Classification of Purchase Invoices to Analytic Accounts with Machine Learning

Samuel Johansson

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Järvenpää 30.12.2022

Supervisor

Prof. Alex Jung

Advisor

M.Sc. (Tech.) Petri Heino

Aalto University
School of Electrical
Engineering

Copyright © 2022 Samuel Johansson

Aalto University
School of Electrical Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Samuel Johansson

Title Classification of Purchase Invoices to Analytic Accounts with Machine Learning

Degree programme Automation and Electrical Engineering

Major Control, Robotics and Autonomous Systems **Code of major** ELEC3025

Supervisor Prof. Alex Jung

Advisor M.Sc. (Tech.) Petri Heino

Date 30.12.2022 **Number of pages** 47 **Language** English

Abstract

The process of analytic account selection for invoices is time-consuming and expensive. These problems can be diminished with an automated system for selecting the analytic accounts on the invoices.

The aim of this thesis is to implement a software system for classifying purchase invoice accounting lines into analytic account in Odoo ERP system. The proposed software uses machine learning methodology as analytic account selection is essentially a multi-class text classification problem. The proposed accuracy goal set is 90% as it is an indicated threshold in a related invoice account selection problem.

The thesis investigates the current text classification algorithms and compares them empirically with each other. The selected algorithms are Multinomial Naive Bayes, Logistic Regression, Linear and Non-linear Support Vector Machines (SVM), k-Nearest Neighbor, Decision Tree, Random Forest, Perceptron and Multilayer Perceptrons (MLP). Additionally a primitive Odoo module is created to use this software system in an Odoo server for automatic analytic account selection. The features of the data contain textual descriptions of an invoice line which require the text data preprocessing before transferred into the models.

The results indicate the best performing models are Logistic Regression, linear SVM and MLP where the linear SVM is the most effective overall. This model is optimized even further where a test instance reached an accuracy of 88% at best. However empirically the macro averages (precision, recall and F1-score) for an average instance is around 0.80.

The thesis presents a functional Odoo module where the training and prediction of analytic accounts can be done within an Odoo server. The Odoo user interface allows to train a model, show the scores of different classes and predict analytic accounts for lines.

Keywords Purchase Invoice, Analytic Account, Account Posting, Software Automation, Machine Learning, Odoo, ERP

Acknowlegements

I would like to thank Professor Alex Jung in his inspirational way of thinking about Machine Learning. His lectures is one of the reason I got excited about ML. Thank you.

I'd also want to thank my colleagues at SprintIT who helped me to complete this thesis. I, Elmeri Niemelä and Miika Rouvinen were doing theses at the same time and helped each other out in the process which was great. Konsta Aavaranta was very helpful with the accounting side of things and I thank him for that. Especially I'd like to thank my advisor Petri Heino for his excellent guidance of the Master's Thesis process.

Järvenpää, 27.12.2022

N. J. Samuel Johansson

Contents

Symbols and abbreviations

Symbols and Operators

$\mathbf{x}$	Input (feature) data vector
$\mathbf{y}$	Output (label) data vector
h	Hypothesis
$\mathcal{X}$	Feature space
$\mathcal{Y}$	Label space
$\mathcal{H}$	Hypothesis space, the model
$\mathbb{E}$	Expected value
L	Loss function
$\hat{L}$	Estimated value (of loss function)
$\|\mathcal{Y}\|$	Size of the label space
$\mathbf{I}$	Indicator function
$p()$	Probability (of the input)
$p(x\|y)$	Conditional probability
$\mathbf{w}^T$	Transpose (of weight vector)

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
BOW	Bag-of-Words
CS	Computer Science
ERM	Empirical Risk Minimization
ERP	Enterprise Resource Planning
ML	Machine Learning
MLP	Multilayer Perceptron
PCA	Principal Component Analysis
RL	Reinforcement Learning
SL	Supervised Learning
SRM	Structural Risk Minimization
SVM	Support Vector Machine
VSM	Vector Space Model
XML	Extensible Markup Language

1 Introduction

The acquisition of goods is a more complex process than the initial simple idea of a trade transaction. Not only must companies agree on the particular goods to be bought and at what price, but they must also account for the transactions themselves. In a sense, accounting is easy because a company only needs to record transactions with the necessary details. However, the involvement of humans in the process makes the accounting error-prone and suboptimal, since individuals do not work or communicate in a rigorously standardized way [1]. These problems and the advent of computer systems have led to the automation of the invoicing process.

Accounting is a vast field that typically involves the process of purchasing as an important subfield of accounting. This subfield is further divided into many sub-processes and sub-tasks. One of these tasks is the selection of analytic accounts for purchase invoices. In this task, the accountant selects an account from a list of accounts based on company-specific accounting rules. However, invoice processing can be one of the most time-consuming tasks in the financial department of a company [2]. In the best scenario, data such as a reference or description of the product is available from the invoice itself, which can be used to identify the account [3].

One approach for automating and speeding up the account selection process would be to use the Machine Learning (ML) methodology. ML is a collection of computer algorithms that can be used to predict new values based on a fitted model trained on data and loss functions. ML methodology enables computers to perform tasks previously limited to only humans, thereby reducing costs and time. One of the main tasks in ML is to classify textual data, and various methods have been developed for text classification [4, 5]. Recently, attempts have been made to automate the account selection task as a text classification task in ML using methods such as Support Vector Machines (SVM), Multilayer Perceptrons (MLP), and Decision Trees [6]. Although other methods (e.g., Random Forests or Naive Bayes) have been used to classify invoices [7, 8], these studies have focused only on classifying invoices themselves as a data point and have not attempted to use accounting lines (e.g., invoice rows) as data points in purchase invoices.

Therefore, the aim of this thesis is to design and implement a software system for classifying purchase invoice accounting lines into analytic accounts based on ML. For this purpose, the thesis will first identify a suitable ML method for meeting requirements such as the scalability of the model, the number of target classes (i.e., labels) and the ability to operate in the Odoo Enterprise Resource Management (ERP) system. Since Odoo already uses machine learning, it will be used as a benchmark in an attempt to outperform Odoo's claimed accuracy of 90% for account prediction. The thesis will eventually present a proof-of-concept software implementation.

The remainder of this thesis is divided into four chapters. Chapter 2 reviews the literature on machine learning methods. Chapter 3 defines the problem more precisely as a ML question and designs the classification software and implements the proposed software. Chapter 4 evaluates the developed software system and verifies that this system complies in terms of the problem definitions and specifications. Chapter 5 summarizes the main work and recommends possible future work.

2 Background

The main purpose of this chapter is to present the relevant technical literature and the mathematical basis used in this thesis. The collected material in this chapter will be used in the next chapter to justify design decisions. In Chapter 4 the system will be tested and verified so that it adheres to the principles, definitions and restrictions stated in this chapter. Additionally, the accounting problem is defined more precisely in the next section, with the relevant restrictions stated. In the beginning of Chapter 3 the problem will be defined in machine learning terms after the required foundations in this chapter have been stated.

2.1 Processing of Purchase Invoices

One of the most resource consuming tasks in company's financial departments is the processing of purchase invoices [2]. The reason for this is that the processing of purchase invoices is not as simple as it seems. Generally there are multiple different problems including duplicate payments and frauds, increasing costs with suboptimal invoice processing pipelines and errors in accounting [1]. The most relevant problem is the excessive time consumed to select an analytic account for a purchase invoice line. The purpose of this thesis is to make that sub-process more time efficient. This section gives a brief introduction to processing of purchase invoice as a whole and concentrates on the analytic account selection. The section will introduce the idealized process and disregard the many of the complexities in the general processing pipeline.

A recurrent activity of a company is the process of buying utilities from vendors. This involves the initial purchase orders, receiving the products and the associated purchase invoices. From the perspective of the responsible financial department the process begins from receiving the purchase invoice. This invoice needs to be posted and approved in order to be payed. If there are no problems with the invoice it should be payed. This payment is then reconciled and matched with relevant purchase order. Finally the data related to this process should be archived for later use [2].

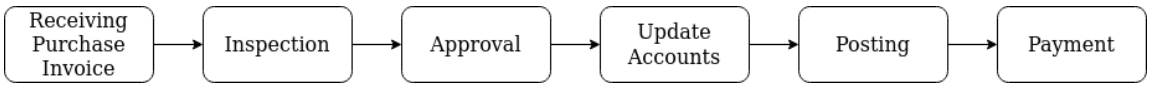

Figure 1: The generalized purchase invoice processing procedure.

The relevant issue is the bookkeeping of transactions in accounts. A bookkeeper needs to post the financial transaction information to the right accounting journal. An accounting journal is one of many accounts in a chart of accounts. The right account needs to be chosen from this chart of accounts by the bookkeeper. The recording of transactions can be mandated by law, although the specific activity of analytic account booking is voluntary as it is used as an analysis tool. There is significant flexibility in the amount and type of accounts, and a company may choose what kind of accounts to use. Type of account include receivables, liabilities, payables, assets and expenses, among others [9, 10]. If the company uses only one

accounts payable for all the purchases then choosing of the account is trivial. However in some cases a company might use multiple analytical accounts. In this case a book keeper needs to choose an account from a list accounts. The choosing of an account from a list of accounts is not trivial. For example, analytical accounts may be used as a more detailed bookkeeping tool, such as the monitoring of cost-center or project based expenses. The bookkeeper needs to infer which account is the most relevant for that specific invoice line.

An invoice is a document sent to the customer which identifies the agreed transactions and the required payments the customer owes to the issuer of the invoice. An invoice should include the contact information of the issuer, the recipient address, due dates, the amounts of money and the products or services which are invoiced. Invoices should contain all the necessary information to make the transaction and the contact information of the issuer for possible reclamation or other matters of discussion. When an invoice is sent by the issuer with the necessary information, the customer should process it and associate that invoice with additional information, such as the internal accounts or cost-centers of that purchase [3].

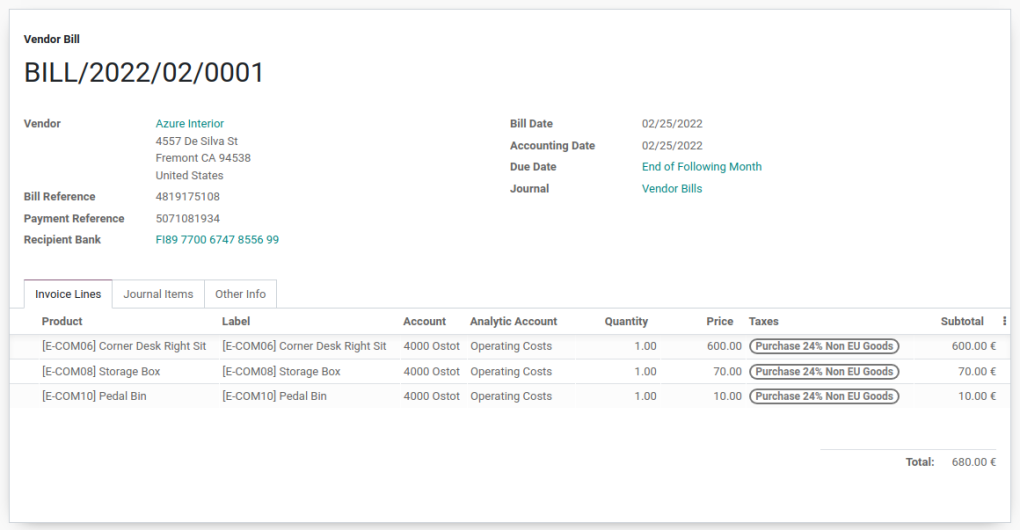

Figure 2: An example of a purchase invoice, vendor bill, in Odoo ERP.

Many different products or service may be invoiced at the same time in an invoice. An invoice line, a row of a product, refers to one specific product in a table of many rows. An invoice line in figure 2 would be, for example, the 'Storage Box' with a quantity of one and a subtotal of 70 euros. Every line should have a description of the product or service and the cost of that purchase. The accounts and analytic accounts are the buyers associated information relating to the accounting needs of the internal company. The account in figure 2 for 'Storage Box' line would be '4000 Ostot' (which in English is 'Purchases') and the analytic account is, in this case, 'Operating Costs'.

To reiterate, a bookkeeper-in-charge may choose the chart of accounts and the way which transactions are accounted for [10]. This means that the choosing of the right account for an invoice line is dependent on the practices in that company.

The classification of that invoice line depends on the description of what have been bought.

Invoices can be sent in paper, or electronic format as an e-invoice. Usually e-invoice standards use XML-based protocols to transmit data, such as Finvoice. When the data is readily available in electronic format the usability and possibility to automate the invoicing process increases. Benefits in e-invoices include the decreasing costs, manual error reductions and customer service increases [11]. The availability of data already in digital format makes it easier to train a software system, because these systems usually need the data in electronic format to be processed.

2.2 Machine Learning

There are multiple attempts to define machine learning (ML). Machine learning can be seen as computational methods applied to some learning task to make sufficiently accurate predictions based on learning from experience [12]. ML can also be seen as answer to "how to get computers to program themselves" with some experience and initial structure [13]. The emphasis can been seen in the name of the methodology itself, 'a machine' which 'learns' [14]. Regardless of the angles of examination the consensus seems to be that machine learning is indeed some collection of computer algorithms which use data to learn, teach some models to predict some wanted value from data and use some loss function to evaluate the performance [15].

The differences of viewpoints come from the intersection of many related fields. Machine learning is a subfield of computer science (CS) and naturally is confined to the restrictions of theoretical CS. The understanding of memory and computational complexities are relevant in understanding the feasibility of ML models [15]. Machine learning can be seen as part of artificial intelligence (AI) or in contrast seen as new extension to traditional AI methods. Nevertheless, the capability to use experience as a tool to learn patterns in complex (sensory) data is vital for intelligence [13, 14]. ML is closely related to statistics as both try to answer the question of "what can be inferred from data" including the reliability and modeling assumptions. Many ML methods also use statistical and probabilistic models as part of their structure [13, 15]. Other related fields include, but is not limited to, linear algebra, optimization, data mining, pattern recognition information theory and databases. ML can be used in multiple different applications, such as text classification, natural language processing, computer vision applications, fraud detection, recommendation systems and in autonomous robotic vehicles [12].

ML can be divided into several subtypes. Supervised learning (SL) is concerned with labeled data. This is a type of data where some expert, usually human, has mapped some input data $\mathbf{x}$ to some output data $\mathbf{x}$ using his or her expertise. The input data $\mathbf{x}$ has in a sense some intrinsic features and information which can be learned to produce the output labels $\mathbf{y}$. The attempt is to generalize, or learn, this mapping to unseen data. Typical tasks in SL include regression and classification [12, 13, 15]. The requirement for supervised learning is that some labeled data exists.

If no labeled data exists, then unsupervised learning can be considered. In this kind of data no expert produced mapping exists. The point of unsupervised learning

is to learn the 'intrinsic' structure of the data, with methods like clustering or feature learning. The reason for unlabeled data might be that it is costly to produce labeled data. Nevertheless, it is important to get value also from data which is not labeled [13, 15]. Some data might also have a mixture of labeled and unlabeled data. Semi-supervised learning tries to learn from this mixture of data and make predictions for other unseen points. Usually in this kind of situation the labeled data is scarce and unlabeled data abundant [12, 13].

In general, a third type of learning scheme is identified. In reinforcement learning (RL) an active agent in some environment tries to learn the most beneficial actions for long-term success. RL is different from SL in the sense that there exists an agent which interacts with an environment. In this scheme there is activity from some agent and some response from the environment. These actions and responses form a feedback loop where actions may be rewarded or penalized. The agent tries to learn which actions are the most beneficial for long term success [16]. In supervised learning there is no agent and the system is taught by using examples input-output mappings and in RL there is a trial-and-error feedback loop by an active learner.

The aim of machine learning methodology is to generalize. It is not only to teach a model to fit the existing data, but also for that model to predict new unseen data. Machine learning methodology, in essence, produce predictors [12]. One difference between data analysis and machine learning is that data analysis tries to explain the existing data, but machine learning attempts to predict values for future data (based on the existing data).

2.2.1 Data, Model and Loss

There are identified three main components of machine learning: data, models and loss. These components have interplay and a process cycle in which the ML methodology works. [15] Data is the observation of some real phenomenon with some error associated to the observation. To teach a ML model to perform better, some **loss** function is used to measure the difference between manifestations of some phenomenon, i.e. **data**, and the current prediction of the **model**. Thus **the loss** function are used to guide **the model** to more accurate direction with the use of **data** [12, 15, 17].

Data is a collection of data points. These data points can be for example images, text documents, signals or random variables. A data point is a single distinct unit of information. What constitutes a single data point is a design choice in ML, for example a data point might be several videos, a single frame in a video or a group of pixels in a frame. Data points in ML is an abstraction of units to be processed for the purpose of that ML application. A specific application might be to classify videos, thus a data point is a video. Likewise, an application might be to identify animals, people or structures in a picture, thus a data point could be a frame in a video. In this sense data point is a choice, because data points vary from the goals and use cases of the application.

Data points have features which are structures within a data point, usually continuous or discrete. The features are also a design choice and they define how the

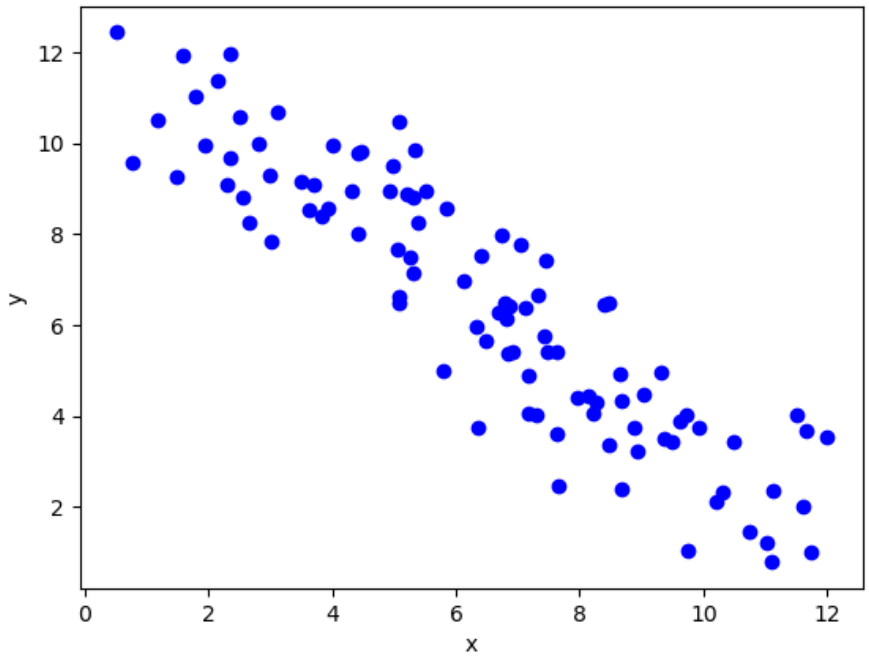

Figure 3: Example of data show on a scatter plot. One dot can be seen as a data point. Horizontal 'x' values can be defined as the features and vertical 'y' values as the labels.

information is structured within that data point [18]. Continuous features can for example be height of a person and a discrete feature might be the gender of that person. These features can be seen as some information source for that specific data point.

Data points also have labels associated with them. Labels are quite similar to features, but are distinct in that they are unit of interest, the target values. Again, the labels for a data point is a design choice and is usually determined by the goal of that application [15]. A label for a data point might be a classification of a persons health, to healthy or non-healthy. The term 'label' is usually confined to supervised learning with labeled data. In unsupervised learning there is also a goal, output or target. This target is usually finding patterns in unlabeled data [19].

In ML, models are a selection of data structures with the attempt to encapsulate the underlying phenomenon of the data. After successfully trained, a model should compress information into a certain hypothesis from the seemingly complex manifestations of that phenomenon. A hypothesis is just one of many, if not an infinite set of hypotheses. A set of hypotheses is called the hypothesis space. Different models have varying hypothesis spaces depending on the construct. Models thus present their set of hypotheses, and with the use of a loss function one hypothesis is manifested as the parameters of that model. [12, 15]

There are different models with different hypothesis spaces. Most commonly

models are divided into linear and non-linear models. In linear models there is a linear mapping between some input **x** to some output **y**. For example in linear regression the attempt is to fit a linear function to some data [17]. Non-linear models include all the mappings which are not linear. For example in Artificial Neural Networks (ANN) can be a complex set of linear mappings which are combined to produce a non-linear mapping in the system as a whole.

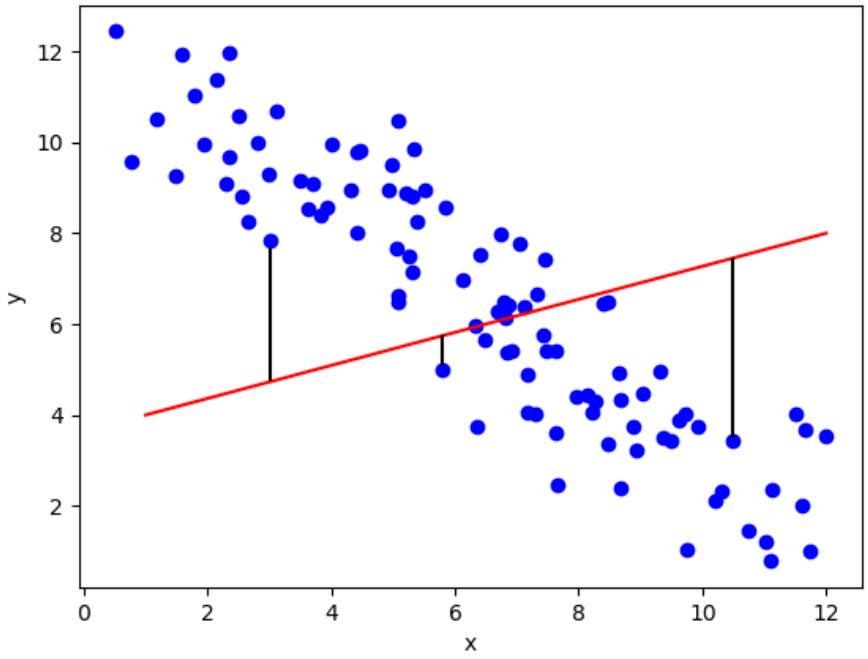

Figure 4: Example of a linear model, plotted as red. The black lines indicate some loss between the model and some data points.

The loss function is used to calculate the difference between the current mapping of the model and the data points. This difference, the loss or cost, can be used to train the model to better fit the data by minimizing the loss [15]. These functions take values of data points and compares them with the model and calculate the cost for those comparisons. Loss functions include the 0/1 -loss, hinge loss or logistic loss. In 0/1 -loss the cost is either 0 if under some threshold value and 1 if over that value. These losses 'punish' the model if the discrepancy is too large.

The 'best' hypothesis has no loss and in this case it would perfectly fit the training data. However, this might not be ideal, because it might not generalize well. The attempt is to predict unseen data as well. Perfectly fitted data this case could probably have overfitted the model to the data points. This model 'explains perfectly' the training data, but it might have worse performance in unseen data points. The model would be overly complex to explain the underlying phenomenon. In the case of overly simple model, it might not be able to capture the complexity of the underlying phenomenon. In this case it would underfit. Underfitting might happen with linear

models, where the phenomenon is some polynomial of a higher degree [20]. Over- and underfitting is discussed more in the next section.

2.2.2 SL and Generalization

One subfield of machine leaning is supervised learning. As discussed in previously, supervised learning (SL) has access to labeled data. This labeled data has been produced by some expert, usually a human. The labels are the unit of interest and the attempt is to produce a mapping between the input features $\mathbf{x} \in \mathcal{X}$ to some output label $\mathbf{y} \in \mathcal{Y}$. The emphasis has to be put on the hypothesis map $h : \mathcal{X} \to \mathcal{Y}$. The h is a function from input feature space $\mathcal{X}$ to output label space $\mathcal{Y}$. In essence this means that the function h converts the input to some output with some internal logic. The attempt is to make and approximation of the true label, i.e. $h(\mathbf{x}) \approx \mathbf{y}$. [12, 15] If the mappings are sufficient then the model has learned something about the true underlying data.

The common tasks in SL are classification and regression. Regression is the analysis of the mathematically continuous relationships between variables. In linear regression this dependency can be expressed as a linear combination of variables, for example $y = a + bx$. Non-linear regression analyses dependencies which are not linear, for example polynomial regression. In multivariate regression these dependencies can be expressed as matrices [21]. For the purposes of ML, these mathematically defined relationships can be used to predict values for future data points. Also one can predict values which were not contained in the original value interval through extrapolating the functions.

The attempt of ML methodology is to generalize. This means that a competently trained model also performs on data, which was not used for training. More precisely, the ML methodology tries to learn the underlying distribution of the data by approximating the true risk, the expected loss, with empirical risk, the training error. [12, 15, 22]. Most cases it is impossible to know the true risk due to the data only containing instances of some unknown distribution.

Empirical Risk Minimization (ERM) tries to minimize the empirical risk by approximating the true risk. The expected loss of a hypothesis $h \in \mathcal{H}$ can be defined as

$$\mathbb{E}\{L((\mathbf{x}, y),\ h)\}. \tag{1}$$

which would be minimized by

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}\{L((\mathbf{x}, y),\ h)\}. \tag{2}$$

with some the argument h [15]. This would be approximated with the empirical risk with

$$\hat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i) \tag{3}$$

and the ERM becomes

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{L}(h). \tag{4}$$

Practically the ERM performs very poorly because either the hypothesis set $\mathcal{H}$ is too simple to capture the complexity or the hypothesis $\mathcal{H}$ is ample but the estimation error bound gets to loose [12]. Additionally, computation of ERM solution can be computationally expensive, because of non-convexity or non-differentiability [15]. ERM is still important, because it is the principle of learning in ML.

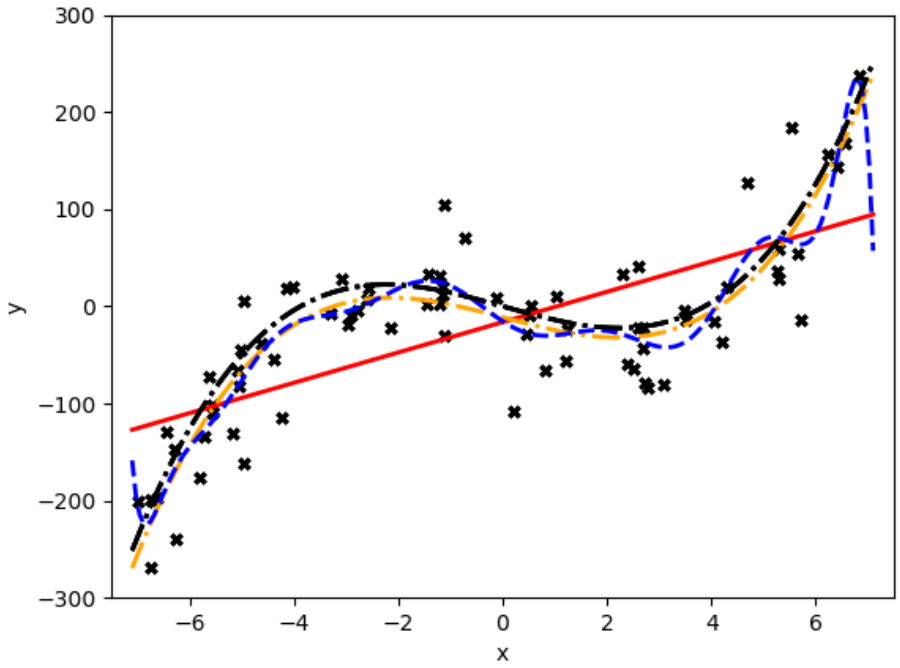

Figure 5: Plot of polynomials fitted to data. The underlying function is a degree 3 polynomial (dot-dashed black). The orange curve (dot-dashed) is a fitted degree 3 polynomial. Underfitted linear model is in red and overfitted polynomial with degree 14 in blue (dashed). Orange

ERM-based methodology fits models to a training dataset which may result in high performance. However, a naive use might lead to overfitting. Overfit is a situation where a complex model performs well for the training data, but performs worse for data which it was not trained for, thus generalizes poorly [20]. In overfitting the model might not learn the underlying phenomenon, but instead remembers the datapoints used for training, including the noise. In the extreme case the training error can be 0, where the model has fitted perfectly on data. This happens when the (effective) dimension of the model is at least the same size as the number of datapoints [15]. The theoretical framework for model dimension risk management is named Structural Risk Minimization (SRM). In SRM the structure and confidence interval is taken into consideration to penalize more complex models [22, 23].

Oppositely, when the model is too simple, i.e. dimension of the model is low, it is called underfitting. In Figure 5 the underlying phenomenon is a degree 3 polynomial. The datapoints have been generated from a random standard distribution generator, where the mean is at the degree 3 polynomial. The red line in the figure is an underfitted model, because it has a too low of a dimension to capture the phenomenon and usually should have the highest error. The dashed blue curve is an overfitted polynomial and should have the lowest (training) error. However, additional datapoints should increase the error dramatically. The fitted orange degree 3 polynomial is has a 'perfect' dimension and it approximates the phenomenon.

However, usually the underlying phenomenon is unbeknownst and some methodology to find the right model dimension is needed, i.e. the most generalizing model. Datasets are usually divided into train and test sets [24]. The train data is used for training and the test data is used to test the model. No test data should be used for training, because the purpose of test set is to test the models performance on new 'unseen' data, i.e. the generalization ability.

2.3 Classification

Classification encompasses situations where the output variables, labels, can attain only discrete and unordered values. These values are called the classes [25]. Examples of classes include different animals, such as dogs or cats, or objects, such as tables, chairs or closets. Intuitively, these possible classes have no order and are defined as separate cases. Input variables, the features, may be continuous and ordered, for example height or age. Classification can be divided into several different instances depending on the number of labels or classes, the linearity of the models or the type of feature space that is used, such as text or signals [26].

Decision boundary is the set of points which divide the decision space into two or more regions. This decision boundary divides that space into separate classes on either side of the boundary. In linear classifiers this decision boundary is linearly defined such as a line, a plane or a hyperplane [27]. In non-linear classification this decision boundary cannot be defined by linear means. A non-linear decision boundary can be, for example, multiple linear boundaries in piecewise definitions. All the subpieces could be linear, but the entire boundary would not be.

One important distinction is the difference between binary and multiclass classification. In binary classifiers the output label is one of two possibilities in some output space, i.e. $\mathbf{y} \in \mathcal{Y}$ where $|\mathcal{Y}| = 2$. In the most basic case a data point belongs to a class or not, and thus the output is binary [26]. Binary classifiers are simpler and in many cases they are sufficient, such as in fraud or disease detection. In multiclass classification this output space larger than two, $|\mathcal{Y}| > 2$, which could be in the case of different animals in image classification, such as dogs, cats or cows. This allows more expressiveness in labels as the space of possibilities is larger, but in turn can become a more complex system.

Multiclass classification can be achieved with either naturally extending classifier to multiple outputs or converting the multiclass problem into a set of binary classifier problems [28]. Algorithms that can be extended into multiple classes include algo-

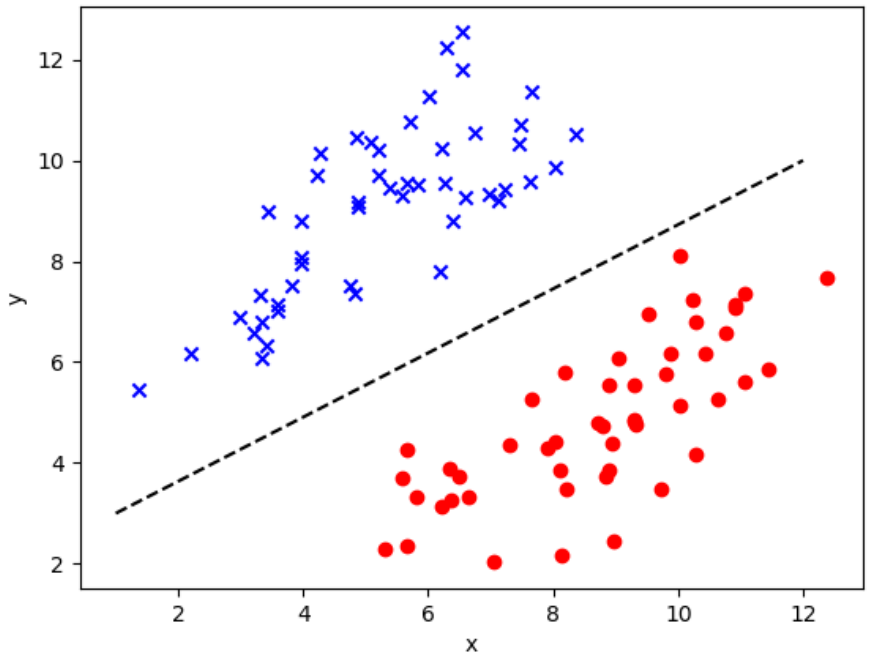

Figure 6: Example of a linear classification boundary (dashed black). In this case it perfectly divides data points, the blue crosses on top and red dots on bottom.

rithms such as ANNs, decision trees, k-Nearest Neighbors, Naive Bayes and extended SVMs. For example in neural networks one can have one output node or multiple output nodes, which is not restricted by the expressiveness of the model.

Commonly multiclass problems are converted into a set of binary problems, which include one-versus-all, one-versus-one and less commonly error-correcting output-coding approaches [28]. In one-versus-all approach a binary classifier is learned for each class y_n, where one classes' instances are put against all of the rest. This would result in total N binary classifiers, one for each class, where N is the amount of classes. In one-versus-one approach you compute a binary classifier for each pair of classes $(y_n,\ y_m),\ n \neq m,$, where the comparison is made only for the classes in question. This results in $N(N-1)/2$ binary classifiers. [29]

Using multiple binary classifiers might result in different levels of confidence of the right class. To combine these binary classifiers as a multiclass classifier a common approach is a aggregate voting method to choose the most fit class for a data point. The class with the most votes is used as the predicted class [29]. In case of ties the simple method is a random choice among the tied classes. More elaborate schemes use a confidence score, such as distance-based loss [30].

2.3.1 Text Data Processing

Text classification is a subset of classification, where datapoints are sequences of characters, words or sentences which constitutes documents, paragraphs or any written language. Text classification is also a subset of text mining, which is a umbrella term for various information retrieval methods and analysis of text data [31]. Text classification can be used for tasks such as sentiment analysis, spam filtering or document categorizing and the process usually consists of text preprocessing, feature extraction and selection, classification and evaluation [4, 32, 33, 34].

Text preprocessing can have a significant impact on the performance of the model, because it cleans and transforms raw data for easier processing [33]. Most commonly, preprocessing includes steps such as tokenization, stop-word removal, lowercase conversion and stemming.

In tokenization raw text are broken to smaller chunks of text such as sentences, words or sets of characters, named as tokens. Delimiter rules commonly divide text by non-alphanumeric characters such as punctuations and whitespaces which results tokens consisting of alphanumeric characters [33]. Stop-word removal deletes tokens, usually words such as "the", "a" or "and", which have low relevance for semantics or little information. Lowercase conversion ensures the same words are represented in the same digital format.

Stemming is the truncation of inflected words to some base representation, namely the stem. It does not consider the correct linguistic representation, but importantly attempts to map related or inflected words to the same stem [35]. An alternative to stemming is lemmatization, where the inflected words are transformed to the lemma of the word, i.e. the canonical and linguistically correct form. Lemmatizers uses complex algorithms to derive the correct lemmas and are usually more difficult to construct than stemmers [36].

After preprocessing the data is in processed text format, such as lemmatized tokens, which is yet difficult for an ML algorithm to process. To solve this, usually some Vector space model, such as a Bag-of-Words (BOW) representation is used [32]. In these models the words are transformed into a numerical value vectors. In BOW specifically, every occurrence of a word is counted, the term frequency, and these occurrences of words are put in a vector. Every element in the vector thus represents a word with an associated frequency, the order and context are disregarded [5]. A BOW vector can be regarded as a datapoint. Multiple vectors can be combined into a matrix where every row is a datapoint and every column is a word feature. Every value in that matrix is thus term frequency of the word in that datapoint.

One of the problems with bag-of-words is the disregard of word contexts and order [5]. One method to fix this is the n-gram, where every consecutive characters or words of size n are combined as unit of interest [5, 32], for instance, a 2-gram (of words) would contain every consecutive pair of words as a token. One of the problem with n-grams is the difficulty to find an effective size of n.

A common text data representation is the Term Frequency - Inverse Document Frequency ('TF-IDF'). TF considers the frequencies of a term (word) and the length of a document, i.e. the number of times term t is in a document divided by the total

number of terms in that document. TF part measures the occurrence of the term in the document. IDF is the logarithm of total number of documents divided by the number of documents with term t. IDF can be interpreted as the amount of information in a term [37]. TF-IDF is common in text classification as it is a simple representation in the importance of a word to a document [38].

2.3.2 Models

Hypothesis space $\mathcal{H}$, the model, is a set of all mappings between feature (input) space to label (output) space where one hypothesis h is mapping of a feature space $\mathcal{X}$ to a label space $\mathcal{Y}$, i.e. $h : \mathcal{X} \rightarrow \mathcal{Y}$. The goal is to find a hypothesis $h \in \mathcal{H}$ where for any datapoint $x \in \mathcal{X}$ as an input to the hypothesis approximates the true output $y \in \mathcal{Y}$ [15]. The relationship with data and loss was described in Section 2.2.1.

Hypothesis space $\mathcal{H}$, is a design choice where computational complexities and statistical properties should be considered [15]. The hypothesis space grows exponentially with increases in dimension and the hypothesis maps often are uncountably infinite. Because it may be computationally infeasible to find a good hypothesis from this space, the practical ML algorithms introduce a search heuristic [39]. Often the problem of finding a good hypothesis relies optimization techniques, for which the performance depends on space complexity and model expressiveness.

To reduce the space complexity, the feature space can pruned only to keep the most relevant dimensions, which is called feature selection. In feature selection for classification the aim is to find the optimal subset of features which maximizes the classification accuracy [40]. Other technique use dimensionality reduction, such as Principal Component Analysis (PCA), where redundant dimensions are transformed into a new dimension [41]. In PCA a new coordinate system is found where the data can be expressed with fewer dimensions without large errors.

Model expressiveness means the model's ability to express a range of possible hypotheses. In linear models the expressive power is limited to linear combination of variables, such as in $\mathbf{y} = A\mathbf{x} + B$, where vector $\mathbf{y}$ would be linearly dependent on vector $\mathbf{x}$. Examples of classification algorithms that use linear models include Logistic Regression and Bayes Classifier. A linear classifier divides a space in half, a linear classification boundary, where the opposite halves are the classes of a binary classifier [15].

Most often the real world data is slightly or highly non-linear. Non-linear models, such as polynomial regression, can capture non-linear dependencies between the input and output variables. Non-linear classification tasks include image and sound classification, which use non-linear models such as neural networks [42, 43]. The benefit of non-linear models is the ability to capture non-linear dependencies, but usually the computation costs are higher. While the real world data might be non-linear, often linear models have comparable performance to non-linear models [5, 27, 44]. One of the problems with non-linear models is their greater need of data. Usually highly non-linear models such as ANNs are beneficial when the data is poorly captured with linear models and excess of data is available with enough of

computational resources.

2.3.3 Loss Function

As discussed in Section 2.2.1 loss functions are used to train model with an loss calculation, the error, between fitted model and the data. This section discusses the different loss functions in classification, namely 0/1-loss, hinge loss and logistic loss.

The difference between regression and classification is the output (label) domain is continuous and discrete, respectively, and this difference is present in the loss functions. In regression the loss functions are 'valley'-like, where the smallest error is in middle and increases as the input approaches negative or positive infinity, and in classification they are 'downhill'-like, where the error approaches 0 as input approaches positive infinity. This is because regression predictions can exceed or subceed true values, but for classification the input values are either correct or exceedingly incorrect [45]. Although this is mostly correct, some loss functions in classification also penalize correctly classifier instances.

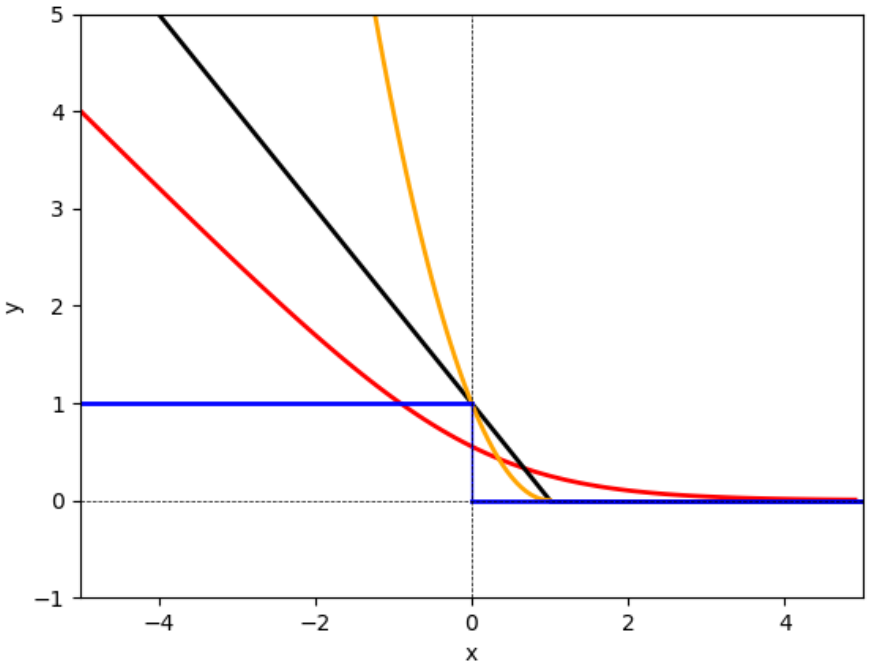

Figure 7: Losses in classification. Blue line is the 0/1-loss, black line is the hinge loss, orange line is the squared hinge loss and red is the logistic loss.

Common loss function is the 0/1-loss, where the loss is 0 for all correctly labeled instances and 1 for all incorrect labeled instances, denoted with

$$L_{0/1} = \mathbf{I}(\hat{y} \neq y) \tag{5}$$

where **I** is the indicator function. The loss function essentially minimizes the number of misclassifications. While the purpose of classification is to generalize and correctly classify examples, i.e. minimize misclassification, the 0/1-loss suffers from non-convexity and non-differentiability, and as such is hard to optimize directly [46]. The benefit of 0/1-loss is the robustness to outliers.

A convex function is preferable as they can be solved reliably and efficiently [47]. Hinge loss and logistic loss are examples of convex functions.

$$L_{Hinge} = max\{0, 1 - yh(\mathbf{x})\}. \tag{6}$$

$$L_{Logistic} = log(1 + exp(-yh(\mathbf{x}))) \tag{7}$$

Hinge loss requires some margin near the decision boundary until the loss is 0, where the additional margin increases confidence [45]. For misclassifications the error increases linearly. Hinge loss is applied successfully in support vector machine. Non-differentiability is one of the drawbacks and smoothed functions like logistic loss can be used.

2.3.4 Performance metrics

Commonly used performance measures in classification include accuracy, precision, recall and F1-score. These measures can be used in multiclass classification, but is more easily understood in terms of binary classification [48]. In binary classification you have one of two options as classes, usually denoted as the positive class (+1) or the negative class (−1). Importantly, when comparing the performance, the comparison is made between the 'true' classes, i.e. the labels stated in the collected data, and the predicted classes, i.e. the labels supplied by the ML predictor.

Cases correctly recognized as positive classes are defined *true positives* (TP), correctly recognized as negative classes are *true negatives* (TN), incorrectly misclassified as positive classes are *false positives* (FP) and incorrectly misclassified as negative classes are *false negatives* (FN). This is usually visualized as a confusion matrix, such as in Figure 8, where the intersections between predicted and actual for positive and negative are shown. In practice the intersecting boxes are filled with integer values to report the amounts of cases for each box. The main diagonal is the 'true' cases, where prediction and actual labels agree on the class.

		Predicted label	
	Label	Positive	Negative
Actual label	Positive	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

Figure 8: A confusion matrix, where the actual and predicted labels can be compared as integer values in the relevant boxes.

The common performance measures accuracy, precision, recall and F1-score, as well as many other metrics, are calculated from these true or false positives or negatives [48, 49]. The most commonly understood metric is the accuracy, which is all the correctly (true) classified instances (TP+TN) divided by all the instances (TP+FN+FP+TN).

$$Accuracy = \frac{\text{correctly classified instances}}{\text{all instances}} \tag{8}$$

Recall is all of the correctly classified positives in total positives, which is the retrieval rate of relevant data points. Precision is all correctly classified positives in total predicted as positives, which is the relevancy rate of the retrieved data points.

$$Recall = \frac{\text{correctly classified positives}}{\text{total positives}} \tag{9}$$

$$Precision = \frac{\text{correctly classified positives}}{\text{total predicted as positives}} \tag{10}$$

F-measure is a weighted harmonic mean of the recall and precision; a combination of precision and recall. The intention is to combine these into an F-score for a measure of 'effectiveness' [50]. F1-score gives an equal weight of the precision and recall.

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

While these performance metrics are widely used in classification they have deficiencies such as their focus on only one class and the biases for the majority class and as such require attention on the interpretation and use of these scores [49, 50].

2.4 Algorithms in Classification

In practice much consideration is in the algorithms, combinations of data, model and loss. As discussed in Section 2.2 the main differentiator between supervised and unsupervised learning is data. In SL the data is labeled and in classification (Section 2.3) the labels are discrete and unordered. The difference between the algorithms should have different feature data, model expressivity and loss calculation. This section briefly discusses each algorithm used later in the thesis.

A naive Bayes classifier can be shown to be the optimal classifier under assumption of independent variables, because this classifier minimizes the misclassification rate, 0/1-loss (Equation 5) [51, 52, 53]. However it is called a 'naive' classifier due to the independent variables assumption as they are rarely truly independent. The naive Bayes classifier is a probabilistic method where the probability being in class 'a' is

$$p(a|E) = \frac{p(E|a)p(a)}{p(E)} \tag{12}$$

which is a Bayesian formula. In the simplest binary case, if the probability of an instance belonging to the positive class (+1) is higher than the negative class

(-1) then the instance is classified as such, and vice versa. A Bayesian classifier requires an assumption of the underlying probability distribution [51]. While a 'naive' assumption of independence is rarely true, the classifier performs if the actual and estimated probability distributions agree on the most-probable class [53].

Logistic regression is technique where the model predicts a probability of a label with logarithm of the odds. Despite its name, logistic regression is a classification algorithm as it expresses a probability of an outcome for a binary case [54, 55]. Logistic regression minimizes a logistic loss (Equation 7) where the hypothesis would be $h(x) = \mathbf{w}^T\mathbf{x}$ [15]. The produced probability can be expressed as

$$p(y = 1) = \frac{1}{1 + exp(-\mathbf{w}^T\mathbf{x})} \tag{13}$$

with a linear combination of the weights $\mathbf{w}$ and inputs $\mathbf{x}$. While it is a continuous function, the output is mapped to a range between [0, 1] as a probability of a binary label. As the hypothesis is a linear combination of variables, logistic regression is a linear model.

Support Vector Machine (SVM) is a binary classification technique. The basic idea is to separate multidimensional data with a maximum margin hyperplane. In 3-dimensional space this hyperplane would be a 2-dimensional plane. The hyperplane can be expressed as hypothesis $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ where b is some (offset) constant [56, 57]. The attempt is to perfectly separate the two classes with this hyperplane, which would be the decision boundary for a prediction.

For linearly separable datasets there can be many if not infinite amount of different data-separating hyperplanes. For a maximum margin, the decision boundary should have the maximum distance from the closest data points, the support vectors. Thus SVMs use hinge loss (Equation 6) to calculate the error, where a margin is added for a correct classification [56]. The minimization function becomes

$$L(\mathbf{x}, y) := max\{0, 1 - y(\mathbf{w}^T\mathbf{x} + b)\}. \tag{14}$$

Usually data is not linearly separable and an additional regularization term $\lambda||\mathbf{w}||$ is added for a soft margin [15]. The ordinary SVM is a linear model.

SVMs can be applied to non-linear data with the use of kernel functions. Kernel functions $K(x, x')$ use a mapping function ϕ where a lower-dimensional space is mapped to some higher dimensional space. Usually this higher-dimension can be computationally inefficient and a so-called kernel-trick is applied where a dot product with the function ϕ equals the kernel function as $K(\mathbf{x}, \mathbf{x}') = \langle\phi(\mathbf{x}), \phi(\mathbf{x}')\rangle$. The specific function ϕ doesn't need to be used directly as the kernel-trick can be applied when the feature vector $\mathbf{x}$ exist within the kernel function [58]. An example is the Radial Basis Function (RBF) where the kernel function is

$$K(\mathbf{x}, \mathbf{x}') = exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2), \ where\ \gamma > 0. \tag{15}$$

Because of the equality between the kernel and the dot product, the information from this higher dimension is extracted without actually converting data to a higher

dimension. A non-linearly separable data can become linearly separable in this higher-dimensional space [59].

k-Nearest Neighbor (kNN) is an algorithm where k classes are set and a similarity measure is used to classify examples. Similarity measure is usually the Euclidean distance between data points. Given some data point m, if the distance to data points of some class k_i is smaller than to other data points of some class then the data point m is assigned to class k_i [60]. The most similar data points are thus found in clusters, as the distance within these data points in the cluster are lesser than to other clusters. The performance is dependent on the preset amount of k classes. In text classification the similarity calculations can be computationally expensive as the dimension of the feature space can be massive, which might require dimension reduction to combat the high computational complexities [61].

Decision trees are classification algorithms in which the data structure is a tree. These trees have (decision) nodes and links to children nodes, and each these nodes encode simple rules to decide which path to take. The end nodes are the classification labels which depend on the taken path. In binary decision trees the rules in the nodes can be a simple 'yes' or 'no' answer to some feature in the data [62].

The construction of a decision tree involves growth and pruning phases [63]. In growth phase each new node, including the root node, partitions the data into new child nodes until the leaf node is associated only with one class. The partitions use some goodness measures to decide on the splitting, usually heuristically based either on information gain, error or statistical significance [63, 64]. An example of a measure is the Gini value

$$Gini(D) = 1 - \sum_{j=1}^{n} p_j^2 \tag{16}$$

where p_j is relative frequency of class j in dataset D [63]. In the pruning phase the least important nodes or subtrees are removed to hinder overfitting. Some pruning can also happen before node splitting if some predetermined threshold is not fulfilled.

Random forests are a type of ensemble methods, which are a group of learning algorithms where multiple base decision makers combine into a unified learner [65]. In particular, the random forest construct multiple decision trees and the ultimate class is the aggregate vote of these trees [66]. Random forests use randomize the data given to individual trees in order to avoid overfitting and bias.

One of the earliest ML algorithms is the Perceptron. The algorithm is a linear classifier in which the output is a linear combination of weights and inputs

$$y = \begin{cases} 1 \; if \; \mathbf{w}^T\mathbf{x} \geq \theta \\ 0 \; else \end{cases} \tag{17}$$

where θ is some threshold [67]. Like the SVM, the perceptron divides a space with an hyperplane to two (binary) subspaces. The perceptron is trained a data point a time where the algorithm predicts the label of the data point using the current weight vector $\mathbf{w}$. Only on wrong predictions the weight vector $\mathbf{w}$ is updated by adding $y\mathbf{x}$. This update corrects the weight vector each time a misclassification

is made [12]. In the linearly separable case the perceptron converges to correctly classify all examples [67].

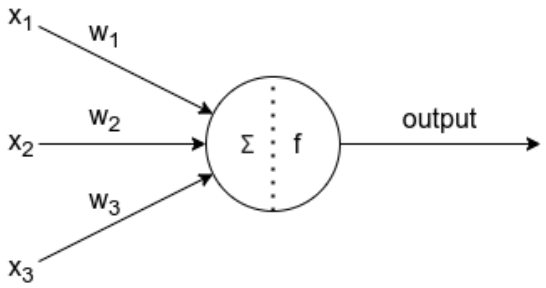

Figure 9: Single neuron with feature x_i and weights w_i. The values of $\mathbf{w}^T\mathbf{x}$ are summed and some activation function f is used to produce output.

The true power of perceptrons manifests in the Multilayer Perceptrons (MLP) which is a type of feed-forward neural network. The simple perceptron unit takes the features $\mathbf{x}$ and the associated weights $\mathbf{w}$ as it's input. In MLP there are multiple perceptron units, the neurons, grouped together as layers in a connected network by the weights [68].

For each neuron the input vector $\mathbf{x}$ and the weights $\mathbf{w}$ are fed through some activation function as $f(\mathbf{w}^T\mathbf{x})$, for example the Rectified Linear unit

$$f(x) = max(0, x) \tag{18}$$

This activation is then forwarded to the next layer of neurons for processing. In multi-class classification the last layer contains multiple neurons, one for each class. The highest activation in the last layer is voted as the predicted class.

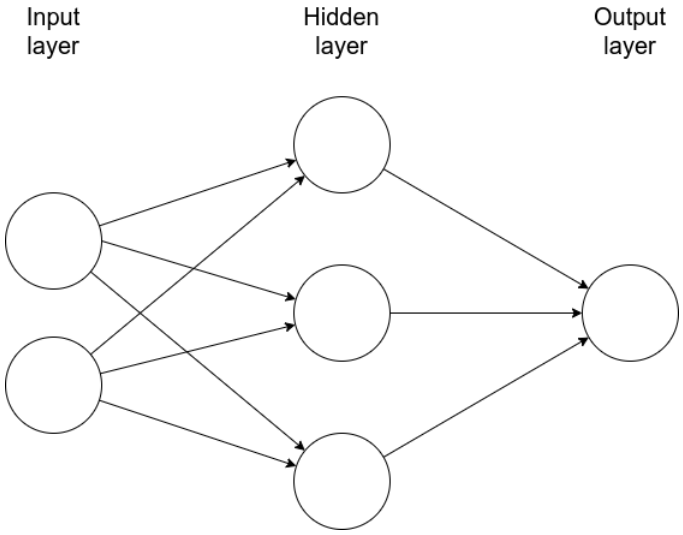

Figure 10: A simple feed-forward neural network with an input layer, a hidden layer and an output layer.

MLP is trained in a process called backpropagation. The network weights are initialized usually randomly. Each input vector is propagated through the network to gain some output and calculate some error, which is propagated in reverse through the network. The neurons are adjusted slightly with the errors in order to minimize the overall error [69]. The training of a MLP is quite expensive, because the amount of neurons (and connections) in the network can be massive.

2.5 State-of-the-art implementations

Purchase invoice account selection is a classification task of text data. This is because the most important feature of the data is the textual description of the item bought. This means that some NLP techniques need to be used. Indeed the research regarding automatic invoice processing usually points the text processing techniques [6, 7]. To be clear, account selection is not the only part that can be improved in processing of purchase invoices. There might be problems in incomplete data, workflow deficiencies and other repetitive tasks. Some solution suggestions to these include electronic invoices, workflow automation and rule-based automation [70, 71].

In this thesis only the analytic account selection is researched and the other problem areas in purchase invoice processing is disregarded. It is suggested that methods such as k-nearest neighbor, Decision Trees, Random Forests, AdaBoost, Naive Bayes, Support Vector Machines and Multi-layer Perceptrons and ensemble methods can be used in ML account selection of purchase invoices [4, 5, 6, 7]. Many different Artificial Neural Networks and Support Vector Machines can have high accuracies, but usually these algorithms can be more complex. The more simple models achieve worse accuracy, but are more explainable and easier to construct [5, 7].

3 Design and Development

The purpose of this chapter is to design and develop the ML system for purchase invoice account processing. This includes the problem definition with the associated restrictions and scope. The available data will be presented and reasonable data processing will be conducted. Models and losses will be designed to adhere to the foundational ML technologies and practices discussed in the previous chapter. Algorithm selection will include the design choices, which include scalability and other needs.

The main tool to be used is the Scikit-learn (sklearn) ML library in Python [72]. It features state-of-the-art implementations of ML algorithms for supervised and unsupervised learning and related model selection, evaluation, inspection, visualization and data transformation. The reason to use sklearn is the easy-to-use implementations of the classical algorithms in supervised learning.

3.1 Problem Definition

The aim of this thesis is to develop a model which classifies purchase invoice lines. A data point would thus be an invoice line in a purchase invoice. The attempt is to classify these invoice lines to some analytic account. The main feature is the textual description of that invoice line. The processing of text requires some natural language processing capabilities. There is a possibility that the many invoice lines in the same purchase invoice are correlated, because they share some data. The research problem of the thesis is a multi-class text classification problem.

The system will be a proof-of-concept, and one of the goals is to assure the continuation of the development afterwards. If the produced software is not satisfactory then the thesis cannot persuade the continuation of development. To make the thesis decently satisfactory some goals, requirements, restrictions and other specifications need to be made. This include that the architecture adheres to the general needs of the commissioning company as well as some more precise ML performance metrics.

One of the requirements is scalability and as such one of the objective is to classify only some of the labels. This requirement is important, because developing a whole software system is too big of a task for this thesis, but instead developing a proof-of-concept system helps the decision making later and scalability helps to make the system more whole later. The model has to be general enough also so it can be applied to multiple different companies. In a sense the requirement is to develop a pipeline where some varied data is fed to the system and it would learn that company specific standards with some model which can handle the varied data.

There is also the possibility for lack of data. For some models and architectures the need for data can be much greater than for others (2.3.2), and it needs to be accounted for. Additionally the time (and memory) complexities should be monitored. At least the training time and the prediction time in production need to be decent. Decent is not specified at this time, because the definition depend on the design.

It is difficult to find performance metrics for Odoo's current ML capabilities. One of the claims is a roughly 90% accuracy for account predictions. Notably, 'an

account' is different than 'an analytic account', and the values cannot be directly compared. However 90% can be set as a subjective 'good' performance goal. As said in Section 2.3.4 the accuracy metric alone can have biases, therefore comparative recall, precision and F1 scores are desirable. While these scores are widely used in ML, it is still important to use them with care and acknowledge their biases and flaws.

3.2 Available data

For this thesis production database data has been received from a customer company. This company wishes to be anonymous for the purposes of this thesis. As such, only the relevant data points with no publicly identified features of this data will be revealed. The data has 32380 data points (invoice lines) in 7138 different purchase invoices. Of those 17652 have an analytic account associated with that data point. There are a total of 39 different analytic account. As stated previously, only a select few analytic accounts will be processed.

count	id
2262	71
1041	88
1617	68
2981	83
1410	92
2455	76

Table 1: The amount of data points (count) per analytic account (id) where the amount is over 1000.

In the table 1 only the number of data points per analytic account with over a 1000 data points are shown. Of the labeled data these six analytic account have a total of 11766 data points, which is two thirds of all the labeled data. There are 18 analytical account with row counts between 1000 and 100. The total amount of data points in these 18 accounts is 5485. If these data points are included we have over a 97 percent coverage of all labeled data. The rest of the 15 labels, with under 100 points per label, have 401 data points associated with them.

To reiterate the data point is an invoice line. An invoice line is a row of products in a purchase invoice. A line has multiple independent features and the purchase invoice itself has its own features. Thus many lines share common features. Purchase invoice include the following features:

- Name (name)
- Vendor (partner_id)

- Bill reference (ref)
- Payment reference (payment_reference)
- Recipient bank (partner_bank_id)
- Bill date (invoice_date)
- Accounting date (date)
- Due date (invoice_payment_term_id)
- Journal (journal_id)

These features are shared with all of the invoice lines within a purchase invoice. These features are not as important, because these features are associated usually with the vendor which can be some outer company unrelated to analytic accounts. The analytic accounts are usually related to some internal bookkeeping. More should be inferrable from the features associated to the individual features in a invoice line. Invoice lines, a row of product, has features such as:

- Product (product_id)
- Label (name)
- Quantity (quantity)
- Price (price_unit)
- Taxes (tax_ids)
- Subtotal (price_subtotal)

The most important feature is the feature 'Label'. As one might have notice, there is a mix of terminology. The feature 'Label' is named as such in the convention of Odoo, but is in fact a feature. The labels in supervised learning are the things to predict. Regardless, the feature 'Label' is a textual description of what have been bought. This feature will be referred as the 'textual description' to lessen confusion between the mixing of terminology. Because the most important feature is this textual description there is a clear need for advanced natural language processing techniques.

Additionally the invoices have more data than is seen from the Odoo user interface. Invoices are usually sent in the Finvoice XML data transmit protocols which are processed to extract more information from the invoices. Some of the Finvoice data is already processed to the textual feature 'Label' and might be redundant, but other might give valuable additional information. Features in the invoice include

- Invoice:
    - Order Identifier (finvoice_order_identifier)

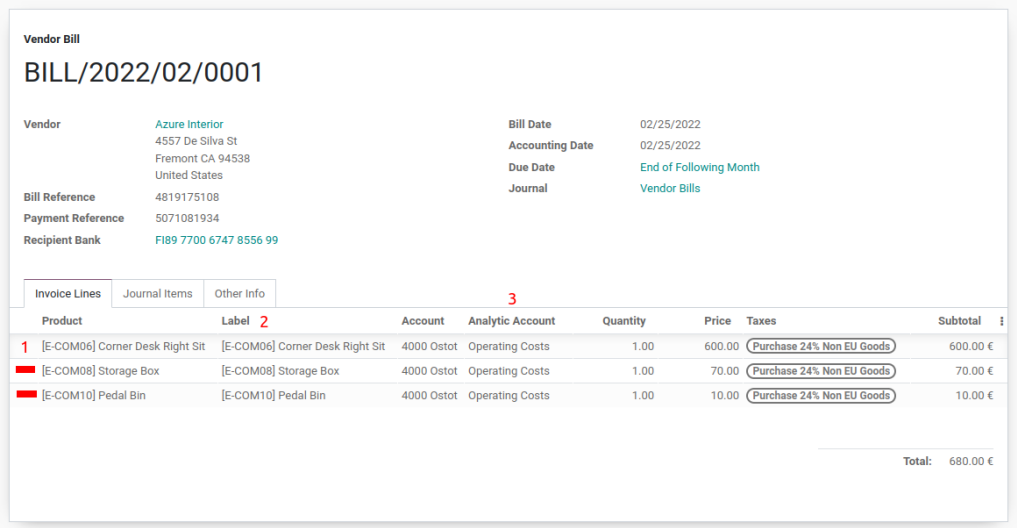

Figure 11: (1): data point (invoice line), (2): main feature (textual description), (3): main label (analytic account). In Odoo the textual descriptions are named as 'Labels', but in terms of machine learning terminology a 'label' is the thing to predict and in this thesis it is the analytic account.

    - Buyer Reference Identifier (finvoice_buyer_reference_identifier)
    - Seller Reference Identifier (finvoice_seller_reference_identifier)
    - Project Reference Identifier (finvoice_project_reference_identifier)
    - Seller organization name (finvoice_seller_organisation_name)
- Rows:
    - Article Identifier (finvoice_article_identifier)
    - Article Name (finvoice_article_name)
    - Row Free Text (finvoice_row_free_text)
    - Row Identifier (finvoice_row_identifier)
    - Row Order Position Identifier (finvoice_row_order_position_identifier)
    - Row Position Identifier (finvoice_row_position_identifier)
    - Row Buyer Reference Identifier (finvoice_row_buyer_reference_identifier)

There can be tens or hundreds different analytic account, the labels in terms of ML. These labels may be cost-centers or project based accounts. A company may choose the way they use the these analytic accounts, which makes the amount of accounts different for each (2.1). The huge amount of labels makes it more difficult to teach a model to correctly predict labels. The approach should be to only learn only some labels to keep the scope of the thesis smaller and the complexity of the modeling lesser.

3.3 Design

One of the requirements is the applicability to multiple companies with different accounting standards. Different standards alone makes it quite difficult to build one general model to be applied to multiple companies. Additionally data for only one company is available and the ML labels for each company might differ, building only one model seems quite challenging. Because of these reasons it seems to be more beneficial to construct new models for every new company.

Additionally there is a requirement for learning new standard or shifting to a newer standard within a company. For a ML model to learn a new hypothesis it needs to be trained again, which can cause some unwanted upkeep and training overhead. This could be apparent in artificial neural networks such as MLPs for their excessive training time despite being quite accurate if used correctly. The system needs to be scalable to multiple labels as well, which requires the model to reliably differentiate the classes.

Models can be trained and stored in Odoo, which is presented in 3.5. Because of the requirements this thesis will make a comparison in Chapter 4 between the different methods presented in Section 2.4. As discussed briefly in Section 2.5 the ANNs and SVMs can have higher accuracies, but can suffer from computational complexities. The simple models are easier to compute, but might not be as scalable to multiple labels. The most successful model will be optimized even further with an extensive gridsearch, which is a parameter optimization technique by searching the combinations of parameters.

One of the benefits of Scikit-learn is the ease-of-use and the availability of many state-of-the-art implementations of ML algorithms. One of the drawbacks is the lack of ANN support in the library, which features mainly multilayer perceptrons. However this might not be a great issue, as MLP's might not even fit to the requirements. This section will briefly discuss parameters for the algorithms. The thesis compares the algorithms identified in Section 2.5, namely Logistic Regression, Naive Bayes, k-Nearest Neighbor, Decision Tree, Random Forest, Support Vector Machines and Multilayer Perceptrons.

3.4 Data preprocessing

Important part of text classification is the preprocessing of text data. Text preprocessing boost the performance and is necessary for models ability to learn the features.

All the invoice and row specific Finvoice text data and the feature 'Label' was used as features for the model. The data is fetched with a simple SQL 'SELECT' statement from the database which has filters ('WHERE') to reduce irrelevant datapoints. Datapoints require to be a posted invoice where an analytic account is set for a row and the analytic account is in active use.

This data is then concatenated as variable, i.e. block of text with spaces between words, and transformed to lowercase, for the characters are represented identically for the computer. The text is tokenized, where a token is at least two consecutive

alphanumeric characters separated with empty characters.

Although the removal of stopwords and stemming should improve the performance, as discussed in Section 2.3.1, empirical testing with this specific data showed negligible improvements, usually less than 1 percent improvement. Although this could be considered an improvement, it seems insignificant enough to justify the computing overhead produced by these methods.

The data contains mostly Finnish words, and presumably stopword removal and Finnish stemmers should improve the performance. The problem is that the data is not really natural language, as in semantically consecutive and coherent text, but the data mostly contains keywords such as 'vuokraus' (eng. 'renting') or 'huolto' (eng. 'maintenance'). Because the data is mostly a collection of keywords, they are usually already in their basic lemma and contain little stopwords.

One of the considerations is also the mixture of English and Finnish words. Although the data contains mostly Finnish with some English, the consideration have to be taken if applied to other datasets. One of the problems is that English and Finnish language have different stopwords and lemmatizers and mixing these processors might lead to poor performance. One solution might be a language detector, such as 'langdetect' python library, and applying the corresponding processors to that language. However, empirically testing lead to significant computing time overhead and the benefit is negligible as the data is mostly a series of keywords.

If some vocabulary is common between different documents it is assumed to be less relevant to classify that document, because as a common feature it is harder to use as a differentiator. Likewise to stemming and stopword removal, the addition of a IDF (in TF-IDF) had empirically little benefit, probably because the data is mostly a collection of keywords and also the amount of data in datapoints is little (compared to full text documents such as articles of books).

The tokenized words are processed to a bag-of-words representation. As discussed in Section 2.3.1, in n-gram representation n consecutive words are grouped together to form a feature in a vector space model to take word context and some order into account. Some vocabulary in the data is clearly a relevant sequence of word pairs or longer n-gram and it seems beneficial to use n-grams. However, empirically testing different combinations of n-grams with exhaustive gridsearch indicates little improvement in the classification. The reason might be that the words are already present in one word BoW, and because of the shortness text per datapoint, they already have significant weight.

3.5 Structure in Odoo

The analytic account classifier module for Odoo is quite simple. Odoo backend is written in python and the scikit-learn can be run easily on an Odoo server. The Odoo backend supports easy definitions of variables which can be shown in the user interface with simple XML annotation. In simple terms one needs to define a python variable (or function), assign it a value and annotate it in the XML file for the views to work.

A classifier can be trained in Odoo server with a click of a button ("Train

Classifier" in Figure 12), given the module and dependencies is installed correctly. In the accounting view of vendor bills another button is present ("Predict Analytic Accounts" in Figure 14), and by clicking it the module will predict analytic accounts for every invoice line. The previous button is an utility which could be replaced with and automatic prediction process where the prediction executes without the need of a button press.

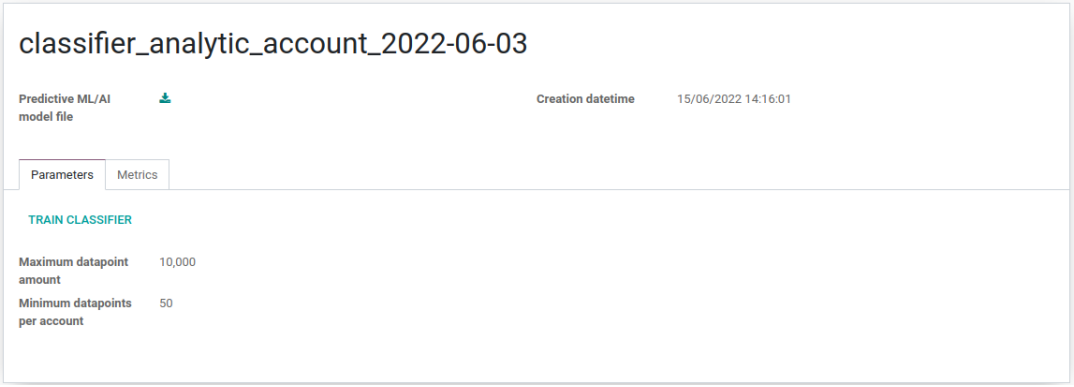

Figure 12: View of a classifier in Odoo. Parameters page allows to limit the maximum processed data points in the whole set, and a limit of minimum data points per trained analytic account.

In Figures 12 and 13 the Odoo views present the basic interface for training and evaluating a classifier. The "Train Classifier" button trains a classifier in the server with two parameters: the maximum amount of data points and the minimum amount of data points per trained account. The limit minimum data points per accounts is a guard to restrict labels with too few data points, which might be harder to learn.

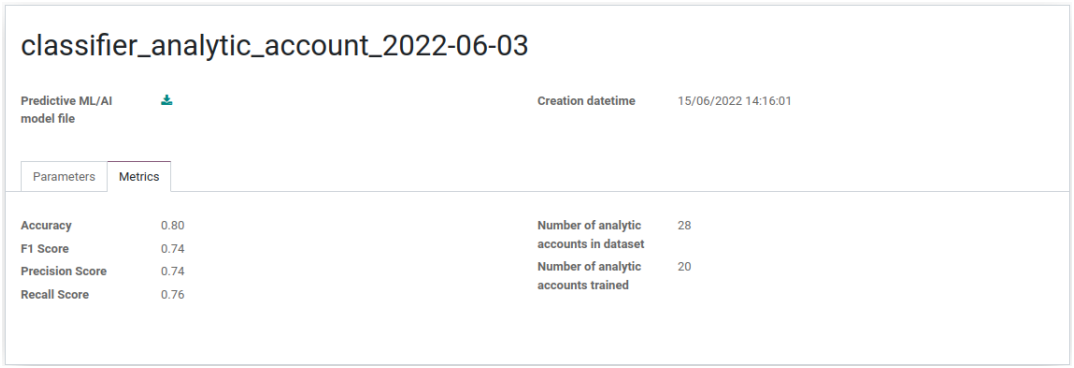

Figure 13: A view of the metrics view of a classifier in Odoo. The four basics macro metrics are given. There was 28 analytic account total in the dataset but only 20 of them had more than 50 data points.

The main reason to limit the amount of data points ("Maximum datapoint amount") in the whole set is the systems ability to forget old data (when new is introduced) so that in time the classifier might adapt to changes in the company's

accounting customs. A job scheduler ("cron") is set in order to train a classifier in set time intervals. When new job is executed the new data is introduced and the old data is forgotten in the cutoff point.

In Figure 13 some basic metrics is shown in the interface, the accuracy and the other macro scores F1, precision and recall. In that example figure the classifier was trained with cutoff of 50 data points per analytic account, which resulted to only train 20 of the 28 total analytic accounts in the data set.

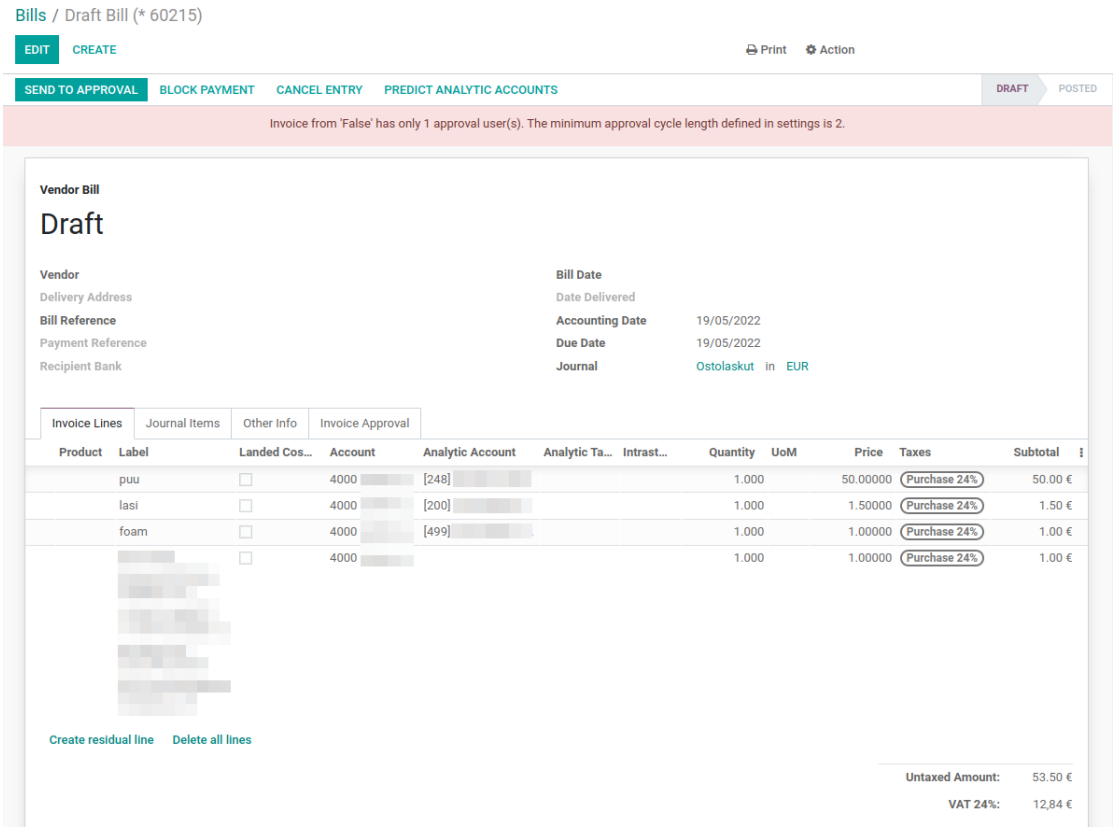

Figure 14: The accounting view for predicting analytic accounts in Odoo. If a classifier is trained, the button "Predict Analytic Account" on top predicts an analytic account for every invoice line. Some of the text is redacted for anonymity.

The model is trained, stored and used within Odoo. The Scikit-learn pipeline produces an object which can be pickled into a file and stored in the database. For predictions the file is unpickled and the line data is fed through this object. The database may store multiple different and older trained versions of the model, but the most recent one is used for predictions. The algorithm used in this module is hard coded, the selected best algorithm from the next chapter.

4 Evaluation

This chapter presents the results of different ML methods applied to the data and discusses strengths and weaknesses of these methods. The strongest method will be reviewed more carefully after the comparison. Issues with the chosen method will be discussed at the end of the chapter.

4.1 Comparison of Methods

Compared algorithms are linear and non-linear SVM, kNN, Decision Tree, Random Forest, Multinomial Naive Bayes, Perceptron and Multilayer Perceptron. The performances are presented in Table 2. Data was processed as discussed in Section 3.4. Data is limited to 10000 data points with 90:10 train-test-split, with 29 different analytic accounts of which 20 was used as they surpassed a 50 data points threshold.

The 9 smallest classes (below the threshold) were grouped together as an aggregate class designated as "-1". This aggregate class is included in the accuracy and macro average metrics. The algorithms have not used parameter optimization yet, such as grid search. However a 5-fold cross validation is performed each time to avoid overfitting.

Method	Multiclass	Accuracy Train	Accuracy Test	Precision	Recall	F1	Time (s)
Multinomial Naive Bayes	extended	0.67	0.69	0.65	0.40	0.44	**1.2**
Logistic Regression	one v. rest	0.78	0.80	**0.81**	0.70	0.73	8.5
Linear SVM	one v. rest	**0.80**	**0.81**	0.76	**0.76**	**0.75**	1.8
Non-linear SVM (rbf)	one v. one	0.76	0.78	0.76	0.65	0.68	35.6
k-Nearest Neighbor	extended	0.74	0.75	0.67	0.68	0.67	2.3
Decision Tree	extended	0.73	0.76	0.69	0.70	0.69	5.1
Random Forest (100)	extended	0.76	0.78	0.71	0.69	0.70	39.3
Perceptron	one v. rest	0.77	0.79	0.72	0.72	0.69	1.6
MLP (10x2)	extended	0.78	0.79	0.73	0.73	0.73	176.7

Table 2: Comparison of algorithms. F1, precision and recall are macro average scores. "Extended" in multiclass are methods which extend naturally from the binary case. Non-linear SVM uses a radial basis function ("rbf") as a kernel. Random forest has 100 trees. Multilayer perceptron uses two hidden layers with 10 nodes each. Best performance is in bold.

In Table 2 the best performances are with the linear SVM, logistic regression and MLP, while Multinomial Naive Bayes performed the poorest. The other algorithms have quite good performance, but have a few percentages less in precision, recall and F1-score compared to the best performer of linear SVM. As suggested by Section 2.5 the SVM and MLP should have the best performances, which is yet enforced by these results.

One notable things is the high accuracies compared to the lower percentages with precision, recall and F1-score. As said in Section 2.3.4 accuracy only measures the correct predictions, but the problem is the uneven sizes of the labels. Most of the data points are belong to few labels which skews the accuracy metric, which means the performance might be worse in most of the labels. However the comparable values between train and test accuracies reflect absence overfitting. Test accuracies are percentage or two higher, but could be explained by random chance of data set splits.

MLP had the greatest computation time. Even with very simple hyper-parametrized MLP, two hidden layers with 10 nodes each, the computation time was orders of magnitudes higher that the fastest. However the performance scores show promise with MLP. Assuming a better hyper-parametrization the performance arguably could exceed the SVM. However adding more layers or neurons the computation time would become worse, which might not be preferable.

The Naive Bayes was the fastest in terms of computation time, but had the worst scores of all the compared algorithms. The problem might be that the algorithm has strong assumptions of data distributions, which might not agree with the underlying distributions.

The linear models, SVM, perceptron and logistic regression, have good scores compared to the more complex models, k-nearest neighbor, decision trees, random forest and non-linear SVM. The reason for this might be a some degree of linear separability between the labels. The added complexity of the non-linear models might not be justified if the simpler models can capture the underlying phenomenon. Also the one-versus-rest strategy in these linear models seems to be quite good as the strategy does not add significant overhead, but is able to differentiate classes.

The best performance scores was in the linear SVM in all scores other than precision and computation time. The requirements for system was scalable and effective predictor for purchase invoices and as the linear SVM is simple and the F1-score is the highest it should meet criteria the best. The MLP might have better scores if hyper-parametrized correctly, but the scalability is a hindering factor, and logistic regression has the best precision, but is somewhat worse in other scores. The linear SVM has best qualities overall and the it will be chosen as the algorithm of choice for the next sections.

4.2 Performance of Linear SVM

In Figure 15 is presented the scores for labels trained with the linear SVM. The test accuracy is 0.81 while the macro average precision is 0.75, recall 0.76 and F1-score 0.74. The accuracy should be higher because the macro average scores take the

macro average whereas the accuracy is skewed by the classes with high amount of data points. The linear SVM gives preference, and higher scores, to classes such as ’71’, ’68’ and ’76’ where the amount of data points is larger (3.2). Generally this should be because in highly uneven labeling correctly classifying the dominant class and incorrectly classifying the smaller class can already give ’good’ performance scores for the amount of data points in the smaller class might be lesser significance.

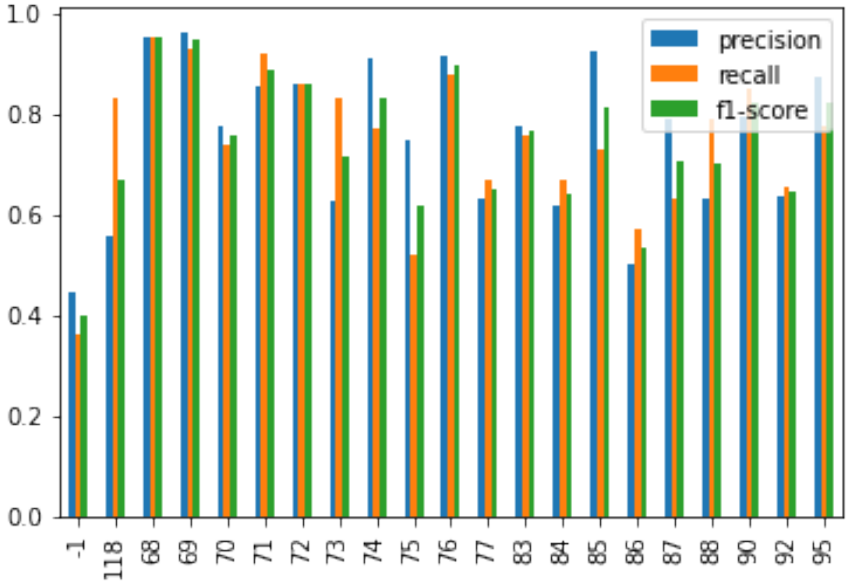

Figure 15: Scores for different labels for the Linear SVM. Test accuracy is 0.81, macro average precision is 0.75, recall 0.76 and F1-score is 0.74. Linear SVM with 10 000 data points of 90/10 train-test-split. Labels with less than 50 data point’s each grouped as label ’-1’.

The other sizeable classes ’88’, ’83’ and ’92’ have slightly worse performances and the largest id ’83’ has scores around 0.80. While bulkier classes should generally be highly scored, they might be lower in some cases because the decision boundary between the classes can be highly non-linear. The linear SVM also uses the ’one v. rest’ strategy in which even the larger classes can be outnumbered by the aggregate of all the other classes. The reasoning for the different performances might be explained by the lesser non-linearity in the decision boundary, or the clearer isolation of some classes. In other words these better performing classes might have words almost exclusively belonging to that class. This idea should be amplified by the fact that the ’trash bin’ class ’-1’ has the worst performance of all. That class is an aggregate of all the left over tiny classes an as such contains features which have little relevance to each other.

The Figure 16 presents a confusion matrix of the same classes. The main diagonal has the correctly labeled classes and show that the majority of instances belong to the main diagonal. Most importantly the figure shows the cases where true and predicted labels do not match. For instance the class ’83’ has many labeling confusion between the other classes. This might be indicative that the class ’83’ is more ill-defined as

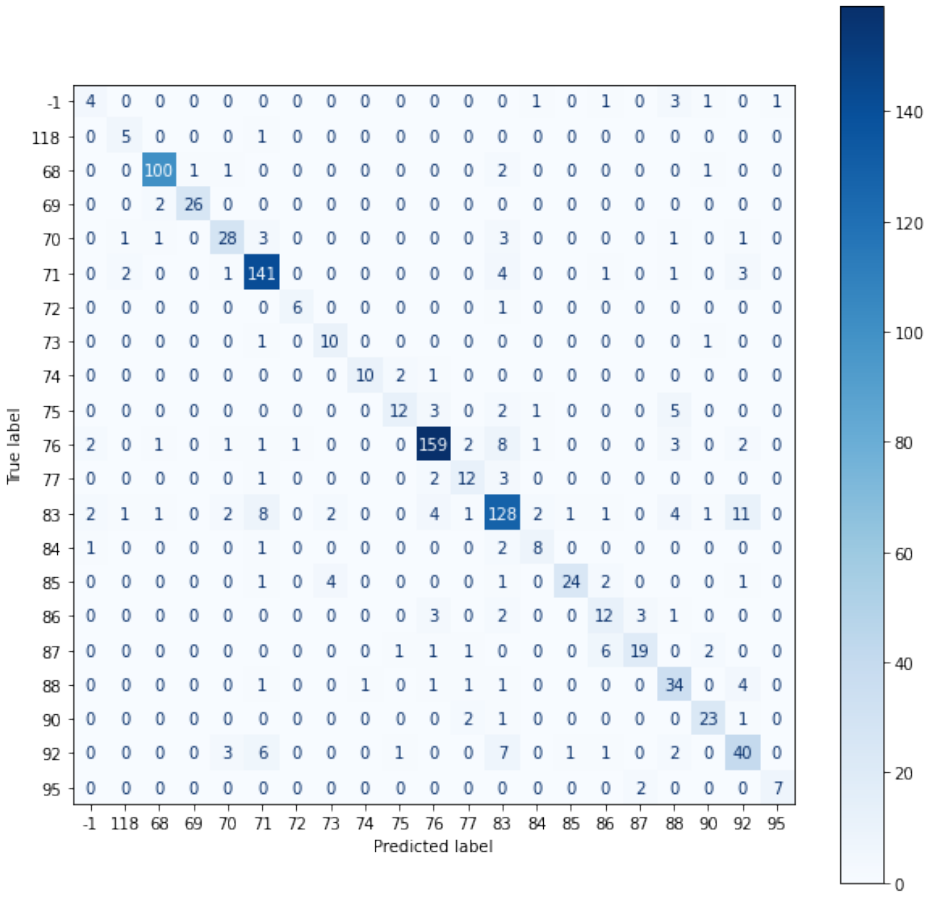

Figure 16: Confusion matrix plot. Linear SVM with 10 000 data points of 90/10 train-test-split. Labels with less than 50 data point's each grouped as label '-1'.

the other classes, i.e. the contained words intersect more with the other classes. This might be true also for the class '92'.

The Figure 17 presents a table in the Odoo interface of classes and their performances trained in an Odoo server. Even though the whole dataset had most instances in the class '83' the cutoff to only 10 000 data points made the class '76' the largest one in this training instance. Also for smaller classes the amount of test instances is quite low, which might be undesirable. The threshold for '-1' cutoff is 25 which results in 23 trained classes (apart from class '-1'). Nevertheless this training instance and test result show some improvements in the performance. The accuracy has risen to 0.85 and the macro average precision is 0.83, recall is 0.81 and F1-score is also 0.81.

In some instances the scores are perfect 1.00 although these only occur when the number of test data points are few in the class. The small amount of test data points in some classes might result in uncertain results even when stratification is used. However the (macro) averages should be more reliable as they aggregate many stochastic processes. Even these macro averages have some variation which have been empirically between 75% and 88% when the data is randomized.

Name	Precision Score	Recall Score	F1 Score	Support	Datapoints
-1	0.75	0.60	0.67	5	54
118	0.94	0.94	0.94	17	174
68	0.98	0.96	0.97	101	1,013
69	0.87	0.95	0.91	21	207
70	0.77	0.79	0.78	38	383
71	0.91	0.94	0.92	144	1,437
72	1.00	1.00	1.00	6	58
73	0.63	0.91	0.74	11	111
74	0.78	0.70	0.74	10	100
75	0.57	0.60	0.59	20	201
76	0.89	0.92	0.90	165	1,648
77	0.63	0.57	0.60	21	208
80	1.00	1.00	1.00	3	34
83	0.83	0.82	0.83	164	1,641
84	0.71	0.92	0.80	13	134
85	0.93	0.89	0.91	45	452
86	0.82	0.67	0.73	27	269
87	0.78	0.64	0.70	22	219
88	0.73	0.73	0.73	55	548
89	1.00	0.50	0.67	4	36
90	0.92	0.92	0.92	26	256
91	0.75	0.75	0.75	4	36
92	0.72	0.71	0.72	73	728
95	1.00	1.00	1.00	5	53
macro avg	0.83	0.81	0.81	1,000	0
weighted avg	0.85	0.85	0.85	1,000	0

Figure 17: A table of an average instance of improved linear SVM from Odoo interface. Parameters optimized with gridsearch. Gridsearch found an n-gram of 1 and 2 tokens preferable with TF-IDF and use a squared hinge loss function together with an one-v-rest strategy. Trained with 10000 datapoints with 10% of datapoints in test set. Accuracy 0.85. Support is the amount of data points in the test dataset. Labels with less than 25 data points grouped as label '-1', and the rest 23 labels trained.

One notable thing with the data in Figure 17 and with previous iterations is the different size cutoff of 25 and thus a larger number of trained classes. While generally it should worsen the performance of the model the scores indicate a better result. This might be due to these new smaller classes gaining a higher score while not being on poorly performing the overhead class '-1'. As said previously, the class

'-1' perform poorly due to it being an overlap of multiple 'real' classes and thus has a higher chance to meddle with other classes.

4.3 Issues

While the highest average scores might be as high as 88% this result should be taken with precaution. Apart from the time overhead to find such an arrangement, the generalization ability of that specific instance should be tested with an additional third data set, a validation set, with data uncoupled from the other data sets. This data could be gathered from the overall dataset outside of the 10 000 data points used in that instance. However the overall dataset (3.2) might contain some unwanted dependencies.

The cutoff to only use 10 000 might not be preferable if some other customers have a higher throughput of invoice lines. The original reason for this pruned set is to forget old data as new data is inputted to the system. In ML usually larger data sets perform better, especially if training an ANN (2.4). This should be true also for other models. The amount of test and train data points for every class might also be quite low as seen in the previous section which lowers the certainty of results.

In the Section 3.4 many of the generally accepted data preprocessing techniques were disregarded due insignificant performance improvements. While this might not be an issue, the possibility of finding improvements might be found from the disregarded techniques. Also a more extensive gridsearch might find some better combinations than the current parameters. Performance could be also improved if some new features not listed in Section 3.2 could be discovered, for example from the Finvoice XML-files.

5 Conclusions

The aim of this thesis was to implement a software system to classify purchase invoice accounting lines into analytic accounts. Requirements include the scalability and operability in multi-class classification setting. This software system aimed to be able to operate in the Odoo environment and outperform the accuracy of 90%.

The methodology was to research the current ML landscape to find suitable practices and compare the suitability of the ML models to meet the requirements. The accounting lines contain textual descriptions and as such required text data preprocessing and supplying this data through the models for comparisons. The most suitable model was manipulated even further to gain performance scores. Additionally an Odoo module was created for the model to work in that environment.

The effective model in the comparison was the linear SVM, which had a test accuracy of 0.81. It outperformed other models in all but precision and computation time. Logistic regression and MLP performed also well and have their own benefits and weaknesses. The MLP might be the most effective in terms of the scores after reasonable engineering work, although the computation time to optimize the model excluded it to be selected for further study in this thesis. After additional work on the selected model, linear SVM, the most effective instance reached 88% accuracy and weighted average scores.

While the presented accuracy of 88% is close to the goal of 90% it should be taken with precaution, as it is a result of randomization of test and train data sets. A more robust methodology would take an additional validation set to validate the generalization ability. Nevertheless, the result of 88% is close to the initial goal of this thesis.

Any future work should investigate more thoroughly the data preprocessing methodology. Also a more advanced system also could train multiple instances of a model and select the most effective model with a validation set. Also a research towards an effective MLP model is intriguing.

References

[1] M. S. Schaeffer, Institute of Management and Administration (IOMA), O. M. a. A. Institute of Management and Administrati, and Lastinstitute of Management and Administration (Ioma), *Accounts Payable: A Guide to Running an Efficient Department*. Hoboken, UNITED STATES: John Wiley & Sons, Incorporated, 2004.

[2] S. Lahti and T. Salminen, *Digitaalinen taloushallinto*. Talentum, 2014.

[3] S. M. Bragg, *Billing and Collections Best Practices*. Hoboken, UNITED STATES: John Wiley & Sons, Incorporated, 2004.

[4] C. C. Aggarwal and C. Zhai, "A Survey of Text Classification Algorithms," in *Mining Text Data* (C. C. Aggarwal and C. Zhai, eds.), pp. 163–222, Boston, MA: Springer US, 2012.

[5] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text Classification Algorithms: A Survey," *Information*, vol. 10, p. 150, Apr. 2019. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[6] C. Bardelli, A. Rondinelli, R. Vecchio, and S. Figini, "Automatic Electronic Invoice Classification Using Machine Learning Models," *Machine Learning and Knowledge Extraction*, vol. 2, pp. 617–629, Dec. 2020. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[7] J. Taussi, "Utilizing machine learning in purchase invoice posting," Oct. 2020. Accepted: 2020-10-25T18:10:52Z.

[8] A. S. Tarawneh, A. B. Hassanat, D. Chetverikov, I. Lendak, and C. Verma, "Invoice Classification Using Deep Features and Machine Learning Techniques," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pp. 855–859, Apr. 2019.

[9] S. Bragg, *Bookkeeping Essentials: How to Succeed as a Bookkeeper*. Wiley, 2011.

[10] V. Curtis, *Bookkeeping for Dummies, 3rd Edition*. For Dummies, 2020.

[11] E. Penttinen, K. Dorota, and Helsingin Kauppakorkeakoulu, *Electronic invoicing initiatives in Finland and in the European Union: taking the steps towards the real-time economy*. Helsinki: Helsingin Kauppakorkeakoulu, 2008. OCLC: 424523195.

[12] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT Press, second edition ed., 2018.

[13] M. Mohammed, *Machine Learning*. CRC Press, 2016.

[14] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, 2014.

[15] A. Jung, *Machine Learning: The Basics*. Singapore: Springer, 2022.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*. MIT Press, Nov. 2018. Google-Books-ID: uWV0DwAAQBAJ.

[17] G. Rebala, A. Ravi, and S. Churiwala, *An Introduction to Machine Learning*. Cham: Springer International Publishing, 2019.

[18] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Springer Science & Business Media, Dec. 2012. Google-Books-ID: aaDbBwAAQBAJ.

[19] Z. Ghahramani, "Unsupervised Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures* (O. Bousquet, U. von Luxburg, and G. Rätsch, eds.), pp. 72–112, Berlin, Heidelberg: Springer, 2004.

[20] H. K. Jabbar and R. Z. Khan, "Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study)," in *Computer Science, Communication and Instrumentation Devices*, pp. 163–172, Research Publishing Services, 2014.

[21] D. Maulud and A. Abdulazeez, "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, no. 4, pp. 140–147, 2020.

[22] V. Vapnik, "Principles of Risk Minimization for Learning Theory," in *Advances in Neural Information Processing Systems*, vol. 4, Morgan-Kaufmann, 1991.

[23] X. Zhang, "Structural Risk Minimization," in *Encyclopedia of Machine Learning* (C. Sammut and G. I. Webb, eds.), pp. 929–930, Boston, MA: Springer US, 2010.

[24] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," Nov. 2020. arXiv:1811.12808 [cs, stat].

[25] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, Nov. 2006.

[26] F. Herrera, F. Charte, A. J. Rivera, and M. J. del Jesus, *Multilabel Classification*. Cham: Springer International Publishing, 2016.

[27] J. E. T. Akinsola, "Supervised Machine Learning Algorithms: Classification and Comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, pp. 128–138, June 2017.

[28] M. Aly, "Survey on Multiclass Classification Methods," 2005.

[29] D. Tax and R. Duin, "Using two-class classifiers for multiclass classification," in *2002 International Conference on Pattern Recognition*, vol. 2, pp. 124–127 vol.2, Aug. 2002. ISSN: 1051-4651.

[30] A. Mandelbaum and D. Weinshall, "Distance-based Confidence Score for Neural Network Classifiers," Tech. Rep. arXiv:1709.09844, arXiv, Sept. 2017. arXiv:1709.09844 [cs, stat] type: article.

[31] C. C. Aggarwal and C. Zhai, eds., *Mining Text Data*. Boston, MA: Springer US, 2012.

[32] V. Korde, "Text Classification and Classifiers:A Survey," *International Journal of Artificial Intelligence & Applications*, vol. 3, pp. 85–99, Mar. 2012.

[33] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing & Management*, vol. 50, pp. 104–112, Jan. 2014.

[34] E. Ikonomakis, S. Kotsiantis, and V. Tampakas, "Text Classification Using Machine Learning Techniques," *WSEAS transactions on computers*, vol. 4, pp. 966–974, Aug. 2005.

[35] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents," in *Proceedings of the Thirteenth ACM conference on Information and knowledge management - CIKM '04*, (Washington, D.C., USA), p. 625, ACM Press, 2004.

[36] A. G. Jivani, "A Comparative Study of Stemming Algorithms," *Int. J. Comp. Tech. Appl*, vol. 2, p. 9, 2011.

[37] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, pp. 45–65, Jan. 2003.

[38] J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242.

[39] R. Dulek, "Properties of the Hypothesis Space and their Effect on Machine Learning," p. 23, 2013.

[40] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, July 2018.

[41] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," Mar. 2014. arXiv:1403.2877 [cs, q-bio, stat].

[42] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Sept. 2015. ISSN: 2378-928X.

[43] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International Journal of Remote Sensing*, vol. 28, pp. 823–870, Mar. 2007. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01431160600746456.

[44] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1310–1315, Mar. 2016.

[45] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A Comprehensive Survey of Loss Functions in Machine Learning," *Annals of Data Science*, vol. 9, pp. 187–212, Apr. 2022.

[46] T. Nguyen and S. Sanner, "Algorithms for Direct 0–1 Loss Optimization in Binary Classification," in *Proceedings of the 30th International Conference on Machine Learning*, pp. 1085–1093, PMLR, May 2013. ISSN: 1938-7228.

[47] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004. Google-Books-ID: mYm0bLd3fcoC.

[48] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, pp. 427–437, July 2009.

[49] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, pp. 27–38, Jan. 2009.

[50] D. M. W. Powers, "What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes," Tech. Rep. arXiv:1503.06410, arXiv, Sept. 2019. arXiv:1503.06410 [cs, stat] type: article.

[51] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss," *Machine Learning*, vol. 29, pp. 103–130, Nov. 1997.

[52] H. Zhang, "The Optimality of Naive Bayes," *Aa*, p. 6, 2004.

[53] I. Rish, "An empirical study of the naive Bayes classifier," *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, p. 6, 2001.

[54] C.-Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An Introduction to Logistic Regression Analysis and Reporting," *The Journal of Educational Research*, vol. 96, pp. 3–14, Sept. 2002.

[55] J. C. Stoltzfus, "Logistic Regression: A Brief Primer," *Academic Emergency Medicine*, vol. 18, no. 10, pp. 1099–1104, 2011. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1553-2712.2011.01185.x.

[56] T. Fletcher, "Support Vector Machines Explained," p. 20, 2009.

[57] D. Boswell, "Introduction to Support Vector Machines," *Departement of Computer Science and Engineering University of California San Diego*, p. 15, 2002.

[58] B. Schölkopf, "The Kernel Trick for Distances," in *Advances in Neural Information Processing Systems*, vol. 13, MIT Press, 2000.

[59] A. Patle and D. S. Chouhan, "SVM kernel functions for classification," in *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pp. 1–9, Jan. 2013.

[60] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of Improving K-Nearest-Neighbor for Classification," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, vol. 1, pp. 679–683, Aug. 2007.

[61] Z. Yong, L. Youwen, and X. Shixiong, "An Improved kNN Text Classification Algorithm based on Clustering," *Journal of Computers*, 2009.

[62] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Nature Biotechnology*, vol. 26, pp. 1011–1013, Sept. 2008. Number: 9 Publisher: Nature Publishing Group.

[63] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, pp. 261–283, Apr. 2013.

[64] J. Quinlan, "Decision trees and decision-making," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 339–346, Mar. 1990. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[65] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1249.

[66] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[67] L. N. Kanal, "Perceptron," in *Encyclopedia of Computer Science*, pp. 1383–1385, GBR: John Wiley and Sons Ltd., Jan. 2003.

[68] A. Abraham, "Artificial neural networks," *Handbook of measuring system design*, 2005.

[69] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, pp. 2627–2636, Aug. 1998.

[70] J. Neuvonen, "Developing purchase invoice processing through workflow automation," 2015. Accepted: 2015-02-16T11:18:42Z.

[71] T. Tolmala, "Ostolaskujen käsittelyn automatisointi," p. 44, 2020.

[72] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.