A Perturbation-based Dataset Evaluation Approach for Fair Classifications

Davide Di Pierro[1,*,†], Stefano Ferilli[1,†]

Abstract
We are witnesses to a society in which the growing need for Artificial Intelligence in every aspect of life has pushed the research in the field. However, this enduring effort often leads to a lack of conscience in the process of evaluation of results from several perspectives. One of the most still underrepresented aspects is the detection of possible biases in the datasets used for model training, leading to unforecastable consequences for society or specific groups of people. Techniques generally used in traditional Machine Learning settings like perturbation or randomization can also be part of the evaluation of the dataset itself, in order to distinguish whether perturbations on sensitive features lead to significant changes in the output. What we propose here is a solution that allows making fictitious instances given the possibility of varying the values, thanks to ontology definitions that specify all the possible combinations for the different instances, and a metric to measure the distance between them.

Keywords
Ontology, Similarity, Reasoning, Fairness

1. Introduction

In the realm of Machine Learning (ML) becoming more and more pervasive and crucial in our everyday life activities, the capability of understanding the process governing decision-making phases of algorithms is essential to connect to Artificial Intelligence (AI) in a responsible and effective way. Basically, all the ML models learn by a first step of instance *training*, in which the model *learns* the correlation between input data and the target (correct in theory) label. The correlation between input and target will be the rationale for the algorithm to predict *unseen* data, the ones that we are interested in and make use of constantly.

In more recent years, the community of ML engineers, but not only, has raised the problem of how the models are trained. Specifically, they refer to the process of collecting and managing *training* data that the model is based on for the rest of its life. The problem we are mentioning is the dataset *unbalancement*, a sort of problem in which the dataset presents some *biases*, which can be defined as patterns that *should* not appear in the data because it does not reflect the reality of the domain we are describing. Many patterns emerging from datasets can be misleading, false, or appear only with the specificity of the data. Still, we are referring to those that can harm some categories of people, or that lead to *unfair* decision-making processes. Recognizing *unfair* datasets or biases is even a recent line of research in which researchers endeavour to find techniques for detecting and (possibly solving) biases in data. One of the most frequent tasks for which a great extent of bias has been detected is the classification of criminals that, unfortunately, in many situations resulted in using ethnicity as one of (if not THE) main features [1]. Among the possible solutions to deal with this problem, much room has been dedicated to explainable or reasoning techniques. They are particularly suitable given their interpretable nature, and the capability of experts to evaluate the quality of the model. There are, on the other hand, techniques for improving ML models based on data perturbation or randomization. Perturbing is the process of (pseudo-)randomly varying values of instances to evaluate how the model behaves after perturbation. In

3rd Workshop on Bias, Ethical AI, Explainability and the Role of Logic and Logic Programming (BEWARE24), co-located with AIxIA 2024, November 25-28, 2024, Bolzano, Italy
[*]Corresponding author.
[†]These authors contributed equally.
davide.dipierro@uniba.it (D. Di Pierro); stefano.ferilli@uniba.it (S. Ferilli)
0000-0002-8081-3292 (D. Di Pierro); 0000-0003-1118-0601 (S. Ferilli)

the context of dataset evaluation, we exploit data perturbation to understand whether slight changes in the instance (most likely in the suspected biased features) lead to significant changes in the prediction. For instance, one of the basic questions we want to be able to answer is 'What happens if I change the ethnicity of the instance?'. Perturbating values can be done in many ways but, in order to create instances that make sense, we should know all the possible domains for the feature, so we know that 'caucasian' is a possible value to test while '42' is not. For this purpose, we work under the assumption that data is based on ontology (or schema). This is twofold: (i) we know exactly the domains of values for each property, and (ii) we have a strategy to measure instances of the same class. Without this assumption, an ontology alignment phase would be necessary.

2. Related Works

In traditional ML settings, bias is statistically detected [2]. Yet these kinds of analyses are time-consuming in large scenarios and do not always provide evidence of the consequences of the biases in the decision process. The most common causes of biases are dataset imbalance and label errors [3]. Statistical analysis can mitigate this problem but, still, there is no measure for computing bias to the best of our knowledge. Given the subtle nature of biases, it is quite common to prefer interpretable models over others. For the sake of interpretability, decision trees are still one of the most frequent solutions to recognize and solve the decision processes. Going further in this direction, Logic in AI is increasingly mixing up with concerns like machine ethics and fairness.

Here we are interested in *measurement bias* [4], i.e. when improper features become too selective for the classification. In [5, 6, 7] a regularization-based technique has been proposed to mitigate unfairness. One of the most common solutions is to train multiple classifiers (ensembled) [8, 9, 10]. Zafar et al. [11] provided a mathematical interpretation of bias and developed a constraint-based solution to find the optimal classifier. Kamirant et al. [] exploited the information available about the possible values for every feature. In this way, they developed a *preferential sampling* that projects instances into a space in which it becomes evident those that are more susceptible to bias. These statistical approaches do not rely on any ontological knowledge, meaning that quite often the discriminative features need to be recognised. Statistical techniques, although highly specialized in tackling the problem, barely cannot provide a measure of imbalance, that is generally recognised as the imbalance in the dataset.

More related to symbolic solutions, Adams et al. [12] designed a Fuzzy Logic-based model capable of outperforming black box technologies for financial support with the introduction of regulations and fair outcomes. Fuzzy Logic was introduced by Zadeh [13] as an extension of predicate logic for the efficient management and computation of fuzzy concepts (like "fairness") and fuzzy rules (bureaucracy in general).

3. Methodology

In this section, we briefly describe our methodology for data perturbation in the context of ontology-based data. In this context, we define an ontology $\mathcal{O} = (\mathcal{C}, \mathcal{P}, \mathcal{R})$ where $\mathcal{C}$ is a set of classes, each of them uniquely identified by its name, $\mathcal{P}$ a named function $\mathcal{P} : \mathcal{C} \times \mathcal{D}$ from entities to a generic domain $\mathcal{D}$. Without losing generality, we suppose $\mathcal{D}$ can only be one of these domains $\{Int, Categorical, Ordered, String\}$. Int represents a set of number withing a range. $Ordered$ is a set of defined values like $Categorical$ but ordered like Likert scales. $\mathcal{P}$ expresses all the possible properties that can be defined for instances. If a property exists for a class at the ontological level, it means that it *might* be present also in instances. On the other hand, if a property is present for instances, it *must* exists also in the ontological level. $\mathcal{R} : \mathcal{C} \rightarrow \mathcal{C}$ is the set of relationships between entities. In this work, we do not take them into account. Among the relationships, *isA* the relationship expressing the *subclass* ontological concept. This will be useful since instances can be compared only if they belong to the same entity or one is *subclass* of the other. We assume every instance belongs to exactly one class and that, for each class, there exists at most one *isA* relationship, i.e. single inheritance.

3.1. Feature Similarity

Given the above setting, we can now define how to measure differences between features of instances. Features are the peculiar characteristics of instances, described in the form of the above-mentioned properties. Remind that the domain $\mathcal{D}$ can be any of $\{Int, Categorical, Ordered, String\}$, and so a distance metric must be defined for each of them. Specifically, we propose the following ones:

- **Int**: given $lower, upper \in \mathbb{N}, lower \leq upper$, a range $\mathcal{I} = [lower, upper]$ and $x, y \in \mathcal{I}$,

$$0 \leq distance(x, y) = \frac{2 \cdot \lceil \log_2 |I| \rceil \cdot |x - y|}{|I|} \leq 2 \cdot \lceil \log_2 |I| \rceil$$

- **Date**: given x, y two dates expressed as `yyyy/mm/dd`, $distance(x, y)$ is equal to the ordinary number of days separating the two.
- **Categorical**: given a set of labels $\mathcal{L}$ and $x, y \in \mathcal{L}$,

$$distance(x, y) = \begin{cases} 1, & \text{if } x \neq y \\ 0, & \text{otherwise} \end{cases}$$

  Note that with $|\mathcal{L}| = 2$ (boolean case), we can map this case onto the Int with $\mathcal{I} = \{0, 1\}$ and we have $distance(0, 1) = \frac{2 \cdot \lceil \log_2 |\in| \rceil \cdot |0-1|}{2} = 1$ as expected.
- **Ordered**: given a set of labels $\mathcal{L}$ and a bijective function $O : \mathcal{L} \rightarrow \{0, 1, ..., |L| - 1\}$ specifying the order and $x, y \in \mathcal{L}$,

$$0 \leq distance(x, y) = |O(x) - O(y)| \leq |\mathcal{L}|$$

- **String**: given an alphabet Σ and $x, y \in \Sigma^*$,

$$0 \leq distance(x, y) = DL(x, y) \leq max\{|x|, |y|\}$$

  where $DL(x, y)$ is the Damerau-Levenshtein distance [14], particularly useful and suited for spell-checking.

In principle, an ontology designer may recognize which are the most prominent features to define similarity between instances of the same class. For instance, the features *name* and *surname* for **Person** will be much more useful than *age* or *title*. Suppose for each property $p \in P$ we have a weight function $W : P \rightarrow R$, the overall distance between two instances d_1, d_2 is

$$distance(d_1, d_2) = \frac{\sum_{p \in P_{d1} \cap P_{d2}} distance(p(d1), p(d2)) \cdot w(p)}{\sum_{p \in P_{d1} \cap P_{d2}} w(p)}$$

where P_d represents the set of properties available for instance d, and $p(d)$ the value of property p for instance d.

Unfortunately, it is not always the case for weighting properties for practical reasons given the time required in specifying all the weights, but also for the specificity of the domain knowledge required. For these reasons, it is much more convenient to identify a generic strategy based on types. Like the example about *name* and *surname*, it is intuitive to assume that $String$ types are much more relevant in similarity computation. This is because the free text provides (in general) more specific information, and their equality/inequality provides quite often a good guess of how similar two instances are. If you reason with generic real-world instances you may think about people (often identified by name and surname), objects (which have name), places (which have names and addresses) and so on. For this reason, we prioritize similarities between strings over the others. Next, we claim that dates are stable, reflecting the date for which events happened. Apart from errors, it can be referred to as a meaningful feature for understanding similarity. Finally, integer values are less stable than categories, in the sense that the number of times for which something happened is subject to changes over time. Following the above-mentioned idea of weighting, we now assign the same weight to all the features of the same type. In this case, it is much easier for an ontology designer to weigh only types, which should be a few in principle. Naming α_{string}, α_{date}, $\alpha_{categorical}$, $\alpha_{ordered}$, α_{int} the weights of types (respectively) $String$, $Date$, $Categorical$, $Ordered$, Int, and $\mathcal{T} = \{String, Date, Categorical, Ordered, Int\}$ the formula will be:

$$distance(d_1, d_2) = \frac{\sum_{t \in T} \alpha_t \sum_{p \in P_{d1} \cap P_{d2}, P(p)=t} distance(p(d1), p(d2))}{\sum_{t \in T} \alpha_t \cdot |p \in P_{d1} \cap P_{d2}, P(p) = t|}$$

Given the interpretable and ontology-based nature of the problem, we implemented the procedure for distance computation in Prolog. Listing 1 shows the main computation.

Listing 1: Prolog Code for an example of distance computation

```prolog
node_distance(N1, N2, AlfaInt, AlfaDate, AlfaCategorical, AlfaOrdered, AlfaString
    , Properties, Distance) :-
        findall(P1, property(N1, P1, _), Properties1),
        findall(P2, property(N2, P2, _), Properties2),
        findall(P, (member(P, Properties1), member(P, Properties2)), Properties),
        findall(P, (member(P, Properties), type(P, int)), IntProperties),
        findall(P, (member(P, Properties), type(P, date)), DateProperties),
        findall(P, (member(P, Properties), type(P, categorical)),
            CategoricalProperties),
        findall(P, (member(P, Properties), type(P, ordered)), OrderedProperties),
        findall(P, (member(P, Properties), type(P, string)), StringProperties),
        node_int_distance(N1, N2, IntProperties, 0, DInt),
        node_date_distance(N1, N2, DateProperties, 0, DDate),
        node_categorical_distance(N1, N2, CategoricalProperties, 0, DCategorical)
            ,
        node_ordered_distance(N1, N2, OrderedProperties, 0, DOrdered),
        node_string_distance(N1, N2, StringProperties, 0, DString),
        Numerator is AlfaInt * DInt + AlfaDate * DDate + AlfaCategorical *
            DCategorical + AlfaOrdered * DOrdered + AlfaString * DString,
        length(IntProperties, LInt),
        length(DateProperties, LDate),
        length(CategoricalProperties, LCategorical),
        length(OrderedProperties, LOrdered),
        length(StringProperties, LString),
        Divisor is LInt * AlfaInt + LDate * AlfaDate + LCategorical *
            AlfaCategorical + LOrdered * AlfaOrdered + LString * AlfaString,
        Distance is Numerator / Divisor.

entity(person). entity(student).
isA(student, person).
attribute(age, person). attribute(date_of_birth, person). attribute(is_worker,
    person).
attribute(qualification, person). attribute(name, person).
type(age, int). type(date_of_birth, date). type(is_worker, categorical).
type(qualification, ordered). type(name, string).
category(is_worker, 'yes'). category(is_worker, 'no').
order(qualification, 'high_school', 0). order(qualification, 'bachelor', 1).
order(qualification, 'master', 2). order(qualification, 'phd', 3).
node(1, person). node(2, student).
property(1, age, 52). property(2, age, 27).
property(1, date_of_birth, '1972/12/10'). property(2, date_of_birth,
    '1997/05/17').
property(1, is_worker, 'yes'). property(2, is_worker, 'no').
property(1, qualification, 'phd'). property(2, qualification, 'master').
property(1, name, 'stefano'). property(2, name, 'davide').
```

4. Dataset Evaluation

Given these instruments, we can perform some analysis of datasets, in order to understand distances among instances that are differently classified by some ML algorithms. Specifically, we mention two possible analyses: distance-based and generative. While the first gives a glance at the relevant distance for an instance to change the label, the latter allows a deeper understanding of every feature in the dataset, providing a measure to verify how variations affect the result.

Distance-based In this analysis, we group instances of the test set according to how they have been classified and find the pair of instances belonging to distinct classifications that minimize the distance.

Generative In this analysis, we take a subset of instances equally classified and, by varying one of its features up to certain distance thresholds, we verify the distance for which the classification changes based only on one feature. After the analysis of one feature, pairs can be considered, followed by triples and so on.

4.1. First Results with Credit Cards Approval

A first analysis has been conducted on the Credit Approval dataset[1]. The dataset contains 690 instances and is composed of 15 features, of types Int, Real and Categorical. Some features regard gender and ethnicity, features that in principle should not be taken into account when deciding credit card approval The goal is to classify whether people At first, we performed a classification based on an interpretable model. Interpretability provided us with an easier way to determine which features to test first. We performed classification tasks with Decision Tree [15] and traditional split between train (80%) and test (20%). Following decision rules, we detected that some relevant rules are governed by the "gender" or "ethnicity" of the person, features that should not be included in the classification process. For this reason, we generated instances distant **1** for the two features and reclassified all instances in the test set. Experiments showed that more than 5% of the people in the test set were classified differently only changing gender or ethnicity. Specifically, 3% of the population changed classification based on gender, and the 2% on ethnicity, and the intersection is empty. Experiments have been conducted on an Intel(R) Core(TM) i7-1065G7 CPU single-core, 16GB of RAM processor. Each node has been compared with all the others, so the number of performed comparisons was $\sum_{i=1}^{N-1} i \simeq 4000$ given $N = 690$. The overall time execution was about 4 minutes, which shows that about 16 distances per second are computed.

5. Conclusions

In this work, we proposed a novel method to evaluate whether there are potential biases in the dataset exploiting a generative distance-based strategy, that can be applied to ontology-guided data. Part of the generative implementation is still ongoing and new experiments need to be conducted. Future works regard applying ML to automatically select suspected biased features in the training process.

[1]https://archive.ics.uci.edu/dataset/27/credit+approval

References

[1] R. d. V. dos Santos Júnior, J. V. V. Coelho, N. A. A. Cacho, D. S. A. de Araújo, A criminal macrocause classification model: An enhancement for violent crime analysis considering an unbalanced dataset, Expert Systems with Applications 238 (2024) 121702.

[2] T. G. Dietterich, E. B. Kong, Machine learning bias, statistical bias, and statistical variance of decision tree algorithms (1995).

[3] J. Chakraborty, S. Majumder, T. Menzies, Bias in machine learning software: Why? how? what to do?, in: Proceedings of the 29th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering, 2021, pp. 429–440.

[4] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, ACM computing surveys (CSUR) 54 (2021) 1–35.

[5] T. Kamishima, S. Akaho, H. Asoh, J. Sakuma, Fairness-aware classifier with prejudice remover regularizer, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23, Springer, 2012, pp. 35–50.

[6] N. Goel, M. Yaghini, B. Faltings, Non-discriminatory machine learning through convex fairness criteria, in: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, 2018, pp. 116–116.

[7] L. Huang, N. Vishnoi, Stable and fair classification, in: International Conference on Machine Learning, PMLR, 2019, pp. 2879–2890.

[8] A. K. Menon, R. C. Williamson, The cost of fairness in binary classification, in: Conference on Fairness, accountability and transparency, PMLR, 2018, pp. 107–118.

[9] M. Hardt, E. Price, N. Srebro, Equality of opportunity in supervised learning, Advances in neural information processing systems 29 (2016).

[10] B. Ustun, Y. Liu, D. Parkes, Fairness without harm: Decoupled classifiers with preference guarantees, in: International Conference on Machine Learning, PMLR, 2019, pp. 6373–6382.

[11] M. B. Zafar, I. Valera, M. G. Rogriguez, K. P. Gummadi, Fairness constraints: Mechanisms for fair classification, in: Artificial intelligence and statistics, PMLR, 2017, pp. 962–970.

[12] J. Adams, H. Hagras, A type-2 fuzzy logic approach to explainable ai for regulatory compliance, fair customer outcomes and market stability in the global financial sector, in: 2020 IEEE international conference on fuzzy systems (FUZZ-IEEE), IEEE, 2020, pp. 1–8.

[13] L. A. Zadeh, Knowledge representation in fuzzy logic, in: An introduction to fuzzy logic applications in intelligent systems, Springer, 1992, pp. 1–25.

[14] C. Zhao, S. Sahni, String correction using the damerau-levenshtein distance, BMC bioinformatics 20 (2019) 1–28.

[15] B. De Ville, Decision trees, Wiley Interdisciplinary Reviews: Computational Statistics 5 (2013) 448–455.