K-Nearest Neighbor's sensitivity to attribute noise, in comparison to other classification algorithms

Voulgari Eleni, Mylonaki Aggeliki, Konstantopoulou Vasiliki and Karydis Athanasios

***Abstract*— Data obtained from real-world are rarely noise free. In classification problems, this extremely affects the models created and the decisions made. Moreover, it is not always possible to perform preprocessing steps in a dataset in order to remove, especially attribute, noise. Consequently, in a data science project, where multiple models can be trained and used, it is very important to know, during the classification algorithm selection phase, which one is more robust to noise and how each one is affected. Because of the fact that class noise is commonly examined in existing research, we focus exclusively on attribute/feature noise. kNN (k-Nearest Neighbors) is a widely used, simple and effective classification algorithm, which is claimed to be sensitive to attribute noise. In this research, we present an evaluation on the effect of attribute noise to the classification accuracy of kNN compared to three other classification algorithms. Our results show that kNN is more sensitive to attribute noise and can be used to guide interested readers to determine the suitability of one of the examined algorithms, to their own "noisy" classification problem.**

I. INTRODUCTION

Nowadays, the volume and velocity of the data have been dramatically increased. Machine learning and Data Mining are gaining more and more space in the technological evolution. Thus, real world applications should be able to extract information from enormous datasets.

One of the main problems, that characterize real world machine learning applications, is that the training and testing of the models should be implemented on datasets containing noisy instances. Attribute/Feature noise as part of a dataset could be described as a) missing values, b) inconsistent entries, c) erroneous values and d) typing errors. In general, the existence of noisy instances in a dataset can dramatically affect the results of an experiment. In classification problems, the quality of the predictions can be extremely reduced.

The first step, when facing a classification problem, is a preprocessing of the data phase. During that phase, and to avoid the above consequences of attribute noise, it is possible that some entries can be removed, since they are characterized as noisy data, but possibly leading to other problems, such as the decrease in the volume of the dataset. Thus, in order to avoid deletions, we can employ data cleaning techniques, but without always being able to identify and replace the erroneous values with the correct ones [7], [8], [10].

Instinctively, we can assume that each classification algorithm reacts to noise in a different way. This may depends on the algorithm's parameters and their tuning, as well as on the basic implementation of the algorithm. Some algorithms are more sensitive to noise than others due to their way of learning as they cannot overcome the noise. Other algorithms may be more resistant to noisy instances, as they handle the dataset in other ways to complete the training procedure.

This research aims to understand the sensitivity of kNN to noisy attributes and to provide a fair comparison between kNN and other well known classification algorithms in regard to this issue.

II. K-NEAREST NEIGHBORS ALGORITHM

kNN is a supervised classification or regression algorithm. It belongs to the instance based algorithms category (lazy learners). This means that the model does not actually learn during the training procedure, but it memorizes all training instances as vectors in order to be able to compare them (measure the distance) with the test instances. Thus, the computation time of the training phase is much lower than the computation time of the testing phase. kNN is also a non-parametric algorithm, in the sense that it does not assume a theoretical distribution of the input data, which is useful in cases that a prior knowledge of the data is absent.

A. The algorithm

To begin with, it is important to describe how kNN works. Each instance of the dataset is converted into a 1-D M-size vector where M is the number of attributes. During the training phase, the algorithm memorizes each vector and its class label. One of the hyper parameters of kNN algorithm is the number of k, which represents the number of the nearest neighbors to each new instance that the algorithm has to examine in order to obtain a class label. k number is selected by the user. More specifically, in a classification problem, during the testing phase, kNN tries to predict the class label of every new instance by finding its k closest training entries, using a distance metric. As a result, the label assigned to the new instance is the most frequent label among the k neighbors [1].

B. Distance Metric

As mentioned above, kNN tries to find the closest entries' labels in order to classify the new instance. The user is also able to select the distance metric, so the algorithm can calculate the distances between all the entries, with this specific measure.

A basic distance metric, commonly used with kNN, is Minkowski distance. The general formula to compute the distance can be found in Table 1.

The Minkowski metric is induced by the L_p norm. Selecting p values as 1 or 2, means that Minkowski distance

corresponds to Manhattan distance or to Euclidean distance respectively.

In case of categorical attributes there are two approaches to compute distance. The first one makes use of Hamming distance which supports the statement that if two values of a categorical attribute are the same, their distance is zero. On the contrary, if two values of the same categorical attribute differ, their distance is 1. The second approach, is to transform categorical attributes to numerical and make use of the Minkowski metric. In order to transform the categorical attributes to numerical, a unique number is assigned to each possible categorical value.

TABLE I
DISTANCE METRICS FORMULAS

Minkowski	$D(X,Y) = \left(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right)^{1/p}$
Euclidean	$d(\mathbf{p},\mathbf{q}) = \left(\sum_{i=1}^{n} (q_i - p_i)^2\right)^{1/2}$

C. K selection

Considering kNN, an important aspect is the selection of the number of neighbors k [4], as this hyper parameter probably affects the model, in regard to noise. k represents the number of the closest entries in the dataset to the instance been investigated. The above consequence can be explained be the fact that the class label of the new instance will be assigned to it by finding the label with the maximum counts derived from the k aforementioned neighbours.

Selecting the number of k neighbors is always an interesting subject for scientific research. This is not a topic we are engaged with in this research, but we show evidence of the influence of the different values of k to how models behave when are being applied to noisy data. Consequently, as k increases, the model examines a larger number of neighbours, which could be computationally expensive, but also essential to eliminating the effect of the noise to the model. kNN models with very low values of k, will produce low accuracy if it happens to ”be around” noisy instances [9].

III. OTHER CLASSIFICATION ALGORITHMS

One of the aspects of this research is the comparison of kNN with other classification algorithms, regarding their sensitivity to attribute noise. For the experiments kNN, SVM (Support Vector Machines), Decision Trees and Naive Bayes are selected. A brief description of each algorithm follows.

SVM is a common used classification algorithm which, during training, is aiming to optimize a hyperplane, so as to be able to separate the classes in the best possible way. The algorithm was originally used for binary classification but extensions are created to deal with multi-class classification problems.

Decision Tree is a rule based classification algorithm. All the internal nodes of the tree represent conditions concerning the attributes/features and all the leaf nodes represent the class labels. The training procedure contains the creation of a decision tree, where the algorithm tries to find the next best attribute to choose for splitting the branches as well as when to stop using the data to avoid overfitting (by hyper parameter tuning).

Naive Bayes algorithm is based on Bayes Theorem and the assumption of independency between the features of a dataset. Naive Bayes algorithm calculates the posterior probability of every class and assigns the label to an instance to the class with the highest probability.

For the experiments, sklearn’s implemented models of the above algorithms have been used with default hyper parameters.

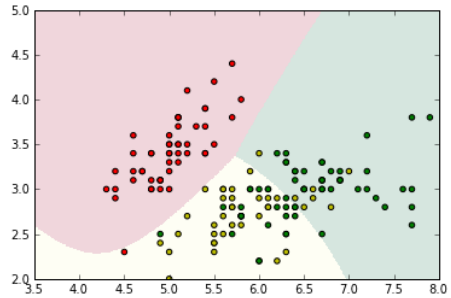

Fig. 1. KNN classification example

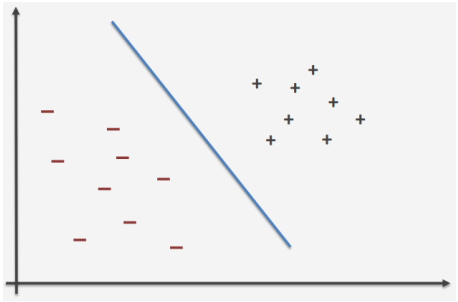

Fig. 2. SVM classification example

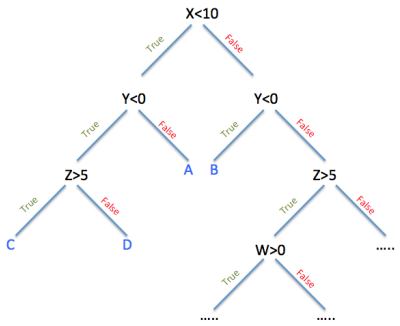

Fig. 3. Decision Tree classification example

IV. DATASETS

The datasets used in this research have been collected from the UCI Machine Learning [2] and the KEEL (Knowledge Extraction based on Evolutionary Learning) [3] dataset Repositories and have all been used in previous noise research related work.

The datasets used are listed below.

i) Mushroom Data Set - The data represent descriptions of 8124 hypothetical samples, corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family, on which the class label represents if the mushroom is edible or poisonous (Categorical attributes - Binary Classification).
ii) Glass Identification Data Set - The data represent measures of chemical ingredients taken over 214 samples of glass, on which the class label represents the type of the glass (Numerical continuous attributes - Multi-class Classification).
iii) Appendicitis data set - The data represent 7 medical measures taken over 106 patients on which the class label represents if the patient has appendicitis or not (Numerical continuous attributes - Binary Classification).

A. Data Cleaning

Due to limited information about the amount of real noise present in the initial datasets, it is assumed that they are all free from erroneous and inconsistent values. Observed instances containing missing values have been removed from the datasets.

B. Noise Insertion

In order to be able to extract useful insight about how noise can affect the performance of classification algorithms, many experiments have been conducted, by introducing various, controlled levels of noise in the clean datasets.

To begin with, the lowest level of noise was set to 10%. This means that 10% of the instances in each dataset would be changed to represent noisy data according to the transformations described below.

In order to add noisy data to the datasets containing categorical values, it was mandatory to transform those values to integers. Continuing, a way to get a discrete distribution that looks like the normal distribution is to draw from a multinomial distribution where the probabilities are calculated from a normal distribution. The values produced were inserted to the datasets and as a result 3 different datasets were created with 10% 20% and 33% of noise.

Regarding the datasets that contain numerical attributes, the optimal way to insert noise is to add real numbers to the feature values of the instances of the dataset that we consider to be noisy. The aforementioned real numbers are randomly drawn from a normal (Gaussian) distribution with mean equal to 0 and standard deviation equal to 1. The instances chosen to be altered correspond to the proportion of the dataset that we want to ”become” noisy.

The same transformation was applied to the analogous number of the instances of the datasets, for simulating the 20% and 30% noise levels.

V. STATISTICAL ANALYSIS

A. Hypothesis Selection

Hypothesis testing was introduced by Ronald Fisher, Jerzy Neyman, Karl Pearson and Pearsons son, Egon Pearson and it is a statistical method that is used in making statistical decisions using experimental data. It is in some way similar to the technique 'proof by contradiction' in mathematics, i.e. if you want to prove something then assume the opposite and derive a contradiction, i.e. something that is impossible.

i) Null hypothesis: Null hypothesis is a statistical hypothesis that assumes that the observation is due to a chance factor. Null hypothesis is denoted by; H0: 1 = 2, which shows that there is no difference between the two population means.
ii) Alternative hypothesis: Alternative hypothesis shows that observations are the result of a real effect.
iii) Level of significance: Refers to the degree of significance in which the researcher accepts or reject the null-hypothesis.

Concerning this work, the main Hypothesis that it is going to be tested is the following:

- H_0: kNN is as sensitive to feature noise as the other algorithms (Decision Trees, SVM, Naive Bayes)
- H_1: the opposite

B. Decision Errors

i) Type I error: A Type I error occurs when the researcher rejects a null hypothesis that it is true. The probability of committing a Type I error is called the significance level. This probability is also called alpha, and is often denoted by .
ii) Type II error: A Type II error occurs when the researcher fails to reject a null hypothesis that is false. The probability of committing a Type II error is called Beta, and is often denoted by . The probability of not committing a Type II error is called the Power of the test.

C. Statistical Test Selection

In order to accept or reject the null hypothesis, the appropriate statistical test should be selected. The assumption of normality of the populations is particularly common in many classical statistical tests. Thus, it is important to perform a normality test for the populations involved.

Performing the normality test on the populations of this research resulted in a large number of them being approximately normally distributed (141 out of 153 populations). However, there are only 10 samples in each population, which doesn't allow one to be confidently sure about their normality. Thus, the choice of the statistical test should derive from the fact that nothing is assumed about the distribution of the populations tested. Another restriction is that the tests will be conducted pairwise. It is also known

that the samples are all independent from each other, as they derived from independent experiments.

According to the restrictions above, a good choice would be the two-sample Kolmogorov-Smirnov test, which is a non-parametric statistical test that assesses whether two independent samples have been drawn from the same population or from two identical populations [12]. The assumptions of the two-sample Kolmogorov-Smirnov test are:

i) The samples are random.
ii) The two samples are mutually independent.
iii) The scale of measurement is at least ordinal.

It assumes nothing about the underlying distribution of the population.

While a statistical test can tell if an effect exists, it does not report the size of this effect, which is often more important for practical use in the real world. The effect size shows the magnitude of the difference between groups.

Cohen's d: The Cohen's d measure is used to evaluate the aforementioned effect size. It measures how large or small an effect size is. The formula to compute Cohen's d value is : $d = \frac{M_1 - M_2}{s}$, where $s = \sqrt{\frac{({s_1}^2 + {s_2}^2)}{2}}$

Basically, it computes the difference in the means of two samples divided by the pooled standard deviation, which is the average standard deviation of the samples involved. If the samples contain less than 50 instances, a correction factor needs to be used,

$$\text{So, } d = \frac{M_1 - M_2}{SAMPLE\ \ SD\ \ pooled} \frac{N-3}{N-2.25} \sqrt{\frac{N-2}{N}}$$

The Cohen's d value indicates the difference of the standard deviation for the two samples. For example, if d value is 1, the two samples differ by 1 standard deviation. For values greater than 0.5 and smaller than 0.8 the effect size is said to be medium and for values greater than 0.8 the effect size is said to be strong [13].

This measure would be very useful for algorithms' comparisons as the evaluation metrics are normally ordinal values and it is necessary to decide which algorithm is more effective.

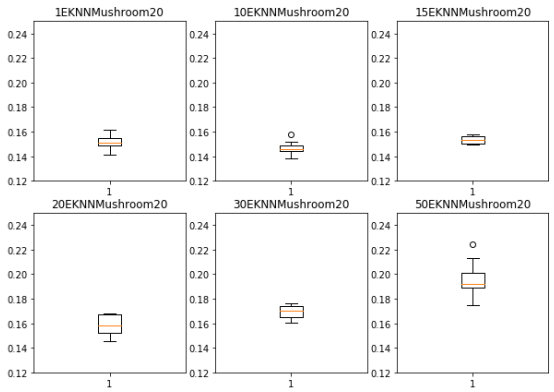

Fig. 4. ELA results for the kNN classification of Mushroom dataset containing 20% feature noise instances for 1, 10, 15, 20,30 and 50 neighbors respectively.

VI. EXPERIMENTS

A. Models' Creation

To examine the aforementioned hypothesis, we created specific models using the algorithms implemented in scikit-learn library. Firstly, kNN models were created with 1, 5, 10, 20, 30 and 50 neighbors. For each of the above neighbors Euclidean and Minkowski distance metrics were employed.

In addition, the models for Naive Bayes, SVM and Desicion Trees classification algorithms implemented in scikit-learn were used, with the default parameters.

To generalize the performance of the models cross validation was used. Many researches have shown that 10-fold cross validation suffers from high Type I error results [6]. In order to avoid this, 5 x 2 fold cross validation method was used, which resulted in 10 accuracy values for each model. All the models were tested with the original datasets, and the altered datasets with 10%, 20% and 33% noise.

Giving as input the datasets described above, the outputs produced contain the values for the accuracy metric, which is then used to compute the ELA metric described below.

B. Models' Evaluation

Accuracy measure is commonly used to evaluate the general performance of models. Many research works evaluate their results when working with noise experiments or on other fields. Although accuracy may represent the general behavior of a model, it does not support directly the comparison between the performance of the noise free and the noisy data models. In order to evaluate the impact of noise on the models, ELA metric is deployed, introduced in [5]. The formula to compute the ELA metric is : $ELA_{x\%} = \frac{(100 - A_{x\%})}{A_{0\%}}$. ELA is very convenient, when examining noise since both robustness and performance are taken into account. It represents the loss of performance (accuracy loss) of the classifier, working on a defined level of noise in relation to its performance working on noise free data. Therefore, the behavior of the classifier is shown, when noise is introduced.

VII. RESULTS

A. Algorithm comparison

In the next step, the test results, generated from the Kolmogorov-Smirnov test, are evaluated using Cohen's d measure. It is noticed that different algorithms, configuration and noise levels significantly affect the performance of the models and thus their loss of accuracy.

Results show that KNN algorithm, in most cases, is more sensitive to noise resulting in incrased ELA result. This is more accurate on some cases and less on others, depending on the models which are compared at each step.

More specifically, some worth mentioning observations generated by the experiments conducted are summarized below:

- Given all three datasets, Mushroom is the one that is the most challenging for kNN, regardless of the noise level. The largest lost of accuracy is generated for this dataset. The reason might be that the Kolmogorov-Smirnov test

gives best results when the attributes are continuous and not categorical as the ones in the Mushroom dataset.
- The dataset containing 10% noise is the one on which the loss of accuracy of kNN compared to the other models is the largest. By this we can understand that as noise levels are increased, Decision Tree, Naive Bayes and SVM models tend to face greater loss of accuracy than before.
- As the number of neighbors are increasing, kNN seems to minimize the loss of accuracy it encounters, some times reaching or even overstepping the algorithms to which it is compared.
- Support Vector Machines and Naive Bayes are, in most of our results, less sensitive to noise than kNN and as a result have a smaller Cohen's d value.
- The times Decision Trees appear to have statistically significant difference compared to kNN, are noticeable less than all the other models, leading us to understand that it is affected by noise in a similar manner to kNN. Even when Decision Tree behaves differently than kNN in terms of noise sensitivity, the effectiveness of this difference is smaller than the one produced by other models.
- As the noise level increases, kNN minimizes the loss of accuracy compared to the other models, but is still the one that is effected the most.

In order to understand which algorithm outperforms the other in terms of noise robustness, we can create a rough estimation by comparing the times an algorithm has significantly greater accuracy loss by the one compared, divided by the times this algorithm appears in the effectiveness test. This means that we only calculate the metric for the pairs where p value was less than the defined threshold (0.05) and only for Cohen's d results with medium or greater effectiveness. In our tests, kNN appeared to be more sensitive in feature noise, followed by Decision trees. SVM and Naive Bayes were generally less affected. The table below shows the order in which the algorithms are affected by feature noise in an descending order.

TABLE II
NOISE SENSITIVITY PERCENTAGE

kNN	53.92%
DT	36.84%
NB	14.06%
SVM	13.48%

B. Evaluating the effect of K

Examining the sensitivity of kNN internally, with regard to the noise level and the effect of the parameter K (number of examined neighbors) in each level of noise leads to the following insights:

- As the noise in the datasets increases, there seems to be a pattern in the number of pairs that present significant statistical difference in their loss of accuracy. The amount of these pairs increases slightly for the datasets with the 20% noise and then notably decreases for the datasets with the 30% noise.
- A general observation is that the models with a lower number of k are more sensitive to noise than the ones with a higher number of k. This makes sense, because the noise in the attributes leads to the change in the distance between the instances. This means that there is high probability for the true neighbors of an instance to be moved further away from it, due to the noise and this will lead the model to choose the next nearest neighbors, that might not be representative of the instance's closest neighborhood.
- For the datasets with the 20% noise, there is a small amount of models with higher number of k that are more sensitive to the noise than their corresponding ones with a lower number of k. This amount increased more than double for the datasets with the 33% noise. This also makes sense, because if higher noise exists in the dataset, the more the neighbors examined, the more the probability of choosing further away neighbors and deciding the wrong class.
- The behavior of KNN with regard to the noise, depends highly on the dataset. The Mushroom dataset seems to create the more significant differences between the models when just the number of k changes. This dataset consists of categorical attributes, which were converted to numerical, not real numbers and this possibly affects the performance of kNN algorithm with induced noise, due to the use of the distance.

In general, the distance metric (Euclidean or Minkowski) doesn't seem to dramatically affect the conclusions above.

Again, by using the same rough estimation as before, we can better give evidence that the less the number of neighbors, the more the sensitivity to attribute noise. The table below shows the order in which the different kNN algorithms are affected by feature noise in an descending order.

TABLE III
NOISE SENSITIVITY FOR KNNS WITH DIFFERENT K

1-NN	24.60%
5-NN	24.06%
10-NN	20.32%
15-NN	12.30%
20-NN	12.30%
30-NN	6.41%
50-NN	0.00%

VIII. CONCLUSIONS

Instinctively, one can assume that each algorithm react in a different way to attribute/feature noise. With the aforementioned work, we tried to find and prove if and how sensitive k-Nearest Neighbor classification algorithm is to attribute/feature noise. In our experiments, kNN appeared to be the one algorithm that is most affected by attribute noise, followed by Decision trees. Furthermore, the results show that attribute noise is an important aspect to be considered,

when using kNN, because it affects its performance, which in the case of noisy data, depends strongly on the selection of the number of k, the level of the noise and of course the nature of the dataset.

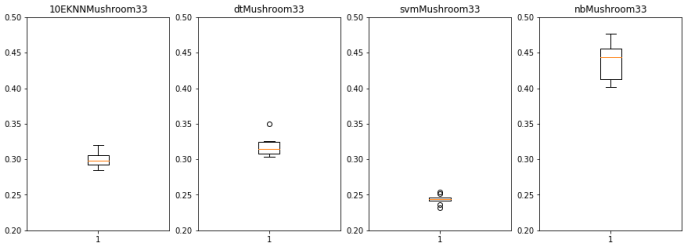

Fig. 5. ELA results for classification of Mushroom dataset containing 33 feature noise instances. The first boxplot is the ELA measure for the kNN with 10 neighbors and Euclidean distance metric classification. The second boxplot refers to decision tree classification, the third to SVM classification and the last to Naive Bayes classification.

The models of the rest of the classification algorithms were applied on the datasets, using the default parameters of the scikit-learn package of python. Thereby, a useful next step would be to repeat the experiments including hyper-parameter tuning for the classification models. Also, in order to generalize the results, more datasets and higher noise levels could be added in the experiments.

Additionally, a second step would be to run some experiments using resent extensions of kNN, that include more parameters and are created to solve some of the noise regarding inconveniences to evaluate the ”kNN category” models in total.

Last but not least, more classification algorithms, as well as classification using the very popular neural networks, could be included in the research to show if and how noise affects the classification problems in general.

REFERENCES

[1] Zhongheng Zhang, Introduction to machine learning: k-nearest neighbors. Ann Transl Med 2016.

[2] UCI Machine Learning Repository - https://archive.ics.uci.edu/ml/datasets.html.

[3] KEEL (Knowledge Extraction based on Evolutionary Learning) dataset Repository - https://sci2s.ugr.es/keel/category.php?cat=clas

[4] PERFORMANCE ANALYSIS OF K-NEAREST NEIGHBOR CLASSIFIER FOR THE DIFFERENT VALUE OF K, chap. 4, http://shodhganga.inflibnet.ac.in/bitstream/10603/20615/10/10_chapter%204.pdf.

[5] Jos A. Sez a, Julin Luengo b, Francisco Herrera, Evaluating the classifier behavior with noisy data considering performance and robustness: The Equalized Loss of Accuracy measure, Elsevier 2014.

[6] Payam Refaeilzadeh, Lei Tang, Huan Liu, Cross-Validation, Arizona State University, November 2008.

[7] Janez Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 2006.

[8] Hetal Bhavsar, Amit Ganatra, A Comparative Study of Training Algorithms for Supervised Machine Learning, International Journal of Soft Computing and Engineering 2012

[9] Seishi Okamoto, Yugami Nobuhiro, An Average-Case Analysis of the k-Nearest Neighbor Classifier for Noisy Domains

[10] David W.Aha, Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms, International Journal of Man-Machine Studies,Volume 36, Issue 2, February 1992, Pages 267-287

[11] David F. Nettleton, Albert Orriols-Puig, Albert Fornells, A study of the effect of different types of noise on the precision of supervised learning techniques, Springer 2010

[12] Siegel, S. and Castellan Jr., N.J., Nonparametric Statistics for the Behavioral Sciences. 2nd Edition, McGrawHill, New York, 1988

[13] The Essential Guide to Effect Sizes: Statistical Power, Meta-Analysis, and the Interpretation of Research Results, Paul D. Ellis, Cambridge University Press, 2010