Identifying dominant models when the noise context is known

Cèsar Ferri, José Hernández-Orallo, Adolfo Martínez-Usó, and M.José Ramírez-Quintana

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
Camí de Vera s/n, 46022, València, Spain
{cferri,jorallo,admarus,mramirez}@dsic.upv.es

Abstract. In many machine learning applications the quality of features during deployment is worse than during training, as more effort and care is usually invested for the purpose of training a good model. In this paper, we consider the situation when this degradation (or noise) depends on a context whose value can be known during deployment. In this case, we can anticipate how a model will behave during deployment for different noise contexts. We introduce context plots for this, with the error on the y-axis and the level of noise on the x-axis. From these plots we can determine the dominance regions for a set of models and discard those models that are not optimal for any noise context. We perform some experiments on several classification and regression problems that show that the dominance regions can be well identified, leading to better decisions than those if the best model (without noise) is used.

Keywords: Attribute noise, noise context, operating conditions, classification, regression.

1 Introduction

One of the most important factors for the quality of a machine learning model is the quality of the training data. Consequently, great care and effort is put in gathering a set of training data with reliable features. It is also very common that training is performed in somewhat 'idealistic' conditions, where the features have been carefully measured. During deployment, however, things are usually very different. Data can appear in real-time and attributes can be measured with less care or with measurement devices or tests in a less idealistic context than training. The degradation in the quality of features is usually known as *attribute noise*. It is not uncommon to be able to determine what causes this degradation and even quantify the degree of noise for each attribute. For instance, some sensors can have error measurement intervals that may vary according to environmental conditions, such as wind, temperature, etc. Similarly, some attributes can be the average of a repeated test measurement, with fewer repetitions during deployment than in training. Other attributes can be contaminated by the

characteristics of each individual, as in biological or physical tests. Examples in this sense are common in real-world applications such as medical diagnosis [6], semantic place prediction [14], social surveys/computing [19], ambient assisted living [25], etc. Let us illustrate this with an example.

Example 1. Let us consider, for instance, an alarm system that has to be triggered from the data captured by several motion detectors on different locations. Two models A and B have been trained under ideal conditions, where some meteorological variables, such as temperature, were in a good operating range for the detectors (e.g., the interval $[5, 30]$ degrees Celsius). The evaluation of the models has concluded that model A is slightly better than B in this context. Let us now suppose that the system has to be deployed. Some extreme temperatures can introduce different levels of noise for the detectors according to the manufacturer specification. For instance, $[0, 30]$ degrees ensures a noise level = 5%, each degree below 0 implies 0.5 units of additional error and each degree above 30 implies 1 unit of additional error. As it is easy to install a thermometer, we know the temperature anytime. This means that we can check the expected level of noise for that temperature looking at the sensor specification. As a result, whenever the model is applied to the incoming data, we can also use the temperature as contextual information, which gives us key information about the degree of noise that is expected from the motion detectors. But is this information useful to make better decisions? For instance, if the temperature is 40 degrees Celsius, which model should I use, A or B?

In order to answer this question we proceed as follows:

1. We need to know the context, which in the above example is given by the temperature.
2. From this context, we must be able to infer a noise distribution. In the above example this is given by looking at the sensor specification error for different temperature operating conditions. This is known as the noise setting.
3. As we will introduce in this paper, we need to evaluate models A and B with different levels of *simulated* noise. This can be done during the training stage, by adding noise to the input attributes of a validation dataset according to the noise setting.
4. We then compare the error of models A and B for all levels of noise and determine the dominance regions, where one model is better than the other.
5. Finally, during deployment, we record the temperature, derive the noise level from it and apply the model that we determined that performs best (i.e., dominates) for that region.

Imagine that in step 4 we may have determined that model A is better than B for noise levels $< 7.2\%$ but B is better for noise levels $\geq 7.2\%$. Now we can answer the question: if the temperature is 40 degrees we expect a noise level of 10%, so model B is preferable in this context.

In this paper we are not interested in a general appraisal of which machine learning methods are most robust to noise, or how models degrade with noise.

All this has been done to a greater or lesser extent in the literature, as we will see in section 2. What we propose in this paper is a procedure where we can use any set of off-the-shelf machine learning methods to train several models and being able to determine which models are best and in which contexts. In order to do this we introduce context plots where noise is shown on the x-axis and error is shown on the y-axis. Each model is represented by a curve that can dominate the rest in some regions (or none). The main question we try to answer in this paper is whether the dominance regions that are identified with a validation dataset and simulated noise after training will translate to the actual dominance regions in deployment. We perform a series of experiments for classification and regression problems to answer this question affirmatively.

The rest of the paper is organised as follows. Section 2 reviews some previous works dealing with learning in the presence of noise. Section 3 formalises the notion of noise context, derives context plots for them and the concept of dominance. Section 4 presents the experimental methodology to analyse whether dominance regions can be well identified and compares three approaches: the best model without noise, the model with lowest overall error for the whole range of contexts and the procedure that takes the dominant model for each region. Finally, section 5 closes the paper with some conclusions and future work.

2 The effect of attribute noise

Many works claim about the ubiquity of noise [12] (for instance, about the 50% of the data sets in the UCI repository have missing values) and the need of implementing techniques to eliminate or reduce noise consequences [16,28]. Likewise, the whole research community agrees about the critical consequences of noise and its potential impact on the performances of induced classifiers and regressors [5,7].

In the past, there have been many approaches that worked on determine which is the real influence of the noise found in the input data. Many of these approaches addressed the problem from the point of view of the techniques, so studying the robustness and efficiency of the learning methods when they are applied to noisy or imperfect data [26,13,27]. A second group of approaches focused on cleaning the data by reducing or minimising the level of noise before model learning and/or model application [24,10,2]. In addition, efforts to study the noise impact have been more concentrated in classification models [17,21] rather than in regression models [3] and in label noise [29,15] rather than attribute noise [11], although there are studies that address both of them: the attribute and the label noise [28,23].

In [8], Hickey defines noise as anything that obscures the relationship between description and class, distinguishing three major sources of noise: 1) insufficient description, 2) corrupted attributes and 3) erroneous classification, although some works also consider outliers as noise [6,20]. From these categories, types 1 and 2 are often referred as attribute noise, i.e., some attribute values are corrupted (erroneous or incomplete) or missing (no value at all). Type 3 are also

known as class noise or misclassification/mislabelling errors, being this type the most studied case in the literature [5].

The topics studied in this paper, to the best of our knowledge, are not well covered in the literature. On the one hand, we are interested in studying those methods that work better according to a given context: the level of noise. In [18], the author analysed the performance (in terms of predictive accuracy) of several well-known classification techniques for different levels of noise in the training and test sets (assuming that the example distribution does not change from training to test), concluding that one of the techniques was the most robust for levels of noise greater than 10%. On the other hand, we consider that noise is only present in the attributes of the test set. In this sense, authors in [13] studied the effect of attribute noise in classification by focusing on the asymmetric case when the level of attribute noise in the training and test datasets is different. The authors give evidence that, in general, a high level of noise in training data and low level of noise in test should be avoided, whereas the opposite case has less effect on the performance of the algorithms. Nevertheless, the authors also conclude that given the interaction among the algorithms, noise and dataset size, any general conclusion cannot be safely applied to a particular problem. In other words, without more information about the problem, each particular technique and level of noise will require of a specific study. This individual study of each particular problem instead of a general approach is what we undertake in this paper.

3 Noise Context, Context Plots and Dominance

Operating conditions have been traditionally applied to sensors, such that measurements are more or less robust depending on different ranges of contexts such as temperature, pressure, acceleration/gravity, windspeed, etc. As in our introductory example, sensors have a narrow operating range provided by the manufacturer in which we consider that the errors made by the sensor are negligible or below a small level. Out of this range, the error increases, usually being more critical depending on how far from the operating range we are.

We here define the notion of *noise context* as follows:

Definition 1. *The noise context θ is a set of measurable parameters that has affect over attribute noise.*

For instance, in the introductory example, $\theta = 40$. Another example of context that may affect noise is the number of repetitions of a measurement. For instance, if for each example the attribute 'blood pressure' is measured 5 times and averaged before recording it into the dataset, then we have that 5 is the context. If for a different patient or dataset this is just measured 3 times we have a different measuring context that will imply more noise.

The following definition determines how we can map a noise context into a noise level.

Definition 2. *A noise level mapping is a function μ that translates contexts into noise levels, $\mu(\theta) = \nu$.*

For instance, in the introductory example, $\mu(\theta) = 5$ iff $\theta \in [0, 30]$, $\mu(\theta) = 5 + 0.5|\theta|$ iff $\theta < 0$ and $\mu(\theta) = 5 + (\theta - 30)$ iff $\theta > 30$. In the blood pressure example, the noise level could be obtained as $\mu(\theta) = 1/\theta$, where θ is the number of measurements.

Finally, we need to determine how we translate a noise level for an attribute into noise for that attribute. This is known as the noise setting:

Definition 3. *A noise setting is any distribution that can be defined from a value x_i of an attribute i, a noise level ν and optionally some statistics of attribute i.*

For convenience, we will assume that no noise is generated (i.e., $x_i \sim x_i$) if $\nu = 0$. If we have more than one attribute with noise, we need to define a distribution for each of them. For instance, for the motion detector, as it is a nominal attribute, a possible noise setting could be a distribution that swaps x_i (from true to false and vice versa) with probability $\nu/100$. If the noise level is 10%, this means that the true value of x_i is swapped with probability 0.1. In the blood pressure case, as it is a numerical attribute, a normal distribution could be used.

It is important to note here that the noise setting depends on the problem and has to be determined from the meaning of the context, the mapping and the problem itself.

Given a dataset D we denote by D_ν the same dataset where we have applied noise level ν using the noise setting of the problem. Clearly $D_0 = D$.

Let us define the error of applying a model m to a dataset D as $q(m, D)$. From here we define the expected error as follows:

Definition 4. *The expected error of a model m, dataset D and noise level ν, $Q(m, D, \nu)$, is the expected value of $q(m, D_\nu)$, where D_ν is any dataset where we have applied ν.*

Clearly $Q(m, D, 0) = q(m, D)$. Thus, we are now able to represent how the expected error function Q behaves for a range of noise levels.

Definition 5. *For a model m and dataset D, a noise context plot shows the expected error $Q(m, D, \nu)$ on the y-axis and noise level ν on the x-axis.*

Figure 1 shows the behaviour of several models in a context plot for a classification problem using the dataset *Vehicle Silhouettes* from [1]. The y-axis represents the cost function Q which is the expected error for each level of noise. In this example, the level of noise is determined by the operating context.

Noise context plots are useful to identify dominance regions. Complete details about the methodology for generating these plots can be found in Section 4.

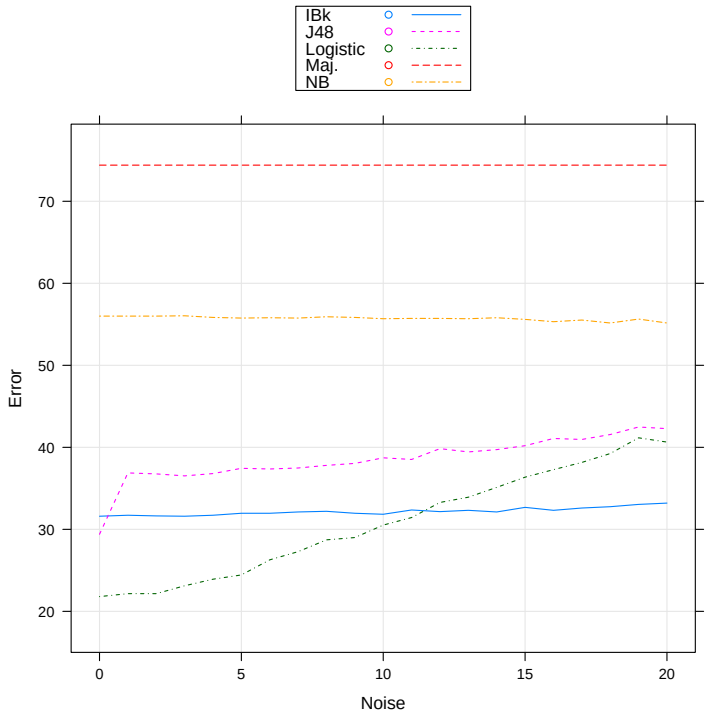

Fig. 1. Example of context plot for the Vehicle Silhouettes dataset, with the expected error Q on the y-axis for four classification methods and the noise level ν on the x-axis.

Definition 6. *Given a dataset D, a model m_1 is said to dominate model m_2 in a noise level interval $[\nu_0, \nu_1]$ iff $\forall \nu \in [\nu_0, \nu_1]$ we have that $Q(m_1, D, \nu) \leq Q(m_2, D, \nu)$.*

In Figure 1 we see different dominance regions. Some models are dominated by at least another model for any noise level and can safely be discarded, such as J48 and NB. We also see that the Logistic model dominates from noise levels $\nu \in [0, 12.7]$ while IBk dominates for noise levels $\nu \in [12.7, 20]$.

Now the question is, can we determine the dominance regions during training? We propose a simple yet very useful procedure, known as **ValNoiseOpt**, summarised as follows:

1. Given some labelled dataset, we divide it into training and validation subsets and use the former for training several models.
2. Different levels of noise according to the operating context range and the mapping μ are added to the validation data following the noise setting. This generates several validation datasets. Models are then evaluated on the noisy validation datasets.

3. A context plot is drawn with all the models. The dominance regions are identified. The models that do not have any dominance region are ruled out.
4. In deployment time and with different data, the operating context is mapped to a noise level and the best model for that noise level is applied.

It is worth noting that rather than knowing which methods work better under different levels of noise, the important point here is selecting the intervals of dominance from step 3 that better works in step 4.

4 Experiments and results

In this section we analyse whether the procedure introduced above (**ValNoiseOpt**) is effective for choosing the optimal model for each noise level.

We will work directly with noise levels, assuming that they are mapped from contexts. We need to define a noise setting, i.e., a way of generating noise that is representative and general, so that the experimental results can be extrapolated to other noise settings. In the literature, there is a large number of approaches for adding artificial noise to datasets [22,4,5,23]. Noise is generated randomly by using some well-known probability distributions [24,29]. Instances are selected by changing their values into a range of possible values. How this range of possible values is defined depends on whether the attribute is nominal or numerical.

In order to inject noise into the features, we follow a similar procedure as the ones mentioned above. We consider two different noise settings depending on the type of the feature.

- **Numerical attributes**: In the case that we want to inject a level of noise ν into a numerical attribute, we proceed as follows. First we estimate the standard deviation σ of all values of this attribute. We modify a value x in the following way: $x' \sim N(x, \sigma \cdot \nu)$, i.e., following a normal distribution using x as mean and σ multiplied by the noise level ν as standard deviation.
- **Nominal attributes**: In this case we estimate the frequency of each of the possible values of the categorical attribute, e.g., if there are 3 possible values (a, b, c) we estimate the array of frequencies $\boldsymbol{p} = (p_a, p_b, p_c)$ where, obviously, $p_a + p_b + p_c = 1$. Consider we have an instance of value $x = b$ for that attribute, we estimate an array $\boldsymbol{t}$ with the following scores $t_a = 0, t_b = 1, t_c = 0$. If we want to insert a noise level ν, we calculate $\alpha = 1 - e^{(-\nu)}$. Then we compute an array of probabilities in the following way $\boldsymbol{p'} = \alpha \cdot \boldsymbol{p} + (1-\alpha) \cdot \boldsymbol{t}$. Finally, we use the $\boldsymbol{p'}$ in order to sample the new value x' of the attribute.

In order to measure the impact of noise over the performance of the models, we are going to employ the following measures. For classification problems, we will use error measured as the percentage of instances that are not correctly predicted. Regarding regression, in order to avoid the problem of comparing different magnitudes in the error of the predicted values of the different datasets, we use relative absolute error. This measure takes the total absolute error and normalises it by the total absolute error of a simple predictor, in this case the simple predictor is the mean computed in the train dataset.

Figure 2 and Figure 3 show examples of the performance of several methods for classification and regression when we introduce some levels of noise into the test data. The noise is generated using the previous approach. We use the *credit-g* dataset as a classification example and *abalone* dataset for the regression example, both from the UCI repository [1]. This dataset is split into 50% train and 50% test. We inject noise from 0 to 20% increasing 1% each step. In order to reduce the variance of results, we repeat 5 times the partition of data into train and test. For each partition we replicate 5 times the noise injection process. With this, we show the average results of 25 executions for different learning methods.

For the learning techniques, we use the *RWeka* [9] library from *R*. Concretely, for classification problems, we use the following techniques: a decision tree method (J48), Naïve Bayes, Logistic Regression and kNN (IBk). We include as a reference method a simple model that always returns the majority class. This simple technique is not affected by noise at attributes since it does not consider them in order to classify new instances. With regards regression problems, we use the following learning methods: Linear Regression, a decision tree for regression (M5P), kNN (IBk), a Support Vector Machines method for regression (SMOreg). We also include the ZeroR method as a reference in the regression example. This method returns for all examples the mean of the train dataset, and for that reason it is not affected by the injected noise.

There are interesting aspects that we can observe in Figure 2 and Figure 3. Figure 2 shows how noise degrades the performance of models in a classification problem. In fact, IBk and J48 obtain worse results than the majority model with even small values of ν. In this example, for almost all values of noise level ν, Logistic is the best method while Naïve Bayes is very close in performance. Figure 3 includes a regression example. Here, we can clearly see how noise level is affecting more to some techniques to others. We also see that the M5P model dominates from noise levels $\nu \in [0, 6.3]$ while SMOreg dominates for noise levels $\nu \in [6.3, 20]$.

In order to study different procedures of selecting models when noise context in test data is known, we performed a set of experiments over 12 datasets of the UCI repository [1], 6 regression problems and 6 classification problems. These datasets are shown in Table 1. For each dataset, we split the data into 3 different sets: Training (50%), Validation (25%) and Test (25%). In order to reduce the variance of results, we repeat 5 times the partition of data. For each partition we replicate 5 times the noise injection process. With this, we show the average results of 25 executions for different learning methods. We inject noise from 0 to 20% increasing 1% each step. We use the same learning methods that in the Figures 2 and 3 except from the baseline methods.

We use the training data for learning the models and the validation data to estimate the performance of the models when we inject noise. Then, we use the remaining test set as the deployment dataset. We consider the following procedures:

– **ValNoNoise**: For all the estimated values of noise, we select the method that obtains the best performance without noise.

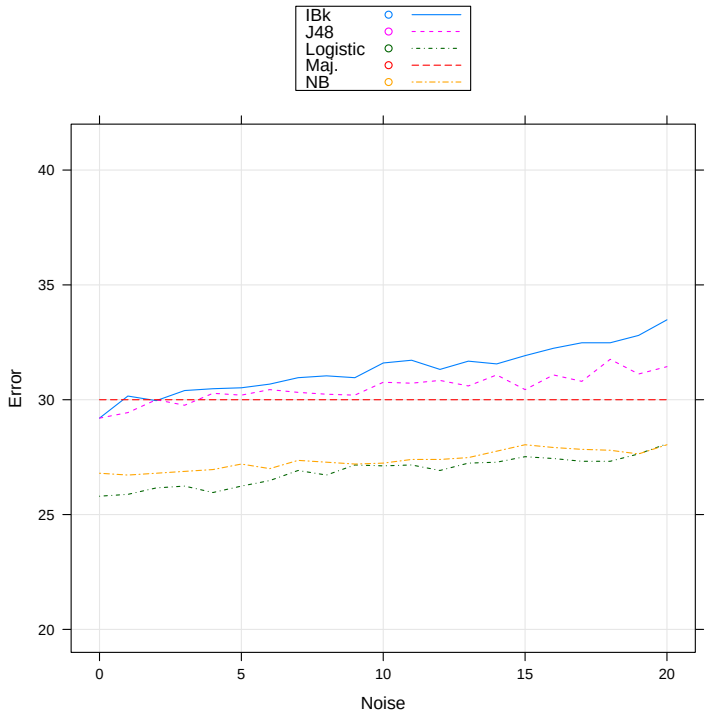

Fig. 2. Example of context plot for a classification problem (Creditg dataset), with the expected error Q on the y-axis (percentage of misclassified instances) for four classification methods and a reference model (majority class) for different noise level ν on the x-axis.

- **ValBestArea**: For all the estimated values of noise, we select the method that obtains the best performance in the validation dataset by averaging all noise levels (i.e., the curve with lowest area in the context plot).
- **ValNoiseOpt**: For each of the estimated values of noise, we select the method with best performance in the validation dataset with that value of noise.
- **Idealistic**: For each of the estimated values of noise, we select the method with best performance in the test dataset with that value of noise. This strategy is not realistic (it cannot be done in practice). We just include this as a reference.

Our question was whether the **ValNoiseOpt** procedure was able to determine the dominance regions such that if we apply them during deployment, we still select the best possible method in general. Hence, we compare **ValNoiseOpt** with the two first procedures. Note that **ValBestArea** is also a new procedure

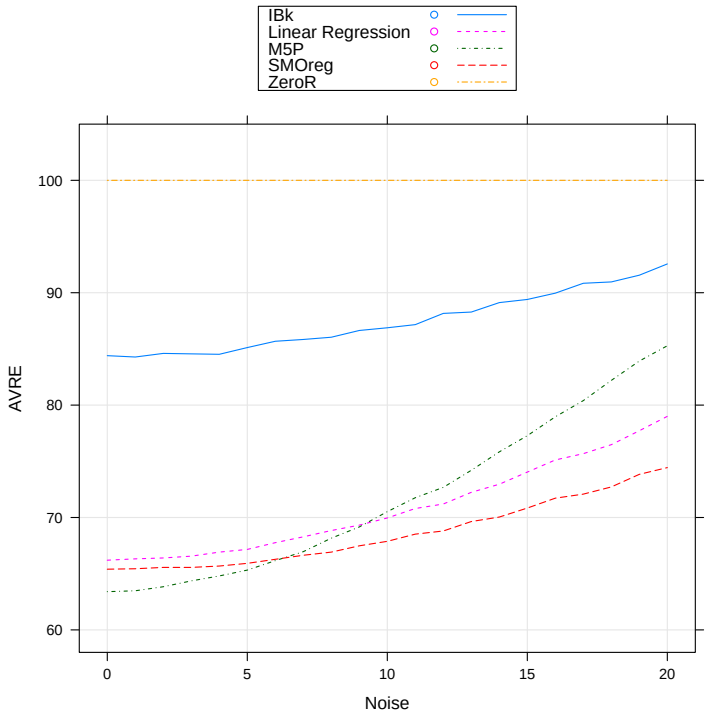

Fig. 3. Example of context plot for a regression problem (Abalone dataset), with the expected error Q on the y-axis (relative absolute error) for four regression methods and a reference model (average of the training set) for different noise level ν on the x-axis.

that we introduce in this paper, although we expect **ValNoiseOpt** to perform better.

In order to better compare regression and classification datasets we normalise the performance results by the **Idealistic** performance, i.e., we divide the average performance of the **Idealistic** strategy (i.e. the smaller value) by the average performance of every strategy and the result is multiplied by 100. In Table 2 we can see the normalised results of the three strategies above (note that, the normalised performance of the **Idealistic** procedure is always equal to 100 for all the problems, so we have not included it in the table). As expected, the best strategy is **ValNoiseOpt** since it obtains the best performance in all datasets except from one. **ValNoNoise** and **ValBestArea** are equivalent in all datasets except for two. This reveals that in most of cases the method with the best performance without noise is also the best method when we average the full range of noise levels.

Dataset	NumInst	NumAtt	RegClas	Numclass	Mean	SD	Median	Max	Min
1 cholesterol	303	14	Reg	-	246.69	51.78	241.00	564.00	126.00
2 bodyfat	252	15	Reg	-	19.15	8.37	19.20	47.50	0.00
3 wisconsin	194	33	Reg	-	46.94	34.52	39.50	125.00	1.00
4 stock	950	10	Reg	-	46.99	6.54	46.69	62.00	34.00
5 meta	528	22	Reg	-	99.55	764.62	9.84	12041.00	0.00
6 abalone	4177	9	Reg	-	9.93	3.22	9.00	29.00	1.00
7 iris	150	5	Class	3	-	-	-	-	-
8 white-clover	63	32	Class	4	-	-	-	-	-
9 hepatitis	155	20	Class	2	-	-	-	-	-
10 breast-cancer	286	10	Class	2	-	-	-	-	-
11 vehicle	846	19	Class	4	-	-	-	-	-
12 credit-g	1000	21	Class	2	-	-	-	-	-

Table 1. Information about the datasets we employed in the experiments: Number of instances, Number of attributes, classification or regression dataset, number of classes in classification datasets. For regression datasets we include the following information about the values to predict: mean, standard deviation, median, maximum, minimum.

Dataset	ValNoNoise	ValBestArea	ValNoiseOpt
1 cholesterol	97.5468	97.5468	97.5468
2 bodyfat	99.2007	99.2007	99.9875
3 wisconsin	91.9391	91.9391	92.1545
4 stock	100.0000	100.0000	100.0000
5 meta	100.0000	100.0000	100.0000
6 abalone	81.6546	99.9358	99.9919
7 iris	89.3179	89.3179	96.4559
8 white-clover	89.9206	89.9206	88.3807
9 hepatitis	100.0000	100.0000	100.0000
10 breast-cancer	87.6362	87.6362	88.2716
11 vehicle	75.4220	96.1937	99.6372
12 credit-g	100.0000	100.0000	100.0000

Table 2. Relative performance of different selection strategies over a range of operating conditions for 6 classification datasets and 6 regression datasets.

5 Conclusions and future work

A great amount of work in the literature has been devoted to the analysis of which machine learning methods are more robust to noise and/or how to clean the training or the test data to reduce noise. In practice, however, we want to use a set of machine learning techniques from our preferred library and get several models that we need to evaluate in order to choose the best for a given deployment situation. In this paper we have analysed the case when the noise level depends on a context that we know in advance or we can infer. As a result, the model that best behaves for a similar noise level over a validation dataset is used.

This procedure requires the knowledge of the context and its mapping to a noise setting. We have discussed some examples in the introduction, but many

other situations could fall under this schema. In the problem with sensors, the time that has passed since the sensor was last calibrated can suggest an easy mapping with the quality of its measurements. In a medical domain, the quality of the data may depend on the hospital, the insurance or the physician. In a multi-platform form, the information quality may depend on the interface that is used to input the data (desktop, mobile, interview, etc.). Finally, some variables could be linked to the quality of other variables, such as the age of a user being linked to the reliability of the data s/he introduces. Overall, many examples can be found, as this is related to a well-known area in information systems known as *data quality*.

It also takes some time if we want to draw the curves for a broad range of noise levels with enough resolution. However, once this has been done, the selection and application of the best model is straightforward and the results are very close to an idealistic process.

In an extended version of this paper, we are working on different noise models, derived from scenarios with real operating conditions and with real sensors, where each attribute has a different operating range, but the context is still given by a single parameter (e.g., temperature, or the number of measurements or tests performed from where the features are averaged).

There are other more ambitious research questions that remain unsolved. If the operating context is not given we cannot obtain the noise level during deployment. In these cases, we could try to infer the noise level from the example itself, depending on the attribute values or the whole dataset (e.g., change in attribute variance or correlation between attributes, etc.). In a smart house project, we consider that if two or more sensors in the same (or nearby) locations give conflicting measurements, then this may be an indication that there might be noise. Apart from the area under the context curve, other global performance metrics could be derived from these plots or by defining a distribution over noise levels or operating contexts. This could be related to some other analysis of performance metrics in the presence of noise [4].

Acknowledgments

This work was supported by the Spanish MINECO under grants TIN 2010-21062-C02-02 and TIN 2013-45732-C4-1-P, and the REFRAME project, granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences Technologies ERA-Net (CHIST-ERA), and funded by MINECO in Spain (PCIN-2013-037). We thank Daniel Borao Bermúdez who developed part of the code employed in the experiments. Finally, we also thank Peter Flach for some suggestions about scenarios where data quality values could be available.

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml 5, 8

2. Brodley, C., Friedl, M.: Identifying mislabeled training data. Journal of Artificial Intelligence Research 9(11), 131–167 (1999) 3
3. Chen, Y., Caramanis, C.: Noisy and missing data regression: Distribution-oblivious support recovery. In: The 30th International Conference on Machine Learning (ICML 2013). vol. 28, pp. 383–391 (2013) 3
4. Ferri, C., Hernández-Orallo, J., Modroiu, R.: An experimental comparison of performance measures for classification. Pattern Recognition Letters 30(1), 27–38 (2009) 7, 12
5. Frenay, B., Verleysen, M.: Classification in the presence of label noise: A survey. Neural Networks and Learning Systems, IEEE Transactions on 25(5), 845–869 (May 2014) 3, 4, 7
6. Gamberger, D., Boskovic, R., Lavrac, N., Groselj, C.: Experiments with noise filtering in a medical domain. In: ICML. pp. 143–151 (1999) 2, 3
7. García-Laencina, P.J., Sancho-Gómez, J.L., Figueiras-Vidal, A.R.: Pattern classification with missing data: A review. Neural Comput. Appl. 19(2), 263–282 (2010) 3
8. Hickey, R.J.: Noise modelling and evaluating learning from examples. Artif. Intell. 82(1–2), 157–179 (1996) 3
9. Hornik, K., Buchta, C., Zeileis, A.: Open-source machine learning: R meets Weka. Computational Statistics 24(2), 225–232 (2009) 8
10. Lam, C.P., Stork, D.G.: Evaluating classifiers by means of test data with noisy labels. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence. pp. 513–518 (2003) 3
11. Lim, S.: Cleansing noisy city names in spatial data mining. In: Information Science and Applications (ICISA), 2010 International Conference on. pp. 1–8 (2010) 3
12. Maletic, J.I., Marcus, A.: Data cleansing: Beyond integrity analysis. In: In Proceedings of the Conference on Information Quality. pp. 200–209 (2000) 3
13. Manini, M., Yang, Y., Ryu, Y.: Classification algorithm sensitivity to training data with non representative attribute noise. Decision Support Systems 46, 743–751 (2009) 3, 4
14. Montoliu, R., Martínez-Usó, A., Sotoca, J.M.: Semantic place prediction by combining smart binary classifiers. In: Mobile Data Challenge by Nokia Workshop, Int. Conf. on Pervasive Computing (2012) 2
15. Natarajan, N., Dhillon, I., Ravikumar, P., Tewari, A.: Learning with noisy labels. In: Advances in Neural Information Processing Systems 26, pp. 1196–1204 (2013) 3
16. Nettleton, D., Orriols-Puig, A., Fornells, A.: A study of the effect of different types of noise on the precision of supervised learning techniques. Artificial Intelligence Review 33(4), 275–306 (2010) 3
17. Neville, J., Jensen, D., Gallagher, B.: Simple estimators for relational bayesian classifiers. In: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), December 19-22, 2003, Melbourne, Florida, USA. pp. 609–612. IEEE Computer Society, Washington, DC, USA (2003) 3
18. Nolan, J.: Computer systems that learn: an empirical study of the effect of noise on the performance of three classification methods. Expert Systems Application 23(1), 39–47 (2002) 4
19. Ouzienko, V., Obradovic, Z.: Imputation of missing links and attributes in longitudinal social surveys. In: Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on. pp. 957–964 (2011) 2

20. Pechenizkiy, M., Tsymbal, A., Puuronen, S., Pechenizkiy, O.: Class noise and supervised learning in medical domains: The effect of feature extraction. In: Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on. pp. 708–713 (2006) 3
21. Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann (1999) 3
22. Ripley, B.D., Hjort, N.L.: Pattern Recognition and Neural Networks. Cambridge University Press, New York, NY, USA (1995) 7
23. Sáez, J.A., Galar, M., Luengo, J., Herrera, F.: Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. Knowledge and Information Systems 38(1), 179–206 (2014) 3, 7
24. Teng, C.M.: Correcting noisy data. In: Proceedings of the Sixteenth International Conference on Machine Learning. pp. 239–248. ICML '99, Morgan Kaufmann Publishers Inc. (1999) 3, 7
25. Vacher, M., Portet, F., Fleury, A., Noury, N.: Challenges in the processing of audio channels for ambient assisted living. In: e-Health Networking Applications and Services (Healthcom), 2010 12th IEEE International Conference on. pp. 330–337 (2010) 2
26. Villeda-Ruz, R., Garcia-Garcia, J.: Meaningful error estimations for data analysis. In: Proceedings of the 2009 Mexican International Conference on Computer Science. pp. 280–288. IEEE Computer Society (2009) 3
27. Zhang, P., Zhu, X., Shi, Y., Guo, L., Wu, X.: Robust ensemble learning for mining noisy data streams. Decision Support Systems 50(2), 469 – 479 (2011) 3
28. Zhu, X., Wu, X.: Class noise vs. attribute noise: A quantitative study. Artificial Intelligence Review 22(3), 177–210 (2004) 3
29. Zhu, X., Wu, X., Chen, Q.: Eliminating class noise in large datasets. In: Proceedings of the Twentieth International Conference on Machine Learning. pp. 920–927 (2003) 3, 7