Systematic Construction of Anomaly Detection Benchmarks from Real Data

ANDREW EMMOTT, Oregon State University
SHUBHOMY DAS, Oregon State University
THOMAS DIETTERICH, Oregon State University
ALAN FERN, Oregon State University
WENG-KEEN WONG, Oregon State University

Research in anomaly detection suffers from a lack of realistic and publicly-available data sets. Because of this, most published experiments in anomaly detection validate their algorithms with application-specific case studies or benchmark datasets of the researchers' construction. This makes it difficult to compare different methods or to measure progress in the field. It also limits our ability to understand the factors that determine the performance of anomaly detection algorithms. This article proposes a new methodology for empirical analysis and evaluation of anomaly detection algorithms. It is based on generating thousands of benchmark datasets by transforming existing supervised learning benchmark datasets and manipulating properties relevant to anomaly detection. The paper identifies and validates four important dimensions: (a) point difficulty, (b) relative frequency of anomalies, (c) clusteredness of anomalies, and (d) relevance of features. We apply our generated datasets to analyze several leading anomaly detection algorithms. The evaluation verifies the importance of these dimensions and shows that, while some algorithms are clearly superior to others, anomaly detection accuracy is determined more by variation in the four dimensions than by the choice of algorithm.

Categories and Subject Descriptors: - [-] -

General Terms: Anomaly Detection, Experimental Design, Algorithm Comparison

Additional Key Words and Phrases: benchmarks, point difficulty, relative frequency, clusteredness, irrelevant features, swamping, masking, Isolation Forest, SVDD, one-class SVM, robust kernel density estimation, Gaussian mixture model

ACM Reference Format:
Andrew Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. 2015. Systematic Construction of Anomaly Detection Benchmarks from Real Data. ACM Trans. Knowl. Discov. Data. - , - , Article - (February 2015), 27 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Anomaly detection is an important inference task with applications across many different domains including identifying novel threats in computer security [Lane and Brodley 1997] [Portnoy et al. 2001] [Lazarevic et al. 2003] [Pokrajac et al. 2007], discovering novel astronomical phenomena [Wagstaff et al. 2013], detecting broken environmental sensors [Dereszynski and Dietterich 2011], identifying machine component failures...
[Xue et al. 2006; Zhang et al. 2008; Alzghoul and Lofstrand 2011], and finding cancer cells in normal tissue [Polat et al. 2005; Greensmith et al. 2006]. Despite the importance of the task, the field of statistical anomaly detection lacks a standard methodology for understanding and evaluating proposed algorithms. Most published experiments evaluate their algorithms via application-specific case studies or ad hoc synthetic datasets. There are very few realistic, publicly-available benchmark datasets. There are two consequences of this. First, it is very difficult to compare different algorithms to assess progress in the field. Second, it is difficult to understand the various factors or dimensions of anomaly detection problems that influence the performance of anomaly detection algorithms. This makes it difficult for experiments to guide research in algorithm development.

The goal of this paper is to develop and test a standardized evaluation methodology for statistical anomaly detection. Our approach is based on identifying four dimensions that characterize anomaly detection problems, creating synthetic—but realistic—datasets that vary these dimensions systematically, and demonstrating experimentally how these dimensions affect the performance of existing state-of-the-art anomaly detection algorithms.

This paper is organized as follows. We first review and assess existing approaches to the evaluation of anomaly detection methods. Based on this, we identify a set of requirements for a benchmarking methodology. Then we present our methodology and provide detailed procedures for constructing realistic benchmark datasets. Finally, we validate our methodology by applying it to evaluate several leading statistical anomaly detection methods. Our evaluation leads to three conclusions. First, the dimensions that we have identified and manipulated show a systematic impact on the performance of all of the anomaly detection algorithms. Second, some algorithms are clearly inferior to others. Third, the changes in performance that result from manipulating dimensions of the anomaly detection problem are significantly larger than the differences in performance among the algorithms. Hence, when confronting a new application, it is more important to understand where it lies along the proposed problem dimensions than it is to choose precisely the best possible anomaly detection algorithm.

1.1. The Task of Anomaly Detection

The anomaly detection setting that we study is the following. We are given a collection of \( N \) data points \( x_1, \ldots, x_N \), each a \( d \)-dimensional real-valued vector. These data points are a mixture of “normal” points and “anomalous” points. In all of the anomaly detection applications cited above, the anomalous points are generated by a process that is distinct from the process generating the normal points. Our goal is to identify those anomalous points. Natural metrics include precision (or \( 1 - \) the false alarm rate), recall (the fraction of anomalous points correctly identified), and composite measures such as the F1 measure and the area under the ROC curve. In this paper, we focus on AUC.

When confronted with an anomaly detection problem, there are two main approaches. One is to model both the normal and the anomalous points. This can succeed if we have a good understanding (or sufficient labeled training data) for both kinds of data points. However, it is in the nature of anomaly detection problems that we usually lack a good understanding of the process that is generating the anomalous points. In computer security [Zhang and Zulkernine; Denning 1987], the adversaries are constantly changing their attack strategies, a fact that invalidates supervised learning studies such as those based on the KDD 99 Challenge Cup [Elkan 1999]. In astronomy, the goal is to discover surprising and unexpected phenomena. For machine failure, there seems to be a Murphy’s Law that machines will always find new ways to
fail. Cancer is difficult to model, because it is not a single disease, but rather a whole constellation of different (and poorly-understood) mechanisms [Aggarwal et al. 2009].

Instead of modeling the processes generating the anomalies, the second approach is to detect the anomalies by relying on statistical signals. Specifically, we look for statistical outliers and hope that those outliers are anomalies. We will refer to this strategy as the “outliers-as-targets” approach. It was nicely-articulated by Hawkins [Hawkins 1980]:

“An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.”

From a methodological perspective, this means that when constructing benchmark datasets for anomaly detection, we should not just sample from a probability distribution with heavy tails. Instead, we should generate data by combining two different generating processes so that we can assess the relationship between being an outlier and being an anomaly.

1.2. Existing Experimental Methodology

In anomaly detection research, three kinds of data have been employed to analyze and evaluate anomaly detection algorithms. First, there are datasets drawn from specific application problems (e.g., [Xue et al. 2006; Lazarevic et al. 2003]). Second, there are synthetic datasets [Rocke and Woodruff 1996]. Third, there are datasets constructed by taking an existing supervised classification problem and treating one or more of the classes as the anomalies.

Application-specific datasets are very useful. They can help understand and evaluate the algorithm refinements needed to achieve high performance in a particular application. However, often the datasets are not publicly available because of privacy or security considerations (e.g., [Senator et al. 2013]).

Synthetic datasets [Liu et al. 2008; Liu et al. 2010; Rocke and Woodruff 1996] permit the systematic manipulation of some properties (e.g., the relative frequency of the anomalies, the distinctiveness of the anomalies). However, decades of experience in machine learning have shown that real data sets are much more complex and idiosyncratic than synthetic data, which undermines the validity of this approach [Mahoney and Chan 2003; McHugh 2000].

Finally, the repurposing of supervised classification data sets has the desirable property that the different classes are the result of different generating processes and the data retain the idiosyncracies of the real application (e.g., [Liu et al. 2008]). However, most studies have treated the datasets “as is” without trying to manipulate properties of the data. One exception is the work of Kim and Scott [Kim and Scott 2008] who subsampled the anomaly class to reduce the relative frequency of the anomalies. Another interesting case is the work of Das, et al., [Das et al. 2008], who generated anomalies by permuting features among a small subset of the data. In a few cases, supervised regression datasets have been repurposed by treating the data points with the most extreme values as anomalies [Cortez et al. 2009].

We propose to combine the idea of repurposing supervised learning datasets with the idea of systematically varying properties of the data. In this paper, we show how to take existing supervised learning datasets and manipulate the relative frequency of the anomalies, the degree of difficulty of individual anomaly points, the degree to which the anomalies are clustered (versus scattered), and the degree to which the features are relevant versus irrelevant for the task. Before describing our techniques, we first collect a list of the requirements that anomaly detection benchmarks should satisfy.
2. REQUIREMENTS FOR ANOMALY DETECTION BENCHMARKS

As discussed above, although anomaly detection algorithms work by searching for statistical outliers, the goal is to identify points that are generated by a process that is distinct from the process generating the “normal” points. This distinction leads to the first two requirements for benchmark datasets.

Requirement 1: Normal data points should be drawn from a real-world generating process.

Requirement 2: Anomalous data points should also be drawn from a real-world process, but one that is distinct from the process generating the normal points. The anomalous points should not just be points in the tails of the “normal” distribution. See, for example, Glasser and Lindauer’s synthetic anomaly generator (Glasser and Lindauer 2013).

Requirement 3: Many benchmark datasets are needed. If we employ only a small number of datasets, we risk developing algorithms that only work on those problems. Hence, we need a large (and continually expanding) set of benchmark datasets to ensure generality and prevent methodological overfitting by the research community.

Requirement 4: Benchmark datasets should be characterized in terms of well-defined and meaningful problem dimensions that can be systematically varied. An important goal for benchmarking is to gain insight into the strengths and weaknesses of the various algorithms. Ideally, we should identify those dimensions along which anomaly detection problems might vary and then generate benchmark datasets that vary these dimensions in a controlled fashion.

There is currently no established set of problem dimensions for anomaly detection. We have identified four dimensions that we believe are important.

Point difficulty measures the similarity of an anomalous data point to the normal data points. The outliers-as-targets approach breaks down as the target points become harder to distinguish from the normal points. This happens when the targets are not confined to extreme outliers, or when the extreme outliers are not anomalies. In either case, the anomalies of interest will be confused with normal points, or with uninteresting outliers.

One aspect of applying anomaly detection in adversarial settings (e.g., intrusion detection or insider threat detection) is that adversaries try to blend in with the distribution of normal points, so we can expect these applications to experience a higher point difficulty. This phenomenon has also been referred to as “swamping” (Liu et al. 2008).

We propose to define point difficulty based on an oracle that knows the true generating processes underlying the “normal” and “anomalous” points. Using this knowledge, the oracle can estimate the probability \( P(y = \text{normal} | x) \) that a data point \( x \) was generated by the “normal” distribution. The larger this value is for an anomalous point \( x \), the more difficult it will be for an anomaly detection algorithm to discover that \( x \) is anomalous.

Semantic Variation is a measure of the degree to which the anomalies are generated by more than one underlying process. A common aspect of many anomaly detection applications is that there can be multiple processes generating anomalies. In a cybersecurity setting, there can be many different kinds of attacks and many different methods for stealing information. In cancer detection, there can be many different biological processes that result in cancerous cells.

We would like to manipulate this dimension in our benchmarks. However, this is difficult. We propose instead to employ a measure of clusteredness as a proxy for semantic variation. If our anomalies are highly clustered, this is suggests that only a single pro-
cess is generating them, whereas if they are highly scattered, this could correspond to multiple generating processes. This is particularly true for benchmark datasets constructed from multiclass problems, where the anomaly points can be drawn from multiple classes.

When anomaly points are tightly clustered, this creates a region of high probability density, which can defeat density estimation-based methods. This phenomenon has also been called “masking” [Liu et al. 2008].

Relative frequency is the fraction of the incoming data points that are anomalies of interest. This is a fundamental issue in anomaly detection: How much can the data be contaminated by anomalies before the anomalies can no longer be reliably detected? This dimension can be easily manipulated. Relative frequency has also been called “plurality” and “contamination rate”.

The behavior of anomaly detection algorithms often changes with the relative frequency. If anomalies are rare, then methods that pretend that all training points are “normal” and fit a model to them may do well. If anomalies are common, then methods that attempt to fit a model of the anomalies may do well. In most experiments in the literature, the anomalies have a relative frequency between 0.01 and 0.1, but some go as high as 0.3 [Kim and Scott 2008; Liu et al. 2008]. Many security applications are estimated to have relative frequencies in the range of $10^{-5}$ or $10^{-6}$.

Feature Relevance/Irrelevance. It is well-established that irrelevant features can degrade the performance of supervised learning methods, and we now have many good algorithms for identifying and removing irrelevant features. We believe that irrelevant features are an even greater problem for anomaly detection. From the statistical perspective, each irrelevant feature increases the dimensionality of the space, and the sample size required by (naive) density estimation methods tends to scale exponentially with the dimension. In addition, as the dimensionality of the data increases, the “surface area” of the volume containing the data also increases, which is a geometric way of saying that there are more “tails” in which the data may lie. This increases the risk that normal points will fall in the tails of the distribution.

From the application perspective, there is a natural tendency to include any feature that could conceivably be informative. But this increases the risk of including irrelevant features. Note that because there are no labels to define the anomaly-detection task, the choice of features is the only way that the user defines the task to an anomaly detection algorithm.

We propose that benchmark datasets should systematically vary the set of features to manipulate both the power of the relevant features and the number of irrelevant or “noise” features.

3. METHODOLOGY

We have developed a methodology that achieves most of the requirements listed above. To satisfy the first three requirements, we develop 24,800 benchmark datasets by transforming 19 datasets chosen from the UC Irvine repository [Bache and Lichman 2013]. For each dataset, we separate its data (e.g., the classes of a classification problem) into two sets: “normal” and “anomalous”. This ensures that these data points are generated by distinct real-world processes rather than from synthesized distributions (Requirement 1). To satisfy Requirement 4, we select a subset of the “anomalous” points according to various criteria and augment the data set with irrelevant features. Our methodology can be viewed as defining an algorithm, so it can be applied to new datasets as they become available.
3.1. Selecting Datasets

To ensure objectivity in the construction of the benchmarks, we only worked with datasets from the UCI data repository [Bache and Lichman 2013]. We selected all UCI datasets (as of the beginning of this study) that matched the following criteria:

— **task**: Classification (binary or multi-class) or regression. No time series.
— **instances**: At least 1000. No upper limit.
— **features**: No more than 200. No lower limit.
— **values**: Numeric only. Categorical features are ignored if present. No missing values, with one exception (see below).

The choice of these criteria was not guided by the performance of any anomaly detection algorithms.

Our criteria do not cover all settings in which anomaly detection is appropriate. Instead, we focused on the common case: high-dimensional, continuous-valued, independent and identically distributed (iid) data. Future work should explore nominal and ordinal features [Otey et al. 2006] as well as more structured (non-idd) settings such as time series [Huang et al. 2013; Mei and Gul 2013] and network data (e.g., [Bridges et al. 2014]).

These criteria yielded a collection of 19 datasets which we refer to as the “mother” sets, since they will produce thousands of “child” benchmark datasets. The 19 selected mother sets are the following:

— **binary classification**: MAGIC Gamma Telescope, MiniBooNE Particle Identification, Skin Segmentation, Spambase
— **multi-class classification**: Steel Plates Faults, Gas Sensor Array Drift, Image Segmentation, Landsat Satellite, Letter Recognition, Optical Recognition of Handwritten Digits, Page Blocks, Shuttle, Waveform, Yeast
— **regression**: Abalone, Communities and Crime, Concrete Compressive Strength, Wine, Year Prediction

Communities and Crime is the one exception to our rule against missing values. In this case, there were some features for which the values where missing for the majority of points, so we simply removed these features from the data set.

In each of these mother sets, each feature was normalized to have zero mean and unit sample variance.

3.2. Defining Normal versus Anomalous Data Points

A central goal of our methodology is that the “normal” and “anomalous” points should be produced by semantically distinct processes (Requirement 2). To achieve this, we relabel each point in the 19 mother sets as either a candidate normal or candidate anomaly, informed by the semantics of the original set. We employed the following methods.

**3.2.1. Binary Classification Problems.** For datasets that were already binary classification problems, the data is already partitioned into two semantically-distinct groups. We chose one class as “candidate normal” (i.e., the set from which we will select the “normal” points) and the other as “candidate anomaly” (i.e., the set from which we will select the “anomaly” points). The class with fewer instances is chosen to be the candidate anomaly class. We do this because the final benchmarks will subsample the candidate anomalies so that the anomalies constitute a very small fraction of all of the data points. The larger the majority class, the easier this is to achieve. In the case that both classes are of equal size, the class with greater variance is defined to be the candidate anomaly class. We do this because we also wish to control our benchmarks along
the axis of scattered/clustered anomaly points, and a greater initial variance among candidate anomalies allows for greater flexibility along this factor.

While binary classification problems are easily partitioned into two candidate sets, they come with some disadvantages. First, there is some risk that the candidate anomalies will have low semantic variation, since they all belong to a single class. Second, there is no guarantee about how well point difficulty can be controlled in this setting. If the original classification problem can be easily solved with high confidence, then benchmarks created from this dataset will exhibit low point difficulty.

3.2.2. Regression Problems. For mother sets that are regression or multi-class problems, our approach is to transform them into binary classification problems and then treat them as described above. For regression datasets, this transformation is simple. We compute the median of the regression response and partition the data into two classes by thresholding on this value. To the extent that low versus high values of the response correspond to different generative processes, this will create a semantic distinction between the candidate normal and candidate anomalous data points. We expect candidate anomaly points near the median will have high point difficulty and candidate anomalies near the extremes will have a low point difficulty. Benchmarks derived from regression problem sets allow for flexible (and easy) control of point difficulty. Further, one can expect that response values near the median might be more tightly clustered while response values near the tails might be more varied, but there is no guarantee that this is true.

3.2.3. Multi-class Problems. For multi-class datasets, we partitioned the available classes into two sets with the goal of maximizing the difficulty of telling them apart. For mother sets with many classes, it can be impractical to try every partition of the classes in the search of the most confusing binary problem, so we employ the following approximation.

Our heuristic procedure begins by training a Random Forest [Breiman 2001] to solve the multi-class classification problem. Then we calculate the amount of confusion between each pair of classes. For each data point \( x_i \), the Random Forest computes an estimate of \( P(\hat{y}_i|x_i) \), the predicted probability that \( x_i \) belongs to class \( \hat{y}_i \). We construct a confusion matrix \( C \) in which cell \( C[j,k] \) contains the sum of \( P(\hat{y}_i=k|x_i) \) for all \( x_i \) whose true class \( y_i = j \). We then define a graph in which each node is a class and each edge (between two classes \( j \) and \( k \)) has a weight equal to \( C[j,k] + C[k,j] \). This is the (unnormalized) probability that a data point in class \( j \) will be confused with a data point in class \( k \) or vice versa. We then compute the maximum weight spanning tree of this (complete) graph to identify a graph of “most-confusable” relationships between pairs of classes. We then two-color this tree so that no adjacent nodes have the same color. The two colors define the two classes of points.

This approximately maximizes the confusion between the candidate normal and candidate anomaly data points and also tends to make both classes diverse, which increases semantic variation in both sets.

Because the candidate anomalies are drawn from many classes, we expect benchmarks constructed from multi-class problems to allow greater flexibility in choosing highly scattered or highly clustered anomalies. However, despite our efforts to maximize the difficulty of the resulting binary problem, flexibility in point difficulty is still subject to the nature of the original problem; there is no guarantee that any partitioning of the classes will prove difficult to distinguish.

3.3. Controlling Problem Dimensions to Generate Benchmark Datasets
The next step in our methodology is to generate benchmark datasets from each mother set by manipulating the four dimensions of point difficulty (pd), relative frequency (rf),
clusteredness (cl), and feature irrelevance (fi). In this section, we define a continuous measure for each of these dimensions. We then define a set of factor levels corresponding to discretizing these continuous measures into bins. Each benchmark dataset corresponds to choosing one level of each factor. For each mother set, we iterate over each combination of problem dimension levels and construct up to 10 benchmark datasets having those settings. As we will describe below, limitations of some mother datasets mean that we cannot always achieve 10 datasets for every desired setting.

We also subsample our candidate normal points to ensure some variability between the benchmarks even when the problem dimension settings are the same. Except where otherwise stated below, each benchmark uses a 90% sample of the candidate normals (drawn without replacement).

In total, from the 19 mother sets listed earlier, this methodology produced 24,800 problem set replicates, all of which we employed to test several statistical outlier detection algorithms.

3.3.1. Point Difficulty. To quantify the difficulty of detecting each candidate anomaly point, we applied Kernel Logistic Regression (KLR; [Jaakkola and Haussler 1999; Zhu and Hastie 2001; Keerthi et al. 2005]) to fit an “oracle” to discriminate the candidate anomalies (labeled $y = 0$) from the candidate normals (labeled $y = 1$). Specifically, we implemented the algorithm described by Keerthi, et al., [Keerthi et al. 2005] with parameters chosen via 5-fold cross-validation. We then computed the predicted probability $P(y = 1|x)$ for each candidate anomaly point $x$. Candidate anomaly points that are easy to discern from the candidate normals will have responses near 0, while points that KLR confuses with the normal class will have responses above 0.5. Hence, for candidate anomalies, the fitted response provides a good quantitative measure of point difficulty.

We binned this measure into four discrete levels:

- **pd-0**: difficulty score $\in (0, 0.15)$
- **pd-1**: difficulty score $\in (0.15, 0.3)$
- **pd-2**: difficulty score $\in (0.3, 0.5)$
- **pd-3**: difficulty score $\in (0.5, 1)$

When creating benchmarks at a particular difficulty setting, we only consider candidate anomaly points from the corresponding bin.

Although we doubt that datasets derived from pd-3 candidate anomalies will resemble any real application domain (as these would be anomalies that are confused with the normal class even in the supervised setting), we decided to include them in our tests to see what impact they have on the results.

A practical note: Some of the original mother sets are very large and performing KLR on the entire set was very time consuming. For sets larger than 12,000 points, we partitioned the mother set into approximately equal subsets of size smaller than 12,000 and applied KLR separately to each partition. Because the difficulty score of a candidate anomaly is influenced by which partition it comes from, we ensured that each benchmark set we created was composed entirely of points (both normal and anomalous) that came from the same partition.

3.3.2. Relative Frequency. This setting is very easily controlled; if we desire that a benchmark is 0.01 contaminated with anomalies, then we simply ensure that the benchmark draws 0.99 of its points from the candidate normals and 0.01 of its points from the candidate anomalies. We defined the following five relative frequency (rf) levels:

- **rf-0**: 0.001 of the benchmark is drawn from the candidate anomalies.
— rf-1: 0.005 of the benchmark is drawn from the candidate anomalies.
— rf-2: 0.01 of the benchmark is drawn from the candidate anomalies.
— rf-3: 0.05 of the benchmark is drawn from the candidate anomalies.
— rf-4: 0.1 of the benchmark is drawn from the candidate anomalies.

On occasion, a particular configuration of levels for a mother set would not yield enough candidate anomalies to reach rf-3 or rf-4 naturally, so the candidate normals were subsampled until the target fraction was met. Similarly, some mother sets did not have very many candidate normal points, so achieving rf-0 required that the candidate normals were resampled until they could outnumber the anomalies such that the anomalies made up only 0.001 of the benchmark dataset. Otherwise, 0.9 of the candidate normals are used, as stated earlier.

We believe that rf-4 is not a realistic setting (though this is debatable), but as with point difficulty we wanted to explore extreme settings to understand their impact on anomaly detection performance.

The choice of relative frequency establishes $k_a$, the number of candidate anomalies, and $k_n$, the number of candidate normals that we need to select to create the desired benchmark dataset.

3.3.3. Semantic Variation and Clusteredness. Suppose we have selected a set of $k_a$ anomalous points and $k_n$ normal points. The normalized clusteredness of this set of points is defined as

$$
\log \left( \frac{\hat{\sigma}^2_n}{\hat{\sigma}^2_a} \right)
$$

where $\hat{\sigma}^2_n$ is the sample variance of the selected normal points and $\hat{\sigma}^2_a$ is the sample variance of the selected anomaly points. When normalized clusteredness is less than 0, the anomaly points exhibit greater semantic variation than the normal points. When normalized clusteredness is greater than 0, the anomaly points are more tightly packed than the normal points (on average).

Given a set of candidate anomaly points, we applied the following algorithms to generate sets of size $k_a$ that are either widely dispersed or tightly clustered (as measured by Euclidean distance). To generate $k_a$ dispersed points, we applied a facility location algorithm [Gonzalez 1985] to choose $k_a$ points as the locations of the facilities. To generate $k_a$ tightly clustered points, we chose a seed point at random and then computed the $k_a - 1$ candidate anomaly points that are closest to it in Euclidean distance.

Note that during this process, the point difficulty is constrained, so only candidate anomalies having the specified point difficulty level are considered.

For purposes of analysis, we grouped the clusteredness (cl) scores into seven discrete levels:

— cl-0: for benchmarks that select only 1 anomaly point.
— cl-1: normalized clusteredness $\in (-\infty, -1.5)$
— cl-2: normalized clusteredness $\in [-1.5, -0.75)$
— cl-3: normalized clusteredness $\in [-0.75, 0)$
— cl-4: normalized clusteredness $\in [0, 0.75)$
— cl-5: normalized clusteredness $\in [0.75, 1.5)$
— cl-6: normalized clusteredness $\in [1.5, \infty)$

We place the benchmarks with solitary anomalies in their own bin, because the normalized clusteredness score derived from such benchmarks is undefined. Mathematically, a solitary anomaly has zero variance and thus infinite clusteredness, but the purpose of a clusteredness score is to indicate the possibility of densely packed anomalies defeating density- or distance-based algorithms. A solitary anomaly poses no such
threat. Hence, we interpret solitary anomalies as having a normalized clusteredness of $-\infty$.

Unlike the other dimensions, we do not have perfect control over how many benchmarks we can create at each level. Instead, we ran the algorithm for choosing scattered anomalies, measured the normalized clusteredness, and assigned it to the appropriate bin. This provided data points for one benchmark set. Then we ran the algorithm for choosing clustered anomalies, measured the normalized clusteredness, and assigned it to the appropriate bin. This provided a second benchmark set. The bins were defined to contain roughly equal numbers of benchmark datasets.

### 3.4. Feature Irrelevance

In the terms discussed earlier, it is unknown to us how relevant the features of the mother set are to the task of detecting targeted outliers, so we assume that the original data offers the most compact set of features for this purpose. To control our measure of feature irrelevance, we added irrelevant features to an otherwise finished benchmark until a desired level of feature irrelevance is introduced.

We quantify the amount of feature irrelevance as follows. First, we measure the average distance between all pairs of points in the benchmark dataset prior to adding irrelevant features. Then we add irrelevant features until the average pairwise distance has increased by a desired ratio. We define four levels of feature irrelevance ($f_i$):

- $f_i-0$: Average distance ratio of 1.0 (no added irrelevant features).
- $f_i-1$: Average distance ratio of 1.2
- $f_i-2$: Average distance ratio of 1.5
- $f_i-3$: Average distance ratio of 2.0

To create a new irrelevant feature, we choose a feature from the original mother set uniformly at random. Then for each data point in the benchmark dataset, we choose a value for this feature by sampling uniformly (with replacement) from the values of the original data points. The result is that this new feature has the same marginal distribution as the original feature, but its values carry no information about the anomaly status of the data points. This preserves the idiosyncracies of the real data.

To simplify the process of determining how many irrelevant features are needed, we compute an estimate of how many extra features will achieve a desired average distance ratio. Note that the expected distance between two vectors ($\alpha$) whose coordinates are drawn at random (e.g., from the unit interval or from a standard normal Gaussian) grows in proportion to $\sqrt{d}$, where $d$ is the dimensionality of the data. Hence, if a dataset already has $d$ dimensions and we want to estimate $d'$, the number of dimensions needed to increase the average pairwise distance by a factor of $\alpha$, then we need

$$d' = (\alpha \sqrt{d})^2$$

dimensions, where $\alpha \in \{1.0, 1.2, 1.5, 2.0\}$ for this study.

### 3.5. Statistics for the Generated Benchmarks

In this section, we describe to what extent we were able to successfully manipulate all four dimensions for each of the mother sets.

#### 3.5.1. Point Difficulty

Of the 19 mother sets, 12 had sufficient candidate anomalies at all four difficulty levels. The following datasets did not:

- **Skin** produced no pd-1 difficulty points (but had points with pd-0, pd-2, and pd-3).
- **Concrete Compressive Strength** produced no pd-3 difficulty points.
Table I summarizes how many of the candidate anomalies were assigned to each point difficulty bin.

| pd    | Total | Binary | Multiclass | Regression |
|-------|-------|--------|------------|------------|
| pd-1  | 0.37  | 0.86   | 0.66       | 0.14       |
| pd-2  | 0.29  | 0.10   | 0.10       | 0.39       |
| pd-3  | 0.21  | 0.03   | 0.03       | 0.31       |
| pd-4  | 0.12  | 0.01   | 0.21       | 0.16       |

Note that 12% of all candidate anomalies were assigned to pd-4. Recall that these are points that are confused with normal points even in a supervised setting. When employing these benchmarks, one may not wish to use our coarsely-defined levels, but ignoring point difficulty entirely may be unwise, as a good number of candidate anomalies will exhibit a degree of “normalness” that may be undesirable.

Second, binary and multiclass problems have the majority of their candidate anomalies at pd-1, while regression problems have more candidates to choose from at pd-2 and pd-3 (while still offering many points at pd-1). This matches our intuition about regression mothersets and suggests that regressions problems offer the greatest flexibility in controlling point difficulty in benchmark datasets.

3.5.2. Normalized Clusteredness. In 7272 benchmark datasets, our methods for manipulating clusteredness fail entirely, because after selecting points based on difficulty level, there was no way to vary the clusteredness. These were datasets with only a single candidate anomaly or datasets where all available points (at the target difficulty level) have been chosen.

For the remaining 17528 benchmark datasets, we evaluated the capability of the facility location algorithm to induce low clusteredness and the ability of the nearest neighbor algorithm to induce high clusteredness. We consider the facility location algorithm to be a success if the normalized clusteredness measure (nc) is less than 0, and we considered the nearest neighbor algorithm to be a success if the nc measure was greater than 0. Table II summarizes the results.

| Origin | Algorithm     | Success Rate | Mean nc | Standard Deviation |
|--------|---------------|---------------|---------|--------------------|
| All    | Facility Location | 0.64           | -0.23   | 0.68               |
|        | Nearest Neighbor | 0.95           | 1.01    | 0.86               |
| Binary | Facility Location | 0.43           | 0.05    | 0.89               |
|        | Nearest Neighbor | 0.95           | 1.01    | 1.02               |
| Multiclass | Facility Location | 0.66           | -0.28   | 0.68               |
|         | Nearest Neighbor | 0.94           | 1.06    | 0.87               |
| Regression | Facility Location | 0.75           | -0.32   | 0.45               |
|         | Nearest Neighbor | 0.95           | 0.74    | 0.63               |

In general we found it very easy to create benchmarks with tightly clustered anomalies; this should be no surprise, especially from binary and multiclass mothersets where the anomalies are usually drawn from well-populated classes. We found it more difficult to create benchmarks with scattered anomalies, especially when drawing them from binary mothersets, which validates our intuitions about binary mothersets exhibiting less semantic variation among their candidate anomalies. We should also
note, however, that our own requirement to only draw anomalies from a specific point
difficulty range is likely also contributing to this difficulty in creating benchmarks with
scattered anomalies.

Regression mothersets exhibit the greatest flexibility in controlling clusteredness,
but they also enjoyed the most even distribution of points across point difficulty lev-
eels, so whether they are truly more flexible in general or just less inhibited by the
constraints of our methodology is unclear.

Unlike point difficulty and relative frequency, we don’t consider any setting of clus-
teredness to be unrealistic. However we do note that application domains will typi-
cally have an expectation as to how distinct or clustered their target anomalies will
be. When creating benchmarks to match an application domain, if scattered anomalies
are desired, using a binary motherset or constraining point difficulty may stymie this
effort.

3.5.3. Feature Irrelevance. Recall that we seek to create datasets with target levels
of feature irrelevance as specified by target levels of the pairwise distance ratio:
$\alpha \in \{1, 1.2, 1.5, 2\}$. Table III shows that we achieved these target levels very well, al-
though there was some variability at the highest level. The Absolute Error is the abso-
late value of the difference between the achieved $\alpha$ and the target $\alpha$. The $R^2$ statisti-
c reports the fraction of variance in the achieved $\alpha$ that can be explained by using our
target $\alpha$ as the estimator. Of course the baseline value of $\alpha = 1$ is achieved perfectly
for all benchmark datasets by not adding any irrelevant features.

| Target $\alpha$ | Mean Achieved $\alpha$ | Mean Absolute Error | $R^2$ |
|----------------|------------------------|---------------------|-------|
| 1.2            | 1.21                   | 0.04                | 0.98  |
| 1.5            | 1.54                   | 0.09                | 0.91  |
| 2.0            | 2.10                   | 0.16                | 0.85  |

4. ANOMALY DETECTION ALGORITHMS

To simultaneously assess the effectiveness of our methodology and compare the per-
formance of various statistical anomaly detection algorithms, we conducted an experi-
mental study with several well-known anomaly detection algorithms. In this section,
we describe each of those algorithms. For algorithms that required parameter tuning,
we employed cross-validation (where possible) to find parameter values to maximize
an appropriate figure of merit (as described below). In all cases, we made a good faith
effort to maximize the performance of all of the methods. Some conservative parame-
terization choices had to be made to ensure that the given algorithm implementation
would return valid results across all problems.

4.1. One-Class SVM (ocsvm)

The One-Class SVM algorithm (Scholkopf et al. [Scholkopf et al. 1999]) shifts the data
away from the origin and then searches for a kernel-space decision boundary that sep-
arates fraction $1 - \delta$ of the data from the origin. We employed the libsvm implemen-
tation of Chang and Lin [Chang and Lin. 2011] available at
http://www.csie.ntu.edu.tw/~cjlin/libsvm/. For each benchmark, we employed a Gaussian radial basis function
kernel and we searched parameter space until approximately 5% ($\delta = 0.05$) of the data
fell outside the decision boundary in cross-validation. We would have preferred to use
smaller values for $\delta$, but OCSVM would not execute reliably for smaller values. The
distance of a point from the decision boundary determines the anomaly score of that
point.
4.2. Support Vector Data Description (svdd)

As proposed by Tax and Duin (Tax and Duin 2004), Support Vector Data Description finds the smallest hypersphere (in kernel space) that encloses $1 - \delta$ of the data. We employed the libsvm implementation available at [http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/] with a Gaussian radial basis function kernel. We searched for parameter values such that approximately 1% ($\delta = 0.01$) of the data fell outside the decision surface in cross validation. The distance of a point from the decision surface determines the anomaly score of that point.

4.3. Local Outlier Factor (lof)

The well-known Local Outlier Factor algorithm (Breunig et al. (Breunig et al. 2000)) computes the outlier score of a point $x$ by computing its average distance to its $k$ nearest neighbors. It normalizes this distance by computing the average distance of each of those neighbors to their $k$ nearest neighbors. So, roughly speaking, a point is declared to be anomalous if it is significantly farther from its neighbors than they are from each other. We employed the R package Rlof available at [http://cran.open-source-solution.org/web/packages/Rlof/].

We chose $k$ to be 3% of the dataset. This was the smallest value for which LOF would reliably run on all datasets.

4.4. Ensemble Gaussian Mixture Model (egmm)

A classic approach to anomaly detection is to fit a probabilistic model to the available data to estimate the density $P(x)$ of each data point $x$. Data points of low density are declared to be anomalies. One approach to density estimation is to fit a Gaussian mixture model (GMM) using the EM algorithm. To reduce the computational cost of fitting, and to improve the numerical stability of the process, we first transformed each benchmark dataset via principle component analysis. We selected principle components (in descending eigenvalue order) to retain 95% of the variance.

A single GMM is not very robust, and it requires that we select the value of $k$. To improve robustness, we computed an ensemble of GMMs and then combined their predicted densities. To generate the members of the ensemble, we varied the number of clusters $k$ by trying all values in {4, 5, 6, 7, 8, 9, 10}. For each value of $k$, we generated 15 GMMs by training on 15 bootstrap replicates of the data and by randomly initializing each replicate. We then computed the average out-of-bag log likelihood for each value of $k$ and discarded $k$ values whose average log likelihood was less than 0.85 times the average log likelihood of the best value of $k$. The purpose of this was to discard GMMs that do not fit the data very well. Finally, an anomaly score is computed for each point $x$ by computing the average "surprise", which is the average negative log probability density $\frac{1}{L} \sum_{\ell=1}^{L} - \log P_{\ell}(x)$, where $L$ is the number of fitted GMMs and $P_{\ell}(x)$ is the density assigned by GMM $\ell$ to data point $x$. We found in preliminary experiments that this worked better than using the mean probability density $\frac{1}{L} \sum_{\ell=1}^{L} P_{\ell}(x)$.

4.5. Robust Kernel Density Estimation (rkde)

Another approach to fitting a flexible probability density model is to fit distributions with simple parameters to each point in the data and then build an additive model that combines them. Such methods are known to be sensitive to outliers and so more complicated methods from robust statistics have been introduced. We implemented the approach described by Kim, et al. (Kim and Scott 2008). We employed a Gaussian radial basis kernel, and we optimized over a Hampel loss function with the additional parameters set as suggested by the authors.
4.6. Isolation Forest (iforest) and Split-selection Criterion Isolation Forest (scif)

The Isolation Forest algorithm (Liu, et al. [Liu et al. 2008]) ranks data points as anomalous if they are easily isolated by random axis-parallel splits. An isolation forest is a set of isolation trees. Each isolation tree is an extremely random decision tree trained on a small subsample of the data and selects its splits by choosing a feature at random and then choosing a splitting threshold uniformly at random between the minimum and maximum observed values of that feature. Given a data set, an isolation tree is grown until each data point is in its own leaf—that is, it is totally isolated from all other data points. For tree \( \ell \), let \( d(x, \ell) \) be the depth of the leaf containing data point \( x \). This is the isolation depth, and the anomaly score is computed from the average isolation depth across an ensemble of 100 isolation trees. The intuition is that outliers are points that are easily isolated at random from the remaining data, so points with small average isolation depth are likely to be anomalies and so are assigned high anomaly scores. For our study we found a subsample size of 2048 (or the entire dataset when a benchmark is smaller than this) to work best.

The Isolation Forest has a known weakness. When anomalous points are tightly clustered, their isolation depth grows, which lowers their anomaly score. To address this weakness, Liu, et al. [Liu et al. 2010] developed the Sparse-selection Criterion Isolation Forest, which employs random projection splits instead of axis-parallel splits in each internal node. An implementation of Isolation Forest was obtained from http://sourceforge.net/projects/iforest/. We wrote our own implementation of SCI Isolation Forest, and used a subsample size of 1024 (or the entire dataset when a benchmark is smaller than this).

5. RESULTS

Running each algorithm on each benchmark produced 173,600 micro-experiments. To summarize the results of all of these experiments, we first compute the AUC (area under the ROC curve) achieved in each micro-experiment. This estimates the probability that the algorithm correctly ranks anomalous points above normal points. We then analyze these AUCs via an analysis of deviance, which provides a way of understanding the effect of each of our four dimensions (as well as the effect of each mother set and each algorithm) on anomaly detection performance.

An analysis of deviance is similar to an analysis of variance, but it is appropriate when the response variable is a Bernoulli random variable or a Bernoulli parameter. It begins by fitting the model

\[
\logit(AUC) \sim set + algo + pd + rf + cl + fi
\]  

using the robust linear model (rlm) found in the R package MASS. In this model, \( set \) is the mother dataset, \( algo \) is the algorithm, \( pd \) is the point difficulty level, \( rf \) is the relative frequency of anomalies, \( cl \) is the clusteredness, and \( fi \) is the feature irrelevance level. The \( pd, rf, cl \) and \( fi \) values are the binned levels as defined above. We used a robust linear model because we believe a portion of our benchmarks to be pathologically too easy or too hard; this is by design for this study. Our model optimized a Hampel loss function.

Because the AUC values are probability estimates, we apply the logit transform \( \logit[AUC]/(1 - AUC) \). This maps AUC scores onto the real line, where we can expect a linear model to give a reasonable fit. Occasionally, the maximum likelihood estimate of the AUC on a particular benchmark dataset may be 1.0 or 0.0, which are highly unlikely to be correct and which the logit transform will map to \( +\infty \) and \( -\infty \). To prevent this problem, we regularize the AUC estimates by assuming a uniform Beta prior distribution on the AUC (much like the Laplace correction for estimating a Bernoulli
parameter). This is equivalent to placing a prior “normal” point ranked above all other points and a prior “anomalous” point ranked below all other points.

When a regression model is fit to inputs that are discrete factors, the data are actually modeled in terms of *contrasts*. That is, one level of the factor is treated as the baseline and its impact on the predicted AUC is absorbed into the intercept term $\beta_0$ of the linear model. Then each of the remaining levels of the factor is represented by an indicator variable $I_{jv}$, which is 1 if the $j$th feature of $x$ is equal to the value $v$ and whose fitted regression coefficient is $\beta_{jv}$. The value of $\beta_{jv}$ tells us how much the logit(AUC) will change if the value of the $j$th feature is changed from its baseline value to $v$. The value of $\beta_0$ is the log odds of the AUC averaged over all benchmark datasets for the baseline configuration. In our analysis, the baseline configuration was set = abalone, alg = egmm, pd = pd-1, rf = rf-1, cl = cl-0, fi = fi-1.

To measure statistical significance, we performed an omnibus F-Test on each factor (including motherset) where the null hypothesis is that each level of a factor produces the same result. Each test yielded a very small p-value, with the largest being less than $2.87 \times 10^{-5}$, so we safely conclude that our factors demonstrate an effect on the outcome of our experiments.

We now examine the $\beta$ values corresponding to each of these factors in order to understand the impact of each factor on AUC when controlling for all of the other factors.

### 5.1. Mother Dataset

Figure 1 shows the fitted coefficients across all original motherset relative to *Abalone*. The variation is quite large. Anomalies were hardest to detect in benchmarks constructed from *Gas Sensor Array Drift* and easiest from *Page Blocks*. Just as in supervised learning, some datasets are much more difficult than others.
5.2. Anomaly Detection Algorithm

Figure 2 shows the contribution of each algorithm to the logit(AUC) relative to EGMM. This provides an assessment of the performance of the algorithms while controlling for the effect of all of the other variables. We see that the classic LOF is the best performer, but iForest is also excellent. We were also pleased with the performance of the flexible density estimation methods RKDE and EGMM. The most surprising result is the very poor performance of OCSVM. Understanding this is a topic for future research. Based on these results, we would recommend against using OCSVM, at least in the way that we applied it where we trained it to contain 95% of the data. Another surprise was that SCIF, which was designed to be an improvement on iForest, performed much worse. Overall, we would recommend iForest and LOF based on these benchmarks, noting that iForest has a much smaller time complexity than LOF.

![Fig. 2. Change in logit(AUC) by Algorithm](image)

Although the analysis of deviance has the advantage that it controls for all sources of variation, the disadvantage is that the logit transform makes the results hard to interpret. To provide a more intuitive display of the results, Figure 3 shows the median AUC of each algorithm (with 95% confidence intervals) computed over the entire benchmark collection. The drawback of this display is that instead of controlling for the other sources of variation, it averages them away. The relative order of the algorithms is the similar as in the analysis of deviance, but now iForest takes the top spot and LOF drops to third. The low average AUC values reflect the fact that the benchmark collection contains many extremely difficult datasets.

5.3. Point Difficulty

Figure 4 shows that as the point difficulty increases (relative to pd-1), the performance degrades for all algorithms. This confirms that our measure of point difficulty correctly captures a notion of difficulty.
To gain intuition, we also plot the median AUC for each algorithm as a function of point difficulty (Figure 5). This confirms the result and also shows that there is much more variation in algorithm performance on easy problems (pd-1 and pd-2) than on hard ones (pd-4).

5.4. Relative Frequency

Figure 6 shows that anomalies are harder to detect as they become more frequent (relative to rf-1), and again this trend is confirmed by the mean AUC for each algorithm at each frequency level as shown in Figure 7. Before doing this experiment, we had conflicting intuitions. One intuition from supervised learning was that having more instances of the minority class (the anomalies) would make the problem easier. This is clearly incorrect. The other intuition is that as the anomalies become more common, they become less distinctive (from each other and from the normal points). One can imagine cases where additional anomalies become progressively more distant from the normal data, for example, where the process generating the anomalies is drifting away from the process generating the normal points. Our benchmark datasets do not mimic such a situation. Instead, because the anomalous points all occupy the original volume spanned by the mother dataset, they will tend to be become more closely packed as they become more numerous. This defeats the anomaly detection algorithms.
5.5. Clusteredness
Figure 3 shows that in general anomalies become harder to detect as they become more clustered (relative to cl-0). This does not match the plot of mean AUCs shown in Figure 9, which shows the advantage of using an analysis of deviance to control for the other factors. The analysis of deviance shows a steady decline in AUC as the data become more clustered, whereas the mean AUC plot shows a paradoxical improvement in performance at the highest levels of clusteredness. These are artifacts of the particular mix of mother sets that are capable of producing cl-5 and cl-6 benchmark datasets.

5.6. Feature Irrelevance
Figure 10 demonstrates the negative impact of adding noise to a benchmark (relative to fi-0). This is an easily anticipated result, but it verifies that our methodology is able to create benchmarks that vary feature irrelevance.

Figure 11 shows that some algorithms (especially OCSVM and EGMM) suffer serious breakdowns at high levels of feature irrelevance while others (LOF and RKDE) maintain good performance. We find this surprising. We expected that any algorithm based on Euclidean distance (i.e., LOF, RKDE, OCSVM, and SVDD) would perform poorly, because the irrelevant features would perturb the computed distances. Whereas, we assumed that because EGMM applies PCA, it would be able to detect and ignore the irrelevant features. However, in retrospect, we hypothesize that because PCA seeks to explain the observed variation in the training data, it must dedicate separate components to each of the (uncorrelated) noise dimensions. Hence, it is not able to remove them.

6. DISCUSSION AND CONCLUSIONS
We offer several observations and discussion about the results, beginning with admission of a few shortcomings.
6.1. Imbalanced Coefficients

In an ideal analysis of deviance, all configurations of the data factors should contain the same number of points (micro-experiments, in our case). However, as we have discussed above, we were not able to completely achieve this. For example, we found that Gas Sensor Array Drift produced the most difficult sets. Recall also that for this set, only benchmark datasets with pd-1 could be produced. Consequently, it is impossible to assess the impact of point difficulty on this mother set, which could exaggerate the importance of all other factors in explaining the AUCs.
A similar artifact may be present in our analysis of clusteredness. All other problem dimensions are rigidly enforced, but variation in clusteredness depends critically on the mother set, relative frequency and how many points were available at a particular point difficulty level. Hence, the impact of clusteredness is not estimated well and may be confounding estimates of the impact of other factors.

6.2. The Failure of One-Class SVM

We made a good faith effort to tune the hyper-parameters of each algorithm to maximize performance, but it can still be difficult to tune algorithms written by other researchers. OCSVM performed much worse than all of the other algorithms. This may be a correct assessment, but it may also be the result of imperfect tuning. There are many proposed improvements to OCSVM in the literature; we applied only the most basic publicly-available implementation.

6.3. Isolation Forest versus SC Isolation Forest

The finding that SCIF performs significantly worse than its predecessor, IForest, was a matter of concern for us. We expected SCIF to be more robust and adaptable, as argued in [Liu et al. 2010]. We discuss the discrepancy here.

The implementation of IForest that we employed is the one distributed by Liu [Liu et al. 2008]. In contrast, we could find no publicly-available implementation of SCIF, so we wrote our own based on Liu, et al. [Liu et al. 2010]. When the results of SCIF were dramatically worse than IForest, we examined the IForest code carefully. We found that Liu’s implementation of IForest differs from the suggestions made in their 2008 paper. The key difference is that the paper recommends that each tree in the forest should be grown using a subsample of size 256, whereas the code uses the whole sample.

We reevaluated how we parameterized both algorithms, and arrived at the sample rates described earlier. However, there are many additional parameters for SCIF, most
notably the number of projections sampled at each split, and it proved impractical to push the algorithm to a level where it might compete with iForest. Given enough projection samples, SCIF might conceivably outperform iForest and LOF, but the algorithm would not be as easy to understand, nor as computationally simple as iForest and so we do not recommend it. We note that that the original authors have abandoned the algorithm in their own publications and have not made an implementation available.

ACM Transactions on Knowledge Discovery from Data, Vol. -, No. -, Article -, Publication date: February 2015.
6.4. Feature Irrelevance and Real Applications

Feature irrelevance is very common in real applications, because the application engineer rarely knows exactly which features are the right ones to include in an anomaly detection system. Hence, it is common to include any feature that might conceivably be relevant. Consequently, it was encouraging to see that, with the exception of OCSVM and EGMM, the performance of the anomaly detection algorithms was only damaged slightly by increasing levels of irrelevance.

EGMM was the top-performing method when no irrelevant features were added. But its performance declines with increasing irrelevant features until at level \( f_i-4 \), it is doing no better than random guessing. An interesting question for future research is to understand why EGMM is so vulnerable to added noise and why the other algorithms are much less affected.

6.5. Problem Dimensions versus Algorithms

It is important to note that in all cases, the changes in performance observed from manipulating problem dimensions were larger than the differences in the performance among the anomaly detection algorithms (with the exception of OCSVM). This has several implications.

For point difficulty, it confirms that on hard points, the choice of algorithm is irrelevant. Clearly, there are anomaly detection problems that are extremely difficult or impossible to solve.

For relative frequency, it suggests that practitioners should look for ways to reduce the relative frequency of the anomalies (e.g., by filtering out some anomalies using domain knowledge or by obtaining more data for normal cases). If the relative frequency of the anomalies can be driven below 1%, then anomaly detection performance is likely to improve.

Increasing clusteredness severely reduced the effectiveness of all the anomaly detection algorithms. This suggests that it is very important to eliminate sources of duplicate or near-duplicate data points (e.g., [Lee et al. 2013]). It also poses an important
challenge for future research: How can we create anomaly detection algorithms that can handle clustered anomalies well?

The feature irrelevance results suggest that practitioners should only include features that they have strong reason to believe are relevant. Unlike in supervised learning, where feature selection methods are very mature, anomaly detection algorithms do not handle irrelevant features well. In particular, we can recommend against using EGMM in settings where there may be many irrelevant features.
Finally, attending to these aspects of problem formulation and data preparation are likely to produce a bigger gain in performance than spending time trying to select the best algorithm. We were impressed by the robust performance of the Isolation Forest method, so we recommend using it as a starting point.

7. SUMMARY

This paper presented a methodology for creating anomaly detection benchmarks and techniques for controlling four important properties of those benchmarks (point difficulty, relative frequency, normalized clusteredness and feature irrelevance). Some evidence suggests that deriving benchmarks from regression problems offers more control over these properties.

Experimental tests based on thousands of benchmark datasets demonstrate that these four properties strongly influence the behavior of several leading anomaly detection algorithms.

Across a large set of benchmarks, we found that Isolation Forest is the most accurate algorithm overall, followed very closely by Local Outlier Factor, Robust Kernel Density Estimation, and Ensembled Gaussian Mixture Models. However, we have shown that the influence of the problem dimensions is greater than the influence of the algorithm.

REFERENCES

Bharat B Aggarwal, Divya Danda, Shan Gupta, Prashamika Gehlot, and others. 2009. Models For Prevention and Treatment of Cancer: Problems vs Promises. Biochemical pharmacology 78, 9 (2009), 1083.

Ahmad Alzghoul and Magnus Lofstrand. 2011. Increasing availability of industrial systems through data stream mining. Computers & Industrial Engineering 60, 2 (2011), 195 – 205. DOI: [http://dx.doi.org/10.1016/j.cie.2010.10.008](http://dx.doi.org/10.1016/j.cie.2010.10.008)

K. Bache and M. Lichman. 2013. UCI Machine Learning Repository. (2013). [http://archive.ics.uci.edu/ml](http://archive.ics.uci.edu/ml)

Leo Breiman. 2001. Random Forests. Machine Learning 45, 1 (2001), 5–32.

M. Breunig, H-P. Kriegel, R. T. Raymond T. Ng, and J. Sander. 2000. LOF: Identifying Density-Based Local Outliers. ACM SIGMOD Record (2000), 93–104.
Robert A. Bridges, John Collins, Eric Ferragut, Jason Laska, and Blair D. Sullivan. 2014. Multi-Level Anomaly Detection on Streaming Graph Data. Technical Report 1410.4355v1. arXiv.
C.-C. Chang and C.-J. Lin. 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2 (2011), 27:1–27:27.
Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. 2009. Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems 47, 4 (2009), 547–553.
Kaustav Das, Jeff Schneider, and Daniel B Neill. 2008. Anomaly pattern detection in categorical datasets. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 169–176.
Dorothy E. Denning. 1987. An intrusion-detection model. *Software Engineering, IEEE Transactions on* 2 (1987), 222–232.

Ethan Derszewski and Thomas G. Dietterich. 2011. Spatiotemporal Models for Anomaly Detection in Dynamic Environmental Monitoring Campaigns. *ACM Transactions on Sensor Networks* 8, 1 (2011), 3:1–3:26.

C. Elkan. 1999. Results of the KDD’99 Classifier Learning Contest. (1999). [http://www.cs.ucsd.edu/users/elkan/cleareults.html](http://www.cs.ucsd.edu/users/elkan/cleareults.html)

Joshua Glasser and Brian Lindauer. 2013. Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. In *2013 IEEE Security and Privacy Workshops*. IEEE Press, 98–104.

Teofil F. Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science* 38, 0 (1985), 293 – 306. DOI: [http://dx.doi.org/10.1016/0304-3975(85)90224-5](http://dx.doi.org/10.1016/0304-3975(85)90224-5)

Julie Greensmith, Jamie Twycross, and Uwe Aickelin. 2006. Dendritic cells for anomaly detection. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 664–671.

D. M. Hawkins. 1980. *Identification of Outliers*. Chapman and Hall. [http://books.google.com/books?id=fbOQAAAAQBAJ](http://books.google.com/books?id=fbOQAAAAQBAJ)

Huaming Huang, Kishan Mehrotra, and Chilukuri K. Mohan. 2013. An Online Anomalous Time Series Detection Algorithm for Univariate Data Streams. In *Recent Trends in Applied Artificial Intelligence*. Springer, 151–160.

Tommi Jaakkola and David Haussler. 1999. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, Vol. 126. San Mateo, CA, 00–04.

S.S. Keerthi, K.B. Duan, S.K. Shevade, and A.N. Poo. 2005. A Fast Dual Algorithm for Kernel Logistic Regression. *Machine Learning* 61, 1-3 (2005), 151–165. DOI: [http://dx.doi.org/10.1007/s10994-005-0768-5](http://dx.doi.org/10.1007/s10994-005-0768-5)

Terran Lane and Carla E. Brodley. 1997. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*. 43–49.

Ar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. 2003. A comparative study of anomaly detection schemes in network intrusion detection. In *In Proceedings of SIAM Conference on Data Mining*.

Jay Yoon Lee, U. Kang, Danai Koutra, and Christos Faloutsos. 2013. Fast anomaly detection despite the duplicates. In *22nd International World Wide Web Conference, WWW ’13, Rio de Janeiro, Brazil, May 13–17, 2013, Companion Volume*. 195–196. [http://dl.acm.org/citation.cfm?id=2487888](http://dl.acm.org/citation.cfm?id=2487888)

F. T. Liu, K. M. Ting, and Z.-H. Zhou. 2008. Isolation forest. In *Proceedings of the IEEE International Conference on Data Mining*, 413–422.

F. T. Liu, K. M. Ting, and Z.-H. Zhou. 2010. On Detecting Clustered Anomalies using SCIForest. In *Machine Learning and Knowledge Discovery in Databases*. 274–290.

Matthew V. Mahoney and Philip K. Chan. 2003. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *Recent Advances in Intrusion Detection*. Springer, 220–237.

John McHugh. 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM transactions on Information and system Security* 3, 4 (2000), 262–294.

Qipei Mei and Mustafa Gul. 2013. An Improved Methodology for Anomaly Detection Based on Time Series Modeling. In *Topics in Dynamics of Civil Structures, Volume 4*. Springer, 277–281.

Matthew Eric Otey, Amol Ghoting, and Sririnivasan Parthasarathy. 2006. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery* 12, 2-3 (2006), 203–228.

D. Pokrajac, A. Lazarevic, and L.J. Latecki. 2007. Incremental Local Outlier Detection for Data Streams. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*. 504–515. DOI: [http://dx.doi.org/10.1109/CIDM.2007.368917](http://dx.doi.org/10.1109/CIDM.2007.368917)

Kemal Polat, Seral Sahin, Halife Kodaz, and Salih Gunes. 2005. A New Classification Method for Breast Cancer Diagnosis: Feature Selection Artificial Immune Recognition System (FS-AIRS). In *Advances in Natural Computation*, Lipo Wang, Ke Chen, and Yew Soon Ong (Eds.). Lecture Notes in Computer Science, Vol. 3611. Springer Berlin Heidelberg, 830–838. DOI: [http://dx.doi.org/10.1007/11539117_117](http://dx.doi.org/10.1007/11539117_117)

Leonid Portnoy, Eleazar Eskin, and Sal Stolfo. 2001. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001).* Celticee.

David M. Rocke and David L. Woodruff. 1996. Identification of Outliers in Multivariate Data. *J. Amer. Statist. Assoc.* 91, 435 (1996), 1047–1061.
B. Scholkopf, J. C. Platt, J. Shawe-taylor, A. J. Smola, and R. C. Williamson. 1999. Estimating the Support of a High-Dimensional Distribution. (1999).

Ted E Senator, G Henry, Alex Memory Goldberg, William T Young, Brad Rees, Robert Pierce, Daniel Huang, Matthew Reardon, Jay-Yoon Lee, Danai Koutra, and others. 2013. Detecting Insider Threats in a Real Corporate Database of Computer Usage Activity. (2013).

Tax and Duin. 2004. Support Vector Data Description. Machine Learning 54 (2004), 45–66.

K. L. Wagstaff, N. L. Lanza, D. R. Thompson, T. G. Dietterich, and M. S. Gilmore. 2013. Guiding Scientific Discovery with Explanations using DEMUD. In Proceedings of the Association for the Advancement of Artificial Intelligence AAAI 2013 Conference.

Feng Xue, Weizhong Yan, Nicholas Roddy, and Anil Varma. 2006. Operational data based anomaly detection for locomotive diagnostics. In International Conference on Machine Learning: 236–241.

Bin Zhang, C. Sconyers, C. Byington, R. Patrick, M. Orchard, and G. Vachtsevanos. 2008. Anomaly detection: A robust approach to detection of unanticipated faults. In Prognostics and Health Management, 2008. PHM 2008. International Conference on. 1–8. doi: http://dx.doi.org/10.1109/PHM.2008.4711445

Jiong Zhang and Mohammad Zulkernine. Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection. (????).

Ji Zhu and Trevor Hastie. 2001. Kernel Logistic Regression and the Import Vector Machine. In Journal of Computational and Graphical Statistics. MIT Press, 1081–1088.