Prediction of Invoice Payment Status in Account Payable Business Process

Tarun Tater(✉), Sampath Dechu(✉), Senthil Mani(✉), and Chandresh Maurya(✉)

IBM Research AI, Bengaluru, India
{ttater03,sampath.dechu,sentmani,cmaurya1}@in.ibm.com

Abstract. Account payables are amount owed to vendors for goods and services delivered to a company. Vendors raise invoices which go through several processing steps before they are paid by a company. Companies have contractual obligations with vendors for paying the invoices within a stipulated time. Invoices that exceed this time attract penalty and affect vendor satisfaction to work with the company. It is very critical for large firms dealing with thousands of vendors for their day to day operations to meet the service level agreements with vendors to avoid penalties. Any assistance for practitioners, warning them of potential invoices that can breach the service level agreements, can help them in minimizing the penalties. In this research, we model the problem of identifying delayed invoices as a supervised classification task. There are three characteristics of this problem which are challenging from a classification perspective: (i) the status of an invoice is affected by other invoices that are simultaneously being processed, as there are limited resources to process the huge volume of invoices, (ii) feature engineering to capture the temporal aspect of the invoice and having the optimal representation of the multiple data entries created per invoice, and (iii) the number of paid late invoices are much smaller in percentage compared to paid on time invoices in the training data set, hence the classes are imbalanced. The results obtained by training an ensemble of classifiers show that penalties can be avoided on more than 82% of the invoices which are currently being penalized.

Keywords: Account payables · Vendor invoice management
Business process optimization · Predictive monitoring

1 Introduction

Procure to Pay (P2P)[1] is a business process that integrates functionalities of purchasing and Accounts Payables (AP) departments in large enterprises. Invoice processing is a cumbersome process involving both manual and automated steps. *Vendor* raises an invoice once he/she supplies goods/services to a *company*. It is

[1] https://wiki.scn.sap.com/wiki/pages/viewpage.action?pageId=14508.

common for large enterprises to deal with thousands of vendors and pay millions of invoices per year. Companies have contractual agreements with the vendors to pay invoices in a stipulated time. *Paid Late* invoices attract penalties and strain the relationship between the company and the vendor. In an independent market research [1] across 500 accounts payable (AP) departments, conducted in UK in 2015, the top concern expressed by participants was mitigating the *paid late* invoices.

An invoice goes through several stages of scrutiny before it gets paid by the account payables department. Large enterprises have dedicated teams for invoice processing and payments. These teams are equipped with enterprise work flow tools such as SAP VIM[2] and similar ERP applications to ease the orchestration of invoice processing, involving multiple levels of sanctity checks to validate the invoice for payment. Most of the process work flow tools log the actions performed on the invoice, the actor who performed the actions, start and end time for every action taken, as change logs or event logs such as those generated by SAP VIM. Companies have organizational roles such as *Functional process owner*, *Regional process owner*, *Global process owner* who use these logs to check on the health of the process, identify bottlenecks and take actions to improve process efficiency and efficacy. There could be multiple reasons for an invoice to get delayed for payment:

1. *Information Mismatch* between Vendor's provided details & Invoice details.
2. *Limited Resources* available for processing the invoice leading to high priority invoices delaying the processing of low priority invoices.
3. *Invoice Expiry*, which is the number of days remaining for the invoice to be processed also affects the priority and assignment of resources to process the invoice.

Identifying invoices which may be delayed is a laborious task due to the large volume of invoices and their attributes. Some of the mentioned problems such as *information mismatch* can be addressed by building software capabilities like data validation in the invoice management system. However, such validation which might require re-work by vendors, does not impact the due payment date of the invoice. Given such constraints, we have approached the problem of predicting invoices' status ("Paid late" or "Paid on time") as a supervised binary classification task. We consider the invoice details and the logs of actions taken on an invoice to predict if the invoice is likely to be delayed and flag it for moderation by concerned people at early stages of processing. This would help in allocating resources appropriately to minimize the penalties being incurred due to delayed payments.

Predicting invoice payment status will enable the *process owners* to take remedial actions such as prioritizing the invoice for processing or re-negotiating contracts with vendors. In our scenario, the problem translates to placing greater emphasis on the *paid late* invoices. In our dataset, approximately 10.3% of the

[2] https://www.opentext.com/what-we-do/products/opentext-suite-for-sap/opentext-vendor-invoice-management-for-sap-solutions.

invoices are *paid late*. We focus on predicting the *paid late* class (minority class) with high precision as identifying the wrong invoices for moderation, will affect the processing of other invoices, resulting in a vicious cycle. Similarly, we need to have a high recall for the *paid late* class to avoid penalties on as many invoices as possible. The metric used for evaluation in related work [16] which is *accuracy*, can be misleading [5] in our scenario. For example, if we simply predict all invoices to be paid on time, then we will report a 90% accuracy, but we would not have addressed the problem of identifying *paid late* invoices. So, we use metrics like precision and recall for the *paid late class*, and F1-score for the classifier.

We use an ensemble of classifiers and achieve a precision of 89.3% and recall of 82.7% on the *paid late* invoices.

The key contributions of our research are:

1. Using machine learning to predict the payment status of invoices to minimize the penalties incurred due to the invoices being delayed. Hence, our predictions can enable the process owners to pro-actively work on flagged invoices rather than rely on teams monitoring invoice processes or conduct time consuming analysis on each invoice.
2. Modeling categorical features in a domain which has historical (temporal) information as numerical features. This reduces the feature space considerably (from ∼1900 to 88) in a way that all the unique values in a category are replaced by few extra columns for each row. Otherwise, if one hot-encoding or indexing is done on categorical features, it would result in ∼1885 features.
3. We propose and evaluate an ensemble approach for invoice late payment prediction, encompassing supervised learning algorithms like Random forest and Boosted Trees which are better suited for categorical features and SVM and Logistic Classification better suited for numerical data.

We discuss prior art in Sect. 2, methodology in Sect. 3, empirical evaluation in Sect. 4 and a better way to represent the categorical type features instead of one-hot encoding for historical analysis of invoices in Sect. 5.

2 Related Work

In this section, we present some related work that has been done in the invoice prediction domain. Zeng et al. [16] tackle the problem of invoice outcome prediction in *Accounts Receivable* (AR) case. They formulate the prediction problem as supervised classification problem and apply the existing classifiers (C4.5, Naive Bayes etc.) to it. Our work differs from theirs in following ways: (1) we tackle the problem in the Account Payables (AP) case while they tackle it in the AR case. Secondly, they report *Accuracy* as a metric for class-imbalance classification problem which is not a suitable metric of choice in this setting. Instead, we report the metrics which are better suited for class-imbalance setting.

Smirnov et al. [12] models the invoice late payment time by survival analysis and an ensemble of random survival forest on real data and show that random survival forest performs better when combined with historical data of repeated debtors. Hu et al. [6] discuss prediction of invoice payment and improvement in the process of collection. They use various supervised algorithms such as decision tree, random forest, logistic regression, SVM, and cost-sensitive learning for prediction and conclude that random forest outperform the other methods. Additionally, their data set has paid late invoices as the majority class while in our dataset, paid late is the minority class. Along similar lines, Hu et al. [7] use supervised learning algorithms for invoice payment prediction.

An important point to note is that all the work discussed in this section solve the problem of invoice prediction in the AR case. From our literature survey, we find one work that solves the invoice prediction in the AP case. Younes et al. [15] attempt to address the problem of invoice processing time, understanding the delinquent invoices and the impact of delay in the invoice processing. They use integrated lean-manufacturing and discrete event simulation as the first approach and Markov chain modeling as the second approach for minimizing the overdue invoices in the AP case. Lean manufacturing borrows the idea from assembly line scheduling for managing the invoices and show encouraging results via simulation. The second approach that uses Markov process theory assumes each service station as node in the Markov graph and compute the transition probability from one node to another node as the service time of the invoice. From a monitoring business processes perspective, Meroni et al. [9] discuss an artifact based approach. Cabanillas et al. [3] also discuss predictive task monitoring to signal and control possible misbehaviors at runtime in business processes.

3 Methodology

In this section, we present our approach as illustrated in Fig. 1 to predict the invoice payment status. We predict the invoice status at 2 logical time steps in the complete process. One, when the invoice comes in the system, and there is only invoice specific data present and no process data on that invoice. From a business process perspective, this would help in flagging an invoice from the start if its likely to be delayed. This prediction would primarily be based on invoice specific features like amount, number of days allocated for invoice payment and vendor details. Another prediction is done three days (variable hyper-parameter) before the invoice is due. This would flag the invoice if its likely to be delayed giving enough time to the processing team to expedite the invoice processing.

3.1 Data Preprocessing

We obtained the invoice process data from one of our large clients, which process around 200 thousand invoices across multiple vendors per month. We had access to two sets of data - basic invoice details, when it was raised in the system for processing and the associated change log consisting of all the process actions taken on the invoice. In total, we had 523147 invoices and associated 3.5M log data. The invoice value ranged from less than a dollar to $\sim$236M\$. We observed

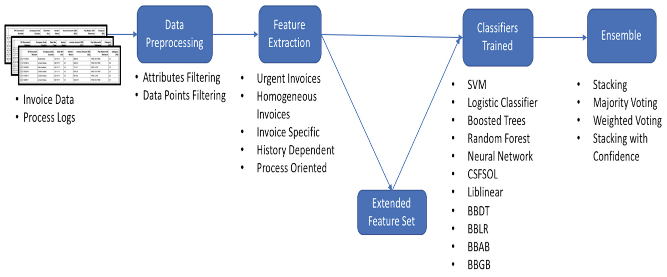

Fig. 1. Approach to predict late payment of invoices

around 7% of invoices as rated urgent, and around 9% of paid late invoices. These paid late invoices were occurring for less than 50% of the vendors. The summary stats of the data set is presented in Table 1, column 2. As illustrated in Fig. 2(a), there was always a steady flow of urgent invoices across the year, and the invoices that were paid late were also observed consistently across the time period (Fig. 2(b)). We now discuss the various data preprocessing techniques applied:

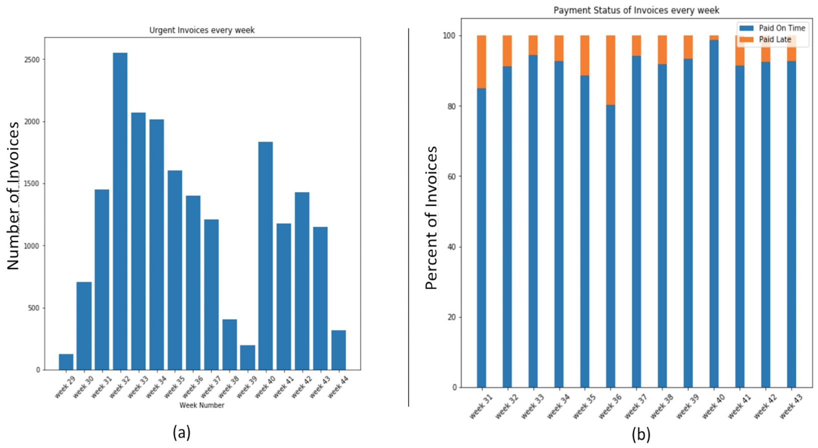

Fig. 2. Urgent invoices and invoice payment statistics

1. **Attributes Filtering:** There are 47 attributes describing each invoice with information regarding vendors, their demographics, raw materials, amount payable and the deadline for the payment. We removed attributes which uniquely identify an invoice such as "Document Id". Also, attributes containing information populated after the invoice was created, which if considered, could bias the prediction of labels i.e., "Paid Late" or "Paid on Time"

Table 1. Summary statistics of our data set

Attribute	Before preprocessing	After preprocessing
Number of invoices	523147	282622
Number of attributes	47	16
Number of paid late invoices	49835 (9.52%)	29221 (10.33%)
Number of paid on time invoices	473312 (90.5%)	253401 (89.6%)
Number of urgent invoices	7.04%	7.67%
Number of vendors	11532	8527
Number of countries	47	47
Categories of raw materials	10	10
Maximum amount of invoice (in $)	236 million	236 million
Mean amount of invoices (in $)	4352	4180

were filtered. Finally, we removed attributes which did not show any variation across invoices. This retained 16 out of the initial 47 attributes.

2. **Data Points Filtering:** We removed data points which had incorrect, misleading or missing values for attributes like “vendor name”, “invoice amount”, “company code”, “due date” and “posting date”. Also, some invoices had amount less than 0 which meant that the invoice was a “credit” invoice i.e. the payment was done before the raw materials were procured. Some invoices had due date prior to the posting date of the invoice making such invoices delayed even before the processing had started. Few invoices did not have matching data in the process logs and hence the information related to the processing was missing. After removing all such erroneous data points, we had 282622 invoices. The summary stats of the processed data set is listed in Table 1, column 3.

3.2 Feature Extraction

In this section we describe the feature engineering approach and the resulting features extracted to train the classifiers.

As discussed in the introduction, there are several reasons for an invoice to be delayed. We concentrate on extracting features which tackle all these challenges. The processing time of an invoice depends on invoice specific attributes as well as on other invoices which are being processed concurrently, as they are competing for the same scarce resource (humans for manual tasks). The physical analogy can be that of vehicular traffic - more cars on the road will strongly correlate to most of the cars being delayed. Similarly, if there are high number of invoices in the system at a given time and each requiring multiple steps, this may create bottleneck at few places in the process leading to delays. On the other hand, lesser invoices may speed up the processing time. Inspired by Senderovich et al. [11], we categorize the features broadly based on inter-case and intra-case invoices:

Table 2. Features & their types used for predicting invoice status

Feature class	No.	Feature & type	Description
Urgent invoices	1	urgent_invoices_due (int)	Number of urgent invoices due on the day of prediction
	2	urgent_overall (int)	Number of urgent invoices which were due over the entire duration of an invoice from "posting date" to "due date"
Homogeneous invoices	3	load_invoices (float)	Number of invoices which were posted between the days of *date posting* and *due date*
	4	actions_load_invoices (float)	Number of actions (process steps) taken on all invoices between the days of *date posting* and *due date*
	5	num_invoice_due_previous_day (int)	Number of invoices that were due the previous day before an invoice was due
Invoice specific	6	number_of_days (int)	Number of days between *date posting* and *due date*
	7	number_of_working_days (float)	Number of weekdays between *date posting* and *due date*
	8	number_of_invoices_due (float)	Number of invoices which are due when an invoice is posted
	9	number_of_outstanding_invoices (float)	Number of invoices which are due and already delayed
	10	ratio_paid_late_outstanding (float)	Ratio of 8 and 9
	11	invoices_amount (in dollars) (float)	The amount due for this invoice
	12	vendor_name (category)	Vendor name who has submitted the invoices
	13	company_code (category)	Division of the multi-national company which requested the material
	14	company_code_country (category)	Country of the division of the multi-national company which requested the material
	15	document_type (category)	Type of submitted invoice
	16	raw_material (category)	Category type of the raw material
	17	commercial_area (category)	Commercial area of raw material
	18	payment_terms (category)	Terms dictating invoice payment
	19	invoice_amount_bucket (category)	Invoice amount bucketed in bins
History dependent	20	paid_late_invoices (float)	Number of invoices which were paid late before the *date posting* of an invoice
	21	paid_on_time_invoices (float)	Number of invoices which were paid on time before the *date posting* of an invoice
	22	ratio_paid_late_paid_on_time (float)	Ratio of 20 and 21
	23	percent_paid_late_vendor (float)	Percentage of invoices which were delayed to a vendor till the present date
Process oriented	24	action (category)	The action code taken on the invoice
	25	number_of_actions (int)	The total number of actions taken on the invoice
	26	number_of_days_before_first_action (int)	The number of days between the invoice is raised and first action is taken on the invoice

Inter-case Invoices

- **Case #1 Urgent Invoices:** Invoices which need to be paid in a day or two get preference over other invoices. If there are a bulk of such invoices coming constantly, it will adversely affect the other invoices which were to be processed and paid. So, we engineer the following two features:
  - Number of invoices which were due in 1–2 days(n), 3 days prior to particular invoice(p_i), since these n invoices may delay the processing for p_i invoice. (#urgent_invoices_due)
  - Number of such invoices which were supposed to be paid in 1-2 days(m) over the complete duration of a particular invoice. (#urgent_overall)

- **Case #2 Homogeneous Invoices:** For invoices other than urgent invoices, we consider that there is no priority between them. So, at a particular time, the processing time of an invoice would depend on the number of other invoices(#load_invoices) and the number of actions(#actions_load_invoices) being taken on them. It is a representative of the amount of load in the system when a particular invoice was being processed.

Intra-case Invoices

- **Case #3 Invoice Specific:** Some invoices may go through a longer process then others depending on multiple reasons such as the demographics, amount(#invoices_amount(in $)), information provided or missing, etc. The processing speed of the invoice will also depend on the number of days(#no_of_working_days) between the invoice posting date and the due date. This would mean that different invoices may be treated differently depending on these criteria. For example: between an invoice which needs to go through 5 stages and is due in 10 days and another invoice which is due only after 45 days, the earlier invoice would take precedence to ensure both the invoices are being paid on time.
- **Case #4 History Dependent:** We only consider invoices from vendors who have at-least 10 invoice payment transaction. We had this threshold since, it represents the importance of the relationship with the vendor based on the transaction history, and to have enough data points to consider history dependent features like #percent_paid_late_vendor. We also consider features which signify historical payment status of all invoices(#paid_late_invoices and #paid_on_time_invoices).
- **Case #5 Process Oriented:** Once the invoice is posted, it goes through multiple checks and steps before the invoice is paid. As discussed earlier, we predict the invoice payment status 3 days before it is due. So, we take into account the type of action(#action) and the total number of actions (#number_of_actions) performed on the invoice at the time of prediction.

To summarize, we have 26 features with 17 numerical and 9 categorical types across the five categories as listed in Table 2.

Table 3. Comprehensive list of various classifiers evaluated in our approach

Classifier	Description
SVM	A non-probabilistic binary classifier
Logistic classifier	Models the probability of the classes using the logit function
Boosted Tree	An ensemble of decision trees using gradient boosting
Random forest	Ensemble of decision trees combining ideas of random selection of features and bagging
Neural network	Trained a neural network with 3–8 layers with binary cross-entropy loss
Liblinear [4]	A classifier for solving large-scale regularized linear classification problems
Cost-sensitive first-order sparse online learning (CSFSOL) [13]	Online learning based algorithm for cost-sensitive learning on large-scale sparse data
BBDT	Balanced Bagging Decision Trees
BBLR	Balanced Bagging Logistic Regression
BBAB	Balanced Bagging Adaboost
BBGB	Balanced Bagging Gradient Boosting

3.3 Classifiers Used

We used a supervised learning approach to train the classifiers listed in Table 3. We evaluated each of these classifiers and used an ensemble on these classifiers to improve our results as different models will be better suited for different subsets of data [10]. We discuss the different approaches we evaluated for the ensemble:

1. Stacking [14]: Different classifiers such as Boosted Trees, Logistic classification, SVM were trained over the predictions of different classifiers giving each classifier an equal weight.
2. Plurality Voting (Most voted): The final prediction is the most predicted value amongst all the classifiers.
3. Weighted Voting: The predictions of each model are weighted according to the number of correct predictions made by them. So, the weight of each model is the accuracy the individual model has. This was tried both for overall accuracy as well as *paid late* accuracy.
4. Stacking with Confidence: Ensemble was trained on predictions from different models. Along with the predictions, the confidence scores from each classifier (wherever possible) are also considered as features.

4 Empirical Evaluation

In this section, we define the metrics and demonstrate the empirical evaluation of different machine learning models on invoice late payment prediction.

4.1 Metrics

Owing to the data imbalance in our case and contrary to the evaluation metrics used in some of the literature for invoice late payment prediction [16] (mostly *accuracy*), we aim to achieve high precision and reasonably high recall on paid late invoices (minority class) because no action is needed for paid on time invoices. High precision would mean that most of the invoices our approach labels as "paid late" are indeed "paid late". High recall here implies that our approach is able to detect majority of the invoices which are going to be "paid late". We report precision-recall (PR) curve rather than Receiver operating characteristic (ROC) curve because PR curve does not account for true negatives (TN) (as TN is not a component of both precision and recall) and would not be affected by the relative imbalance. The metric used for our evaluation are Precision, Recall, F1-score, Average precision (AP) score and Area under PR-curve (AUPRC).

4.2 Training

For evaluating our approach, we consider only those invoices which have a minimum of 10 days for payment as majority of the invoices (93%) are due only after 15 or more days from the date of posting. Also, for evaluating, we consider invoices only from vendors which have more than 10 transactions. This serves couple of purposes. First, this helps in concentrating on only vendors which are dependable which in turn implies the importance of the relationship with that vendor which may be because of the raw materials, cost or other demographics. Therefore, invoices from such vendors should be given attention to maintain this relationship. Secondly, it helps us identify additional features that capture the vendor behavior, e.g. the number of times payment is delayed to a vendor. We had a approximately 60:20:20 data split across train, test and validation. Since, our task is time dependent, we don't split according to approaches such as cross-validation. We split the data based on time. Invoices which are cleared before a date are considered for training and after that date in test data. The data split is shown in Table 4.

Table 4. Data split for evaluating our classifiers

	Training	Validation	Test
Paid on time	165721	41589	46091
Paid late	21112	5109	3200

4.3 Results

As discussed, we make predictions at 2 time steps, once when the invoice is raised and once 3 days before the due date of the invoice. The precision (P), recall

(R) and F1-score of the classifiers evaluated are listed in Table 5. In summary, “Boosted Trees” and “Random Forest” performed the best. Although, we did an extensive parameter search for these models, the recall of “paid late” class was poor across classifiers. We had 26 derived features out of which 9 were categorical type. Since, we had ∼1885 categorical values for these 9 categorical features, the best results were observed for decision tree based models namely random forest and boosted trees as the decisions at each node are based on values of the features. Inspired by Avati et al. [2], we tried to address this problem through Deep Learning. But the results were not satisfactory. Upon further analysis of results, we figured one of the major reasons could be the explosion of features while converting categorical features to numerical features. To avoid having an implicit ordering between the categorical features when converted to numerical features, the conversion was done using one-hot encoding instead of indexing the values. This meant that there were ∼1885 binary features out of ∼1900 features.

Table 5. Precision, Recall and F1-score of classifiers evaluated.

Prediction time	Method	F1-score (%)	Paid late (R) (%)	Paid late (P) (%)
Prediction when invoice raised	Boosted Trees	**95.84**	**39.93**	**91.10**
	Random forest	95.21	30.25	88.24
	Logistic classification	94.69	35.4	67.32
	SVM	95.48	36.18	86.41
	Neural network	94.38	35.34	61.80
Prediction 3 days before invoice due	Boosted Trees	**96.24**	**43.87**	**96.36**
	Random forest	95.63	37.25	89.08
	SVM	95.74	41.71	85.19
	Logistic classification	95.16	33.18	72.58
	Neural network	95.61	39.78	84.41

5 Extended Feature Set

To tackle the above mentioned feature explosion problem, we devised a better representation of these categorical features into numerical features based on the historical information present without resulting in the explosion in feature space. The representation is based on the intuition about, how these features would affect or factor into an individual’s analysis while processing these invoices. And also how the historical data flow about each categorical type feature serves a meaningful purpose for analysis. Our data comprised of categorical features like vendor details, country, and other such demographics. So, for each of the 9 categorical type features and their values, we derived following features which is a representation of their influence on payment status. For each invoice, until the date (d_1) of prediction (i.e. 3 days before the due date):

1. n_1 - Number of total invoices for particular value of that categorical feature prior to d_1
2. Percentage and number of invoices paid late out of n_1
3. Percentage and number of invoices paid on time out of n_1.

Further, for vendors, which had the most number of unique values (1459) among other categorical features, we considered a moving window to accommodate *seasonality* and change in the vendor's recent history of payment status. For example: if a vendor has 25 invoices, out of which 20 are paid late and 5 are paid on time, it might be that the last 5 invoices were paid on time. So, this seasonality was taken into account. We derived more vendor specific features which accounted for the payment status of vendor's previous 3 and previous 5 invoices. So, vendor names(#vendor_name) were represented as following features:

- Total number of invoices processed for the vendor prior to a date.
- Percentage and Number of times invoices were paid late and paid on time for the particular vendor prior to a date. This is to understand the previous record with a particular vendor.
- Percentage and Number of times invoices paid late and paid on time for the particular vendor during the last "n" $(3, 5)$ times. (seasonality)

Table 6. Precision (P), Average Precision (AP), Recall (R), and F1-score of classifiers evaluated with extended feature set.

Method	F1-score (%)	AP	Paid late (R) (%)	Paid late (P) (%)
Liblinear	97.58	0.65	64.41	97.59
CSFSOL	**98.40**	**0.77**	**78.24**	**96.57**
Boosted trees	96.74	0.54	51.96	96.18
Random forest	96.73	0.52	53.46	93.34
Logistic classification	97.56	0.65	68.80	91.50
SVM	97.95	0.70	77.8	89.31
BBDT	96.24	0.49	59.19	77.55
BBLR	**97.09**	0.64	**84.83**	74.15
BBAB	96.40	0.54	70.95	72.93
BBGB	98.01	0.72	81.86	86.77
Neural network	96.15	0.50	60.23	78.20

5.1 Experimental Testbed and Settings

In this section, we show the parameters and setting used in all the experiments. For Liblinear, command line options were (-s 5 -B 1 -e 0.001 -c .1 -w-1 4 -w1 1). That means, we train L_1-regularized l_2-loss SVM with a bias of 1 added to the training examples, parameter C is set to 0.1, weights given to negative and positive examples are 4 and 1 respectively. We train the SVM until error tolerance falls below 0.001. For training CSFSOL, we vary the parameter λ and η in the range $\{0.003, 0.09, 0.3, 1, 2, 4, 8, 16, 32, 64\}$ and $\{0.0312, 0.0625, 0.125, 0.25, .5, 1, 2, 4, 8, 16, 32\}$ respectively while weights are set to $(0.1, 0.9)$ for the negative and positive examples respectively. For training ensemble of classifiers based on Balanced Bagging approach default settings of Imblearn python package [8] is used.

For the Ensemble with confidence (SVM) used on all different models, we performed a grid search on validation dataset, the parameters which worked best were *penalty* (SVM) of 0.001 (penalty term on the misclassification loss of the model), *max iterations* as 300 and *class weights* were inversely proportional to the number of examples in the training data for each class. One of the intuitive reason behind using the low penalty value was that: for a model, higher the penalty, the model tries to maximize the margin for correctly classified examples. But, the aim was to correctly classify as many invoices as possible (Fig. 3).

5.2 Results

In this section, we discuss the result of our empirical evaluation of the methods used to predict paid late invoices with extended feature set (Table 6). *CSFSOL* gave the best result when an individual classifier is considered. The Ensemble classifier which performed the best was a SVM trained on predictions of all classifiers along with the confidence score wherever available (Table 7). We would like to emphasize the fact that F1-score is an overall performance score on both the classes.

Table 7. Results obtained on different metrics using the various ensemble models. AP = Average Precision P = Precision, and R = Recall

Method	F1-score (%)	AP	Paid late R (%)	Paid late P (%)
Majority voting	**96.30**	0.46	43.96	**98.04**
Weighted voting	98.04	0.72	79.28	89.45
Stacking	98.07	0.73	83.46	86.38
Stacking with confidence	**98.23**	**0.75**	**82.75**	**89.33**

The precision-recall curve shows the relationship between precision and recall for different thresholds. Our PR curve demonstrates that the Ensemble classifier with confidence (0.75) and CSFSOL (0.77) are better suited for our task as they

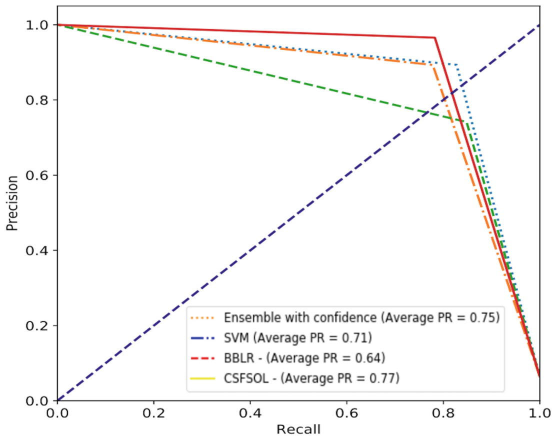

Fig. 3. PR curves for best results

have a greater area under the curve. And with 0.77 and 0.75 AP score, both precision and recall are reasonably good without affecting the other.

6 Discussion and Conclusion

From the results, we can observe that Ensemble with confidence and CSFSOL outperform other methods in terms of the metrics evaluated.

Table 8. Top 5 Influential Features across classifiers

Feature category	No.	Feature & type	Description
History dependent	1	paidLate_vendor_3 (int)	Number of times the payment was made late to a particular vendor out of the previous 3 times
History dependent	2	paid_on_time_vendor_5 (int)	Number of times the payment was made on time to a particular vendor out of the previous 5 times
Inter-case invoices	3	num_invoice_due_previous_day (int)	Number of invoices that were due the previous day before an invoice was due
History dependent	4	category_paid_late_3Days (int)	Number of times the payment was made late in a particular category out of the previous 3 times
Process oriented	5	process_id_4 (int)	Number of times the step 4 (step id - one of the steps invoices may go through) was done on the invoice

Influential Features: The top 5 influential features as shown in Table 8 prove that historical dependence and seasonality play a major role in deciding whether an invoice will be "paid on time" or "delayed". Further, how far the invoice has been in the process, along with number of concurrent invoices being processed, also affects the invoice payment status.

Generalizability: Based on our experience, most of the inter-case and intra-case features used for prediction are expected to be available for any account payables process, and collecting this data is very much feasible. The analysis, preprocessing, features or models have nothing specific to the dataset we have evaluated on. Hence, we can safely argue that most of our features and models are generalizable for other data sets obtained from client accounts. We are in the process of evaluating the same with few more client accounts data.

Future Work: Possible future work in this research includes, predicting the number of days by which an invoice would be delayed, suggesting advancing of processing of invoices in favor of other invoices such that the penalty (if any) on late payments is minimized. Also, if the amount of invoice is high, they should be treated separately since the penalty incurred on the delay would be higher. Finally, these is scope for identifying which process steps are most time consuming and provide suggestions on human resource management.

Implementation: We implemented the preprocessing of data using python2.7 and pandas library. Different libraries were used for different classifiers namely PyTorch for implementing neural networks, liblinear [4] for liblinear, imbalance-learn [8] for BBDT, BBLR, BBAB & BBGB. We implemented the CSFSOL algorithm in C++. We used scikit-learn libraries for SVM, logistic classification and boosted trees. The service to predict the payment status for a new invoice was hosted on a server using flask which is a python based framework.

References

1. http://www.cimaglobal.com/Documents/Thought_leadership_docs/white-paper-hub/2016-04-12-Invoice-Benchmark-Report.pdf. Concur (2016)
2. Avati, A., Jung, K., Harman, S., Downing, L., Ng, A., Shah, N.H.: Improving palliative care with deep learning. arXiv preprint arXiv:1711.06402 (2017)
3. Cabanillas, C., Di Ciccio, C., Mendling, J., Baumgrass, A.: Predictive task monitoring for business processes. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 424–432. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10172-9_31
4. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**(Aug), 1871–1874 (2008)
5. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
6. Hu, P.: Predicting and improving invoice-to-cash collection through machine learning. Ph.D. thesis, Massachusetts Institute of Technology (2015)
7. Hu, W.: Overdue invoice forecasting and data mining. Ph.D. thesis, Massachusetts Institute of Technology (2016)
8. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**(17), 1–5 (2017). http://jmlr.org/papers/v18/16-365
9. Meroni, G., Di Ciccio, C., Mendling, J.: An artifact-driven approach to monitor business processes through real-world objects. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 297–313. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_21

10. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1–2), 1–39 (2010)
11. Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., Maggi, F.M.: Intra and inter-case features in predictive process monitoring: a tale of two dimensions. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 306–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_18
12. Smirnov, J., et al.: Modelling late invoice payment times using survival analysis and random forests techniques. Ph.D. thesis (2016)
13. Wang, D., Wu, P., Zhao, P., Hoi, S.C.: A framework of sparse online learning and its applications. arXiv preprint arXiv:1507.07146 (2015)
14. Wolpert, D.H.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)
15. Younes, B.: A framework for invoice management in construction. University of Alberta, Canada (2013)
16. Zeng, S., Melville, P., Lang, C.A., Boier-Martin, I., Murphy, C.: Using predictive analysis to improve invoice-to-cash collection. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1043–1050. ACM (2008)