Empirical Case Studies in Attribute Noise Detection

Taghi M. Khoshgoftaar, *Member, IEEE*, and Jason Van Hulse, *Member, IEEE*

***Abstract*—The quality of data is an important issue in any domain-specific data mining and knowledge discovery initiative. The validity of solutions produced by data-driven algorithms can be diminished if the data being analyzed are of low quality. The quality of data is often realized in terms of data noise present in the given dataset and can include noisy attributes or labeling errors. Hence, tools for improving the quality of data are important to the data mining analyst. We present a comprehensive empirical investigation of our new and innovative technique for ranking attributes in a given dataset from most to least noisy. Upon identifying the noisy attributes, specific treatments can be applied depending on how the data are to be used. In a classification setting, for example, if the class label is determined to contain the most noise, processes to cleanse this important attribute may be undertaken. Independent variables or predictors that have a low correlation to the class attribute and appear noisy may be eliminated from the analysis. Several case studies using both real-world and synthetic datasets are presented in this study. The noise detection performance is evaluated by injecting noise into multiple attributes at different noise levels. The empirical results demonstrate conclusively that our technique provides a very accurate and useful ranking of noisy attributes in a given dataset.**

***Index Terms*—Attribute noise, data cleaning, data quality, noise detection, pairwise attribute noise detection algorithm (PANDA).**

I. Introduction

THE QUALITY of data, which is often compromised by the presence of noise, is an important research issue in the data mining and knowledge discovery field. Noise in a dataset can adversely affect decisions that are based on modeling and analysis of that data. Consequently, it is very important to employ data cleansing procedures to enhance the quality of data prior to any data mining analysis.

Generally speaking, data noise is categorized into two groups [2]: class noise and attribute (feature) noise. Class noise occurs when an error exists in the class label of an instance, while attribute noise occurs when the values of one or more attributes of an instance are corrupted or incorrect. Detecting instances that contain class noise has received relatively more attention in data mining research [1], [6], [10], [20], in part due to the greater impact of class noise on classification accuracy [19].

In real-world data mining projects, data may be collected from multiple heterogeneous sources and aggregated into a single dataset. Given the number of ways errors can be introduced into data and the increasing size of databases and data warehouses, data cleaning is a prevalent problem in building a useful data warehouse [15]. One common cause of low-quality data, known as the merge/purge problem [7], [13], occurs when records appearing in different sources that correspond to the same real-world entity are not matched when the data sources are merged together. It has been suggested that programs be used to gain metadata information to aid in the detection of data inconsistencies, which can be used by the researcher during analysis [15].

We recently proposed the pairwise attribute noise detection algorithm (PANDA) for the noise detection problem [18]. PANDA yields a relative ranking of instances from the most to least noisy. It examines, for each instance, each pair of attributes and computes the deviation of the second attribute from its mean value given the partitioned value of the first attribute. For a given instance, if these deviations occur often and severely enough when compared to the remainder of the dataset, that instance will appear more noisy. PANDA is attractive because it can be used with or without knowledge of class labels, in contrast to other noise detection procedures [6], [17]. If the (binary) class labels are available, however, they can be incorporated into PANDA by treating them as another attribute.

This study proposes a technique that can be used to obtain a relative ranking of the attributes in a dataset from most to least noisy. This is in contrast to our previous work [18] mentioned above, which specifically identifies noisy *instances*, another very important problem in data mining. Aggregating noise in the instances to the attribute level provides insight to the domain-specific practitioner regarding the relative quality of the attributes. In a classification setting, attributes determined to be relatively noisy could be eliminated from further analysis if they are not very useful class predictors. The noisy attributes that are known to be significant for predicting the class label can undergo cleansing prior to further data analysis.

Our procedure may also be used to understand the relative amount of noise in the class when compared to the other attributes in the dataset. If the class label is ranked as one of the noisiest attributes, the user is alerted that the class should undergo treatment because of its relative importance in the construction of a classification model. On the other hand, if the class label is determined to contain minimal noise, data cleaning efforts might be directed toward the more important independent variables that contain considerable noise.

Regardless of whether the data will be used for a supervised or unsupervised learning task, our technique can provide valuable information to a user about the relative quality of each attribute. Such information can be used to scrutinize the data collection process, potentially improving the quality of data for future analysis. A simple example might be a measurement device that records the value of a specific attribute related to an entity. Noise detected in that attribute may potentially uncover a calibration error in that device, which can be corrected so that future data

Manuscript received December 28, 2007; revised October 25, 2008. First published April 17, 2009; current version published June 17, 2009. This paper was presented in part at the 2005 IEEE Information Reuse and Integration (IRI) Conference [11]. This paper was recommended by Associate Editor S. Rubin.

The authors are with the Empirical Software Engineering Laboratory, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: taghi@cse.fau.edu; jvanhulse@gmail.com).

Digital Object Identifier 10.1109/TSMCC.2009.2013815

will no longer be biased. By providing a ranking of attributes, time can be spent examining and treating those attributes that are deemed as noisy, thereby maximizing resource utilization during the data preprocessing phase of a data mining task. Note that the dataset may or may not have a dependent variable or class; hence our procedure can be considered an unsupervised data mining task. To the best of our knowledge, this is the first technique to address the problem of ranking attributes in a dataset based on the relative amount of noise they contain.

Feature selection is an important concept in data mining, and has received significant attention [4]. Since one of the primary uses of our procedure is to remove noisy features from the dataset, our technique can be viewed as a type of feature selection algorithm. However we are not aware of any feature selection procedures that specifically identify noisy features, as most concentrate on attribute redundancy or predictive efficacy.

The utility of our methodology is investigated using two real-world software measurement datasets and one simulated dataset into which noise was injected at varying levels. The performance of our approach on the real-world datasets is evaluated based on inspection of the results by a software engineering domain expert. In the case of the simulated dataset, noise was injected into various attributes at different levels. Noise was injected into either a single attribute, two attributes independently, or three attributes independently. We present the results for 5%, 10%, and 20% noise levels. A total of 39 different datasets were derived from the simulated data with varying amounts of noise and the results are presented in this study. In addition, many other experiments with both real-world and simulated data have demonstrated similar results and are therefore not presented due to space limitations. The large number of experiments conducted and the strength of the results lead us to conclude that our procedure is very successful at identifying noisy attributes.

The remainder of this paper is organized as follows. Section II describes the algorithm proposed in this study. Section III provides descriptions of the case study datasets, and Section IV presents empirical results of the case studies. Section V presents conclusions and future work.

II. Methodology

An outline of the PANDA algorithm for detecting noisy instances is presented in Section II-A while the approach for obtaining a noise-based ranking of attributes is presented in Section II-B.

A. Instance Noise Detection Algorithm

PANDA is presented in Fig. 1. Attribute x_{*j} is partitioned into L disjoint bins[1], creating the attribute $\hat{x}_{*j}$ in step 1. If $\hat{x}_{ij}$ denotes the value of partitioned attribute j for instance i, then $\hat{x}_{ij} \in \{1, \ldots, L\}$. In other words, the output of the binning process is a set of disjoint intervals of attribute x_{*j} labeled by integers 1 to L. In step 2, the conditional mean and standard deviation for each of the nonpartitioned attributes $x_{*k}, k \neq j$, are calculated relative to the partitioned attribute $\hat{x}_{*j}$. In other words, the mean and standard deviation of attribute x_{*k} is calculated relative to the set of instances with the same value for attribute $\hat{x}_{*j}$, which ranges between 1 and L. The noise factor S_i of an observation, calculated in step 3, is the sum of all of the standardized attribute values for the observation over all ordered pairs of attributes $(\hat{x}_{*j}, x_{*k})$, $1 \leq j \neq k \leq m$, where m is the total number of attributes.

PANDA Algorithm

input: Dataset $X = [x_{ij}]_{n \times m}$ with n observations and m attributes, where x_{ij} is the value of the j^{th} attribute for the i^{th} observation. x_{*j} denotes the j^{th} attribute.

output: Noise Factor S_i for all the observations in Dataset X

1) Partition each attribute x_{*j} into disjoint bins $\{1, \ldots, L\}$, where $L =$ # of partitions. Denote the partitioned attribute by $\hat{x}_{*j}$ and the value of the j^{th} partitioned attribute for instance i as $\hat{x}_{ij}$. Note in particular that $\hat{x}_{ij} \in \{1, \ldots, L\}\ \forall\ i$.
2) Calculate $Mean(x_{*k} \mid \hat{x}_{*j} = l)$ and $Std(x_{*k} \mid \hat{x}_{*j} = l)$ for each pair of atributes $(\hat{x}_{*j}, x_{*k})$, $1 \leq j \neq k \leq m$ and for $l = 1, \ldots, L$.
3) For each instance $i \in X$, calculate

$$S_i = \sum_{k=1}^{m} \sum_{\substack{j=1 \\ j \neq k}}^{m} \frac{\mid x_{ik} - Mean(x_{*k} \mid \hat{x}_{*j} = \hat{x}_{ij}) \mid}{Std(x_{*k} \mid \hat{x}_{*j} = \hat{x}_{ij})}.$$

4) Sort the instances based on S_i from the largest to the smallest value.

Fig. 1. Pairwise Attribute Noise Detection Algorithm.

If x_{ik} is the value of attribute k for instance i, then $m - 1$ standardized values are computed for this attribute based on the conditional mean and standard deviations of attribute x_{*k} given the partitioned value of each of the other $m - 1$ attributes. This process is repeated for each attribute value x_{ij}, $1 \leq j \leq m$. A total of $m * (m - 1)$ standardized values are calculated for each observation. The noise factor S_i of the instance is the sum of these standardized values. The observations with relatively large values of S_i may be considered as potential noise because an observation with a large standardized attribute value indicates a deviation from normal for that attribute.

Our recent work using a derivative of the JM1-8850 dataset that is considered in this case study (see Section III), called JM1-2445, has shown PANDA to be a useful technique for identifying noisy instances [18]. The case study included a validation of the results of PANDA by a domain expert after a manual inspection of the top 10% noisy instances as declared by PANDA. Based on that study we concluded that PANDA is indeed useful for identifying noisy instances in a dataset.

B. Ordering Attributes with Respect to Noise

The procedure to rank the attributes from most to least noisy is presented in Fig. 2. Suppose there are a total of n instances and m attributes. If it is available, the class can be used as one of the attributes, although it is not required for execution of the algorithm. PANDA is executed with all m attributes and the instance noise ranking is created. Denote this ordering by $rank$ and the rank of instance i as rank(i), where $1 \leq i \leq n$ (step 1 of Fig. 2). Each of the m attributes is removed and PANDA

[1]The partitioning algorithm utilized in our study is implemented in SAS [16], which uses an equal frequency binning algorithm. In practice, any partitioning technique can be used with PANDA.

Ranking Noisy Attributes
input: Dataset $X = [x_{ij}]_{n \times m}$ with n observations and m attributes, where x_{ij} is the i^{th} observation, j^{th} attribute in X.
output: Ranking of attributes from most to least noisy

1) Execute PANDA on the dataset X with all m attributes. The instance noise ranking produced by PANDA is denoted by $rank$, and $rank(i)$ is the ranking for instance i.
2) Do for j = 1 to m:
3) Execute PANDA on the dataset X with attribute j removed and denote the ranking as $rank_j$.
4) Calculate Kendall's Tau correlation between $rank$ and $rank_j$.
5) End
6) Sort the attributes based on their correlation to $rank$ using Kendall's Tau statistic. The noisiest attribute will have the lowest correlation with the ranking $rank$, while the cleanest attribute will have the highest correlation.

Fig. 2. Noisy Attribute Ranking Methodology.

is executed using only the remaining $m - 1$ attributes (step 3). The output is another instance ordering from most likely to least likely noise. If the jth attribute is removed, denote the output of PANDA by rank_j and the rank of instance i as $\text{rank}_j(i)$. For example, if the second attribute is removed and instance 100 is ranked by PANDA as most noisy, then $\text{rank}_2(100) = 1$.

Kendall's Tau rank correlation (see Section II-C) between rank_j and $rank$ is calculated for each attribute j (step 4). Attributes are ordered from most noisy to least noisy based on their correlation with the ranking $rank$ when the attribute is removed from the dataset. The attribute that creates the ranking with the lowest correlation to $rank$ after it is removed has the most noise, while the attribute with the highest correlation to $rank$ is considered to be the cleanest attribute.

Our attribute noise ranking procedure measures the volatility in the instance noise ranking when an attribute is removed from the dataset. If a large amount of volatility (i.e., a low correlation) is seen after the removal of an attribute, the removed attribute had a significant effect on the noise ranking. An attribute with very little volatility does not significantly contribute to the instance noise rankings and hence is a relatively clean attribute.

Suppose that a dataset contains only a single attribute with noise, while the remaining attributes are relatively clean. The initial noise ranking produced by PANDA utilizes the noisy attribute, and instances that contain noise will have a ranking close to one. If any of the relatively clean attributes are removed from the dataset and PANDA is again executed, the noisy attribute continues to be present and the instance noise ranking will not change significantly. On the other hand, if the noisy attribute is removed, PANDA is executed with only the remaining attributes, which are relatively clean. There is no reason to believe that the most noisy instance when all attributes are considered will also be ranked as very noisy when the noisy attribute is removed. This will result in a significant amount of volatility in the instance noise rankings and a low Kendall's Tau correlation to the ranking produced with all attributes present.

C. Kendall's Tau Rank Correlation

Kendall's Tau rank correlation statistic [3], [8] is used to measure the degree of similarity between the instance noise ranking produced before and after each attribute is removed. Suppose the rankings $rank$ and rank_j are being compared. For each observation i we have the ordered pair $(\text{rank}(i), \text{rank}_j(i))$, where $\text{rank}(i)$ is the rank of observation i produced by PANDA using all attributes and $\text{rank}_j(i)$ is the rank of observation i when attribute j is removed. For each pair of observations (x, y) the rankings $(\text{rank}(x), \text{rank}_j(x))$ and $(\text{rank}(y), \text{rank}_j(y))$ are compared and given a value of $+1$ or -1 depending on whether the two rankings are concordant or discordant. Assuming that both $rank$ and rank_j do not contain tied ranks, a pair of observations (x, y) is considered *concordant* if $\text{rank}(x) > \text{rank}(y)$ and $\text{rank}_j(x) > \text{rank}_j(y)$ or $\text{rank}(x) < \text{rank}(y)$ and $\text{rank}_j(x) < \text{rank}_j(y)$. Otherwise, (x, y) are said to be *discordant*. There are a total of $n(n-1)/2$ pairs of observations. If S is the sum of the scores for each pair of observations as determined by their concordance or discordance, Kendall's Tau is calculated as $\tau = S/\frac{n(n-1)}{2}$. If all pairs are concordant, then the two rankings are in complete agreement and $\tau = 1$. If the two rankings are exactly opposite, (i.e., $\forall\ i$, $\text{rank}_j(i) = n - \text{rank}(i) + 1$), then all pairs will be discordant and $\tau = -1$. If τ is close to zero, then the correlation between the two rankings is very weak.

D. Time Complexity

The time complexity of our technique can be analyzed as follows. Suppose m represents the number of attributes and n the number of instances. Computing the mean and standard deviations in the PANDA procedure for each pair of attributes and standardizing each attribute value has complexity $m^2 n$. Repeating this procedure by removing each attribute results in m executions of PANDA. Therefore, our technique for ordering noisy features has complexity $m^3 n$. Our technique is part of the data preprocessing tasks of a data mining initiative, so the calculations would be performed offline in a batch environment. Additionally, as $m \ll n$ in many real-world datasets, the scalability of our procedure is not a major concern, especially for small- and medium-sized datasets.

III. Description of Case Study Datasets

Experiments conducted in this study used two real-world datasets, JM1-8850 and CCCS, as well as an artificial dataset. JM1-8850 was derived from a National Aeronautics and Space Administration (NASA) software project written in C. The CCCS dataset is a large military command, control, and communications system written in Ada. The simulated dataset consists of nine numeric attributes derived from a multivariate Gaussian distribution.

A. JM1-8850 Dataset

The JM1 dataset originally consisted of 10 883 program modules (instances) [12]. Software metrics data was collected at the function level; hence a program module was defined as a function or subroutine. After inconsistent observations (identical values for the software attributes but different class labels) were removed, the dataset was reduced to 8850 observations, resulting in the JM1-8850 dataset.

TABLE I
SOFTWARE METRICS FOR JM1-8850 DATASET

Metric	Symbol	Abbreviation
Branch	Branch Count	bcnt
Line Count	Executable LOC	eloc
	Comments LOC	cloc
	Blank LOC	bloc
	Code and comments LOC	ccloc
	Total Lines of Code	tloc
Basic Halstead	Total Operators	toper
	Total Operands	topan
	Unique Operands	uopan
	Unique Operators	uoper
McCabe's	Cyclomatic Complexity	ccomp
	Essential Complexity	ecomp
	Design Complexity	dcomp

TABLE II
SOFTWARE METRICS FOR CCCS DATASET

Symbol	Abbreviation
Unique Operators	uoper
Total Operators	toper
Unique Operands	uopan
Total Operands	topan
Cyclomatic Complexity	ccomp
Logical Operators	logop
Total Lines of Code	tloc
Executable LOC	eloc

There were a total of 21 attributes (i.e., software metrics [5]) in the JM1-8850 dataset, including McCabe Metrics, derived and basic Halstead Metrics, Line Count and Branch Count Metrics. The eight derived Halstead metrics were not used in our study, leaving 13 software metrics remaining. These 13 software product metrics are shown in Table I, where LOC represents the lines of code in a program module.

The JM1-8850 dataset was also characterized by a dependent variable, class, with values nfp (not fault prone) and fp (fault prone). An instance was labeled fp, if it contained at least one fault and nfp otherwise.

B. CCCS Dataset

The CCCS dataset contains 282 instances, where each instance is a software (program) module from a large military command, control, and communications system [9]. More specifically, each instance is an Ada package, consisting of one or more procedures. CCCS contains eight independent variables or attributes along with an additional attribute, labeled $nfaults$, indicating the number of faults attributed to the module during the system integration and test phases, and during the first year of deployment. Of the 282 instances in the CCCS dataset, 136 contain at least one fault. The software metrics in the CCCS dataset are listed in Table II.

C. Simulated Dataset

The simulated dataset contains 1000 instances with nine numeric attributes derived from a multivariate Gaussian distribution with expert-specified parameters selected to resemble those of a real-world dataset such as CCCS. After the dataset was generated, noise was injected into certain attributes in order to test the effectiveness of our procedure. By maintaining control over the noise injection process, the attribute noise ranking capability of our procedure can be objectively evaluated.

TABLE III
STATISTICS FOR ARTIFICIAL DATASET PRIOR TO NOISE INJECTION

Attribute #	Kendall's Tau	Rank
1	0.81679	4
2	0.79777	2
3	0.83590	7
4	0.82030	5
5	0.83263	6
6	0.79607	1
7	0.81403	3
8	0.84372	8
9	0.90197	9

Table III displays Kendall's Tau correlation prior to any noise injection procedures, comparing the instance noise ranking created by PANDA with all attributes present to the noise ranking produced after removing each of the nine attributes. Even before noise injection, attributes contain different amounts of noise in a relative sense, as can be seen from Table III. Attribute 6, for example, has $\tau = 0.79607$, making it the noisiest attribute, whereas attribute 9 is the (relatively) cleanest attribute with $\tau = 0.90197$.

In the dataset corruption process, noise was injected into either one, two, or three attributes at the 5%, 10%, or 20% levels. We denote the dataset derived from the simulated data with attribute numbers α corrupted in β percent of the instances as $\alpha - \beta$. For example, 5–10 is used to denote the dataset where attribute 5 is corrupted in 10% of the instances while 413–5 is the dataset where attributes 4, 1, and 3 are each corrupted independently in 5% of the instances. When more than one attribute is chosen for noise injection, instances are corrupted independently for each attribute. For example, if attributes 4 and 5 are injected with 5% noise in the dataset of 1000 instances, 50 randomly chosen instances are corrupted in attribute 4 and an independently chosen set of 50 instances are also corrupted in attribute 5. Therefore some instances will be corrupted in two attributes due to random sampling while the majority of the chosen instances are corrupted in only a single attribute.

Once an instance was selected for corruption in a particular attribute, the attribute value was changed to the opposite extreme of the univariate distribution for that particular attribute. Details of our noise injection process are presented in Fig. 3. Corruption was done in this manner because we are interested in detecting noise that significantly changes the distribution of the dataset. Adding 20% to the maximum or subtracting 20% from the minimum value was deemed reasonable for our application, although other values can be chosen as well, for example 5% or 25%. Any process that corrupts the attribute values towards the extreme ends of the univariate distribution is reasonable, and this scheme was chosen based on expert guidance and because of its simplicity. The injection of noise into a dataset in a manner that simulates noise that might be seen in a real-world environment is an open research issue. Therefore, other corruption mechanisms can be used and will be further evaluated in future work.

Suppose:

- X is the attribute to be corrupted
- X_i is the value of attribute X for instance i, which was selected for corruption
- $\max(X)$ is the largest attribute value for X
- $\min(X)$ is the smallest attribute value for X
- $\text{median}(X)$ is the median attribute value for X
- $\hat{X}_i$ is the new attribute value for instance i after corruption
- $k = 20\% * \max(X)$

The corruption process is implemented as follows:

- If $X_i < \text{median}(X)$ then $\hat{X}_i = \max(X) + k$.
- If $X_i > \text{median}(X)$ then $\hat{X}_i = \min(X) - k$.
- If $X_i = \text{median}(X)$ then $\hat{X}_i = \text{random}(\min(X) - k, \max(X) + k)$ (i.e., if X_i is equal to the median value for X, then one of the two corrupted values is randomly selected with a 50% probability).

Fig. 3. Noise Injection Process.

Different noise levels were selected in order to test the sensitivity of our technique to various amounts of noise. The three noise levels presented here are 5%, 10%, and 20%. Attributes were chosen for corruption using the statistics presented in Table III. For the one attribute case, noise was injected into attributes 9, 4, 5, 6, and 8. In the simulated data before any noise injection, attributes 9 and 8 are the two cleanest attributes based on their value for τ, while attributes 4 and 5 are neither relatively clean nor noisy. Attribute 6 is the noisiest attribute in the dataset. The remaining attributes were not considered because they have a very similar value for τ as attributes that were already selected for noise injection. For example, attribute 2 is the second noisiest attribute with $\tau = 0.79777$, very close in value to attribute 6 that has $\tau = 0.79607$.

We also examined the results when two and three attributes are corrupted. For two attributes, four combinations of attributes were created: attributes 2 and 6, representing the case where the two noisiest attributes are corrupted, attributes 8 and 9, representing the case where the two cleanest attributes are corrupted, attributes 4 and 5, which are neither clean nor noisy in a relative sense, and attributes 6 and 9, which combine the noisiest and cleanest attributes. These four combinations of two attributes represent a wide array of scenarios.

For three attributes, attributes 1, 4, and 5 in the first case, 3, 8, and 9 in the second case, 7, 2, and 6 in the third case and 4, 6, and 9 in the final case were corrupted. The second and third cases represent the cleanest and noisiest attributes, respectively, while the final case covers one relatively clean attribute (attribute 9), one relatively noisy attribute (attribute 6), and one attribute in the middle. The first case covers the scenario when all three attributes are neither relatively clean nor dirty. All combinations of one, two, or three noisy attribute datasets were corrupted with either 5%, 10%, or 20% noise in each attribute, creating a total of 39 different datasets with varying degrees of noise in a variety of attributes.

IV. Empirical Case Studies

The objective of the studies presented in this section is to determine the effectiveness of our technique at identifying noisy attributes. The performance of the technique is studied using two real-world datasets, where inspection by a software engineering expert of the results is necessary to judge the performance of the algorithm, and on artificial data where noise is injected in a controlled manner and the noisy attributes are known in advance. For the CCCS and JM1-8850 datasets, expert inspection is needed only to measure the effectiveness of the algorithm and is not required for the algorithm itself to execute. In the case of artificial data with noise injection, the initial distribution of the dataset is known (i.e., multivariate Gaussian). The noise injection process provides a controlled environment, where different noise parameters can be tested (e.g., the number of attributes with noise as well as the amount of noise in each attribute), and the capability of our technique to rank noisy attributes can be evaluated.

This methodology was chosen for experimentation instead of considering publicly available datasets (for example the UCI repository [14]), which come from an unfamiliar domain. Using a dataset with randomly injected noise to measure the performance of our procedure without proper domain knowledge can be problematic. First, it is unknown whether the original dataset was noise-free before the injection process. Injecting noise into a dataset that already contains noise makes the analysis of the effectiveness of a noise detection algorithm very difficult. Second, injected noise may not represent the types of noise present in a real-world dataset from the given domain. Therefore we demonstrate the effectiveness of our procedure using real-world datasets with real-world noise as well as a simulated dataset with various amounts of injected noise.

The primary statistic of interest for each attribute is Kendall's Tau rank correlation, produced after the attribute is removed from the dataset and PANDA is executed with the remaining attributes. Kendall's Tau correlation is denoted by τ, and the notation $\tau_{\alpha-\beta}(n)$ denotes Kendall's Tau correlation of attribute n given dataset $\alpha - \beta$. When the dataset of interest is obvious we will shorten the notation and simply use $\tau(n)$. In a similar manner, the ranking of attribute n in dataset $\alpha - \beta$ will be denoted by $\rho_{\alpha-\beta}(n)$. The simulated dataset before any artificial noise injection will be denoted by SIM, and $\tau_{\text{SIM}}(n)$ is Kendall's Tau rank correlation of attribute n in the simulated dataset before any noise injection.

Although Kendall's Tau correlation is the measure used to rank the attributes, we also on occasion provide some other statistics of each attribute derived from the distribution of the difference in ranking for all of the observations in the dataset, specifically $|\text{rank}_j(i) - \text{rank}(i)|$. These statistics are the maximum difference $= \max_i[|\text{rank}_j(i) - \text{rank}(i)|]$, minimum difference $= \min_i[|\text{rank}_j(i) - \text{rank}(i)|]$, average difference $= (\sum_{i=1}^{n} |\text{rank}_j(i) - \text{rank}(i)|)/n$, and the standard deviation of the difference in ranking. Also calculated are the 10th, 25th, 50th, 75th, and 90th percentiles in the distribution.

As an additional enhancement to reduce the effect of noise inadvertently injected into the dataset as a result of binning, PANDA is run multiple times with a different number of bins specified in each execution. The final instance noise ranking that is produced by PANDA is calculated as the median ranking over all bins. Using multiple bin sizes and calculating the median ranking helps diminish any adverse effects binning may have as

TABLE IV
NOISE RANKING, ARTIFICIAL DATASET 9

	Dataset 9-5		Dataset 9-10		Dataset 9-20	
	τ	ρ	τ	ρ	τ	ρ
1	0.82824	5	0.83178	5	0.83154	5
2	0.80562	2	0.81170	2	0.81168	3
3	0.84368	7	0.84846	8	0.84638	8
4	0.82768	4	0.83553	6	0.83183	6
5	0.83704	6	0.84441	7	0.83823	7
6	0.80386	1	0.81112	1	0.80854	2
7	0.82293	3	0.83114	4	0.82573	4
8	0.84953	8	0.85196	9	0.84903	9
9	0.86283	9	0.82033	3	0.78363	1

well as reducing the possibility that an instance was mistakenly identified as noise due to an anomaly introduced by the binning procedure. Our experiments have shown that the procedure is not particularly sensitive to the selection of bin sizes, especially when an average over multiple bin sizes is used. We recommend using a range of bin sizes as in our experiments, with no fewer than five bins in total. Conversely, each bin should have at least 10 examples, so no more than $N/10$ bins should be used, where N is the number of instances in the dataset.

A. Case Study with Simulated Noise

1) Injected Noise in One Attribute: This case study examines the impact of noise injection into one attribute using our simulated dataset. The goal of this case study is to measure the impact of noise injection into various attributes ranging from the cleanest attributes to the most noisy. We anticipate the noise ranking of an attribute to decrease as more noise is injected, regardless of the original noise level of the attribute. We also anticipate that the attributes that do not receive noise injection should maintain a very similar noise ranking, indicating a measure of stability in our procedure. For all of the experiments using simulated data, bin sizes 5, 7, 9, 12, 15, 17, 20, 25, and 30 were used, with the final instance ranking calculated as the median ranking over these nine iterations.

The results for dataset 9–5, which has attribute 9 corrupted in 5% of the instances, are presented in Table IV. Since there are a total of 1000 instances, this equates to 50 randomly selected instances with noise injected into attribute 9. Injecting 50 noisy instances into attribute 9, as in dataset 9–5, does not dramatically change the attribute noise ranking. In fact, all of the attributes maintain exactly the same relative ranking prior to noise injection except for attributes 1 and 4, which had rankings 4 and 5 in the original dataset but are now ranked as 5 and 4. By examining the Kendall's Tau correlation of these two attributes in dataset 9–5, however, one can see that the values are extremely close (i.e., $\tau(1) = 0.82824$ and $\tau(4) = 0.82768$). These two attributes, therefore, have essentially the same noise characteristics.

When 5% noise is injected into attribute 9, $\tau_{9-5}(9)$ drops to 0.86283. From Table III, which displays the attribute noise ranking before noise injection, attribute 9 is the cleanest attribute with $\tau_{\mathrm{SIM}}(9) = 0.90197$. For attribute 8, the second cleanest attribute in the simulated dataset, $\tau_{\mathrm{SIM}}(8) = 0.84372$, while after noise injection, $\tau_{9-5}(8) = 0.84953$. Noise injection has little effect on τ for attribute 8, as would be expected, but for attribute 9, τ drops over 4.3% to 0.86283. While attribute 9 is still rated as the cleanest attribute after noise injection in dataset 9–5, in a relative sense it is not as clean as it was, as can be seen with the drop in τ for attribute 9 when compared to the change in τ for attribute 8.

Table IV also displays the results when attribute 9 is corrupted with 10% and 20% noise. From the columns labeled "dataset 9–10", representing the attribute noise ranking results when 10% or 100 noisy instances are injected into attribute 9, $\tau(9) = 0.82033$, making attribute 9 the third noisiest attribute in dataset 9–10. Attribute 8 has become the cleanest attribute, with $\tau(8) = 0.85196$, while attribute 3 has become the second cleanest attribute, with $\tau(3) = 0.84846$.

For dataset 9–20, attribute 9 is now determined to be the noisiest attribute, with $\tau(9) = 0.78363$. Attribute 6 becomes the second noisiest attribute, with $\tau(6) = 0.80854$. This is in contrast to the scenarios when less noise is present in attribute 9, as in these cases, attribute 6 is ranked as the noisiest. On the other hand, attributes 8 and 3 continue to be relatively clean attributes.

These results are significant for a number of reasons. First, as more noise is injected into attribute 9, the relative noise ranking of attribute 9 steadily decreases (i.e., attribute 9 becomes more noisy relative to the other attributes in the dataset). In summary, we have $\rho_{9-5}(9) = 9$, $\rho_{9-10}(9) = 3$ and $\rho_{9-20}(9) = 1$. These results are intuitive, as it is not reasonable to expect the injection of 50 noisy instances (in the case of dataset 9–5) to necessarily result in the cleanest attribute becoming the most noisy, as noise already exists in other attributes. From Table III, attribute 9 is clearly seen to be the cleanest attribute by a wide margin relative to the other eight attributes.

Second, Kendall's Tau rank correlation exhibits a high degree of stability not only in the rankings but in the values of τ as the amount of noise increases. For example, attribute 2 has $\tau_{\mathrm{SIM}} = 0.79777$, $\tau_{9-5} = 0.80562$, $\tau_{9-10} = 0.81170$ and $\tau_{9-20} = 0.81168$. The relative noise ranking of attribute 2 in each of these cases is 2, 2, 2, and 3 for datasets SIM, 9–5, 9–10, and 9–20, respectively. Similar results can be seen for the other attributes, further supporting the effectiveness and validity of our technique.

The results when 5%, 10%, and 20% noise were injected into attribute 4 are displayed in Table V. From Table III, attribute 4 has $\tau_{\mathrm{SIM}}(4) = 0.82030$ and $\rho_{\mathrm{SIM}}(4) = 5$. In a relative sense, therefore, attribute 4 is neither clean nor noisy. It is therefore anticipated that the injection of 50 noisy instances will have a more dramatic impact for attribute 4 than it did for attribute 9. From Table V, $\tau_{4-5}(4) = 0.80740$ and $\rho_{4-5}(4) = 3$. After the injection of 50 noisy instances into attribute 4, τ has dropped by 0.0129 and the noise ranking of attribute 4 has dropped by two places to 3. Note that the noisiest attribute in dataset 4–5, attribute 6, has $\tau = 0.80561$, which is not much lower than $\tau(4)$.

For dataset 4–10, $\tau(4) = 0.77224$ and $\rho(4) = 1$ making attribute 4 the noisiest attribute. The second most noisy attribute is attribute 6, with $\tau(6) = 0.80718$. Attribute 4 is now clearly the most noisy attribute in dataset 4–10 by a wide margin over attribute 6. When 20% noise is injected into attribute 4, $\tau(4)$

TABLE V
NOISE RANKING, ARTIFICIAL DATASET 4

	Dataset 4-5		Dataset 4-10		Dataset 4-20	
	τ	ρ	τ	ρ	τ	ρ
1	0.82426	5	0.82770	5	0.82524	4
2	0.80615	2	0.80774	3	0.80451	2
3	0.84623	7	0.84715	7	0.84131	7
4	0.80740	3	0.77224	1	0.75750	1
5	0.84056	6	0.84231	6	0.84048	6
6	0.80561	1	0.80718	2	0.80810	3
7	0.82416	4	0.82737	4	0.82571	5
8	0.85196	8	0.85462	8	0.85021	8
9	0.90248	9	0.90348	9	0.89872	9

TABLE VI
NOISE RANKING, ARTIFICIAL DATASET 6

	Dataset 6-5		Dataset 6-10	
	τ	ρ	τ	ρ
1	0.82198	4	0.82340	4
2	0.80678	2	0.81051	2
3	0.84126	7	0.83839	6
4	0.82438	5	0.82529	5
5	0.83688	6	0.83863	7
6	0.78744	1	0.76185	1
7	0.82042	3	0.82134	3
8	0.85227	8	0.85441	8
9	0.90564	9	0.90406	9

drops further to 0.75750. Attribute 2 has now become the second most noisy attribute with $\tau(2) = 0.80451$, although Attribute 6 is very close with $\tau(6) = 0.80810$. As expected, the injection of 20% noise has caused $\tau(4)$ to drop even further relative to the values of the second most noisy attribute, compared to the injection of only 10% noise.

On the other hand, attributes 8, 9, and 3 continue to rank as the cleanest three attributes in the dataset regardless of the amount of noise injected into attribute 4. The value of τ remains stable for these three attributes, as does the relative difference in τ. As anticipated, noise injection into attribute 4 has no effect on these attributes.

Table VI displays the results when attribute 6 is corrupted with 5% and 10% noise (20% noise was also evaluated but the results are not shown). Before noise injection, attribute 6 is ranked as the noisiest attribute in the simulated dataset with $\tau_{\mathrm{SIM}}(6) = 0.79607$. Given that attribute 6 is already the noisiest attribute, we would expect noise injection to result in the ranking $\rho(6)$ to stay the same and for Kendall's Tau correlation to drop further in comparison to the other attributes. When 5% noise is injected into attribute 6, we have $\tau_{6-5}(6) = 0.78744$. As τ for attribute 2, the second most noisy attribute, has increased slightly to 0.80678, we have seen an increase in noise in attribute 6 relative to the other attributes. With 10% noise, $\tau(6) = 0.76185$, and with 20% noise (not shown), $\tau(6) = 0.74888$. Based on these results, we can conclude that the impact of noise injection can be measured even for attributes that already have a significant amount of noise.

Additional analysis was completed with noise injection into attributes 5 and 8. We omit the results, however, for space considerations and due to similarity in empirical conclusions to what has already been discussed.

TABLE VII
NOISE RANKING, ARTIFICIAL DATASET 89

	Dataset 89-5		Dataset 89-10		Dataset 89-20	
	τ	ρ	τ	ρ	τ	ρ
1	0.83185	5	0.84309	6	0.83268	6
2	0.81554	2	0.82584	4	0.82006	4
3	0.84684	8	0.85997	9	0.85134	9
4	0.83673	6	0.84504	7	0.83758	7
5	0.84384	7	0.85438	8	0.84401	8
6	0.81319	1	0.82271	3	0.81713	3
7	0.82706	4	0.83521	5	0.82515	5
8	0.82007	3	0.80206	1	0.76444	1
9	0.84752	9	0.82109	2	0.78245	2

TABLE VIII
NOISE RANKING, ARTIFICIAL DATASET 45

	Dataset 45-5		Dataset 45-10		Dataset 45-20	
	τ	ρ	τ	ρ	τ	ρ
1	0.82944	6	0.83420	5	0.83326	5
2	0.80762	3	0.82072	4	0.81570	4
3	0.84271	7	0.85184	7	0.84915	7
4	0.80226	1	0.79983	2	0.75863	1
5	0.81046	4	0.79201	1	0.76637	2
6	0.80581	2	0.81643	3	0.81454	3
7	0.82549	5	0.83502	6	0.83640	6
8	0.85265	8	0.86302	8	0.85896	8
9	0.89909	9	0.89926	9	0.89430	9

2) Injected Noise in Two Attributes: This case study is designed to evaluate the attribute noise ranking of our technique when various amounts of noise is independently injected into two attributes. We created four combinations of attributes: attributes 2 and 6, representing the case where the two noisiest attributes are corrupted, attributes 8 and 9, representing the case where the two cleanest attributes are corrupted, attributes 4 and 5, which are neither clean nor noisy in a relative sense, and attributes 6 and 9, with the noisiest attribute and the cleanest attribute together.

Table VII displays the results when attributes 8 and 9 are corrupted independently with 5%, 10%, and 20% noise. Before noise injection, attributes 8 and 9 are ranked as the two cleanest attributes in the dataset with $\tau_{\mathrm{SIM}}(8) = 0.84372$ and $\tau_{\mathrm{SIM}}(9) = 0.90197$. With 5% noise, attribute 8 has $\tau = 0.82007$, making it the third-noisiest attribute, while attribute 9 has $\tau = 0.84752$, thereby staying as the cleanest attribute. Note, however, that $\tau_{89-5}(3) = 0.84684$, so attribute 9 is only slightly cleaner than attribute 3. When 10% noise is injected, attributes 8 and 9 become the two noisiest attributes in the dataset, with $\tau(8) = 0.80206$ and $\tau(9) = 0.82109$. The two cleanest attributes in the dataset are now attributes 3 and 5. Attribute 6, which was the noisiest attribute, is now ranked as the third noisiest with $\tau(6) = 0.82271$. When 20% noise is injected, the rankings remain exactly the same as the dataset with 10% noise. We now have $\tau_{89-20}(8) = 0.76444$ and $\tau_{89-20}(9) = 0.78245$.

Attributes 4 and 5 are injected with 5%, 10%, and 20% noise, and the results are displayed in Table VIII. Before noise injection, attributes 4 and 5 had noise rankings of 5 and 6, implying in a relative sense that both attributes are neither very clean nor very noisy. From Table III, $\tau_{\mathrm{SIM}}(4) = 0.82030$ and $\tau_{\mathrm{SIM}}(5) = 0.83263$ and after 5% noise injection, $\tau_{45-5}(4) = 0.80226$ and $\tau_{45-5}(5) = 0.81046$, making attributes 4 and 5 the

TABLE IX
NOISE RANKING, ARTIFICIAL DATASET 389

	Dataset 389-5		Dataset 389-10		Dataset 389-20	
	τ	ρ	τ	ρ	τ	ρ
1	0.84153	6	0.85168	7	0.84527	7
2	0.82364	2	0.83725	5	0.83669	6
3	0.82730	3	0.80660	1	0.77088	1
4	0.84410	7	0.85337	8	0.84664	8
5	0.84994	8	0.86054	9	0.85404	9
6	0.82052	1	0.83158	4	0.82558	4
7	0.83805	5	0.84451	6	0.83666	5
8	0.82985	4	0.80984	2	0.77109	2
9	0.85413	9	0.82856	3	0.79265	3

noisiest and fourth-noisiest attributes, respectively. With 10% and 20% noise, attributes 4 and 5 are ranked as the two noisiest attributes in the dataset.

Noise was also injected into attributes 2 and 6, which are the two noisiest attributes in the simulated dataset before noise injection, as well as for attributes 6 and 9, the noisiest and cleanest attributes in the simulated dataset. A detailed description of the results is omitted due to space limitations, but are instead summarized here. After 5% noise injection, $\tau(6)$ increases slightly to 0.79923, and then drops to 0.78329 with 10% noise and 0.76195 with 20% noise. For attribute 2, τ changes from 0.79045 with 5% noise, 0.78369 with 10% noise, and 0.76255 with 20% noise. Regardless of the amount of noise injected, attributes 2 and 6 remain as the two noisiest attributes in the dataset.

When attributes 6 and 9 are injected with 5% noise, $\tau(6) = 0.79798$ and $\tau(9) = 0.85134$. Attribute 6 stays as the noisiest attribute, while attribute 9 switches places with attribute 8, which now becomes the cleanest attribute. When 10% noise is injected, $\tau(6) = 0.77842$ and $\tau(9) = 0.81994$, making attributes 6 and 9 the noisiest attributes. This trend continues when 20% noise is injected, with $\tau(6) = 0.76419$ and $\tau(9) = 0.79897$.

3) Injected Noise in Three Attributes: In the final case study using simulated data, we examined the results when three attributes are independently corrupted in a dataset. A total of 4 datasets were created, based on corrupting the following sets of attributes. We corrupted attributes 1, 4, and 5 in the first case, 3, 8, and 9 in the second case, 7, 2, and 6 in the third case, and 4, 6, and 9 in the final case. The second and third cases represent the cleanest and noisiest attributes, respectively, while the final case covers one clean attribute (attribute 9), one noisy attribute (attribute 6), and one attribute that is neither clean nor noisy in a relative sense. The first case, with attributes 1, 4, and 5 corrupted, is the case where three attributes that are neither clean nor dirty are injected with noise.

Table IX displays the results when attributes 3, 8, and 9 are each corrupted with 5%, 10%, and 20% noise. Before any noise injection is undertaken, these three attributes are the cleanest, in a relative sense, with attribute 9 being the cleanest attribute. We would anticipate that noise injection would make each of these attributes noisier, and hence lower the rankings. For 5% noise, $\rho(3) = 3$, $\rho(8) = 4$, and $\rho(9) = 9$. In all three cases, injecting 50 noisy instances has resulted in making each of the attributes more noisy. Even though attribute 9 remains the cleanest attribute, given that $\tau(9) = 0.85413$ and $\tau(5) = 0.84994$, the gap between the cleanest and second cleanest attributes (attribute 5 in the case of 5% noise) has been reduced significantly. With 10% noise attributes 3, 8, and 9 become the three noisiest attributes. Attributes 5 and 4 are now the two cleanest attributes. The same exact noise ranking is produced when 20% noise is injected, with the only change being that the gap between the three noisiest attributes, 3, 8, and 9, and the fourth most noisy attribute, 6, widens with respect to τ. Similar results were observed in the other datasets with three noisy attributes.

B. Case Study with Real-World Datasets

1) JM1-8850 Dataset: As the effectiveness of our procedure was objectively evaluated using simulated data, we now consider the JM1-8850 dataset, a real-world software measurement dataset known from our previous experiments and from expert inspection to contain noise in a number of attributes. For JM1-8850, PANDA is executed 11 times with 5, 10, 15, 20, 25, 30, 40, 50, 60, 75, and 100 bins. Table X displays Kendall's Tau correlation and distributional statistics for this dataset. The class attribute has the largest median difference (384), the largest average difference (533.66), and the largest standard deviation (678.38). The class attribute also has the lowest Kendall's Tau correlation to the ranking obtained with all 14 attributes present in the dataset, with a value of 0.84982. Based on our technique, therefore, we conclude that the class attribute has the most noise among all 14 attributes in the JM1-8850 dataset.

Attribute *uoper* has a Kendall's Tau correlation of 0.88452, making *uoper* the second most noisy attribute in JM1-8850, followed by *uopan* and *bloc*, with $\tau = 0.89499$ and 0.89943 respectively. Alternatively, attributes *ccomp*, *bcnt*, and *tloc* are the three cleanest attributes with a value of $\tau = 0.93950$, 0.93823, and 0.93792 respectively.

Inspection by a software engineering expert is used to verify the accuracy of the results obtained from executing our technique for the detection of noisy attributes on JM1-8850, as it is not clear a priori which attributes are clean and which are noisy. Expert input is not needed within the noise detection algorithm itself and is used only for the evaluation of the output of our technique. Inspection was carried out by an expert with over 15 years of experience in the software engineering domain. The expert calculated numerous summary statistics for each of the attributes, and compared the values of correlated attributes to one another to determine if some particular attribute values were not reasonable given the values of the other attributes. Other domain-specific analysis was performed to evaluate the noise in the datasets, which we cannot include due to space limitations.

The expert determined that the class attribute does indeed contain the most noise of any attribute in the dataset, and attributes *uoper*, *uopan*, and *bloc* contain more noise relative the other attributes in the dataset. The fact that the dependent variable was determined by our algorithm to be the noisiest corresponds well with the domain-specific nature of the data generation process. In many software quality datasets, module fault information is provided through manual procedures by the team that developed the software, leading to two significant reasons why this

TABLE X
DIFFERENCES IN RANKING FOR JM1-8850

Attribute	Min	10%	25%	Median	75%	90%	Max	Mean	Standard Deviation	Kendall's Tau	Rank
bcnt	0	21	61	149	275	411	2490	198.65	202.26	0.93823	13
tloc	0	16	51	135	270	441	2514	195.53	208.64	0.93792	12
eloc	0	18	53	136	270	448	2803	203.31	233.46	0.93507	11
cloc	0	30	100	223	384	655	4879	322.24	403.68	0.89977	5
bloc	0	26	76	215	410	694	4695	311.62	360.34	0.89943	4
toper	0	18	53	140	274	460	3080	202.78	221.36	0.93459	10
topan	0	18	53	144	294	501	3099	216.82	243.33	0.92993	9
uoper	0	35	107	270	501	808	4837	370.35	383.81	0.88452	2
uopan	0	32	95	238	453	723	4425	334.10	358.21	0.89499	3
ccomp	0	20	60	145	265	403	2482	194.20	199.30	0.93950	14
ecomp	0	37	101	226	375	571	3398	305.82	338.30	0.90990	8
dcomp	0	32	94	210	388	613	2691	291.40	299.01	0.90980	7
ccloc	0	27	82	204	325	431	7565	305.09	513.31	0.90871	6
class	0	38	150	384	608	882	5198	533.66	678.38	0.84982	1

TABLE XI
DIFFERENCES IN RANK DISTRIBUTION FOR CCCS

Attribute	Min	10%	25%	Median	75%	90%	Max	Mean	Standard Deviation	Kendall's Tau	Rank
uoper	0	2	6	16	26	41	126	19.97	20.16	0.80389	3
toper	0	2	5	10	19	30	102	14.74	15.00	0.85422	6
uopan	0	1	2	6	12	20	46	8.63	7.88	0.91636	9
topan	0	1	2	6	12	21	50	8.89	9.09	0.91227	8
ccomp	0	2	5	13	22	35	141	17.15	17.85	0.83186	5
logop	0	3	9	18	26	41	123	21.89	21.76	0.80162	2
tloc	0	2	6	15	26	42	88	18.67	16.56	0.81929	4
eloc	0	1	4	9	16	25	64	11.77	11.10	0.88410	7
nfaults	0	2	7	18	29	48	181	23.38	25.77	0.77547	1

attribute might be noisy. First, there is a strong incentive to misrepresent the fault-proneness of modules if the team is evaluated based on the quality of the software. Second, the data are often manually gathered and recorded, and hence unintentional transcribing errors can occur. Code metrics, on the other hand, are often generated by software tools and are therefore less likely to have severe data quality issues. Conversely, *ccomp*, *bcnt*, and *tloc* are the relatively cleanest attributes in JM1-8850. If this dataset was going to be used to develop a classification model, the expert strongly recommended that the class attribute undergo cleansing prior to model development to remove or correct noise that may adversely impact the training process. The expert also recommended filtering *uoper* and *uopan* from the dataset due to the high level of noise.

In order to test the robustness of our technique on real-world data, the class attribute was removed from JM1-8850 and our procedure was executed with only the remaining 13 attributes. No additional tables are included for space reasons but the results are summarized here. Attribute *uoper* becomes the most noisy attribute with $\tau(uoper) = 0.87127$. When all 14 attributes are considered, *uoper* was the second most noisy attribute after the class, with $\tau(uoper) = 0.88452$. In a similar manner, *uopan* becomes the second noisiest attribute. The remaining attributes with the exception of *bcnt*, *eloc*, and *ccomp* have their noise ranking reduced by one place when the class attribute is removed. *bcnt* and *ccomp* each drop by two places, while the noise ranking of *eloc* increases by one. This is not a cause for concern, however, as Kendall's Tau correlation for these three attributes is very close in both cases. Along with *tloc*, these attributes should all be considered relatively clean, which was confirmed by expert inspection. These results are satisfying as they demonstrate the robustness of our technique on real-world data and confirm the stability of the attribute noise ranking.

2) CCCS Dataset: For the CCCS dataset, which has only 282 instances, 5, 7, 9, 12, 15, 17, and 20 bins are used, with the final ranking calculated as the median ranking over the seven different bin sizes. Results for the CCCS dataset are displayed in Table XI. Attribute *nfaults*, representing the number of faults in the software module, was ranked as the most noisy of the nine attributes available with $\tau = 0.77547$. The number of logical operators, *logop*, was determined to be another noisy attribute with $\tau = 0.80162$. Attribute *uopan*, with $\tau = 0.91636$, was determined to be the cleanest attribute in the CCCS dataset. This is in contrast to the JM1-8850 dataset, where *uopan* is considered one of the most noisy attributes. Noise is generally a characteristic specific to a dataset or application domain, and it is possible that a noisy attribute in one dataset can be clean in another.

In a manner similar to the JM1-8850 dataset, an expert is needed to judge the effectiveness of our technique since the relative quality of each attribute is unknown before inspection. The results obtained for the CCCS dataset match with what is obtained by expert inspection of the dataset. More specifically, the expert has acknowledged that *nfaults* is indeed the noisiest attribute in the CCCS dataset, and that *uopan* and *topan* are relatively clean attributes. These results provide further evidence

that our technique is providing valuable information regarding the relative amount of noise in the attributes of a given dataset.

V. Conclusion

Techniques that provide information on quality of data can help a practitioner better understand and treat noisy data. The proposed methodology provides insight into the quality of attributes in a given dataset by obtaining an attribute ranking from most to least noisy. Supplied with this knowledge, a user of this technique can apply various treatments depending on the data mining application.

In a classification setting, noisy attributes that have a weak correlation with the class label could be eliminated, while noisy attributes with a high correlation with the class label may be subjected to cleansing treatments. This knowledge may also be used to identify and correct the source of noise during the data collection process.

The proposed methodology investigated in this study for ranking attributes based on the amount of noise they contain uses in part our recently proposed technique for the detection of noisy instances called PANDA. A comprehensive empirical investigation using both real-world and simulated datasets provided conclusive evidence toward the effectiveness and promise of the proposed approach.

The two real-world datasets are software measurement datasets from high-assurance software projects, while the simulated dataset was derived from a multivariate Gaussian distribution. The performance on the real-world datasets was based on inspection by a domain expert of the obtained noise-ranking of attributes. With both real-world datasets the results obtained from our technique matched very well with the observations of the software engineering expert. In contrast the performance on the simulated datasets was gauged by the detection of injected noise into the attribute(s). A total of 39 different datasets were derived from the simulated dataset. The results observed in the simulated data case studies corresponded very well with what was expected.

Many additional empirical investigations on a variety of both real-world and simulated datasets were also performed. The results, however, have been omitted for space considerations and similarity of empirical conclusions. Based on results of many empirical studies, we conclude that the proposed technique for ranking noisy attributes exhibits a high degree of accuracy and stability. Future work will evaluate our attribute noise ranking methodology using additional datasets.

References

[1] C. E. Brodley and M. A. Friedl, "Identifying and eliminating mislabeled training instances," in *Proc. 13th Nat. Conf. Artif. Intell.*, Portland, OR: AAAI Press, 1996, pp. 799–805.

[2] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *J. Artif. Intell. Res.*, vol. 11, pp. 131–167, 1999.

[3] W. J. Conover, *Practical Nonparametric Statistics*, 2nd ed. New York: Wiley, 1971.

[4] M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, no. 3, pp. 131–156, 1997. [Online]. Available: citeseer.ist.psu.edu/dash97feature.html

[5] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd ed. Boston, MA: PWS-Kent ITP, 1997.

[6] D. Gamberger, N. Lavrač, and S. Džeroski, "Noise elimination in inductive concept learning: A case study in medical diagnosis," in *Proc. 7th Int. Workshop Algorithmic Learning Theory*. Berlin, Germany: Springer-Verlag, 1999, pp. 199–212.

[7] M. A. Hernandez and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data Mining Knowl. Discov.*, vol. 2, no. 1, pp. 9–37, 1998.

[8] E. S. Keeping, *Introduction to Statistical Inference*. New York: Dover, 1995.

[9] T. M. Khoshgoftaar and E. B. Allen, "Classification of fault-prone software modules: Prior probabilities, costs and model evaluation," *Empirical Softw. Eng.*, vol. 3, pp. 275–298, 1998.

[10] T. M. Khoshgoftaar and N. Seliya, "The necessity of assuring quality in software measurement data," in *Proc. 10th Int. Softw. Metrics Symp.*. Chicago, IL: IEEE Computer Society, Sep. 2004, pp. 119–130.

[11] T. M. Khoshgoftaar and J. Van Hulse, "Empirical case studies in attribute noise detection," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Las Vegas, NV, Aug. 2005, pp. 211–216.

[12] T. M. Khoshgoftaar, S. Zhong, and V. Joshi, "Enhancing software quality estimation using ensemble-classifier based noise filtering," *Intell. Data Anal.: Int. J.*, vol. 9, no. 1, pp. 3–27, 2005.

[13] M. Lee, H. Lu, T. Ling, and Y. Ko, "Cleansing data for mining and warehousing," in *Proc. 10th Int. Conf. Database Exp. Syst. Appl. (DEXA)*, Aug. 1999, pp. 751–760.

[14] P. M. Murphy and D. W. Aha. (1998). "UCI repository of machine learning databases," Univ. California, Dept. Inf. Comput. Sci. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[15] E. Rahm and H. Do, "Data cleaning: Problems and current approaches," *Bull. Tech. Committee Data Eng.*, vol. 23, no. 4, pp. 3–13, 2000.

[16] SAS Institute, *SAS/STAT User's Guide*. Cary, NC: SAS Institute Inc., 2004.

[17] C. M. Teng, "Correcting noisy data," in *Proc. 6th Int. Conf. Mach. Learning (ICML 1999)*, pp. 239–248.

[18] J. Van Hulse, T. M. Khoshgoftaar, and H. Huang, "The pairwise attribute noise detection algorithm," *Knowl. Inf. Syst. J., Spec. Issue Mining Low Qual. Data*, vol. 11, no. 2, pp. 171–190, 2007.

[19] X. Zhu and X. Wu, "Class noise vs attribute noise: A quantitative study of their impacts," *Artif. Intell. Rev.*, vol. 22, no. 3–4, pp. 177–210, Nov. 2004.

[20] X. Zhu and X. Wu, "Cost-guided class noise handling for effective cost-sensitive learning," in *Proc. 4th IEEE Int. Conf. Data Mining (ICDM 2004)*, Nov., pp. 297–304.

Authors' photographs and biographies not available at the time of publication.