PREDICTING LATE PAYMENT OF SALES INVOICES WITH STATISTICAL LEARNING METHODS

Lappeenranta-Lahti University of Technology LUT

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

2023

Matti Martikainen

Examiners: Professor Lasse Lensu
M.Sc. (Tech.) Pauli Immonen

ABSTRACT

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

Matti Martikainen

Predicting late payment of sales invoices with statistical learning methods

Master's thesis
2023
45 pages, 11 figures, 8 tables, 2 appendices
Examiners: Professor Lasse Lensu and M.Sc. (Tech.) Pauli Immonen

Keywords: statistical learning, gradient boosting, ensemble learning, machine learning, data science, enterprise resource planning, cash flow forecasting, liquidity management

Predicting the time at which a company is bound to receive money from their outstanding sales invoices is a part of cash flow forecasting, which helps companies make better financial decisions. This study explores the feasibility of utilizing statistical learning methods to predict late payments of sales invoices in advance, using a data set obtained from a Finnish Enterprise Resource Planning (ERP) software product. Based on past studies on the topic, several ensemble learning classifier models were evaluated for the task. In addition, a naive benchmark classifier was proposed, which based its prediction entirely on the classes of previous invoices from the same customer, prioritizing their most recent invoices with exponential discounting. The evaluated models narrowly outperformed this benchmark on some experiments, while in others the benchmark performed slightly better. The performance difference across the evaluated models was found to be relatively small in any given task. Prior information from previous invoices paid by the same customer was found to be the most significant contributor to the predictions.

TIIVISTELMÄ

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

Matti Martikainen

Myyntilaskujen myöhässä maksamisen ennustaminen tilastollisen oppimisen menetelmillä

Diplomityö
2023
45 sivua, 11 kuvaa, 8 taulukkoa, 2 liitettä
Tarkastajat: Professor Lasse Lensu ja M.Sc. (Tech.) Pauli Immonen

Hakusanat: tilastollinen oppiminen, gradienttitehostaminen, kokoonpano-oppiminen, koneoppiminen, datatiede, toiminnanohjaus, kassavirtaennustaminen, likviditeetin hallinta

Avointen myyntisaatavien maksuajan ennustaminen on osa kassavirtaennustamista, joka auttaa yrityksiä tekemään parempia taloudellisia päätöksiä. Tämä tutkimus selvittää, kuinka hyvin suomalaisesta ERP-järjestelmästä kerättyjen avointen myyntilaskujen maksupäivää voitaisiin ennustaa statistisen oppimisen menetelmiä hyödyntäen. Aiempiin tutkimuksiin perustuen arvioitavaksi valittiin useita kokoonpano-oppimiseen (*ensemble learning*) perustuvia luokittelumalleja. Lisäksi kehitettiin naiivi vertailumenetelmä, jonka ennuste perustui ainoastaan saman asiakkaan aiempien laskujen luokituksiin, painottaen lähiaikoina maksettuja laskuja eksponentiaalisella diskonttauksella. Arvioitavat mallit suoriutuivat osassa kokeista hieman vertailumenetelmää paremmin, mutta toisissa heikommin. Keskinäiset erot mallien suoriutumisessa olivat kaiken kaikkiaan melko pieniä. Arvioitujen mallien luokittelutulokseen vaikutti eniten asiakkaan laskutushistoriasta koottu tieto asiakkaan aiemmasta maksukäyttäytymisestä.

ACKNOWLEDGEMENTS

I would like to humbly thank my supervisors Lasse Lensu and Pauli Immonen for the continuous, insightful guidance and encouraging feedback, providing fresh points of view and nudging me into the right direction whenever I felt stuck. I am also very grateful towards my employer for their years-long flexibility, allowing me an opportunity to pursue this thesis and to continuously work on interesting, practical problems alongside my studies. I also want to mention my co-workers for the companionship and camaraderie throughout the years, with a special thanks to Matti Siltanen for coming up with the final research topic which narrowed the initial scope of the project down to a manageable level.

Finally, I would like to express my utmost gratitude towards my friends and family. Being able to share the ups and downs of this journey with others has been very important to me. Thanks to Mom and Dad for raising me with warmth and care and continuously encouraging me to pursue my interests.

Lappeenranta, July 26, 2023

Matti Martikainen

LIST OF ABBREVIATIONS

AR	Accounts receivable
AUC	Area under the Receiver Operating Characteristic curve
ERP	Enterprise Resource Planning
IG	Information Gain
LIME	Local Interpretable Model-agnostic Explanations
SHAP	Shapley Additive Explanations
SME	Small and medium-sized enterprises
SMOTE	Synthetic Minority Oversampling Technique

CONTENTS

1 INTRODUCTION

1.1 Background

One of the major challenges in managing the finances of a growing business comes from maintaining sufficient liquidity while investing into growth [1]. Liquidity refers to a company's ability to pay its short-term operating expenses, such as rent, salaries, debts, insurances and costs of goods sold. To meet these obligations, the business must always maintain a sufficient supply of cash and/or assets that are easily convertible to cash as needed, referred to as liquid assets. However, excess liquid assets typically provide little financial value to the company compared to fixed assets: longer-term investments that cannot be as easily converted back to cash, such as property, buildings, equipment and machinery. Therefore, it would be ideal to maximize the amount of money invested, while always keeping just enough liquid assets on hand to be able to meet all expected and unexpected financial demands.

Cash flow forecasting is a process used to predict the incoming and outgoing cash flows of a business over a period of time [2]. It can help a company make better financial decisions: if a potential cash shortage is detected ahead of time, the company can make adjustments such as borrowing additional money or selling off assets. Conversely, during times of expected cash surplus, more money can be allocated into investments or paying off debts. Being able to accurately predict future cash flow could thus bring a large competitive advantage for a business, but the task is quite challenging due to a number of factors. Overall cash flow is made up of many individual incoming and outgoing streams which vary greatly across lines of businesses, making modeling all of them potentially very complicated, and overall there is always an inherent degree of uncertainty in predicting future events.

Accounts receivable (AR) refers to the money owed to a business from goods or services that have been sold and invoiced, but not yet paid by the customer. It is a major stream of incoming cash flow for many businesses. Forecasting cash flow from accounts receivable requires modeling these customers' payment behavior on a large scale to approximate the date at which each outstanding invoice will be paid. While it is clear that this behavior can be affected by variable factors that are impossible to model accurately, it is reasonable to assume that some patterns are present. For instance, in business-to-business trade the buyer generally prefers to delay payments as long as possible to maximize their own cash reserves. This dynamic is evident from the popularity of various supply chain financing

arrangements: sellers may provide motivation for early payments by offering discounts, offer credit to the buyer to facilitate longer payment terms, or utilize third party financing to arrange said credit (reverse factoring) [3, 4]. Without such arrangements in place, it would be reasonable to assume that invoices are not often paid before the due date. Some customers may also delay payments past the due date of the invoice due to cash shortages or other reasons, and in some cases the late payments could potentially form a predictable pattern.

A collection of recent studies [5–11] have attempted to model these patterns using statistical learning techniques, which have been an increasingly popular approach in finding patterns in complicated data. This prior work has mainly focused on classification of sales invoices to predict whether their payment will be on time, or delayed from the expected due date. In some cases, multiple-class classification has been used to further categorize the delayed invoices into multiple categories depending on the amount of lateness (1-14 days late, 15-30 days late, etc.). While such date ranges inherently leave a degree of uncertainty within the model's predictions, it is still a promising start to a very complicated problem.

The work presented in this thesis aims to to improve cash flow forecasting within a Finnish enterprise resource planning (ERP) system targeted towards small-to-medium enterprises (SME). As the problem space is quite broad and complex for the purposes of a master's thesis, the scope of the project was narrowed to specifically improving accounts receivable (AR) forecasting by making predictions on the expected payment dates of outstanding sales invoices, building upon the mentioned studies. In addition to improving cash flow forecasting, these predictions could also help companies manage the financial risks associated with individual large invoices and take more proactive measures with invoices that are expected to be paid late.

As the ERP system in question is operating primarily in the Finnish SME market, a large portion of its users' customers are also Finnish businesses. Some extra information about these customers can be acquired through an open data service provided by the Finnish Patent and Registration Office. The main areas of interest within the available data include each company's main line of business, as well as information about ongoing and past bankruptcies, liquidations, or restructuring proceedings. These data could potentially be used to acquire some additional insight about the customer, especially if information about their prior payment behavior is not yet available.

1.2 Objectives and delimitations

The main objectives of the thesis can be summarized with the following research questions:

- To what degree is it possible to predict delayed payments of sales invoices ahead of time, utilizing data in the ERP system?
- Which factors are the most relevant predictors for payment delays?
- What is the relevance of openly available business data for improving the prediction accuracy?

The aim of the work is to find a suitable method for prediction of payment delays within the available data. If such a method is found, it could be used to provide warnings about specific high-risk invoices in advance and/or improve cash flow estimation on a broader scale.

As the work is done in collaboration with an accounting software business, it would be preferable to arrive at a solution that can provide immediate value to the users of the software. As such, it is likely that this study will focus on exploring whether results obtained in prior studies done on the subject can be reproduced with the available data. More novel approaches can then be experimented with in later studies if the results from this one seem promising.

1.3 Structure of the thesis

Chapter 2 presents a short summary of machine learning fundamentals and some classification techniques relevant in the later chapters. Chapter 3 explores prior work done on the prediction of sales invoice payment delays. The methods chosen for the study are then introduced in further detail in Chapter 4. A description of available data and experiments performed on the data is shown in Chapter 5. The results of the experiments are discussed in Chapter 6, and conclusions made from the experiments are presented in Chapter 7.

2 STATISTICAL LEARNING

2.1 Supervised learning

In a typical statistical learning problem, a mathematical model is used to predict one or more output variables from a set of input variables (also referred to as *features*). As an example, one might want to predict the sale price of a house on the basis of some known information, such as its size, number of rooms and year of construction. The parameters of the chosen statistical model are optimized against a *training set* consisting of several example data points, with the goal of arriving at a model that can make accurate predictions not only on this training set, but also on new unseen data points. The ability to generalize predictions to unseen data is a crucial trait for the model to be useful in a real-world setting. [12]

Supervised learning is one of the main paradigms of statistical learning. In supervised learning, each data sample in the training set also contains the output variable to be predicted, referred to as a *target variable*. The training process aims to minimize the error between these known labels and the model's predictions across the training set. Conventionally, a portion of training data is set aside as a *test set*, which is not used for training the model. The test set is used after the training process to validate the model's performance, i.e. calculate the error between true labels and ones predicted by the model using data that was not available to it during the training process. For a more robust measure of model performance, this process of training and validating the model can be repeated multiple times in a technique called cross-validation, where a different portion of the data is selected as the validation set for each iteration and the average performance of all trained models is chosen as the final result. [12]

If the variable to be predicted is categorical, meaning all possible outcomes can be defined as a finite set of categories or classes (for example, an email could be categorized as either *spam* or *not spam*), the prediction task is referred to as *classification*. A trained classification model splits the input space into distinct regions that each represent one class based on the class distribution present in the training data, using one or more *decision boundaries* (see Figure 1). Predictions on new data samples can then be obtained by checking which region the sample falls into. Variables can also be continuous (i.e. the aforementioned house sale price), in which case predicting an outcome is referred to as *regression*. [12]

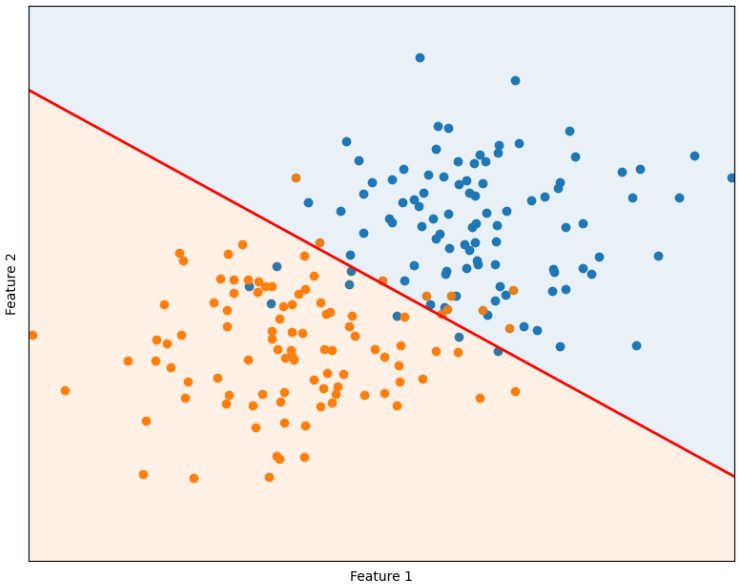

Figure 1. Example of a two-class classification problem. The colors of the dots represent the true class distribution within the training data. A logistic regression model is used to fit a linear decision boundary (drawn in red) to the data, splitting the feature space into two regions which can be used to classify future data samples.

Many statistical learning models have been proposed over the years, and no model has been found to be clearly superior to all others. It is common practice to test many of these models on any given set of data to find which ones are most suited for the task. The following chapters focus on presenting the theory behind decision trees, which form the basis for methods that have seemingly performed well in the task of invoice payment delay classification, as will be shown in Chapter 3. The source material covers many more statistical learning approaches in more detail.

2.2 Decision trees

Decision trees are simple statistical prediction models that can be fit to a set of data with a supervised learning approach. The model begins from a single *decision rule*, in the simplest form presented as a yes/no question which can be used to split the full set of training data into two subsets. Usually the decision rule is a numerical threshold on one of the input variables, creating two subsets: if the value of the chosen variable is above the threshold, the sample belongs to the first subset, and if not, it belongs to the second one. The decision rule is selected with the goal of increasing *purity* within each subset. A pure

subset contains homogeneous values of the variable to be predicted (i.e. only examples of a single class). [13, 14]

Impure subsets can be split repeatedly with additional decision rules, until they either become pure or some other stopping condition is reached. The value contained in each of these outcome sets is chosen as a prediction outcome for each new data point that matches the decision rules leading to it. The resulting set of rules and outcomes can be visualized as a tree graph, which is followed from the topmost decision node downwards until an outcome node is reached (see Figure 2). [13]

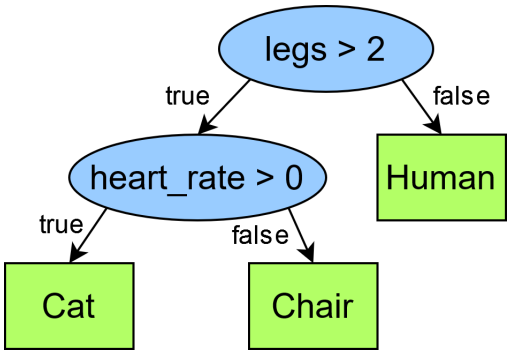

Figure 2. A basic example of a classification decision tree. Two decision rules are used to assign one of three class labels to the data.

The threshold value chosen for each decision split is selected to maximize Information Gain (IG), which is a metric measuring how well the split separates classes from one another [13]. It is defined as

$$IG(S_1, S_2) = I(S) - \frac{n_1}{n} I(S_1) - \frac{n_2}{n} I(S_2) \tag{1}$$

where S is the original set, S_1 and S_2 are subsets created by the split, n, n_1 and n_2 are the number of samples contained in each respective set, and $I(x)$ is an *impurity function*, measuring class mixing within a set. A commonly used impurity function for classification trees is the Gini index, defined as

$$I_g(S) = \Sigma p_i(1 - p_i) \tag{2}$$

where p_i is the fraction of samples within a set S that belong in class i. In summary, the information gain is highest when the combined proportional impurity of the subsets is the lowest in comparison to the original set. The impurity of a set decreases as the fraction of samples belonging to a single class increases.

Decision splits can be created repeatedly until the tree can accurately classify all examples in the training set, but typically this leads to an overfit model that performs poorly on unseen data [12, 13]. Trying to account for every outlier found in the initial training set tends to create overly specific decision boundaries that fail to generalize well. Two main regularization techniques exist to counteract this: the tree can either be grown to full size first, after which some leaves are pruned, or the growth can be stopped early.

Decision trees are popular due to their ease of interpretability, ability to handle a mixture of categorical and continuous input variables, robustness to outliers and low computational complexity, among other factors. However, their prediction accuracy is often relatively poor in comparison to other approaches, and they are known to be *unstable*, meaning that small changes in input data can result in large changes in the tree structure. [14]

2.3 Ensemble models

An ensemble model arrives at a prediction by combining the results of multiple individual predictive models [12, 15]. Typically the base models used within an ensemble are simple models such as decision trees, due to their low computational complexity. Ensemble-based learners have gained popularity in recent years and are generally capable of achieving a higher predictive accuracy than would be possible with any single model contained within the ensemble [15, 16]. They have also been noted to be a popular solution for handling data with unbalanced classes [17].

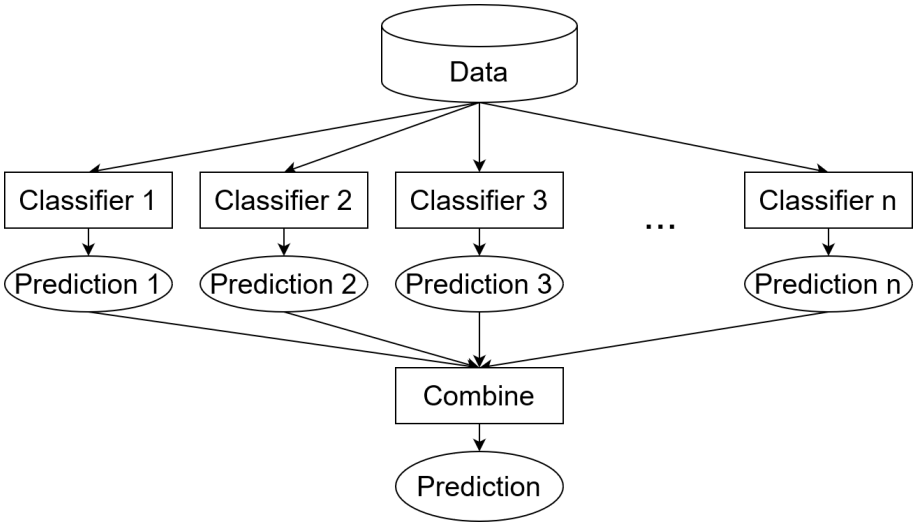

Figure 3. A typical ensemble model architecture.

The increased predictive power of an ensemble model results from disagreement among the base learners, as clearly there is no benefit to be gained from combining predictions of several models that produce identical results [15]. This disagreement can be achieved by introducing variation into the training process. *Bagging* (short for *bootstrap aggregation*) and *boosting* are two popular methods for creating such variation by altering the training data shown to each learner.

In bagging, each learner is trained on a randomized subset of the full data where samples can be selected multiple times (sampling with replacement). This aims to create individual learners with a specialized understanding of different portions of the training data. In boosting, the learners are trained sequentially such that erroneously predicted samples from the previous learners are weighted to be selected more often for the next training sets. In essence, this supplements the ensemble with models specialized in understanding the specific portions of the data that it does not yet understand well. [12, 15]

3 PREDICTING INVOICE PAYMENT DELAYS

3.1 Delayed payment prediction with classification techniques

Research into prediction of late invoice payments does not seem particularly active, but a handful of prior studies on the topic have been conducted in recent years. In prior work, statistical classification has been the most common approach of tackling the problem. In most studies the problem was treated strictly as a binary classification task, where only two classes were used: an invoice was either paid on time, or paid late. Some studies also utilized multiple-class classification, where the late invoices were further divided into several smaller categories (for example: 1-30 days, 31-60 days, and 61-90 days, and 90+ days late). In studies where the results of both approaches were reported [6,11], the multi-class approach resulted in a noticeable decrease in overall prediction accuracy. It has not been discussed whether the increased granularity of predictions warrants this trade-off, but supposedly this depends on one's business requirements for the task. Typically the invoices with the longest expected delays were viewed as the most critical ones to classify accurately from a business standpoint [5].

While each of these studies was conducted on a different set of data, a common finding across many has been that the data is heavily unbalanced. Typically, most invoices across each data set were paid on time, and in multiple-class classification cases the class with the highest amount of delay was unilaterally a small minority. This presents a major challenge for classification, as classifiers tend to ignore the minority class when optimizing for overall accuracy. In some studies classification performance was evaluated with metrics other than overall accuracy, such as accuracy for the minority class only [5], F1-score [8–11] and/or area under the Receiver Operating Characteristic curve (AUC) [9, 10].

Table 1 summarizes the methodology across previous studies on the topic where classification techniques were used, noting whether binary or multi-class classification was used, the best performing classifier according to the performance metric chosen in the study, and techniques chosen to counteract the class imbalance. While each study has been conducted on a different set of data, making it difficult to compare approaches between studies or draw conclusions about their effectiveness, some clear trends can be seen. The following subsections in this chapter further cover a brief introduction into the most commonly appearing techniques across the studies found.

Table 1. Methodology used in previous studies on invoice payment delay prediction utilizing statistical classification.

Study	Year	Problem type	Selected classifier	Class imbalance mitigation
[5]	2008	5-class	C4.5	Cost-sensitive learning
[6]	2013	Binary, 4-class	Random forest	Cost-sensitive learning
[7]	2014	Binary	Random forest	Cost-sensitive learning
[8]	2018	5-class	LightGBM	Cost-sensitive learning
[9]	2019	Binary	Random forest	Oversampling
[10]	2019	Binary	Ensemble (Random forest + XGBoost)	None
[11]	2021	Binary, 3-class, 5-class	LightGBM	Synthetic Minority Oversampling Technique (SMOTE), undersampling

3.1.1 Classifier performance

According to Table 1, decision tree-based algorithms have unilaterally performed the best on the performance metrics measured in each study. C4.5 is an algorithm that is used to construct a single decision tree, while random forests and the various gradient boosting methods (XGBoost, LightGBM) used in the later studies are ensemble classifiers based on a combined prediction obtained from a large number decision trees.

From a practical standpoint, a major difference between utilizing random forests and gradient boosting methods is the importance of hyperparameter tuning. A comparative analysis on recent gradient boosting algorithms concluded that while random forests have relatively few parameters to tune and perform notably well with the default settings, the boosting approaches require extensive tuning to create accurate models [18]. Across 28 datasets, boosting algorithms were found to perform slightly better than random forests on average with hyperparameter tuning, but slightly worse when default parameters were used. The authors noted that hyperparameter tuning utilizing a grid search took up over 99.9% of computational effort for training, and that the time required to find a reasonable set of hyperparameters must be taken into consideration when measuring the training time of a model.

3.1.2 Accounting for class imbalance

Across the studies presented in Table 1, most authors recognized the class imbalance present in their respective datasets and the challenges it presented for classifying the minority classes accurately. In general, some basic strategies well-known to improve learner performance in class-imbalanced tasks include:

- **Random undersampling**: samples from majority classes are randomly excluded until there is an equal number of samples in each class.
- **Random oversampling**: samples from minority classes are randomly duplicated until there is an equal amount of each class.
- **Synthetic Minority Oversampling Technique (SMOTE)**: synthetic data points with minor variations are generated from minority classes until there is an equal amount of each class.
- **Cost-sensitive learning**: the cost of misclassifying minority classes is increased in the learning algorithm, often defined as a cost matrix.

The most often utilized technique for handling class imbalance across these studies was cost-sensitive learning. In the earliest of the studies [5], the cost of misclassification was increased for only the class of invoices paid more than 90 days late, which were noted to be the most critical from a business perspective. Three different cost matrices with increasingly large costs of misclassification for the 90+ class were tested, using weights that were seemingly hand-selected by the authors, and it was found that higher costs increased classification accuracy for the targeted class at the expense of overall accuracy. The authors noted that desired performance objectives for the classifier can be used to balance the right trade-off between these two factors.

Another study [6] experimented with both cost-sensitive learning and random undersampling (implemented in the Balanced Random Forests classifier). Both approaches were found to improve classification accuracy in the minority class, called "long delay" in this study, at the cost of overall accuracy. As shown in Table 2, undersampling with Balanced Random Forests resulted in a more drastic increase in the accuracy of the "long delay" class, but also a large decrease in overall accuracy, while cost-sensitive learning achieved a more modest, yet still sizeable improvement in the minority class with very little impact in overall accuracy.

Table 2. Comparison of the effects of undersampling and cost-sensitive learning on classification accuracy in the same study [6].

	Baseline	**Undersampling**	**Cost-sensitive learning**
Overall	81.6%	61.0%	81.2%
"Long delay" class	51.9%	83.5%	61.9%

To explore the difference between resampling methods and cost-sensitive learning on a broader scale, a comparative analysis concluded that both approaches improved performance over the baseline across classification tasks performed on 66 datasets using AUC as the performance metric [19]. However, neither approach was found to perform clearly better than the other. In a literature review analyzing classification techniques with class-imbalanced data across 527 articles, cost-sensitive learning was found to be far less popular compared to resampling [17]. The authors noted that the implementation of cost-sensitive learning requires expert knowledge to define the weights of the cost matrix and sometimes modification of the learning algorithm, which could be a cause for its relative unpopularity.

3.1.3 Role of feature engineering

Across multiple studies, it has been found that features based on customer-level payment history provide a noticeable increase in classification accuracy over using features inferred from a single invoice only. Table 3 presents a majority of the features used across the studies. Commonly used customer level features include the mean payment delay of previous invoices in days, the number of past late invoices, the monetary sum of past late invoices, and the ratio of late invoices versus paid-on-time invoices. In studies where the influence of individual features has been considered, these features have generally showed up among the most important features. [6, 7, 11].

Typical invoice-level features included the monetary sum of the invoice and the payment term, i.e. the number of days from the date of the invoice to the due date. The payment term was found to be one of the most valuable features in one study [11]. Multiple studies also utilized a binary feature indicating whether the due date of an invoice is in the last three days of the month, originally proposed in [6], where the author noted that almost 30% of late invoices in their dataset had their due date within this three-day window. They theorized that businesses typically need to pay their salaries during this time, which could

Table 3. Summary of features used in previous studies on invoice payment delay prediction. Studies are presented in chronological order.

Feature		[5]	[6]	[7]	[9]	[10]	[11]
1.	Invoiced amount	x	x	x	x		x
2.	Average invoiced amount over all prior invoices			x	x		x
3.	Payment term	x			x		x
4.	Number of prior invoices	x	x	x	x	x	x
5.	Number of prior invoices paid late	x	x	x	x	x	x
6.	Ratio of Feature 5 over Feature 4	x	x	x	x		x
7.	Total invoice amount of prior invoices	x	x	x	x	x	x
8.	Total invoice amount of prior invoices paid late	x	x	x	x	x	x
9.	Ratio of Feature 8 over Feature 7	x	x	x	x		x
10.	Average delay of prior late invoices	x	x	x			x
11.	Average delay of all prior invoices		x		x	x	
12.	Number of currently outstanding invoices	x			x	x	x
13.	Number of currently outstanding invoices that are late	x				x	x
14.	Ratio of Feature 13 over Feature 12						x
15.	Total invoice amount of currently outstanding invoices	x			x	x	x
16.	Total invoice amount of currently outstanding invoices that are late	x				x	x
17.	Ratio of Feature 16 over Feature 15						x
18.	Due date is within the last three days of the month		x	x			x
19.	Due date is within the latter half of the month		x	x			
20.	Average payment term of prior invoices		x	x			x
21.	Sales representative of the invoice		x				

potentially lead to short-term liquidity challenges. However, in later studies where this feature was used, it ranked quite low on feature importance scores [7, 11].

In addition to using customer and invoice level features, one study [9] embedded invoicing history between all businesses in their data set into a graph utilizing the node2vec [20] framework. They proposed that especially in cases where no invoicing history has been established with the customer directly, some indication of their payment behavior could be inferred from their invoicing history with other businesses in the graph. Utilizing features from the payment graph at the node level (i.e. both parties' payment history towards all of their customers, not just each other) resulted in some increases in accuracy over the baseline classifier that only utilized invoice-level features and history data between the parties.

3.2 Other approaches for invoice payment delay prediction and related topics

One related study [21] approached late invoice classification from the accounts payable perspective, trying to detect invoices that a large accounts payable department would most likely have trouble paying on time. Similarly to the accounts receivable case, many of the most important predictors for a delayed invoice were related to whether prior invoices from the vendor had been delayed.

Instead of classifying invoices into date range categories based on the degree of lateness, one study instead used a survival analysis model to calculate the cumulative probability of an invoice being paid by a given date [22]. The author concluded that the machine learning based Random Survival Forest model outperformed the more traditional Cox Proportional Hazards model for the task, but both models were able to rank payment times to an acceptable degree.

Some parallels from late invoice detection can also be drawn to the much better researched problem of bankruptcy prediction or credit scoring. In credit scoring, various statistical models are used by financial institutions to decide upon the creditworthiness of a borrower. A large literature survey [23] reviewed various machine learning approaches that have been researched as a possible replacement to the logistic regression model, which is traditionally used in the industry due to its transparency (a regulatory requirement for decisions made by these institutions) and low complexity. The study concluded that ensemble classifiers and convolutional neural networks generally outperformed other mod-

els across the studies reviewed, and suggested Local Interpretable Model-agnostic Explanations (LIME) [24] as a method for providing the required transparency into their decisions.

3.3 Summary

Based on the previous studies on invoice payment delay prediction, a classification-based approach was selected as the most widely researched, and therefore most promising one to focus on. Both binary and multiple-class approaches seemed worth experimenting with. Random forests and gradient boosting algorithms were selected as the main prediction algorithms to focus on, as they have been commonly found to outperform other methods at the task.

4 PROPOSED METHODS

4.1 Naive benchmark classifiers

To evaluate the necessity of more complex statistical learning approaches, two simple benchmark classifiers were developed to set the baseline. These classifiers were developed based on the observation found within literature that a customer's past payment behavior serves as a strong predictor for their current situation: i.e. if a customer has recently shown signs of financial difficulty, the payment of their next invoice is also more likely to be delayed.

The first naive classifier was designed to be as simple as possible, serving as the baseline. It selects the class of the most recent invoice sent to the customer as its prediction. If no invoices have been sent to the customer, the classifier predicts that the invoice is paid on time.

The second classifier extends the idea slightly by considering the class distribution of multiple previous invoices within the customer's invoicing history, weighted such that the contribution of most recent invoices is heavily prioritized over ones further in the past. To achieve this, each previous invoice's class label is assigned a weight calculated with the exponential decay formula

$$w(t) = e^{-\lambda t}, \tag{3}$$

where λ is a positive constant parameter that controls the rate of decay, and t is a variable describing the passage of time. In this case, t was selected to be the difference in days from the invoice to be classified. The optimal value for λ can be determined with a hyperparameter search. The classifier sums these weights per class for each invoice across the customer's invoicing history and selects the class with highest total weight as its prediction. If no history is available, it is again assumed that the invoice is paid on time.

4.2 Random forests

A random forest [25] is an ensemble learning model consisting of many small decision trees. The trees are trained on randomized subsets of the full training data, each sampled from the full data through bagging (see Section 2.3). Trees are good candidates for bag-

ging as they can learn complex interaction structures in the data, but are noisy. Averaging the results of many trees reduces the noisiness.

To further reduce correlation between the trees, the tree learning algorithm is also modified: at each decision split, only a randomly chosen subset of all input variables is used for determining the best variable and threshold for the split. Because of this, the optimal choice can be randomly left out of the consideration, resulting in more unique trees. The full training algorithm is outlined in Algorithm 1. [12]

Algorithm 1 Random forest algorithm, adapted from [12].

Definitions: Let x denote a set of data, and y denote the corresponding labels.

Hyperparameters: Amount of trees in the ensemble B. Bootstrap sample size N. Minimum leaf node size $n_{\min}$. Amount of randomly chosen variables per decision split m.

function GROW_TREE($node, x, y$)
    $x_r \leftarrow$ a subset of x containing only m randomly chosen input variables
    $rule \leftarrow$ optimal variable and split-point in x_r to maximize purity across y
    $x_1, x_2, y_1, y_2 \leftarrow$ split x, y according to $rule$
    $node_1, node_2 \leftarrow$ new binary tree nodes

    if $|x_1| >= n_{\min}$ **then**
        $grow\_tree(node_1, x_1, y_1)$
    end if
    if $|x_2| >= n_{\min}$ **then**
        $grow\_tree(node_2, x_2, y_2)$
    end if

    assign $rule$ as the value of $node$
    assign $node_1, node_2$ as the children of $node$
    return $node$
end function

function RANDOM_FOREST(x, y)
    $forest[B] \leftarrow$ new array
    for $b \leftarrow 1 : B$ **do**
        $node_b \leftarrow$ new binary tree node
        $x_b, y_b \leftarrow$ a bootstrap sample of size N drawn from x, y
        $forest[b] \leftarrow grow\_tree(node_b, x_b, y_b)$
    end for
    return $forest$
end function

The method can be adapted for both regression and classification with minor changes. In

the case of classification, the class prediction from a random forest ensemble is decided by majority vote. Predictions are first obtained from each individual tree, and the most often predicted class is selected as the result. [12]

The main adjustable parameters for the training process are the amount of variables considered for the decision split m, and the minimum node size $n_{\min}$. For classification, $m = \sqrt{p}$ and $n_{\min} = 1$ are recommended as a starting point [12]. The total number of trees to be generated B can also be adjusted, with higher numbers of trees generally smoothing out variance within the model and therefore increasing the accuracy of predictions. However, this increase has been observed to stabilize such that extensive tuning of the variable provides little practical benefit, so long as it is set to a high but computationally feasible number [12, 26]. Similarly, the performance benefits from controlling tree sizes with $n_{\min}$ have been found to be rather small [12].

4.3 Gradient boosting

Gradient boosting is a generalized form of several boosting algorithms. As discussed in Section 2.3, boosting is a popular ensemble learning approach that builds many weak learner models iteratively to create a stronger combined model. The learners are trained and added to the model sequentially, such that each new learner corrects the errors of the previous ones. LightGBM [27], XGBoost [28] and CatBoost [29] are three currently popular state-of-the-art implementations of gradient boosting, each sharing the same basic principles but introducing some further algorithmic developments.

In a gradient boosting algorithm, new models added to the ensemble of learners are not trained to predict the output variable directly. Instead, after making an initial guess, a loss function is used to calculate an error measure for each prediction, and a new learner is fit against the negative gradient of this loss function. Thus, the new learner outputs prediction values that are inversely correlated with the errors of the current ensemble. These error values are sometimes referred to as *pseudo-residuals*, drawing parallels to residuals used in linear regression. The predictions of the new learner are then combined with the previous ones, which nudges the ensemble's combined prediction towards the direction where the loss is minimized. Repeating this process multiple times iteratively minimizes the loss and has been shown to result in a model that is stronger than any of its individual components. The generic gradient boosting algorithm is presented in Algorithm 2. [12, 14]

Algorithm 2 Gradient Boosting algorithm, adapted from [12, 14].

Definitions: Let x denote a set of data, y the corresponding labels, and N the number of samples in $\{x, y\}$. Let $F(x)$ denote an individual weak learner, $f(x)$ the ensemble of learners, and $L(y, F(x))$ a differentiable loss function.

Hyperparameters: Learning rate ν ($0 < \nu < 1$). Number of learners in the ensemble M.

function GRADIENT_BOOST(x, y)
  $f_0(x) = \arg\min_F \sum_{i=1}^{N} L(y_i, F(x_i))$. ▷ Initialize the ensemble.
  for $m \leftarrow 1 : M$ **do**
    for $i \leftarrow 1 : N$ **do**
      $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$ ▷ Compute the "pseudo-residuals" r.
    end for
    $F_m = \arg\min_F \sum_{i=1}^{N} (r_{im} - F(x_i))^2$ ▷ Fit a weak learner to predict r.
    $f_m(x) = f_{m-1}(x) + \nu F_m(x)$ ▷ Update the ensemble with the predictions.
  end for
  return $f(x) = f_M(x)$
end function

Various loss functions $L(y, f(x))$ have been proposed to be used with Algorithm 2, resulting in a variety of more specific gradient boosting algorithms. The rationale behind certain types of functions having desirable properties for certain tasks is discussed further in [12, 14]. In classification tasks, multinomial deviance is typically used [12]:

$$\begin{aligned} L(y, p(x)) &= -\sum_{k=1}^{K} I(y \in C_k) \log p_k(x) \\ &= -\sum_{k=1}^{K} I(y \in C_k) f_k(x) + \log(\sum_{l=1}^{K} e^{f_l(x)}). \end{aligned} \tag{4}$$

Here $C = \{C_1 ... C_k\}$ represents the set of all prediction outcomes (classes), and $I(y \in C_k)$ is an indicator function returning 1 if the sample belongs in class C_k, 0 otherwise. In multiple-class prediction tasks, K separate trees are built at each step of the prediction algorithm, one for each class. Each tree F_{km} is fit to the negative gradient of the deviance [12]:

$$\begin{aligned} -r_{ikm} &= -\left[\frac{\partial L(y_i, f_1(x_i), ..., f_K(x_i)}{\partial f_k(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)} \\ &= I(y_i \in C_k) - p_k(x_i), \end{aligned} \tag{5}$$

where $p_k(x_i)$ is a logistic function representing the probability of sample x_i belonging in

the *k*th class [12]:

$$p_k(x) = \frac{e^{f_k(x)}}{\sum_{i=1}^{K} e^{f_i(x)}}. \tag{6}$$

The tree producing the largest value of $p(x)$ is selected as the classification result. [12]

Contrary to bagging, where increasing the amount of trees averages out the variance of individual learners within the ensemble (as they are all trained on an identical distribution), in gradient boosting each iteration of added trees fits the ensemble more tightly to the training data [12]. This can lead to an overfit model if too many iterations are made. The model can be regularized by limiting either the amount of iterations M, or the maximum amount of leaf nodes allowed for each tree. Additionally, the contributions of each tree are scaled by a learning rate parameter ν, $0 < \nu < 1$. Small values of ν have been found to lead to better results, with $\nu = 0.1$ being commonly suggested [12, 30].

4.4 Tree-structured Parzen Estimation

Hyperparameters refer to various settings or variables of a statistical learning algorithm that can be used to influence the training process and are set before training. These variables can have a large impact on the model's tendency to over- or underfit to the training data, and a balance between the two is necessary for good model performance. The process of finding optimal hyperparameters for a given model and dataset involves repeatedly training the model with a set of hyperparameters, evaluating the performance, changing the parameters according to some strategy, and repeating the process. After the search is stopped, the best-performing combination of parameters is chosen for the final model.

Tree-structured Parzen Estimation (TPE) [31] is a novel strategy for exploring the hyperparameter space that attempts to direct the search towards the most promising parts of the space. In contrast to a traditional grid search, which explores each possible combination of hyperparameters in order, TPE is not guaranteed to arrive to the global optimum within the space, but in practice it allows searching across a wider range of parameters in a computationally feasible manner and usually arriving at a "good enough" solution across that range.

4.5 Shapley additive explanations

One of the core benefits of individual decision trees is the ability to easily interpret the model's outputs: one can simply follow the decision splits from start to finish to see on what basis the model arrived to a given prediction. However, this becomes infeasible for a more complex ensemble model, so alternative approaches are needed. Shapley Additive Explanations (SHAP) [32] is a recently developed method for this task, unifying ideas from several previously proposed methods such as LIME [24] and classic Shapley value explanation. The method presents SHAP values as an unified, model-agnostic metric for explaining an individual feature's contribution to a prediction outcome. SHAP values provide both local and global interpretability to a model: they are calculated at the level of individual predictions, giving insight into how each specific prediction was reached, and across a large set of predictions the values can be summarized to visualize the overall feature importance of the model.

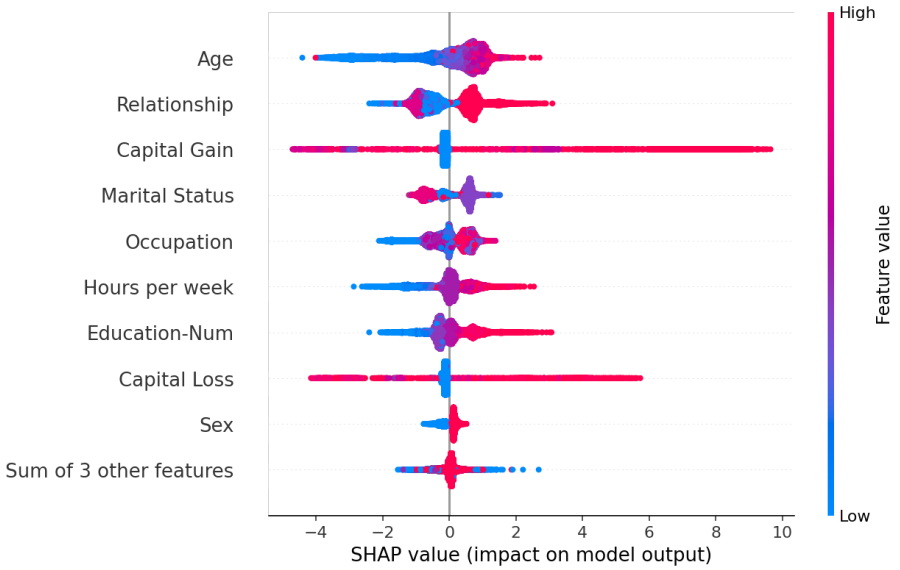

Figure 4. A SHAP summary plot visualizing feature importance for a classifier trained on the UCI adult income dataset. The goal of the task is to predict whether a person's income exceeded $50000. The plot shows that age and relationship status are the most significant predictors, with older and married people more likely to exceed the threshold. [33]

The original article [32] states that computing the exact SHAP values is difficult, but proposes both model-agnostic and model-specific approximations that can be used to estimate them with reasonable computational complexity. A later development into the SHAP framework introduced TreeExplainer [34], which uses the inherent properties of tree-based ensembles to compute exact SHAP values for these models in low-order poly-

nomial time.

The SHAP package provides many visualizations that aid in explaining the predictions of a model. The commonly used one is a summary plot, also called a beeswarm plot. An example is presented in Figure 4. In the summary plot, the SHAP values for a group of predictions are drawn on the x-axis per feature, indicating the change in the prediction caused by each given feature: dots far to the left from the center line signify large changes in the negative direction, and dots far to the right signify large changes to the positive direction. The dots are color-coded based on the value of the feature. The y-axis ranks the features in order of importance, with the ones causing the largest change on average appearing highest.

5 EXPERIMENTS

5.1 Data

The dataset used to perform the following experiments consisted of approximately 100 000 sales invoices from the ERP system. The dataset was constructed for this task with the goal of obtaining a representative sample of the companies using the software, with sufficient invoicing history data present for making predictions, while also remaining light enough for quick experimentation. As customer-level invoicing history has been previously shown in literature to be a significant factor in making accurate predictions, the invoices were selected from a random subset of companies using the accounting software, such that the entire invoicing history of each chosen company over a three-year period was selected. The maximum amount of invoices chosen per company was limited to 5 000, to prevent the inclusion of one large company from skewing the dataset too much.

Despite the relatively long period of invoicing history chosen, repeat invoicing to the same customers was still relatively scarce across the dataset. 27.5% of customers were only invoiced once over the three-year period, and 53.1% were invoiced five times or less. This could be because the ERP system is targeted mainly towards small and medium-sized enterprises, and the companies were selected randomly besides the limitations discussed earlier.

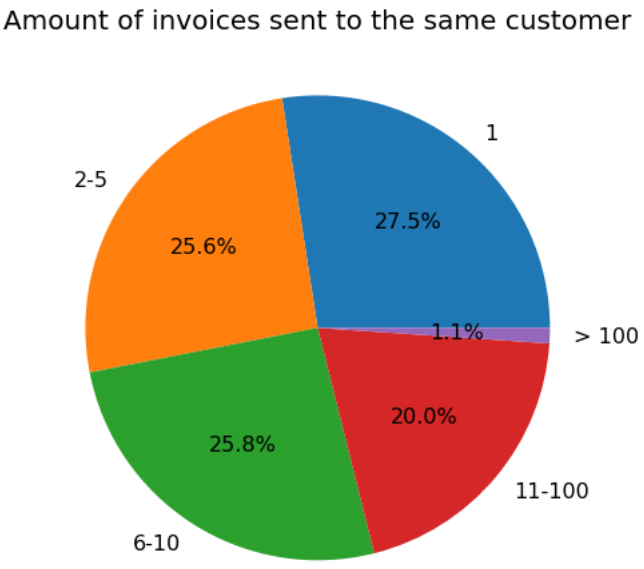

Figure 5. Amount of invoices sent to a given customer by a given company within the dataset.

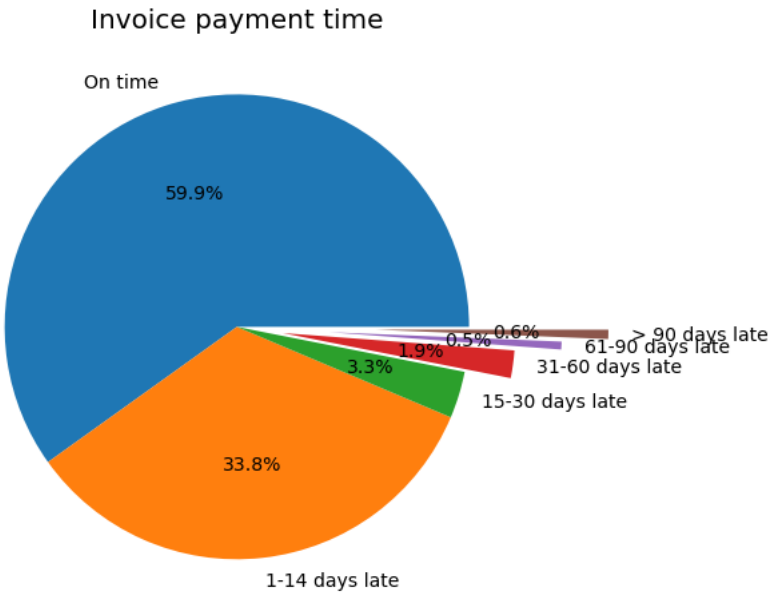

Figure 6. Class distribution across the data.

Alongside the invoice data, payment data linked to each invoice was used to determine the dates of payment. According to the data, 59.9% of invoices within the dataset were paid on time, and another 33.8% were paid less than 14 days late. Invoices with long delays were a clear minority, similar to the datasets used in previous studies.

5.2 Experiments

Several experiments were conducted to find answers to the research questions, utilizing the methods presented in Chapter 4. The classification algorithms evaluated were the two naive classifiers proposed in Section 4.1, random forests, XGBoost, LightGBM and CatBoost. In the results, the baseline naive model that predicts the class of the customer's previous invoice is referred to as "Naive 1", and the model utilizing the entire invoicing history with exponential discounting is referred to as "Naive 2".

At the start of each experiment, the hyperparameters of each model were optimized with the Tree-Structured Parzen Estimator algorithm implemented in the Optuna framework. The ranges for the hyperparameters were chosen based on the documentation for each classifier, and are reported in Appendix 1. The chosen hyperparameters for each classifier are presented in Appendix 2.

Nested 5-fold cross-validation was used to improve the robustness of the performance

evaluation. In k-fold cross-validation, the data is split into k equally sized folds, of which k-1 are used for training a model, and one is left out as an evaluation set. The evaluation process is repeated such that each fold is used as a testing set once, and the average of the evaluation metrics is reported as the final score. In nested cross-validation [35, 36], this idea is applied in two stages: an outer loop evaluates the performance of the model with each of the cross-validation folds in turn, and an inner loop searches for the optimal combination of hyperparameters utilizing the training data from the outer loop. The inner loop also utilizes cross-validation to evaluate the goodness of each given set of hyperparameters, splitting the given training data further into multiple folds, training a corresponding number of models, and taking their average result as the evaluation metric for the set of hyperparameters. After the search concludes, a final model is trained with the best found combination of parameters from the inner loop, and evaluated against the evaluation fold of the outer loop.

Balanced accuracy was selected as the main performance metric of the models, and therefore also the target variable for hyperparameter optimization, due to the class imbalance observed within the data. For binary classification it is defined as

$$BA = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right), \tag{7}$$

where TP refers to the amount of true positives (samples of the positive class that were correctly predicted by the model), TN to the amount of true negatives (samples of the negative class that were correctly predicted), FP to the amount of false positives (samples of the positive class that were incorrectly predicted), and FN to the amount of false negatives (samples of the negative class that were incorrectly predicted) within the data. F1-score was also reported as a secondary evaluation metric for binary classification tasks, defined as

$$F_1 = 2 * \frac{precision * recall}{precision + recall}, \tag{8}$$

where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$. For multiple-class classification, only balanced accuracy was reported, defined as the average recall across all classes.

Based on conclusions drawn from the literature review (see Section 3.1.3) and availability of information within the dataset, a list of baseline features was constructed. These features, presented in Table 4, were used in all of the experiments.

Table 4. Features constructed from the available data.

	Feature name	Description
1.	InvoiceSum	The monetary amount of the invoice.
2.	PaymentTerm	Payment term given to the invoice recipient, measured in days.
3.	IsPrivateCustomer	Determines whether the invoice recipient is an organizational entity or a private customer.
4.	NumberOfPaidInvoices	Number of times the recipient has been invoiced previously.
5.	NumberOfLateInvoices	Number of prior invoices paid late by the recipient.
6.	LateInvoiceRatio	Ratio of feature 5 over feature 4.
7.	TotalInvoiceAmountPaid	Total invoice amount of prior paid invoices.
8.	TotalInvoiceAmountLate	Total invoice amount of prior invoices paid late.
9.	LateInvoiceAmountRatio	Ratio of feature 8 over feature 7.
10.	MeanDelay	Average delay of all prior invoices paid by the customer.
11.	MeanDelayOfLateInvoices	Average delay of prior invoices that the customer has paid late.

5.2.1 Experiment 1: Binary classification utilizing base features

For initial experimentation, the task was reduced to a two-class problem, where the goal would be to simply distinguish late invoices from ones paid on time. Only features found in the original dataset were considered, so that the relevance of additional features constructed from Finnish Business Information System data could be verified separately.

The class imbalance was relatively mild in the two-class case: 59.9% of the invoices were paid on time, and 40.1% late. Before classification, the class distribution was balanced using random undersampling. This approach was selected for all binary classification experiments due to its simplicity, and the relatively low amount of data being discarded. The naive models were evaluated against the original non-resampled data set due to their method of prediction: given that predictions are solely based on the classes of preceding instances within the data, modifying the class distribution could have skewed their outcomes.

The results of the experiment, presented in Table 5, showed very similar performance measurements across all of the evaluated classifiers. Interestingly, the simplest naive

benchmark predicting the class of the previous invoice was able to achieve similar results as the evaluated classifiers, and the second one with exponential decay came out ahead of all classifiers. In terms of SHAP feature importance, the features *LateInvoiceAmountRatio* (the total monetary amount of invoices paid late divided by the total monetary amount of all invoices for the customer) and *MeanDelay* (average payment delay in days) were selected as the two most important features by all classifiers. Besides that, each classifier seemed to value different features, but the relationships between the low/high value of a given feature and the prediction outcome seemed to remain similar across classifiers. Figure 7 presents the SHAP beeswarm plot for the XGBoost classifier, which achieved the highest balanced accuracy besides the benchmarks.

Table 5. Classifier performance in binary classification with the baseline features. Standard deviation is presented for classifiers that were evaluated with nested cross-validation. The naive approaches were evaluated only once with the entire data set.

	Balanced accuracy	F1-score
Naive 1	0.716	0.653
Naive 2	**0.725**	0.660
Random forest	0.717 ± 0.012	0.706 ± 0.018
LightGBM	0.716 ± 0.014	0.704 ± 0.017
XGBoost	0.723 ± 0.011	0.699 ± 0.013
CatBoost	0.719 ± 0.016	$\mathbf{0.709 \pm 0.020}$

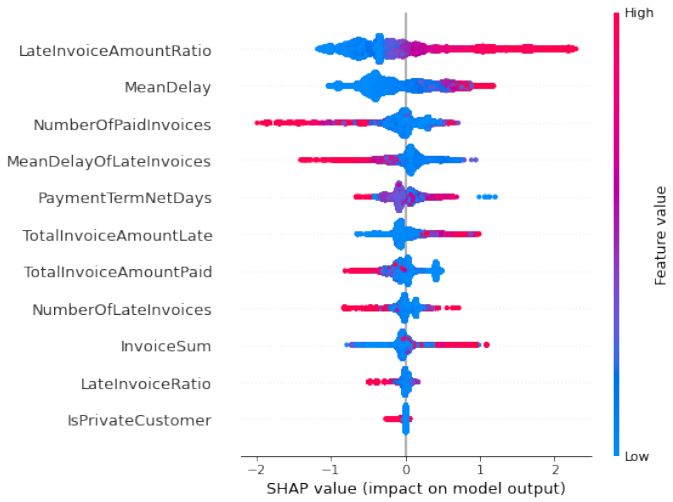

Figure 7. SHAP feature importance graph for the XGBoost classifier trained in Experiment 1. The left side of the graph corresponds to the class "on time", while the right side of the graph corresponds to the class "delayed". For example, the graph suggests that an invoice is more likely to be paid on time if the values of the features *LateInvoiceAmountRatio* and *MeanDelay* are low.

5.2.2 Experiment 2: Relevance of Finnish Business Information system data

For this experiment, additional features were added into the dataset based on Finnish Business Information System data published by the Finnish Patent and Registration Office. Among other things, the system publicizes information about bankruptcies and liquidation or restructuring proceedings for several types of companies. This information was extracted for each customer found within the dataset that had a valid Finnish Business ID and then converted into four binary features: *IsBankrupt*, *IsInLiquidation*, *IsInRestructuring* and *IsActivitySuspended*. The value for the feature was 1 if the customer had a given marking active at the time of receiving the invoice, 0 otherwise. Unfortunately only 11 out of 3 151 invoiced companies had any kind of a marking in the register within the 2019-2022 time period chosen for the data set.

In addition to these binary features, each company's main line of business was extracted as a categorical feature. The lines of business were reported with a five-digit code defined in Standard Industrial Classification TOL 2008 by Statistics Finland. These codes were grouped under the top-level categories defined in the standard, of which 20 appeared in the data set. The categories were one-hot encoded, thus resulting in 20 additional binary features.

Table 6 presents the classification results with these features added. The naive benchmarks perform identically to Experiment 1, as this additional data has no effect on their predictions. The addition of these features seemed to improve the results of the classifiers somewhat, allowing them to now very narrowly outperform the naive benchmarks. The SHAP plot of the XGBoost model presented in Figure 8 suggests that the model is picking up on certain lines of businesses to some degree, but their contribution is minimal compared to the more important features.

Table 6. Classifier performance with the additional features added.

	Balanced accuracy	F1-score
Naive 1	0.716	0.653
Naive 2	0.725	0.660
Random forest	0.727 ± 0.016	0.722 ± 0.014
LightGBM	0.726 ± 0.014	0.715 ± 0.010
XGBoost	$\mathbf{0.732 \pm 0.016}$	$\mathbf{0.727 \pm 0.011}$
CatBoost	0.728 ± 0.015	0.720 ± 0.011

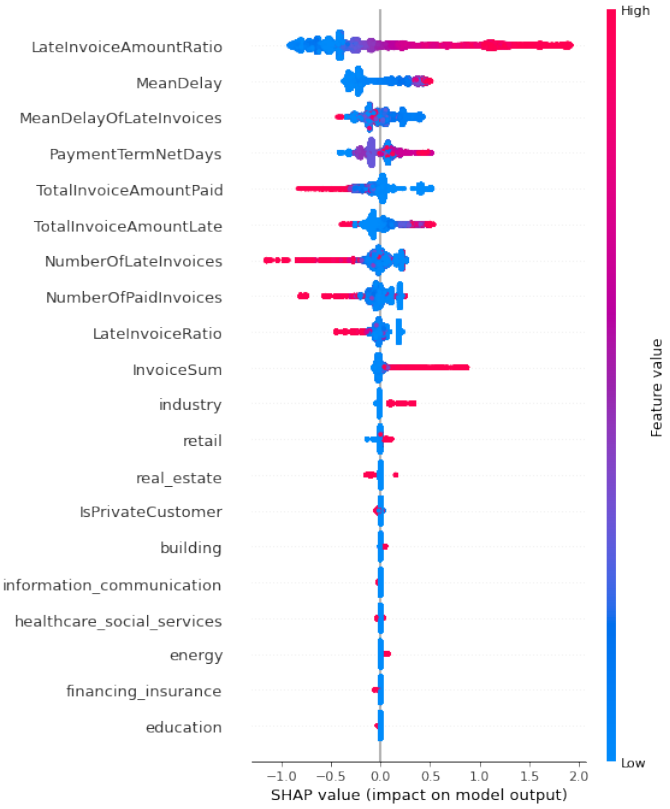

Figure 8. SHAP feature importance for the XGBoost model in Experiment 2. Each feature name in lowercase represents a possible value of the added line of business feature, which was one-hot encoded. The feature seems to carry minimal predictive value compared to the others.

5.2.3 Experiment 3: Effect of customer invoicing history

For this experiment, all invoices of customers with low amounts of invoicing history were cut from the data set. Three cutoff points were evaluated to assess correlation between amount of invoicing history and the quality of predictions. While not a part of the original research questions, the goal of this experiment was to find whether the models would be able to make more reliable predictions for customers with large invoicing volumes.

The results, presented in Table 7, seem to indicate that the predictions are more accurate for customers with more history data available, but even with high amounts of prior history there is a significant degree of uncertainty inherent to the task. The naive benchmarks achieved the highest balanced accuracy scores, while the XGBoost model scored highest in terms of F1-score in two out of three cases. Feature importance for the XGBoost model, presented in Figure 9, painted a similar story as previous experiments, with the feature *LateInvoiceAmountRatio* carrying most importance for prediction.

Table 7. The results of Experiment 3.

Prior invoices	Metric	Naive 1	Naive 2	Random forest	LightGBM	XGBoost	CatBoost
At least 5	BA	0.737	**0.747**	0.711 ± 0.014	0.717 ± 0.014	0.723 ± 0.011	0.715 ± 0.017
	F1-score	0.679	0.688	0.690 ± 0.017	0.700 ± 0.018	**0.704 ± 0.014**	0.698 ± 0.021
At least 20	BA	0.770	**0.779**	0.734 ± 0.014	0.752 ± 0.012	0.754 ± 0.013	0.739 ± 0.010
	F1-score	0.721	0.730	0.708 ± 0.017	0.736 ± 0.016	**0.740 ± 0.018**	0.720 ± 0.013
At least 50	BA	**0.834**	0.833	0.768 ± 0.007	0.783 ± 0.006	0.789 ± 0.009	0.779 ± 0.006
	F1-score	**0.808**	0.806	0.747 ± 0.010	0.772 ± 0.008	0.780 ± 0.012	0.764 ± 0.007

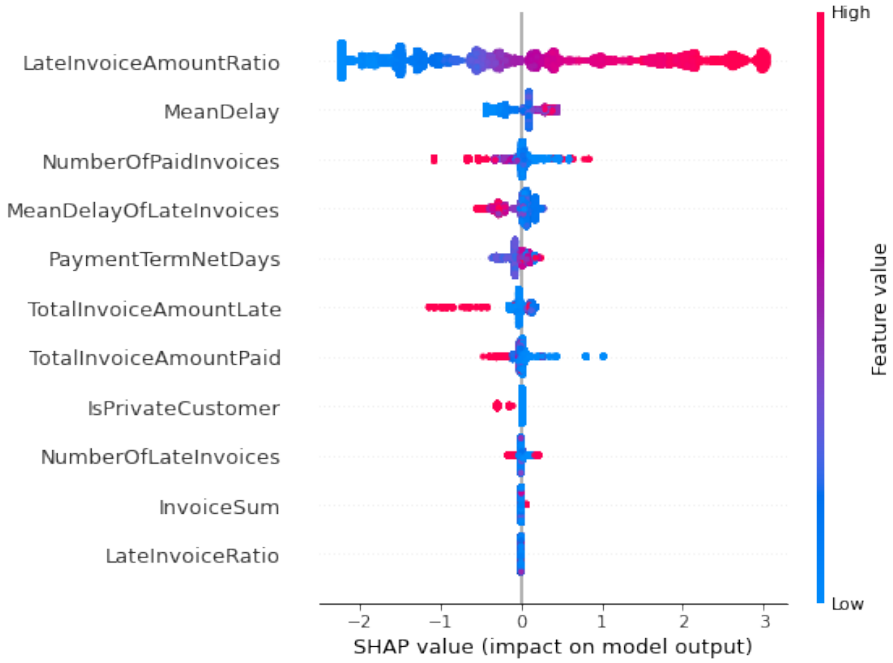

Figure 9. SHAP feature importance plot for the XGBoost model in Experiment 3, with the cutoff set to only include customers with at least 50 prior invoices.

5.2.4 Experiment 4: Multiple-class prediction

For this experiment, the invoices were divided into five classes, as presented in Figure 6. The additional features from the Finnish Business Information System were also included in this experiment. In the multiple-class classification case, the heavy class imbalance caused random undersampling to shrink the training set from 80 000 to just 2 000 samples, so several other oversampling-based class imbalance mitigation methods implemented in the `imbalanced-learn` Python library were evaluated in addition. The results obtained with each method are reported in Table 8. Naive benchmarks 1 and 2 were also evaluated against the full data set, both achieving an identical score of 0.388.

The combination of random oversampling with the LightGBM classifier seemed to perform best with this data, but again the differences in performance were quite minor across

Table 8. Balanced accuracy scores of evaluated classifiers, utilizing a variety of data balancing techniques.

	Random forest	LightGBM	XGBoost	CatBoost
Random undersampling	0.418 ± 0.013	0.421 ± 0.016	0.411 ± 0.018	0.408 ± 0.007
Random oversampling	0.430 ± 0.011	$\mathbf{0.432 \pm 0.004}$	0.431 ± 0.008	0.418 ± 0.021
SMOTE	0.420 ± 0.006	0.421 ± 0.005	0.419 ± 0.007	0.401 ± 0.018
SMOTE-ENN	0.412 ± 0.017	0.425 ± 0.012	0.421 ± 0.012	0.409 ± 0.017

the classifiers. The confusion matrix of this LightGBM classifier, presented in Figure 10, indicates that the classifier tries to predict the minority classes quite often compared to the class distribution of the holdout set, but the true class of most of these predictions is one of the majority classes. Other evaluated classifiers showed a similar trend. In essence, this seems to be an over-correction for the typical minority class problem, where classifiers tend to ignore the minority classes entirely in favor of prediction the majority classes.

The SHAP feature importance presented in Figure 11 shows similar tendencies to the binary classification problem, but the feature *NumberOfPaidInvoices* has risen as the most important one. It seems to have an especially large impact on the classification into the two smallest classes (61-90 days late and 90+ days late).

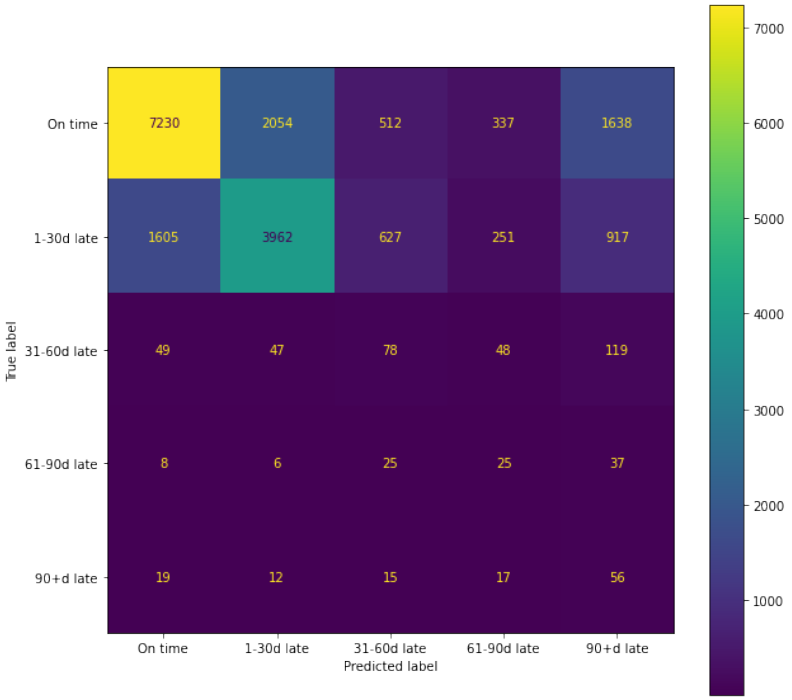

Figure 10. Confusion matrix for the LightGBM model in Experiment 4.

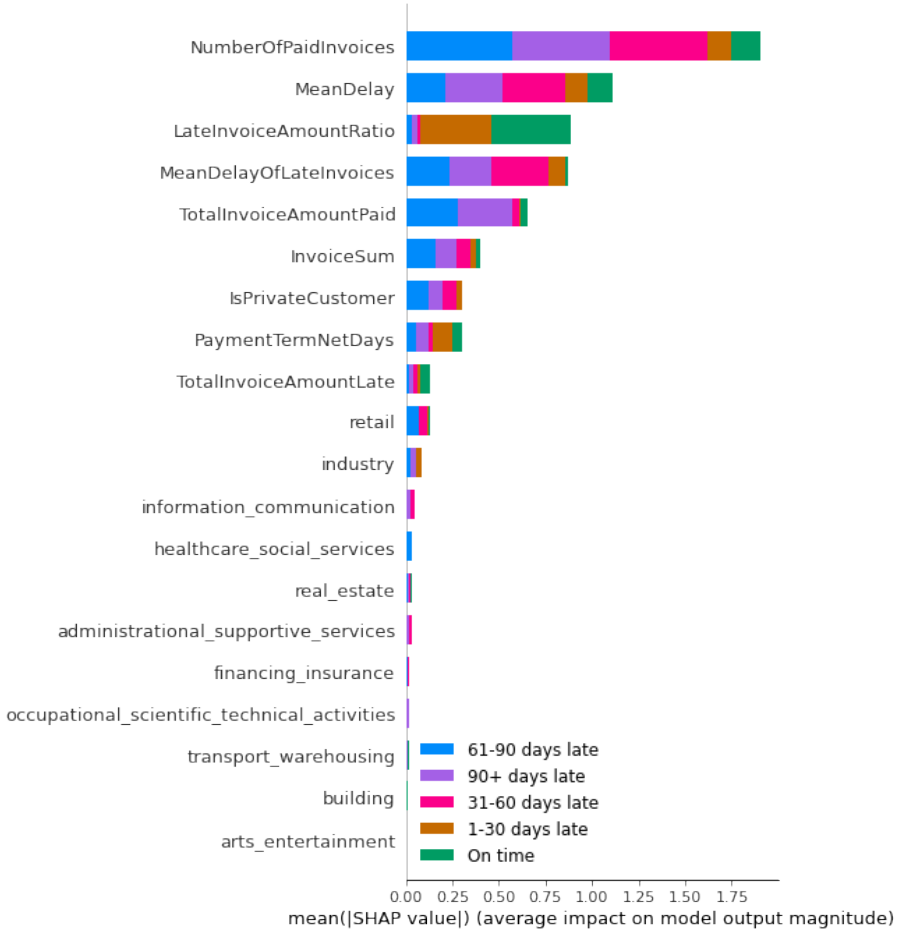

Figure 11. SHAP feature importance for the LightGBM model in Experiment 4.

6 DISCUSSION

6.1 Current study

This study explored methods for improving the accuracy of cash flow forecasts by predicting the lateness of sales invoices. Based on past studies on the topic, a classification-based approach was selected as the area of focus. The goal of the study was to replicate the results of previous studies with the set of data available, and hopefully improve on them with the help of the additional data from the Finnish Business Information System. All classifiers that performed well in the previous studies were evaluated, with the goal of finding the best-performing one for this set of data. The introduction of the naive baseline classifier was a somewhat novel contribution to the topic, as only one of the previous studies had reportedly used one. In this study [11], the baseline was even simpler than presented here, predicting all invoices to be on time.

Across the various experiments, all evaluated classifiers performed very similarly to one another. The results of the naive benchmarks were also surprisingly comparable, and in many cases even slightly better than those of the more complex classifiers. This clearly suggests that the customer's recent payment behavior is an especially strong predictor for whether their next invoice will be delayed, and it seems like many of the other features suggested in literature might be contributing much less additional predictive information than previously thought. There is certainly a significant degree of randomness and noise inherent in this task, and it is a possibility that there simply are not clear enough patterns present in the noise for a model to learn. This points towards the conclusion that attempts to tune the method of classification (algorithm selection, hyperparameter optimization etc.) are not currently a viable avenue for improving the quality of predictions to a meaningful degree, and future improvements should be sought through other means.

In each binary classification experiment, the features *LateInvoiceAmountRatio* and *MeanDelay* were found to contribute to the predictions the most out of the chosen features across all evaluated classifiers. Both of these features are customer-level history features, supporting the previous observation in literature that these features contribute to the outcome the most. The real-world connection seems rather straightforward: if a customer has previously delayed their invoice payments, it is quite logical to assume that they are more likely to also delay the next payment. However, in the multiple-class classification experiment, the highest performing feature was *NumberOfPaidInvoices*, which seems far less sensible. While it is definitely possible that long-standing customers are either more

or less likely to delay invoice payments than ones with less payment history, this connection seems difficult to explain. Reflecting on the low overall accuracy achieved in the multiple-class task, it seems more likely that the classifier has fixated on a pattern in the training data that does not generalize to the real world.

The effect of Finnish Business Information System data was found to be rather limited, as very few customers within the dataset had information about bankruptcies or liquidations. There could be multiple possible explanations for this: either this data is not added to the system frequently or for all types of businesses, or it could be that the demographic of the users of the accounting software makes them less likely to conduct business with financially troubled companies.

At first reaction, the misclassification rates of the models can seem rather high even in the "simple" binary classification case, and if one tries to draw comparisons to the multitudes of complex problems that can already be solved with a near perfect accuracy with statistical learning, the applicability of the models may seem limited. However, in the case of any financial predictions the level of uncertainty and randomness is very high, and perfect accuracy is not necessarily needed for a model to be useful. Due to the low accuracies obtained, any individual prediction from the created models should not be used without criticism, but over a sufficiently large number of invoices the predictions could provide a much better estimation of future accounts receivable cash flow than what has been previously available. As such, these estimations would be most useful for businesses with large invoicing volumes: if a company is awaiting payment from only a few large invoices at a given time, it would likely be unwise to make any future investment decisions based on these predictions, but companies with dozens to hundreds of incoming payments at a given time could be able to benefit from the ability to predict general trends within the mass of payments.

6.2 Future work

This study seems to suggest that classifier algorithms matter very little for this task with the current set of features found in literature. Although novel classification methods such as deep learning architectures for tabular data have seemingly not yet been evaluated for this task and could be worth exploring, it appears that the time would be better spent working on other areas for now. For instance, it could be a good idea to try weighting the history features such as the mean delay of previous invoices, such that they cover only a certain amount of months or only the past n invoices from the customer. Additional

sources of data could also be explored, such as customer credit information from commercial agencies, stock prices, or some other broad indicators of the current economic situation across an industry.

Another avenue for future study could be trying to explore the potential seasonal fluctuations within the data, as the current approach does not account for seasonality. It could be possible that invoices are more likely to be paid late during certain times of the year, either across certain industries or as a whole.

For the purposes of estimating time to invoice payment, it might also be interesting to tackle the problem with regression methods instead of classification. Regression would provide an estimation for the time of payment that is more specific than the date range offered by multiple-class classification, and it would be interesting to compare the accuracy of these predictions to ones obtained from multiple-class classification.

7 CONCLUSION

With the data available, it was shown that statistical learning methods can be used to estimate payment delays of sales invoices to some degree. In a binary classification task, where invoices were classified as "late" or "on time" based on prior information about the customer's payment behavior, a XGBoost model was used to classify invoices with a balanced accuracy score of 72.3% and a F1-score of 69.9%. Notably, a naive benchmark predicting the most frequently appearing class in the customer's recent invoicing history achieved a slightly higher balanced accuracy score of 72.5%, but a slightly lower F1-score of 66.0% in the same task. With a 5-class approach, dividing the invoices into classes "on time", "1-30 days late", "31-60 days late", "61-90 days late" and "90+ days late", a LightGBM model reached a balanced accuracy of 43.2% across the classes.

According to SHAP values obtained from the classifiers, the two most significant features that contributed to the prediction of a late payment were the average delay of previous invoices by the customer, and the percentage of money that had been paid late out of all money previously paid by the customer. Additional features obtained from the Finnish Business Information System did not seem to provide any meaningful benefit to the predictions. The amount of available bankruptcy and liquidations proceedings data was extremely small, and it remains inconclusive whether these features would improve the predictions if obtained from another source with more data available. The customers' line of business data was readily available, but the feature extracted from this data had minimal impact on the classifiers' predictions.

REFERENCES

[1] Heitor Almeida, Murillo Campello, Igor Cunha, and Michael S Weisbach. Corporate Liquidity Management: A Conceptual Framework and Survey. *NBER Working Paper Series*, page 19502, 2013.

[2] Andrew Fight. *Cash Flow Forecasting*. Elsevier Butterworth-Heinemann, 2006.

[3] Yaobin Wu, Yingying Wang, Xun Xu, and Xiangfeng Chen. Collect payment early, late, or through a third party's reverse factoring in a supply chain. *International Journal of Production Economics*, 218:245–259, 2019.

[4] David A. Wuttke, Constantin Blome, H. Sebastian Heese, and Margarita Protopappa-Sieke. Supply chain finance: Optimal introduction and adoption decisions. *International Journal of Production Economics*, 178:72–81, 2016.

[5] Sai Zeng, Prem Melville, Christian A. Lang, Ioana Boier-Martin, and Conrad Murphy. Using predictive analysis to improve invoice-to-cash collection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1043–1050, 2008.

[6] Hu Peiguang. Predicting and Improving Invoice-to-Cash Collection Through Machine Learning. Master's thesis, Massachusetts Institute of Technology, 2013.

[7] Weikun Hu. Overdue Invoice Forecasting and Data Mining. Master's thesis, University of Washington, 2014.

[8] Jean-Loup Ezvan. Predicting late payment of an invoice. Master's thesis, University Paris-Dauphine, 2018.

[9] Arthur Hovanesyan. Late payment prediction of invoices through graph features. Master's thesis, Delft University of Technology, 2019.

[10] Ana Paula Appel, Victor Oliveira, Bruno Lima, Gabriel Louzada Malfatti, Vagner Figueredo de Santana, and Rogerio de Paula. Optimize Cash Collection: Use Machine learning to Predicting Invoice Payment. 2019.

[11] Susana Lopes da Costa Rebelo. Predicting Account Receivables Outcomes with Machine-Learning. Master's thesis, Universidade Nova de Lisboa, 2021.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2017.

[13] Martin Krzywinski and Naomi Altman. Classification and regression trees. *Nature Methods*, 14(8):757–758, 2017.

[14] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, 2022.

[15] D. Opitz and R. Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11(1):169–198, 1999.

[16] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

[17] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[18] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54:1937–1967, 2021.

[19] Victoria López, Alberto Fernández, Jose G. Moreno-Torres, and Francisco Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012.

[20] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, San Francisco, California, USA, 2016.

[21] Tarun Tater, Sampath Dechu, Senthil Mani, and Chandresh Maurya. Prediction of Invoice Payment Status in Account Payable Business Process. In *Service-Oriented Computing*. Springer International Publishing, 2018.

[22] Janika Smirnov. Modelling Late Invoice Payment Times Using Survival Analysis and Random Forests Techniques. Master's thesis, University of Tartu, 2016.

[23] Xolani Dastile, Turgay Celik, and Moshe Potsane. Statistical and machine learning models in credit scoring: A systematic literature survey. *Applied Soft Computing*, 91:106263, 2020.

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

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

[26] Philipp Probst and Anne-Laure Boulesteix. To Tune or Not to Tune the Number of Trees in Random Forest. *Journal of Machine Learning Research*, 18(1):6673–6690, 2018.

[27] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the International Conference on Neural Information Processing Systems*, volume 30, pages 3146–3154, 2017.

[28] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[29] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Proceedings of the International Conference on Neural Information Processing Systems*, volume 31, pages 6639–6649, 2018.

[30] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[31] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization.

[32] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the International Conference on Neural Information Processing Systems*, volume 31, pages 4768–4777, 2017.

[33] Beeswarm plot — SHAP latest documentation. https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/beeswarm.html.

[34] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020.

[35] M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[36] Gavin C Cawley and Nicola L C Talbot. On Over-fittingg inModel Selectionn and-Subsequent Selection Biass inPerformance Evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

Appendix 1. Hyperparameter search ranges

Table A1.1. Parameters chosen for the hyperparameter search and the chosen search ranges for each of the evaluated models.

Model	Parameter name	Type	Range of values
Naive 2	lambda	float	[0, 1]
Random forest	max_depth	integer	[8, 32]
	n_trees	integer	[50, 200]
	max_features	categorical	'sqrt', 'log2'
LightGBM	n_leaves	integer	[2, 100]
	min_child_samples	integer	[1, 100]
	max_depth	integer	[2, 32]
	max_bin	integer	[16, 512]
XGBoost	learning_rate	float	[0.01, 0.4]
	min_split_loss	integer	[0, 10]
	max_depth	integer	[2, 32]
	colsample_bytree	float	[0.2, 1]
CatBoost	iterations	integer	[500, 1500]
	learning_rate	float	[0.01, 0.4]
	depth	integer	[4, 10]
	l2_leaf_reg	float	[0.1, 0.5]
	random_strength	float	[0.5, 2]

Appendix 2. Hyperparameter search results

Table A2.1. Hyperparameter search results for all experiments. For Experiment 4, results are provided for the run where random oversampling was used.

Model	Parameter name	Exp. 1	Exp. 2	Exp. 3, 5 invoices	Exp. 3, 20 invoices	Exp. 3, 50 invoices	Exp. 4
Naive 2	lambda	0.350	0.350	0.355	0.258	0.696	0.760
Random forest	max_depth	10	8	8	8	8	9
	n_trees	168	51	191	150	123	121
	max_features	log2	sqrt	sqrt	sqrt	sqrt	sqrt
LightGBM	num_leaves	12	12	5	2	2	7
	min_data_in_leaf	19	12	68	18	13	16
	max_depth	23	14	15	27	25	20
	max_bin	319	101	22	304	242	485
XGBoost	learning_rate	0.133	0.092	0.028	0.045	0.090	0.185
	min_split_loss	8	2	6	0	9	10
	max_depth	7	3	2	2	2	3
	colsample_bytree	0.561	0.320	0.575	0.0708	0.849	0.560
CatBoost	iterations	1015	745	1253	605	918	566
	learning_rate	0.016	0.017	0.010	0.010	0.010	0.036
	max_depth	6	5	4	4	4	5
	l2_leaf_reg	0.349	0.299	0.241	0.387	0.102	0.453
	random_strength	0.532	0.550	1.995	1.758	1.487	1.239