. Section 2.4 presents social effects of artificial intelligence and machine learning. Section 2.5 explains the difference between supervised and unsupervised machine learning. Section 2.6 goes into detail about three different machine learning algorithms and how they work.

2.1 Machine Learning

According to the paper [1] humans and machines differ in how they perform a task, whilst a machine only executes a given task, humans usually devote efforts to improve their skills simultaneously while performing a task. Today’s computers can be very efficient but they do not self-improve over time. Machine learning is a field within computer science that enables computers to learn and continuously improve from big data sets. This is possible by analysing data and detecting patterns in aforementioned data to later make predictions on new unknown data. By letting the system learn continuously and be corrected when failing a task it will improve and be more accurate over time.

How do we know that the machine has learnt? Avlan H. et al [2] says that the term machine learning is confusing because learning implies thinking and today’s machines cannot think and to learn you have to do so deliberately. In the field of machine learning we are interested in performance improvements because we can test if a machine has learnt something by testing if it performs a given task better, that is if it has self-improved.

Several different algorithms and methods have been developed to extract the patterns, with different advantages and objectives.

Machine learning can be used for a wide variety of applications, ranging from email-spam detection, visual object detection and customer purchase patterns [3] and there is great value for businesses to obtain this information.

2.2 SciKit

SciKit is an open source python-module containing multiple machine-learning algorithms. It includes a great variety of both supervised and unsupervised algorithms and focuses on giving non-specialists an easier start with machine learning. The libraries all come with extensive documentation
as well as comprehensive code examples that gives the users a quick introduction to the technology. [4]

2.3 Previous Work

Mikael et al. [5] used product classification for determining environmental and social impacts when consuming a product. With 800 categories and a training data set of 3.1 million products different algorithms were used to show that the best classifier algorithm gave a hierarchical precision of 0.81.

Sushant et al. [6] present a method for classifying products by using supervised machine learning. Products were classified into known categories based on information about the product, such as name and description. Categories were for example ‘Electronics’ or ‘Automotive’. One of the conclusions from this work was that using multiple classifications that is, one product - multiple categories, greatly improved the accuracy. Filtering items also helped the accuracy. Filtering was made by not including items that don't have enough information to be properly classified.

Yu et al. [7] developed a tool called “LibShortText” that is used for short-text classification and analysis. The default options for this tool were selected based on a study [8] of how the pre-processing and extracting of features from short texts could be optimised. The tool showed an accuracy of over 90% when classifying 10,000,000 short texts into 34 categories. The texts were classified with only their titles. Training the machine with 10,000,000 items took 37 minutes whilst predicting with new items took 28 minutes with the same number of items.

2.4 Social effects of Artificial Intelligence and Machine Learning.

Technologies such as artificial intelligence and machine learning could have a substantial effect on the current workforce-situation. For example some of the biggest and highest value companies today have a significantly smaller amount of employees compared to a few decades ago [9]. As the implementation of the new technologies will certainly replace humans in some sectors, it will also create new, safer and better jobs more fit for humans. Researchers are however unsure if the removal of some number of positions will result in an equal number of new positions [10].

2.5 Supervised & Unsupervised Learning

There are two dominating approaches in machine learning considering the initial training of the system [10], supervised and unsupervised learning. The first method, called supervised learning, consists of providing the system with a pre-defined data set, or items with a defined category. The system can
then build the classification model based upon these categories. When a new item is to be classified, the model will predict which category to place it in. In the second method, unsupervised learning, the data used is unlabeled. The system then has to find patterns within the data and do the categorising itself, without any learning between data sets.

2.6 Machine Learning Algorithms

2.6.1 Naive Bayes Algorithm

According to K. Ming [12] the naive Bayes is a collection of classification algorithms based on Bayes Theorem:

\[
P(b|a) = \frac{P(a|b)P(b)}{P(a)}\]  

(1)

Where \( P(a) \) and \( P(b) \) are events and \( P(a) \neq 0 \). Naive Bayes is often used in machine learning and data mining, and is the simplest of the “Bayesian” classifiers [2]. This family of algorithms all share the principle that each feature examined during the classification contribute independently to the outcome of the classification. Real world data often has some form of correlation that the algorithm cannot make use of, and is therefore called “naive” [12]. This does however give the algorithm some benefits. For example the classification process is often much more efficient, compared to other more complex algorithms, although the result may not be as precise. The classifier itself is also fast and relatively simple to train. [13]

Because of the simplified classification model used by naive Bayes it can easily handle a great amount of features, or words. Therefore the algorithms is widely used in text-classification applications, such as spam-email detection, sentiment analysis and other forms of text categorisation [14].

2.6.2 Support Vector Machine

Support Vector Machines (SVMs, or support vector networks) in machine learning are supervised learning models used to analyse and classify data [15]. According to Jason [16] SVM is a classification algorithm that learns the mapping \( x \rightarrow y \) where \( x \in X \) and \( y \in Y \) and \( x \) is an object and \( y \) is a classification label. During this training the machine builds a model using labeled data and assigns every entry to a point in a vector space. Considering a two-dimensional input space, where \( x \in R^2 \) and the training data are given as \( (x_1, y_1), \ldots, (x_n, y_n), \ x \in R^2, \ y \in \{1, -1\} \), separation can be done using different hyperplanes that linearly separates the data. As seen in figure 2.1 there can be different hyperplanes and the support vector machine chooses the hyperplane with the biggest margin between the classes. [17]
SVM finds parameters $w = [w_1, w_2, \ldots, w_n]^T, w \in R^2$ and then the machine is given an unseen pattern $x_p$ and the machine calculates if $x_p$ belongs to class 1 or class 2. [17]

2.6.3 DBSCAN

DBSCAN, or “Density-based-spatial clustering of applications with noise”, is a data clustering algorithm proposed in 1996. “Density-based” implies that the algorithm groups points close together, and considers points under a certain threshold as noise [18]. The algorithm works by taking two parameters, a positive integer “epsilon” and a natural number “minPoints”, for “minimum points”. It then starts by picking a random point in our data set. If there are more “minPoints” than points within a distance of epsilon from that point, we consider all of them to be part of a cluster. We then expand that cluster by checking all of the new points and seeing if they too have more points than “minPoints” within a distance of epsilon, growing the cluster recursively [19]. Eventually, we run out of points to add to the cluster. We then pick a new arbitrary point and repeat the process.

DBSCAN is one of the most widely used clustering algorithms today and has several advantages compared to other clustering algorithms such as “k-means”. For example DBSCAN does not need to know the number of expected clusters, compared to k-means [20].
3. Methods of solution

This chapter deals with the different methods used during the work in order to go from the problems in section 1.2 to achieving the goals presented in 1.3. Section 3.1 describes the initial research that was done in order to achieve the goals presented in 1.3.1. Section 3.2 describes the preparation of the data that was done in order for it to be compatible with the prototypes. The preparation was needed in order to be able to achieve the goals presented in 1.3.3. Section 3.3 describes the methods chosen for the prototype development stage that was done to achieve the goals presented in 1.3.2. Section 3.4 describes the methods used for testing and optimizing the prototypes in order to achieve the goals described in 1.3.3.

3.1 Initial literature studies

The initial part of the project consisted of a literature study. The study started on topics regarding machine learning in the current society, different types of applications and previous attempts at using machine learning in product categorisation. This was needed to get a broader knowledge of the topic. Since there are several different methods and techniques to use in machine learning, the study then continued to more specific implementations to evaluate the most fitting method for the project’s purpose. Several libraries and implementations for different coding-languages were available and a suitable language and library were selected based on features, instructions, guides and overall usage.

3.2 Data preparation

The data used for the categorisation were provided by the company, and simulates pre-categorised product data. Only simulated data was used during the work. The data sets used consisted of 567 columns and up to 200,000 rows per data set. The columns contained a mixture of text and numeric values. The information about the categories was in text, in form of a category name. Missing values were not rare and this was a problem that had to be dealt with. Instead of removing the whole row that consisted of one or more missing values, the method of dummy substitution was chosen. “Null” was chosen as a replacement for missing values in text columns and 0 in number columns. To decide how to use the provided data for machine-learning purposes, the group consulted employees [21] that were currently working with the manual categorisation, and got a better understanding on how to use the data. Because of the variety in the different data sets it was not possible to only use a few of the features and therefore all features were used.
To get the best possible outcome and the most accurate categorisation from machine learning the data has to be processed in a way to remove data not directly related to the item itself. This can for example be IDs and dates. Should these values be included they would add an extra inaccurate dimension in the machine-learning-process and could significantly reduce the accuracy of the finished product [22]. The process of cleaning the data this way was done manually by carefully analysing the columns in the data sets, evaluating the importance of each column and remove the selected columns.

3.3 Prototype development

The method chosen for this project was to develop own prototypes instead of using an already working product from another company, like Google’s “Cloud Machine Learning Engine” [23] or Microsoft’s “Azure Machine Learning Studio” [24]. Since the result of this project could be included in future developments and possibly deployed in production environments using real data, the applications and data had to run and be stored locally. Another reason for this method is to provide easier future integration in Navetti’s current systems.

Three of the most widely acknowledged algorithms were selected for prototype development and testing. A qualitative approach was used when selecting the algorithms based on their previous successful implementations and available documentations and instructions. Two supervised algorithms and one unsupervised algorithm were used for prototyping. Since these two types of algorithms work differently in their categorisation, both were used for comparison and evaluation.

3.4 Prototype testing and optimising

When the prototypes were fully developed the project could continue to the testing phase. A quantitative approach was used in testing, with multiple tests executed to thoroughly analyse the algorithms. All tests were repeated 30 times and using the result of the tests, an average value and a confidence interval were calculated. The number 30 was chosen because when conduction 30 or more tests one can assume that it is normally distributed.

The tests were conducted using Python 2.7.10 and a MacBook Pro running MacOS 10.12.4 with a 2.2 GHz Intel Core i7 with 4 physical cores and 16GB of RAM.
The tests consisted of:

1. **Validating that the prototypes works as expected by categorising a widely used data set.**
   The iris data set [25] is a data set containing different species of flowers, with multiple features like length and width of its pedals. These tests were conducted in order to insure that the prototypes correctly builds the matrix that was later passed to the algorithms.

2. **Categorising data sets of different sizes.**
   This test started with using smaller data sets with only a few hundred items and only a few categories. For each following test the amount of items and categories were increased to be able to compare the results from a variety of data set-sizes.

3. **Categorising raw and clean data.**
   Testing with raw data and comparing the result with results from the preprocessed data evaluated the possibility of incorrect categorisation when including unimportant values in the data sets.

4. **Categorising raw and normalised data.**
   Raw data often contains different ranges of values in its numerical features. Since the majority of machine learning classifiers calculate the distance between data point in a graph by the Euclidean distance [26], features containing values in a wider range will affect the classification to a greater extent compared to features containing values in a smaller range. Therefore the data had to be normalised to give its resulting data point’s proportional distances, and its features comparable weights [26].

   The simplest method of normalisation is called “Rescaling”, and works by rescaling the features to values between 0 and 1, according to the formula:

   \[
   x' = \frac{x - \text{min}(x)}{\text{max}(x) - \text{min}(x)}
   \]  

   Where \(x\) is the raw value, \(\text{min}(x)\) is the lowest feature value and \(\text{max}(x)\) is the highest. The result \(x'\) is the normalised value.

5. **Categorising using different input-variables for the algorithms.**
   In order to fine-tune the results of the categorisation each algorithm has a set of input-variables to specify different aspects of the algorithms itself. An example of an input-variable is “min_points” and “epsilon” in one of the algorithms. By adjusting these values the
thresholds on what items will end up in which category is changed, and will generate different results.

3.4.1 Result gathering

Each algorithm was first validated by comparing the results from the iris data set to the correct categorisation. When the functionality of the prototypes were proven the analysis continued to the data provided by Navetti. The performance factor of the prototypes were measured in execution-time, and the accuracy factor were measured in amount of items placed in the correct category.
4. Result

This chapter presents the result. Section 4.1 describes the result of the literary study that was conducted according to 3.2.1. It answers the question from 1.3.1 about how artificial intelligence and machine learning are affecting the society. Section 4.2 describes the prototypes that were developed according to the methods described in 3.2.3. Section 4.3 presents the test results that were performed according to the methods described in 3.2.4 and a comparison of the different prototypes are presented.

4.1 Literary Study Result

4.1.1 Social aspects
As proven by previous studies all types of computerisation in society poses a serious threat to workers in certain sectors. The first jobs to be replaced by some form of computer are the simpler jobs like conveyor-belt inspection and assembly, where human specific qualities are less of a key factor. As mentioned in the literary study, this can bring dramatic consequences for middle class jobs and incomes.

4.1.2 Algorithms and libraries
Based on the studies, the decision to use Python for the prototype development was made. Python is a well-known programming language and widely used amongst data scientists and often used in machine learning applications. The language has a big community and an arsenal of tutorials and guides, which also contributed to the decision.

One of the main reasons for choosing Python for this project was the library “scikit-learn”. This is a library containing a wide variety of machine-learning and data analysis tools. It has support for several of the algorithms used in machine learning, as well as tools for visualising data and plotting graphs.

The choice for algorithms were based on three main criteria: previous implementations, ease of use and variety. These criteria were set to make sure well-tested and well-performing algorithms were used, as well as making sure different types of algorithms were tested to evaluate which type would perform best in this case. Since there are two dominating types of algorithms used for this type of applications, unsupervised and supervised, both were included in this work.

Representing the unsupervised side of the spectra, DBSCAN were used. DBSCAN works by clustering unlabeled data. DBSCAN were chosen due to its ability to find clusters of more complex sizes and shapes.

“Naive Bayes” and “Support Vector Machine” were chosen as the supervised algorithms. The first is known for its performance in terms of speed, due to
its naive approach in finding the right categories, although lowering its accuracy. The second is often used when the lower accuracy of the first mentioned algorithm is too low. These two algorithms provided a good variety of performance and accuracy.

4.2 Prototypes

The three different prototypes were all developed in Python and their main activities were:
1. Read product data from a CSV file
2. Separating text columns and number columns
3. Perform vectorising on the text columns
4. Normalising the data
5. Join together the vectorised text columns and the number columns
6. Feed the data to the selected algorithm

The prototypes made a matrix of the data read from the CSV file. This matrix consisted of \( x \) number of rows and \( y \) number of columns where the \( x \) stands for products and the \( y \) stands for all the features of a product. Then they iterated through the columns, separating the columns where the values were in text format from the ones in number format and splitting the matrix into two matrices.

Because the algorithms are not directly applicable on text columns, the format of the text columns had to be changed. The prototypes transforms the collection of text into a matrix of token counts, meaning that they iterate over a column and create a vocabulary of words assigning an integer to each word in the vocabulary. This is known as vectorising. Then the occurrences of each word is counted for every row in the column where a row is a product and a column is a feature of that product.

Normalising was done according to the method described in section 3.2.4 and the prototypes were constructed so that the user could choose if the data would be normalised or not in order for a comparison to be made.

The final data was treated differently based on what algorithm was chosen, if it was a supervised or unsupervised machine learning algorithm.

![Flow diagram of the functionality of the prototypes](image-url)
4.2.1 Prototypes using supervised machine learning

The two finished prototypes that used supervised machine learning algorithms, naive Bayes classifier and support vector machine were very similar in structure. The only difference between them was which algorithm they used. The prototypes were constructed so that they could be fed with the final data explained in 4.3. The main features of these prototypes were:

1. To shuffle the data and then split it into two matrices, one for training and one for testing.
2. To apply the algorithm on the train data set consisting products with a known category.
3. To predict categories for the test data set also consisting products with known category.
4. To compare the predicted categories to the actual categories.

These prototypes were constructed so that one part of the data set was used for training the machine and the other part for testing the machine. The data set was therefore shuffled before being divided because the data set was ordered by category and in order for the training process to use as many categories as possible, the data set was shuffled. Applying the algorithm on the test data set is equal to training the model on a given data set which gives the machine the ability to learn how the categorisation should be done.

After the model was created the machine was able to make predictions on new, for the machine unseen, data. The result of the prediction was a target category for every row in the test data set. The prototypes were then able to compare the predicted categories with the actual categories. These prototypes were also able to learn over time and they could be corrected by retraining the model with new targets. The model could be saved as a file on a computer and reused and retrained with new data sets. The prototypes could be retrained by using an already trained model and partially train it with new data.

An illustration of how a prediction could look is presented in figure 4.2 and figure 4.3. These two figures represent tests conducted using only the first two features, or the first two columns of the iris data set explained in 3.2. This data set consisted of three different categories, setosa (blue dots), versicolor (red dots) and virginica (yellow dots). The dots were x and y coordinates where x was the sepal length and y was the sepal width. Predicting on a new flower would automatically put the flower in one of the categories.
Figure 4.2: A trained model using the naive Bayes algorithm on only the first two features of the iris data set explained in 3.2.

Figure 4.3: A trained model using the support vector machine algorithm on only the first two features of the iris data set explained in 3.2.
Running the application resulted in an average precision score that was the same for the two prototypes. This score $P$ was calculated as followed:

$$P = \frac{t_p}{t_p + f_p}$$

(3)

Where $t_p$ is the number of true positives and $f_p$ the number of false positives.

4.2.2 Prototype using unsupervised machine learning

The finished prototype using the DBSCAN-algorithm worked similarly to the prototypes for the other algorithms. It had a number of input variables to tune the end result, such as number of rows to read, whether to normalise the data or not and values for the min\_samples and eps.

The prototype started by reading the selected CSV-file into a data-frame. It then iterated through all rows, column by column, building a suitable matrix for the following DBSCAN-algorithm. Depending on the type of column, numeric or categorical, it performed different operations and applied the “Bag of words”-method on the categorical columns, similar to the previous prototypes.

When the matrix was constructed it could be fitted in the DBSCAN-object provided by the SciKit library, alongside the input variables. On completion the model returned a list of suggested cluster labels, for each of the rows in the original CSV-file. The application continued by building a dictionary, pairing and grouping the items in their suggested cluster for cluster-analysis.

Since there are no clear, simple ways to validate the correctness of the suggested clusters, other than manual inspection, the application performed calculations and extracted a series of values to accompany the manual inspection:

1. Number of suggested clusters compared to the number of actual categories.
2. Number and percentage of items clustered, and number of unclustered items or “noise”.
3. The average homogeneity, or similarity, among the items in each cluster in regards to their actual categories. The value was calculated as an average between all suggested clusters.

An illustration of how the finished clustering by the DBSCAN-algorithm could look like is presented in figure 4.4. The points in the plot represents items from the iris data set, containing features like length and width of the flower-pedals, the same as in figure 4.2 and 4.3. The algorithm finds dense clusters and assigns labels, and in this case colors for visualization.
4.3 Test results

This section will present the results of the conducted tests starting with the tests that were done using the supervised machine learning algorithms and ending with the tests conducted using the unsupervised machine learning algorithm. Each algorithm were first validated using the iris data set and then tested with simulated company data.

**Naive Bayes and Support Vector Machine**

The two prototypes that were using supervised machine learning algorithms were validated by first training the machine on the iris data set explained in 3.2.4 and then letting the machine predict the categories on the same data set. Table 4.1 presents the result of the validation. The support vector machine performed best with 149 out of 150 items correctly categorised. These tests were conducted 30 times using the same 150 items and the precision showed the same result every time and therefore no confidence interval was calculated.
Table 4.1: Prediction on the iris data set using all 150 items and the algorithms naive Bayes and Support Vector Machine.

<table>
<thead>
<tr>
<th>Items:</th>
<th>Algorithm</th>
<th>Correct predictions:</th>
<th>Average precision:</th>
<th>Average time [s]:</th>
<th>Confidence interval with 99% certainty:</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>NB</td>
<td>144</td>
<td>96.0%</td>
<td>0.002</td>
<td>0.00%</td>
</tr>
<tr>
<td>150</td>
<td>SVM</td>
<td>149</td>
<td>99.3%</td>
<td>0.008</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

The following tests were all executed using the company’s simulated data explained in 3.2 and using the methods described in 3.4. The tests consisted of categorising different kinds of data sets using different sizes of the data set for the two supervised machine learning algorithms, naive Bayes and support vector machine. All tests executed were set to start on 100 items, and continuing to 1,000, 10,000, 50,000, 100,000, 200,000 and 300,000. The average precision and the time it took to train the items and predict categories on new items are presented in the following tables.

**Categorising raw data**

These tests were conducted with raw simulated data. No preprocessing was made before letting the machine do an automated categorisation. As shown in table 4.2 the average precision was relatively low and the tests show no significant increase in precision as the number of products were increasing.

Table 4.2: Average time and precision using raw data

<table>
<thead>
<tr>
<th>Items:</th>
<th>Algorithm</th>
<th>Average time [s]:</th>
<th>Average precision:</th>
<th>Confidence interval with 99% certainty:</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>NB</td>
<td>0.003</td>
<td>12.1%</td>
<td>7.90%</td>
</tr>
<tr>
<td>100</td>
<td>SVM</td>
<td>0.06</td>
<td>10.0%</td>
<td>4.33%</td>
</tr>
<tr>
<td>1,000</td>
<td>NB</td>
<td>0.05</td>
<td>5.2%</td>
<td>3.30%</td>
</tr>
<tr>
<td>1,000</td>
<td>SVM</td>
<td>0.5</td>
<td>7.2%</td>
<td>5.87%</td>
</tr>
<tr>
<td>10,000</td>
<td>NB</td>
<td>1.9</td>
<td>2.2%</td>
<td>0.90%</td>
</tr>
<tr>
<td>10,000</td>
<td>SVM</td>
<td>10.1</td>
<td>6.7%</td>
<td>5.20%</td>
</tr>
<tr>
<td>50,000</td>
<td>NB</td>
<td>21.5</td>
<td>2.3%</td>
<td>0.44%</td>
</tr>
<tr>
<td>50,000</td>
<td>SVM</td>
<td>168</td>
<td>11.1%</td>
<td>3.91%</td>
</tr>
<tr>
<td>100,000</td>
<td>NB</td>
<td>59.2</td>
<td>1.7%</td>
<td>0.32%</td>
</tr>
<tr>
<td>100,000</td>
<td>SVM</td>
<td>493.7</td>
<td>10.8%</td>
<td>3.92%</td>
</tr>
<tr>
<td>200,000</td>
<td>NB</td>
<td>363.2</td>
<td>1.8%</td>
<td>0.29%</td>
</tr>
<tr>
<td>200,000</td>
<td>SVM</td>
<td>1333.3</td>
<td>14.8%</td>
<td>6.47%</td>
</tr>
<tr>
<td>300,000</td>
<td>NB</td>
<td>729.5</td>
<td>2.1%</td>
<td>0.21%</td>
</tr>
<tr>
<td>300,000</td>
<td>SVM</td>
<td>2242</td>
<td>14.1%</td>
<td>7.38%</td>
</tr>
</tbody>
</table>
Categorising normalised data
These tests were conducted using normalised data. All numeric columns were rescaled in a scale between 0 and 1 using the normalisation method described in 3.2. The average precision was higher than on previous tests using raw data. There was a significant increase in precision following the number of items used. As presented in table 4.3 the highest precision was achieved using the support vector machine with 300,000 items.

Table 4.3: Average time and precision using normalised data

<table>
<thead>
<tr>
<th>Items:</th>
<th>Algorithm</th>
<th>Average time [s]</th>
<th>Average precision</th>
<th>Confidence interval with 99% certainty:</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>NB</td>
<td>0.003</td>
<td>11.3%</td>
<td>1.60%</td>
</tr>
<tr>
<td>100</td>
<td>SVM</td>
<td>0.006</td>
<td>17.5%</td>
<td>3.42%</td>
</tr>
<tr>
<td>1,000</td>
<td>NB</td>
<td>0.05</td>
<td>13.6%</td>
<td>0.86%</td>
</tr>
<tr>
<td>1,000</td>
<td>SVM</td>
<td>0.5</td>
<td>24.5%</td>
<td>1.74%</td>
</tr>
<tr>
<td>10,000</td>
<td>NB</td>
<td>2.2</td>
<td>13.1%</td>
<td>0.28%</td>
</tr>
<tr>
<td>10,000</td>
<td>SVM</td>
<td>9.7</td>
<td>28.9%</td>
<td>1.07%</td>
</tr>
<tr>
<td>50,000</td>
<td>NB</td>
<td>19</td>
<td>15.1%</td>
<td>0.13%</td>
</tr>
<tr>
<td>50,000</td>
<td>SVM</td>
<td>165</td>
<td>50.7%</td>
<td>0.61%</td>
</tr>
<tr>
<td>100,000</td>
<td>NB</td>
<td>65.8</td>
<td>15.1%</td>
<td>0.10%</td>
</tr>
<tr>
<td>100,000</td>
<td>SVM</td>
<td>496.5</td>
<td>58.7%</td>
<td>0.58%</td>
</tr>
<tr>
<td>200,000</td>
<td>NB</td>
<td>353</td>
<td>15.1%</td>
<td>0.12%</td>
</tr>
<tr>
<td>200,000</td>
<td>SVM</td>
<td>1301.2</td>
<td>63.2%</td>
<td>0.75%</td>
</tr>
<tr>
<td>300,000</td>
<td>NB</td>
<td>698.9</td>
<td>15.2%</td>
<td>0.09%</td>
</tr>
<tr>
<td>300,000</td>
<td>SVM</td>
<td>2215.1</td>
<td>63.5%</td>
<td>1.04%</td>
</tr>
</tbody>
</table>

Categorising cleaned data
These tests were conducted using cleaned data. Cleaning was done with the method explained in 3.2. The numeric columns were not rescaled during these tests and we see an increase in precision comparing to the tests on raw data, presented in table 4.2. In 9 of the 14 different tests using cleaned data, the average precision was higher than using normalised data but table 4.3 shows 63.5% as the best result whilst table 4.4 shows 30.3% as the best result.
Table 4.4: Average time and precision using cleaned data

<table>
<thead>
<tr>
<th>Items:</th>
<th>Algorithm</th>
<th>Average time [s]</th>
<th>Average precision</th>
<th>Confidence interval with 99% certainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>NB</td>
<td>0.003</td>
<td>19.7%</td>
<td>9.60%</td>
</tr>
<tr>
<td>100</td>
<td>SVM</td>
<td>0.05</td>
<td>24.2%</td>
<td>6.14%</td>
</tr>
<tr>
<td>1,000</td>
<td>NB</td>
<td>0.05</td>
<td>18.1%</td>
<td>5.96%</td>
</tr>
<tr>
<td>1,000</td>
<td>SVM</td>
<td>0.5</td>
<td>30.3%</td>
<td>2.11%</td>
</tr>
<tr>
<td>10,000</td>
<td>NB</td>
<td>1.3</td>
<td>22.7%</td>
<td>3.76%</td>
</tr>
<tr>
<td>10,000</td>
<td>SVM</td>
<td>8.9</td>
<td>27.3%</td>
<td>0.77%</td>
</tr>
<tr>
<td>50,000</td>
<td>NB</td>
<td>13.5</td>
<td>25.3%</td>
<td>3.49%</td>
</tr>
<tr>
<td>50,000</td>
<td>SVM</td>
<td>119.3</td>
<td>27.5%</td>
<td>0.76%</td>
</tr>
<tr>
<td>100,000</td>
<td>NB</td>
<td>0.357</td>
<td>24.2%</td>
<td>2.90%</td>
</tr>
<tr>
<td>100,000</td>
<td>SVM</td>
<td>385.4</td>
<td>28.1%</td>
<td>1.07%</td>
</tr>
<tr>
<td>200,000</td>
<td>NB</td>
<td>216.1</td>
<td>25.5%</td>
<td>4.10%</td>
</tr>
<tr>
<td>200,000</td>
<td>SVM</td>
<td>1046</td>
<td>28.1%</td>
<td>0.79%</td>
</tr>
<tr>
<td>300,000</td>
<td>NB</td>
<td>404.3</td>
<td>28.9%</td>
<td>4.60%</td>
</tr>
<tr>
<td>300,000</td>
<td>SVM</td>
<td>1811</td>
<td>28.0%</td>
<td>1.30%</td>
</tr>
</tbody>
</table>

Categorising normalised and cleaned data

These tests were a combination of the normalisation which was done in table 4.3 as well, and the cleaning which was done in table 4.4. As presented in table 4.5 both precision and time increases with the number of items and these tests showed 44.8% as the highest precision rate.
Table 4.5: Average time and precision using data that were both cleaned and normalised.

<table>
<thead>
<tr>
<th>Items</th>
<th>Algorithm</th>
<th>Average time [s]</th>
<th>Average precision</th>
<th>Confidence interval with 99% certainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>NB</td>
<td>0.003</td>
<td>13.2%</td>
<td>3.03%</td>
</tr>
<tr>
<td>100</td>
<td>SVM</td>
<td>0.06</td>
<td>21.5%</td>
<td>5.74%</td>
</tr>
<tr>
<td>1,000</td>
<td>NB</td>
<td>0.05</td>
<td>13.3%</td>
<td>0.86%</td>
</tr>
<tr>
<td>1,000</td>
<td>SVM</td>
<td>0.5</td>
<td>29.5%</td>
<td>2.19%</td>
</tr>
<tr>
<td>10,000</td>
<td>NB</td>
<td>1.5</td>
<td>16.3%</td>
<td>0.35%</td>
</tr>
<tr>
<td>10,000</td>
<td>SVM</td>
<td>9.4</td>
<td>36.5%</td>
<td>0.90%</td>
</tr>
<tr>
<td>50,000</td>
<td>NB</td>
<td>11.6</td>
<td>20.7%</td>
<td>0.41%</td>
</tr>
<tr>
<td>50,000</td>
<td>SVM</td>
<td>133</td>
<td>42.0%</td>
<td>0.39%</td>
</tr>
<tr>
<td>100,000</td>
<td>NB</td>
<td>36.9</td>
<td>23.6%</td>
<td>0.59%</td>
</tr>
<tr>
<td>100,000</td>
<td>SVM</td>
<td>408</td>
<td>43.1%</td>
<td>0.56%</td>
</tr>
<tr>
<td>200,000</td>
<td>NB</td>
<td>211</td>
<td>27.8%</td>
<td>0.60%</td>
</tr>
<tr>
<td>200,000</td>
<td>SVM</td>
<td>1093.9</td>
<td>44.0%</td>
<td>0.31%</td>
</tr>
<tr>
<td>300,000</td>
<td>NB</td>
<td>383.8</td>
<td>30.7%</td>
<td>0.42%</td>
</tr>
<tr>
<td>300,000</td>
<td>SVM</td>
<td>1905</td>
<td>44.8%</td>
<td>0.58%</td>
</tr>
</tbody>
</table>

In 2 of the 4 tests the average precision increases with the number of products, especially for the support vector machine using normalised data as seen in figure 4.4.

Figure 4.4: Chart showing the result of categorising normalised data. Average precision (y-axis) for different number of products (x-axis).
DBSCAN
The initial test for the prototype was the validation test using the iris data set. The clustering was executed 30 times and average values and confidence was calculated, as shown in table 4.7.

Table 4.7: Iris validation test

<table>
<thead>
<tr>
<th>eps</th>
<th>min_samples</th>
<th>Clusters Found</th>
<th>Items Clusted</th>
<th>Percentage Clustered</th>
<th>Avg. Cluster Homogeneity</th>
<th>Average time [s]</th>
<th>Confidence interval with 99% certainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>4</td>
<td>3</td>
<td>150</td>
<td>137</td>
<td>91.3%</td>
<td>84.1%</td>
<td>0.005</td>
</tr>
<tr>
<td>0.6</td>
<td>4</td>
<td>3</td>
<td>150</td>
<td>125</td>
<td>83.3%</td>
<td>97.9%</td>
<td>0.004</td>
</tr>
<tr>
<td>0.5</td>
<td>5</td>
<td>2</td>
<td>150</td>
<td>135</td>
<td>88.8%</td>
<td>76.1%</td>
<td>0.003</td>
</tr>
<tr>
<td>0.6</td>
<td>5</td>
<td>4</td>
<td>150</td>
<td>118</td>
<td>78.6%</td>
<td>94.6%</td>
<td>0.005</td>
</tr>
</tbody>
</table>

The following tests consisted of clustering data sets using different input variables, different sizes on the data sets as well as implementing normalisation on the numerical columns. Since these tests were executed on a standard personal computer the item count were limited to max 15,000 items. This was needed for the DBSCAN algorithm to avoid potential memory-issues. All tests executed were set to start on 1,000 items and continuing to 5,000, 10,000 and 15,000 items.

Clustering raw data
The first test consisted of clustering raw data, and the results are presented in table 4.8. The results show an initial high cluster-percentage with a cluster homogeneity of 80.5%. As the amount of items increases the cluster-percentage stays consistent, although the average homogeneity decreases substantially.

Table 4.8: Clustering raw data

<table>
<thead>
<tr>
<th>eps</th>
<th>min_samples</th>
<th>Clusters Found</th>
<th>Items Clusted</th>
<th>Percentage Clustered</th>
<th>Avg. Cluster Homogeneity</th>
<th>Average time [s]</th>
<th>Confidence interval with 99% certainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>1000</td>
<td>12</td>
<td>75.2%</td>
<td>86.9%</td>
<td>0.82%</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5000</td>
<td>25</td>
<td>90.1%</td>
<td>37.8%</td>
<td>18.4%</td>
</tr>
<tr>
<td>10</td>
<td>88</td>
<td>88</td>
<td>10000</td>
<td>68</td>
<td>87.7%</td>
<td>25.4%</td>
<td>72.20%</td>
</tr>
<tr>
<td>10</td>
<td>109</td>
<td>109</td>
<td>15000</td>
<td>112</td>
<td>87.9%</td>
<td>17.8%</td>
<td>180.013%</td>
</tr>
</tbody>
</table>

Clustering cleaned data
The second test consisted of clustering cleaned data, presented in table 4.9. The input parameters were set to the same values as in the previous test, and the result did not show any clusters in the first two iterations. Clusters were however found in the two bigger data sets.
### Table 4.9: Clustering cleaned data

<table>
<thead>
<tr>
<th>eps</th>
<th>min_samples</th>
<th>Items:</th>
<th>Categories:</th>
<th>Clusters Found:</th>
<th>Clustered Items:</th>
<th>Percentage Clustered:</th>
<th>Avg. Cluster Homogeneity:</th>
<th>Average time [s]:</th>
<th>Confidence interval with 99% certainty:</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>1000</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5000</td>
<td>58</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10000</td>
<td>88</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>15000</td>
<td>105</td>
<td>9</td>
<td>13673</td>
<td>91.2%</td>
<td>17.91%</td>
<td>25.210</td>
<td>1.9%</td>
</tr>
</tbody>
</table>

**Clustering raw, normalised data**

The third test consisted of clustering raw, normalised data, presented in table 4.10. This test showed that the found clusters were significantly lower than the actual categories. The percentage of clustered items also decreased on the increasing item count.

### Table 4.10: Clustering raw, normalised data

<table>
<thead>
<tr>
<th>eps</th>
<th>min_samples</th>
<th>Items:</th>
<th>Categories:</th>
<th>Clusters Found:</th>
<th>Clustered Items:</th>
<th>Percentage Clustered:</th>
<th>Avg. Cluster Homogeneity:</th>
<th>Average time [s]:</th>
<th>Confidence interval with 99% certainty:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>10</td>
<td>1000</td>
<td>5</td>
<td>3</td>
<td>754</td>
<td>75.4%</td>
<td>78.8%</td>
<td>0.850</td>
<td>6.0%</td>
</tr>
<tr>
<td>1.5</td>
<td>10</td>
<td>5000</td>
<td>58</td>
<td>5</td>
<td>1792</td>
<td>35.8%</td>
<td>36.98%</td>
<td>17.087</td>
<td>1.5%</td>
</tr>
<tr>
<td>1.5</td>
<td>10</td>
<td>10000</td>
<td>88</td>
<td>8</td>
<td>1840</td>
<td>18.4%</td>
<td>22.7%</td>
<td>20.632</td>
<td>3.0%</td>
</tr>
<tr>
<td>1.5</td>
<td>10</td>
<td>15000</td>
<td>105</td>
<td>10</td>
<td>1883</td>
<td>12.6%</td>
<td>17.0%</td>
<td>176.664</td>
<td>1.0%</td>
</tr>
</tbody>
</table>

**Clustering cleaned, normalised data.**

The final test consisted in clustering cleaned and normalised data, presented in table 4.11. The test resulted in a found cluster amount similar to the actual categories. The percentage clustered were also more consistent than the previous test. The average homogeneity in the first iteration was reasonably high, but the trend of decreased homogeneity continued as the item count increased.

### Table 4.11: Clustering cleaned, normalised data

<table>
<thead>
<tr>
<th>eps</th>
<th>min_samples</th>
<th>Items:</th>
<th>Categories:</th>
<th>Clusters Found:</th>
<th>Clustered Items:</th>
<th>Percentage Clustered:</th>
<th>Avg. Cluster Homogeneity:</th>
<th>Average time [s]:</th>
<th>Confidence interval with 99% certainty:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>10</td>
<td>1000</td>
<td>5</td>
<td>3</td>
<td>777</td>
<td>77.7%</td>
<td>82.0%</td>
<td>0.103</td>
<td>2.1%</td>
</tr>
<tr>
<td>1.5</td>
<td>10</td>
<td>5000</td>
<td>58</td>
<td>34</td>
<td>2817</td>
<td>56.3%</td>
<td>37.5%</td>
<td>2.131</td>
<td>3.1%</td>
</tr>
<tr>
<td>1.5</td>
<td>10</td>
<td>10000</td>
<td>88</td>
<td>50</td>
<td>6667</td>
<td>66.6%</td>
<td>22.8%</td>
<td>8.209</td>
<td>0.8%</td>
</tr>
<tr>
<td>1.5</td>
<td>10</td>
<td>15000</td>
<td>105</td>
<td>91</td>
<td>10119</td>
<td>67.5%</td>
<td>18.8%</td>
<td>19.668</td>
<td>0.6%</td>
</tr>
</tbody>
</table>
5. Analysis and discussion

This chapter presents evaluations of the result and the authors interpretations. Section 5.1 deals with the chosen algorithms which were used for evaluation and testing. Section 5.2 evaluates what kind of effect preprocessing the raw data had on the test results. Section 5.3 is an evaluation of the implemented prototypes based on their test result. Section 5.4 analyses alternative frameworks that are used in the field of machine learning. Section 5.5 deals with the social and economic aspects that has followed throughout this work.

5.1 Analysis of implemented prototypes

5.1.1 Supervised machine learning

Support vector machine was clearly performing better regarding precision. In 26 of the 28 different tests that were conducted, support vector machine outperformed Naive Bayes. However, when training and predicting using raw data, the highest precision average was only 14.8%. Regarding the different number of products used in the tests, for 2 of the 4 kinds of data sets the average precision increased as the number of products increased. This was expected for all 4 data sets because the more you train it the better it should perform but while using raw data this was not the case. Possibly because as the number of products increases, so does the effect of bad columns and the wide range between unscaled features.

As expected, naive Bayes was significantly faster than the support vector machine. The tests shows that the only factor for how long it took to train and predict was the number of items used. Using the support vector machine it could take up to about 36-37 minutes. Compared to naive Bayes which could take up to about 12-13 minutes and this is only for 300,000 items. As the number of products increase, the time increases, especially for the support vector machine. Fortunately one does not have to retrain the whole model every time a new data set is presented, as explained in 4.3.1 the model can be saved and partially trained.

It’s difficult to say exactly why the support vector machine performs so much better than the naive Bayes but one assumption is that the data used for these tests is not compatible with the naive Bayes principle that each feature of a product contributes independently to the result of the classification. While the support vector machine looks more directly at the data, naive Bayes tries to find indirect connections which makes the support vector machine a more advanced machine learning technique. This is also believed to be the reason for why the naive Bayes is faster than the support vector machine.
5.1.2 Unsupervised machine learning

The validation test for the DBSCAN-prototype were executed using different values for “min_points” and “eps”. These values can be hard to determine beforehand, when unfamiliar with the data. The selected values that were used in this work were based on recommendations from previous tests by other data scientist as well as evaluating tests.

The result form the validation tests looked as expected. The amount of suggested clusters were consistent between 2 and 4, with a cluster-homogeneity of between 76.1% and 97.9%. Since the actual amount of categories in the data set is 3, this is a positive result for the validation, and indicates that the prototype works as expected.

The results from the validation test also gives a better understanding of how the input parameters affect the clusters. By lowering the “eps”, fewer items will be allowed in the same clusters, and therefore lowering the percentage of clustered items. This creates more noise in the result, but also resulting in a higher cluster homogeneity as shown in table 4.7.

The following tests gave some unexpected results. Because of the size of the data it was extremely hard to determine the right input parameters for the algorithm. Only by testing an appropriate value could be selected, and used in the tests. The first test with completely raw data used “eps”=10 and “min_samples”=10. During the testing these values gave promising results at 1000 items. At 1000 items the prototype generated clusters with an average homogeneity of 80.5%. As the item count increased the homogeneity decreased significantly. There can be several explanations for this result, one being the compatibility of the data sets used. Since the data set contains over 500 columns, with the majority of these columns containing similar values, this will create an extremely dense graph. Each row could contain only a few different values together with several hundred similar. When the difference between each item gets this small, the task to find separate cluster increases dramatically.

The second test implemented used a cleaned data set which proved to have both a negative and positive effect on the resulting clusters. The algorithm took the same input parameters as the test before, and resulted in a significant decrease in clusters found. At the two higher item counts the algorithm did however start to find clusters with a slightly higher homogeneity average than the previous test.

The last two tests consisted in using normalisation for the numerical columns. The previous trend of lowering the average homogeneity did unfortunately continue in these tests.
The results for the DBSCAN-prototype were quite inconsistent, and not optimal for using with data sets similar to the ones used in this project. The trend of lowering the average homogeneity on increasing items seem to indicate that the data is too similar and consistent over the categories, and the task of separating the items into appropriate clusters gets more complicated on each item added.

5.2 Analysis of data preparation

Normalising the data with the rescale method proved to have a significant increase in precision both for the naive Bayes and the support vector machine. It had the biggest effect on the support vector machine. It also resulted in a slightly improved result for the DBSCAN-prototype when using normalised and cleaned data, although not as significant as the other two. The time it took to train and predict using the machines showed no significant change therefore normalising was proven to be an effective preprocessing method. Having different scales means having different ranges between features. If length is in meters and width is in centimeters than there is a bigger range between the widths of the products than between the lengths of them and that affects how a machine learning algorithm performs.

Cleaning the data by removing unwanted columns, as explained in 3.2 was proven effective. Comparing the result from the tests which used raw data with the tests that used cleaned data there was a significant increase in precision for both naive Bayes and support vector machine. The raw data included columns like ID’s and dates and these were not really product properties and they do not say anything about the item itself. Exploring these columns manually it was clear that the dates were the same for many products and therefore in the eye of the algorithms, the products were more similar to each other even though they could be two completely different products.

5.3 Analysis of automated categorisation

One of the goals with this work was to evaluate if the manual categorisation could be replaced with automated categorisation. The tests showed promising results, a machine can be trained and then used for new products but unfortunately the highest average precision from the two supervised machine learning techniques were 63.5% using 300,000 products. Increasing the number of products was clearly improving the result and if we were to continue increasing the sizes to 500,000 and even 1,000,000 and more the average precision should also increase. Unfortunately it was not possible using a personal computer because of memory issues. The iris validation showed that when using structured data, where all features
contained values and not 0 or “NULL” values which were the case with the data sets provided to us, all three algorithms performed well.

DBSCAN was proven to be a quick and effective way of grouping products into clusters. The downside of DBSCAN is that it does not learn over time like naive Bayes and the support vector machine does. When adding new data to the data set the machine has to be retrained all over again. With enough training, support vector machine should be able to replace the manual categorisation done at Navetti. It would have to be trained consistently until the average precision shows a satisfying percentage.

One nice to have feature from Navetti’s point of view was that the prototypes could apply pricing methods on new products, based on previous experience. This was not fully evaluated but when using the supervised machine learning algorithms, the new products were placed in a category and it was possible to assign a pricing method to the item based on what pricing method the products in that assigned category previously had. The result of this was not evaluated but it was achievable.

5.4 Analysis of alternative frameworks

There was a multitude of different methods to use during this project a qualitative decision had to be made about how to execute the given task. “Node.js” is a framework mostly for writing server side JavaScript programs, that also had started finding its way into the machine learning area. One library in particular seemed interesting, but after investigating the functionality and earlier implementations it was decided that the library was still to untested for this type of project.

“R” is another language often used in statistical analysis and data science and could have been a valid choice. However Python had, at the time of writing this paper, the best tools and libraries to quickly get started on machine learning.

5.5 Social and economic aspects

During the initial literary study and the developing process in this project both social and economic aspects has been taken in consideration. The goal of the project was to evaluate the possible methods of implementing machine learning algorithms in a way to automate product categorisation.

By developing applications that take advantage of machine learning, similar to the prototypes developed in this project, both economic and efficiency related advantages can be gained. Eliminating the human factor in these kinds of tasks will contribute to lower costs in terms of employees, as well as increasing the efficiency and perhaps also accuracy of the categorisation.
As shown during the literary study, several of the largest companies today have significantly fewer employees when compared to a few decades ago. The computerisation is a large contributor to this trend, and further implementation with different types of artificial intelligence will most likely continue to lower this number.

As machine learning and other types of computerisation will continue to lower the need for employees, new jobs will also be created. For example replacing ten conveyor belt workers with robots might result in the need for one robot-supervisor. At the same time completely new positions will be introduced within the fields of data mining, data management and data analysing among other. Unfortunately the people losing their jobs probably won’t have the required skillset to fill these positions.

There is no question about whether these technologies will or will not eliminate jobs executed by humans. The real question is if there will be enough new jobs created in the process.
6. Conclusion and future recommendations

The objective to evaluate different methods and possibilities to implement machine learning in product categorisation have been met. The initial literary study resulted in appropriate methods and algorithms to use for the continued development. Three different prototypes were developed and they all showed promising results when categorising the validation data set. The results from the tests using the simulated product data showed that the “Support Vector Machine”-algorithm gave the best categorisation compared to the actual categories. By increasing the training data for said algorithm the accuracy increased as well. The unsupervised algorithm DBSCAN gave more inconsistent results. It showed that adding more items to the classification resulted in more inconsistent results and more noise. This algorithm proved not suitable for this application according to the tests performed in this work.

The preprocessing of the data used consisted of removing non-important features and normalising the numeric data. This resulted in an improved result for all prototypes, but most significantly for the Support Vector Machine.

The results from the tests executed in this work shows that implementing machine learning algorithms, using suitable data preprocessing and a sufficient amount of data could be a viable solution to automate the product categorisation. Some algorithms performed better than others did and the selection of algorithm can greatly affect the result.

6.1 Future recommendations

Since the testing of the prototypes were executed on personal computers with relatively low performance and the data sets used for testing were extremely large in both dimensions, testing could be continued on more appropriate hardware. This would allow for more data to be tested and more true-to-life result.

Another improvement for future work is to more closely evaluate the data used in the data sets. The data preprocessing used in this project could be extended to further remove or keep certain features in the data sets.
7. References


http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8889613&fileOId=8889614


[7] Hsiang-Fu Yu, Chia,Hua Ho, Yu-Chin Juan, Chih-Jen Lin “A library for Short-text Classification and Analysis”, Published: Unknown, Cited: 2017-04-01

[8] Hsiang-Fu Yu, Chia,Hua Ho, Prakash Arunachalam, Manas Somaiya, Chih-Jen Lin “Product Title Classification versus Text Classification” Published: Unknown, Cited: 2017-03-22

https://ijcai.org/Proceedings/73/Papers/013.pdf
Cited: 2017-03-22

Published: 2004, Cited: 2017-04-11
http://mlg.eng.cam.ac.uk/pub/pdf/Ras04.pdf

http://cis.poly.edu/~mleung/FRE7851/f07/naiveBayesianClassifier.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.9324&rep=rep1&type=pdf


[18] Domenica Arlia, Massimo Coppola "Experiments in Parallel Clustering with DBSCAN”, Published: Unknown, Cited: 2017-03-22
https://pdfs.semanticscholar.org/60df/aff344b1cfd4b7013bbb27659974567063d.pdf


[22] Data Mining: Concepts and Techniques, Third Edition, Morgan Kaufmann, Jiawei Han, Micheline Kamber, and Jian Pei, Published: 2011, Cited: 2017-05-22


