A First Experimental Study on Functional Dependencies for Imbalanced Datasets Classification

Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici

A First Experimental Study on Functional Dependencies for Imbalanced Datasets Classification

Marie Le Guilly, Jean-Marc Petit, and Marian Scuturici

Université de Lyon, CNRS, INSA-LYON, LIRIS, UMR5205
F-69621, Villeurbanne, France
{marie.le-guilly, jean-marc.petit, marian.scuturici}@insa-lyon.fr

Abstract. Imbalanced datasets for classification is a recurring problem in machine learning, as most real-life datasets present classes that are not evenly distributed. This causes many problems for classification algorithms trained on such datasets, as they are often biases towards the majority class. Moreover, the minority class often yields more interest for data scientist, when at the same time it is also the hardest to predict. Many different approaches have been proposed to tackle the problem of imbalanced datasets: they often rely on the sampling of the majority class, or the creation of synthetic examples for the minority one. In this paper, we take a completely different perspective on this problem: we propose to use the notion of *distance* between databases, to sample from the majority class, so that the minority and majority class are as distant as possible. The chosen distance is based on functional dependencies, with the intuition of capturing inherent constraints of the database. We propose algorithms to generate distant synthetic datasets, as well as experimentations to verify our conjecture on the classification on distant instances. Despite the mitigated results obtained so far, we believe this is a promising research direction, at the intersection of machine learning and databases, and it deserves more investigations.

1 Introduction

Databases, machine learning and data mining, are very important domains in computer science. With the recent context of *Big Data*, they are receiving an increased interest and experiencing an important expansion, both in academia and industry. However, as important as they all are, they have tended to grow in parallel, with limited interactions with one another.

There is everything to gain in bringing those domains together, and combining them to propose innovative solutions: this argumentation was already defended in 1996 in [14], and has since been followed in recent works trying to increase the permeability between these domains. Among those, some have tried to make use of machine learning algorithms to improve the use of databases, with query discovery [24], query inference [6], or query rewriting [9]. Others are trying to integrate machine learning into databases such as [26] or [10].

On the one hand, data dependencies are a powerful notion, that has been thoroughly studied for both relational databases and data mining. It has been applied in various situations, from the discovery of data quality rules [8] to association rule mining [2]. On the other hand, supervised classification is a long studied problem in machine learning.

This paper investigates whether or not those two independent domains can benefit from one another. Indeed, existing supervised learning solutions [19] do not seem to make any use of data dependencies. Is there a reason for this, and is it possible to find supervised classification problem that would be suited for the use of data dependency ?

Data dependencies, and more specifically functional dependencies, give global constraints on a dataset: they describe constraints between attributes, and in some way describe the structure of a given relation. If two relations are defined on the same schema, they might satisfy similar dependencies, or even the exact same one, or on the opposite completely different ones: quantifying this difference of functional dependencies, with some sort of metric or distance, is then a way to evaluate the similarity of two relations in terms of structure and underlying patterns. With such a distance, which is then more semantic than "traditional" distances (e.g. euclidean, Manhattan, ...), interesting sub-problems of supervised classification could be addressed with a new approach. In particular, it can be interesting to look at the problem of imbalanced datasets: this happens in binary classification, when one class is much bigger than the other. In this situation, classifiers are biased and tend to always predict the majority class as there are many more examples of it. But often the minority class is actually much more interesting and is the one that data analysts want to predict accurately. One example of this is a dataset of banking transactions, which only contains a tiny proportion of fraudulent transactions, against thousands of regular ones. But this tiny portion is still much more interesting as they are the one that are crucial to detect !

Dealing with imbalanced datasets is a common issue in machine learning, and various strategies have been proposed [18]. Among those, a common one is the undersampling of the majority class: select only a subset of examples in it, such that its size is similar to the one of the minority class. There are several undersampling strategies: basic random sampling, identifying clusters in the majority class, ... But to the best of our knowledge, there is no technique that makes use of global constraints for imbalanced datasets. Using the notion of a distance based on data dependencies, and selecting *distant* tuples (or on the contrary *close* ones), makes sense: such an undersampling strategy could provide a disruptive approach to the imbalanced dataset problem, by not only looking at data values but also at the structural properties of data.

The general idea is to compute a set of functional dependencies satisfied by the minority class, and compute the set of functional dependencies that are as distant (or close) as possible from it.

The challenge is then to identify the tuples in the majority class that, together, satisfy exactly (or as many as possible) this set of dependencies. This is

not a trivial problem, and it raises several combinatorial challenges. Moreover, it is based on this notion of semantic distance based on data dependencies: it makes the study of such a problem really ambitious but also quite perilous. As a first step, it is therefore possible to consider several subproblems, that can be summarized as follow:

> Is it easier to classify imbalanced datasets between *distant* relations ? And can functional dependencies help to identify better balanced classification datasets ?

In this paper, we present our ongoing work, to partially answer these two difficult questions. Our approach works as follows: instead of getting dependencies from existing data, we propose to first determine the required constraints in order to then generate synthetic data accordingly. We can then refine our problem statement:

> Is it possible to find synthetic datasets verifying this conjecture: datasets that are distant in terms of DF are easier to classify ?

To answer this, the contributions of this paper are then :

- The use of a semantic distance based on functional dependencies and closure systems, as described in [17].
- The construction of a synthetic imbalanced dataset such that the distance between the minority and the majority class is maximum.
- Experimentations, applying various classification models, to compare classifiers performances when discriminating between the different relations generated.

The purpose is first to point out if synthetic relations generated as *distant* are easier to classify than random relations. Then, further experimentations verify how well classifier trained on such relation adapt when tested on the imbalanced dataset. The obtained results are mitigated, reflecting the difficulty of the new problem presented in this paper. Even if further investigations will be necessary in future works, especially regarding the values of the synthetic data, we believe that we propose a first attempt at tackling an important problem at the intersection of machine learning and databases.

Section 2 introduces the necessary preliminary notions required to understand the paper. Then section 3 explicits the notion of distance between instances of a database, and explains the intuitions lying behind it. Sections 4 describes the generation strategies, from closure systems to imbalanced dataset. Based on this, experimentations are detailed in section 5, comparing different approaches, and testing the conjecture of this paper on various classification algorithms. Finally, section 6 offers a discussion on the presented approach and related works, as well as a conclusion.

2 Preliminaries

We first recall basic notations and definitions that will be used throughout the paper. It is assumed that the reader is familiar with databases notations (see [20]).

Let U be a set of attributes. A relation schema R is a name associated with attributes of U, i.e. $R \subseteq U$. A database schema $\mathcal{R}$ is a set of relation schemas.

Let D be a set of constants, $A \in U$ and R a relation schema. The domain of A is denoted by $dom(A) \subseteq D$. A tuple t over R is a function from R to D. A relation r over R is a set of tuples over R. The active domain of A in r, denoted by $ADOM(A, r)$, is the set of values taken by A in r. The active domain of r, denoted by $ADOM(r)$, is the set of values in r.

We now define the syntax and the semantics of a Functional Dependency (FD).

Let R be a relation schema, and $X, Y \subseteq R$. A FD on R is an expression of the form $R : X \rightarrow Y$ (or simply $X \rightarrow Y$ when R is clear from context)

Let r be a relation over R and $X \rightarrow Y$ a DF on R. $X \rightarrow Y$ is satisfied in r, denoted by $r \models X \rightarrow Y$, if and only if for all $t_1, t_2 \in r$, if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$.

Many concepts have been defined on set of FDs, some of them are presented below.

Let F be a set of FDs on U and $X \subseteq U$. The closure of X w.r.t F, denoted by X_F^+, is defined as : $X_F^+ = \{A \in U \mid F \models X \rightarrow A\}$ where $\models$ means "logical implication". X is closed w.r.t F if $X_F^+ = X$. The closure system $CL(F)$ of F is the set of closed sets of $F : CL(F) = \{X \subseteq U | X = X_F^+\}$
There exists a unique minimal subfamily of $CL(F)$ irreducible by intersection, denoted by $IRR(F)$, and defined as follows:

- $IRR(F) \subseteq CL(F)$, $U \notin IRR(F)$
- for all $X, Y, Z \in IRR(F)$, if $X \cap Y = Z$, then $X = Z$ or $Y = Z$.

Finally, the concept of Armstrong relations [3] allows to obtain relations satisfying a set of functional dependencies, and only those dependencies. Let F be a set of FD on R. A relation r on R is an Armstrong relation for F if $r \models X \rightarrow Y$ if and only if $F \models X \rightarrow Y$.

There exists a relationship between Armstrong relations and closure systems [4]. First, agree sets have to be defined. Let r be a relation over R and $t_1, t_2 \in r$. Given two tuples, their agree set is defined as: $ag(t_1, t_2) = \{A \in R | t_1[A] = t_2[A]\}$. Given a relation, their agree sets are then: $ag(r) = \{ag(t_1, t_2) | t_1, t_2 \in r, t_1 \neq t_2\}$.

Then, the relationship can be given as a theorem [4]:

Theorem 1 *Let F be a set of FDs on R and r be a relation over R.*
r is an Armstrong relation for F if and only if $IRR(F) \subseteq ag(r) \subseteq CL(F)$

3 Distance between databases

The notion of distance has been studied for years. In computer science, distances are often required, especially in machine learning, for example for Clustering algorithms or K-nearest-neighbors (see [13]). It is possible to define distances between numerical values, vectors, but also words, sentences, ... However, the notion of distance between databases is not a notion that seems to have been given much attention. A first attempt can be found in [21], that proposes an update distance between databases, similarly to the edit distance for strings: the distance between two databases is the minimal number of modification operations to be applied to one database to obtain the other one. However, this distance is not symmetric, and is mostly defined for cases of multiple replications of a database, when the same database is dulicated and modified at different places.

Another definition of distance between databases is given in [17]. This distance is defined in terms of functional dependencies, using the notion of closure systems. Formally, the distance between two databases, instances of the same schema, is defined as follows:

Definition 1 *[17] Let r_1 and r_2 be two relations over $\mathcal{R}$, and F_1 (respectively F_2) the FDs statisfied in r_1 (respectively r_2). The distance between r_1 and r_2 is:*

$$d(r_1, r_2) = |CL(F_1) \triangle CL(F_2)|$$

where $A \triangle B$ denotes the symmetric difference of the two sets, i.e:

$$A \triangle B = A \setminus B \cup B \setminus A.$$

This definition can be puzzling at first, and it is necessary to understand the intuitions that lie behind it. Closure systems are intimately related to functional dependencies: therefore similar closure systems lead to similar FDs, and vice-versa. As a consequence, regardless of the values taken by the values of the instances, this distance characterizes the similarity of two sets of functional dependencies. Indeed, distant relations tend to have opposite FDs, while close ones will have similar or even shared FDs.

Moreover, the following property holds:

Property 1 *[17] Let $|\mathcal{R}| = n$. Then $d(r_1, r_2) \leq 2^n - 2$ for any two instances of schema $\mathcal{R}$.*

Example 1 *Let's take the two following closure systems:*

- $CL_1 = \{ABC, AB, AC, A, \emptyset\}$
- $CL_2 = \{ABC, BC, B, C, \emptyset\}$

For two relation r_1 and r_2 with respective closure systems CL_1 and CL_2, the distance is:

$$d(r_1, r_2) = |\{AB, AC, A, BC, B, C\}| = 2^3 - 2 = 6$$

r_1 is a far as possible from r_2 and vice-versa. Moreover, they respect the following sets of functional dependencies:

- $r_1 \models \{B \rightarrow A, C \rightarrow A\}$
- $r_2 \models \{A \rightarrow BC\}$

This illustrates this intuition of "opposites" functional dependencies in distant relations.

This notion of distance between databases is clearly semantic: it does not look at all at the domains of the relation's attributes, or at their types. Instead, it captures underlying patterns and structure, through functional dependencies, and evaluates the distance between relations as the similarity in terms of such structure and patterns between them.

This is exactly the semantic distance that gives an indication of how two relations are distant regarding their functional dependencies. The strong semantic meaning carried by this distance could match the needs required for the problem defined in this paper as motivated in the introduction.

4 Imbalanced datasets generation

In order to verify the conjecture that distant relations, in terms of functional dependencies, should be easier to classify, a five step process is proposed. The idea is to recreate the conditions of an imbalanced dataset, simulating two different sub-samples r^+ and s from a majority class Z, to compare their classification against a minority class r^-. More specifically, we simulate an imbalanced dataset, where r^+ and s are two different samples of Z, generated with two different strategies: tuples in s are selected at random from Z, while r^+ is generated to be as distant from r^- as possible. The structure of those different relations is outlined on figure 1.

We follow the following process:

Step 1: For a given schema $\mathcal{R} = \{A_1, ..., A_n\}$ of size n, create two closure systems CF^+ and CF^-, as close as possible in size, but as different as possible in terms of attribute set.

Step 2: From step 1, generate two Armstrong relations r^+ and r^- for CF^+ and CF^-.

Step 3: Apply a classification model to discriminate between r^+ and r^-.

Step 4: Create a synthetic majority relation Z over $\mathcal{R}$, from which r^+ is supposed to be a sample.

Step 5: Sample Z to get s, such that $|s| = |r^-|$, and classify between s and r^-.

The objective is to see if the performances of classifiers are better when discriminating between r^+ and r^-, two relations constructed using "opposite" functional dependencies, than between $|s|$ and $|r^-|$, where s is "only" a random

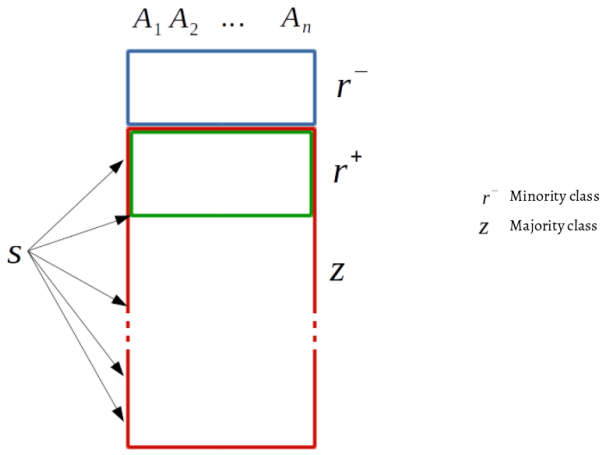

Fig. 1. Generation process for random data

sample. This approach raises several questions and sub-problems, that are addressed in the following sections. As expected, the most crucial one is the data values taken to build r^-, r^+ and therefore Z. Indeed, by definition, FDs care about data equality, but independently of the data values themselves. However, the classification algorithms do care about them, and therefore the generation strategy might modify the results obtained in our experiments. It is therefore important to approach this problem carefully.

4.1 Synthetic closure systems generation

The first problem is to generate two closure systems such that 1) the size of their symmetric difference is maximum and 2) the sizes of the two closure systems is as close as possible. This is not a trivial problem: given a schema $\mathcal{R}$ of size n, many different closures system can be obtained satisfying this condition.

This problem is interesting and difficult, but is not the center of our paper. To be able to generate automatically two closure systems with a maximized symmetric difference, we propose algorithm 1. It uses a level-wise (top-down) breadth-first approach strategy on $\mathcal{P}(\mathcal{R})$:

The algorithm works as follows:

- The two closure systems are initialized with $\mathcal{R}$ as it necessarily belongs to each of them.
- At a given level, all elements that do not belong in either CF^- or CF^+ are considered as available candidates for insertion in one of the closure systems. They are selected in a random order, so that each execution of the algorithm does not ensure to produce the same result. This way, we can obtained various closure systems to work with.

- Before inserting an element e, it is necessary to verify if its insertion would satisfy the properties of a closed set. Thus, if the intersection of an element e with any of the elements of same size in CF^- is an element from CF^+, then e has to be added to CF^+, and vice-versa.
- Otherwise an element is added to the closed set with the smallest number of elements. This is to obtain the closest possible size between CF^- and CF^+.
- Once it is decided in which set an element is added, it is also necessary to add all its intersections with elements of same size that already are in the closure system: this is the role of the recursive function **add**.

Algorithm 1: Closure systems generation procedure

```
1  procedure ClosureSystems (R);
   Input  : A schema R
   Output: Two closure systems CF⁻ and CF⁺ such that
           CF⁻ △ CF⁺ = 2^|R| − 2
2  CF⁻ = {R, ∅}
3  CF⁺ = {R, ∅}
4  for l = |R| to l = 1 do
5  |  available = {e ∈ P(R)| |e| = l, e ∉ CF⁻ and e ∉ CF⁺}
6  |  for each e in available do
7  |  |  if ∃a ∈ inter(e, CF⁻) such that a ∈ CF⁺ then
8  |  |  |  add(CF⁺, e)
9  |  |  end
10 |  |  else if ∃a ∈ inter(e, CF⁺) such that a ∈ CF⁻ then
11 |  |  |  add(CF⁻, e)
12 |  |  end
13 |  |  if e ∉ CF⁻ and e ∉ CF⁺ then
14 |  |  |  if |CF⁻| < |CF⁺| then
15 |  |  |  |  add(CF⁻, e)
16 |  |  |  end
17 |  |  |  else
18 |  |  |  |  add(CF⁺, e)
19 |  |  |  end
20 |  |  end
21 |  end
22 end
23 return CF1, CF2
24 Function add(CF, e):
25 |  sameSize = {a ∈ CF|length(a) = length(e)} for a ∈ sameSize do
26 |  |  i = a ∩ e
27 |  |  if i ∉ CF then
28 |  |  |  add(CF, i)
29 |  |  end
30 |  end
```

The idea behind this algorithm is, given a schema R, to divide evenly all elements from $\mathcal{P}(\mathcal{R})$ into the closure systems. Therefore, they can not be constituted at random, and the insertion of an elements in a closed set has to guaranty some properties, especially regarding the closure by intersection. This algorithm allows to obtain diverse closure systems even for schema of consequent size, automatizing a task which is not feasible "manually". This turns out to be valuable for experimentations, as various closure systems, and therefore various relations, can be tested using this algorithm. It should however be noted that this algorithm has an exponential complexity in the size of $\mathcal{R}$, that limits the size of schema that can be used, if the closure systems are to be obtained in a reasonable amount of time. In practice, we set $|\mathcal{R}| = 12$ in our experimentations.

4.2 Data generation from closure systems

Once two closure systems are generated it is quite easy to derive relations from them, using Armstrong relations, as explained in [3]. Indeed, the structure of an Armstrong relation for a set of functional dependencies is a problem that as already been addressed (see [4]). It relies on the results of theorem 1.

Given a closure system, an Armstrong relation is defined with a reference tuple t_0 to encode each member of the irreducible set obtained from the closure system with respect to this reference: for a tuple t encoding the i^{th} element X, $t[X] = t_0[X]$ and $t[A] = i$, with $A \in \mathcal{R} \setminus X$.

Example 2 *Let's take the first closure systems from example 1:*

$CF^- = \{ABC, AB, AC, A, \emptyset\}$ *and* $IRR^- = \{ABC, AB, AC\}$

Relation r^- *is derived from* CF^-*:*

r^-	A	B	C	*encodes*
	0	*0*	*0*	*reference*
	0	*0*	*1*	AB
	0	*2*	*0*	AC

In our setting, the situation is slightly more complicated, as there are two closure systems. Therefore it implies to make some additional choices regarding the generation of relations, especially the second one. The future use of those relations, i.e for classification problems, should also be taken into account.

Indeed, there is a balance to be found, in order to obtain a convincing classification dataset: the dataset should have two classes, so that their values are not completely similar, but also such that the problem to solve is not trivial, meaning the two classes should overlap in some way.

In particular, there is the question of the reference value for the second relation: it could be the same for both of them. But that would mean that the two relations would have one tuple in common, and a great number of similar tuples, as they would use the same values and therefore have the exact same

active domain. Moreover, if in one relation, all the tuples share the same reference value, the classification problem might become too trivial, as the algorithm might be biased toward learning weather or not this specific reference value appears in a tuple, and therefore discriminate between the classes solely based on their respective reference value.

This balance is a crucial point of our process, that deserves more investigations in future works. In this paper, we propose two elements to address this problem:

- For one relation, the reference value to use is based on the previous tuple that was inserted in the relation.
- The two relations do not share any common reference value

We formalize our approach in algorithm 2, that details how the relations are created in order to respect the given constraints.

Algorithm 2: Relations generation given two closure systems

```
1  procedure RelationsFromCS (R, CF⁻, CF⁺);
   Input  : A schema R, two closures systems CF⁻ and CF⁺
   Output: Two relations r⁻ and r⁺ with respective closure systems CF⁻
           and CF⁺ and overlapping active domains
2  n = |R|
3  values = [0...2^n + 2]
4  r⁻ = ArmstrongRelation(CF⁻, values, n)
5  r⁺ = ArmstrongRelation(CF⁺, values, n)
6  return r⁻, r⁺
7  Function ArmstrongRelation(CF, values, n):
8      r ← |R| * |CF| matrix
9      refvalue = random(values)
10     values.remove(refvalue)
11     t_ref = [refvalue] * n
12     r[0] = t_ref
13     i = 1
14     for each e ∈ CF do
15         randomvalue = random(values)
16         values.remove(randomvalue)
17         t = [randomvalue] * n
18         for each X ∈ e do
19             t[X] = r[i − 1][X]
20         end
21         r[i] = t
22         i + +
23     end
24     return r
```

It works as follows:

- Given a schema $\mathcal{R}$ of size n, a pool of possible values is generated, with all integers values from 0 to $2^n + 2$, which is equal to $|CF^-| + |CF^+|$ plus two reference values. This represents all the values that have to be used to generate the two Armstrong relations.
- Then function **ArmstrongRelation** is applied to CF^- to generate r^-.
- A reference value is selected at random in the pool of possible values. It is used to construct the reference tuple, so it is then no longer a possible new value (line 10).
- Then for each element in the closure system, a random value is selected (at random) in the pool of remaining possible values. It is used to create a new tuple, and the random value is thus removed from the pool. Each attribute from the considered element is encoded with respect to the previous tuple in the relation.
- Once r^- is complete, the same procedure applies for r^+, using the remaining values in the pool of available values.

Example 3 *Following example 2, and using the second closure system from 1:*

$CF^+ = \{ABC, BC, B, C, \emptyset\}$ *and* $IRR^+ = \{ABC, BC, B, C\}$

Applying algorithm 2 we could obtain:

r^+	A	B	C	*encodes*
	3	*3*	*3*	*reference*
	7	*3*	*3*	BC
	2	*3*	*2*	B
	9	*9*	*2*	C

r^-	A	B	C	*encodes*
	1	*1*	*1*	*reference*
	1	*1*	*6*	AB
	1	*4*	*6*	AC
	1	*5*	*5*	A

4.3 Random Data generation

Once the two relations derived from closure systems are generated, it is time to generate relations Z and s, in order to give a basis for comparison of classifiers. The generation of such a relation Z has to be thought carefully, and be coherent with the generation technique previously applied. For sake of clarity, we only consider values in $\mathbb{N}$.

The final objective is to study the case of imbalanced dataset, and to propose a new undersampling technique based on functional dependencies. The active domain of this relation is crucial, and various strategies are possible:

- $ADOM(Z) = ADOM(r^+)$

- $ADOM(Z) = ADOM(r^+ \cup r^-)$
- $ADOM(Z) = ADOM(r^+ \cup r^-) \cup I$ where I is an interval in $\mathbb{N}$.

The last solution has some flaws, because it would add tuples with values in I that would be easier to classify, with the same problem than explained in example 3: better classification score would then not be due to functional dependencies but to the fact that tuples are on two distinct domains that are easy to separate. But the two first possibilities are both interesting, giving two possible levels of difficulty. The first one, $ADOM(Z) = ADOM(r^+)$, proposes the exact same conditions when comparing between r^- versus r^+ and r^- versus s: indeed both r^+ and s have an active domain that is overlapping with r^-'s one, but with no common value. The second possibility, $ADOM(Z) = ADOM(r^+ \cup r^-)$, introduces some noise as s and r^- could have values in common, which would provide a more challenging dataset. Therefore, these two options are explored in the experimentations (section 5).

Once the active domain of Z is set, its generation is not complicated, and the number of possible tuples is bounded.

Property 2 *Let $\mathcal{R}$ be a schema of size n, and Z a relation such that $|ADOM(Z)| = m$. The number of possible tuples for Z is the number of permutations of size n from an alphabet of m elements, i.e $max(|Z|) = m^n$.*

This grows considerably fast, and clearly the size of Z can be arbitrarily large. Generating all possibilities would take time and consume memory. Therefore, to limit the size of Z, we consider the factor sf, which is the ratio of size difference between r^- and Z:

$$sf = \frac{|Z|}{|r^-|}$$

This allows to adapt the "imbalanceness" of the dataset, and to generate only a limited number of tuples for Z: the generation method for this relation is simply to create a relation on a schema R of size n, with a fixed number of $|r^-| * sf$ tuples, and for each tuple and each attribute, select randomly a value in $ADOM(Z)$.

Finally, relation s is just a random sample of size $|r^-|$ over $Z \cup r^+$, just like a random undersampling of the majority class for an imbalanced dataset problem.

4.4 Classification problem

At this point, all necessary relations have been generated. We have the minority class r^-, and the majority one Z. r^+ and s are samples from Z, such that r^+ is as distant as possible from r^- in terms of FDs, while s is just a random sample from Z without any FD consideration.

The purpose is now to apply classification algorithms on different datasets and compare their performances. The first objective is to see if it is easier to classify between distant datasets, and the comparison is therefore done between r^- versus r^+ and r^- versus s. Classification datasets are constituted only by combining two relations and adding one additional *class* attribute.

Example 4 *Let's take relations r^- and r^+ from example 3 . They constitute the following classification dataset:*

	A	B	C	*class*
r^-	3	3	3	-
	7	3	3	-
	2	3	2	-
	2	3	2	-
r^+	1	1	1	+
	1	1	6	+
	1	4	6	+
	1	5	5	+

Using the available datasets, we build two classification models:

- r^- versus r^+
- r^- versus s

Each dataset has then to be divided into training and testing sets, with proportions to be given depending on the experimentation setting. Algorithms are then trained on the training set and their performances evaluated on the testing one. For this preliminary study, the score used to compare classifiers is **accuracy**. This first test relates to one our our initial questions, which is whether or not it is easier to classify between distant sets.

For our second experimentation, we come back to the problem of imbalanced datasets. To do so, the same relations can be used but with a slight change in the testing and training sets. Indeed, the philosophy of the data generation presented previously is to emulate such a problem, with r^+ and s being two different undersampling of a bigger relation Z. Therefore we train the models exactly as before, but the testing sets are now different: both models are now evaluated using tuples from r^- and Z, as in a real imbalanced dataset scenario.

5 Experimentations

5.1 Implementation

The algorithms have been implemented using Python 3. All classification algorithms are from the scikit-learn machine learning library [22].

Ten classification algorithms were selected for the experimentations (see [13] for details), with a fixed parametrization as follows:

- K Nearest Neighbors: classification according to the class of surrounding examples. Fixed $k = 3$.
- Decision Tree: learns decision rules and builds a tree. Fixed a maximum depth of 5 for the tree.
- Random Forest: several decision trees on different subsamples of the data. Maximum depth of 5 for 10 trees in total.

- AdaBoost on a decision tree: give different weights to examples, by increasing the weight of misclassified examples.
- Neural Network: fixed two hidden layers with 12 neurons each.
- Naive Bayes: probabilistic model based on Bayes theorem, with the "naive" assumption that variables are independent.
- Quadratic Discriminant Analysis: finds a quadratic decision surface.
- Linear Support vector machine: classic SVM with linear kernel
- Radial Basis Function (RBF) kernel Support vector machine: RBF kernel.

5.2 Semantic distance and classification

The purpose of the first set of experimentations was to study the first conjecture on which this paper is based: it is easier to classify between *distant* sets. The experimental verification of this is simple and follows the explanations given in section 4.

For experimentations, the size of the schema is fixed to $n = 12$. Then:

- $|\mathcal{P}(\mathcal{R})| = 4096$
- $|r^+ \cup r^-| = 4099$

Moreover, sf is fixed to 100: $|Z|$ is therefore around 200 000 tuples. Training test is composed of 80% of a dataset, the remaining 20% are for testing.

The experience was done on ten different instances generated with algorithms 1 and 2, with each time a new closure system generation and new relations, considering the random components of each algorithm. This is done to make sure any observation is not due to a specific relation, but really a general phenomenon.

	$ADOM(Z) = ADOM(r^+)$		$ADOM(Z) = ADOM(r^+ \cup r^-)$	
Classifier	r^- **vs** r^+	r^- **vs** s	r^- **vs** r^+	r^- **vs** s
Nearest Neighbors	0.95	0.87	0.93	0.77
Decision Tree	0.99	0.99	0.99	0.96
Random Forest	0.99	0.99	1.0	0.99
AdaBoost	0.99	0.99	0.99	0.99
Neural Net	0.81	0.72	0.85	0.77
Naive Bayes	0.99	0.99	1.0	0.75
RBF SVM	0.82	0.79	0.77	0.70
Linear SVM	0.62	0.48	0.67	0.47

Table 1. Accuracy of each classifier for each data generation strategy. Both models are evaluated on their own testing sets.

Table 1 presents the average accuracy score obtained for each algorithm over the ten iterations, when data is generated such that $ADOM(Z) = ADOM(r+)$, so that Z and r^- do not share any common value. Those results are very encouraging, as it appears that classifiers perform better for the distant instances,

supporting the conjecture that is easier to classify between distant sets. Indeed, this tendency does not appear to be limited to only one or a few algorithms. Moreover, if the observed difference is anecdotal for some algorithms such as Random Forest and Adaboost, it is pretty important for others. This also opens the way for other paths of research, on how each specific algorithm can be affected by functional dependencies, or how functional dependencies could be integrated in the algorithms themselves to improve their performances. Finally, the active domain on the majority class does not seem to affect our observations, as results are also good with noisy data, and even better in some cases.

5.3 General imbalanced dataset problem

With the good results obtained for the first experimentation, a second one was conducted, closer to the initial imbalanced dataset problem. Indeed, in the end, a classifier should be able to classify correctly on the general dataset: it is trained on a subsample of the majority class, but in the end will be confronted to the real dataset where the classes are imbalanced again. Therefore, in this second experimentation, the training sets stay the same: two balanced datasets, one with two classes voluntarily built as *distant*, and another with a random sample of the majority class against the minority one. But the testing conditions are different, as the objective is now to see if this different training can improve classifier's performances on the imbalanced dataset. The testing set is therefore an imbalanced dataset, with all samples from r^- and Z that have not been used for training.

The conditions are exactly the same as previously with $|\mathcal{R}| = 12$ and $sf = 100$. Results are presented in table 2. They are less positive than the ones observed in 1: the difference between the two models is less pronounced, and seems to be more algorithm-specific.

	$ADOM(Z) = ADOM(r^+)$		$ADOM(Z) = ADOM(r^+ \cup r^-)$	
Classifier	r^- **vs** r^+	r^- **vs** s	r^- **vs** r^+	r^- **vs** s
Nearest Neighbors	0,70	0,72	0,68	0,71
Decision Tree	0,79	0,81	0,72	0,74
Random Forest	0,95	0,87	0,92	0,86
AdaBoost	0,75	0,78	0,70	0,73
Neural Net	0,66	0,70	0,66	0,77
Naive Bayes	0,83	0,78	0,94	0,75
QDA	0,75	0,89	0,80	0,85
RBF SVM	0,83	0,56	0,77	0,55
Linear SVM	0,99	0,99	0,99	0,99

Table 2. Accuracy of each classifier for each data generation strategy. Both models are evaluated on the same testing set, corresponding to data from an imbalanced datasets, with tuples from r^- and Z.

The results of this experimentations are mitigated, but also encouraging: considering the difficulty and novelty of the considered problem, more work is required, but this first study shows that there is interesting things to investigate. the choice of the values for the synthetic relations is a central problem, and it might be a potential source of improvement. Other types of data could also be used; if letters were used instead of integers, this might bypass the algorithms that rely more or the values.

6 Related work and conclusion

The work presented in this paper brings together two problems that do not seem to have been combined before: functional dependencies, a powerful notion in databases on the one hand, and the imbalanced datasets problem, which often occurs in classification, on the other hand. It could impact many application domains, from medical diagnosis to fraud detection. Many overviews on how to handle classification datasets can be found, with various methods relying on diverse computing and mathematical tools. Some solutions work at the data level and resample the data distribution to work on a balanced dataset, with undersampling of the majority class as exposed in this paper, or oversampling of the minority one [7]. Other solutions focus on the algorithms, by trying to adapt them for imbalanced datasets [12]. Finally there are solutions combining both data and algorithmic solutions for imbalanced datasets, such as cost-sensitive approaches [25], as well as boosting algorithms [11]. However some recent works try to address the imbalanced classes problem from a different perspective, like [23]: they use the Mahalanbois distance to create synthetic samples for the majority class.

Functional dependencies have proven to be powerful, in the design of databases [1], or data quality and data cleaning [5], or even to constrain the parameters of type classes in languages such as Haskell [16]. In this paper, we propose to use functional dependencies for the well-known problem of imbalanced datasets. If some results on the classification of distant sets are encouraging, they do not, for now, highlight if functional dependencies could significantly improve the results in imbalanced classification problems. It should however be kept in mind that this is only a first study, that addresses a completely new problem, that is both complex and difficult.

Further experiments are required in future works; studying other data generation strategy, and applying our approach to real data.

It could be argued that on real data, the approach presented could suffer from the lack of existence of functional dependencies in real datasets: however this can be tackled by releasing a bit the constraint of functional dependencies. This is a subject that has already been addressed abundantly in the literature, especially with the concept of fuzzy functional dependencies (FFDs) [15]. The application of this approach to real data will therefore be a logical continuation of this work, but is far from trivial and certainly raises a few combinatorial problems: given a set of functional dependencies, how to select the tuples in the dataset such that

they satisfy those FDs, or as much as possible of them ? Moreover, this first contribution opens the way to many more possibilities, as there are presumably other machine learning or data mining problems that could benefit from the use of functional dependencies.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases: the logical level*. Addison-Wesley Longman Publishing Co., Inc., 1995.
2. R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.
3. W. W. Armstrong. Dependency structures of database relationship. *Information processing*, pages 580–583, 1974.
4. C. Beeri, M. Dowd, R. Fagin, and R. Statman. On the structure of armstrong relations for functional dependencies. *Journal of the ACM (JACM)*, 31(1):30–46, 1984.
5. P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 746–755. IEEE, 2007.
6. A. Bonifati, R. Ciucanu, and S. Staworko. Interactive join query inference with jim. *Proceedings of the VLDB Endowment*, 7(13):1541–1544, 2014.
7. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
8. F. Chiang and R. J. Miller. Discovering data quality rules. *Proceedings of the VLDB Endowment*, 1(1):1166–1177, 2008.
9. J. Cumin, J.-M. Petit, V.-M. Scuturici, and S. Surdu. Data exploration with sql using machine learning techniques. In *International Conference on Extending Database Technology-EDBT*, 2017.
10. K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2842–2856, 2016.
11. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
12. V. Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47, 2012.
13. J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
14. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
15. L. Jekov, P. Cordero, and M. Enciso. Fuzzy functional dependencies. *Fuzzy Sets and Systems*, 317(C):88–120, 2017.
16. M. P. Jones. Type classes with functional dependencies. In *European Symposium on Programming*, pages 230–244. Springer, 2000.
17. G. O. Katona, A. Keszler, and A. Sali. On the distance of databases. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 76–93. Springer, 2010.

18. S. Kotsiantis, D. Kanellopoulos, P. Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.
19. S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.
20. M. Levene and G. Loizou. *A guided tour of relational databases and beyond*. Springer Science & Business Media, 2012.
21. H. Müller, J.-C. Freytag, and U. Leser. Describing differences between databases. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 612–621. ACM, 2006.
22. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
23. S. Sharma, C. Bellinger, B. Krawczyk, O. Zaiane, and N. Japkowicz. Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. 2018.
24. Y. Shen, K. Chakrabarti, S. Chaudhuri, B. Ding, and L. Novik. Discovering queries based on example tuples. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 493–504. ACM, 2014.
25. B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 435–442. IEEE, 2003.
26. B. Zou, X. Ma, B. Kemme, G. Newton, and D. Precup. Data mining using relational database management systems. In *Pacific-asia conference on knowledge discovery and data mining*, pages 657–667. Springer, 2006.