CLASSIFYING CUSTOMER COMPANIES IN AN ENTERPRISE RESOURCE PLANNING SYSTEM USING MACHINE LEARNING METHODS

Lappeenranta-Lahti University of Technology LUT

Master's Program in Computational Engineering and Analytics, Master's Thesis

2022

Juho Kauppala

Examiners: Associate Professor, D.Sc. Tuomas Eerola
M.Sc. Pauli Immonen

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Analytics

Juho Kauppala

Classifying customer companies in an enterprise resource planning system using machine learning methods

Master's thesis
2022
48 pages, 17 figures, 6 tables, 0 appendices
Examiners: Associate Professor, D.Sc. Tuomas Eerola and M.Sc. Pauli Immonen

Keywords: pattern recognition, machine learning, multi-label classification, customer data analysis

Information systems such as smart phone applications collect large amounts of data about their users. The data is used mostly for the system's primary task, but machine learning methods can be used to get additional value out of the collected data. The goal of the thesis is to use customer data of an enterprise resource planning (ERP) system to classify customer companies based on which plugins they have selected, to create a plugin recommender model and increase plugin usage in the system. A review to customer data analysis and machine learning classification is presented. A method to classify the companies is proposed. Neural network, random forest, support vector machine forest, and metric learning models are compared with a dataset consisting of the ERP's data. Multi-label classification model's output is interpreted using top-k and threshold functions. The models are evaluated using the mean of plugin-specific F_1-scores. The highest mean (0.558) is achieved by a support vector machine forest. The result is considered promising, perhaps good enough for a plugin recommender, but currently insufficient for business-critical applications. Testing alternative solutions, such as smaller decision trees and top-1 interpretation, might improve results.

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Laskennallinen tekniikka ja analytiikka

Juho Kauppala

Asiakasyritysten luokittelu koneoppimismetodeilla taloushallinnon järjestelmässä

Diplomityö
2022
48 sivua, 17 kuvaa, 6 taulukkoa, 0 liitettä
Tarkastajat: Tutkijaopettaja, TkT. Tuomas Eerola ja DI. Pauli Immonen

Avainsanat: hahmontunnistus, koneoppiminen, usean luokan luokittelu, asiakasdatan analyysi

Informaatiojärjestelmät, kuten älypuhelinsovellukset, keräävät suuria datamääriä käyttäjistään. Tätä dataa käytetään enimmäkseen järjestelmien varsinaiseen toimintatarkoitukseen, mutta koneoppimisen avulla kerätystä datasta voidaan saada entistä enemmän hyötyä. Tämän työn tavoitteena on käyttää taloushallinnon ohjelmiston asiakasdataa asiakasyritysten luokitteluun heidän valitsemiensa lisäosien perusteella, jotta järjestelmään voitaisiin kehittää lisäosien suosittelija ja kasvattaa lisäosien käyttöä. Työssä esitetään kirjallisuuskatsaus asiakasdatan analysointiin ja koneoppimiseen, ehdotetaan mallia yritysten luokittelutehtävään ja vertaillaan neuroverkkoa, päätösmetsää, tukivektorikonetta sekä metriikkaoppimismallia käyttäen taloushallinnon ohjelmiston dataa. Usean luokan luokittimen tulosta tulkittiin valiten parhaat tai raja-arvon ylittävät luokat positiivisiksi. Mallien tarkkuudet laskettiin lisäosakohtaisten F_1-score-arvojen keskiarvona. Tukivektorikonemetsä tuotti parhaan keskiarvon (0.558). Tulosta voidaan pitää lupaavana, ehkä jopa riittävänä lisäosasuosittelijalle, mutta riittämättömänä bisneskriittisille sovelluksille. Vaihtoehtoisten ratkaisujen, kuten pienempien päätöspuiden ja top-1-tulkinnan, kokeileminen saattaisi tuottaa parempia tuloksia.

ACKNOWLEDGEMENTS

I would like to thank everyone who believed in me and supported me while working on this thesis, especially my family, girlfriend, friends, and colleagues. Regardless of how familiar you were with my topic and field, you showed interest in me and my progress. Even when I did not show it, I did need every bit of that support.

I am grateful for my supervisors Tuomas Eerola and Pauli Immonen for guiding me and helping me understand a little bit more about this world of opportunities, also called machine learning. Thank you Teppo Salmi for customer data access guidance and for your patience and interest in my thesis work, even though it was not expected of you.

Thanks to the student association of computational engineering and analytics, Lateksii ry, and the student union, LTKY, for making the studying environment at LUT University welcoming, supporting, and encouraging.

And thank you for reading!

Lappeenranta, October 31, 2022

Juho Kauppala

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
DAGSVM	Directed Acyclic Graph SVM
ERP	Enterprise Resource Planning
FTRL	Follow the regularized leader
GDPR	General Data Protection Regulation
KNN	K-Nearest Neighbour
MAR	Missing At Random
MCAR	Missing Completely At Random
NMAR	Not Missing At Random
PCA	Principal Component Analysis
RBF	Radial Basis Function
RFM	Recency, Frequency, Monetary
SGD	Stochastic gradient descent
SVM	Support Vector Machine

CONTENTS

1 INTRODUCTION

1.1 Background

As information technology keeps evolving in every segment of life, more capable computing and measurement devices generate large amounts of data. Increasing number of different information systems are also used in everyday life to automate tasks, which has led to those systems gathering large amounts of customer data. This has been the trend since 1980's, and as the amount and nature of customer data has evolved, so has the mindset of how it could be utilized in the information system [1]. The data, which has been originally used in the basic functionality of the information system, is now utilized to gain knowledge on a higher abstraction level. This is often done with data analysis tools and machine learning methods.

An example of a new use case for the collected data could be a grocery store or retailer company as described in [2]. The original reason to collect data of sold products has been to anticipate which products need resupply and to optimize storage space. Afterwards, a grocery store might notice that by analyzing which products the individual customers are buying, customers can be categorized, and advertising can be better targeted for them. This information can be either sold or utilized inside the organization.

The data usage of a digital Enterprise Resource Planning (ERP) is similar to the described example. It gathers large quantities of data about customer companies', which is mandatory for the system to function. However, there are notable opportunities to gain additional information from the patterns and correlations in the system data.

Machine learning studies algorithms that improve with experience. [3] Machine learning models are constructed in different ways, and some are much more complex than others, but they are similar in the way that decision logic is not programmed to the model. Instead, machine learning models are trained using real or artificial data, and with numerous iterations, they try to find the best possible parameters for themselves. The idea of automatically learning computer programs is decades old, but the increase of computational power and available data has created a growing demand for machine learning solutions in modern world applications.

Machine learning models excel at finding previously unknown patterns from data, which makes them a good tool to find higher level knowledge from customer data. Different big

data analysis tools exist, but often they display graphs and reports but leave the deduction to the users. Deep machine learning models attempt to perform the deduction part too, and in the best case, they might be able to give more accurate predictions than human analysts. However, their output must be validated with care to ensure the learning process succeeds in generalization but does not overlearn the training data.

In this thesis, machine learning models are trained using ERP system data to classify customer companies based on which plugins they have selected in the ERP. The purpose is to create a recommender which picks the most suitable plugins for a company and displays them in the ERP, so the customers would use more plugins. This way, machine learning could bring value for the product by discovering new uses for the existing data, which would otherwise be difficult to achieve.

1.2 Objectives and delimitations

The objectives of the thesis are as follows:

1. To provide a review to relevant published work concerning customer data analysis and data classification.
2. To implement a suitable machine learning model for customer company classification.
3. To prepare a dataset for the training and evaluation of the machine learning model.
4. To train and evaluate the implemented model.

The ERP system has multiple purchasable plugins, which are different additional services and have extra cost. The classification task is to find the plugins which a company is likely to use. During training, there is a many-to-many relation between companies and plugins, but the output of the model can be a single most likely plugin for the input company.

Although the data is from an ERP system, no business knowledge is applied in the model creation or data preprocessing. All data is treated as numerical values, and if feature values are categorized, the categorization is based on the data structure, such as variable means and variances, not any business-related knowledge.

Data fields can be used either directly with normalization, or indirectly with preprocessing. Preprocessing could mean calculating averages, numbers of occurrences, sums, or other similar indicators. For time-related data fields, like page visits or sent invoices, time series analysis could also be utilized if additional or more correct information can be extracted with it. Discrete data fields should be enumerated, so machine learning models can process them among other fields. A dimensions reduction method should be used to reduce the dimensions of the dataset.

All data should be treated anonymously. Names of companies, users or employees should not be included in the machine learning model or data preprocessing in any way. Aggregate values should be computed for sensitive information, such as employee salaries. Identifiers should be used in the presentation of results; for example, numerical identifiers should be given to companies and plugins.

1.3 Structure of the thesis

The rest of the thesis is organized as follows. The next two chapters provide a literature review to customer data analysis and data classification, respectively. In Chapter 4, a method is proposed to perform the task described in previous chapters. Chapter 5 describes experiments conducted and analyses the results. Chapter 6 discusses the results and future work, and Chapter 7 draws conclusions of the thesis.

2 CUSTOMER DATA ANALYSIS

2.1 Background

Because customer data has different forms in different information systems, there seems to be no standardized pipeline for its processing. However, many publications seem to prefer a few methodologies in customer data analysis [1, 2, 4]. The steps include collecting the data, accessing collected data, anonymizing customers, dealing with missing data, normalizing data, reducing dimensions, and selecting features, before performing the actual data analysis. Figure 1 presents the steps, with green background indicating common steps and yellow ones indicating optional steps. After discussing the common methods in this subsection, two publications, which are selected as example cases, are presented in the following subsections, and a short analysis is provided about how they process customer data in their work.

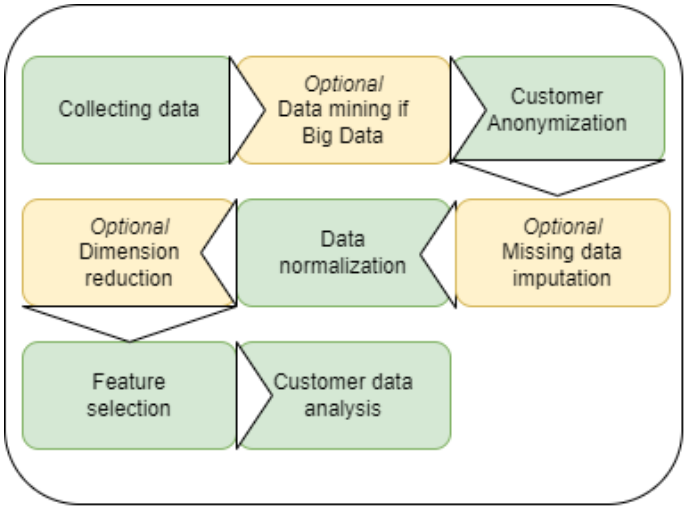


Figure 1. Rough pipeline of how customer data is typically treated.

Customer data analysis is often done by collecting event data of the usage of the system. Event data consist of different types of interactions the customer can do in the system. When a system has lots of users, the event data often fills the criteria of big data. In [5], big data is described as "datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyse." Therefore, big data mining tools, such as Recency, Frequency, Monetary (RFM) [6], are often utilized in customer data analysis [2, 5, 7].

What is also common in customer data analysis is the need to distance both the data

points (customers) and the data owner (the information system) from the presented results. Although this is a standard practice in all data analytics, extra care must be taken when analyzing real customer data. The General Data Protection Regulation (GDPR), which is applicable in the European Union since May 25th, 2018, has clarified the individuals' rights to their privacy, and it has generated discussion and new ways of data handling, like presented in [8].

Other common aspects in customer data seem to be that there are often a large number of features and having missing data is quite usual. Finding suitable feature selection methods for customer data appears to be a valid research question itself [9–11]. Data imputation is also common part of customer data handling [12, 13], and is present also in the example work presented in Section 2.3.

2.2 Example 1: Customer behavior analysis

In [2], a method to analyze the customer data of a large retailing company is presented. The retailing company consist of a large number of food-based stores. The aim of the paper is to segment different categories of users by using data mining techniques on a transactional database. First, typical shopping basket is gathered by forming product categories from similar products. Then a clustering algorithm is used to create clusters of similar transactions. Based on the clusters, six typical lifestyles are presented, into which the customers are intended to be divided. The lifestyles are presented from many perspectives, for example, proportion of products in the main category and proportion of products of each brand position (Figure 2). The brand position means whether the brand aims to have the highest quality in the market (Premium), the lowest price (Economic), or something in between.

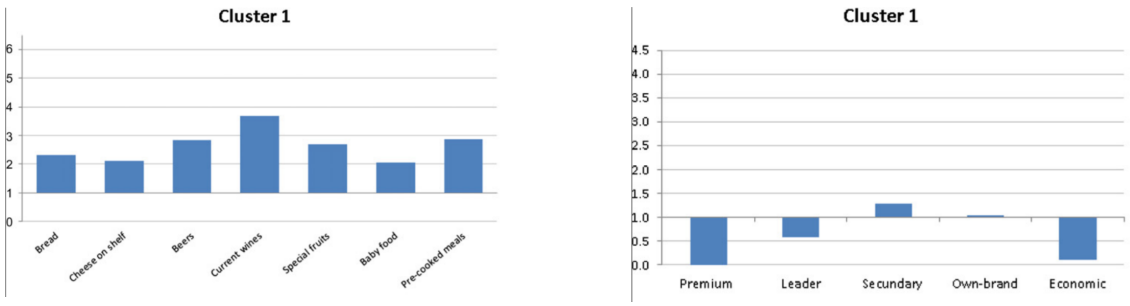


Figure 2. An example cluster: (a) Presented by the main product category; (b) Presented by brand position [2].

In [2], a relatively large data set is used in the analysis, allowing to drop out any information which could connect the results to any retailer or customer. The clusters contain information about what kind of products people in different lifestyle segments are likely to buy, and which is a good example of the higher abstraction level information that can be obtained. Using the results, the retailing company can optimize what they sell in which store by analyzing which products of which quality are often purchased by customers sharing the same lifestyle.

2.3 Example 2: Customer retention prediction

In [14], machine learning techniques for customer retention prediction are compared. 18 features are used in the customer dataset, which is from a telecommunication company. The dataset contains three categorical variables, including the independent variable "Churn" (synonym for customer retention), and 15 integer variables. The two categorical explanatory variables describe if the customer has a feature enabled, and the integer explanatory variables describe the usage of the service. The larger the integer value, the more the customer uses the service or the longer they have been a customer. A few example variables are "Total day calls made", "Total international minutes used", and "Number of customer service calls made".

The preprocessing steps used are data transformation, data cleaning and feature selection. In data transformation, categorical explanatory variables are converted into integers with values 0 or 1. Data cleaning is performed using random forest imputation technique [15] for numerical data and binary imputation [16] for categorical data. A combination of random forest technique [17] and Boruta [18] is used to reduce dimensions and select best possible features for machine learning models.

After the data preprocessing, 10 different machine learning methods are used to predict customer churn. The minimum and maximum accuracy were measured for each model and displayed ordered by the maximum accuracy (Figure 3), where accuracy is defined to be the proportion of correct classifications in all classifications. The objective of the paper was to provide a benchmark on how well the different models can predict customer retention, and they ended up recommending ensemble-based learning techniques.

Similarly to [2], the proposed method succeeds in distancing their study from recognizable traits of customers and the telecommunication company, to which the dataset belongs. All the explanatory variables are converted to integers to enable the machine

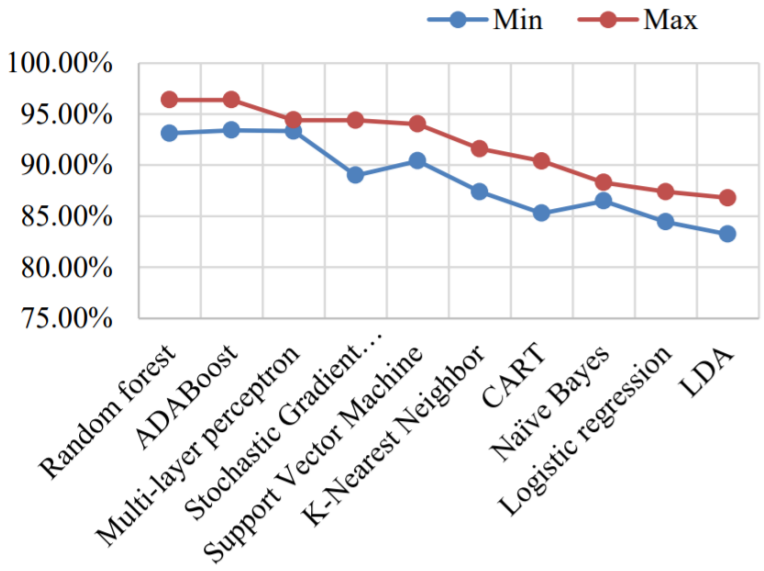


Figure 3. Accuracies of churn predicting machine learning models [14].

learning methods to take them into account without, for example, hard-coded input types or other special arrangements. Explanatory variables that were found to be not significant were left out from the final dataset.

2.4 Challenges of customer data analysis

Challenges in customer data analysis seem to arise from the following three sources:

1. The amount of data. Big data mining tools are often necessary to be able to access customer behaviour data, especially when its transactional or event-based.
2. The disintegrity of the data. Data collection methods and collected features tend to change over time, which creates a situation where some features are systematically missing for some span of time.
3. The interpretation of the data. While it is common to calculate aggregate values of the data, like means, rates, and ratios, a bigger challenge arises in drawing conclusions based on the values, because it requires knowledge of causation between the data (or the aggregate values) and high-level information.

Especially the interpretation of the data is a challenge to which deep machine learning models offer potential tools. Due to their ability to learn the features of a dataset, the deep

machine learning models can compensate for the lack of domain expertise of the dataset. However, some domain knowledge should still be present to validate the output of the deep learning model.

3 DATA CLASSIFICATION

3.1 Machine learning techniques for classification

3.1.1 Overview

Machine learning is a way of programming where the program learns the characteristics of the data without being explicitly programmed for the case. The machine learning model maintains an internal state of parameters, which it uses while performing its task. When the model is trained, the parameters are updated so that the model would produce results closer to the values in the training data. Depending on which machine learning algorithm is used, the decision rules may be understandable for humans or not [19]. The parameters which describe the machine learning model itself are called hyperparameters. Suitable values for hyperparameters enable the model to learn and generalize the patterns or correlations in the data, but prevent it from overlearning them. Overlearning means that the model learns to make decisions based on patterns which exist in the training data by chance but are not generally applicable [20].

Machine learning models require input and produce output. In classification, the input is often called features, and the output is called labels. Typically, machine learning classifiers are designed to work with numbers, but in practice the input can be in any format, such as image, text, or voice, as long as the machine learning model has means to convert it to numerical representation. A good example about this is the Convolutional Neural Network (CNN), in which a neural network uses convolutions to extract numerical features out of images.

While training, the dataset is given as input to the model usually multiple times. Going through the training dataset once is called an epoch. Traditionally, a model training consists of hundreds of epochs. [21]

Classification tasks occur when data points are divided into a pre-defined number of categories, and the task is to select the proper category for each data point based on its features. Classification is one of the most popular tasks for machine learning [22]. A typical pipeline for creating machine learning classifiers includes data preprocessing, classification model implementation and training, tuning and testing the model, evaluating the model and comparing alternative options, and repeating the steps until the model is estimated to be good enough. The pipeline is presented in Figure 4 with green background

indicating common steps and yellow ones indicating optional steps. There are different ways to divide classification methods into categories, but arguably the most common is the division presented in Figure 5, which is used, among others, in [19, 23, 24].

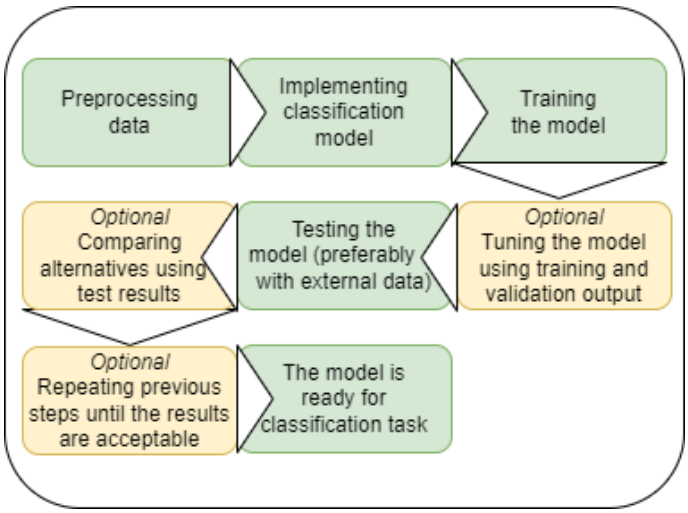


Figure 4. An example pipeline of creating a machine learning classification model.

Using machine learning for classification tasks has its characteristic challenges. Different models have different parameters, but most models need trial and error to find optimal hyperparameters for each unique classification task. Expertise is often needed to be able to tell whether the model is overlearning the training data or not. Some machine learning models have poor transparency, which means that their internal state and decision flow are hard to interpret. On top of that, training a model usually requires a lot of computational resources, and models can be relatively large files to transfer.

As explained in [25], the opportunities of machine learning classifiers surpass those created with traditional programming. Machine learning models can learn to detect complex patterns, of which the creators have not been aware to start with. Models can be created for many fields and can process different types of data, including video and audio data. There are also benefits in decision rules not being created by human programmers, since there is no chance of forgetting to observe or compare a value.

3.1.2 Logic-based techniques

Logic-based techniques contain decision trees (Figure 6) and rule-based classifiers [23]. They seek to acquire results by setting a group of tests, and choosing the next test based

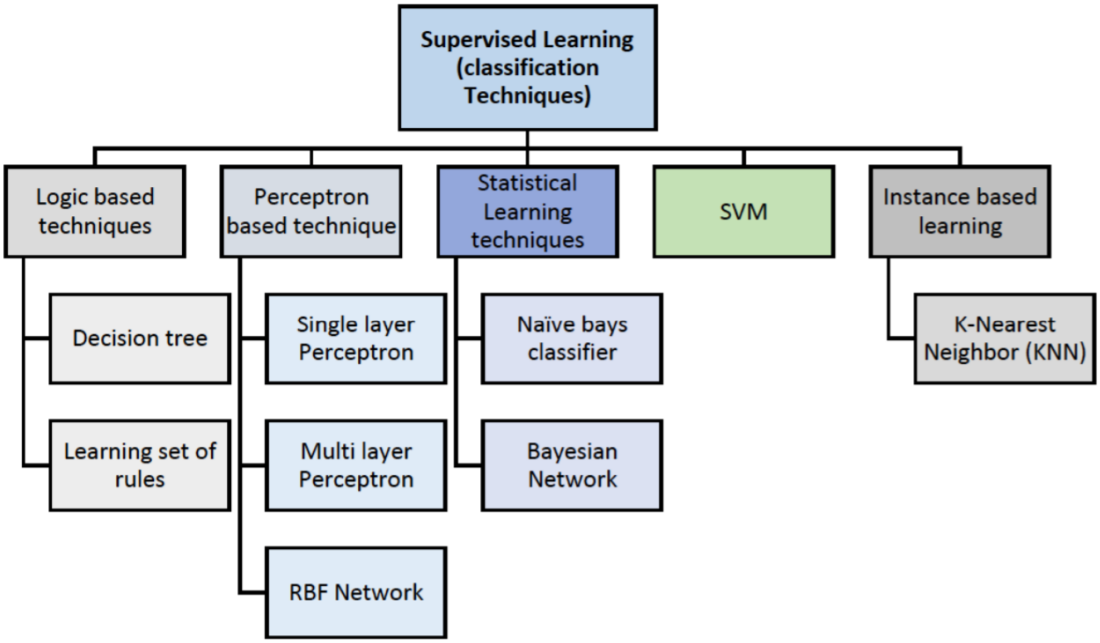


Figure 5. Categorization of machine learning classification techniques [23].

on the result of the previous one. The execution of a decision tree resembles following a flowchart, where the next node to process depends on which decision was made on the previous one. After having arrived in the last node of a branch, which is called a leaf node, the result of the last test is the classification result.

The training of decision trees consists of two phases: Tree growth and Tree pruning. [23] The Tree growth phase creates new nodes in the decision tree until most of the samples are classified correctly. The Tree pruning phase reduces the size of the tree while preserving the classification functionality as well as possible, which makes observation of the tree easier. In [23], two famous decision tree algorithms called ID3 (Iterative Dichotomiser 3) and C4.5 are presented and explained.

3.1.3 Perceptron-based techniques

The perceptron [26] is a classifier which imitates the way neurons process and transmit signals in human brains. A single perceptron receives an input vector, multiplies each input value with its corresponding weight, sums the products, and returns 1 if the sum is above a threshold, and otherwise 0. A perceptron is only capable of solving linearly separable problems, as it is basically a linear combination of inputs with a bias [19]. A simple perceptron is presented in Figure 7.

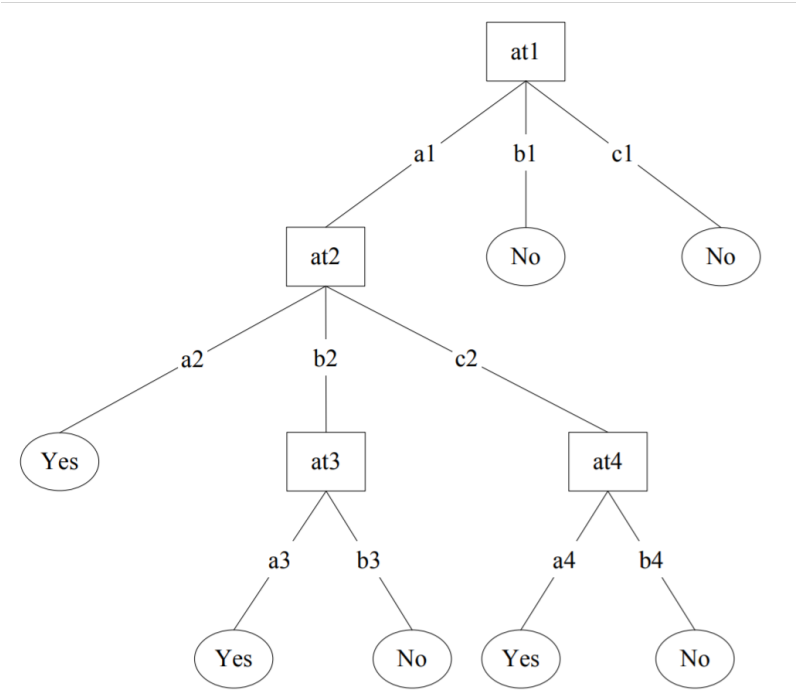


Figure 6. An example structure of a decision tree [19]. Yes and No are outputs of the tree, atn stands for the n:th node and an, bn, cn, etc. for the corresponding decision options.

When multiple perceptrons receive the same inputs, they form a layer. When the outputs of perceptrons act as inputs for the perceptrons of the next layer, they form a multilayer perceptron [24]. A multilayer perceptron is capable of solving linearly inseparable problems, such as the XOR problem [27].

More modern perceptron-based techniques evolved from the multilayer perceptron is the Artificial Neural Network (ANN). ANNs contain usually three segments: Input layers, hidden layers, and output layers. Input layers accept data, hidden layers process it, and final result is readable from the output layers. A common challenge with ANNs resides in how many and how large hidden layers it should contain. Too few and too small hidden layers may fail to capture the features in the data, and too many and too large hidden layers are likely to overlearn the local features of the training data [24].

ANNs can have specialized neurons for processing certain type of data. A well-known example of this is the CNN, where the input layers perform convolution on image data. A Radial Basis Function (RBF) network is another special case of an ANN. It has three layers: an input layer, a hidden layer with radial basis activation function, and an output layer. Its working principle is presented in more detail in [24] and [19].

While training an ANN, a backpropagating algorithm is used to update the weights of each

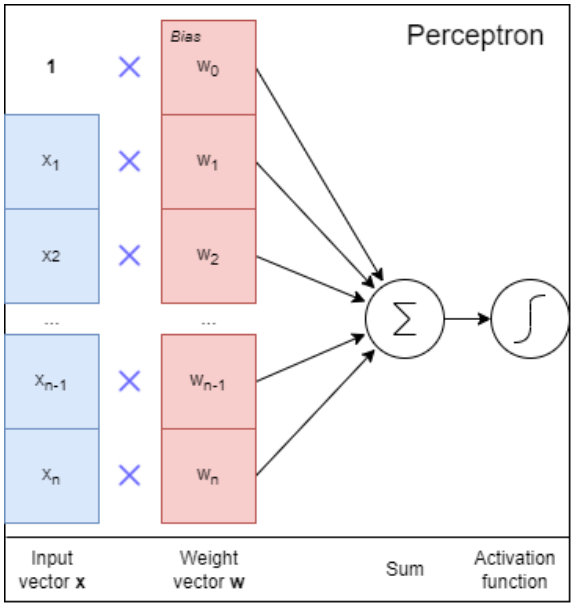


Figure 7. A perceptron illustrated.

neuron. The difference of the correct result and the current result forms a gradient, which tells each neuron separately how its weights should be modified. The backpropagation algorithm is explained in more detail in [19].

3.1.4 Statistical learning techniques

Statistical learning techniques differ from other presented categories so, that they assume there are probabilities for phenomena in data and try to estimate those probabilities and their dependencies on each other. They utilize a priori probabilities and a posteriori probabilities as well as conditional probabilities to form the calculation models. Two most important statistical learning techniques are the naïve Bayes classifier and the Bayesian network [19].

Bayesian probability models are often constructed to be in the form of a Directed Acyclic Graph (DAG). A DAG comprises nodes which are connected to each other directionally and without forming a cycle. Using DAGs, it is possible to apply the Bayes theorem [28] and form the conditional probabilities for each node. Depending on whether the shape of the DAG is fixed or not, the training of the Bayesian network includes either only tuning the probabilities or changing the shape of the DAG, too [23]. Figure 8 shows an example of a Bayesian network of juvenile *Astacopsis gouldi* habitat suitability.

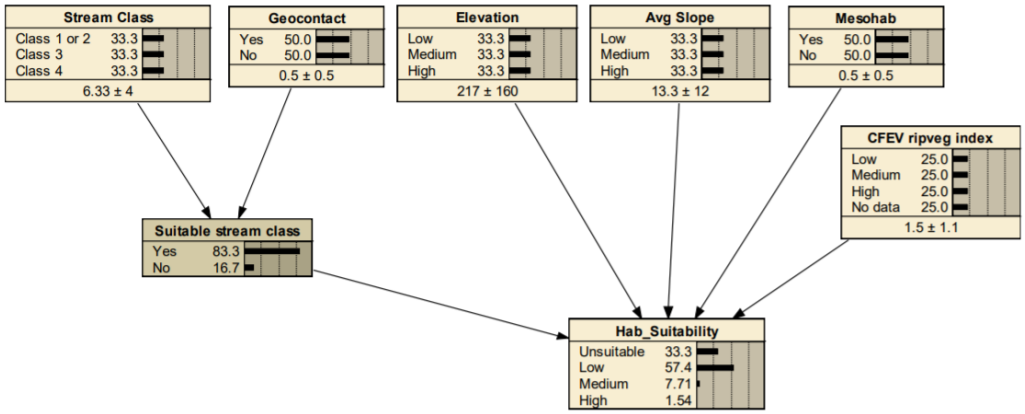


Figure 8. An example Bayesian network [29].

3.1.5 Support vector machine

Support Vector Machine (SVM) [30] is a binary classification algorithm which attempts to divide two classes by fitting a decision line or hyperplane between the classes. It operates by choosing the data points nearest to the decision plane as support vectors, and resulting in such a decision hyperplane, which has the greatest margin between classes, i.e., the greatest distance from the hyperplane to the nearest points of both classes [22]. Two example images of how SVM generates its decision boundary are shown in Figure 9.

There are several ways to apply SVM in multi-class classification, 3 of which seem to be most common and simplistic. They are called One-Against-All, One-Against-One, and Directed Acyclic Graph SVM (DAGSVM). In One-Against-All, there is one SVM for each class, and they are trained with their corresponding class data as positive labels and the rest as negative. In One-Against-One and DAGSVM with k classes, one SVM is trained for each distinct pair of classes, resulting in $\frac{k(k-1)}{2}$ classifiers. In One-Against-All and One-Against-One, the result is achieved with models voting. In DAGSVM, a DAG with binary nodes and k leaves is constructed, and the SVM models produce the decisions for each node. [31]

A basic SVM can only classify linearly separable data. In linearly inseparable cases, kernel trick can be applied to make the classification possible. A kernel is a nonlinear function which transforms the data to make it linearly separable. In [32], a high-level presentation of the kernel trick is provided with examples.

It is stated in [23], that SVM in classification, learning, and prediction tasks, SVM has the reputation of being one of the most reliable and efficient method. Another advantage of SVM is its robustness with different kinds of classification problems. Linearly inseparable

cases and high dimensionality are handled by SVM relatively well. The drawbacks of SVM are its high number of parameters and poor interpretability [19].

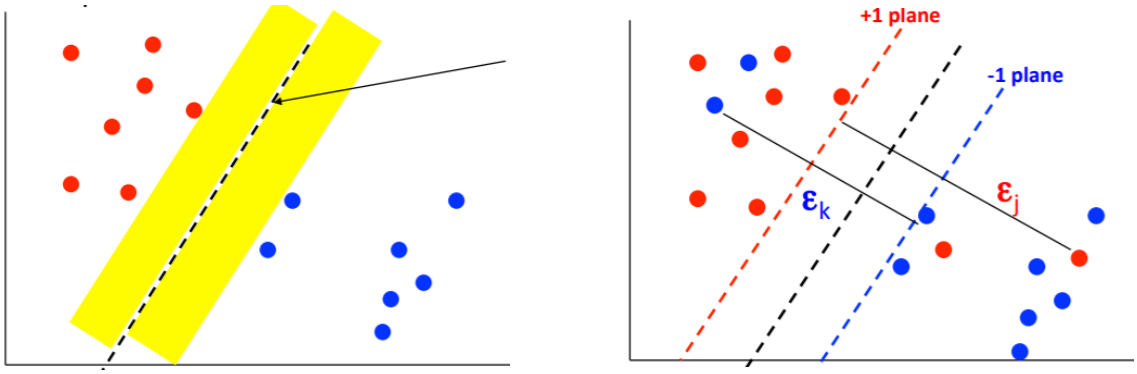


Figure 9. Illustration of SVM: (a) SVM creates maximum margin to nearest points; (b) In a linearly inseparable case, SVM considers the distance to the points on the wrong side of the decision boundary [33].

3.1.6 Instance-based learning

Instance-based learning algorithms are lazy-learning algorithms, meaning that they do not form rules or tests during training, but instead they perform the generalization or induction process during the actual classification. The most notable instance-based learning method is the K-Nearest Neighbour (KNN) algorithm. It selects k nearest neighbours of the input data point, and classifies it based on which class is the most represented in the group of neighbours [24].

Figure 10 demonstrates how the value of parameter k impacts the KNN process. The yellow point is to be classified to either Class 1 (green) or Class 2 (blue). If $k = 1$, Class 1 would be chosen, because the nearest neighbour belongs to Class 1. However, if $k = 3$, Class 2 would be chosen, because 2 out of the 3 nearest neighbours belong to Class 2. With $k = 5$, Class 1 would be chosen again. Figure 10 does not represent any real data, and is crafted to behave so that KNN changes its value based on the parameter k, but similar behaviour affects real-world scenarios, too.

Based on [23], the advantages of KNN are its robustness, simplicity and transparency. It has severe disadvantages, though, which make it rarely the best choice. Based on [19], the major disadvantages of KNN include long classification time, large storage requirement. There is also no means to optimize the parameter k without remarkable computational

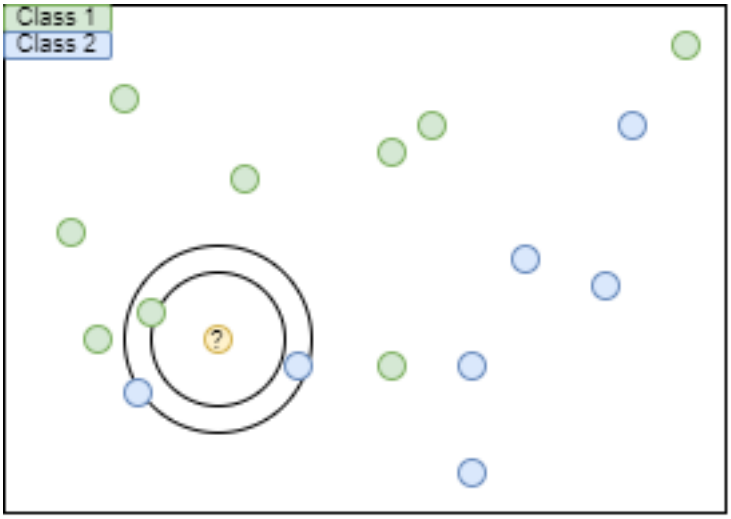


Figure 10. An example illustration of how KNN would classify a data point.

time, because having different values is a trade-off between execution time and model robustness [19].

3.1.7 Comparison of techniques

Different machine learning classification techniques are tested in [19]. It is discovered, that when the data has continuous features and many dimensions, SVM and neural networks seem to be the best choices, but to achieve good results, they require more training data than the other techniques. For categorical features, logic-based techniques are recommended. Naïve Bayes method is noted to perform well with relatively small datasets. A table of test results is presented, where the performance of different models is presented in respect to 13 measurements, which include among others accuracy, speed, transparency, and tolerances to different data features. It is suggested that the next step to achieve better results is to combine individual classifiers.

Similar results are presented in [24]. Combining different methods using so called ensemble learning is claimed to be the best option for raw classification accuracy, but its weaknesses are increased need for storage space, computational power, and decreased transparency of the classification process. In [23], methods are not compared to each other in particular, but similar advantages and disadvantages are mentioned there, too.

3.2 Typical data preprocessing for classification

Figures 1 and 4 present the typical steps in customer data analysis and machine learning classifier creation. The typical preprocessing steps of classification data resembles the steps in Figure 1. An important addition to the steps is that input data for machine learning classifiers should usually be numerical or convertible to numerical, because calculations are performed on the input values. If the input data is not in numerical form, then the input layer typically calculates numerical values out of it, e.g., by applying convolution for images or using text processing methods.

Like in any other fields of data analysis, normalization is typically included in data preprocessing. Standard score is a simple normalization method targeted especially for normally distributed data, where variance is reduced to 1 and mean to 0, without affecting the relative distances between data points. Other normalization methods

Dimensionality reduction can also be necessary, since the size of the model is in proportion to the required computational power to train the model, and therefore to the required time. Principal Component Analysis (PCA) [34, 35] is one of the most commonly used global methods for dimensionality reduction, because of the way it preserves as much of the data variation as possible. If global dimension reduction methods do not produce sufficient output, for example because of data sparsity, local dimension reduction methods like whitening or sphering can be used instead [36].

3.3 Classification using partial data

A standard way to store especially relational data is to define a set of different features for which each point of data has one value [37]. The features have different data types, such as integer, string, or byte, depending on the kind of data stored in them. In table-like thinking, columns stand for features and rows for data points, respectively. This way the columns and rows form a 2-dimensional grid, where there is equal amount of storage space for each data point.

A dataset produced by collecting measurements is often partial, meaning that there are systematic missing values. A measurement device could be malfunctioning or out of use for a span of time, causing all the values for a specific feature to be missing or erroneous for all data points. Another scenario which produces missing values is when measurements for a new feature are started, and the values for that feature will be missing for all

values gathered before the time of starting. It is usual that some features have missing values in large datasets, and there has been research on how to compensate for this flaw in the dataset.

In [13], a Theory of Missing Data (based on [38]) is presented, and the categorization of the types of missing data in machine learning, how they impact the model, and what can be done to mitigate the impact is presented. The classification for missing data types is presented in Table 1.

Table 1. Types of missing data [13].

Missing Completely At Random (MCAR)	Whether the data is missing is completely random, and existing data can be considered a valid random sample of the whole data.
Missing At Random (MAR)	Whether the data is missing can depend on something else than the value of the data. The existing data can be biased.
Not Missing At Random (NMAR)	Whether the data is missing depends either on the value of the missing data or the value of latent variables contained in the missing data.

In short, it is stated that if missing data is of type MCAR or MAR, it can often be used with precautions. However, if the condition to satisfy either of those types is not met, missing data is of type NMAR, and additional actions must be considered to be able to model the data.

An approach proposed by [39] is to use different subsets of the data, divided in a way that no subset itself is partial. While the data is not partial in the sense described above, the dataset is still very sparse with features. Machine learning is used for data analysis too, and the solution is to have multiple machine learning models who vote for the output by analyzing different parts of the data — the first one analyzes extracted local features, the second one generalized features, and the third one raw data. The output of the model is the mode output of the three machine learning models.

In [13] and [40], the following imputation options are presented. Missing data can be set as zeros or mean or median values, but it is suggested that using statistical imputation methods for machine learning training data produce better quality models. Advanced imputation techniques mentioned include KNN imputation, Predictive Mean Matching, Bayesian and non-Bayesian Linear Regression, and Random Sampling.

In KNN imputation, missing values are imputed by copying values from similar points

in the same dataset. In Predictive Mean Matching, missing values of a continuous variable are imputed by sampling from the observed values of the variable while matching predicted values as closely as possible. In Bayesian Linear Regression, univariate missing data is imputed using Bayesian linear regression analysis. In non-Bayesian Linear Regression, imputation is done by fitting a linear regression line and picking values from a spread around it, ignoring model error. Random sampling imputes data by selecting observed data points randomly into missing data cells. [40]

4 PROPOSED METHODS

4.1 Goal of the proposed methods

The objective of the method is to produce a machine learning classifier for classifying customer companies based on the plugins they have selected in the ERP. The companies have at least one plugin selected, but the maximum number is not limited. The proposed method should learn to output the correct labels based on which plugins the company has selected. The method does not need to output the number of plugins, only plugin-specific probabilities. The number of positive labels in the output is determined using different output interpretation functions, which are introduced in Section 5.2.1.

The classifier model will receive numerical data about the customer companies as input. The data will consist of both ERP-specific usage data, e.g., how often they use the system and which features they use, and business data, e.g., how high their turnover is. The data the model receives will be anonymous and normalized.

Neural network, Random forest, and SVM can be altered to perform multi-label classification. Metric learning, however, has a different working principle and cannot be straightforwardly converted. It is proposed that the techniques are implemented and compared so, that the three are implemented as multi-label classifiers and metric learning as a single-label, multi-class classifier where different combinations of plugins form classes.

4.2 Multi-label customer data classification using machine learning methods

4.2.1 Multi-label problem in classification

Typically, the best-performing machine learning classification methods are neural network, random forest, and support vector machine. The simplest classification tasks are single-label classification, and to be able to classify multi-label data, the models need some adjustments. The designs of the methods impact how much alteration is needed.

4.2.2 Neural network

The neural network is a black box method, which means that it is challenging to view the decision logic once the network has been trained. Looking at Figure 11, this can be understood when keeping in mind that usually there are many hidden layers, and they can contain hundreds of neurons each. During training, the weights for the network's parameters are tuned according to how correct results the model is able to produce. The weight adjustments are controlled by a loss function, which judges how correct the output was and how much the weights should be changed.

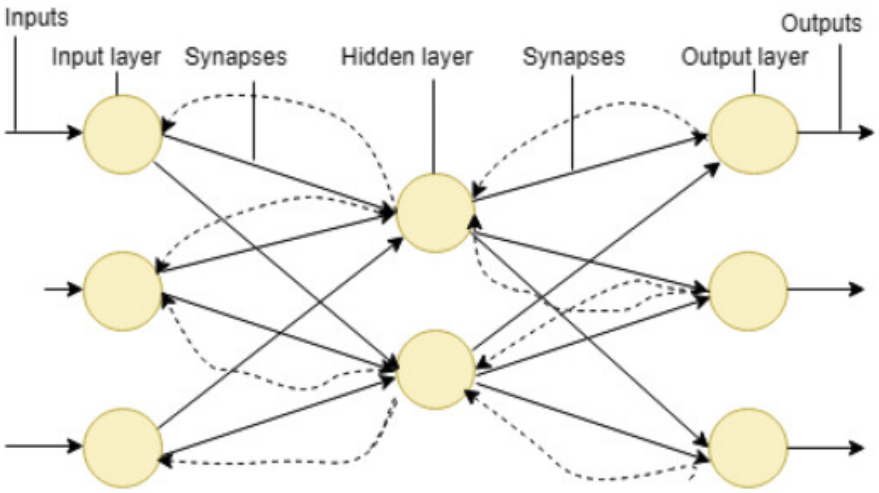


Figure 11. Neural network visualized in [41].

The decision of which loss function to select depends on the problem domain. If the neural network is to produce numeric values, a regression loss, such as mean squared error, should be selected. Binary classification uses binary cross-entropy function to calculate loss. Multi-class single-label classification can be achieved with categorical cross-entropy loss function. In [42], a loss function for neural network multi-class multi-label classification is proposed, and it is chosen to be utilized in this work, too. The function is called Bp-mll, and its basic principle is as follows.

With number of data points m, number of classes Q, model output c, desired output d, and global error E, the loss (global error) is defined as

$$E = \sum_{i=1}^{m} E_i \quad , \tag{1}$$

where

$$E_i = \sum_{j=1}^{Q} (c_j^i - d_j^i)^2 \quad . \tag{2}$$

In [42], the equations are optimized further, but in this context, it is enough to understand that the prediction loss is the sum of squared errors per label, and the global loss is the sum of prediction losses. It is also good to keep in mind, that global does not necessarily mean the entire dataset, but if training was done in batches, global would mean the batch error.

The proposed neural network consists of the following layers:

1. Input layer.
2. 1 or more hidden layers and Rectified Linear Unit activation function [43].
3. Dropout layer to prevent overfitting by randomly setting some of its inputs 0 during training.
4. Decision layer with Softmax activation function [44].

The number of hidden layers and dropout rate should be tested with different values to find the best neural network architecture for this task.

4.2.3 Random forest

Random forest (Figure 12) is a machine learning method which comprises of multiple decision trees. The decision trees are simple classifiers which have a discrete set of possible output values. Normally, some kind of voting mechanism is implemented in random forests to make a decision based on the outputs of the trees [17]. However, multi-class multi-label classification can be achieved by merely outputting the sum or mean of the trees' outputs. This way, the random forest is capable of performing multi-class multi-label classification with only a little adjustment.

The random forest consists of multiple decision trees similar to each other. The output of the forest is the mean of the outputs of the individual decision trees. The trees have Sigmoid [44] activation functions in their nodes, and the forest has Bp-mll loss function.

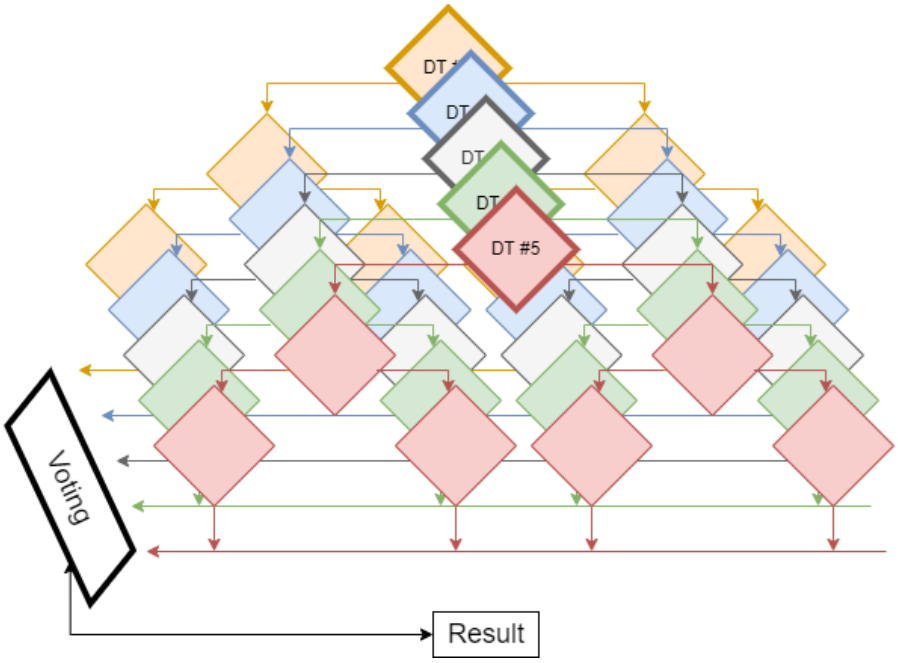


Figure 12. Random forest visualization where DT stands for decision tree.

The numbers of trees as well as tree depth should be tested to find the optimal forest architecture.

4.2.4 Support vector machine

SVM is another binary classifier which operates by trying to maximize the distance on both sides of its decision plane (Figure 9). In Section 3.1.5, different ways to combine SVMs for multi-class output are presented, but without modification they would still be single-label classifiers. A binary or single-label classifier is not sufficient for this work, so the SVM should be modified to be able to classify linearly inseparable, multi-label cases.

Using a kernel enables the SVM to classify linearly inseparable data. The Random Fourier Features kernel seems to be one suitable option. Its working principle is based on the following theorem: "a continuous kernel $k(x, y) = k(x - y)$ on R^d is positive definite if and only if $k(\delta)$ is the Fourier transform of a non-negative measure." [45]

To make multi-label classification possible, multiple SVM models should be utilized. In the same way how random forest combines its output from multiple decision trees, having an ensemble of SVMs and a voting mechanism should enable multi-label classification for the SVM. It would be neither One-Against-One nor One-Against-All approach, but the SVM model instances should be able to learn whatever significant patterns there are in

the data.

The proposed SVM method for this classification task is what could be called an SVM forest. The forest would be similar to random forest with the exception that instead of decision trees, there would be a number of SVM instances which use the Random Fourier Features kernel. The forest's output would be the mean of the SVM outputs, and the forest would use the Bp-mll loss function to calculate the loss of the prediction.

4.3 Similarity measurement with metric learning

4.3.1 Working principle of distance metric learning

Distance metric learning (described in [46]) is a method that fundamentally differs from other machine learning classifiers described in Section 4.2. Instead of trying to learn a way to separate classes from each other using mathematical expressions, metric learning maps the data points to a new space, with number of dimensions as a hyperparameter (Figure 13). The mapping is considered successful if points of the same class are close to each other and points from different classes are far from each other. The process of mapping is called embedding, and different embedding functions can be used, both linear and non-linear. If the embedded space has fewer dimensions than the original (observed) space, then the method works as a dimension reduction tool, too.

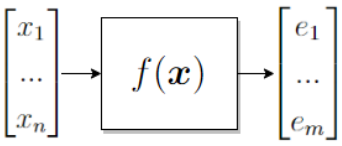


Figure 13. A metric learning model converts an observed point $\boldsymbol{x}$ with n dimensions to an embedded space point $\boldsymbol{e}$ with m dimensions. Typically, $m \leq n$.

A metric learning model is trained using groups of data points. Like many other machine learning models, a metric learning model training is controlled with a loss function which evaluates the goodness of the prediction made by the model. The simplest loss function uses pairs of data points (tuplets). In the training process, the embedding function is trained to place tuplets of same class near each other. Groups of three data points (triplets) are often used so, that there is one anchor point, one point from the same class (positive), and one point from a different class (negative). This way the embedding function is trained simultaneously to both place the anchor and the positive near each other, and

the anchor and the negative far from each other. Loss functions with more than three simultaneous data points exist, too, but they seem to be less common at the time of writing.

Metric learning can be trained as strongly supervised, weakly supervised, or unsupervised. Strongly supervised training means that each training data point has a class label. Weakly supervised training means that tuplets or triplets are provided, but no information about their classes is provided; only that these points are of the same or of a different class. Unsupervised training is merely calculating the covariance, which does not utilize the characteristics and strengths of metric learning.

The metric learning itself does not limit what is done with the resulting embedded space. For classification, it is common to use the KNN to perform classification, since the data points of same class should lie close to each other in the embedded space. However, the embedded space can also be used, for example, for clustering.

4.3.2 Pipeline for metric learning in customer classification

In this work, a simple metric learning method is implemented to compare against more traditional classifiers. As metric learning needs the data points to be either same class or different class, the concept of "partially same class" present in multi-label classification could not be implemented. Therefore, with number of classes n, an integer label l is calculated for each binary vector of labels y as follows.

$$l = \sum_{i=1}^{n} 2^{i-1} y_i \quad . \tag{3}$$

After converting the multi-label problem to a single-label one, the pipeline of metric learning model (Figure 14) is straightforward. A metric learning model maps the data points into an embedded space, where a KNN performs the classification. The sparse categorical cross-entropy loss is used for the model, which is identical to categorical cross-entropy [47], but takes correct labels as integers instead of binary arrays.

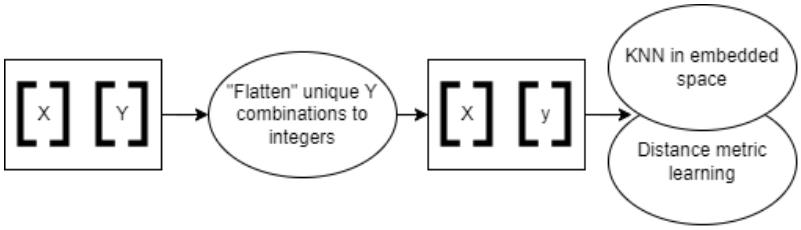


Figure 14. Pipeline for metric learning in this task.

5 EXPERIMENTS

5.1 Data

5.1.1 Raw customer data

For the ERP system to work, it has to keep track of quite much data. However, the data chosen for this work needs to be properly handled as the agreements with the customers state. The input data for the machine learning models does not contain any data which is classified as sensitive in the GDPR, or anything else which could be used to identify single companies or users. The dataset comprises of the following types of fields:

- Numerical values, e.g., how many sales invoices did the company send using the ERP?
- Binary values, e.g., is the company direct customer or a customer of an account office using the ERP?
  - Categories as one-hot-encoded binary values, e.g., which of the ERP pricing models does the company have?

The dataset contains 116 numerical and 168 binary input features. The binary features are mostly whether the company has specific sections of the ERP enabled or disabled. The numerical features contain the number of logins, number of users, turnover, and the number of API requests, as well as a large number of counts of different kinds of transactions made in the ERP. The numeric data which counts something from a span of time, such as number of sent and received invoices, is collected from one month. This way of collecting data has its limitations, because some companies might operate cyclically, and they would get different data depending on the month. On the other hand, if a longer period of time were chosen, it would discriminate newer clients which would not have usage data from the required period for a longer time.

The dataset obtained from the ERP was found to be very biased. All companies which have at least 1 plugin in use were included in the dataset. 3647 out of 5511 total companies have only a single, same plugin, and most plugins are used by a very small group of companies. Also, only a small fraction of possible plugin combination were present. The plugin distribution before bias reduction is presented in Figures 15 and 16.

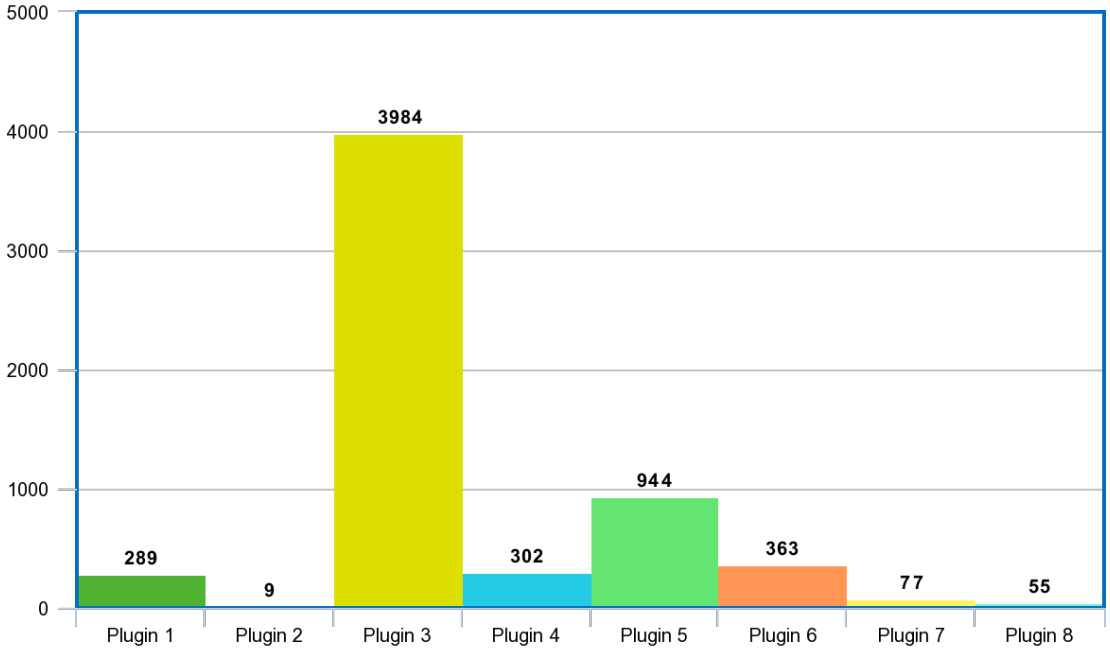


Figure 15. Plugin popularity. Number of companies using a plugin.

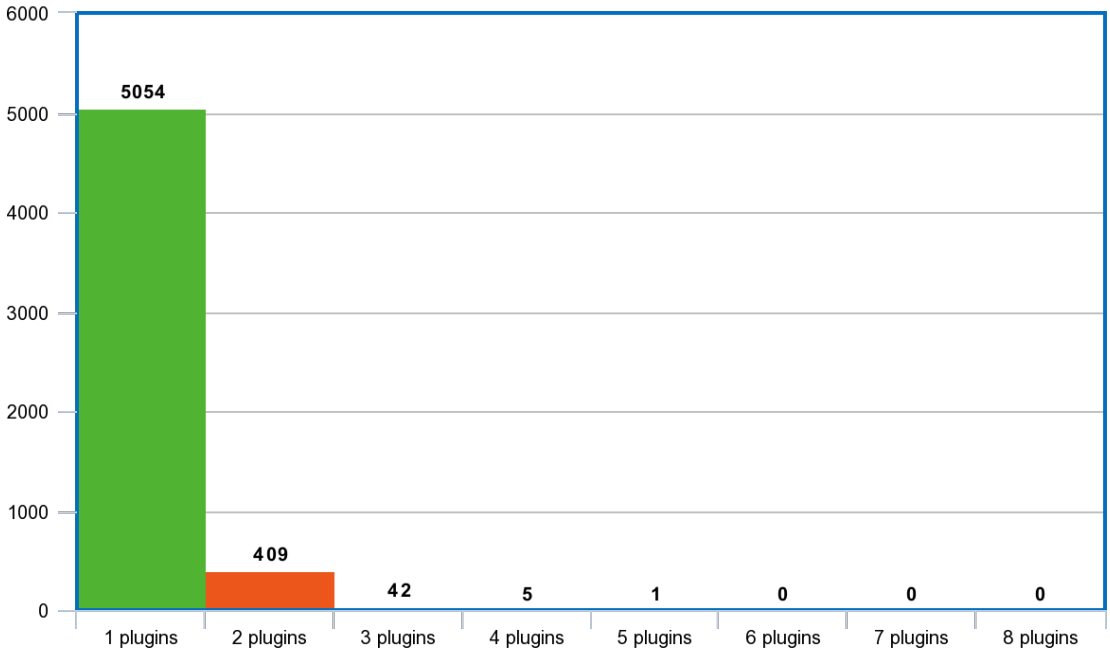


Figure 16. Number of total plugins used by companies.

Partial data and its imputation methods are discussed in Section 3.3, because it is to be expected that data would be missing for two reasons:

1. Incomplete data collecting in the ERP: All events and actions might not be logged in detail.
2. Data not existing in the ERP: The company could send and receive invoices outside the ERP.

However, since the research question is to classify companies based on plugin usage in the ERP, it can be argued that no imputation is necessary in this work. The reason 1 applies similarly for every company, thus not creating missing data fields or rows inside the dataset, but only limiting which features can be chosen. The reason 2 can be ignored: It does not make any difference how many invoices the company might handle outside the ERP, they still will not need the invoice-related plugin in the ERP.

5.1.2 Data preprocessing

The data preprocessing pipeline (Figure 17) in this work is relatively simplistic. It includes normalization, bias reduction and division into training and testing datasets. Due to the small size of the dataset, a separate validation set is not constructed. Instead, the training continues for a fixed number of epochs. The reason behind this decision is that many plugins and combinations have so few data points, that the validation dataset would be very small, and the selection of data points would have bigger impact than desired. After all, the validation dataset should represent similar data than the training set but with different data points, and it is argued that the similarity could not be achieved in this work.

The first preprocessing step is normalization of the data. Each feature is marked as numerical or binary as presented in Section 5.1.1. Binary features are left untouched as they already have values 0 or 1, but numerical features are normalized using the feature-wise zscore-algorithm [48]:

$$x_i = \frac{x_i - mean(x)}{std(x)}, \tag{4}$$

where x contains the values of the feature from all companies and i is the index of company.

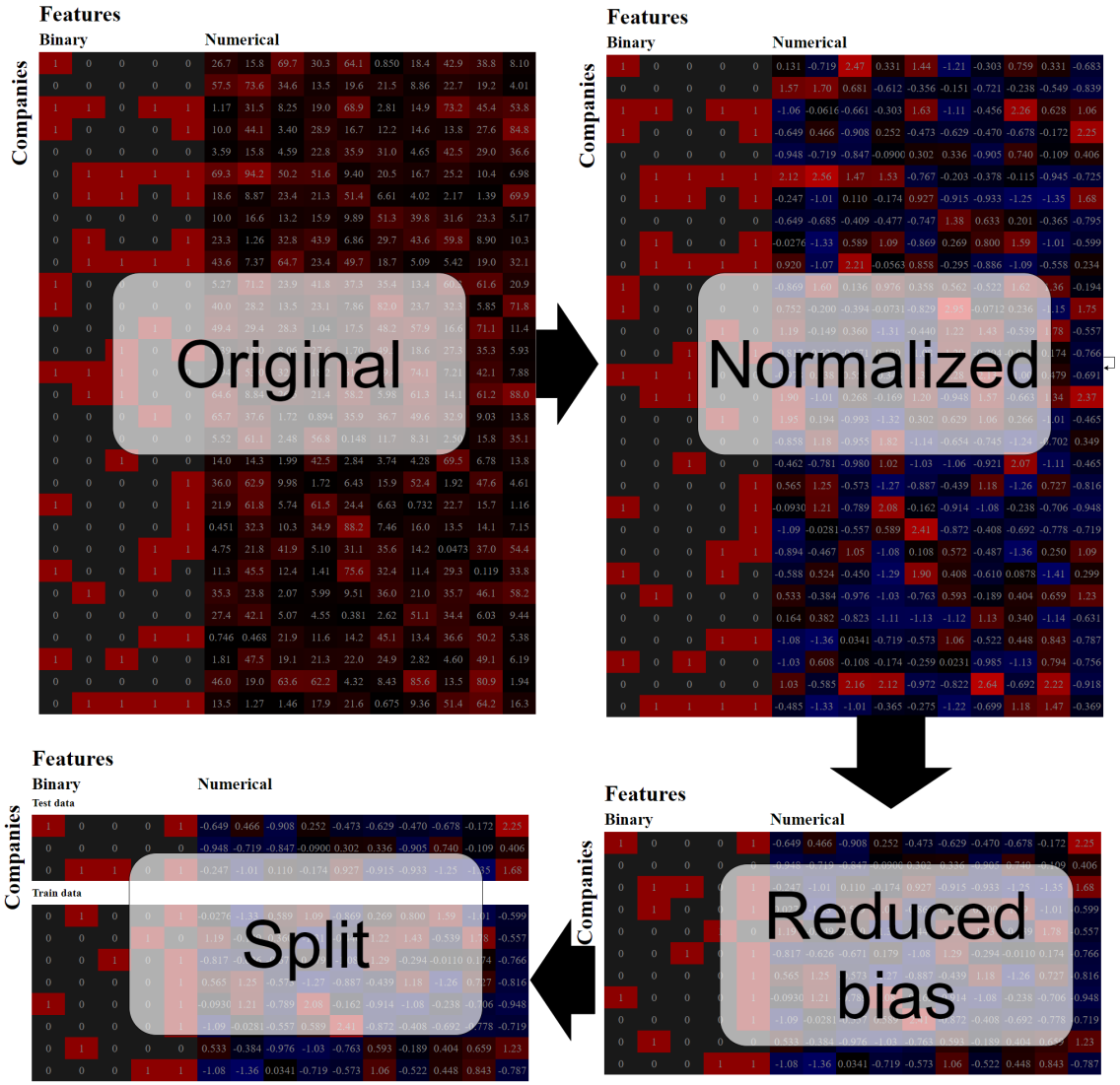


Figure 17. Data preprocessing pipeline.

The second preprocessing step is the reduction of bias. All companies which have more than one plugin selected are kept automatically, because they are a relatively small percentage, and they bring valuable variation in the data. The removal of companies using only a single plugin is done with the following heuristic algorithm for each plugin i separately:

1. Choose a maximum limit, L, for companies using only a single plugin. $L = 150$ in this work, based on the distribution of the data.
2. If the number of companies using only plugin i, $n_i > L$: remove randomly $n_i - L$ companies using only plugin i from the dataset.

The third preprocessing step is the division into training and testing datasets. The division was made by randomly choosing 20 % of data to be the testing set, and the rest 80 % to be the training set. For metric learning, two extra steps were also taken, due to the different nature of the model. First, the data points which were the only ones of their class in the training dataset were also removed, because the model is taught using pairs of data from the same class. Then, the data points were removed from the testing dataset which did not have any matching classes in the training set, because they would be always classified incorrectly.

5.2 Evaluation criteria

5.2.1 Interpretation of model output

Evaluating a multi-label classification model requires more consideration than a single-label model, because what is correct becomes ambiguous. When different data points have different number of correct classes in training set, how should the number of classes for the prediction be determined? The output of the model is an array of class probabilities: $[p_1, p_2, ..., p_n]$, where $0 \leq p_i \leq 1$. In search of the best predictions during training, different constant thresholds were tried as well as choosing top-k class predictions, where $k \in \mathbb{N}$. In [49], top-k is presented as a method to include the correct result in k highest predictions, but in this multi-label classification work it simply means choosing top k predictions as the positive labels. These different methods are used to choose classes from the prediction array for each model, and their results are handled separately from each other.

For every model, the following ways of interpreting the model output were applied, and their respective accuracies calculated. The interpretation functions are visualized in Figure 18.

- Constant thresholds: For each output p_i and threshold t, the output is interpreted as:

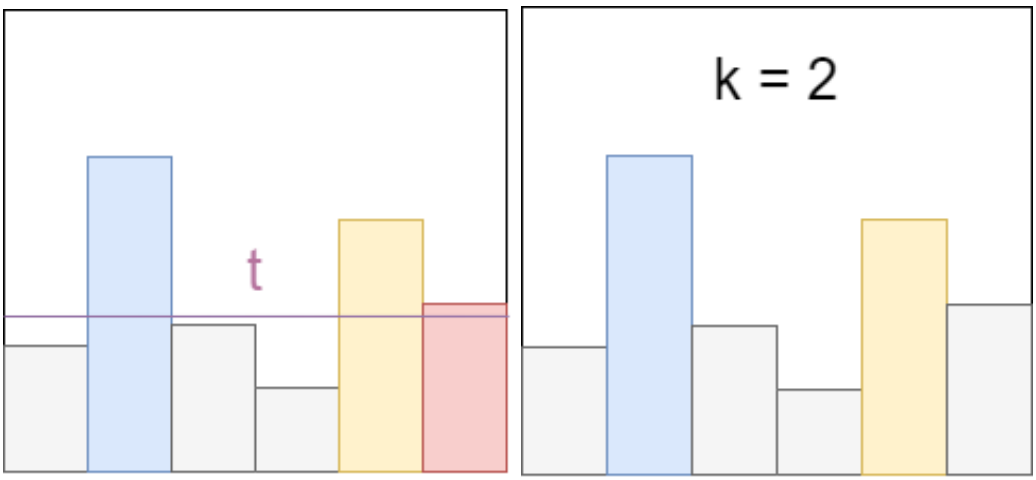
$$p_i = \begin{cases} 1, & \text{if } p_i \geq t \\ 0 & \text{otherwise} \end{cases}$$

  - The following threshold values t were tested: 0.2, 0.3, 0.4, 0.5

- Top-k: For each output p_i and value k, the output is interpreted as:

$$p_i = \begin{cases} 1, & \text{if } p_i \text{ in } k \text{ highest values of } p \\ 0 & \text{otherwise} \end{cases}$$

  - The following values of k were tested: 2, 3, 4, *as many as in the correct output*



(a) (b)

Figure 18. Visualization of threshold and top-2 interpretation functions. Colourful bars represent positive labels, grey bars negative.

It can be argued that it is very important to find the best interpretations for each model. Evaluating the models is relatively cheap in performance, compared to training additional models, and thus it would be unwise not to thoroughly test the capability of the trained models.

5.2.2 Model accuracy calculation

Because the bias reduction cannot achieve equal number of data points for each class, also evaluation should consider the possible class imbalance. If accuracy were measured as correct predictions, very good results would be achieved by predicting only the most popular plugin and nothing else. To take other plugins into account, the accuracy of the model is calculated as the sum of plugin-specific F_1-score values.

To understand a plugin-specific F_1-score, let us define some measures:

- TP_i: True positives by plugin. How many times the plugin was selected by the company and also predicted by the model.
- FP_i: False positives by plugin. How many times the plugin was predicted by the model but not selected by the company.
- FN_i: False negatives by plugin. How many times the plugin was selected by the company but not predicted by the model.
- TN_i: True negatives by plugin. How many times the plugin was neither selected by the company nor predicted by the model.

Using these, plugin-specific precision P_i, recall R_i, and F_1-score F_{1i} can now be calculated as follows:

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{5}$$

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{6}$$

$$F_{1i} = 2\frac{P_i R_i}{P_i + R_i} \tag{7}$$

The total F_1-score F_1 is the sum of plugin-specific ones:

$$F_1 = \sum_{i=1}^{n} F_{1i} = \sum_{i=1}^{n} 2\frac{P_i R_i}{P_i + R_i} \tag{8}$$

This kind of an evaluation criteria gives as much value for each plugin. However many data points belong to the most popular class, it will only have $\frac{1}{n}$ impact on the final accuracy value. This way, the best model should be the one which can capture the characteristics of the classes in the features and get decent precisions and recalls for each

class, instead of learning one or two very well. However, one could argue that this kind of calculation gives too much power to the classes with fewer data points, thus leading to overlearning the features of the rarest points to gain the easiest increase of accuracy value. At this point, one should keep in mind, that many multi-label combinations do not exists, and therefore it cannot be assumed nor expected that there would be an equal number of data points representing the other combinations, either

5.3 Description of experiments

The experimental arrangement combines the data preprocessing, model training, and model evaluation. The program was written in Python (v. 3.10), and the machine learning models were implemented using the libraries Keras (v. 2.9.0) and Tensorflow (v. 2.9.1).

The most important model hyperparameters and evaluation methods were applied using the 2^n factorial experiment design [50], where all combinations are systematically tested. These included:

- Optimizer functions (Follow the regularized leader (FTRL) [51], Stochastic gradient descent (SGD) [52], and Adam [53])
- Neural network number of hidden layers (2, 3, 4, or 7)
- Neural network dropout layer input dropout rate (0, $\frac{1}{4}$, or $\frac{1}{2}$)
- SVM and random forest number of trees (3, 7, 11, or 15)
- Random forest depth of trees (8, 12, 16)

Other hyperparameters, such as learning rate, regularization algorithm and strength, and neural network layer widths, were not tested systematically due to the performance limitations brought by the 2^n factorial design but altered manually with the most promising models. The results were saved so that the best accuracies can be queried and the hyperparameters observed. The default values to begin with were as follows:

- Neural network layer width 4096 neurons.
- Learning rate 0.001.
- Metric learning classifier KNN value $k = 1$.

- No regularization.

5.4 Results

5.4.1 Neural network results

The results of neural network models are shown in Table 2. The neural network seems to be one of the worse two models in this experiment. What the best neural network models seem to have in common is the top-n (*as many as in the correct output*) interpretation. The best neural network with any other interpretation (24th best neural network) has the mean F_1-score 0.402, with difference over 0.05 to the best one. However, the number of hidden layers and the dropout rate seems to vary, suggesting that they have little contribution to the accuracy of the model.

Table 2. 5 best performing neural network models

Model type	Details	Interpretation	Precision	Recall	F_1
Neural network	layers: 3, dropout: $\frac{1}{4}$	top-n	0.447	0.469	0.453
Neural network	layers: 3, dropout: $\frac{1}{2}$	top-n	0.440	0.463	0.448
Neural network	layers: 3, dropout: 0	top-n	0.433	0.466	0.445
Neural network	layers: 4, dropout: $\frac{1}{2}$	top-n	0.434	0.458	0.443
Neural network	layers: 7, dropout: $\frac{1}{2}$	top-n	0.426	0.458	0.440

5.4.2 SVM forest results

The results of the SVM forest models are shown in Table 3. The SVM forest is the best of the three models which actually perform multi-label classification by a notable margin. However, as Table 3 shows, the higher accuracies were only achieved by three model instances, and beyond them the SVM forest models seem to perform on the same level as the weaker models (mean F_1-score between 0.40 and 0.45). The three best instances seem to have in common that they all contain only 3 SVM:s, which suggests that the classes could effectively be separated using 3 decision surfaces. After that, the number of SVM:s seems to vary, however. This distribution is visualized in Figure 19.

SVM forest seems to perform well while interpretation is such that it picks few classes. Thresholds 0.2 and 0.3, as well as top-2, top-3, and top-n are present in the 30 best SVM

Table 3. 5 best performing SVM forest models

Model type	Details	Interpretation	Precision	Recall	F_1
SVM Forest	3 SVM:s	top-n	0.583	0.560	0.558
SVM Forest	3 SVM:s	top-2	0.524	0.608	0.537
SVM Forest	3 SVM:s	threshold: 0.2	0.551	0.592	0.530
SVM Forest	9 SVM:s	top-n	0.518	0.558	0.494
SVM Forest	5 SVM:s	top-n	0.526	0.454	0.454

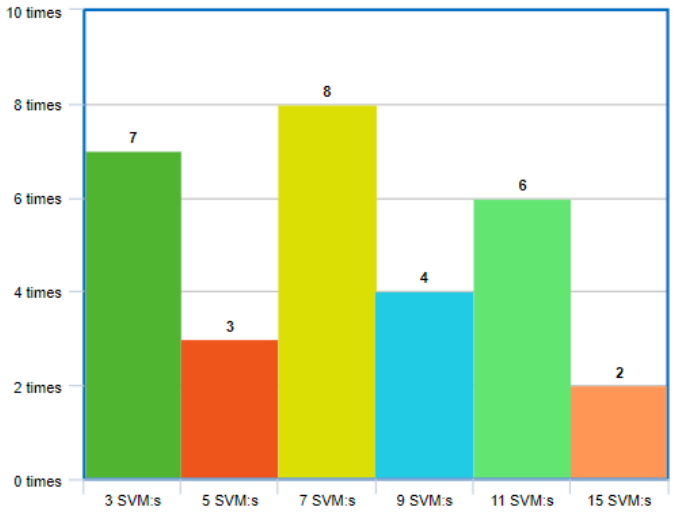


Figure 19. Distribution of number of SVM:s in the top 30 SVM forest models.

forest evaluations. This is to be expected, however, considering that most companies only have 1 plugin - having more positives in predictions leads to decrease of precision.

5.4.3 Random forest results

The results of random forest models are shown in Table 4. The random forest seems to be the worst performing model in this work, though having a very small difference to neural network. The other hyperparameters of the random forest seem to vary, but not before the 21st best model are there 15 trees contained. This, combined with the findings about the SVM forest necessary decision surfaces, might implicate that the random forests used in this work are unnecessarily large. Especially the arbitrary minimum tree depth of 8 should probably be questioned.

Table 4. 5 best performing random forest models

Model type	Details	Interpretation	Precision	Recall	F_1
Random forest	trees: 7, depth: 12	top-n	0.443	0.454	0.443
Random forest	trees: 3, depth: 12	top-n	0.498	0.410	0.406
Random forest	trees: 11, depth: 8	top-n	0.428	0.424	0.405
Random forest	trees: 3, depth: 8	top-n	0.432	0.414	0.402
Random forest	trees: 7, depth: 12	threshold: 0.3	0.465	0.380	0.397

5.4.4 Metric learning results

The results of metric learning models are shown in Table 5. The metric learning model has achieved the second highest accuracy in this work but is not entirely comparable with the other models due to the fact that it performs single-label classification with plugin combinations acting as classes. In evaluation, its output is treated as if it was a multi-label prediction. It is compatible, and the model's uncertainty acts as secondary and tertiary class predictions. However, there is room for improvement, for example, by attempting triplet learning instead of tuplet learning.

What is interesting about the metric learning model is that its best results are achieved with different interpretations than the other models. The most prevalent interpretation is top-4, and the recall values seem to be often higher than precision values. The best metric learning model (Table 5) is a great example of such. The model predicts too many classes, but the correct classes are so often among them, that the high recall keeps the mean F_1-score decent.

It is noticeable how quickly the great F_1-scores of metric learning models fall. The difference between the 1st and the 5th models' accuracies is remarkable. One should keep in mind, however, that due to the 2^n factorial design, for each metric learning model trained, there are 12 random forest models, 4 SVM models, and 12 neural network models trained. Assuming that training with the same parameters leads to roughly similar accuracies, it is a very different thing to display 5 best metric learning models than 5 best neural network models.

5.4.5 Overall results

The overall best model evaluation results are shown in Table 6. The best performing model instances seem to be either SVM forests or metric learning models. The best

Table 5. 5 best performing metric learning models

Model type	Details	Interpretation	Precision	Recall	F_1
Metric learning		top-4	0.493	0.646	0.556
Metric learning		top-3	0.511	0.519	0.512
Metric learning		threshold: 0.2	0.506	0.523	0.509
Metric learning		top-n	0.525	0.438	0.473
Metric learning		threshold: 0.3	0.507	0.408	0.443

neural network is the 10th best overall model, and the best random forest is the 15th. The highest accuracies per model type are:

- SVM forest: 0.558
- Metric learning: 0.556
- Neural network: 0.453
- Random forest: 0.443

Based on these findings, it could be argued that the SVM forest and metric learning are better performing models for this work. Their best accuracies are very close to each other, as are those of neural network's and random forest's, but the difference between the two groups is roughly 0.1.

Table 6. 5 best performing models

Model type	Details	Interpretation	Precision	Recall	F_1
SVM Forest	3 SVM:s	top-n	0.583	0.560	0.558
Metric learning		top-4	0.493	0.646	0.556
SVM Forest	3 SVM:s	top-2	0.524	0.608	0.537
SVM Forest	3 SVM:s	threshold: 0.2	0.551	0.592	0.530
Metric learning		top-3	0.511	0.519	0.512

6 DISCUSSION

6.1 Current study

The objective of the thesis was to review customer data analysis and classification with machine learning, create a suitable model for multi-label classification, train it with a prepared dataset, and evaluate it. Four different methods were used for the task; neural network, SVM and random forest were altered to be able to perform multi-label classification, and a metric learning classifier was trained for single-label classification by processing data. The models were trained, and different evaluation methods were applied to be able to measure the performance of the models.

The highest F_1-score achieved was 0.558 with a SVM forest model. Given the selected accuracy calculation and considering the application as a recommender model whose errors do not cause damage, the results can be considered promising. Neural network and random forest models had worse performance but refining them might produce better results.

The highest accuracies (mean F_1 = 0.558) are not nearly as high as single-label classifiers produce, especially in binary classification tasks. However, in the context of the current work and evaluation method, the results are arguably promising. A business-critical system should not be implemented using the current models, but for an application with less consequences on incorrect predictions, such as a recommendation tool, the best models might be good enough in their current state.

The reason why as small value as 0.558 could be considered promising is the calculation of the F_1-score. As can be seen from Equation 8, if either precision or recall is low, so is the F_1-score. While the models are expected to learn to predict those classes well which have many training data points, there are many classes which do not have, as Figure 15 shows. The case is most likely that those classes are difficult for the models, and as they have few data points, every false negative prediction lowers recall significantly. On the other hand, if the model learns to predict the rare class often, precision decreases rapidly. And if there are 0 true positives, both precision and recall are 0, and so is the F_1-score. This way, the predictions of the rarest plugins have most impact on which models reach the highest accuracies and expecting as high accuracy values as with single-label classification tasks would be unrealistic.

The evaluation of the models could be further developed. The plugin-specific way of calculating accuracy ensures that every plugin is taken into account, but this also introduces a problem. Some plugins have very few datapoints, which raises a question, whether it impacts too much if those few datapoints are predicted correctly or not.

The previous question could be approached by contemplating the actual problem that the models are trying to solve. If we should ask "which plugin would the company most likely want to use?", the answer might be every time the most popular plugin (Plugin 3 in Figure 16). However, if we asked, "for which plugin does the company suit best?", the bias would be meaningless and every plugin would be seen as equal. The current accuracy calculation using plugin-specific F_1-scores is targeted for the latter approach, but it should not be accepted without questioning.

The comparison between metric learning and multi-label classifiers is not fair in this work. Neural network, support vector machine, and random forest models are adapted to perform multi-label classification, but the metric learning model performs ordinary single-label classification, where all combinations of multiple labels are converted into unique labels. While the output of the metric learning model is compatible with the multi-label model evaluation calculation, the model never learns to output secondary or tertiary values on purpose, instead it outputs them as uncertainty. The goal of this work is to create a working implementation, but to be able to actually compare the metric learning with the other models, a similar implementation should be created for both.

The model output interpretation option *as many as in the correct output* is not very realistic. It was used to be able to see how the models learn, because with hard-coded top-k or threshold values there will always be wrong number of output classes. However, in real life application there will not be the information of how many classes should be chosen, and thus it would make sense to remove the *as many as in the correct output* entirely from the evaluation process.

The usability if the current metric learning model should be taken with caution. Even though metric learning achieves the accuracy of 0.556, the prominent top-4 interpretation and precision lower than recall undermine its performance as a recommender. A good recommender should achieve good results with top-2 interpretation, because most likely only 1 or 2 plugins would be recommended in the application. Even though metric learning successfully finds the correct classes in top-4, it might not be suitable for the recommending task.

Also, the ground truth of the classification task should be considered. The fact that com-

panies have selected some plugins does not guarantee that those plugins are the best fit for the company, yet this data is used to train and evaluate the model. For example, if a new plugin was recently added, very few, if any, companies would have it at the moment and the model would not recommend it for most companies, regardless of if it would actually fit their needs.

6.2 Future work

Many problems have been solved one way in this thesis but exploring alternative solutions might be useful as future work. For example, the bias reduction method is not very sophisticated and uses an arbitrary value which was decided by looking at the data. Being able to use a larger share of the original data without exposing the model to remarkable class-imbalance could improve the learning of the generalized features in the data.

A k-fold cross validation might be useful with such a small dataset as well. When there are only a few companies with specific plugins, the division of data points between datasets becomes important for those classes especially. In this thesis, there is no confirmation logic that each class would have more points in the training dataset than the testing one. In the worst case, this could lead to all data points being in the same dataset, thus causing the model not to learn them or getting a 0 in precision because there was no chance to make any true positive predictions.

Different model hyperparameters could be further experimented. The metric learning model could use triplet loss, and especially the neural network was left with many untested options and parameter combinations. Other machine learning models could be tried as well. And if there were more data, a validation dataset should be used to prevent the model from overlearning the training dataset by observing validation accuracy.

The neural network is the model type which has clearly the most hyperparameters and options. A limited number of non-systematic alterations were made to hyperparameters like optimizer function, learning rate, regularization, layer width, initialization, loss function, and activation function, but many parameter combinations were not tested. Of all the models present in the work, neural network was left with the most unexplored tuning, which might have improved its performance, given the time and resources for systematic testing.

For the metric learning model, open set classification [54] with a similarity threshold

could also be tested. This would remove the problem that testing data might contain labels which do not exist in the training data. However, if the result that the open set classifier would give below the similarity threshold was 'some other class', it would not be very useful as a plugin recommendation.

It should not be taken for granted that there should be only one model. A One-Against-All model for each plugin could be worth testing in the future. The reason for the current solution lies in the scarcity of the data. Some models might only have a few positive datapoints, and the risk of overlearning would be remarkable. However, having one model per plugin might enable using different models for different plugins based on how they perform.

7 CONCLUSION

Many information systems collect data about their users to be able to perform in their primary task. The data can be further utilized by machine learning methods, which might be able to learn such patterns from the data, that could be otherwise challenging to discover. This way machine learning can provide additional value for the system using the existing user data.

In this thesis, customer companies in an ERP system were classified using machine learning methods. A literature review of customer data analysis and machine learning classification was provided, and a method for solving the research problem was proposed. Four different machine learning methods were tailored to be able to perform multi-label classification. Data was preprocessed and the models were trained and evaluated using class-specific F_1-score.

Different interpretation and evaluation methods were presented and applied. The accuracies achieved with the models were promising, highest mean F_1-score being 0.558. The SVM model and metric learning performed better than the neural network and the random forest. The results were analyzed and discussed, and improvements and future work were presented.

REFERENCES

[1] Hannu Saarijärvi, Heikki Karjaluoto, and Hannu Kuusela. Customer relationship management: the evolving role of customer data. *Marketing Intelligence & Planning*, 31, 2013.

[2] Vera L Miguéis, Ana S Camanho, and João Falcão e Cunha. Customer data mining for lifestyle segmentation. *Expert Systems with Applications*, 39(10):9359–9366, 2012.

[3] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[4] T. K. Das. A customer classification prediction model based on machine learning techniques. In *International Conference on Applied and Theoretical Computing and Communication Technology*, pages 321–326, 2015.

[5] Qianling Chen, Min Zhang, and Xiande Zhao. Analysing customer behaviour in mobile app usage. *Industrial Management & Data Systems*, 117, 2017.

[6] Arthur Middleton Hughes. *The complete database marketer: Second-generation strategies and techniques for tapping the power of your customer database*. Irwin Professional Pub., 2nd edition, 1996.

[7] Ammar Mars and Mohamed Salah Gouider. Big data analysis to features opinions extraction of customer. *Procedia Computer Science*, 112:906–916, 2017.

[8] Zhiqiang Yang, Sheng Zhong, and Rebecca N Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SIAM International Conference on Data Mining*, pages 92–102, 2005.

[9] Tzu-Liang Bill Tseng and Chun-Che Huang. Rough set-based approach to feature selection in customer relationship management. *Omega*, 35(4):365–383, 2007.

[10] Jin Xiao, Hanwen Cao, Xiaoyi Jiang, Xin Gu, and Ling Xie. Gmdh-based semi-supervised feature selection for customer classification. *Knowledge-Based Systems*, 132:236–248, 2017.

[11] Bingquan Huang, Brian Buckley, and T-M Kechadi. Multi-objective feature selection by using nsga-ii for customer churn prediction in telecommunications. *Expert Systems with Applications*, 37(5):3638–3646, 2010.

[12] Pete Rotella and Sunita Chulani. Analysis of customer satisfaction survey data. In *IEEE Working Conference on Mining Software Repositories*, pages 88–97, 2012.

[13] Benjamin Marlin. *Missing data problems in machine learning*. PhD dissertation, University of Toronto, 2008.

[14] Sahar F Sabbeh. Machine-learning techniques for customer retention: A comparative study. *International Journal of Advanced Computer Science and Applications*, 9(2), 2018.

[15] Anoop D Shah, Jonathan W Bartlett, James Carpenter, Owen Nicholas, and Harry Hemingway. Comparison of random forest and parametric imputation models for imputing missing data using mice: a caliber study. *American Journal of Epidemiology*, 179(6):764–774, 2014.

[16] Munevver Mine Subasi, Ersoy Subasi, Martin Anthony, and Peter L Hammer. A new imputation method for incomplete binary data. *Discrete Applied Mathematics*, 159(10):1040–1047, 2011.

[17] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[18] Miron B Kursa, Aleksander Jankowski, and Witold R Rudnicki. Boruta–a system for feature selection. *Fundamenta Informaticae*, 101(4):271–285, 2010.

[19] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering*, 160(1):3–24, 2007.

[20] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[21] Jason Brownlee. What is the difference between a batch and an epoch in a neural network. *Machine Learning Mastery*, 20, 2018.

[22] F.Y. Osisanwo, J.E.T. Akinsola, O. Awodele, J.O. Hinmikaiye, O. Olakanmi, and J. Akinjobi. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology*, 48(3):128–138, 2017.

[23] Aized Amin Soofi and Arshad Awan. Classification techniques in machine learning: applications and issues. *Journal of Basic and Applied Sciences*, 13:459–465, 2017.

[24] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, 2006.

[25] Chhaya Khanzode and Ravindra Sarode. Advantages and disadvantages of artificial intelligence and machine learning: A literature review. *International Journal of Library & Information Science*, 9(1):3, 2020.

[26] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[27] Vaibhav Kant Singh. Proposing solution to xor problem using minimum configuration mlp. *Procedia Computer Science*, 85:263–270, 2016.

[28] Thomas Bayes. An essay toward solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1764.

[29] Serena H Chen and Carmel A Pollino. Good practice in bayesian network modelling. *Environmental Modelling & Software*, 37:134–145, 2012.

[30] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[31] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[32] Eric Kim. Everything you wanted to know about the kernel trick. *URl: https://web.archive.org/web/20221028150810/https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html*, 2013. Accessed 30 October 2022.

[33] Abhisek Ukil. Support vector machine. In *Intelligent Systems and Signal Processing in Power Engineering*, pages 161–226. 2007.

[34] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

[35] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.

[36] Miguel A Carreira-Perpinán. A review of dimension reduction techniques. *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09*, 9:1–69, 1997.

[37] Nishtha Jatana, Sahil Puri, Mehak Ahuja, Ishita Kathuria, and Dishant Gosain. A survey and comparison of relational and non-relational database. *International Journal of Engineering Research & Technology*, 1(6):1–5, 2012.

[38] Roderick Little and Donald Rubin. *Statistical analysis with missing data*. John Wiley & Sons, Inc., 1987.

[39] Bin Jiang, Xi Fang, Yang Liu, Xingzhu Wang, and Jie Liu. Spectral feature extraction using partial and general method. *Advances in Astronomy*, 2021.

[40] Anil Jadhav, Dhanya Pramod, and Krishnan Ramanathan. Comparison of performance of data imputation methods for numeric dataset. *Applied Artificial Intelligence*, 33(10):913–933, 2019.

[41] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[42] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[43] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[44] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards data science*, 6(12):310–316, 2017.

[45] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[46] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State University*, 2(2):4, 2006.

[47] Kevin Koidl. Loss functions in classification tasks. *Presentation, School of Computer Science and Statistic Trinity College, Dublin*, 2013.

[48] S Patro and Kishore Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[49] Felix Petersen, Hilde Kuehne, Christian Borgelt, and Oliver Deussen. Differentiable top-k classification learning. In *International Conference on Machine Learning*, pages 17656–17668, 2022.

[50] Charles Robert Hicks. *Fundamental concepts in the design of experiments*. Holt, Rinehart and Winston, 1964.

[51] Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *International Conference on Artificial Intelligence and Statistics*, pages 525–533, 2011.

[52] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[53] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[54] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3614–3631, 2020.