Automated anomaly detection for categorical data by repurposing a form filling recommender system

HICHEM BELGACEM[*], Luxembourg Institute of Science and Technology, Luxembourg
XIAOCHEN LI[*], Dalian University of Technology, China
DOMENICO BIANCULLI, University of Luxembourg, Luxembourg
LIONEL BRIAND[*], Lero SFI Centre for Software Research, University of Limerick, Ireland and University of Ottawa, Canada

Data quality is crucial in modern software systems, like data-driven decision support systems. However, data quality is affected by data anomalies, which represent instances that deviate from most of the data. These anomalies affect the reliability and trustworthiness of software systems, and may propagate and cause more issues. Although many anomaly detection approaches have been proposed, they mainly focus on numerical data. Moreover, the few approaches targeting anomaly detection for categorical data do not yield consistent results across datasets.

In this paper, we propose a novel anomaly detection approach for categorical data named LAFF-AD (LAFF-based Anomaly Detection), which takes advantage of the learning ability of a state-of-the-art form filling tool (LAFF) to perform value inference on suspicious data. LAFF-AD runs a variant of LAFF that predicts the possible values of a suspicious categorical field in the suspicious instance. LAFF-AD then compares the output of LAFF to the recorded values in the suspicious instance, and uses a heuristic-based strategy to detect categorical data anomalies.

We evaluated LAFF-AD by assessing its effectiveness and efficiency on six datasets. Our experimental results show that LAFF-AD can accurately determine a high range of data anomalies, with recall values between 0.6 and 1 and a precision value of at least 0.808. Furthermore, LAFF-AD is efficient, taking at most 7000 s and 735 ms to perform training and prediction, respectively.

CCS Concepts: • **Computing methodologies** → **Anomaly detection**; *Bayesian network models*; • **Information systems** → *Data mining.*

Additional Key Words and Phrases: Data quality, Data anomaly detection, Categorical data, Machine Learning

ACM Reference Format:
Hichem Belgacem, Xiaochen Li, Domenico Bianculli, and Lionel Briand. 2024. Automated anomaly detection for categorical data by repurposing a form filling recommender system. *ACM J. Data Inform. Quality* 1, 1, Article 1 (January 2024), 28 pages. https://doi.org/10.1145/3696110

[*]Part of this work was done while the author was affiliated with the University of Luxembourg, Luxembourg.

Authors' addresses: Hichem Belgacem, Luxembourg Institute of Science and Technology, Luxembourg, hichem.belgacem@list.lu; Xiaochen Li, Dalian University of Technology, China, xiaochen.li@dlut.edu.cn; Domenico Bianculli, University of Luxembourg, Luxembourg, domenico.bianculli@uni.lu; Lionel Briand, Lero SFI Centre for Software Research, University of Limerick, Ireland and University of Ottawa, Canada, Lionel.Briand@lero.ie.

1936-1955/2024/1-ART1
https://doi.org/10.1145/3696110

1 INTRODUCTION

Data plays a central role in modern software systems, which are very often powered by machine learning (ML) and used in critical domains of our daily lives, such as finance, health, and transportation [49]. However, the effectiveness of ML-intensive software applications highly depends on the quality of the data [44]. When data quality is poor, the output of these systems cannot be trusted. This makes data quality of utmost importance since it impacts how trustworthy and reliable these applications are (e.g., decision support systems).

Data quality is affected by data anomaly. An anomaly is defined as data instances that deviate from other data instances [17]. Data anomalies can arise in many practical situations, such as wrong values entered by users during the data entry process through form filling (e.g., typos) and errors made during data management (e.g., faulty sources of data) or data integration [32].

Data anomalies need to be detected; once detected, they have to be fixed or excluded from the data used in ML-intensive applications. If the anomalies are not detected in a timely fashion, they may propagate and cause more data quality issues, leading to serious problems that affect decision-making [44]. For example, one study showed that the financial loss because of fraud, a kind of data anomaly in the financial domain, has increased by almost 56.6% since 2009. The value of this loss was estimated to be 5.1 trillion USD in 2018 [15]. Another study revealed that the average cost of network downtime resulting from data anomalies is 100 000 USD per hour; this value can grow as more people are using or become dependent on web applications [27]. This explains the fact that anomaly detection has been applied in several critical domains such as fraud detection [39], health care [12], and network intrusion identification in computer science [23].

To detect data anomalies, many approaches have been proposed. However, their focus has been mainly on numerical data. The main idea is to define a proximity measure between data instances [22], and use this measure to detect data instances that deviate from the majority of the data. In contrast, fewer studies attempt to detect anomalies for categorical data [48]. Since in practice data is often described by categorical attributes [22], we focus in this article on detecting anomalies in categorical data.

There are two reasons to motivate this work. First, categorical data anomaly is more challenging to detect [22]. The fundamental issue is to define a proximity measure over categorical values [41]; however, it is not easy to devise a criterion to separate between anomalous and non-anomalous categorical data [2]. Second, empirical studies show that more than 50% of anomalies in critical applications (e.g., medical records) are in categorical fields [40].

Although anomaly detection for categorical data has been investigated in the literature (e.g., with frequency-based approaches [35, 37], clustering-based approaches [46, 47], semi-supervised approaches [34]), our preliminary experiments show that the effectiveness of these approaches is not stable across datasets. More specifically, especially since these approaches include different parameters to be tuned, one technique can perform greatly in terms of accuracy on one dataset but poorly on another.

To detect data anomalies effectively, one intuitive solution is to infer the correct value in a categorical dataset; then, an anomaly can be detected by comparing the inferred correct value with the recorded one. Using such inferred values, data anomalies can also be easily fixed. Since one of the main sources of data anomaly are wrong values entered by users during form filling [45], in the literature many form filling approaches [7, 30] have been proposed to accurately predict the correct value to be filled in a data entry form. For example, the state of the art (SOTA) form filling approach LAFF (Learning-based Automated Form Filling) [7], relies on multiple Bayesian networks and a heuristic-based endorser to suggest a ranked list of candidate values to fill in a categorical field in the data entry form. The suggestions made by LAFF allow users, in between 79.0% and 90.1% of the cases, to find the correct value among the 5% top-ranked suggested values.

Driven by the high accuracy of existing form filling tools such as LAFF, in this article we propose a LAFF-based Anomaly Detection approach ("LAFF-AD" in short) to effectively detect categorical data anomalies. The basic idea of LAFF-AD is to take advantage of the learning ability of LAFF to perform value inference on suspicious data. The output of such inference is used by LAFF-AD to determine the presence of anomalies in data.

LAFF-AD includes three main phases: LAFF offline prediction, anomaly detection, and threshold determination. Similar to LAFF, LAFF-AD starts by learning Bayesian network models on a clean set of data. Given a set of suspicious data, LAFF-AD runs a variant of LAFF that handles offline prediction (i.e., in contrast to real-time during the data entry process) to predict the value of a categorical field in a suspicious instance. This variant returns a ranked list of possibly correct values for the suspicious instance, the probability of each value in the ranked list, and a flag indicating whether the prediction comes with high confidence. In the next phase, LAFF-AD leverages the output of LAFF to detect data anomaly with a heuristic-based strategy. LAFF-AD analyzes the outputs of LAFF by checking three heuristics. The first one checks if LAFF has enough confidence to make a prediction (i.e., the prediction is endorsed). The second one checks if the suspicious value is ranked to the top in the list predicted by LAFF. The third heuristic checks if the anomaly score of the suspicious value is higher than a certain anomaly score threshold. The anomaly score is computed based on the probability of the suspicious value in the ranked list predicted by LAFF. According to the three heuristic rules, LAFF-AD determines the presence of a data anomaly. In the threshold determination phase, LAFF-AD automatically determines the value of the "anomaly score threshold" used in the anomaly detection phase for each dataset; this phase minimizes the influence of selecting appropriate parameters for anomaly detection approaches.

We evaluated LAFF-AD using six anomaly detection datasets, with different characteristics. The experimental results show that LAFF-AD can accurately detect a high range of data anomalies, with a precision value of at least 0.808.

To summarize, the main contributions of this paper are:

- The LAFF-AD approach, which addresses the problem of anomaly detection for categorical data. To the best of our knowledge, LAFF-AD is the first work to repurpose a form filling recommender system to detect data anomalies. LAFF-AD provides effective data anomaly detection results without the need for manual tuning.
- An extensive evaluation assessing the effectiveness and efficiency of LAFF-AD, including a comparison with SOTA approaches. Our evaluation shows that LAFF-AD yields stable results outperforming SOTA algorithms for most datasets.

The rest of the paper is organized as follows. Section 2 provides a motivating example and presents the concept of data anomaly detection and its challenges. Section 3 reviews the SOTA and its limitations. Section 4 introduces the LAFF approach that underlies our approach for anomaly detection. Section 5 describes the core phases of LAFF-AD. Section 6 reports on the evaluation of LAFF-AD. Section 7 discusses the usefulness of LAFF-AD, taking into account the practical implications of the experimental results. Section 8 concludes the paper.

2 DATA ANOMALY DETECTION

2.1 Motivating example

Real-world data contains data anomalies that affect data quality. These anomalous data can have severe consequences, especially in critical domains such as finance and health care. Let us assume a dataset for an energy provider contains the following columns "Tariff plan", "Customer segment", "Fixed fees", and "Consumption average" (as shown on the right of Figure 1). "Tariff plan" and "Customer segment" are two categorical columns with the following values ("`Standard`", "`Time of use`", "`Renewable energy`") and ("`Residential`", "`Commercial`", "`Non-profit organization`", "`Industrial`"), respectively. The remaining columns "Consumption average" and "Fixed fees" are numerical columns. Based on this information, the energy provider relies on an ML-based bill calculator to decide the "Rate per kWh" according to the needs and the tariff plan of different customers. After a certain period of time, some customers with a standard tariff plan started complaining about extra charges even though their consumption of energy had not increase.

The energy provider team decided to investigate the reason behind this problem. The investigations showed that the main reason for the extra charges is that the bill calculator relied on bad quality data containing some anomalies. The decisions were based on anomalous instances having "Tariff plan" equal to "`Standard`", "Customer segment" equal to "`Industrial`", "Average consumption" equal to "`400`", and "Fixed Fees" equal to "`20`". These instances deviate from the normal data instances since "Tariff plan" equal to "`Standard`" is usually associated with "Customer segment" equal to "`Residential`" and "Average consumption" equal to "`100`". These anomalous instances misled the bill calculator, which used a high "Rate per kWh" equal to "`0.5`" instead of "`0.1`" for residential customers. Indeed, residential customers pay fewer taxes for energy since they use it for personal reasons; on the opposite, industrial customers need to pay more. These anomalies in the data can be introduced in different ways such as during data entry (e.g., typos), data management (e.g., faulty data source), and data integration (when assembling data from different sources) [32].

To solve this problem, the energy provider decides to do an audit on the dataset to check the data quality of different data instances, and to apply anomaly detection in order to detect anomalous instances that affect the data and decision quality.

2.2 Problem Definition

In this paper, we deal with the problem of anomaly detection for categorical data in relational databases. This problem can be informally defined as the problem of deciding whether the value of a target column in a given suspicious instance is anomalous or not.

In this work, we target categorical columns since in practice data is often described with categorical attributes [22]. This type of column is subject to anomalous values since, for example, the filling process of categorical attributes is error-prone and time-consuming [7]. Another reason is that identifying anomalies in such kinds of columns is difficult since it is not easy to devise criteria to separate between anomalous and non-anomalous data [2].

We define the anomaly detection problem as follows. Let D be a dataset composed of a set of n columns $C = \{c_1, c_2, \ldots, c_n\}$. Let $C^c \subseteq C$ be the set of categorical columns. Each column c_i can take a value from a certain domain V_i. D can be partitioned into datasets CD and SD, representing respectively clean data and suspicious data possibly containing data anomalies; we have $CD \cup SD = D$ and $CD \cap SD = \emptyset$. During the anomaly detection process, the columns are partitioned in two groups, i.e., a set of features C^f and one target column $C^t \in C^c$; we have that $C^f \cup C^t = C$ and $C^f \cap C^t = \emptyset$. Let $v_t^{co} \in V_t$ be the correct value that the target column C^t should have for a given instance, and $v_t^o \in V_t$ be the observed value. In other words, v_t^o is the observed value of the column C^t in the suspicious instance. In our definition, we define an anomaly when the observed value is different from the correct value, i.e., when $v_t^{co} \neq v_t^o$. Given a clean subset of the data CD, a set of features C^f, and a target column C^t, we want to build a model M that can predict if the observed value of the target is anomalous or not (i.e., $v_t^{co} \neq v_t^o$) based on C^f and CD.

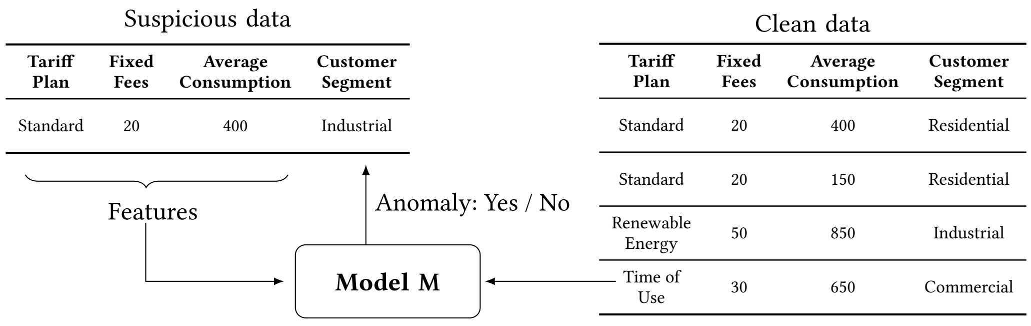

Suspicious data
Tariff Plan | Fixed Fees | Average Consumption | Customer Segment
Standard | 20 | 400 | Industrial
Features
Anomaly: Yes / No
Model M
Clean data
Tariff Plan | Fixed Fees | Average Consumption | Customer Segment
Standard | 20 | 400 | Residential
Standard | 20 | 150 | Residential
Renewable Energy | 50 | 850 | Industrial
Time of Use | 30 | 650 | Commercial

Fig. 1. Running example for problem definition

Application to the running example. Figure 1 shows an example illustrating the anomaly detection problem. We have a dataset of an energy provider with four columns, c_1: “Tariff plan”, c_2: “Fixed Fees”, c_3: “Average consumption”, and c_4: “Customer Segment”. Among the columns, “Tariff plan” and “Customer Segment” are categorical columns $C^c = \{c_1, c_4\}$. The table on the right-hand side of the figure represents the clean data *CD* and the small table in the left represents the suspicious data *SD*. Using the clean data, we want to build a model M to learn relationships between columns c_1 to c_4 in the clean data. This model is then used to check any instance on the suspicious data. Going back to the example, let us assume that we want to check if the value of the “Customer Segment” column in the suspicious test instance is an anomaly or not (i.e., C^t = “Customer Segment”). In this case, as shown in the figure, the rest of the columns are considered as features, i.e., $C^f = \{c_1, c_2, c_3\}$. Our goal is to use the model M to predict whether the observed value of “Customer Segment” (i.e., v_i^o = “Industrial”) represents an anomaly or not based on the values of $C^f = \{c_1, c_2, c_3\}$ and the clean data *CD*.

3 STATE OF THE ART

The approach proposed in this paper is related to anomaly detection for categorical data. Many research works tried to tackle this problem. The proposed approaches can be classified based on different aspects. In this section we followed the classification proposed by Taha and Hadi [48]: frequency-based, Bayesian/conditional frequency-based, density-based, clustering-based, distance-based, information-theoretic, and unsupervised/semi-supervised.

Frequency-based approaches [35, 37] rely on the frequency of categories to detect data anomalies. Less frequent categories are more likely to be anomalies. A typical frequency-based method is CBRW [35], which relies on two kinds of distributions (i.e., “intra-feature” and “inter-feature” distributions) to detect anomalies. The first one computes the frequency of categories on the target column, while the second one analyzes the distributions of categories in different columns. Based on these two distributions, CBRW computes an outlier score, and returns M instances with the highest outlier score as anomalies.

Bayesian/conditional frequency-based approaches [13, 33, 42] follow another definition of anomalies: no matter whether the categories are frequent or not, an infrequent combination of categories is considered as an anomalous instance. These approaches compute a rareness score between categorical values in an instance. Instances with a rareness value less than a certain threshold are considered anomalous.

Density based approaches detect anomalies in subgroups of data (referred as "local area") that share similar characteristics. A typical density-based method is WATCH [28]. WATCH detects anomalies in two phases, feature grouping and anomaly detection. First, it regroups related columns (i.e., features) having the same meaning or correlated to each other in the same group. Then, in the second phase, it detects anomalies in these groups by computing an anomaly score for each instance in each feature group. A higher anomaly score indicates a higher probability that an instance is anomalous regarding a feature group. WATCH declares the M instances with the highest anomaly score in each group as anomalies. The algorithm takes the union of anomalies sets in all feature groups to determine the anomalous instances. WATCH determines at most $M \times g$ instances as outliers, where g represents the number of features groups.

Clustering-based approaches [46, 47] define anomalous instances as the ones located in a sparse region from other clusters. For example, ROAD [46] determines k clusters on the data using the k-mode algorithm. Then it defines a set of big clusters having a number of instances higher than a certain threshold. To detect anomalies, ROAD computes the distance between the test instance and different clusters. A test instance has a higher chance to be an anomalous instance, if it has a larger distance with the nearest big cluster. ROAD finishes by providing M instances with the highest distances.

Distance-based approaches regard data instances far from the majority of instances as anomalies. An example of distance-based method is ORCA [6]. ORCA computes the distance between two categorical data instances using the Hamming distance, which measures the number of mismatches between them. For each instance i, ORCA computes an outlying score that is the average Hamming distance between i and its k-nearest neighbors. ORCA declares a parameter m, which is used to select the top-m instances with the highest outlying scores as anomalous instances.

Information theoretic approaches transform the problem of anomaly detection into an optimization problem [18, 19]. These approaches use information entropy to detect anomalous instances. One instance is considered to be an anomaly if the entropy of the dataset exhibits a large decrease after removing the instance. Specifically, these approaches first compute the entropy of the original dataset. Then, for each instance, they remove it in the dataset, compute a new entropy value of the dataset, and finally determine the difference between the original entropy value and the one obtained after removing the instance. The k instances with the highest difference in entropy value are selected, and returned as anomalous instances.

Unsupervised and semi-supervised approaches work as follows. The former are used when there is no information about the anomaly labels (i.e., anomalous or non-anomalous) of data instances. The baselines iForest, LOF, and EMAC are traditional unsupervised anomaly detection approaches. iForest [29] relies on an ensemble of decision trees to detect data anomalies where it decides instances with shorter average paths as anomalous. LOF [10] compares the density of one instance with its k nearest neighbors. Then it classifies the instances with lower density values when compared to neighbors as anomalies. EMAC-SCAN [51] is another unsupervised method, that takes advantage of embedding-based approaches to capture the relationship between categorical features and use them for anomaly detection.

Semi-supervised approaches take advantage of existing non-anomalous instances. In the training phase, they create a novelty model over data instances. Any test instance that deviates from normal data is considered anomalous. Our baseline OCSVM [11] uses the training data to find a hyperplane separating between anomalous and normal data; this hyperplane is used during the anomaly detection phase where all instances with a high distance to the hyperplane are considered as anomalous. Frac [34] uses non-anomalous instances to build an ensemble of classification models; during the test phase, Frac uses the predictions of previously trained models to determine anomalies. A test instance is considered as anomalous if there is a disagreement among the outputs of the different models. However, Frac is not designed for categorical data. To detect categorical anomalies with Frac, it should rely on any algorithm that can deal with categorical variables.

Limitations. Our preliminary experiments on commonly-used anomaly detection datasets show that the effectiveness of the SOTA approaches is unstable. For example, the precision of both iForest [29] and OCSVM [11] can vary between 0.03 and 0.9 depending on the datasets. Moreover, existing approaches are highly sensitive to the configuration parameters [52]. These algorithms include different parameters (e.g., OCSVM has two parameters) which need to be carefully chosen. In this work, we address these challenges and improve the anomaly detection precision. To detect data anomalies effectively, an intuitive solution is to infer the correct value in a categorical column; then, data anomaly can be detected by comparing the inferred correct value with the observed one. Another advantage is that, using such inferred value, a data anomaly can also be easily fixed.

Data Anomaly and Form Filling. Since one of the main sources of data anomaly is the wrong values entered by users during form filling [45], in the literature many form filling approaches [7, 30] have been proposed to accurately predict the correct value to be filled in a data entry form. In this work, we repurpose an automated form-filling approach to detect data anomalies. To do so, we need first to adapt these approaches to perform predictions offline instead of online (during the form-filling process). Moreover, form-filling approaches typically return to the user a ranked list of items. In order to detect anomalies, we need to fully take advantage of all this information. The main challenge is how to fully leverage the characteristics and outputs of form-filling approaches to effectively perform data anomaly detection.

4 PRELIMINARIES

In this section, we explain LAFF [7], a learning-based automated form filling suggestion tool, which provides the basic information for LAFF-AD to detect anomalies. LAFF is designed to suggest a ranked list of the most probable values to be selected by users for a categorical field during form filling.

LAFF has two main phases: model building and form filling suggestion. During the first phase, LAFF learns the relationship between different columns in the dataset where each column represents a field in the data entry form. LAFF builds a set of Bayesian network (BN) models over the training instances in the dataset. Specifically, LAFF learns one BN over the entire training set. This model is called "global model", which represents the general dependencies between columns. In order to capture fine-grained dependencies, LAFF applies the k-mode algorithm to cluster the data into k similar groups of instances. LAFF then applies a local modeling strategy by building a BN model on each cluster of data. At the end of this phase, LAFF builds k+1 BN models.

In the form filling phase, LAFF tries to select one of the models built in the previous phase to make a prediction. The model is selected based on the values of the filled fields by the user during the form filling process.

The selected model corresponds to the cluster with a minimal distance to the test instance. The global model is used in the case where there is no single model that can be selected. After the selection of the appropriate model, LAFF predicts a ranked list of the top values that are likely to be correct based on their probabilities. In order to avoid providing inaccurate suggestions, LAFF includes a heuristic-based endorser, which decides if the prediction is accurate enough or not. LAFF analyzes the dependency between the filled fields and the target, and the probabilities of the top-n suggested values. If there is a direct dependency between the target and the filled fields, or the sum of the probabilities of the top n values is higher than a certain threshold, then LAFF suggests the list to users.

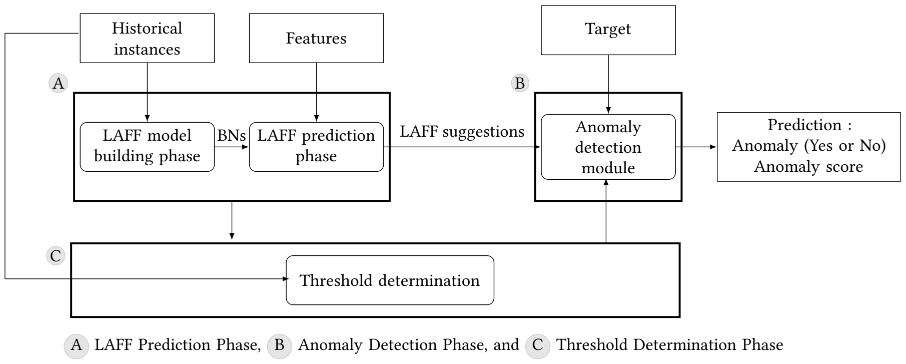

Historical instances
Features
Target
A
B
LAFF model building phase
BNs
LAFF prediction phase
LAFF suggestions
Anomaly detection module
Prediction : Anomaly (Yes or No) Anomaly score
C
Threshold determination
A LAFF Prediction Phase, B Anomaly Detection Phase, and C Threshold Determination Phase

Fig. 2. Main Steps of LAFF-AD

5 APPROACH

In this section, we show how to repurpose an automated form filling approach LAFF for performing anomaly detection of categorical data; we call the resulting approach LAFF-AD (LAFF-based Anomaly Detection).

As shown in Figure 2, LAFF-AD includes three phases: LAFF offline prediction, anomaly detection, and threshold determination. LAFF-AD starts by running a variant of LAFF on historical instances to train BN models. Then LAFF is used to predict the values of the target on suspicious data. In the second phase, based on LAFF's prediction, LAFF-AD uses a heuristic anomaly detection strategy to predict if there is an anomaly or not for a given test instance. LAFF-AD uses a threshold determination phase to automatically decide the values of its parameters.

In the rest of the section, we illustrate the three main phases of LAFF-AD.

5.1 LAFF offline prediction

LAFF is mainly designed to predict a ranked list of values for a categorical field during the form-filling process. LAFF-AD repurposes LAFF and takes advantage of it to detect data anomalies in categorical columns. The main reason behind repurposing LAFF to detect data anomalies is the high ability it has shown to correctly provide suggestions during form filling [7].

In the context of anomaly detection, LAFF needs to perform predictions offline on suspicious data that may contain anomalies. This means that LAFF-AD runs a variant of LAFF that handles offline prediction. This variant considers a special case during the form filling process, where all fields are filled except the target field. The model building phase is the same as in the original definition of LAFF. LAFF starts by preprocessing the historical instances and then creates different BN models (i.e., global and local models).

During the prediction phase, LAFF considers each instance in the suspicious data as data entered by a user. Since we aim to detect anomalies for a given categorical column (i.e., the target column) based on the values of other columns (i.e., the features), we consider the features as filled fields and use LAFF to predict the value of the target column on each suspicious instance. As we mentioned before, LAFF first tries to select one model to predict. This model is selected based on the feature values of the suspicious data. LAFF then makes predictions and returns a ranked list of values based on their probability. Thanks to its endorser module, LAFF has the ability to avoid inaccurate suggestions: it can label a suggestion as not endorsed if it has no sufficient confidence in the suggestion.

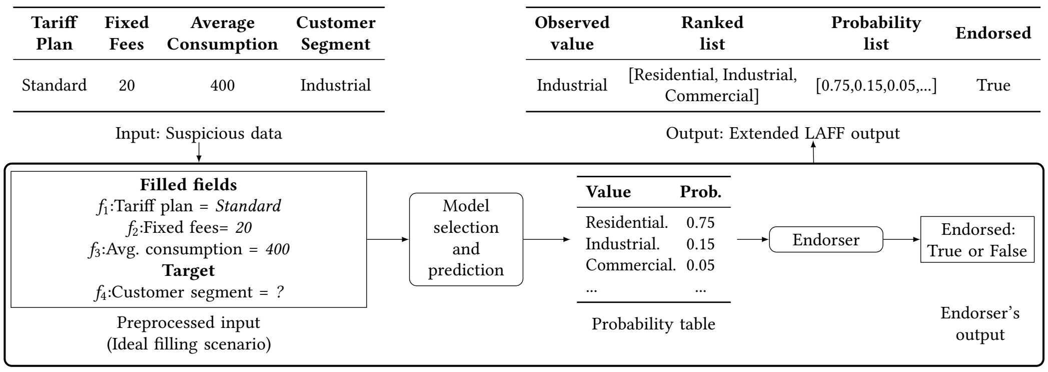


Fig. 3. LAFF prediction phase

Since LAFF-AD repurposes LAFF to detect data anomalies, the output of LAFF needs to be adapted to our context. Our variant of LAFF transforms the output of LAFF into a table representing a summary of the prediction result. This table contains the following information: "Observed value" representing the original value in the suspicious data, "Ranked list" representing the candidate values in the order suggested by LAFF, "Probability list" containing the probability of each value in the ranked list, and a Boolean column "Endorsed" that records whether the prediction of the current test instance is endorsed.

Application to the running example. Let us consider a suspicious test instance identified by domain experts. As shown on the left-hand side of Figure 3, this table contains the following values: "Tariff plan"= "*Standard*", "Fixed fees" = *20*, "Average consumption"= *400*, and "Customer segment" = "*Industrial*". This instance is considered to be anomalous because it deviates from the normal data instances, in which the value "*Standard*" for "Tariff plan" is usually associated with the value "*Residential*" for "Customer segment" and the value *100* for "Average consumption" (see Section 2.1). Let us assume that we want to check whether the "Customer segment" value is correct. In this case, the "Customer segment" column is the target and the remaining columns are features. LAFF preprocesses these values and uses them as input for model selection and prediction. Let us assume that LAFF predicts the following values "*Residential*", "*Industrial*", "*Commercial*" with the probability values *0.75, 0.15, 0.05*, respectively; the predicted list is checked by the endorser module of LAFF. Let us assume that the endorser threshold is equal to 0.8; in this case, the prediction should be endorsed since the sum of the top n values (0.75+0.15+0.05 = 0.95) is higher than the endorser threshold. The detailed output of this variant of LAFF is shown in the top-right part of the figure, in the form of a table. The original value in the suspicious data ("*Industrial*") is shown in column "Observed value"; the candidate values (in the order suggested by LAFF) and the corresponding probability values are shown in the second and third columns; the last column indicates the output of the endorser module ("True").

5.2 Anomaly detection phase

This is the main phase to detect data anomalies. It assumes that LAFF made a prediction over an instance from the suspicious data. LAFF-AD takes the output of LAFF as determined in the previous phase and uses it to detect anomalies based on a heuristic. The main steps of the anomaly detection algorithm are shown in Algorithm 1. The inputs of the algorithm are the output of LAFF

Algorithm 1: Anomaly detection

```
Input: Triple of ranked values, probability list, endorser decision $\langle rv, pl, endorsed \rangle$
       Target column $C^t$
       The value of $C^t$ in the suspicious data: $v_t^o$
       Threshold $\theta_t$
Output: A flag $checkERR_t$, representing the decision to label the suscpicious value $v_t^o$ as an anomaly
1  Boolean $checkEndorsed_t \leftarrow isEndorsed(endorsed)$;
2  topRankedValues $\langle rv_{top}, pl_{top} \rangle \leftarrow getTopNpRanked(rv, pl)$;
3  Boolean $checkNotTop_t \leftarrow isNotInTopNp(rv_{top}, v_t^o)$;
4  Float anomalyScore $\leftarrow 1 - getProb(v_t^o, pl)$;
5  Boolean $checkProb_t \leftarrow (anomalyScore > \theta_t)$;
6  $checkERR_t \leftarrow False$;
7  if $checkEndorsed_t$ then
8      if $checkNotTop_t$ and $checkProb_t$ then
9          $checkERR_t \leftarrow True$
10     end
11 else
12     $checkERR_t \leftarrow Not\ Conclusive$
13 end
14 return $checkERR_t$;
```

(consisting of the ranked values, the probability list, and the endorser decision), the target column C^t, its observed value in the suspicious data v_t^o, and the anomaly detection threshold θ_t. This threshold is used to predict the existence of data anomalies; its value is automatically determined (see Section 5.3).

Based on the output of LAFF, LAFF-AD collects three Boolean flags needed for anomaly detection: $checkEndorsed_t$, $checkNotTop_t$, and $checkProb_t$. First, LAFF-AD checks if the LAFF's prediction was endorsed, and saves the value to the Boolean flag $checkEndorsed_t$ (line 1). In order to assign a value to the $checkNotTop_t$ flag, LAFF-AD collects the list of the top-n ranked values by LAFF (line 2). Then, it checks if the observed value in the suspicious data v_t^o is not present in the top-ranked list (line 3). If so, the value of $checkNotTop_t$ is set to *True*. After that, LAFF-AD computes an anomaly score based on the probability of the observed value in the suspicious data v_t^o in LAFF's prediction (line 4) and saves the value in the variable *anomalyScore*. LAFF-AD checks if the value of the *anomalyScore* is higher than the anomaly detection threshold θ_t, and saves the value in the Boolean flag $checkProb_t$ (line 5).

After obtaining these three Boolean flags, LAFF-AD determines the existence of an anomaly based on the following three conditions. The first condition checks if LAFF's prediction is endorsed or not. The second condition checks whether the observed value of the target in the suspicious data is in the "top-n" values predicted by LAFF. The third condition checks if the anomaly score (i.e., $1 - getProb(v_t^o, pl)$) is higher than the anomaly detection threshold, where $getProb(v_t^o, pl)$ represents the probability of the observed value of the target in the ranked list. If the first condition is satisfied, it means that LAFF has enough confidence to predict and thus LAFF-AD can confidently detect anomalies. Regarding the second condition, any observed target value in the suspicious data that does not exist in the ranked list predicted by LAFF represents a potential anomaly, since LAFF usually predicts correct values at the top of the ranked list. As for the third condition, if the anomaly score $(1 - getProb(v_t^o, pl))$ is higher than the anomaly score threshold, this means that the observed value of the target represents a potential anomaly, since the probability of the observed value to be the correct one is very low.

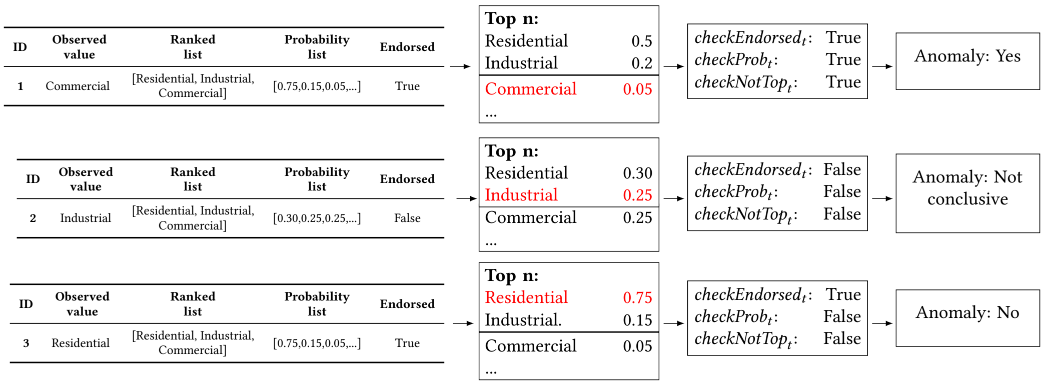


Fig. 4. Examples of LAFF-AD predictions

In Algorithm 1, LAFF-AD first checks if the LAFF prediction is endorsed (line 7). If the value of this flag is *True*, LAFF-AD checks the values of flags $checkProb_t$ and $checkNotTop_t$. If both $checkProb_t$ and $checkNotTop_t$ evaluate to *True*, LAFF-AD predicts there is an anomaly and sets the flag *checkERR* to *True* (line 10). If the LAFF's prediction is not endorsed, this means that we do not have enough information to make a prediction. We assign the value "Not conclusive" to the decision flag *checkERR* (line 12). The algorithm ends with returning the value of the *checkERR* flag.

Application to the running example. Figure 4 shows three instances that need to be checked by LAFF-AD. For each instance, we have the output from the LAFF offline prediction step (see section 5.1), which is the input for anomaly detection.

For the first suspicious instance in Figure 4, LAFF provides an endorsed prediction ($checkEndorsed_t$= *True*). In this case, LAFF-AD therefore has enough confidence to make a decision on the existence of an anomaly. LAFF predicts *Residential, Industrial, and Commercial* with the following probability list *0.5, 0.2, and 0.05*, respectively. If we assume that the value of the anomaly score threshold for the "Customer segment" column is equal to 0.9, LAFF-AD then predicts the existence of an anomaly since the $checkProb_t$ flag is evaluated to *True* ((1-0.05)≥ 0.9) and the observed value *Commercial* is not in the top-*n* values predicted by LAFF (i.e., $checkNotTop_p$= *True*).

The second case illustrates the scenario when LAFF-AD does not have the confidence to determine the existence of an anomaly. Specifically, even though the observed value in the suspicious instance is predicted in the top-*n* ($checkNotTop_t$=*False*) and the anomaly score is less than the threshold ($checkProb_t$= *False*), LAFF-AD decides to set the value of *checkERR* to *Not conclusive* since the prediction of LAFF is not endorsed.

The last case shows the scenario when a suspicious instance should not be treated as an anomaly. In fact, as shown in the figure, the prediction is endorsed by LAFF ($checkEndorsed_t$= *True*) and the observed value of the target is predicted at the top of the ranked list (i.e., $checkNotTop_t$=*False*). Concerning the flag $checkProb_t$, as shown in the figure, the probability of the observed value is very high (equal to 0.75) which leads to a low anomaly score (i.e., $checkProb_t$=*False*). Since both of the flags evaluate to *False*, LAFF-AD can confidently predict that this instance is normal.

5.3 Threshold tuning phase

In this phase, we aim to automatically determine the value of the anomaly score threshold for each target, in each dataset. This value is used in the anomaly detection algorithm and plays a

Algorithm 2: Threshold determination

```
Input: Set of pre-processed historical instances $I^H(t)_{tune}$ for tuning
       LAFF's predictions on historical instance for tuning: list of triples of ranked values, pl , endorser decision
       $\langle rv, pl, endorsed\rangle_{tune}$
Output: Dictionary of thresholds $\theta$
1  $\theta$ ←empty dict;
2  List of targets $targets \leftarrow getTargets(I^H(t)_{tune})$;
3  foreach $target\ t_i \in targets$ do
4      $temp_{th}$ ←empty dictionary;
5      for $n$= 0 to 1 (step 0.05) do
6          $predictedAnomalyAll = predictAnomalyAllInstances(I^H(t)_{tune_i}, \langle rv, pl, endorsed\rangle_{tune}, n)$;
7          $score = evaluate(I^H(t)_{tune_i}, predictAnomalyAll)$;
8          $temp_{th}[n] = score$;
9      end
10     $\theta[i] = getBestScore(temp_{th})$;
11 end
12 return $\theta$;
```

determinant role. This step assumes that LAFF was executed on a set of historical instances for tuning and that the LAFF predictions are available.

The basic idea is that, for a given dataset and target column, we try to detect anomalies in each instance of the tuning data set by varying the anomaly detection threshold θ_i. Then, for each threshold, we measure the prediction accuracy on the tuning set instances. The threshold on which LAFF-AD yields the highest accuracy is set as the target threshold.

The main steps of our threshold determination phase are shown in Algorithm 2. First, as inputs, the algorithm takes the set of preprocessed historical instances for tuning $I^H(t)_{tune}$ and LAFF predictions on these historical instances $\langle rv, pl, endorsed\rangle_{tune}$.

For each target t_i in the list of targets extracted from $I^H(t)_{tune}$ (line 2), based on LAFF predictions, we check the different tuning instances and analyze the existence of anomalies using the approach explained in section 5.2 (line 6). For the purpose of anomaly detection, we try different thresholds, ranging from 0 to 1 with steps equal to 0.05. For each threshold value, we compare predicted anomalies with actual anomalies of the target t_i in each input instance of $I^H(t)_{tune_i}$ to calculate the prediction accuracy (line 7). LAFF-AD selects the value of θ_i that leads to the highest prediction accuracy value for a target t_i in $I^H(t)_{tune_i}$ as the value of its threshold (line 10). The algorithm ends by returning a dictionary containing the thresholds of all targets.

6 EVALUATION

In this section, we report on the evaluation of our anomaly detection approach. We focus on two aspects, effectiveness and efficiency, which we compare with SOTA approaches. Efficiency is defined in terms of training and prediction time, to understand the suitability of an approach for practical applications. Moreover, we investigate the ratio of non-conclusive instances determined by LAFF-AD, and the impact of the Boolean flags used within the core LAFF-AD algorithm, as well as the percentage of anomalous instances on the accuracy. More specifically, we evaluated LAFF-AD by answering the following research questions (RQs):

RQ1 *Can LAFF-AD accurately detect data anomalies on categorical columns and how does it compare with existing anomaly detection approaches?*

RQ2 *Is the performance of LAFF-AD, in terms of training and prediction time, suitable for practical applications and how does it compare with existing approaches?*

Table 1. Description of Datasets

Dataset	# of columns	# of instances	# of categorical columns	Range of candidate values	Ratio of anomalous instances
NCBI-E	25	74105	5	3–84	n/a
U2R	7	60821	7	2–23	0.003
Probe	7	64759	7	2–47	0.064
CelebA	40	202599	40	2	0.022
Covertype	45	581012	45	2	0.004
Census	34	299285	34	2–47	0.062

RQ3 *What is the ratio of non-conclusive instances for which LAFF-AD does not yield an anomaly detection prediction?*

RQ4 *What is the impact of the Boolean flags used within the core LAFF-AD algorithm on its accuracy?*

RQ5 *What is the impact of the percentage of anomalous instances on the effectiveness of LAFF-AD?*

6.1 Dataset and Settings

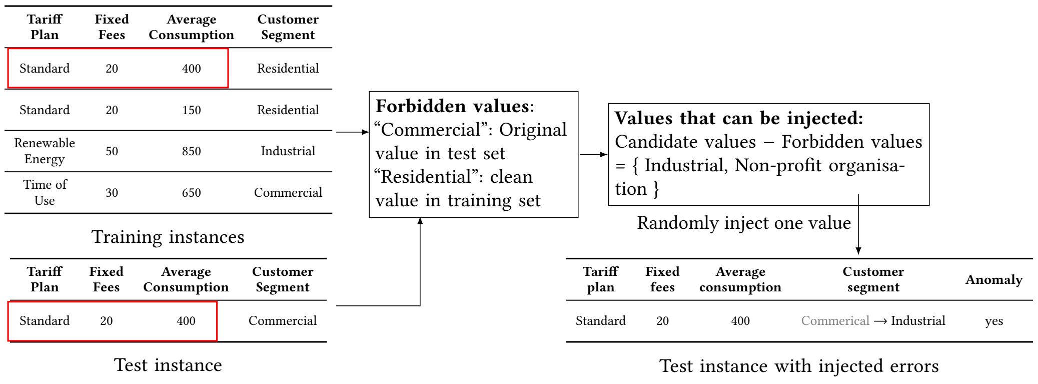
Datasets. Table 1 shows an overview of the datasets used in our evaluation, including the total number of columns (# of columns), the number of instances (# of instances), the number of categorical columns (# of categorical columns), the range of the number of possible values across these columns (Range of candidate values), and the ratio of anomalous instances.

The first dataset is NCBI-E, which is a variation of the public dataset NCBI from the biomedical domain. The original NCBI dataset contains data for different types of biological samples from multiple species [5]. The main reason to use this dataset is that it was used to evaluate LAFF [7]. Similar to LAFF, we considered a sub-sample of the data related to the species "Homo Sapiens". In our evaluation, we created NCBI-E by removing the column *ethnicity* from the NCBI dataset, since as shown in the existing study [7] only 15.6% of instances are non-empty in this column, which leads to incorrect suggestions from LAFF.

The remaining datasets are from the publicly-available benchmark that is commonly used to evaluate anomaly detection approaches for categorical columns [36]. The original benchmark has 12 datasets. Based on guidelines provided in our previous work [7], only the datasets with more than 56 000 instances available for training were selected for our evaluation, since LAFF achieves accurate suggestions only in this context: U2R, Prob, CelebA, Covertype, and Census. We therefore excluded the remaining datasets: Bank (41188), AID (4279), W7A (49749), CMC (1473), APAS (12695), Chess (28056), AD (3279), Solar (1066), and R10 (12897).

As shown in Table 1, the number of instances in these datasets varies from 60 821 (U2R) to 581 012 (Covertype). These datasets feature at least 7 columns, with more than 5 being categorical. The number of candidate values for these categorical columns varies significantly. For example, in the NCBI-E dataset, categorical columns feature between 3 and 84 candidate values, whereas the Covertype dataset contains only binary categorical columns. The ratio of anomalous instances varies from 0.003 to 0.064.

Dataset preparation. For the NCBI-E dataset, we considered all the categorical columns as possible targets. This dataset is not mainly designed to evaluate anomaly detection algorithms since anomalous values are not labeled. To solve this issue, we adopted an anomaly injection strategy which has been used to evaluate anomaly detection algorithms [1].



Training instances

Tariff Plan	Fixed Fees	Average Consumption	Customer Segment
Standard	20	400	Residential
Standard	20	150	Residential
Renewable Energy	50	850	Industrial
Time of Use	30	650	Commercial

Test instance

Tariff Plan	Fixed Fees	Average Consumption	Customer Segment
Standard	20	400	Commercial

Forbidden values:
“Commercial”: Original value in test set
“Residential”: clean value in training set

Values that can be injected:
Candidate values − Forbidden values = { Industrial, Non-profit organisation }

Test instance with injected errors

Tariff plan	Fixed fees	Average consumption	Customer segment	Anomaly
Standard	20	400	Commerical → Industrial	yes

Fig. 5. Error injection example

Following the methodology used by Das and Schneider [13], we injected synthetic anomalies in the NCBI-E dataset by randomly flipping the values of a target column. Specifically, we partitioned the NCBI-E dataset into three subsets of 80%, 10%, and 10% of instances, used for training, testing, and tuning, respectively. For each target, we created a separate test set where we kept the values in the feature columns the same and injected errors in the target column. We randomly selected a value that is different from the original value in the target column from candidate values. We then replaced the original value with the selected value to inject anomalies, and labeled the instance as anomalous.

For the other datasets, as mentioned by Pang et al. [36], each dataset has one target column that contains an anomalous value. Similar to NCBI-E, we split the datasets into three subsets containing 80%, 10%, and 10% of instances used respectively for training, testing, and tuning.

As suggested in different studies [22, 35, 47], and also visible in the last column of Table 1, there is usually a small ratio of anomalous instances in real-world data. For example, the ratio of anomalous instances used by Suri et al. [47] ranged from 10% to 16%. We followed the methodology used by Ienco et al. [22], setting the ratio of anomalous instances to 10% of instances for the NCBI-E dataset. For the other five benchmark datasets, which already contain different percentages of manually labeled anomalous instances, we used the original datasets without changing their percentage of anomalous instances. We analyze the impact of the percentage of anomalous instances on the effectiveness of LAFF-AD as part of RQ5 (Section 6.6).

Dataset preparation Example of Application. Figure 5 show an example of our error injection process. Let us consider the two tables on the left of the figure as the training (top left) and testing (bottom left) sets. Following our running example, the columns “Tariff plan”, “Fixed fees”, and “Average consumption” represent the features and the column “Customer segment” represents the target (where we want to inject errors).

Given the training and test instances, we need first to determine the set of forbidden values that should not be injected in the target column of the current test instance. This set should obviously contain the values *Commercial*, representing the original value of the test instance. But since we assume that the training set is clean, if there is any training instance with the same feature column values, the value of the target column in this instance must also be part of the forbidden values. For example, based on Figure 5, the value *Residential* should also not be injected to avoid having instances that are considered clean during training but anomalous during testing. After determining forbidden values, the remaining candidate values for the target (i.e., “Industrial” and “Non-profit organisation” in our example) can be injected into the test instance. In our example, we randomly select the value “Industrial” to be injected. After injection, this instance is considered “Anomalous” and we label it as such (i.e., filling ‘yes’ in the column Anomaly in the table on the right).

Implementation and Setting. LAFF-AD is implemented as a Python program. In order to run LAFF on different datasets, we used its default configurations mentioned in [7]. We performed experiments with a computer running macOS 10.15.5 with a 2.30 GHz Intel Core i9 processor with 32 GB memory.

6.2 Effectiveness (RQ1)

To answer RQ1, we analyzed anomaly detection with LAFF-AD for each of the targets in different datasets. We compared LAFF-AD with iForest (Isolation Forest) [29], LOF (Local Outlier Factor) [10], OCSVM (One Class SVM) [11], and EMAC-SCAN (Embedding-based coMplex vAlue Coupling learning framework) [51]. These approaches are commonly used as baselines to evaluate categorical data anomaly detection approaches [22, 37]. Moreover, there are publicly available replication packages including their implementations.

LOF uses the distance between instances to detect data anomalies. Given a data instance, LOF compares the distance between the instance and its nearest neighbors to assess density, which measures how closely packed the data instances among those neighbors. A data instance is considered an anomaly if it has a lower density value compared to its neighbors. In other words, anomalous data instances are relatively far from local groups.

OCSVM uses the training instances to iteratively find a hyperplane that separates normal instances from anomalies. In order to detect anomalous instances, OCSVM computes the distance between new instances and the hyperplane. Instances with high distances are marked as anomalous.

iForest builds an ensemble of decision trees over a given dataset. These decision trees are used to detect anomalous instances based on the number of splits needed to separate data instances. The intuition behind this algorithm is that anomalous data instances can be easily separated from normal instances. Based on this intuition, iForest classifies instances with a smaller average number of splits as anomalous.

EMAC-SCAN starts by embedding values of categorical features into continuous vectors by employing a skip-gram architecture (i.e., node2vec). These vectors represents the relationships between different categorical values. Then, the algorithm learns a coupling function that assigns an anomaly score to each vector. The algorithm considers instances with an error score higher than a certain threshold as anomalies.

We did not compare with other categorical data anomaly detection approaches such as WATCH [28] and ROAD [46] since the corresponding articles do not include any link to a replication package; moreover, we could not find any third-party implementation of these approaches on open-source platforms (e.g., GitHub).

Effectiveness metrics. In order to select the evaluation metric, we checked around 25 papers included in a recent survey [48] on anomaly detection algorithms for categorical data. We also checked other recent papers citing these papers.

Table 2 presents a summary of our literature review. This table contains three columns, showing the metrics, their description, and the number of papers using each metric. As shown in the table, seven metrics have been commonly used. Precision, Recall, AUC (Area under Curve), and ROC (Receiver Operator Characteristic) curve are the most used ones. AUC is based on the ROC

Table 2. Main metrics used in the area of anomaly detection for categorical fields

Metric	Description	Cited papers
Precision	The fraction of correctly detected anomalies among all the predicted anomalies [42].	5
Recall	The fraction of correctly detected anomalies among all the anomalous instances [42].	7
Accuracy	The fraction of correctly predicted anomalies among the total number of predictions. [42]	3
Number of detected anomalies	The number of correctly detected anomalies.	2
F_1-score	The harmonic mean of precision and recall [4].	1
ROC curve	ROC plots the true positive rate and false positive rate of a classification model [21].	4
AUC	The ability of a binary classifier to distinguish between classes; it is used as a summary of the ROC curve [9].	9

curve, which is usually used to evaluate the performance of a classification model under different thresholds. Therefore, we use precision, recall, F_1-score, and AUC as evaluation metrics.

Methodology. We assessed the accuracy of different algorithms using Precision (Prec), Recall (Rec), F_1-score (F_1), and AUC, which are computed from the confusion matrix summarizing the classification outputs. The confusion matrix includes True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). In our context, TP are correctly classified as an anomaly, FP are wrongly classified as an anomaly, TN are correctly predicted as not anomalous, and FN are misclassified as not anomalous. Based on the confusion matrix, we have $Prec = \frac{TP}{TP+FP}$ and $Rec = \frac{TP}{TP+FN}$. [1]

Precision is the ratio of correctly detected anomalies over all the values classified as anomalies. Recall is the ratio of correctly predicted anomalies over all actual anomalies. High values of precision and recall imply that an algorithm can correctly detect most anomalies. In the scenario of anomaly detection, the main goal is to successfully detect all the anomalies in a dataset, while avoiding predicting non-anomalous values as anomalies. In other words, our approach needs to have both high precision and recall values.

To measure the overall accuracy, F_1-score and AUC can be used. The F_1-score is the harmonic mean between precision and recall, i.e., $F_1 = \frac{2*Prec*Rec}{Prec+Rec}$. AUC is computed based on the area under the ROC curve. ROC stands for Receiver Operating Characteristic and is the plot of the false positive rate $\frac{FP}{FP+TN}$ against the true positive rate (i.e., Recall). AUC is distributed between zero and one. The higher the value, the better the performance of an anomaly detection approach. A value of 0.5 indicates that the performance is roughly equal to random guessing.

Regarding the four baselines, OCSVM, LOF and iForest are used to detect anomalies in numerical data [37]. To be able to run these algorithms on our categorical data, it is necessary to convert categorical columns into numerical ones. One common conversion method used in categorical anomaly detection is *1-of-l* [16]. This method converts a categorical column with l candidate values

[1]We note that LAFF-AD can label a data instance as "non-conclusive"; in this case, the instance does not belong to any class of the confusion matrix. Therefore, we removed "non-conclusive" predictions when computing the confusion matrix.

Table 3. Main parameters used in each anomaly detection approach

Alg.	Parameter	Dataset					
		NCBI-E	U2R	Probe	CelebA	Covertype	Census
LOF	n_neighbors	130	10	20	130	10	20
	contamination	0.01	0.01	0.01	0.01	0.01	0.01
OCSVM	γ	0.001	0.001	0.001	0.001	0.001	0.001
	nu	0.1	0.4	0.3	0.1	0.1	0.1
iForest	n_estimators	80	60	80	60	70	80
	contamination	0.01	0.01	0.01	0.01	0.01	0.01
	max_samples	256	256	256	256	256	256
EMAC-SCAN	n/a	*	*	*	*	*	*
LAFF-AD	θ	0.10	0.10	0.70	0.10	0.75	0.10

to l binary columns representing each categorical value. Such a column can take the values "1" or "0", representing respectively if the categorical value associated with the column is selected or not. Further, these algorithms have different parameters to set, which may affect their effectiveness. Since the values of these parameters may vary from one dataset to another, in order to ensure a fair comparison, we explored the settings used for these parameters in related works [11, 29, 50]. For each algorithm, if there is only one parameter value used in different papers, we consider it as default value for this parameter. If a parameter was set with different values in related works, we tune this parameter using grid search. The tuning follows this strategy: for each parameter we explore a tuning range from the minimum to the maximum values reported in the literature in steps of 10 or 0.1; for each algorithm we then use the parameter values with the highest accuracy for a given dataset.

Table 3 lists the main parameters used for each anomaly detection approach, on each dataset. For example, the threshold θ of LAFF-AD was set, after the threshold tuning phase, to 0.10 on the NCBI-E dataset. Since we used the implementation of LOF, OCSVM, and iForest provided by the `sklearn` library, the name of parameters in the table follows the naming conventions of `sklearn`. We note that the implementation of these techniques has numerous parameters in `sklearn`; for instance, LOF contains parameters such as *leaf_size*, *metric*, and *novelty*. We only set the parameters listed in Table 3, since they are frequently tuned in previous studies [11, 29, 50]; we did not specify the values of other parameters, and relied on their default values. For EMAC-SCAN, we directly used the publicly available replication package, which did not require to set any parameter.

Results. Table 4 shows the results of the various algorithms on the different datasets used in our evaluation; the highest value of each evaluation metric on each dataset is set in bold. Column *Alg* indicates the algorithm, while columns *Prec*, *Rec*, F_1, and *AUC* indicate the precision, recall, F_1-score, and AUC values, respectively.

Starting with the results on the NCBI-E dataset where anomalies are randomly injected, LAFF-AD outperforms baselines in terms of all four evaluation metrics, ranging from +59 pp to +64 pp for Prec, +4.5 pp to +26.3 pp for Rec, +54.3 pp to +60.3 pp for F_1-score, and +22.8 pp to +50.1 pp for AUC. When we compare the recall value of LAFF-AD and LOF, both approaches have similar recall value but LAFF-AD performs much better in terms of Prec (0.808 vs. 0.164).

Looking at the results on the benchmark datasets, LAFF-AD substantially outperforms the baselines for the U2R, CelebA, Covertype, and Census datasets when we compare the F_1-score of these approaches. More in detail, LAFF-AD outperforms all the baselines on these datasets in terms of precision by at least 43.8 pp; in terms of recall, LAFF-AD outperforms the other baselines

Table 4. Anomaly Detection Effectiveness

	Alg.	Accuracy				Train	Predict (ms)	
		Prec	Rec	F_1	AUC	(s)	avg	min–max
NCBI-E	OCSVM	0.202	0.707	0.314	0.666	21	3	3–3
	LOF	0.164	0.925	0.279	0.734	61177	33	27–57
	iForest	0	0	0	0.461	264	36	30–64
	EMAC-SCAN	0.213	0.828	0.339	0.651	0	1	0.767–2
	LAFF-AD	**0.808**	**0.970**	**0.882**	**0.962**	1522	14.81	6–37
U2R	OCSVM	0.016	**1**	0.031	0.928	10585	5	4–9
	LOF	0.384	0.333	0.357	0.699	24813	0.6	8–13
	iForest	0.169	0.733	0.275	0.862	43	23	23-23
	EMAC-SCAN	0.562	0.6	0.58	0.799	0	0.74	0.71–0.75
	LAFF-AD	**1**	**1**	**1**	**1**	535	6	5–15
Probe	OCSVM	0.183	**1**	0.309	0.928	2136	3	3–3
	LOF	0.770	0.241	0.367	0.619	14330	7	7–7
	iForest	0.940	0.892	**0.915**	**0.945**	52	27	23–48
	EMAC-SCAN	0.864	0.851	0.857	0.894	0	0.864	0.860–0.868
	LAFF-AD	**0.981**	0.266	0.419	0.623	984	8	5–37
CelebA	OCSVM	0.052	0.535	0.095	0.72	709	2	2–2
	LOF	0.005	0.005	0.005	0.503	10359	10	9–14
	iForest	0.134	0.153	0.143	0.572	204	18	17–22
	EMAC-SCAN	0.112	0.295	0.162	0.663	0	21	18-25
	LAFF-AD	**1**	**1**	**1**	**1**	6187	479	443–712
Covertype	OCSVM	0.003	**1**	0.006	**0.815**	3085	2	2–2
	LOF	0.107	0.085	0.095	0.542	9630	12	11–16
	iForest	0.124	0.554	0.203	0.773	643	21.172	20–25
	EMAC-SCAN	0.036	0.973	0.069	0.600	0	95	93– 97
	LAFF-AD	**1**	0.570	**0.726**	0.782	4450	476	425–735
Census	OCSVM	0.062	0.211	0.096	0.559	10916	5	5–5
	LOF	0.008	0.002	0.003	0.497	18208	74	63–105
	iForest	0.047	0.015	0.023	0.503	1504	30	24–38
	EMAC-SCAN	0.054	0.649	0.100	0.676	0	74	74 – 80
	LAFF-AD	**0.970**	**0.928**	**0.949**	**0.961**	6710	186	27-330

by at least 26 pp, except for the Covertype dataset where the recall of EMAC-SCAN is higher than LAFF-AD by around 40 pp. In this last case, we remark that, even though the recall value of EMAC-SCAN is very high compared to LAFF-AD, the precision value is very low (Prec=0.036). This means EMAC-SCAN always predicts instances as anomalous. On the contrary, LAFF-AD can accurately detect almost 60% of the anomalous instances with a precision equal to 1. Regarding AUC, LAFF-AD outperforms the baselines on three out of the four datasets (i.e., U2R, CelebA, and Census); on Covertype, the difference in terms of AUC value with respect to OCSVM is only 3.3 pp.

Looking at the results of the Probe dataset, we can notice that both baselines iForest and EMAC-SCAN outperform LAFF-AD in terms of Rec by almost 60 pp. Also, these baselines provide accurate suggestions with F1-score equal to 0.915 and 0.857 for iForest and EMAC-SCAN, respectively. We analyze the reasons of the weak results of LAFF-AD on the Probe dataset in § 6.2.1 below.

Table 5. Maximum Carmér's V for different datasets

Dataset	Max Carmér's V	Column
U2R	0.664	Service
Probe	0.848	Flag, Service
CelebA	0.223	Att13
Covertype	0.275	Dim-17
Census	0.438	Att4

Comparing the results of the baselines between (Probe, U2R) and (Census, Covertype and CelebA), we notice that the results of the baselines are very low for Census, Covertype and CelebA, especially in terms of Prec. One possible reason can be the values of the parameters used in these baselines; however, we have tried to use tuning to solve this problem. Besides, we believe that the high number of columns in the dataset (34 and 45 columns) may be the reason of the low precision [26]. High-dimensional datasets (i.e., with high number of columns) are challenging ML algorithms [3]. High dimensionality can make it difficult for machine learning algorithms to learn information from observed data, a problem referred to as the curse of dimensionality [43]. For example, the Census dataset is composed of 34 categorical columns where the range of number of candidate values varies from 2 to 47. As we mentioned before, we need to transform these columns to binary columns in order to run our baselines. The number of columns in the transformed Census dataset is equal to the sum of the number of candidate values of each categorical column in the dataset; this value may affect the ability of our baselines.

The results in terms of Precision and Recall values achieved by LAFF-AD show that it can help correctly ($prec \geq 0.808$) detecting a high ratio of data anomalies in different datasets with a recall ranging from 0.570 to 1. The values of F_1-score and AUC also confirm this conclusion.

6.2.1 Error analysis. As presented in Table 4, the recall value of LAFF-AD for the *Probe* dataset is low compared to the results of LAFF-AD on other benchmark datasets. In order to understand the reason, we checked the prediction details of LAFF-AD in *Probe* and found that all the missed anomalies predicted by LAFF-AD have an anomaly score (as defined in § 5.2) of 0. This means that the form filling tool LAFF predicts the anomalous values in the top first position of the ranked list with a prediction confidence of 100%. Here the anomalous values represent values or dependencies that seldom occur in the data. LAFF-AD cannot detect the anomaly because the anomalous values are predicted in the top of the ranked list (leading to $checkNotTop = False$ when executing Algorithm 1) and with low anomaly score (leading to $checkProb = False$).

We further analyze the reason behind the high confidence of the prediction of the anomalous values. To do so, we performed the Chi-square test [14] and use Carmér's V [25] to compute the association between all the features and the target class for each dataset. Carmér's V is a typical measurement to compute the association of two categorical columns.

The results of this test are presented in Table 5 where we show the maximum Carmér's V value and the name of columns having this value for each dataset. As shown in the table, the Probe dataset contains two strongly dependent features ("flag" and "service") with the target class where both of them have a Carmér's V equal to 0.844. However, for other datasets, there are no fields with such strong dependencies to the target class. A strong association between two categorical columns means that the value of one column depends on the value of another column [24]. Since we have strong association between the "service" or "flag" column and the target class, the value of the latter is related to the value of the former. LAFF is a form filling recommender system completely

Table 6. Results on Probe variations

	Prec	Rec	F_1	AUC
Probe	0.981	0.266	0.419	0.623
Probe-S	0.935	0.295	0.448	0.630
Probe-F	0.857	0.215	0.344	0.607
Probe-SF	1	1	1	1

agnostic to anomaly detection; it learns the dependencies between the columns and use them to perform prediction of the target class. In the *Probe* dataset, LAFF uses the dependencies between "flag", "service", and the target class to perform prediction. Since these dependencies are strong (due to the strong association), LAFF always returns a high confidence value for each prediction. This occurs even if the predicted value turns out to be anomalous. Hence, during the anomaly detection phase, the anomaly score is low, and the flag $checkProb_t$ in Algorithm 1 evaluates to false. As a result, LAFF-AD is not able to detect these anomalies.

In order to check if the strong dependency between "service" and "flag" features and the target class is the reason behind the inability of LAFF-AD to detect data anomalies (resulting in a low recall value), we ran LAFF-AD on four variations of the Probe dataset. We created the following datasets "Probe-S", "Probe-F", and "Probe-SF" representing the Probe dataset after removing the "service" column, the "flag column", and both of them from the dataset, respectively. The results of this experiment are presented in Table 6, where we computed the precision, recall, F_1-score, and AUC of LAFF-AD on the four variations. As shown in the table, removing only one of the columns "service" and "flag" does not affect the results of LAFF-AD. In fact, the results of LAFF-AD on Probe-s and Probe-F are quite similar to those obtained on the original Probe dataset. For the third variation, when we remove both columns, we can see that LAFF-AD can accurately detect all the anomalous instances (i.e., achieving a recall value of 1) with a precision value equal to 1. These results confirm our hypothesis that the strong association (dependency) with the "flag" and "service" columns leads LAFF to learn rare behaviors and predict the anomalous class with high confidence. This problem can be easily addressed during preprocessing, by removing columns that have degree of association higher than a certain threshold [31].

The answer to RQ1 is that LAFF-AD performs better than SOTA baselines on five datasets out of six used in our evaluation by at least 43.8 pp *and* 26 pp *in terms of precision and recall, respectively. The F_1-score and AUC of LAFF-AD are also higher than the baselines on the majority of datasets. LAFF-AD can detect at least 26% of data anomalies with a precision above 0.8.*

6.3 Performance (RQ2)

To answer RQ2, we measured the time needed to perform model training (considered as training time) and the time needed to predict anomalies (considered as prediction time). Training time reflects the feasibility of using LAFF-AD in contexts where the training set is regularly updated with new instances. Prediction time indicates the ability of LAFF-AD to detect anomalies in a short period of time, for example as new data is acquired.

Methodology. In this RQ, we followed the same settings as RQ1, where we compared the time needed by LAFF-AD to train and predict with the same baselines that we used in RQ1. Training time measures the time to learn from the training data in order to detect data anomalies. Prediction time represents the average time needed by an algorithm to perform prediction for one suspicious instance. LAFF-AD's prediction time is measured as the sum of the prediction time taken by LAFF and the time taken by our anomaly detection algorithm.

Results. The results of this RQ are presented in the last two columns, *Train* and *Predict*, in Table 4. The *Train* column reports the training time in seconds whereas the *Predict* column contains two subcolumns indicating the average prediction time and the minimum/maximum time (in milliseconds).

As expected, training time for an algorithm varies from one dataset to another. LOF has the highest training time across datasets with a minimum of 9630 s. EMAC-SCAN and iForest are the fastest algorithms: EMAC-SCAN does not require training while for iForest the training time is at most 1504 s. As for LAFF-AD, the training time is less than 7000 s.

In terms of prediction time, LAFF-AD has the highest prediction time when compared to all the baselines. As we mentioned before, LAFF-AD's prediction time is the sum of the prediction time taken by LAFF and the time taken by our anomaly detection algorithm. As shown in Table 4, LAFF-AD takes on average at least 6 ms to perform prediction and at most 479 ms. The reason behind the high prediction time for Covertype and CelebA is that the prediction relies on complex BNs. The complexity of the BN is defined in terms of the number of nodes (one node corresponds to one column) and the number of dependencies between different columns. As shown in Table 1, Covertype and CelebA have the highest number of columns among datasets.

These results need to be interpreted in our context. The training for the anomaly detection process is done offline and periodically. A training time of at most 7000 s is acceptable from a practical standpoint. This training time allows LAFF-AD to be trained daily if needed, especially when the training data is updated daily with thousands of instances.

Since anomaly detection is an offline process, a prediction time of at most 735 ms is fast enough. It can even enable online anomaly detection during the form filling process. Indeed, human-computer interaction standards [20] indicate that a seamless interaction between a user and a data entry form can be ensured with a prediction time below 1 s. Since the prediction time proposed by LAFF-AD is comparable with the results of LAFF [7], this further confirms the possibility of using LAFF-AD to perform online data anomaly detection.

The answer to RQ2 is that the performance of LAFF-AD, with a training time below 7000 s *(less than 2 hours) and a prediction time of at most* 735 ms, *is suitable for practical applications. The training time of LAFF-AD usually lies between that of EMAC-SCAN and LOF. Concerning prediction time, LAFF-AD has a higher time compared to baselines but the difference has no practical implications since anomaly detection is an offline process.*

6.4 Ratio of Non-Conclusive Prediction (RQ3)

The output of LAFF-AD is different from existing anomaly detection approaches, because LAFF-AD includes a "non-conclusive" label, which indicates that LAFF-AD does not have enough information to make an anomaly detection prediction on a data instance. Since it may affect the usability of LAFF-AD in practical scenarios, this RQ investigates the ratio of non-conclusive predictions.

Methodology. We ran LAFF-AD on the testing instances of the six datasets used in our evaluation and counted the number of instances labeled as "non-conclusive" by LAFF-AD. We then computed the ratio ρ_{nc} of non-conclusive instances in these datasets by dividing this number with the total number of testing instances.

Results. As shown in Table 7, for four out of the six datasets, LAFF-AD did not label any testing instance as "non-conclusive". This means that LAFF-AD was able to perform the anomaly detection task on all the data instances. For the Probe dataset, there was only one non-conclusive instance,

Table 7. Ratio of non-conclusive predictions

Dataset	# of Non-Conclusive	ρ_{nc}
NCBI-E	1058	0.174
U2R	0	0
Probe	1	0.00016
CelebA	0	0
Covertype	0	0
Census	0	0

resulting in ρ_{nc} = 0.00016. However, the ratio of non-conclusive instances is relatively higher in the NCBI-E dataset (ρ_{nc} = 0.174), where LAFF-AD labeled 1058 instances as non-conclusive.

We believe the main reason behind this difference is the quality of the training sets. As discussed in our previous work [8], *NCBI* is a public dataset where anyone can submit data using the corresponding data entry form. Users do not follow any rule to submit data, potentially leading to a low-quality dataset (e.g., with wrong or empty values). When training LAFF-AD on such a dataset, LAFF-AD may not have enough knowledge to learn the pattern of normal data instances. Consequently, during the anomaly detection phase, LAFF-AD labels many non-conclusive instances to avoid providing misleading predictions. In contrast, the other five datasets used in this study are commonly used to evaluate anomaly detection approaches [36]; these are high-quality datasets, whose data have been manually examined and labeled. The number of non-conclusive instances reported by LAFF-AD on such datasets can be expected to be negligible.

The inclusion of the non-conclusive label in LAFF-AD was influenced by our previous work [7], in which we noted that users might be reluctant to use an automated tool if it provides many inaccurate predictions. From a practical standpoint, when the ratio of non-conclusive instances is low (as it is the case in most of the datasets considered in this work), users can manually check these instances in practice. On the other hand, when LAFF-AD labels many non-conclusive instances on a dataset (as it did in the case of the NCBI-E dataset), this can indicate a quality problem with the dataset and makes the application of LAFF-AD less beneficial, though still useful. Other means to analyze non-conclusive instances should be investigated in the future.

The answer to RQ3 is that, on five out of six datasets, LAFF-AD yields only one non-conclusive instance, leading to a negligible ratio of non-conclusive instances (0.00016) for one dataset. On the NCBI-E dataset, due to the low quality of the data, the ratio of non-conclusive instances rises to 0.174. We therefore expect LAFF-AD to be highly applicable in many practical situations.

6.5 Impact of Boolean Flags (RQ4)

LAFF-AD uses two Boolean flags, namely $checkNotTop_t$ and $checkProb_t$, to determine whether an endorsed (conclusive) data instance is an anomaly or not (see Algorithm 1). $checkNotTop_t$ indicates whether the observed value in the suspicious data is not present in the top-ranked list; $checkProb_t$ indicates whether the value of the *anomalyScore* of the suspicious data is higher than the anomaly detection threshold. One could wonder whether these two flags are redundant. In this RQ, we assess the impact of these two flags on the effectiveness of LAFF-AD.

Methodology. We measured the effectiveness of LAFF-AD considering two additional variants representing alternative configurations of LAFF-AD: LAFF-AD$_{top}$ and LAFF-AD$_{prob}$. LAFF-AD$_{top}$ represents the configuration where only the $checkNotTop_t$ flag is used to detect data anomalies; dually, LAFF-AD$_{prob}$ denotes the configuration where LAFF-AD only uses the $checkProb_t$ flag. We

Table 8. Effectiveness of LAFF-AD with different Boolean flags

Dataset	LAFF-AD$_{top}$				LAFF-AD$_{prob}$				LAFF-AD			
	Prec	Rec	F$_1$	AUC	Prec	Rec	F$_1$	AUC	Prec	Rec	F$_1$	AUC
NCBI-E	0.782	0.970	0.866	0.960	0.512	0.996	0.676	0.886	0.808	0.970	0.882	0.962
U2R	1	1	1	1	1	1	1	1	1	1	1	1
Probe	0.867	0.287	0.431	0.633	0.896	0.354	0.507	0.676	0.981	0.266	0.419	0.623
CelebA	1	1	1	1	1	1	1	1	1	1	1	1
Covertype	0.294	0.676	0.423	0.704	1	0.570	0.726	0.782	1	0.570	0.726	0.782
Census	0.970	0.928	0.949	0.961	0.970	0.928	0.949	0.961	0.970	0.928	0.949	0.961

run LAFF-AD and the two additional variants using the same settings and evaluation metrics as in RQ1 to analyze the impact of these flags.

Results. As shown in Table 8, on the U2R, CelebA, and Census datasets, there is no difference in terms of the four evaluation metrics when the different flags are used. This means that both flags can be used to accurately identify data anomalies on the endorsed data instances. On the Covertype dataset, *checkProb*$_t$ is more important than *checkNotTop*$_t$, since the accuracy of LAFF-AD$_{prob}$ is the same as LAFF-AD (LAFF-AD$_{top}$ returns many false positives). On the NCBI-E and Probe datasets, using both flags largely reduces the number of false positives in LAFF-AD. For example, on the NCBI-E dataset, the precision values of LAFF-AD$_{top}$ and LAFF-AD$_{prob}$ are 0.782 and 0.512, respectively; after enabling both flags, LAFF-AD increases the precision to 0.808.

The impact of the two flags on the NCBI-E and Probe dataset can be explained by the characteristics of the two datasets. On the one hand, the quality of the training sets for NCBI-E is poor, since NCBI-E is a public dataset where anyone can submit data through the corresponding data entry form without further data validation check (as shown in RQ3). On the other hand, there are strong dependencies between the columns of Probe (as shown in RQ1). On such two datasets, LAFF-AD may not have enough knowledge to correctly learn data anomaly patterns, leading to similar probability values for different suggestions. In this case, both the position (i.e., *checkNotTop*$_t$) and probability (i.e., *checkProb*$_t$) of the suggestions should be checked to avoid providing users with many incorrect and misleading results.

The answer to RQ4 is that, the two flags can improve the overall accuracy of LAFF-AD. On the NCBI-E and Probe datasets, using both flags together can reduce false positives. On other datasets, using both flags has no impact; any of them can be used.

6.6 Impact of the Percentage of Anomalous Instances (RQ5)

To answer RQ5, we assess the impact of the percentage of injected anomalous instances on the effectiveness of LAFF-AD. As discussed in Section 6.1, five out of the six benchmark datasets used in this work (U2R, Probe, CelebA, Covertype, Census) contain manually labeled and verified anomalies (see also Table 1). We did not vary the percentage of anomalous instances in these five datasets, as there is no guidance (in the benchmark description) to automatically create additional anomalous instances that follow the same data characteristics as the existing anomalies. Therefore, to answer this RQ, we only considered the NCBI-E dataset, which does not contain any labeled anomalous instance.

Methodology. We evaluated LAFF-AD by varying the percentage p of injected anomalous instances from 1 % to 20 % in the NCBI-E dataset, with a step of 5% (i.e., p = 1%, 5%, 10%, 15%, 20%). We did not go beyond 20% since there is usually a small ratio of anomalous instances in real-world

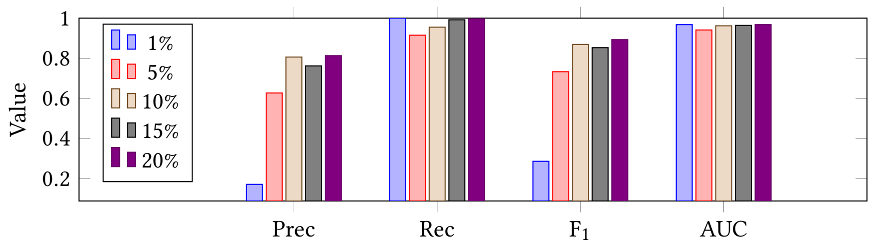


Fig. 6. Impact of the percentage (p = 1%, 5%, 10%, 15%, 20%) of anomalous instances on the effectiveness of LAFF-AD

data [22, 35, 47]. For each dataset variant associated with a given percentage value, we adopted the same settings as in RQ1 to assess the effectiveness of LAFF-AD.

Results. As shown in Figure 6, the percentage of injected anomalous instances affects the accuracy of LAFF-AD. As p increases, the precision and F_1-score of LAFF-AD also increase, stabilizing after $p > 10\%$. For example, the F_1-score is only 0.286 when $p = 1\%$ but increases to 0.882 when $p = 10\%$. On the other hand, the AUC values are almost stable, ranging from 0.941 ($p = 5\%$) to 0.968 ($p = 20\%$). The difference in trend between F_1-score and AUC can be explained by the fact that a small percentage of injected anomalous instances in the dataset (i.e., tuning set) makes it difficult for LAFF-AD to find the best threshold for anomaly detection, thus resulting into many false positives.

The answer to RQ5 is that the accuracy, in terms of F_1-score, increases when more anomalous instances are present in a dataset, while the AUC remains stable.

6.7 Threats to Validity

The size of the training sets is a common threat to all machine learning-based approaches. LAFF-AD is mainly designed to be used on datasets related to enterprise software systems, these datasets usually contain enough data for training LAFF-AD. In addition, these systems are often updated daily with thousands of new instances, which makes this limitation not particularly relevant.

To increase the generality of our results, LAFF-AD needs to be evaluated on different datasets from different domains. To deal with this issue, we evaluated LAFF-AD using benchmark datasets that have been previously used to evaluate anomaly detection approaches for categorical fields. Moreover, we selected datasets from different domains (e.g., biomedical, security, and finance). As shown in Table 1, these datasets also have different characteristics with respect to the number of rows, columns, and the range of categorical data. Also, we tried to evaluate LAFF-AD on different kinds of datasets, including datasets with synthetically-injected anomalies through our error injection strategy (i.e., the NCBI-E dataset) and benchmark datasets with real anomalies.

The implementation of the baselines can be considered an external threat. To minimize this threat, we used the official implementation of LOF and iForest in the `sklearn` library [38]. As for EMAC-SCAN, we used the available implementation provided by Xu et al. [51]. All the scripts used to get the results were double-checked by the first two authors of the article.

Another threat is the choice of parameter values of the baselines (i.e., iForest, OCSVM, and LOF). The value of the parameters depends mainly on the dataset because some values can work with one dataset but yield poor results on another dataset. In order to mitigate this threat, we checked the literature to identify possible ranges of values for each parameter in these baselines. We performed parameter tuning using grid search to select optimal parameter values for each baseline on each dataset.

6.8 Data Availability

The implementation of LAFF-AD, the different datasets, and the scripts used for the evaluation are available at https://doi.org/10.6084/m9.figshare.24648387.

7 DISCUSSION

7.1 Usefulness

The main goal of LAFF-AD is to detect data anomalies in a suspicious data set. In order to evaluate the anomaly detection ability of LAFF-AD, we applied it on 6 datasets from different domains with varying characteristics, such as the number of rows and columns. Five out of six of these datasets are commonly used to evaluate existing anomaly detection methods and one of them (i.e., “NCBI-E”) was used to evaluate LAFF, on which we build here.

Our results show that LAFF-AD outperforms different baselines on nearly all datasets (except Probe, see section 6.2) with a recall value of at least 0.570 and a precision higher than 0.808. In the context of anomaly detection, these results indicate that LAFF-AD can accurately detect at least 57% of the anomalous instances in the data, which is of practical significance. The high precision (Prec $\geq$ 0.808) also suggests a low number of *false alarms*). For some datasets (NCBI-E, Census, U2R, and CelebA), very high Recall imply that LAFF-AD can detect almost all the anomalies.

Regarding performance, LAFF-AD performs training in less than 6710 s (less than 2 hours) and has a prediction time of at most 735 ms. These results suggests we can deploy LAFF-AD during the form-filling process as an additional step for data quality check. In other words, LAFF-AD can be useful to check if the data filled by the user is correct in real time. Further, applying LAFF-AD during the form-filling process can reduce the cost of fixing the detected anomalies [32] because it can provide a list of suggested values directly to the user. Also, anomaly detection during form filling can prevent error propagation, since if errors may affect subsequent decisions if they are not quickly fixed [44].

7.2 Practical Implications

This subsection discusses the practical implications of LAFF-AD for different stakeholders: Data quality engineers, researchers, and software developers.

7.2.1 Data quality engineers. LAFF-AD is a data anomaly detection approach that can be easily used by data quality engineers. To be able to run LAFF-AD, data engineers need one training set and one dataset with suspicious instances that need to be checked. LAFF-AD does not require data engineers to tune any parameter. Another advantage of LAFF-AD is that it can prevent providing data quality engineers with many misleading predictions by adding a “non-conclusive” label. This could be useful in practice, since engineers might be reluctant to use such a tool if it provides many inaccurate predictions.

7.2.2 Researchers. In this paper, we repurpose a form-filling recommender system to detect data anomalies in categorical fields. To the best of our knowledge, our approach is the first approach that uses the output and characteristics of an automated form-filling tool like LAFF for anomaly detection. We speculate that our proposed solution can inspire researchers to use other recommender systems to extend our approach, or to repurpose similar automated form-filling tools for similar tasks such as anomaly detection for numerical fields.

7.2.3 Software Developers. Thanks to its short prediction time, LAFF-AD can be used as a data quality check step during the data entry process. After filling the fields in a form or a page, the user can run LAFF-AD to check the filled value before submission. Since LAFF-AD is based on a form-filling recommender system, LAFF-AD can be integrated as a stand-alone tool to perform online anomaly detection. Similar to LAFF, deploying LAFF-AD needs only a mapping between columns in the dataset and the data entry form fields. This mapping can be found in software documentation such as the database schema and the description of the UI widgets in the data entry forms [7].

8 CONCLUSION

In this paper we have proposed LAFF-AD, an approach to automatically detect categorical data anomalies. LAFF-AD runs an adaptation of a form filling approach (LAFF) to predict the value of a suspicious categorical field in an input instance. LAFF-AD leverages the output of LAFF to detect data anomaly with a heuristic-based anomaly detection module. We evaluated LAFF-AD using six datasets with different characteristics. Experimental results show that LAFF-AD can accurately detect a high range of data anomalies, with recall values between 0.6 and 1 and a precision value of at least 0.808. Further, note that accuracy increases when more anomalous instances are present in a dataset. The results also show that LAFF-AD is fast enough to be applied to detect data anomalies in practice: LAFF-AD takes at most 7000 s and 735 ms to perform training and prediction, respectively.

As part of future work, we plan to conduct a study to analyze the impact of LAFF-AD on reducing the time of manual data anomaly detection by users. We also plan to investigate the possibility to deploy LAFF-AD during the form filling process as an online anomaly detection technique for categorical fields.

ACKNOWLEDGMENTS

Part of this work was financially supported by the Alphonse Weicker Foundation and by our industrial partner BGL BNP Paribas Luxembourg. We thank Anne Goujon, Clément Lefebvre Renard, and Andrey Boytsov for their feedback on LAFF-AD and earlier drafts of the paper. We also thank Chunfeng Yuan, for his help with the experiments. Lionel Briand was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), through the Canada Research Chairs and discovery programs, and the Science Foundation Ireland grant 13/RC/2094-2. This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant reference C22/IS/17373407/LOGODOR. For the purpose of open access, and in fulfillment of the obligations arising from the grant agreement, the authors have applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] Charu C Aggarwal and Charu C Aggarwal. 2017. *An introduction to outlier analysis.* Springer, Cham.

[2] Fabrizio Angiulli, Fabio Fassetti, Luigi Palopoli, and Cristina Serrao. 2022. A density estimation approach for detecting and explaining exceptional values in categorical data. *Applied Intelligence* 52 (2022), 1–23.

[3] Oluseun Omotola Aremu, David Hyland-Wood, and Peter Ross McAree. 2020. A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data. *Reliability Engineering & System Safety* 195 (2020), 106706.

[4] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy. 2014. Dimensions and metrics for evaluating recommendation systems. In *Recommendation Systems in Software Engineering*. Springer, Berlin, Heidelberg, Germany, 245–273.

[5] Tanya Barrett, Karen Clark, Robert Gevorgyan, Vyacheslav Gorelenkov, Eugene Gribov, Ilene Karsch-Mizrachi, Michael Kimelman, Kim D Pruitt, Sergei Resenchuk, Tatiana Tatusova, et al. 2012. BioProject and BioSample databases at NCBI: facilitating capture and organization of metadata. *Nucleic acids research* 40, D1 (2012), D57–D63.

[6] Stephen D Bay and Mark Schwabacher. 2003. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proc.KDD'03*. Association for Computing Machinery, New York, NY, USA, 29–38.

[7] Hichem Belgacem, Xiaochen Li, Domenico Bianculli, and Lionel Briand. 2023. A Machine Learning Approach for Automated Filling of Categorical Fields in Data Entry Forms. *ACM Trans. Softw. Eng. Methodol.* 32, 2 (apr 2023), 47:1–47:40.

[8] Hichem Belgacem, Xiaochen Li, Domenico Bianculli, and Lionel Briand. 2024. Learning-Based Relaxation of Completeness Requirements for Data Entry Forms. *ACM Transactions on Software Engineering and Methodology* 33, 3 (2024), 77:1–77:32.

[9] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition* 30, 7 (1997), 1145–1159.

[10] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proc.SIGMOD'00*. Association for Computing Machinery, New York, NY, USA, 93–104.

[11] Barbara Caputo, K Sim, Fredrik Furesjo, and Alex Smola. 2002. Appearance-based object recognition using SVMs: which kernel should I use?. In *Proc.NIPS'02*, Vol. 2002. N/A, N/A, 1 pages.

[12] Vendula Churová, Roman Vyškovskỳ, Kateřina Maršálová, David Kudláček, Daniel Schwarz, et al. 2021. Anomaly detection algorithm for real-world data and evidence in clinical research: implementation, evaluation, and validation study. *JMIR Medical Informatics* 9, 5 (2021), e27172.

[13] Kaustav Das and Jeff Schneider. 2007. Detecting anomalous records in categorical datasets. In *Proc.KDD'07*. Association for Computing Machinery, New York, NY, USA, 220–229.

[14] Stephen E Fienberg. 1979. The use of chi-squared statistics for categorical data problems. *Journal of the Royal Statistical Society: Series B (Methodological)* 41, 1 (1979), 54–64.

[15] Jim Gee and Mark Button. 2019. The financial cost of fraud 2019: The latest data from around the world. *N/A* N/A, 28 (2019), N/A pages.

[16] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.

[17] Douglas M Hawkins. 1980. *Identification of outliers*. Vol. 11. Springer, Dordrecht.

[18] Zengyou He, Shengchun Deng, and Xiaofei Xu. 2005. An optimization model for outlier detection in categorical data. In *Proc.ICIC'05*. Springer, Springer, Berlin, Heidelberg, 400–409.

[19] Zengyou He, Shengchun Deng, Xiaofei Xu, and Joshua Zhexue Huang. 2006. A fast greedy algorithm for outlier mining. In *Proc.PAKDD'06*. Springer, Springer, Berlin, Heidelberg, 567–576.

[20] Carrie Heeter. 2000. Interactivity in the context of designed experiences. *J. of Interactive Advertising* 1, 1 (2000), 3–14.

[21] Zhe Hui Hoo, Jane Candlish, and Dawn Teare. 2017. What is an ROC curve? , 357–359 pages.

[22] Dino Ienco, Ruggero G Pensa, and Rosa Meo. 2016. A semisupervised approach to the detection and characterization of outliers in categorical data. *IEEE transactions on neural networks and learning systems* 28, 5 (2016), 1017–1029.

[23] Faisal Jamil and Dohyeun Kim. 2021. An ensemble of prediction and learning mechanism for improving accuracy of anomaly detection in network intrusion environments. *Sustainability* 13, 18 (2021), 10057.

[24] LOX JELLY and Cream Cheese. 2009. Association between Two Categorical Variables: Contingency Analysis with Chi Square. *Business Statistics for Competitive Advantage with Excel 2007: Basics, Model Building, and Cases* N/A, NA (2009), 28 pages.

[25] Jalayer Khalilzadeh and Asli DA Tasci. 2017. Large sample size, significance level, and the effect size: Solutions to perils of using big data for academic research. *Tourism Management* 62 (2017), 89–96.

[26] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. 2008. Angle-based outlier detection in high-dimensional data. In *Proc.KDD'08*. Association for Computing Machinery, New York, NY, USA, 444–452.

[27] Peter Kromkowski, Shaoran Li, Wenxi Zhao, Brendan Abraham, Austin Osborne, and Donald E Brown. 2019. Evaluating statistical models for network traffic anomaly detection. In *Proc.SIEDS'19*. IEEE, IEEE, Charlottesville, VA, USA, 1–6.

[28] Junli Li, Jifu Zhang, Ning Pang, and Xiao Qin. 2018. Weighted outlier detection of high-dimensional categorical data using feature grouping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 11 (2018), 4295–4308.

[29] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 1 (2012), 1–39.

[30] Marcos Martínez-Romero, Martin J O'Connor, Attila L Egyedi, Debra Willrett, Josef Hardi, John Graybeal, and Mark A Musen. 2019. Using association rule mining and ontologies to generate metadata recommendations from multiple biomedical databases. *Database J. Biol. Databases Curation* 2019 (2019), 25 pages.

[31] Caitlin Mills, Nigel Bosch, Art Graesser, and Sidney D'Mello. 2014. To quit or not to quit: predicting future behavioral disengagement from reading patterns. In *Proc.ITS'14*. Springer, Springer, Cham, 19–28.

[32] Kıvanç Muşlu, Yuriy Brun, and Alexandra Meliou. 2015. Preventing data errors with continuous testing. In *Proc.ISSTA'15*. Association for Computing Machinery, New York, NY, USA, 373–384.

[33] Kazuyo Narita and Hiroyuki Kitagawa. 2008. Detecting outliers in categorical record databases based on attribute associations. *Lecture Notes in Computer Science* 4976 (2008), 111–123.

[34] Keith Noto, Carla Brodley, and Donna Slonim. 2012. FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. *Data mining and knowledge discovery* 25 (2012), 109–133.

[35] Guansong Pang, Longbing Cao, and Ling Chen. 2016. Outlier detection in complex categorical data by modeling the feature value couplings. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, New York, 2016 July 9-15* N/A, 6 (2016), 6 pages.

[36] Guansong Pang, Longbing Cao, and Ling Chen. 2021. Homophily outlier detection in non-IID categorical data. *Data Mining and Knowledge Discovery* 35, 4 (2021), 1–62.

[37] Guansong Pang, Kai Ming Ting, David Albrecht, and Huidong Jin. 2016. Zero++: Harnessing the power of zero appearances to detect anomalies in large-scale data sets. *Journal of Artificial Intelligence Research* 57 (2016), 593–620.

[38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[39] Tahereh Pourhabibi, Kok-Leong Ong, Booi H Kam, and Yee Ling Boo. 2020. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems* 133 (2020), 113303.

[40] Siyu Qian, Esther Munyisia, David Reid, David Hailey, Jade Pados, and Ping Yu. 2020. Trend in data errors after the implementation of an electronic medical record system: A longitudinal study in an Australian regional Drug and Alcohol Service. *International Journal of Medical Informatics* 144 (2020), 104292.

[41] NNR Ranga Suri, Narasimha Murty M, G Athithan, NNR Ranga Suri, Narasimha Murty M, and G Athithan. 2019. Outlier detection in categorical data. *Outlier Detection: Techniques and Applications: A Data Mining Perspective* N/A, N/A (2019), 69–93.

[42] Lida Rashidi, Sattar Hashemi, and Ali Hamzeh. 2011. Anomaly detection in categorical datasets using bayesian networks. In *Proc.AICI'11*. Springer Berlin Heidelberg, Springer, Berlin, Heidelberg, 610–619.

[43] Stuart J Russell. 2010. *Artificial intelligence a modern approach*. Pearson Education, Inc., N/A.

[44] David Saff and Michael D Ernst. 2003. Reducing wasted development time via continuous testing. In *Proc.ISSRE'03*. IEEE, IEEE, Denver, CO, USA, 281–292.

[45] Andrew Sears and Ying Zha. 2003. Data entry for mobile devices using soft keyboards: Understanding the effects of keyboard size and user tasks. *International Journal of Human-Computer Interaction* 16, 2 (2003), 163–184.

[46] NNR Ranga Suri, M Narasimha Murty, and Gopalasamy Athithan. 2012. An algorithm for mining outliers in categorical data through ranking. In *Proc.HIS'12*. IEEE, IEEE, Pune, India, 247–252.

[47] NNR Ranga Suri, Musti Narasimha Murty, and Gopalasamy Athithan. 2013. A rough clustering algorithm for mining outliers in categorical data. In *Proc.PReMI'13*. Springer, Springer, Berlin, Heidelberg, 170–175.

[48] Ayman Taha and Ali S Hadi. 2019. Anomaly detection methods for categorical data: A review. *ACM Computing Surveys (CSUR)* 52, 2 (2019), 1–35.

[49] Florian Tambon, Gabriel Laberge, Le An, Amin Nikanjam, Paulina Stevia Nouwou Mindom, Yann Pequignot, Foutse Khomh, Giulio Antoniol, Ettore Merlo, and François Laviolette. 2022. How to certify machine learning based safety-critical systems? A systematic literature review. *Automated Software Engineering* 29, 2 (2022), 38.

[50] Hongzuo Xu, Yongjun Wang, Li Cheng, Yijie Wang, and Xingkong Ma. 2018. Exploring a high-quality outlying feature value set for noise-resilient outlier detection in categorical data. In *Proc.CIKM'18*. Association for Computing Machinery, New York, NY, USA, 17–26.

[51] Hongzuo Xu, Yongjun Wang, Zhiyue Wu, and Yijie Wang. 2019. Embedding-based complex feature value coupling learning for detecting outliers in non-iid categorical data. In *Proc.AAAI'19*, Vol. 33. N/A, N/A, 5541–5548.

[52] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.