

Classifying process deviations with weak supervision

Peer-reviewed author version

LAGHMOUCH, Manal; JANS, Mieke & DEPAIRE, Benoit (2020) Classifying process deviations with weak supervision. In: 2020 2nd International Conference on Process Mining (ICPM), IEEE COMPUTER SOC, p. 89 -96.

DOI: 10.1109/ICPM49681.2020.00023

Handle: <http://hdl.handle.net/1942/34547>

Classifying Process Deviations With Weak Supervision

Manal Laghmouch
Faculty of Business Economics
UHasselt - Hasselt University
Hasselt, Belgium
manal.laghmouch@uhasselt.be

Mieke Jans
Faculty of Business Economics
UHasselt - Hasselt University
Maastricht University
Belgium - The Netherlands
mieke.jans@uhasselt.be

Benoît Depaire
Faculty of Business Economics
UHasselt - Hasselt University
Hasselt, Belgium
benoit.depaire@uhasselt.be

Abstract—Although conformance checking is great at detecting process deviations, it still poses challenges that hinders adoption in auditing practice. A major challenge is that in real life a large number of deviating cases is often detected of which only a small amount are true anomalies and thus of real interest to auditors. The number of deviations are often too large to inspect one by one, which explains why auditing requires a sample-based approach. This paper contributes to the research on the practical feasibility of continuous auditing and studies the potential of weak supervision to classify deviations into anomalies and exceptions, allowing auditors to do a full-population analysis of the identified deviations. The Snorkel framework is applied which uses a set of imperfect domain expert rules to classify the set of deviations into anomalies and exceptions. A controlled and artificial experiment has been set up to explore the relation between the performance of this approach and the number and quality of domain expert rules. The results demonstrate the potential of this approach as a limited number of medium to high quality domain expert rules succeeds to classify deviations with acceptable accuracy.

Index Terms—auditing, conformance checking, declare, process deviation, snorkel, weak supervision

I. INTRODUCTION

Process mining is a family of process analysis techniques that discovers a process model from real process executions (process discovery), detects deviations between a normative model and the real process (conformance checking) and enhances the process (process enhancement) [1].

Process mining can be useful for auditing purposes to get a better understanding of the audited company's business environment, including the business processes. Mainly *conformance checking* has great potential in auditing [2] as it provides a tool to automatically discover deviations between the normative process model and the actual process [1]. However, various challenges remain which hinder full adoption in practice.

A major challenge relates to the nature of identified deviations. Deviations can be classified into two types, i.e. *exceptions* which are deviations from the normative model but acceptable because of specific conditions that hold ¹ and

anomalies which represent deviations that are an issue and require follow-up by the auditor [3], [4]. Theoretically, the auditor could check each deviating case to verify it can be classified as an exception or anomaly, but in practice this quickly becomes infeasible because of the large number of deviations identified by conformance checking [5]. Therefore, current conformance checking approaches do not entirely simplify the auditor's work, which prevents the realization of full-population auditing [4].

In this paper, we study the potential of a combination of conformance checking, weak supervision and domain expert rules as a step forward towards auditing the complete set of transactions of a specific business process, replacing up-front sampling methods [4]. The specific problem at hand is where the auditor is confronted with a large set of deviating cases, as identified by the conformance checking approach, which need to be classified as exceptions or anomalies. In the envisaged solution, the auditor only needs to provide a limited set of rules they would use to label deviating cases as exceptions or anomalies in combination with a weak-supervision approach to use this domain knowledge to label all deviations in the initial set. *Weak supervision* is a branch of machine learning that uses noisy, limited, or imprecise sources and while labels derived from such sources are also 'weak' (but inexpensive), weak supervision manages to construct powerful prediction models [6], [7].

We build upon the weak supervision framework Snorkel [8], to classify process deviations into anomalies and exceptions based on a limited set of expert rules. We analyze its potential in an experimental and artificial setting which replicates the context of a procure-to-pay process. For our approach, we opted for the Snorkel Framework as it allowed the injection of domain knowledge in a machine learning model by means of rules and we could extract the labeling model functionality from the larger framework. Auditor's expertise, encoded as *labeling functions*, is used to guide the classification process [8].

Our study shows how a realistic set of deviating cases can be classified into anomalies and exceptions without the need for manually evaluating each case individually or the need for a large set of labeled cases to train a classification model

¹Because normative models are typically simplified and idealized representations of how a process should look like, such specific conditions are often not modeled along, which results deviations that are only exceptions rather than anomalies.

(which also requires manual labeling work). The contribution of this paper is three-fold:

- 1) We demonstrate the potential of weak supervision, and the Snorkel framework in particular, to bring full-population based auditing closer to reality.
- 2) We demonstrate that Snorkel’s labeling model has a higher performance, in terms of accuracy, than a majority voting regardless of the quality of the expert rules.
- 3) We indicate that six high to medium quality rules per label category suffice to identify 60% of the anomalous cases and reach an overall accuracy of 80%.

This paper is structured as follows. In Section II, we explain how the weak supervision system Snorkel can be used in auditing practice. Our research design is explicated in III. Section IV shows the quantitative results of our study. Section V discusses our findings and highlights the implications. In Section VI, we provide some related work. The paper is concluded in Section VII.

II. SNORKEL FOR AUDITING

The machine learning system Snorkel can help to overcome the challenges related to conformance checking in auditing. Traditional supervised machine learning models need a large set of labeled training data to train predictive models. Snorkel does not. Instead, it combines weak supervision sources for training [8]. In this section, we explain how Snorkel can be used to classify process deviations in the context of auditing.

A. Theoretical background

A recent paradigm for weak supervision that does involve domain knowledge in machine learning models is *data programming*. Data programming combines the labels from many weak supervision sources, like heuristic rules, to label datasets programmatically without using any ground truth [9]. Snorkel is a system that implements the idea of data programming. It trains models without manually labeling any training data. The main principle behind Snorkel is that domain knowledge, encoded as *labeling functions*, is used for labeling purposes [8].

In this study, we implement Snorkel in the context of labeling cases as either anomalous (1) or exceptional (0). The labeling procedure needs deviating cases detected by conformance checking, in the form of an event log, as input. The event log contains a set of cases, which represent sequences of one execution of a process [1]. The log consisting of only deviating cases is unlabeled, meaning that the cases are not yet labeled as 0 or 1. The objective of applying Snorkel to an unlabeled set of deviating cases is to identify which cases are anomalous and which are exceptional. In what follows we explain how Snorkel works by walking through two steps: (1) Add a label to each case for each labeling function, and (2) Combine labeling function outputs in a generative model.

1) *Add a label to each case for each labeling function:* In the first step, auditors need to provide Snorkel with labeling functions. Labeling functions are rules that encode domain knowledge. They enable auditors to inject their knowledge into

machine learning models. Therefore, it is important that the auditor has enough knowledge to explicitly determine rules that identify anomalous and exceptional cases. A more formal representation of a labeling function λ_i is given below.

$$\lambda_i \mapsto \{-1, 0, 1\}$$

Each λ_i takes a deviating case and labels it as either exceptional (0), anomalous (1) or it can abstain (-1) from labeling. To illustrate the concept of abstaining, we define the following labeling function: “if a signature is missing, then the case is anomalous.”. The example labeling function identifies whether a case is anomalous (1) or not. If the case is not anomalous, according to this labeling function, it is not necessarily exceptional (0). The abstain label (-1) enables modeling that type of behaviour.

One specific labeling function labels a subset of the data. More labeling functions can overlap (agree) or conflict (disagree) with each other. Notice that each labeling function labels each case in the given event log. The output label that an individual labeling function provides for a specific case, is stored in a $j \times i$ labeling matrix L , with j being the number of cases in the event log and i the number of labeling functions. The labeling matrix L is used as input for the next step.

In Algorithm 1, we provide the pseudo-code that illustrates how we implemented this step in Snorkel.

Algorithm 1

Given:

cases: set of deviating cases in event log

Let:

$l(i)$ be a labeling function with index i , defined by an auditor
 $L(i,j)$ be a label outputted by labeling function $l(i)$ for case j

```

for labeling function  $l(i)$  do
  for case  $j$  do
    if labeling function  $l(i)$  outputs anomaly then
      Store 1 in  $L(i,j)$ 
    else if labeling function  $l(i)$  outputs exception then
      Store 0 in  $L(i,j)$ 
    else Store -1 in  $L(i,j)$ 
    end if
  end for
end for

```

2) *Combine labeling function outputs in a generative model:* In the previous step, each of the j case is labeled by each of the i labeling functions as either an anomaly (1), an exception (0) or unknown (-1), resulting in an $j \times i$ labeling matrix L . As these labeling functions are imperfect and often local rules, labels for a specific case can contradict and two random cases are often labeled by a partially differing set of labeling functions. The goal of a labeling model is to take the i labels for a specific case and map it to a final predicted label.

A first naive approach is to use a ‘majority vote’ labeling model, which predicts the label most often assigned by the labeling functions to the specific case. This approach can become particularly troublesome as labeling functions are

correlated as this increases the weight of some rules in the set. In other words, correlated labeling functions are together more influential when assigning the final label than uncorrelated labeling functions. Furthermore, this also ignores the fact that these labeling functions are of varying quality and some rules should not receive too much weight.

The Snorkel framework follows a different approach by estimating a generative model for the unlabeled data, in order to derive the appropriate weights for each labeling function. Intuitively, Snorkel models the true label for a case as a latent variable in a probabilistic model. It encodes a generative model using parameters which represent the probability that a rule either labels a case as an anomaly or exception rather as unknown and the probability that a rule makes a correct prediction. Next, these parameters are learned on the unlabeled data set by minimizing the negative log marginal likelihood. Ultimately, the generative model uses the accuracies and correlations of the labeling functions to combine the individual output labels into a single confidence-weighted label per case. For technical details on *data programming* and *Snorkel*, we refer to works of [8], [9].

In Algorithm 2, we provide the pseudo-code of this step.

Algorithm 2

Given:

j : case index

i : labeling function index

L : $i \times j$ label matrix, consisting of labeling function outputs

Let:

$y(j)$ be the final output label for case j

```

for labeling function  $l(i)$  do
  for case  $j$  do
    Learn labeling function accuracies
    Learn labeling function correlations
    Reweight labeling functions
    Calculate final label
    Add final label to  $y(j)$ 
  end for
end for

```

B. Assumptions

With Snorkel, we aim to label the unlabeled event log by using labeling functions. Since labeling functions represent the auditor’s domain knowledge, the auditor’s role is essential in our study. Consequently, an important assumption in testing whether weak supervision systems can be used to classify deviations, is that an auditor is capable of defining labeling functions that identify anomalous and exceptional cases. Notice that this shifts the auditor’s role from checking individual cases on anomalies or exceptions to providing high level rules that identify anomalous or exceptional cases.

C. Implementation

Snorkel requires Python 3.6 or a later version. We implemented Snorkel and the algorithms from Sections II-A

in Python 3.6 for Linux. The codes are stored in a GitHub repository²

III. EXPERIMENTAL DESIGN

This section explicates the research design of our study. First, we explain our research goal and design. Next, we describe how data were artificially generated. Final, we provide some background on how we constructed labeling functions.

A. Research goal and design

We define a set of deviating cases that needs to be classified as anomalous or exceptional. More specifically, we generate an unlabeled set of deviating cases and a labeled validation set that is used to validate our results. Subsequently, we use heuristics, representing auditing knowledge, as sole input to label the unlabeled set of deviating cases.

In order to test how good the weak supervision system Snorkel performs at classifying deviating cases as anomalous or exceptional, we run a set of computational tests under controlled conditions. We want to measure whether following parameters have an impact on the model’s performance:

- The quality of the labeling functions, expressed in terms of accuracy
- The number of labeling functions

Above parameters are both associated with labeling functions. To determine their quality, we first construct three sets of different quality levels of labeling functions: high ($> 80\%$), medium ($60\% - 80\%$) and low ($< 60\%$) accuracy labeling functions. Every set consists of at least 10 labeling functions. A labeling function is a proxy for the auditor’s domain knowledge. Therefore, respectively, a high, medium and low accuracy labeling function represents that an auditor is very good, medium or not good at all at defining labeling functions that accurately express their knowledge.

We chose for an artificial setting because this research is still in an exploratory phase. It is expensive to collect data when it is unclear which relations we are looking for. Therefore, it is more logical to start with an artificial, but realistic, setting in which we can control the data. Building upon this, it is important to know what the data comprises, because there is a high risk that some patterns in the data point to correlations (rather than causal relations). This can lead to a wrong interpretation of the results. Furthermore, the artificial setting enables us to simulate auditors of different expert levels by defining different quality levels of rules. Nevertheless we call the artificial setting ‘realistic’, because we start from a specific and realistic setting: a procure-to-pay process.

B. Data

The set of deviating cases, including exceptions and anomalies, that is used in this study is artificially generated. This allows, amongst other things, for a set-up with a validation set. To this purpose, we start from three declarative process models. We created a *normative*, an *auditor* and a *real* process

²<https://github.com/manallaghmouch/snorkelforauditing>

model for a procure-to-pay process; the first being the most restrictive, and each of the latter two being a less restrictive version of the previous one. The models consist of eight activities: *create purchase request*, *approve purchase request*, *create purchase order*, *sign*, *receive goods or services*, *receive invoice* and *pay*.

The normative process model represents the process that a company desires to follow and holds the most restrictions. It contains 14 declarative constraints in our design. Deleting some of these constraints results in the auditor process model (7 constraints). This model contains the process paths that the auditor would accept, even if they deviate from the normative process model. Deleting again some declarative constraints from the auditor model results in the real process model. This is the least restrictive model of the three, and consists of the lowest number of declarative constraints (3 constraints). Behaviour that fits in this model, but not in the auditor model, are examples of deviating cases that are anomalies. The declarative constraints in the real model simulate the user constraints that are imposed by the information system's configuration. As such, the behaviour that is modeled in the real process model represents the as-is process.

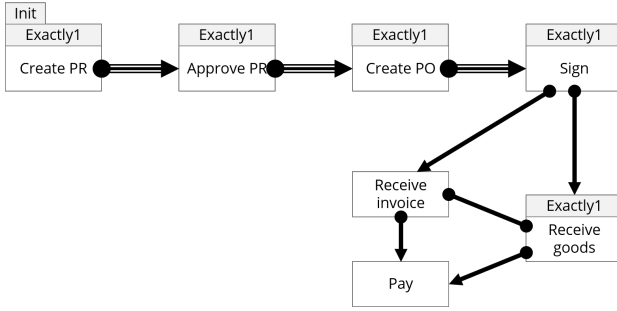


Fig. 1: The designed normative process model.

To get a better understanding of how the designed Declare models look like, we show the normative process model in Figure 1. An example of a declarative constraint in this model is a chain response between the activities *Create PR* and *Approve Pr*. This means that if *Create PR* occurs, then *Approve PR* directly follows. The auditor model is not visualised, but consists of seven constraints that are a subset of the normative process model. Following the same rationale, the real process model is a subset of the auditor model and consists of three constraints. For more information on the DECLARE template, we refer to the work of [10].

Recall that the declarative process models were designed with the sole purpose to artificially create an unlabeled event log and a validation set. We used the *real* process model to artificially generate two event logs using the *MinerFUL Log Generator* from [11]: one log of 2000 cases and another log of 1000 cases. Both logs contain cases of 3 to 10 events per case. The first event log represents the unlabeled event log, meaning that this set consists of cases that are not yet labeled as anomaly or exception. The second event log is used to obtain the validation set. Because the validation set is used

to validate the outcomes of the labeling algorithm, it should consist of only labeled cases. We obtain the labels of the validation set cases by performing a conformance check in the following two steps:

- 1) The event log of 1000 cases is compared with the *normative* process model. The result is a binary classification into *deviation* and *no deviation*. The cases that contain at least one deviation are extracted as 'deviating cases'.
- 2) The deviating cases from step 1 are compared with the *auditor* process model. The result of this conformance check is a binary classification into *anomaly* and *exception*.

The cases that are labeled as 'anomaly' or 'exception' constitute the validation set. After the two-step conformance check, we obtain the following two datasets. (1) An unlabeled event log consisting of 2000 deviating cases, and (2) A balanced, validation set consisting of 1000 deviating cases (452 anomalies, 548 exceptions). Note that none of the 1000 artificially generated traces of the *real* process model fell within the constraints of the *normative* process model. As a consequence, all 1000 cases are deviating cases and are part of the validation set.

C. From rule sets to labeling functions

We define a *rule set* as a grouping of one or more rules that identify a concept. Because we are dealing with a binary classification problem, we need two key rule sets, respectively a set of rules that identify an anomaly (Rule Set Anomaly - RSA) and a set of rules that identify an exception (Rule Set Exception - RSE). In order to obtain the two key rule sets, we construct four additional rule sets: Rule Set Normative (RSN), Rule Set Auditor (RSAU), Rule Set Real (RSR) and Rule Set Deviations (RSD). RSN is the collection of declarative rules that constitute the normative process model. RSAU is the collection of declarative rules of the auditor process model. RSR is the collection of declarative rules of the real process model. RSD is the difference between the declarative rules in the normative process model and the declarative rules in the real process model. Mathematically, the rule sets are formulated as follows.

$$\begin{aligned}
 \text{RSN} &= \{x : x \text{ is declarative rule from normative model}\} \\
 \text{RSAU} &= \{x : x \text{ is declarative rule from auditor model}\} \\
 \text{RSR} &= \{x : x \text{ is declarative rule from real model}\} \\
 \text{RSD} &= \text{RSN} \setminus \text{RSR}
 \end{aligned}$$

RSA and RSE are then as follows.

$$\begin{aligned}
 \text{RSA} &= \text{RSD} \setminus \text{RSAU} \\
 \text{RSE} &= \text{RSAU} \setminus \text{RSN}
 \end{aligned}$$

IV. RESULTS

In this section, we show the results of our experiments. First, we describe the effect of the quality of the labeling functions on the accuracy of Snorkel's labeling model, the LabelModel. Subsequently, we show the effect of the amount of labeling functions on the accuracy of the LabelModel.

A. Effect of the quality of labeling functions on performance

As described in the previous section, we constructed three sets of different quality levels of labeling functions: high, medium and low. Furthermore, we distinguished between anomaly and exception rules.

To test the effect of labeling function quality on the performance of the LabelModel, we limit the amount of labeling functions to a balanced set of five anomaly and five exception functions that were randomly chosen. How the labeling function sets are composed in terms of quality level, differs over the experimental runs.

We executed 66 experimental setups in total. We can divide this in six parts of 11 setups. The 11 setups were structured as follows. We fixed the amount of the anomaly (exception) rules to five at a constant level of quality, while gradually decreasing the quality of the exception (anomaly) rules. For example, we start with 5 high quality anomaly rules and 5 high quality exception rules. In the second setup, we still have 5 high quality anomaly rules, but 4 high quality exception rules and 1 medium quality exception rule. In the third setup, we again still have 5 anomaly rules, but 3 high quality exception rules and 2 medium quality exception rules. We keep on replacing higher level exception rules by lower level exception rules until we are left with 5 low quality exception rules.

Each setup was repeated 15 times to obtain an average accuracy. The mean accuracies of Snorkel’s LabelModel and the naive approach of majority voting were calculated. Figure 2 visualizes our findings. Notice that the ‘lift’ in the Figures should be interpreted as the improvement the LabelModel offers relative to the naive approach of majority voting. To get an understanding of the behaviour of labeling function quality per category on the accuracy of the models, we kept one category fixed, while decreasing the quality of the labeling rules of the other category. For example, in (a), we show the effect of the quality of exception rules on the accuracy, keeping anomaly rules at a constant level of high quality.

Figure 2 shows that Snorkel’s LabelModel performs significantly better in classifying deviating cases than the naive approach of majority voting ($t = 0.00, p < 0.01$). High- and medium-quality anomaly rules are robust to changes in quality of the exception rules (Figure 2 (a) and (b)). Furthermore, for the fixed medium- and low-quality anomaly rules, the LabelModel decreases abruptly at the point where the quality level of exception rules shifts from high quality to medium/low quality. The same reasoning holds for the fixed high- and medium-quality exception rules, where the quality level of the anomaly rules shifts from high quality to medium/low quality. This effect is mainly noticeable for the LabelModel. It suggests that the LabelModel loses most performance when the set of labeling functions consists of about 25% low-quality rules. However, adding even more low-quality rules to the model at the expense of high-quality rules does not have an additional decreasing effect on accuracy. All figures have about the same gradient, except for (c). This figure shows that majority voting may perform better than the LabelModel in some specific

cases. A possible explanation for this behaviour in our setting is that the set of low quality exception rules contains rules with a zero accuracy, while the set of anomaly rules does not.

B. Effect of the number of labeling functions on performance

In order to test the effect of the number of labeling functions on the performance of the LabelModel and the majority voting approach, we increase the number of randomly chosen labeling functions gradually. We do this for an equal mix of high and medium quality labeling functions from the anomaly and exception categories. Since Snorkel requires a minimum of 3 labeling functions, our minimum is set to 4 (to keep a balanced set of anomaly and exception rules). For every setup, we increased the number of labeling functions per category by one.

First, we test the sensitivity of our model by showing how many anomalies the model classifies correctly with a balanced set of only high- and medium-quality anomaly rules as input. We focus on anomaly rules, because discovering anomalous cases is highly important from an auditing perspective. Second, we perform a similar analysis on a balanced combination of both anomaly and exception rules. In Figure 3, we graphically present our results.

As both (a) and (b) show, the LabelModel is better at identifying anomalous cases than majority voting, regardless of the number of labeling functions and regardless the use of only anomaly rules or a combination of anomaly and exception rules (lift). When only using anomaly rules, the LabelModel correctly classifies 65% to 80% of the anomalies (a), when using both anomaly and exception rules, the LabelModel correctly classifies 50% to 75% of the anomalies (b). Furthermore, for both setups, is significantly better at correctly classifying anomalies than majority voting ($t = 0.00, p < 0.01$). The difference is more noticeable when both anomaly and exception rules are taken into account. However, the LabelModel labels more anomalous cases correctly when the model is provided with anomaly rules only. This might indicate that including additional exception rules, on top of the anomaly rules, in the model has a negative impact on correct classification.

Although from an auditing point of view, we focus on correctly classifying anomalous cases, we are still interested in predicting both anomalies and exceptions. Classifying both is important because at the end of the classification, a limited amount of cases might still be unlabeled. In the case of auditing, we would see these cases as ‘potentially anomalous’, to depict that the model was unsure about the label and therefore abstained from making a final decision on the label. To avoid the negative effect of having a balanced set of anomaly and exception rules on correctly classifying anomalies, we could provide the model with more anomaly rules than exception rules or assign more weight to the anomaly rules.

We also have to take into account that adding more rules to the model diminishes the added value when the model already consists of about 12 rules. With 12 rules consisting of a balanced set of high- and medium-quality rules, the LabelModel identified about 60% of the anomalous cases

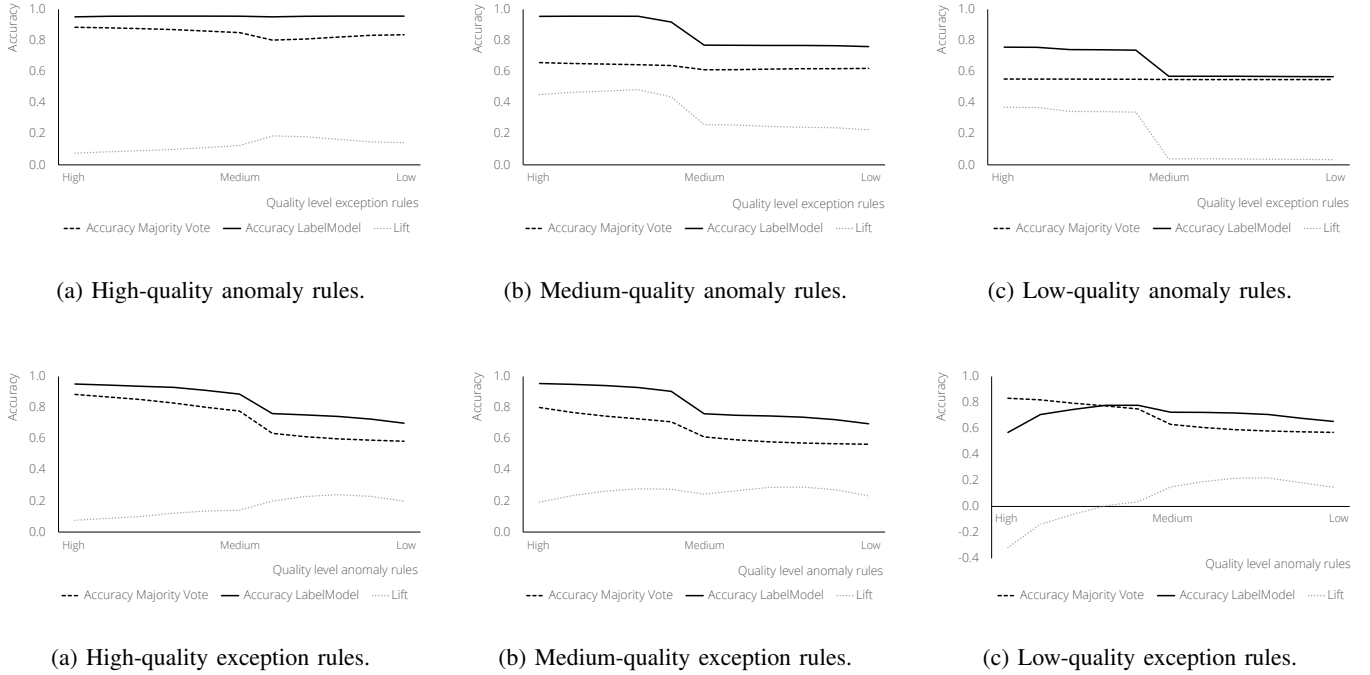


Fig. 2: The effect of decreasing the quality of rules on accuracy.

correctly (twice as much than majority voting), the other 40% were not identified (i.e. the model abstained from labeling). The overall model accuracy at this point was 80%, meaning that many cases were correctly classified as anomalous or exceptional.

V. DISCUSSION

This section points out the relevance and implications of our results in a broader perspective. We discuss the implications for research and practice and provide an overview of some research limitations.

A. Implications

Our work has implications for the research field of continuous auditing. Research in this area focuses on automation of tasks that highly require domain expertise. A general question here is to what degree full automation is practical in auditing practice. Recent studies suggest that automation could have a powerful effect on today's accounting practices [12]. However, it is less likely that auditing tasks will be fully automated. Rather, auditors will work together with machines to complete a task [13], [14]. The findings of our research implicate the same. Computational tools and techniques will only be valuable in auditing as long as using them significantly simplifies current auditing tasks, which is currently not the case. Conformance checking results in a large set of process deviations and forces an auditor to take samples [4], [5], [15].

Our study shows that weak supervision can offer a solution for the problem of alarm floods in auditing, enabling full population testing. We demonstrate that the weak supervision

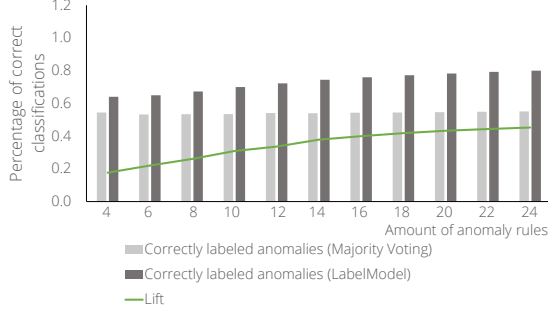
system Snorkel smoothly integrates domain knowledge, in the form of rules, in a machine learning model. Our results suggest that the provided rules have to be of medium to high quality to obtain reasonable results. Furthermore, with only six high- to medium-quality labeling rules for both anomalies and exceptions, the Snorkel model reaches an acceptable accuracy.

B. Limitations

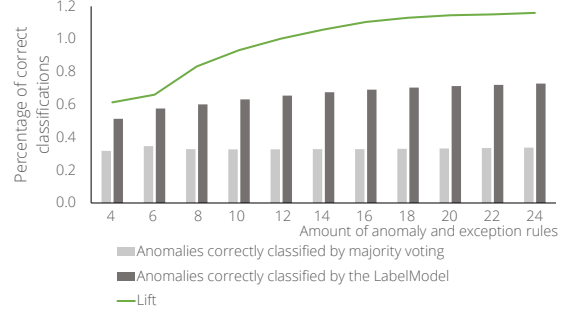
Our study has some limitations that are inherent to its artificial setup. First, the generated event logs are based on a procure-to-pay process. While we designed a realistic process model and obtained powerful results, future research has to investigate the potential effect of using different process models. A reasonable starting point for such research can be changing the complexity of the declarative model and explore its effect on model performance.

Second, although the declarative rules are a good proxy of an auditor's knowledge, it does not fully account for an actual auditor. Now we know weak supervision potentially holds a solution for continuous auditing, a follow-up fine-tuning of our research direction might be as follows. As a first step, we can design a semi-artificial experiment in which auditors have to verify rules from the auditor model. Subsequently, we let auditors design rules from scratch and use these as labeling functions.

Third, to label the generated set of deviating cases, we used Snorkel's generative model. While this model effectively labeled our dataset, a discriminative model could generalize beyond the provided heuristics without losing precision [8]. The obtained (labeled) event log can serve as input to train a



(a) Correctly classified anomalies, based on anomaly rules.



(b) Correctly classified anomalies, based on anomaly and exception rules.

Fig. 3: The effect of increasing the number of rules on correct classification of anomalies.

discriminative model on unseen data. The possibilities in the context of auditing have to be further investigated in future research.

VI. RELATED WORK

This section describes background literature that is relevant to our research. We start with discussing prior research on conformance checking in auditing. Thereafter, we provide some background on weak supervision and Snorkel.

A. Continuous auditing of business processes

The increased availability of digital data and information systems is changing the current way of auditing [16]–[18]. Continuous Auditing is the idea of automating audit procedures. Such techniques can assist auditors in objectively auditing a company [4], [16], [19], [20]. Continuous auditing enforces traditional auditing to change. Auditors who are familiar with the flow of transactions and related control activities become essential, as they can now analyze business processes based on digital transactions [21].

Many continuous auditing tools and techniques can be used to analyze business processes [22], [23]. In particular, process mining is a family of process analysis techniques that provides an objective view on the process. It uses an event log as input to initialize automatic analysis [1]. Conventionally, process mining is divided in three types: process discovery, conformance checking and process enhancement. With process discovery, a process model is discovered from real process executions. With conformance checking, a normative process model is compared with the real process to detect process deviations. With process enhancement, additional analysis like time and resource analysis are performed with the objective to enhance the actual process. Using solely an event log as input, process mining techniques can provide insightful information about a business process, like sequence, duration and interaction information [1], [24].

Mainly the type *conformance checking* is useful for more in-depth auditing analyses [2], [4]. Although conformance

checking smoothly automates the identification of a large set of process deviations, using this in auditing practice bears some challenges. Since the normative process model is a simplified representation of how the process should look like, an alarm flood of process deviations is detected. This set consists of a relatively small set of deviations that are important from an auditing point of view (anomalies) and a relatively large set of technical deviations that do not matter from an auditing perspective (exceptions) [4]. Current conformance checking output does not simplify the auditor’s work, with the consequence that sampling is still necessary. Some studies propose frameworks that could provide a solution for alarm floods in continuous auditing [4], [15], [25]. However, a practical integration of existing techniques is not yet tested. In this paper, we demonstrated and experimentally tested how a weak supervision system can be leveraged to classify deviating cases as anomalous or exceptional.

B. Weak Supervision

Machine learning based systems are getting increasingly more attention nowadays. With deep learning techniques, can effectively be used to predict the labels of unseen data instances. However, those techniques require large training sets of labeled examples to reach high predictive performance. Obtaining these large sets of labeled data is time consuming and costly [26]. Because of this, weak supervision is gaining more attention. It is a branch of machine learning in which cheaper sources of heuristic or noisy labels are used [27]. Some popular forms of weak supervision are distant supervision [28], [29], crowd-sourced labeling [30], [31], and heuristics for labeling data [32]. Although these weak supervision forms are inexpensive, they have a rather limited accuracy [27].

A recent paradigm for weak supervision that has been proposed in literature is *data programming*. Data programming combines the labels from many weak supervision sources, without using ground truth labels, to increase the accuracy and coverage of training data. Furthermore, probabilistic training labels that represent the lineage of individual labels, are

generated. Consequently, with only a generative mode, source accuracies and correlation structures are recovered without using labeled training data Snorkel is a system that implements data programming to train machine learning models without manually labeling any training data. For this purpose, it uses a set of labeling functions (i.e. rules) as input [8]. A labeling function encodes domain knowledge and other supervision sources programmatically. After defining a set of labeling functions, the labeling functions are combined in a machine learning model to build labeled training sets that can be used to train a discriminative model [8], [27]. Since domain knowledge is very crucial in auditing and Snorkel provides a way of injecting domain knowledge into machine learning models, we used the Snorkel system in this paper to demonstrate its potential to classify process deviations.

VII. CONCLUSION

In this paper, we show how weak supervision effectively labels deviating cases as *anomalous* or *exceptional*. We did this by implementing the weak supervision system Snorkel. We ran a set of computational experiments on a realistic, but artificially generated, event log of a procure-to-pay process, along with a normative and auditor model, representing the desired business process on the one hand and the acceptable process on the other hand. The results of this study suggest that Snorkel's labeling model performs better than the naive approach of majority voting. With each six high- to medium-quality labeling functions for both anomalies and exceptions, the Snorkel model reaches a relatively high accuracy of 80% and identifies at least 60% of the anomalous cases.

REFERENCES

- [1] W. Van der Aalst, *Process mining: Data Science in Action*, 2016.
- [2] M. Jans, M. Alles, and M. Vasarhelyi, "The case for process mining in auditing: Sources of value added and areas of application," *International Journal of Accounting Information Systems*, vol. 14, no. 1, pp. 1–20, 2013.
- [3] B. Depaire, J. Swinnen, M. Jans, and K. Vanhoof, "A process deviation analysis framework," in *International Conference on Business Process Management*. Springer, 2012, pp. 701–706.
- [4] M. Jans and M. Hosseinpour, "How active learning and process mining can act as Continuous Auditing catalyst," *International Journal of Accounting Information Systems*, vol. 32, pp. 44–58, Mar. 2019.
- [5] M. G. Alles, A. Kogan, and M. A. Vasarhelyi, "Putting continuous auditing theory into practice: Lessons from two pilot implementations," *Journal of Information Systems*, vol. 22, no. 2, pp. 195–214, 2008.
- [6] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2018.
- [7] A. Ratner, S. Bach, P. Varma, and C. Ré, "Weak supervision: the new programming paradigm for machine learning," *Hazy Research*. Available via <https://dawn.cs.stanford.edu/2017/07/16/weak-supervision/>. Accessed, pp. 05–09, 2019.
- [8] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, no. 2, pp. 709–730, 2020.
- [9] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," in *Advances in neural information processing systems*, 2016, pp. 3567–3575.
- [10] M. Pesic, H. Schonenberg, and W. M. Van der Aalst, "Declare: Full support for loosely-structured processes," in *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*. IEEE, 2007, pp. 287–287.
- [11] C. Di Ciccio, M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Generating event logs through the simulation of declare models," in *Workshop on Enterprise and Organizational Modeling and Simulation*. Springer, 2015, pp. 20–36.
- [12] C. B. Frey and M. A. Osborne, "The future of employment: How susceptible are jobs to computerisation?" *Technological forecasting and social change*, vol. 114, pp. 254–280, 2017.
- [13] J. Mendling, G. Decker, R. Hull, H. A. Reijers, and I. Weber, "How do machine learning, robotic process automation, and blockchains affect the human factor in business process management?" *Communications of the Association for Information Systems*, vol. 43, no. 1, p. 19, 2018.
- [14] N. Berente, S. Seidel, and H. Safadi, "Research commentary—data-driven computationally intensive theory development," *Information Systems Research*, vol. 30, no. 1, pp. 50–64, 2019.
- [15] P. Li, D. Y. Chan, and A. Kogan, "Exception prioritization in the continuous auditing environment: A framework and experimental evaluation," *Journal of Information Systems*, vol. 30, no. 2, pp. 135–157, 2016.
- [16] D. Y. Chan and M. A. Vasarhelyi, "Innovation and practice of continuous auditing," *International Journal of Accounting Information Systems*, vol. 12, no. 2, pp. 152–160, 2011.
- [17] H. Chen, R. H. Chiang, and V. C. Storey, "Business intelligence and analytics: From big data to big impact," *MIS quarterly*, pp. 1165–1188, 2012.
- [18] M. Jans, M. Alles, and M. Vasarhelyi, "The case for process mining in auditing: Sources of value added and areas of application," *International Journal of Accounting Information Systems*, vol. 14, no. 1, pp. 1–20, 2013.
- [19] S. Groomer and U. Murthy, "Continuous auditing of database accounting systems using embedded audit modules," *Journal of Information Systems*, vol. 3, no. 1, pp. 53–69, 1989.
- [20] M. A. Vasarhelyi, M. G. Alles, and A. Kogan, "Principles of analytic monitoring for continuous assurance," *Journal of emerging technologies in accounting*, vol. 1, no. 1, pp. 1–21, 2004.
- [21] M. Werner, "Financial process mining-accounting data structure dependent control flow inference," *International Journal of Accounting Information Systems*, vol. 25, pp. 57–80, 2017.
- [22] A. Datta, "Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches," *Information Systems Research*, vol. 9, no. 3, pp. 275–301, 1998.
- [23] S. X. Sun, J. L. Zhao, J. F. Nunamaker, and O. R. L. Sheng, "Formulating the data-flow perspective for business process management," *Information Systems Research*, vol. 17, no. 4, pp. 374–391, 2006.
- [24] R. J. C. Bose and W. van der Aalst, "Trace alignment in process mining: opportunities for process diagnostics," in *International Conference on Business Process Management*. Springer, 2010, pp. 227–242.
- [25] J. L. Perols and U. S. Murthy, "Information fusion in continuous assurance," *Journal of Information Systems*, vol. 26, no. 2, pp. 35–52, 2012.
- [26] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.
- [27] A. J. Ratner, S. H. Bach, H. R. Ehrenberg, and C. Ré, "Snorkel: Fast training set generation for information extraction," in *Proceedings of the 2017 ACM international conference on management of data*, 2017, pp. 1683–1686.
- [28] A. Madaan, A. Mittal, G. Ramakrishnan, S. Sarawagi *et al.*, "Numerical relation extraction with minimal supervision," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [29] A. Smirnova and P. Cudré-Mauroux, "Relation extraction using distant supervision: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–35, 2018.
- [30] G. Xu, W. Ding, J. Tang, S. Yang, G. Y. Huang, and Z. Liu, "Learning effective embeddings from crowdsourced labels: An educational case study," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1922–1927.
- [31] R. Lotfian and C. Busso, "Curriculum learning for speech emotion recognition from crowdsourced labels," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 4, pp. 815–826, 2019.
- [32] C. De Sa, A. Ratner, C. Ré, J. Shin, F. Wang, S. Wu, and C. Zhang, "Deepdive: Declarative knowledge base construction," *ACM SIGMOD Record*, vol. 45, no. 1, pp. 60–67, 2016.