GENERAL ARTICLE

Machine learning methods for short-term Probability of Default
A comparison of classification, regression and ranking methods

Lize Coenen[a], Wouter Verbeke[a] and Tias Guns[a]

[a]Department of Business technology and Operations, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Elsene, Belgium

ARTICLE HISTORY
Compiled October 27, 2020

ABSTRACT
Probability of default estimation via machine learning on historical data is widely studied in credit risk modeling. In this work, we investigated the use of machine learning for a finer-grained risk estimation task, namely spot factoring. Here, the goal is to estimate the likelihood that an invoice will be paid in an acceptable timeframe. In this case, risk is more related to the *overdueness* of an invoice. Based on this observation, we investigate three possible machine learning tasks for estimating this risk: binary classification for a predetermined overdue days cutoff; regression of the overdue days; and learning-to-rank which learns to optimize the risk-related ranking for the full range of instances. We model and evaluate these tasks using real-life spot factoring data. Finally, we perform a profit-driven evaluation that shows that regression models can lead to higher profits and better spread the risk than classification and ranking models for spot factoring.

KEYWORDS
Machine learning; Spot factoring; Probability of default; Credit risk; Learning-to-Rank

1. Introduction

Granting credit is at the heart of the banking industry. It comes however with the risk that a borrower will not repay the amount owed. Accurate credit risk modelling therefore constitutes the core business of a bank. A well-known metric in credit risk modeling is called *probability of default* (PD), which expresses the probability that a borrower fails to meet payment obligations over a specific time horizon (Baesens, Rösch, & Scheule, 2016). The rise of machine learning has seen the direct and automated estimation of this metric. Supervised machine learning methods allow using vast amounts of historical financial data to model probability of default and predict the metric for new borrowers (Lessmann, Baesens, Seow, & Thomas, 2015).

In the aftermath of the 2008 financial crisis, traditional banks tightened their lending standards (*Global Financial Stability Report: Moving from Liquidity- to Growth-Driven Markets*, 2014). This sparked an upsurge in alternative financing

CONTACT Lize Coenen. Corresponding author. Email: Lize.Coenen@vub.be
CONTACT Wouter Verbeke. Email: Wouter.Verbeke@vub.be
CONTACT Tias Guns. Email: Tias.Guns@vub.be

solutions (Kalara & Zhang, 2018) for companies that lack a credit history or collateral and are therefore more risk-exposed, like starters, small and medium-sized enterprises (SMEs) and businesses dealing with long payment terms. One such financing solution is spot factoring. A spot factor typically acts as an intermediary between investors and businesses which are in need of constant cash flow. In exchange for immediate payment, the latter businesses are willing to sell one or more invoices for provided goods or services at a discount. The spot factoring company will naturally want to audit these businesses before agreeing to a purchase. Such an assessment is in the credit industry often performed using *scorecards*, that is, tools relying on a retrospective database with descriptive characteristics of past customers to map new customer information to possible actions (Hand, 2005). One such action can be to consult a human operator (e.g. in difficult cases). The use of machine learning methods can help this decision making process by optimizing the performance of the scorecard model and guiding the human operator when necessary.

Spot factoring is, as yet, a largely untapped research context in machine learning and earlier research efforts specifically for this task are - to the best of our knowledge - non-existent. Smart decision support in spot factoring shares a common goal though with credit risk modeling: both banks and spot factors want to identify customers that will fail to meet their payment obligations and reject them. A spot factoring decision is usually taken within one or a few days, on a per-invoice basis and involves a one-time payment, which makes the decision more fine-grained than what can be expected in e.g. a typical mortgage lending scenario. These characteristics can however be found in other financial products like loans that were granted online, peer-to-peer loans and bullet loans. The specifics of spot factoring lead to a less clear-cut nature of its defaulting process though. Invoices for example might only get paid back after an extra effort of the factoring company, such as a reminder phone call or registered mail. Furthermore, late payments are common in spot factoring. Some defaults are also partially paid back by the insurer whereas others are not (in case of fraud). It is important to note that, although fraudulent cases indeed occur in our spot factoring dataset, their amount is so small that the use of fraud detection is rendered futile.

Traditional PD estimation amounts to using a binary classifier to predict the event of a borrower defaulting on a loan. In a spot factoring context, we can use this approach to predict the event of an invoice never being paid back. To address the less clear-cut nature of a spot factoring outcome, however, we use the notion of overdueness estimation in this paper. A spot factor is interested in more than the repayment of an invoice - it is also concerned with its *overdueness* w.r.t. a contractual due date.

In order to express overdueness in a spot factoring context, we construct three new variables to predict. One such target variable directly shows the overdueness in terms of *days overdue*. A second, *graded*, target variable that we use stems from the way decision makers categorize invoices in risk groups that reflect increasingly later payment periods and more thorough collection efforts. We obtain a third and *binary* target variable by merging the two best and merging the three worst risk groups. This amounts to splitting the invoices between those that are paid within the first 25 days after their due day and those that are not.

We investigate the suitability of three prediction tasks when using the new target variables: classical binary classification, regression of the days overdue and

learning-to-rank methods that optimize the ranking of all invoices w.r.t their overdueness. To evaluate the suitability of the labels rather than the learning methods, we evaluate across three method families using metrics from all three tasks. Furthermore, we perform a profit aware analysis to quantify the performance with a single metric.

Our main contributions are the following:

- We investigate the use of machine learning techniques for risk estimation in the emerging problem of spot factoring
- We propose three different machine learning tasks that are suitable for risk estimation in terms of overdueness: binary classification, regression of days and ranking of invoices
- We evaluate each of the tasks across three families of machine learning methods, namely generalized linear models, support vector machines and gradient boosted trees and evaluate it on a use case with two years of data
- Using profit driven evaluation as unifying metric across all tasks, we identify regression to be most suitable

The remainder of this paper is structured as follows. Section 2 presents an overview of related literature. Section 3 introduces our proposed methodology for overdueness estimation in terms of three prediction tasks. Section 4 outlines the set-up and results of our performance experiments. Section 5 draws some conclusions and identifies future work.

2. Related Work

The Basel regulation defines a default event to have occurred when the lender considers that an obligor - a person or company obliged to pay an amount owed - is unlikely to pay its credit obligations to the lending group in full or when the obligor is past due more than 90 days on any material credit obligation to the lending group or both (Basel Committee on Banking Supervision, 2006). The estimation of the probability that a default event will occur has been extensively investigated in credit risk literature. We refer the reader to detailed surveys on the matter (Baesens et al., 2003; Lessmann et al., 2015; Thomas, 2000).

Whereas credit risk modelling typically involves predictions over long-term horizons starting from the present moment (e.g. the best practice of estimating a 12-month probability of default), a spot factoring company is interested in short-term predictions (usually spanning one month) w.r.t. a certain due date. An application field that shares this goal of overdue estimation is train delay estimation. In this setting, recent research efforts include the use of Support Vector Regression (SVR) (Marković, Milinković, Tikhonov, & Schonfeld, 2015), Extreme Learning Machines (MLS) (Oneto et al., 2017), neural network models (Yaghini, Khoshraftar, & Seyedabadi, 2013), a Fuzzy Petri Net (FPN) (Milinković, Marković, Vesković, Ivić, & Pavlović, 2013) and graph-based approaches (Berger, Gebhardt, Müller-Hannemann, & Ostrowski, 2011; Kecman & Goverde, 2015).

Factoring is an either privately or bank-owned financing solution that consists of a factor buying invoices from a client company at a discount (Soufani, 2002). Afterwards, the factor receives the invoice payment from the client's customers. Spot

factoring is a type of factoring that allows obligors to sell at a per-invoice basis (Brownstein, 2003) instead of selling all of their credit-worthy invoices as is typical in standard factoring. Spot factoring is therefore more risky than standard factoring and usually requires the factor to ask a higher discount on the original invoice amount.

Elliott and Curet (1999) give the first framework to study invoice discounting, a generic term for financing solutions that use invoices as collateral for a loan. They suggest the use of an inductive algorithm such as case-based reasoning (CBR) and notice the lack of knowledge about invoice discounting cases. The amount of papers that have been published since then regarding invoice discounting cases is, to the best of our knowledge, rather limited. Dorfleitner, Rad, and Weber (2017) estimate default events in the online invoice trading market using logit models. Zhang and Thomas (2015) consider the merits of including economic variables into a logistic regression based credit scorecard in an invoice discounting context. In this way, they want to directly estimate a short-term, dynamic version of the probability of default, i.e. Point-in-Time (PIT) probability of default. The work of Perko (2017) also involves an adapted definition of probability of default to a more short-term and fine-grained one: he considers predicting the outcome at a fixed timeframe of 30 days ahead in the future, rather than overdue days.

A related domain to estimating the overdueness of an invoice is accounts receivable (AR) management or invoice collection management. AR refers to money a company is owed from its customers. Whereas a spot factor can choose to accept an invoice, a company interested in optimizing its AR management cannot, as the goods or services have already been delivered. Such a company wants to optimize the use of its collections resources, e.g. by not contacting customers that have a high probability of paying on time. Similar to the spot factoring case, machine learning techniques can support AR management in predicting an invoice's payment outcome. For example, Zeng, Melville, Lang, Boier-Martin, and Murphy (2008) use supervised classification to build a model that classifies new invoices into one of five target classes expressing the extent to which an invoice is late. They use five traditional classification methods, concluding that the rule learner and C4.5 outperform the competing methods. To account for different misclassification costs, they apply instance re-weighting of the minority class. Hu (2015) gives another example of supervised learning in an invoice collection context, aimed at modeling the probability of default as well as the magnitude of the delay. In the work of Appel et al. (2019) the predicted probability of an invoice being late is used to rank customers and in that way improve the prioritization of contacting clients with overdue invoices. Tater, Dechu, Mani, and Maurya (2018) consider the opposite problem of accounts payable (AP); i.e. minimizing the penalties due to outstanding amounts payable to creditors instead of optimizing the allocation of resources to collect outstanding amounts receivable from debtors. They regard the problem as a binary classification task as well.

Among the risk estimation techniques, we use a regression model to predict the amount of days between due date and payment date of an invoice. In credit risk modeling, survival analysis methods like the Cox proportional-hazards model are often used for duration modeling (Alves & Dias, 2015; Carling, Jacobson, Lindé, & Roszbach, 2007; Dirick, Claeskens, & Baesens, 2017; Leow & Crook, 2016; Noh, Roh, & Han, 2005). These methods estimate whether a borrower will default within a granted payment period. Such a bounded time-span causes censorship in credit scoring data:

one cannot state anything about the creditworthiness of a customer outside of the payment period; one can only determine whether there was a default event during that time. This dependence on a time-period is missing in a spot factoring context: we want to model the event of an invoice (or judicial fees) being paid relative to the due date because a spot factor's profits or costs depend upon the payment delay. Apart from dealing with censorship, Cox proportional-hazards models allow for the inclusion of time-dependent explanatory variables - which are not present in our data - and result in hazard and survival functions of the time while we would like to predict the time of the event itself. Therefore, we choose to directly model invoice overdueness via a regression model of the payment delay.

In order to optimize predicted invoice rankings, we will focus on well-established Learning-to-Rank methods. Related to these techniques are ordinal classification and regression methods that exploit the ordinal nature of the dependent variable in the data. A few attempts have been made to apply these methods in the context of estimating the risk of companies (Swiderski, Kurek, & Osowski, 2012) or expert ratings (García, Giménez, & Guijarro, 2013) and credit ratings (Baourakis, Conisescu, van Dijk, Pardalos, & Zopounidis, 2009; Dikkers & Rothkrantz, 2005; Hirk, Hornik, & Vana, 2018; Kim & Ahn, 2012; Van Gestel, Baesens, Van Dijcke, Suykens, & Garcia, 2005). The used approaches include binary goal programming (García et al., 2013), SVM variants for ordinal multi-class classification (Dikkers & Rothkrantz, 2005; Kim & Ahn, 2012; Swiderski et al., 2012), ordinal regression methods (Baourakis et al., 2009; Hirk et al., 2018) and combinations of ordinal regression and SVMs (Van Gestel et al., 2005). In this paper, we consider standard regression as well as Learning-to-Rank.

We are not aware of any credit risk studies using Learning-to-Rank (L2R) methods, although it is the mainstay of handling order-aware prediction problems. L2R methods originate from the information retrieval community for applications like search queries and online advertising. In both of these examples, one is mainly interested in correctly ranking the top-k items with the best ranking score. Because spot factoring requires the correct overdueness estimation of all invoices, our main interest concerns L2R methods that are able to optimize the ranks of the full range of items. Two of these, SVMRank (Joachims, 2002) and CRR (Sculley, 2010), are well-known pairwise estimators, which are trained to maximize the number of correct pairs among all pairs in a ranking. SVMRank extends the notion of support vector machines to minimize the disconcordance between corresponding pairs in the predicted and true ranks, i.e. minimize the amount of cases where the predicted and true ranks of pairs disagree. Ranking algorithms can be time-demanding due to the necessary comparison of all possible pairs within a dataset. By sampling from the set of candidate pairs, the Combined Regression and Ranking (CRR) algorithm delivers more scalability. CRR alternates between optimizing a regression and ranking objective function. A third L2R method, LambdaMART (Burges, 2010), concerns a listwise approach involving a loss function to optimize over a whole list of borrowers. LambdaMART directly optimizes the Normalized Discounted Cumulative Gain (NDCG) measure but can be used with other evaluation measures as well.

3. Methodology

To answer the question of how we can support a spot factoring company in the decision to accept or reject an invoice, we will investigate the performance of three types of prediction tasks in the next sections. Several relevant target variables present themselves, each of them giving rise to another prediction task and therefore other prediction methods. The three tasks are summarized below.

Classification for Probability of Default estimation In our first approach, we estimate the probability that an invoice is going to be paid within a fixed number of days after the *due date*. We set this limit to 25 days past the due date as invoices become increasingly undesirable from then on due to additional administrative and judicial costs. This results in a binary target label distinguishing invoices based on their risk of going 25 days past their due date. Notice that this approach is an alteration in terms of scope and time frame to the interpretation of Perko (2017) of the probability of default, which considered the time frame of 30 days from the current moment across all invoices.

Regression of days overdue Our second approach is based on the idea that a simple classifier is ill-fit to support the decision process. Firstly, the cutoff of 25 days is a part of the business model rather than an empirically estimated parameter and therefore should not be 'hard coded' into the algorithm. Secondly, it does not account for possible flexibility the factoring company might want to demonstrate regarding risks it is willing to take or guarantees it received w.r.t. the estimated overdue date. Meanwhile, a regression model can be used both for estimation of an invoice's overdueness as well as for binary classification. As a target variable, this task uses a variable containing the difference in days between the payment day and due date of the invoice.

Learning-to-rank The goal of our prediction tasks amounts to smart decision support. When looking at the decision procedures which are currently at place in spot factoring companies, we notice that human operators assess invoices on a graded scale (with the possible values being *good*, *acceptable*, *undesirable*, *bad* and *horrible*). In machine learning, the prediction of graded relevance labels is typically associated with Learning-to-Rank (L2R) techniques. Ranking has the advantage to regression that outcomes reflect a deliberate ordering between instances, while ranks derived from regression outcomes can be greatly affected by even the smallest regression errors. Indeed, classification and regression methods look at one instance at a time during training - amounting to a *pointwise estimation*, whereas L2R methods use a pair of instances at a time (*pairwise estimation*) or even the entire list of instances (*listwise estimation*) while learning. Furthermore, ranking outcomes in the form of relevance scores still allow for flexible thresholding. Learning-to-Rank methods are most suited for settings with few unique labels (e.g. binary or graded overdueness) but we run them on the relative overdueness labels too (where all negative overdueness, e.g. paid in time, is rounded to 0). We investigate this prediction task using the five-class graded label as described above, as well as the binary probability of default label and regression label showing days overdue from the other two prediction tasks.

To perform the first prediction task of probability of default estimation, we focus on three distinct methods that performed well in the overview of credit scoring methods in (Lessmann et al., 2015): support vector classifiers (SVC), logistic regression (LR) and random forests (RF). For the regression task estimating days overdue, we use the regression counterparts of the earlier mentioned classifiers, i.e. support vector regression (SVR), ridge regression (RR) and random forest regression (RFR). In case of the Learning-to-Rank task, we decide upon the following methods that are able to optimize over the full range of invoices: SVMRank (Joachims, 2002), CRR (Sculley, 2010) and LambdaMART (Burges, 2010).

In order to understand what is being optimized in every prediction task for every method family and how this can be evaluated, we dive deeper into the method specifics and evaluation metrics in the next two sections.

3.1. Method families

In order to draw conclusions on the three prediction tasks, rather than the specific learning methods, we compare the tasks using three diverse types of method families: generalized linear models, support vector machines and gradient-boosted trees. The use of classifiers from these three families have been extensively explored for modelling credit risk in literature (Lessmann et al., 2015). Also, each family allows for a method that learns to rank. Neural networks, which are increasingly popular in the wider research field, can be seen as non-linear extensions of generalized linear models. They can use the same loss functions as the linear models, as we do in the experiments. To better understand the difference in tasks for each of the learning methods, we discuss the different objectives that the different methods are optimizing below.

3.1.1. Generalized linear models

A first family of methods that we will study is the family of generalized linear models (GLM), i.e. models where the model prediction $y(\mathbf{x}, \mathbf{w})$ is a linear function of the parameters $\mathbf{w}$, e.g. $\mathbf{x} * \mathbf{w}$, optionally with a transformation according to a function $f(\cdot)$ (Bishop, 2006).

3.1.1.1. Classification. For the binary prediction task, we will use the well-known GLM method *logistic regression* (LR). Here the transformation function f is the logistic sigmoid function $\sigma(a) = 1/(1 + exp(-a))$. This transformation maps all real numbers to the range between 0 and 1 and is therefore well-suited to model (class) probabilities. The model parameters $\mathbf{w}$ can then be found as the values that maximize the likelihood function $p(\mathbf{t}|\mathbf{w})$ for the binary target variables $t_1, ..., t_N$ with $t_n \in \{0, 1\}$. Maximizing this likelihood function amounts to minimizing the log-likelihood, which is equivalent to the cross-entropy loss function:

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\{\mathbf{t}^T \ln(\mathbf{y}) + (1 - \mathbf{t}^T) \ln(1 - \mathbf{y})\} \tag{1}$$

This means that samples belonging to a certain class but being assigned a low predicted probability for that class get penalized exponentially.

3.1.1.2. Regression. For the regression task we will choose *ridge regression* (RR), which performs L2 regularized linear regression. This means that the loss function is the linear least squares function and the regularization is given by the L2-norm of the weight vector (Bishop, 2006):

$$E(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{t}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2}$$

In this way, large differences between predictions and true labels will be penalized. The regularization term limits the magnitude of the weights and therefore complexity of the model. The hyperparameter λ represents the trade-off between predictive power and model simplicity.

3.1.1.3. Ranking. To perform learning-to-rank, we use the Combined Regression and Ranking (CRR) method as described in (Sculley, 2010) which optimizes both regression and ranking objectives simultaneously:

$$E(\mathbf{w}) = \alpha L(\mathbf{w}, D) + (1 - \alpha)L(\mathbf{w}, P) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{3}$$

Here D represents the dataset and P the set of candidate pairs with different labels within the datasets. In both terms, the loss function is the same function, with $L(\mathbf{w}, D)$ comparing true and predicted labels and $(\mathbf{w}, P)$ comparing the true and predicted label difference within a pair. The logistic loss or squared loss as used in LR and RR as described above are two possible loss functions that can be used here. The parameter α represents the trade-off between regression and ranking.

3.1.2. Support vector machines

A second method family concerns methods that try to find the largest possible margin between a hyperplane - the *decision boundary* - and the data points that lie closest to it, also known as *support vectors*. Consider a training dataset with N input vectors $x_1, ..., x_N$ and corresponding target values $t_1, ..., t_N$ and the function whose parameters to train $y(x) = \mathbf{w}^T\phi(\mathbf{x}) + b$. The objective function of such a support vector machine (SVM) can then be written as (Bishop, 2006):

$$\sum_{n=1}^{N} E(y(x_n), t_n) + \lambda \|\mathbf{w}\|^2 \tag{4}$$

The first part of this objective function is specific for each type of support vector machine. The second part tries to maximize the margin.

3.1.2.1. Classification. For binary classification we will use a classic support vector machine. Consider target values $t_n \in \{-1, 1\}$ and the function $y(x)$ assigning a class to x according to the sign of $y(x)$. The SVM error function is called *hinge loss* and penalizes predictions that lie on the wrong side of the margin boundary:

$$E_{hinge}(y(x_n), t_n) = \max(0, 1 - y(x_n)t_n) \tag{5}$$

This margin maximization objective gives rise to more robust classifiers compared to e.g. GLM. Furthermore, the way SVMs are formulated allows them to apply a kernel trick and model non-linear relationships.

3.1.2.2. Regression. For the regression task, we will consider a support vector machine adapted for regression (SVR). To obtain this, we regard $y(x_n)$ as a regression curve that tries to approximate the real-valued target labels $t_n \in \mathbb{R}$. Now a tube with width ϵ around $y(x_n)$ is optimized containing as many data points as possible (instead of a margin containing as few as possible), giving the following ϵ-insensitive error function:

$$E_\epsilon(y(x_n), t_n) = \begin{cases} 0 & \text{if } |y(x_n) - t_n| < \epsilon \\ |y(x_n) - t_n| - \epsilon, & \text{otherwise} \end{cases} \tag{6}$$

3.1.2.3. Ranking. For the learning-to-rank task, we will use the proposal of Joachims (2002) for a ranking SVM algorithm (SVMRank). Here, the training dataset consists of pairs (x_i, x_j) where datapoint x_i is ranked higher than x_j according to the true labels. The error function then amounts to the classification SVM's error function for the *pairwise* difference between x_i and x_j:

$$E_{rank}(x_i, x_j) = \max(0, 1 - \mathbf{w}(x_i - x_j)) \tag{7}$$

By minimizing this error function, the number of discordant pairs - i.e. pairs whose rankings disagree - will be minimized as well.

3.1.3. Gradient-boosted trees

The last family of methods that we will study is the family of gradient-boosted trees (GBT). Such methods sequentially create an ensemble from weak decision trees, correcting a previous tree's prediction by optimizing a differentiable loss function. This is an ensemble method which is known to reduce variance and give state-of-the-art results.

3.1.3.1. Classification. For the binary classification task, we make use of the *eXtreme Gradient Boosting* (XGBoost) toolkit by Chen and Guestrin (2016). XGBoost decreases the following objective function:

$$E = L(\mathbf{w}) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2, \tag{8}$$

with T the amount of leaves, w_j^2 the score on the jth leaf and γ and λ trade-off parameters between loss and regularization. L represents the loss function and equals the logistic loss [1].

3.1.3.2. Regression. For the regression task, we again use XGBoost, but with the squared loss [2].

3.1.3.3. Ranking. To perform the learning-to-rank task, we use LambdaMART (Burges, 2010). This method uses gradient boosted trees as well, but includes a

different cost function inspired by its precursor LambdaRank to accommodate the ranking purposes. LambdaMART penalizes large deviations between the predicted and true relevance difference within a pair. It scales this penalization with a metric's value indicating the significance of ordering that pair correctly:

$$C = \sum_{\{i,j\} \rightleftharpoons I} |\Delta Z_{ij}| \log(1 + e^{-2(s_i - s_j)}) \text{ with } \sum_{\{i,j\} \rightleftharpoons I} \lambda_{ij} = \sum_{j:\{i,j\} \in I} \lambda_{ij} - \sum_{j:\{j,i\} \in I} \lambda_{ij} \quad (9)$$

Here, I is the set of pairs of indices i, j for which we desire instance i to be ranked higher than instance j and s_i is the predicted ranking score for instance i. ΔZ_{ij} is the utility difference generated by swapping the rank positions of instance i and instance j (Burges, 2010). Z is a ranking metric that is typically chosen to be Normalized Discounted Cumulative Gain ($NDCG$). We use the top-N version of NDCG ($NDCG@N$):

$$DCG_N = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)} \text{ and } NDCG_N = \frac{DCG_N}{IDCG_N}, \quad (10)$$

with N the amount of instances in your dataset, $IDCG_p$ the ideal discounted cumulative gain and rel_i the graded relevance of the result at position i. Compared to the two previous ranking methods, LambdaMART gives more emphasis to the instances ranked highest.

3.2. Evaluation metrics

We evaluate the three different tasks - classification, regression and ranking - with metrics from each of the three tasks. This will allow us to compare them in different ways and to evaluate the quality of the resulting methods in general. The applicability of a metric depends upon the type of target label that is used during the training and the one used during the evaluation of the considered method. Since all methods output numerical values, constituting an ordering, ranking metrics can be used for all types of labels.

3.2.1. Classification metrics

We use two metrics that are common to evaluate probability classifiers: the Area Under the Receiver Operating Characteristic curve (AUROC) and Area Under the Precision Recall curve (AUPR). By plotting the corresponding curves, we visualize the performance of each method for a binary classification task using the binary labels of the data (e.g. positive if less then a fixed number of days, otherwise negative). As classifiers are often used to predict whether or not an event will occur, classification metrics expect binary prediction labels. Regression and ranking methods do not provide such predictions. However, ROC curves and PR curves show classification metrics for all possible threshold values and can therefore handle non-binary predictions. In a ROC curve, a method's false positive rate is plotted versus its true positive rate for the full range of threshold values. Ideally, we want a method's ROC curve to approximate the upper left corner, which amounts to methods that assign higher prediction scores to all good invoices than to the bad invoices. ROC curves regard the classification of true positive

labels as separate from the classification of true negative labels, rendering them unaware of possible large skews in the class distribution of a dataset. We therefore include PR curves in our analysis, which plot a method's recall versus its precision for all possible thresholds. A perfect PR curve approximates the upper right corner. The **AUPR**, also known as Average Precision (**AvP**) (Boyd, Eng, & Page, 2013) in the L2R literature, and the **AUROC** (Fawcett, 2006) can be viewed as the summary statistics of resp. the PR and ROC curve.

3.2.2. Regression metrics

To evaluate the quality from a regression perspective we use the metrics Mean Squared Error (**MSE**) and Coefficient of Determination ($\mathbf{R^2}$). MSE shows the average squared difference between predicted values and true target values. R^2 represents the proportion of the variance for a dependent variable that can be explained by the independent variables. Both of these metrics are designed to compare variables whose values and order of magnitude are meaningful. We will therefore not use them for the Learning-to-Rank task and binary classification task as their outcomes are resp. arbitrarily high ranking scores and class labels.

3.2.3. Ranking metrics

To evaluate the quality of ranking we use two standard rank correlation coefficients, namely **Kendall's** τ and **Spearman's** ρ. Kendall's τ represents the difference in number of concordant and number of discordant pairs between the ranks implied by the true target variable and ranks implied by the predictions, normalized so that $-1 \leq \tau \leq 1$. We will use the tau-b version that accounts for ties (Knight, 1966). Spearman's ρ is another rank correlation coefficient, which is the Pearson correlation coefficient between the rank implied by the true target variable and the rank implied by the predictions. As Kendall's τ and Spearman's ρ are not limited to the graded label, we use them for the days label as well, e.g. to evaluate how well ranking an output by method predictions corresponds to ranking by the true days labels. This is an advantage to MSE and R^2, which can only handle regressor predictions.

3.2.4. Profit-driven evaluation

All measures discussed so far either evaluate correct class assignment, regression value accuracy or pairwise ranking concordance between the prediction and true target variable values of the data. Profit-driven evaluation measures regard the performance of a model considering its practical purpose. In order to achieve such a cost-aware comparison, we plot the profit curves of the different methods, showing a spot factor's profit for all possible acceptance threshold values. Verbeke, Baesens, and Cristián (2018) use profit-driven evaluation techniques as well by considering the costs associated with accepting or rejecting good or bad observations when calculating the average profit per threshold. Instead of using such binary misclassification costs, we identify an average return or cost for each of five possible gradations of invoices and sum the respective values for all accepted invoices to calculate the profit per threshold.

4. Experiments

We present three experiments pertaining to the problem of how we can support a spot factoring company in the decision to accept or reject an invoice. More specifically, we address the three following questions: (RQ1) how do the different tasks compare from a probability-of-default perspective, (RQ2) which prediction task is suited best when evaluating the output accuracy to the different metrics of different tasks and (RQ3) how do the different tasks compare from a profit maximization perspective.

Experimental set-up

Dataset Our dataset consists of 5491 invoices and 117 features. From those features, 62 are binary, 38 numeric and 17 categorical. They contain i.a. background information about the involved companies, details of the invoice and information from secondary sources but also aggregate features that we created from historical data, similar to the approach in (Appel et al., 2019). These latter features are aggregated over all invoices of a certain debtor, creditor or even debtor-creditor combination and in that way can be compared to the behaviour-enriched data with multiple levels as presented in (Perko, 2017). In Fig. 1, the cumulative Fisher score (in blue) or information value (in orange) is shown per variable type, which gives an idea of their predictability w.r.t. the outcome variable. It is clear that all types have their merits in predicting the outcome. The variables about the background of the involved companies and aggregate variables of historical data about these companies seem especially useful. Please note that the Fisher score and information value are different quantities and these values should therefore not directly be compared. Further information about the features cannot be disclosed due to the sensitivity of the data.

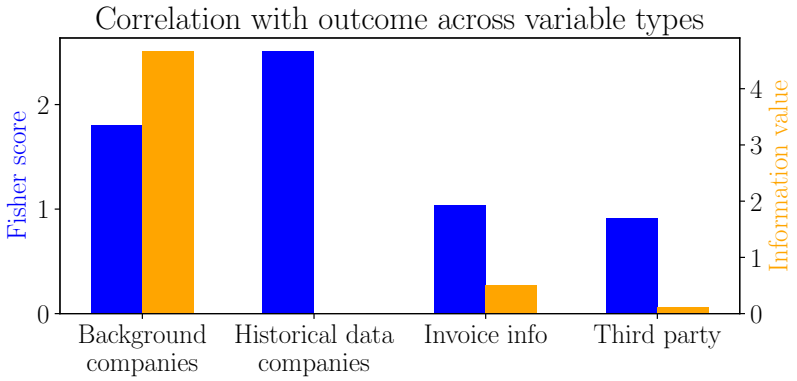


Figure 1.: Fisher score of the 100 numerical variables in blue and information value of the 17 categorical variables in orange, per variable type.

The data was collected at a European spot factoring company over a period of two years for the purpose of this research. We use the outcome of invoices accepted by the spot factoring company. The dataset is imbalanced: only 10% of the invoices are not paid within the first 25 days after the due date. From the dataset, three possible target variables are constructed: (I) a regression label specifying the amount of days between the payment and due date of an invoice, which equals zero for invoices that are paid

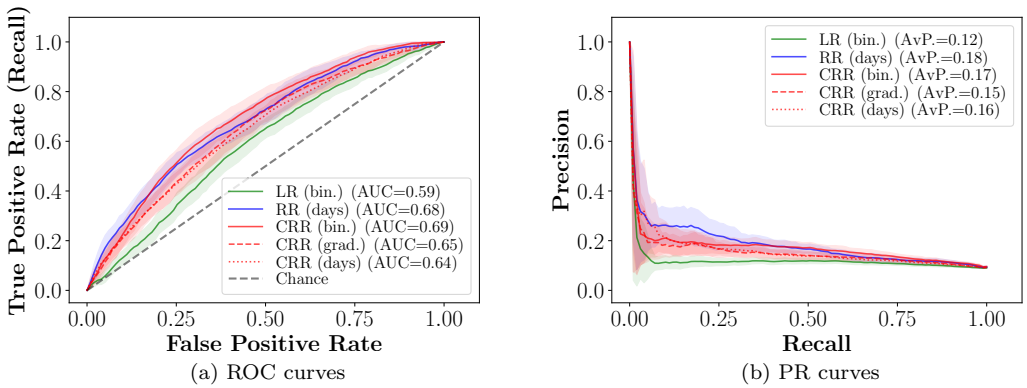
in time or too early, (II) a graded variable based on expert appreciation of increasingly later payment dates, with the possible values being *good*, *acceptable*, *undesirable*, *bad* and *horrible* (with resp. relative frequencies 55%, 36%, 9%, 0.1% and $< 0.1\%$) and (III) a binary variable specifying whether an invoice was paid 25 days after the due date.

Preprocessing In case of the experiments with MLPClassifier and MLPRegressor, we preprocessed our features by normalizing them and reducing the dimensionality to the 12 most informative features according to their impurity-based feature importance when training random forests on the dataset. For the experiments involving MLPClassifier, we oversampled the minority class (i.e. 1) in the binary target variable.

Categorical variable conversion During data processing we use a modified version of weight-of-evidence (WOE) encoding to transform ordinal categorical variables to numerical ones, as is common in probability of default estimation. Of the 117 features in our dataset, 17 are categorical. Indeed, if a dichotomous target variable is present, WOE encoding (Hand & Henley, 1997) expresses a certain category's potential to separate datapoints with a good target label from datapoints with a bad one. By using the modification of WOE via a Laplace smoothing technique as defined in (Zdravevski, Lameski, & Kulakov, 2011), category values that only include positive or negative samples can be transformed as well. In the experiments involving non-binary target labels, we use label encoding (Boschetti, 2016). This involves transforming categorical variables to numerical ones by using a dictionary that maps each category alphabetically to a subsequent integer, e.g. {'FirstCategory': 1, 'SecondCategory': 2, ...}. Both encodings are used for all methods. The other 100 (binary and numeric) features are left as is.

Partitioning To make optimal use of our moderately sized dataset, we apply two levels of cross-validation and average our results over multiple repetitions. We start off selecting a (5%) part of the data according to a stratified split. In the experiments involving WOE encoding (i.e. those that use binary target variables), we use this part to calculate the WOE encoding for the categorical variables in the rest of the data. In the rest of the experiments (involving label encoding), this part will be unused. The rest of the data is used in inner-outer stratified cross-validation (Bishop, 2006) that is repeated five times to ensure that our results are not due to chance. Cross-validation is regarded best practice in model selection and training. It optimally uses the available data and accounts for variability in the choice of hyperparameters, which are integral aspects of getting good results with machine learning methods. We use the stratified 5-fold outer cross-validation to create five distinct outer loops, each one considering another part (20%) of the data as the hold-out test set and the rest as the training set used to build the model. In each of the outer loops, hyperparameter tuning is implemented by evaluating the performance of the training set for each possible hyperparameter combination. This is done by subjecting the training set to stratified 4-fold inner cross-validation, resulting in four inner loops with distinct (75%) training and (25%) validation parts. We included the hyperparameter values that we considered during tuning in Appendix B and underlined the values that were most often deemed the best.

Computational details We perform our experiments using Python 3.7.3. We use the SVC, SVR, RandomForestClassifier, RandomForestRegressor, MLPClassifier and MLP-



(a) ROC curves (b) PR curves

Figure 2.: Probability of default curves for generalized linear model methods when using different labels. The shaded regions represent the ± standard deviation of the curves.

Regressor methods as implemented in the Python machine learning library scikit-learn (v0.21.2). The implementations for XGBoostClassifier and XGBoostRegressor come from the software library XGBoost (v0.82). To run our CRR, SVMRank and LambdaMART experiments, we respectively use the suite of fast incremental algorithms for machine learning (sofia-ml), the SVMRank program by Joachims (2002) and the L2R library RankLib (v2.11).

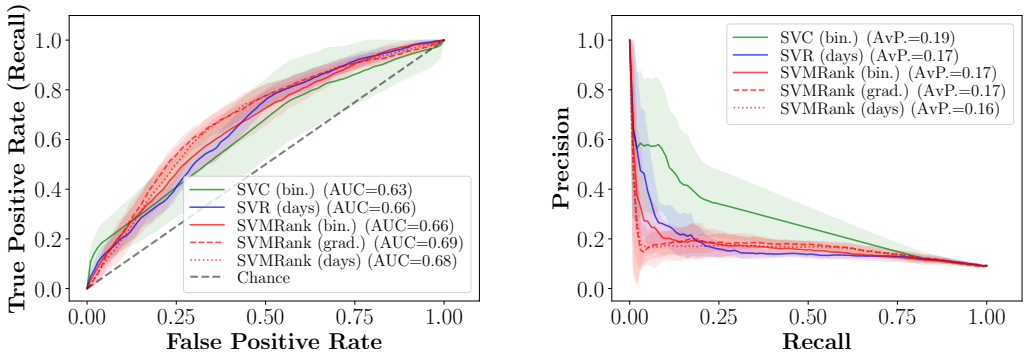
4.1. RQ1: How do the different tasks compare from a probability of default perspective

The first question assesses each prediction task from the position of a classic probability of default estimation. In such a context, the target label is a binary label indicating whether an invoice was paid before it became 25 days overdue. We can compare all prediction tasks' performances by ranking the output of metrics that are typically used to evaluate binary classification. The non-binary target labels that are learned in the regression and Learning-to-Rank task are no problem, for they can be transformed using appropriate thresholding methods. We plot the ROC and PR curves, which compare numerical predictions to binary true labels over the full range of threshold values.

Figure 2 shows the ROC and PR curves for the generalized linear model family for the methods as described above. Hyperparameter optimization of the neural networks either leads to networks without a hidden layer in case of classification (see Appendix B) or networks with smaller AUC values than the linear methods in case of regressors and are therefore not shown. We can observe in the figure that the best AUC value, i.e. 0.69, is for Combined Regression and Ranking (CRR) with binary labels, which is known to optimize AUC directly, followed by Ridge Regression (RR) with a value of 0.68. Other methods perform slightly worse. Also, for the PR curves, CRR with binary labels and RR perform best, with RR better at low recall rates.

Figure 3 shows the ROC and PR curves for the support vector machine family. There is a clear difference in tendencies between both curves. For small FP rates,

SVC's ROC curve dominates all other curves. For larger rates, SVC's ROC curve is overtaken by all other curves. There, SVMRank with graded labels and SVMRank with days labels top all other curves. SVMRank with binary labels and SVR show lower and similar results. In the PR curves, the superiority of the SVC curve can be observed. This can be explained by noticing the class imbalance of the dataset and the steep start of the SVC ROC curve. For high threshold values, SVC immediately shows a high TPR versus FPR ratio. This corresponds to a method that correctly assigns a high score to a lot of the bad invoices without assigning high scores to a lot of the good invoices. Because there is an imbalance in the amount of good versus bad invoices, this effect is even more outspoken than can be seen on the ROC plot, which contains relative axes. The PR plot shows the amount of bad invoices that get noticed as being bad versus the amount of invoices with a 'bad' predicted label that are actually bad. Here it is clear that SVC starts off detecting a lot of the bad invoices while the amount of 'false hits' is limited. SVC therefore seems to be particularly strong in detecting bad invoices. Notice however the large variance of the SVC curves, indicating that an SVC will not give consistently good results. The PR curves of SVR and SVMRank with binary labels top the curves of SVMRank with graded labels and SVMRank with days labels, although the four curves coincide for higher recall rates.

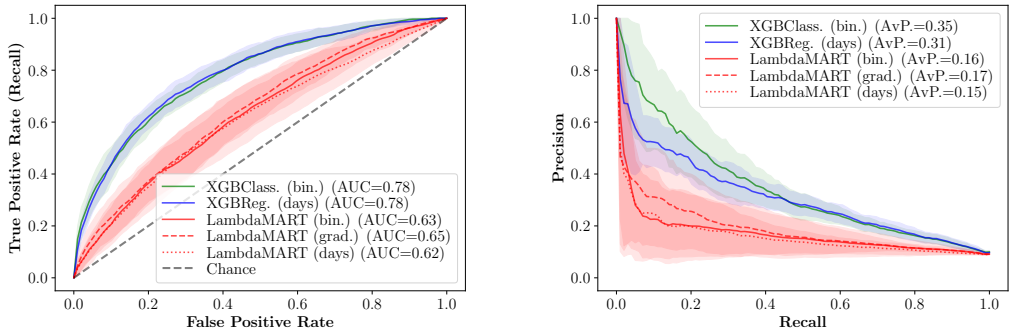


(a) ROC curves (b) PR curves

Figure 3.: Probability of default curves for support vector machine methods when using different labels. The shaded regions represent the ± standard deviation of the curves.

Figure 4 shows the ROC and PR curves for the gradient boosted trees family. The classifiers and regressors with XGBoost clearly outperform the LambdaMART method. In both plots we can observe that the XGBoost classifier curve is slightly higher than the regressor curve for small recall/true positive rates. For higher rates, the XGBoost regressor curve slightly tops the classifier one. The LambdaMART curve for graded labels is consistently the highest of all LambdaMART curves in both the ROC and PR plots.

Learning-to-Rank methods performs well for the generalized linear models family and support vector machines, but not for the gradient boosted trees. In all families, regression is among the best techniques, even if evaluated in terms of classification metrics.



(a) ROC curves

(b) PR curves

Figure 4.: Probability of default curves for gradient-boosted trees methods when using different labels. The shaded regions represent the ± standard deviation of the curves.

4.2. RQ2: How do the different tasks compare across the different task-based perspectives

The second question addresses the performance of the three prediction tasks over a broader range of evaluation metrics. In the previous question, we evaluated all methods for a binary classification task. To have a more complete view on each method's merit, we will here compare the methods from the previous question according to regression metrics, i.e. mean squared error and the coefficient of determination R-squared, and ranking metrics like Kendall's tau and Spearman's rho as well.

Table 1 shows the evaluation metrics corresponding to the three families for each of the possible tasks. The first two columns represent the AUROC and AvP values from Figure 2, 3 and 4. For the generalized linear models, the rankers perform the best: CRR with binary labels has the highest value for AUROC, CRR with graded labels has the highest value for the evaluation with graded labels and CRR with days labels has the highest value for the evaluation with days labels. It is interesting to notice that, while the sofiaml implementation of CRR only allows for the logistic loss and hinge loss to be chosen as loss functions during hyperparametertuning, all CRR variations have higher values than LR and SVC for almost all evaluation metrics. This speaks in favour of the additional difference loss function that CRR optimizes, which discerns it from LR or SVC.

We also compare GLM with neural networks in more detail in this experiment, more specifically for the standard classification and regression tasks. The results in Table 1 show that the added expressivity leads to marginal improvements for all metrics in case of classification. However, when we consider the most common hyperparameter values for MLP Classifier (see Appendix B), we notice that most of the used neural nets do not have a hidden layer and are therefore essentially doing linear regression (followed by a possible non-linear transformation depending on the activation function). Using neural networks leads to a somewhat better MSE score in case of regression. Comparing ridge regression (RR) and the MLP Regressor, we see that only on the task-specific MSE score the MLP performs better; on all other scores the RR results are more favourable including the AUROC.

Table 1.: Evaluation metrics for different learning tasks per method family. The classifiers use the binary label, the regressors use the days label and the Learning-to-Rank methods use all three (binary, graded and days) labels. Higher is better for all metrics except Mean Squared Error (MSE). The highest score in each respective group is underlined, the highest across all groups is shown **in bold**. The method families GLMs, SVMs, GBTs and MLPs are resp. the *generalized linear models*, *support vector machines*, *gradient-boosted trees* and *multilayer perceptrons*.

Evaluation label:	**Binary**		**Graded**		**Days**			
	AUROC	AUPR	K' τ	S' ρ	MSE	R^2	K' τ	S' ρ
GLMs								
class.: LR (bin.)	0.59	0.12	0.1	0.12	/	/	0.09	0.13
rank.: CRR (bin.)	0.69	0.17	0.18	0.23	/	/	0.17	0.24
rank.: CRR (grad.)	0.65	0.15	0.26	0.33	/	/	0.24	0.34
rank.: CRR (days)	0.64	0.16	0.25	0.31	/	/	0.25	0.35
regr.: RR (days)	0.68	0.18	0.22	0.28	300.79	-0.0	0.22	0.3
MLPs								
regr.: MLPClassifier (bin.)	0.61	0.13	0.16	0.2	/	/	0.16	0.22
regr.: MLPRegressor (days)	0.59	0.13	0.19	0.24	277.17	0.02	0.19	0.27
SVMs								
class.: SVC (bin.)	0.63	0.19	0.12	0.13	/	/	0.11	0.14
rank.: SVMRank (bin.)	0.66	0.17	0.13	0.16	/	/	0.13	0.18
rank.: SVMRank (grad.)	0.69	0.17	0.35	0.44	/	/	0.33	0.46
rank.: SVMRank (days)	0.68	0.16	0.32	0.4	/	/	0.32	0.44
regr.: SVR (days)	0.66	0.17	**0.37**	**0.46**	306.52	-0.02	**0.37**	**0.5**
GBTs								
class.: XGBClas. (bin.)	**0.78**	**0.35**	0.23	0.29	/	/	0.22	0.31
rank.: λMART (bin.)	0.63	0.16	0.09	0.11	/	/	0.09	0.13
rank.: λMART (grad.)	0.65	0.17	0.21	0.25	/	/	0.2	0.28
rank.: λMART (days)	0.62	0.15	0.07	0.09	/	/	0.07	0.1
regr.: XGBReg. (days)	**0.78**	0.31	0.35	0.44	**265.84**	**0.12**	0.35	0.47

Except for those metrics that evaluate for a binary classification task, SVR performs best across all support vector machines and even across all values in the table.

In case of the gradient-boosted trees, the regressor again performs best. Only for the AvP values, the classifier with XGBoost scores sligthly better. The LambdaMART version with graded labels scores better than the binary label and days label version. Compared across all families, the regressor with XGBoost scores best as well according to the MSE and R^2 values.

Overall, regressors score best with SVR the best performer for the rank correlation coefficients with graded and days evaluation labels, closely followed by the GBT regressor with XGBoost which shows much better results for the classification metrics AUROC and AvP. and for the regression metrics MSE and R^2.

4.3. RQ3: How do the different tasks compare from a profit maximization perspective?

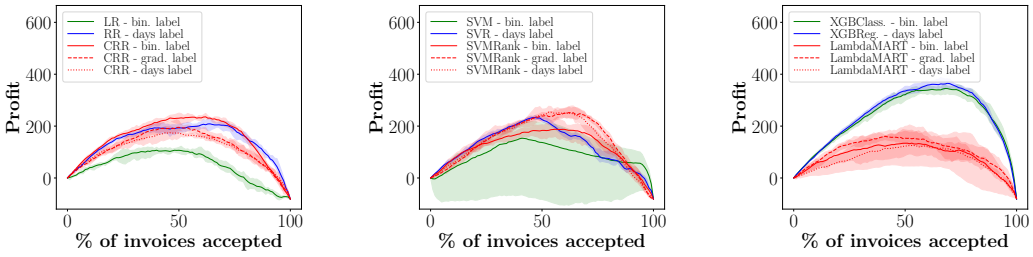
In the previous two sections, we compared the three prediction tasks according to a plethora of metrics and plots and saw different methods excel for different metrics. Ideally, we have one clear metric to evaluate the different prediction tasks within

the three method families, as the values of the previous experiment's metrics did not always coincide. Since our main goal is to support a spot factoring company, we would like to evaluate the effect in practice of a certain method. We therefore assign a decreasing profit to every graded relevance level in the dataset, showing the higher costs for increasingly worse invoices: 1 for *good*, 0.8 for *acceptable*, -10 for *undesirable*, -20 for *bad* and -50 for *horrible* invoices. These values reflect realistic proportions and illustrate the fact that a spot factoring dataset typically has a lot of good invoices - 'good' and 'acceptable' make up for more than 90% of the invoices in our dataset - with a relatively small profit margin (namely the discount that was applied to the invoice amount) and a few bad invoices that have big costs due to e.g. extra efforts of the factor, delays in (or lack of) repayment and claim costs. The profit values show this imbalance, e.g. the loss due to one horrible invoice nullifies the profits of 50 good invoices.

In Figure 5, the profit curves for the different prediction tasks and target labels are shown per method family. For the generalized linear model family, CRR with binary labels shows the highest possible profit, which corresponds to the AUROC results. Its curve is overtaken by the curve of ridge regression for high percentages of invoices accepted though. For the support vector machines, the SVMRank profit curves for graded and days target labels show the highest profits. In the previous experiment, these methods scored the best or second-to-best values across all support vector machines as well. Nevertheless, for higher percentages of invoices accepted, the profit decreases fast for SVMRank and the SVM and SVR curves outperform it. This seems to correspond with our goal of selecting a small portion of bad invoices. The variance of the SVM curve is again exceptionally high though. The LambaMART rankers seem to perform noticeably weaker than their CRR and SVMRank counterparts. This is to be expected, since the latter methods optimize the full range of ranks whereas LambdaMART employs logarithmically lower penalties for errors in the lower ranks. While the results in Table 1 show similar results for SVR and the classifier and regressor with XGBoost, the latter seems to be far superior here. The regressor with XGBoost even has a slightly higher profit than the classifier for the full range of possible percentages of invoices accepted. The good performance of tree-based methods is not suprising, as tabular data (such as spot factoring data) consist of possibly very different types of features that benefit from the axis-aligned splits and consequent ability for feature selection that these methods have to offer.

We do note that SVMRank with days labels took extreme amounts of computation time (5+ days). In this respect, non-ranking methods may be preferred even though the task is inherently a ranking task.

Overall, regressors give consistently good results across all three plots. They achieve high profits in both the generalized linear models and support vector machines families and are associated with the highest possible profit overall in the gradient boosted trees family. This experiment also shows that, if possible, evaluation w.r.t. profit is to be preferred to traditional measures, as the latter may not capture the impact of the business properly.



(a) Linear models (b) SVMs (c) Gradient-boosted trees

Figure 5.: Profit curves for different method families when performing different prediction tasks. The shaded regions represent the ± standard deviation of the curves. (LR = Logistic Regression, RR = Ridge Regression, CRR = Combined Regression and Ranking, RF = Random Forest, RFR = Random Forest Regressor, SVM = Support Vector Machine, SVR = Support Vector Regressor)

5. Discussion

While the research goals in related invoice discounting works are plentiful - ranging from investigating determinants of probability of default (Dorfleitner et al., 2017) to the effect of using economic variables (Zhang & Thomas, 2015), our work specifically addresses the suitability of methodologies and learning tasks. In this regard, the work of Perko (2017) investigating several methods for short-term probability of default is similar, although limited to the setting of a fixed number of overdue days (binary classification). From a methods point of view, their multi-layer perceptron neural nets perform better than the SVMs and no tree-based ensembles were used. On our data random forests clearly have an edge over MLPs and SVMs. From a methodological point of view, our study of the use of learning-to-rank and regression for credit risk estimation are also applicable in that setting.

Tater et al. (2018) also focussed on binary classification, in the context of an Accounts Payables problem, and found boosted trees to be the top performer as well. Even after limiting the amount of features, their highest-scoring cost-insensitive method is a tree-based ensemble (i.e. Balanced Bagging Gradient Boosting). The good performance of tree-based ensembles (more specifically Random Forest) in an invoice risk estimation context is also clear in the work of Hu (2015). Both Random Forest (with an AUROC of 0.86) and XGB methods (with an AUROC of 0.85) have the best results as well in the Accounts Receivables paper by Appel et al. (2019), where also binary classification is considered.

Other works that were described in Section 2 do not allow for direct comparison because of different assumptions, i.e. we assume all training examples have a known outcome while in survival analysis some instances have unknown outcome after a point in time. This could be relevant when also considering invoices that are still 'open' during training, that is, accepted some time ago but with as-of-yet unknown outcome.

6. Conclusion

In this paper, we studied the suitability of three prediction tasks for spot factoring, a context that requires estimating when an invoice will be paid rather than if it will be paid duly. We distinguished between the classic binary classification task, the regression of amount of days overdue and a Learning-to-Rank task ranking according to an invoice's binary, graded or integer-valued relevance. We compared the tasks using a spot factoring dataset and evaluated them for three method families from a classic probability of default (PD) perspective, the different task-based perspectives and a profit-based perspective. Regression methods achieved consistently high scores across the evaluation measures and for all method families. The profit-based evaluation identified the regressor with XGBoost as an outspokenly good performer. We recommend evaluating from a profit-based perspective in a spot factoring context, as it is easily interpretable, consolidates the business objective and addresses the different possible costs associated with different invoice outcomes.
This paper documented the first research effort investigating the applicability of predictive machine learning methods in a spot factoring context. It considered three different learning tasks and method families and showed the superior performance of regressors and - to a lesser extent - Learning-to-Rank methods compared to binary classifiers, i.e. the preferred methods in the related task of Probability-of-Default estimation. Furthermore, it identified profit-driven evaluation as a unifying metric.
We performed this research using the spot factoring dataset that was available to us. If more datasets come available - if only due to the increasing popularity of spot factoring and open-source datasets - the application of the methods described in this paper to new datasets makes for an interesting comparison. Also, we chose to focus on classifiers, regressors and rankers in this paper. Other possibilities and potential future research directions include the use of ordinal regressors and outlier detectors.

Future Work In the last experiment we used a cost-based technique to evaluate the different prediction tasks. To optimally account for different costs, we would like to include *cost-sensitive machine learning* (Elkan, 2001; Sheng & Ling, 2006) in the training phase as a next step in estimating financial risk.
While we chose to focus on classification, regression and Learning-to-Rank methods in this paper, we would like to broaden our scope towards other methods such as *ordinal regression* in the future.
By training our predictive model using two years of data, we silently assumed that there exists a static relationship between the independent and dependent variables. However, spot factoring involves a lot of non-stationary factors that might cause a static machine learning model to deteriorate over time, such as changing macro-economic situations, human deception of fraudulent customers and an expansion of a factor's customer base towards new countries. We will therefore look into solutions that deal with *concept drift* (Žliobaitė, Pechenizkiy, & Gama, 2016).
Finally, all of the experiments in this paper use a dataset consisting of invoices that the spot factoring company has already accepted. The distribution of this dataset does not necessarily correspond to the true distribution of submitted invoices at a spot factoring company. To be able to include the information of rejected instances in our dataset and possibly identify missed opportunities, we will investigate the suitability of machine learning methods for *reject inference* as an exciting future research opportunity.

Acknowledgements

To run our experiments, we used the infrastructure provided by the VSC Flemish Supercomputer Center. This project was funded by the Institute for the encouragement of Scientific Research and Innovation of Brussels (Innoviris, 2017-COTEAM-UP-19).

References

Alves, B. C., & Dias, J. G. (2015, May). Survival mixture models in behavioral scoring. *Expert Systems with Applications*, *42*(8), 3902–3910.

Appel, A. P., Oliveira, V., Lima, B., Malfatti, G. L., de Santana, V. F., & de Paula, R. (2019, December). Optimize Cash Collection: Use Machine learning to Predicting Invoice Payment. *arXiv:1912.10828*.

Baesens, B., Rösch, D., & Scheule, H. (2016). *Credit risk analytics.* John Wiley & Sons, Inc.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003, June). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, *54*(6), 627–635.

Baourakis, G., Conisescu, M., van Dijk, G., Pardalos, P. M., & Zopounidis, C. (2009, August). A multicriteria approach for rating the credit risk of financial institutions. *Computational Management Science*, *6*(3), 347–356.

Basel Committee on Banking Supervision. (2006, June). *International Convergence of Capital Measurement and Capital Standards.* Bank for International Settlements.

Berger, A., Gebhardt, A., Müller-Hannemann, M., & Ostrowski, M. (2011). Stochastic Delay Prediction in Large Train Networks. , 12 pages.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Boschetti, P. J. J. H. B. S. L. M. A. (2016). *Python: Real World Machine Learning.* Place of publication not identified: Packt Publishing.

Boyd, K., Eng, K. H., & Page, C. D. (2013). Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals. In D. Hutchison et al. (Eds.), *Advanced Information Systems Engineering* (Vol. 7908, pp. 451–466). Berlin, Heidelberg: Springer Berlin Heidelberg.

Brownstein, H. (2003, April). The Evolution of Factoring. *ABF Journal*, 1–3.

Burges, C. J. (2010, June). *From RankNet to LambdaRank to LambdaMART: An Overview* (Tech. Rep. No. MSR-TR-2010-82).

Carling, K., Jacobson, T., Lindé, J., & Roszbach, K. (2007, March). Corporate credit risk modeling and the macroeconomy. *Journal of Banking & Finance*, *31*(3), 845–868.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* (pp. 785–794). San Francisco, California, USA: ACM Press.

Dikkers, H., & Rothkrantz, L. (2005). Support vector machines in ordinal classification: An application to corporate credit scoring. *Neural Network World*, *15*(6), 491–507.

Dirick, L., Claeskens, G., & Baesens, B. (2017, June). Time to default in credit scoring using survival analysis: a benchmark study. *Journal of the Operational Research Society*, *68*(6), 652–665.

Dorfleitner, G., Rad, J., & Weber, M. (2017). Pricing in the online invoice trading market: First empirical evidence. *Economics Letters*, *161*, 56–61.

Elkan, C. (2001, August). The Foundations of Cost-Sensitive Learning. *IJCAI'01 Proceedings of the 17th international joint conference on Artificial intelligence*, *2*, 973–978.

Elliott, J., & Curet, O. (1999). Invoice Discounting - A Strategic Analysis Using Case-Based Reasoning. *Milne, R., Macintosh, A. and Bramer, M. (Eds.) Applications and Innovations in Expert Systems VI, Springer-Verlag, London.*.

Fawcett, T. (2006, June). An introduction to ROC analysis. *Pattern Recognition Letters*,

27(8), 861–874.

García, F., Giménez, V., & Guijarro, F. (2013, April). Credit risk management: A multicriteria approach to assess creditworthiness. *Mathematical and Computer Modelling*, *57*(7-8), 2009–2015.

Global Financial Stability Report: Moving from Liquidity- to Growth-Driven Markets. (2014, April). International Monetary Fund.

Hand, D. J. (2005, September). Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society*, *56*(9), 1109–1117.

Hand, D. J., & Henley, W. E. (1997, September). Statistical Classification Methods in Consumer Credit Scoring: a Review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *160*(3), 523–541.

Hirk, R., Hornik, K., & Vana, L. (2018, August). Multivariate ordinal regression models: an analysis of corporate credit ratings. *Statistical Methods & Applications*, 1–33.

Hu, P. (2015). Predicting and improving invoice-to-cash collection through machine learning. *Massachusetts Institute of Technology*.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02* (pp. 133–142). Edmonton, Alberta, Canada: ACM Press.

Kalara, N., & Zhang, L. (2018). *The changing landscape of firm financing in Europe, the United States and Japan.* CPB Netherlands Bureau for Economic Policy Analysis.

Kecman, P., & Goverde, R. M. P. (2015, February). Online Data-Driven Adaptive Prediction of Train Event Times. *IEEE Transactions on Intelligent Transportation Systems*, *16*(1), 465–474.

Kim, K.-j., & Ahn, H. (2012, August). A corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach. *Computers & Operations Research*, *39*(8), 1800–1811.

Knight, W. R. (1966, June). A Computer Method for Calculating Kendall's Tau with Ungrouped Data. *Journal of the American Statistical Association*, *61*(314), 436–439.

Leow, M., & Crook, J. (2016, March). The stability of survival model parameter estimates for predicting the probability of default: Empirical evidence over the credit crisis. *European Journal of Operational Research*, *249*(2), 457–464.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015, November). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124–136.

Marković, N., Milinković, S., Tikhonov, K. S., & Schonfeld, P. (2015, July). Analyzing passenger train arrival delays with support vector regression. *Transportation Research Part C: Emerging Technologies*, *56*, 251–262.

Milinković, S., Marković, M., Vesković, S., Ivić, M., & Pavlović, N. (2013, April). A fuzzy Petri net model to estimate train delays. *Simulation Modelling Practice and Theory*, *33*, 144–157.

Noh, H., Roh, T., & Han, I. (2005, May). Prognostic personal credit risk model considering censored information. *Expert Systems with Applications*, *28*(4), 753–762.

Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., ... Anguita, D. (2017). Delay Prediction System for Large-Scale Railway Networks Based on Big Data Analytics. In P. Angelov, Y. Manolopoulos, L. Iliadis, A. Roy, & M. Vellasco (Eds.), *Advances in Big Data* (Vol. 529, pp. 139–150). Cham: Springer International Publishing.

Perko, I. (2017, March). Behaviour-based short-term invoice probability of default evaluation. *European Journal of Operational Research*, *257*(3), 1045–1054.

Sculley, D. (2010). Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10* (pp. 979–988). Washington, DC, USA: ACM Press.

Sheng, V. S., & Ling, C. X. (2006, July). Thresholding for Making Classifiers Cost-sensitive. *AAAI'06 Proceedings of the 21st national conference on Artificial intelligence*, *1*, 476–481.

Soufani, K. (2002, January). The decision to finance account receivables: the factoring option.

Managerial and Decision Economics, *23*(1), 21–32.

Swiderski, B., Kurek, J., & Osowski, S. (2012, January). Multistage classification by using logistic regression and neural networks for assessment of financial condition of company. *Decision Support Systems*, *52*(2), 539–547.

Tater, T., Dechu, S., Mani, S., & Maurya, C. (2018). Prediction of Invoice Payment Status in Account Payable Business Process. In C. Pahl, M. Vukovic, J. Yin, & Q. Yu (Eds.), *Service-Oriented Computing* (Vol. 11236, pp. 165–180). Cham: Springer International Publishing.

Thomas, L. C. (2000, April). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, *16*(2), 149–172.

Van Gestel, T., Baesens, B., Van Dijcke, P., Suykens, J., & Garcia, J. (2005). Linear and non-linear credit scoring by combining logistic regression and support vector machines. *The Journal of Credit Risk*, *1*(4), 31–60.

Verbeke, W., Baesens, B., & Cristián, B. (2018). *Profit-Driven Business Analytics*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Yaghini, M., Khoshraftar, M. M., & Seyedabadi, M. (2013, April). Railway passenger train delay prediction via neural network model: RAILWAY PASSENGER TRAIN DELAY PREDICTION. *Journal of Advanced Transportation*, *47*(3), 355–368.

Zdravevski, E., Lameski, P., & Kulakov, A. (2011, July). Weight of evidence as a tool for attribute transformation in the preprocessing stage of supervised learning algorithms. In *The 2011 International Joint Conference on Neural Networks* (pp. 181–188). San Jose, CA, USA: IEEE.

Zeng, S., Melville, P., Lang, C. A., Boier-Martin, I., & Murphy, C. (2008). Using predictive analysis to improve invoice-to-cash collection. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08* (p. 1043). Las Vegas, Nevada, USA: ACM Press.

Zhang, J., & Thomas, L. C. (2015). The effect of introducing economic variables into credit scorecards: an example from invoice discounting. , *9*(1), 57–78.

Žliobaitė, I., Pechenizkiy, M., & Gama, J. (2016). An Overview of Concept Drift Applications. In *Big Data Analysis: New Algorithms for a New Society* (Vol. 16, pp. 91–114). Cham: Springer International Publishing.

Appendix A. Full result table of RQ2

Evaluation label:	**Binary**		**Graded**		**Days**			
	AUROC	AvP.	K' τ	S' ρ	MSE	R^2	K' τ	S' ρ
GLMs								
class.: LR (bin.)	0.594±0.023	0.119±0.014	0.098±0.021	0.123±0.026	/	/	0.09±0.019	0.128±0.026
rank.: CRR (bin.)	0.691±0.028	0.17±0.021	0.18±0.021	0.227±0.026	/	/	0.169±0.019	0.239±0.027
rank.: CRR (grad.)	0.65±0.028	0.151±0.021	0.259±0.028	0.326±0.035	/	/	0.243±0.022	0.342±0.03
rank.: CRR (days)	0.645±0.022	0.156±0.024	0.246±0.024	0.31±0.031	/	/	0.249±0.02	0.348±0.028
regr.: RR (days)	0.679±0.031	0.182±0.024	0.222±0.023	0.279±0.029	300.785±103.315	-0.005±0.108	0.216±0.02	0.302±0.028
MLPs								
regr.: MLPClassifier (bin.)	0.606±0.011	0.129±0.013	0.155±0.011	0.196±0.014	/	/	0.157±0.015	0.221±0.021
regr.: MLPRegressor (days)	0.593±0.012	0.126±0.01	0.19±0.01	0.24±0.013	277.287±24.76	0.023±0.007	0.194±0.012	0.272±0.017
SVMs								
class.: SVC (bin.)	0.632±0.12	0.195±0.06	0.117±0.102	0.135±0.117	/	/	0.112±0.1	0.145±0.13
rank.: SVMRank (bin.)	0.662±0.039	0.166±0.027	0.13±0.029	0.164±0.036	/	/	0.127±0.032	0.179±0.046
rank.: SVMRank (grad.)	0.686±0.023	0.166±0.018	0.353±0.014	0.443±0.018	/	/	0.332±0.014	0.458±0.019
rank.: SVMRank (days)	0.68±0.029	0.16±0.024	0.315±0.027	0.396±0.033	/	/	0.32±0.032	0.438±0.039
regr.: SVR (days)	0.658±0.024	0.171±0.025	**0.365±0.016**	**0.461±0.02**	306.517±100.126	-0.025±0.012	**0.368±0.014**	**0.498±0.018**
GBTs								
class.: XGBClas. (bin.)	0.783±0.029	**0.346±0.058**	0.23±0.031	0.287±0.038	/	/	0.218±0.024	0.306±0.033
rank.: λMART (bin.)	0.628±0.048	0.158±0.031	0.093±0.054	0.114±0.064	/	/	0.094±0.049	0.131±0.068
rank.: λMART (grad.)	0.645±0.058	0.171±0.052	0.205±0.071	0.252±0.089	/	/	0.199±0.066	0.275±0.091
rank.: λMART (days)	0.616±0.072	0.152±0.045	0.075±0.062	0.093±0.076	/	/	0.069±0.061	0.097±0.084
regr.: XGBReg. (days)	**0.784±0.029**	0.311±0.037	0.353±0.02	0.437±0.025	**265.844±82.86**	**0.12±0.064**	0.346±0.016	0.47±0.021

Appendix B. Table with experiment hyperparameters

Method	Hyperparameters
GLMs	
LR	**C**: {0.1,0.2,0.3,0.4,0.5,0.6,0.7}
	solver: {'newton-cg'}
	max_iter: {400, 800}
CRR	**learner_type**: {'pegasos', 'sgd-svm', 'passive-aggressive', 'margin-perceptron', 'romma', 'logreg-pegasos'}
	loop_type: {'stochastic', 'balanced-stochastic', 'rank', 'roc' 'query-norm-rank','combined-ranking', 'combined-roc'}
	eta_type: {'basic', 'pegasos', 'constant'}
	prediction_type:{'linear', 'logistic'}
	dimensionality:{118, 131072}
	iterations: {100000}
	lmbd: {0.00001, 0.001, 0.3, 0.5}
	rank_step_probability: {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9}
RR	**alpha**: {1e-15, 1e-10, 1e-8, 1e-4}
SVMs	
SVC	**C**: {0.1,1}
	class_weight: {'balanced'}
	gamma: {1e-11, 1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5}
SVMRank	**c**:{1e-5, 1e-6}
	e:{0.5}
	t:{0,1}
SVR	**C**: {0.001, 0.1,1}
	gamma: {1e-16, 1e-15, 1e-14, 1e-13, 1e-11}
GBTs	
XGBClass.	**max_depth**: {3,5,7,9,11}
	learning_rate: {0.01, 0.03, 0.75, 1, 1.1}
	n_estimators: {50, 100, 150, 200}
	objective: {'binary:logistic'}
λMART	**tree**:{50, 125, 500, 1000, 2000}
	leaf:{10,30,50}
	shrinkage:{0.05,0.1,0.2,0.4,0.8}
	estop:{50,100,150}
	metric:{'NDCG@5216'}
XGBReg.	**max_depth**: {3,5,7,9,11}
	learning_rate: {0.01, 0.03, 0.75, 1, 1.1}
	n_estimators: {50, 100, 150, 200}
	objective: {'reg:squarederror'}
MLPs	
MLPClassifier	**hidden_layer_sizes**: {(), (40,),(80,),(120,),(140,),(400,),(800,)}
	activation:{'logistic','tanh','relu'}
	alpha:{1-e1, 1-e2, 1-e3, 1-e4, 1-e5, 1-e6}
	max_iter: {400, 800, 1200}
	warm_start:{True,False}
MLPRegressor	**hidden_layer_sizes**: {(),(40,),(80,),(120,),(140,),(400,),(800,)}
	activation:{'logistic','tanh','relu'}
	alpha:{1-e1, 1-e2, 1-e3, 1-e4, 1-e5, 1-e6}
	max_iter: {400, 800, 1200}
	warm_start:{True,False}