Guided data selection for predictive models

Marie Le Guilly

N°d'ordre NNT : 2020LYSEI072

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
INSA de Lyon

Ecole Doctorale N° 512 InfoMaths

Spécialité : Informatique

Soutenue publiquement le 24/09/2020 par :
Marie Le Guilly

Guided Data Selection for Predictive Models

Devant le jury composé de :

CORNUEJOLS, Antoine	Professeur des Universités, AgroParisTech	Rapporteur
LINK, Sebastian	Professor, The University of Auckland	Rapporteur
BIDOIT-TOLLU, Nicole	Professeur des Universités, Université Paris Saclay	Examinatrice
HACID, Mohand-Saïd	Professeur des Universités, Université Lyon 1	Examinateur
PETIT, Jean-Marc	Professeur des universités, INSA Lyon	Directeur de thèse
SCUTURICI, Vasile-Marian	Maitre Maître de conférences, HDR	Directeur de thèse

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** http://www.edchimie-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr INSA : R. GOURDON	**M. Stéphane DANIELE** Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX directeur@edchimie-lyon.fr
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** http://edeea.ec-lyon.fr Sec. : M.C. HAVGOUDOUKIAN ecole-doctorale.eea@ec-lyon.fr	**M. Gérard SCORLETTI** École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 gerard.scorletti@ec-lyon.fr
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** http://e2m2.universite-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES secretariat.e2m2@univ-lyon1.fr	**M. Philippe NORMAND** UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX philippe.normand@univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** http://www.ediss-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Curien - 3ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tel : +33(0)4 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** http://edinfomaths.universite-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tel : 04.72.44.83.69 hamamache.kheddouci@univ-lyon1.fr
Matériaux	**MATÉRIAUX DE LYON** http://ed34.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction ed.materiaux@insa-lyon.fr	**M. Jean-Yves BUFFIÈRE** INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 jean-yves.buffiere@insa-lyon.fr
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** http://edmega.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX jocelyn.bonjour@insa-lyon.fr
ScSo	**ScSo*** http://ed483.univ-lyon2.fr Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 veronique.cervantes@univ-lyon2.fr	**M. Christian MONTES** Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 christian.montes@univ-lyon2.fr

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Abstract

Databases and machine learning (ML) have historically evolved as two separate domains, even if proposing approaches at their intersection is getting more and more mainstream. However, they are still dedicated to two complementary yet different tasks: roughly speaking, while databases are used to store and query the data, machine learning uses the data to perform analysis, build predictive models, detect patterns etc. As a result, when confronted to a new problem for which machine learning techniques can be applied, the data is usually extracted from the database, and processed outside of it. This extraction process may seem trivial, as it consists in writing a single query, often in SQL, to select the relevant data. It is therefore often underestimated, but this data preparation phase is known as the main bottleneck in machine learning projects, as many issues can arise: ill-defined database schema, data integration problem, incomplete data, etc. All these issues can have an important impact on the performances of the final machine learning models, and we raise and address some of them in this thesis.

First, the database from which the data is extracted usually contains more than the data of interest: the first question is how to separate the data that the analyst wants from the one she does not want? Interestingly, this can be seen as a classification problem, where the task is to build a model that predicts if a tuple is of interest or not. But in such a configuration, the dataset is likely to be imbalanced: the proportion of interesting data is often very small with respect to the amount of available tuples in the database. There exists several solutions to deal with imbalanced data in machine learning. In this thesis, we develop alternative approaches by taking advantage of databases' constraints such as functional dependencies (FDs). We therefore propose an undersampling approach based on the FD-distance between the minority and majority classes, using notions related to closure systems.

However, by considering the data extraction only as a classification task, several problems can appear. First, it requires to have labelled tuples, a task that can only be performed manually by the analysts that knows what data is of interest for her considered task. Additionally, it is important to be able to describe the selected data, to understand what is represents and to make sure it makes sense with the analyst's requirements. Considering a relational database, this description should be a SQL query, so that it can be easily used, refined, and stored for future similar applications.

To this end, we propose a SQL query completion solution, that starts from a very general query, and helps an analyst to refine her SQL query until she selects her data of interest, without requiring her to manually label individual tuples. This process aims at helping the analyst to design the query that will eventually select the data she requires.

Then, assuming the data has successfully been extracted from the database, the next natural question is: is the selected data suited to answer the considered ML problem? This is a broad question, that can be extremely difficult in practice. If the objective is to build a predictive model, is it possible to say if the selected data is consistent with the task at end, and will it allow the model to have satisfying performances? Since getting a predictive model from the features to the class to predict amounts to providing a function, we point out that it makes sense to assess the existence of that function in the data. This existence can be studied through the prism of functional dependencies. The objective is to decide if the learning process can start, or if more time should be spent on the data selection process. We show how the dataset can be explored to understand the limitations of the model when the performances are not satisfying. Additionally, we question how to refine the data selection in the situations for which the initially selected data does not seem adequate with the considered predictive task, by analyzing counterexamples, i.e pairs of tuples preventing the existence of a function. To this end, we consider the context for which the data is selected, and identify additional selection predicates to select the data that is the most coherent with this context.

Résumé

Historiquement, les bases de données et l'apprentissage automatique sont deux domaines qui ont évolués séparément, même s'il est désormais de plus en plus courant de proposer des approches à leur intersection. Cependant, ces deux domaines sont dédiés à des tâches différentes, bien que complémentaires: tandis que les bases de données sont utilisées pour stocker et interroger les données, l'apprentissage propose des analyses de ces données, construit des modèles prédictifs, etc. Ainsi, pour un problème d'apprentissage, les données sont généralement extraites de la base, et traitées à l'extérieur de celle-ci. Ce processus d'extraction peut sembler trivial, puisqu'il s'agit simplement d'une requête, souvent en SQL, pour sélectionner les données pertinentes. Les difficultés inhérentes à ce procédé sont donc souvent sous-estimées, et cette phase de sélection des données est souvent chronophage, en raison des différents problèmes qui peuvent survenir: schéma mal spécifié, problème d'intégration des données, données incomplètes, etc. Toutes ces problèmes peuvent ensuite avoir un impact sur les performances du modèle d'apprentissage. Dans cette thèse, nous proposons d'étudier les problèmes relatifs à cette sélection de données et leurs impact sur les modèles prédictifs, en se basant sur une base de données relationnelles, interrogée en SQL.

Tout d'abord, la base de laquelle sont extraites les données contient certainement plus de données que ce qui est nécessaire pour le problème considéré: ainsi, la première question est de déterminer comment séparer ce que souhaite l'utilisateur du reste de la base. Nous proposons de voir ce problème comme de la classification, où il faut séparer les tuples "intéressants" des autres. Plus précisément, puisqu'il est probable qu'il y ait bien plus de tuples dans la base que ce que souhaite l'utilisateur, il s'agit d'un problème de classification déséquilibrée. De nombreuses solutions existent alors pour ce problème bien connu en apprentissage: nous proposons une alternative, en tirant parti des contraintes de la base de données, et plus particulièrement de ses dépendances fonctionnelles (DF). Nous proposons donc une approche de sous-échantillonnage basée sur une distance via les DFs, via les systèmes de fermeture.

Cependant, la sélection de données n'est pas uniquement un problème de classification: en effet, il est aussi important de pouvoir décrire les données sélectionnées,

pour les comprendre et valider la sélection. Dans une base de données, il est important d'avoir une requête SQL, qui peut être modifiée et réutilisée. Ainsi, nous proposons une forme de complétion de requêtes SQL, sous la forme d'extensions de requêtes qui permettent de mieux spécifier une requête très générale, jusqu'à arriver aux tuples désirés.

Enfin, une fois les données sélectionnées dans la base, il est légitime de se demander si elle permettent réellement de construire un modèle prédictif. Nous nous posons donc la question d'évaluer l'adéquation entre des données et la tâche d'apprentissage pour laquelle elles ont été sélectionnées. Puisqu'un modèle prédictif cherche à définir une fonction entre les attributs et la classe à prédire, nous proposons d'évaluer l'existence de cette fonction dans les données, via les dépendances fonctionnelles. L'objectif est de déterminer s'il fait sens de construire un modèle à partir des données, ou s'il est nécessaire d'affiner la sélection. Nous montrons comment comprendre les limitations d'un modèle, notamment en déterminant les contre-exemples qui empêchent la dépendance fonctionnelle entre les attributs et la classe d'être satisfaite. Nous étudions enfin comment raffiner la sélection des données, en prenant en compte le contexte d'apprentissage.

Acknowledgement

First of all, I would like to thank the two reviewers of this thesis, Sebastian Link and Antoine Cornuejols: your insights on the manuscript and during the defense were extremely valuable. I would also like to thank Nicole Bidoit-Tollu and Mohand-Said Hacid for your presence during the defense and during my PhD committee.

I also have to express my deepest gratitude to my two supervisors, Jean-Marc Petit and Marian Scuturici. You've trusted me since my first internship with you, and taught me a lot. This three years of PhD have been intense and full of twists, but I'm glad that you were by my side to guide me through all this ! I'm grateful for all of our scientific discussions, for your support every time one of our paper was rejected, and more generally for your kindness and support over the last years.

More generally I would like to thank the members of the LIRIS lab, with a special thank to the database team (BD), for providing a rich and stimulating research environment. I'm also grateful for everyone I had the pleasure to teach with at INSA Lyon, an activity I've deeply enjoyed, and that allowed me to get my out of my research from time to time. A big thank you to the administrative team for always being helpful. I also thank the DataValor team, and to all the people I met during the industrial collaborations: your insights have clearly been an important part of this manuscript.

I also owe a huge thank you to the fellow PhD students that crossed my path during these few years: Tarek, Hind, Chabha, Maelle, Amine, Julien, Romain V., Diana, Florian, Aimene, Paul, Mohamed, Yann, Rémi, Adnene , Romain M., Lucas,... I missed our daily lunch talks, and I'm sad covid did not give me the oppportunity to enjoy them in the final months of this PhD. Also thank you to all the intern I've met and sometimes supervised in these three years. And thank you to everyone I met during my stay in Waterloo, with a special thank to Mina, Michael and Melica.

Finally, I would like to thank my friends, for their presence during this PhD, and Julie D., for the guidance towards the end of this thesis. I also owe everything to my family: I'd like to thank my parents for their help and support, in every sense of the word, and for always encouraging me to go further during my studies. Thank you also to my brothers, Vincent for you encouragement, presence, and for always making me laugh (even during my defense) ; Thibaut also, especially for showing

me the way towards a PhD, and for motivating me to be a doctor like you. I should also thank the Dimech's family for the support over the last eight years !

Finally, I have to address the final and biggest thank you to Maxime, who has been here every step of the way, and has been an incredible support for me: this thesis would clearly not be here without you. I'm incredibly thankful to have you by my side, and I can't wait for you to be my husband. Also I guess I should also thank our cat Ella, who arrived in the middle of the PhD, clearly improved our everyday life, and helped me to calm and relax while I was writing this thesis in the middle of a global pandemic.

List of publications

International Conferences

- Marie Le Guilly, Nassia Daouayry, Pierre-Loic Maisonneuve, Ammar Mechouche, Jean-Marc Petit and Marian Scuturici. Contextualisation of Datasets for better classification models: Application to Airbus Helicopters Flight Data, *24th European Conference on Advances in Databases and Information Systems (ADBIS)*, 2020, 10 pages.

- Marie Le Guilly, Claudia Capo, Marian Scuturici, Jean Marc Petit, Rémi Revellin, Jocelyn Bonjour, and Gérald Cavalier. Attempt to better trust classification models: Application to the Ageing of Refrigerated Transport Vehicles. *25th International Symposium on Methodologies for Intelligent Systems (ISMIS'20)*, 2020, 10 pages.

- Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici and Ihab Ilyas. ExplIQuE: Interactive Databases Exploration with SQL (demo paper). *28th ACM International Conference on Information and Knowledge Management, CIKM 2019, China*, 2019, 4 pages.

International conferences with abstract-based selection

- Marie Le Guilly, Claudia Capo, Léo Pape, Marian Scuturici, Jean Marc Petit, Rémi Revellin, Jocelyn Bonjour, Gérald Cavalier. Ageing of refrigerated transport vehicles: development of a numerical predictive model. *6th IIR conference on sustainability and the cold chain*, 2020, 9 pages.

- Marie Le Guilly, Jean-Marc Petit and Vasile-Marian Scuturici. A First Experimental Study *on Functional Dependencies for Imbalanced Datasets Classification. Communications in Computer and Information Science » series, Springer, Revised Selected Papers from the International Workshop ISIP*, 2018, 17 pages.

Journals

- Marie Le Guilly, Jean-Marc Petit and Vasile-Marian Scuturici. Evaluating classification feasibility using functional dependencies. *Transactions on Large-*

Scale Data and Knowledge-Centered Systems (TLDKS), Revised selected paper from BDA'2019, 2019, 27 pages.

National Conferences

- Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici. Evaluation de la faisabilité de classification en utilisant les dépendances fonctionnelles. *BDA 2019 35ème conférence sur la Gestion de Données — Principes, Technologies et Applications.* 2019, 12 pages.
- Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici, Ihab Ilyas. ExplIQuE : Exploration Interactive de Bases de Données en SQL (démo). *BDA 2019 35ème conférence sur la Gestion de Données — Principes, Technologies et Applications.* 2019, 4 pages.
- Marie Le Guilly, Ihab Ilyas, Jean-Marc Petit and Vasile-Marian Scuturici. Partitioning queries for data exploration using query extensions. *BDA 2018 34ème conférence sur la Gestion de Données — Principes, Technologies et Applications,. 2018*, 12 pages.
- Marie Le Guilly. Langages de requêtes interactifs pour l'exploration de données (Article Doctorant). *BDA 2017 33ème conférence sur la Gestion de Données — Principes, Technologies et Applications, 2017*, 2 pages.
- Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici. Retour d'expérience sur l'analyse des données d'un tunnelier. *BDA 2017 33ème conférence sur la Gestion de Données — Principes, Technologies et Applications.* 2017, 6 pages.

Contents

1 Introduction

1.1 Databases and Machine Learning

The term "relational database" was first introduced in 1970 by Edgar F. Codd, in his paper entitled "A relational model of data for large shared data banks" [Cod02]: since then, relational databases, with databases management systems (DBMS), have become a standard for storing and querying data, especially using SQL (structured query language) to store, access, query and select the data that a database user is looking for.

In parallel, as the volume of data recorded and stored has been continuously increasing, there has been a growing interest in machine learning techniques, that aim at building algorithms that can improve through experience, by *learning* from the data they are provided: machine learning is therefore a way to gain value out of data, to better understand phenomena, build predictive models, identify patterns in the data, etc. Over the last decades, many tools such as machine learning librairies and platforms have therefore been developed to make it easier to apply such techniques, that are now widely used by companies in their daily processes, for all sorts of application, from medical diagnosis to banking investment prediction, defining a new filed which is roughly refered to as *artificial intelligence*.

Because they are both centered around data processing, relational databases and machine learning are two closely related domains, that are involved at different stages of the same process. But in practice, there is not so much overlapping between the two, and they have historically grown as two separate research domains. On one side, databases are used to store and query the data. Afterwards, when a machine learning algorithm is applied, the data is extracted from the database, and processed entirely externally, without any new interaction with the database. In this philosophy, it can be roughly said that, in practice, the database is mainly used as a data container, and the extraction process as a simple transformation from relational tables to a CSV file. Even if research is now working on more integrated solutions, many concrete applications in companies are based on this pattern.

From this general observation, the broad starting point of this thesis was to propose solutions to increase the cross-fertilization between relational databases (DB) and machine learning (ML), by proposing new ideas at the intersection of the two domains. Such questions on this intersection of DB and ML have been raised in the last decades; in 1996, Mannila and Imielisky introduced the term of *database mining* [IM96], and argued in favor of the seeing data mining as a querying process:

> " It is, of course, important work to close the gap between the inductive learning tools and the database systems currently available. "

In this paper, they argue in favor of using all the performance enhanced that have been developed for databases in order to also improve the performances in terms of knowledge discovery. Similarly in [Cha98], Surajit Chaudhuri question the intersection of databases and data mining:

> "This raises the question as to what role, if any, database systems research may contribute to area of data mining"

S. Chaudhuri argues that bringing databases and machine learning algorithms closer might only be beneficial in terms of performance. These two approaches relate to the notion of *inductive databases* introduced by Luc de Raedt [DR02], that would not only store data objects but also relations and patterns between those objects. Several solutions have been proposed that go in the direction claimed in these papers: for example in [Zou+06], the authors have implemented an entire machine learning library so that the algorithm can work with data stored in a DBMS rather than in main memory. Alternatively, ATLAS [Wan+03] offers a SQL extension to perform in-database data mining. More recently, [KR18] introduced a relational framework for classifier engineering. Moreover, new solutions have been developed to propose in-database learning, by relying on key concepts from relational databases, such as data dependencies, to perform and optimize learning models directly in the DBMS [Sch+16; AK+18].

All the aforementioned approaches aim at merging databases and machine learning (or data mining), mainly by using key concepts from the database to optmise the in-database learning of models, but the other way around is also possible: [Wan+16] proposes a survey showing how deep learning and DBMS can benefit from each other, in one direction or the other. Indeed, there also exists many approaches that take another angle by using machine learning to address database problems. It is for example very mainstream to build model to provide data exploration and assist users in writing their queries (see for example [Bon+14; Tra+09; She+14]), or to use recurrent neural networks to translate natural language sentences into SQL

queries [Aff+19; Zho+17; Sah+16]. But other database-related problems can also be addressed by ML algorithms, such as query plan generation [Mar+19], join order selection [MP18; Tru+19], cardinality estimation [Liu+15; Ort+19], etc. Machine learning has even been used for automatic DBMS configuration tuning [VA+17].

In this context, the overall objective of this thesis is to explore these two possible directions, by considering how relational databases and machine learning can benefit from one another. We propose to consider the traditional setting in which these two domains are naturally used together in the same process, which is the building of a predictive model, using a ML algorithm and training data stored in a relational database.

1.2 Research questions

1.2.1 Overview

As we focus on the building of a predictive model, let's start by considering how it usually works in a traditional setting. The process is usually divided into two distinct steps:

- The process starts with extracting the data from the DBMS in which it is stored: this is done with one or several SQL queries, and the results in stored in a single file (usually CSV).
- Then the model is built, by learning from the data contained in the file, and the traditional ML worflow is applied (training, testing, validation, vizualisation, etc).

This illustrates the main functionalities attributed to each domain: DMBS are for storing and querying, ML for models' construction. But these two simple steps hide much more difficult questions, and in practice, it is not so easy to extract the desired data, so that it corresponds to the data necessary for the considered predictive model. Actually, in turns out in practice to be the main bottleneck in many ML processes, as around 80% of the time is dedicated to data preprocessing [Scu+15]. This is due to many different factors: the data quality might not be satisfying, the schema might not be correctly specified, there might be missing values, etc. As a result, writing a SQL query is hard, especially because the reality of the available data is usually far from the expectations and the needs for the considered predictive task. Additionally, in this thesis we limit the scope to a single database, and do not consider the case of

data scattered among different databases, requiring to perform schema mapping or more generally data integration (see [Roh+19] for an overview): these additional issues are also extremely time consuming.

As a result, in this thesis, our main focus is on this transition from the database to the training model, by first enumerating the various research questions that arise in this context, and then addressing each of them with solutions proposal. As a result, the general problem that is the guideline of this thesis can be summarized by the following question:

> How to select in relational databases the most appropriate data for the building of a considered predictive model?

This question is heavily motivated by observations from the processes that take place in many companies' workflow for predictive task, and that was observed in several industrial collaborations that took place during this thesis. Several contributions have been developed from problems encountered in these industries: Dodin Campenon Bernard (tunnel boring machine), Cemafroid (refrigerated transport vehicles), and Airbus Helicopters. During all these collaborations that all concerned different predictive tasks, similar problems appeared around the data selection. Indeed, in practical context, there is a very high volume of available data, from which it is sometimes hard to select. For example, it can be hard to have access to the raw data, because it is scattered among various databases, and is therefore concatenated already using SQL queries that cannot be changed or accessed. Sometimes, the data is managed by an external service provider, making it difficult to have access to all the necessary data: in some cases, the company has to pay the provider for each new SQL query, making it difficult (and costly) to modify the initial data selection. All these observations made in practical settings highlighted the difficulty to control the available data, and the necessity to develop techniques to make the most of what is available to domain experts. For this reason, we consider that the available data is sometimes far from databases experts expectation, and is sometimes more like a single CSV file dumped into a table, than a nice well-defined database with clean constraints.

Moreover, it should be underlined that such tasks are at the intersection of ML and DB, and are therefore highly dependent on the user's level in each field: database experts might have less knowledge on ML and vice-versa, because they come from different training backgrounds. Therefore, a specialist of one filed is likely to have more basic knowledge of the other. Additionally, it is easy to think that the other field is easy, because the tricky problems appear only when diving more deeply into the process.

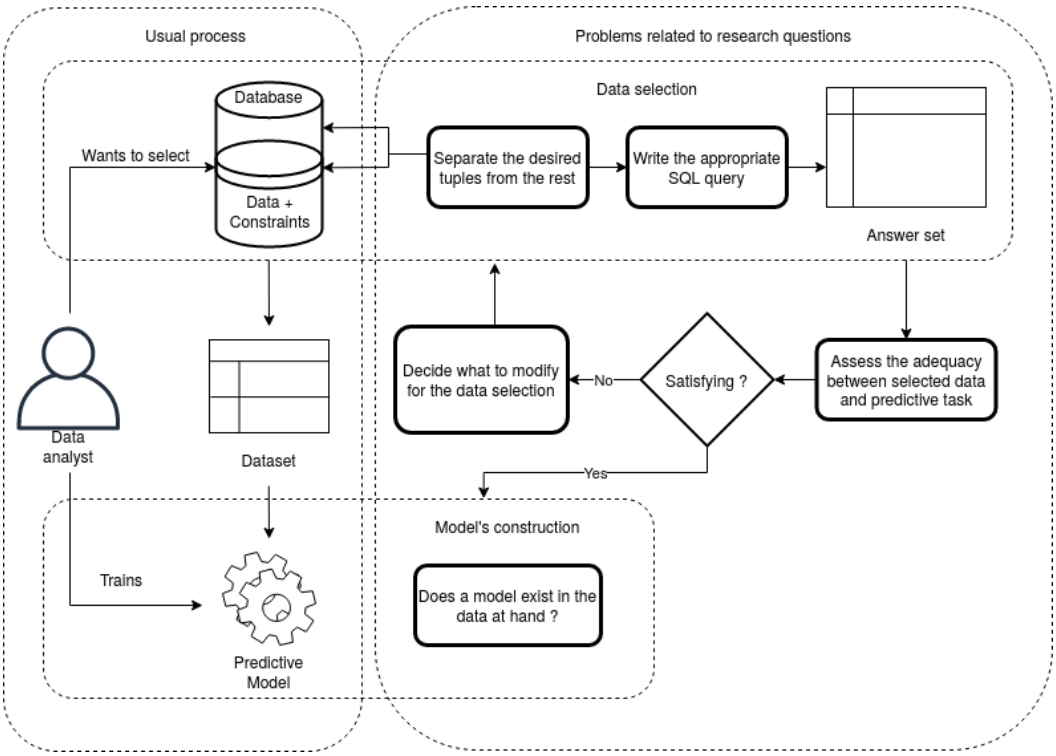


Fig. 1.1: Integration of the research questions in the traditional process of data selection for the construction of predictive models

From all of these considerations, we defined a general framework, that is traditionally followed by database users to select the data in such contexts: they start with a query to obtain a set of tuples from the relational database, and use these data to train (and then test and validate) a predictive model: the relation obtained with a SQL query therefore becomes a dataset[1] for a ML process. This "simple" process is represented on the left part of figure 1.1. But this linear workflows holds many questions, that the user is likely to ask herself:

- In the relational database, how can she selects the relevant data she **wants**?
- How can she know she selected the data she **needs** for her predictive task?

These two questions are easy to formulate, but much more trickier to answer: they indeed hide many research questions, summarized on the right part of figure 1.1: each bold rectangle represents one of these questions, addressed in this thesis. We now detail each of the aforementioned user's questions, and link them to the problems represented on figure 1.1.

[1] In the remainder of the thesis, when clear from context, we will use both terms *relation* and *dataset* to designate the set of tuples selected in the database.

1.2.2 Selecting the desired data

Let's start with the first question, that represents the upper part of figure 1.1: considering a relational database, the user is trying to select part of it to use as training data. To this end, she is likely to have a general idea of what are the tuples she wants. But to get these tuples requires her to perform two intermediate steps:

- First, she needs to define what are the tuples she is looking for
- Then, she needs to formulate the SQL query that retrieves these tuples

These two steps are tightly linked together, and their difficulty varies according to different factors: size of the database, schema's complexity, user's SQL experience, etc. It is therefore important to propose general solutions adapted to these different contexts, and to keep in mind the complexity of databases that, in the current *Big Data era*, are only getting bigger. Because they are bigger, the overall quality of the database can be impacted, by ill-defined schemas, large tables, useless names of columns, etc.

In addition, because getting value out of data now appears as a priority in many industrial applications, more and more users are confronted to the databases, some with only a limited experience of data processing and querying: in this context, the above questions are even more complex and important.

Identifying the tuples

Considering the first question, it appears that before writing a SQL query, it is important to have an idea of what are the tuples the user is looking for, and more importantly what distinguishes them from the other tuples in the database: being able to separate the desired tuples from the others is what makes it possible to later write a SQL query, with selection conditions that capture the desired characteristics. This problem is well-known and relates to *query-by-example* approaches [Lis+18], as it seeks to start from the data to identify the tuples of interest, before writing the corresponding query in a second time, a problem also known as *query reverse engineering* [Tra+14].

It is important to underline that the set of tuples a user is trying to select is usually several orders of magnitude smaller than the total number of tuples in the database. As a consequence, the valuable information is often difficult to locate, because it is hidden among all the other tuples that are not of interest. From these considerations,

our proposition in this thesis is to see this problem as an imbalanced classification one, for which the objective is to classify between the tuples of interest (the minority class) and the others (the majority class). However, this imbalanced classification problem is very specific, because it takes place in the context of a relational database, so that the data is likely to have some specificities, such as, for example, data dependencies that might exists in the DB to due schema's normalization, of thanks to domain experts specifications. We therefore ask the following research question:

> Given a dataset obtained from a SQL query, can we use the functional dependencies that hold in the database to improve the classification between relevant and non-relevant tuples?

In this thesis, we propose to consider this problem from a theoretical point of view by restricting ourselves to a single relation, and by studying various conjectures on synthetic dataset. The ultimate goal is to assess whether or not such an approach can be extended to real-life datasets to improve the identification of relevant tuples in a database.

Writing the SQL query

However, being able to identify the relevant tuples is not enough, and it is important to be able to select them using a SQL query. First, such a query is a characterization of the selected tuples, helping her to better understand the selected data. Moreover, having a query allows her to modify some of the selection conditions, and to therefore explore the surroundings of the selected data, to make sure she has considered all possibilities, and has therefore selected all the tuples she needs: this is especially useful if the user is not entirely confident of her selection, and does not necessarily know immediately how to write her final SQL query. Finally, having a SQL query makes it possible to reuse it later, if the database has been modified and/or enriched with additional tuples: in this scenario, using the same query, the user is able to catch the new tuples that correspond to the same selection conditions, and to therefore improve her selected data with tuples from this new instance of the database.

It is not always easy to come up with the final SQL query selecting all the desired data at once. Therefore, in this thesis, our proposition is to guide the user to write a query iteratively until she reaches exactly the data she is looking for. We therefore consider to following question:

> How to identify the necessary selection clauses to capture all the tuples the user is looking for?

This relates to classic databases issues on query writing, that are often addressed recently with natural language processing (NLP) solutions [Aff+19]. Answering this research question also requires to combine the query specification with some data exploration [Idr+15], to better grasp the selection condition that characterize what the user is looking for, and to remove any tuples that she does not want. In this thesis we therefore propose an iterative solution to query writing, by suggesting additional selection predicates, called extensions, that allow the user to specify an initial general query: the purpose of such extensions is to both refine the query to reach the desired tuples, and to better understand the currently selected tuples, to assess whether or not the data selected is what the user is looking for. Such extensions are a form of query completion, as they allow a user to complete her query when she is stuck and does not know what to add to her query.

1.2.3 Data adequacy with the selected model

Once the the appropriate tuples are selected, the user should be able to obtain a relation containing all the tuples is looking for, which is the answer to the query she has managed to write. This is where the second main question of this thesis starts, related to the the predictive model the user is trying to build. This question represents all the bottom part of figure 1.1, and can be reformulated as follows:

> Is the selected data, identified by the user, adequate to answer the considered predictive task?

In this question, the adequacy of the data requires to evaluate if the selected data is the one that will allow to built the most qualitative predictive model. It is a broad question, known to be hard in machine learning and statistics. If data selection is difficult, it is even trickier if it has to be optimized with respect to the predictive model to be built. As a consequence, this initial problem suggest two more questions:

- If the selected data seems inadequate, how to identify what should be modified to better select the data with respect to the predictive task
- If the data seems adequate, then the model can be trained. However, the data used to train the model has a huge impact on the performances and limitations of the dataset: based on the selected data, how to identify and explain such limitations?

In total, we therefore identified three main research questions, highlighted on figure 1.1. To answer these questions, we propose to start from the same consideration for all of them: the purpose of predictive models is to approximate a function, that maps the features (the data used to make the prediction) to the value to predict. Based on this, it appears necessary for the training data to correspond to a function, so that the predictive algorithm can try to infer it. Therefore, to assess the adequacy of the selected data to the predictive task, it is possible to evaluate the existence of a function between the selected features and the class. This can be seen as a way to avoid building models on data in which there does not exists one: it's always possible to run an algorithm, but what it produces should make sense and be representative of an existing phenomenon in the data. The following question can therefore be asked:

> How can we assess whether or not the selected data follows a function so that it makes sense for an algorithm to define one?

Such a question is related to estimating a model's performances before building one, only by analyzing the data available to train the model. In this context, Bayes error [Fuk13] is a way to estimate classifier's accuracy for a given problem: but this upper bound turns out to be very complicated to compute in practice.

In this thesis, we propose to evaluate the satisfaction of the functional dependency between such features and the class: if this dependency is satisfied (or satisfied enough), then it can be assumed that the selected data is adequate, because there exists a function in the data, that an algorithm can seek to approximate. Otherwise, the data selection might have to be refined.

Regarding such refinement of the data selection, we propose to analyze what, in the selected data, prevents the functional dependency from being satisfied: by understanding what are the tuples that prevents this satisfaction, it is possible to construct selection clauses that can be used to remove these tuples.

Moreover, if the functional dependency is considered to be enough satisfied in the data for a model to be trained, or if additional data selection is not possible,it is also possible to use the few tuples that are not consistent with the dependency to explain the limitations of the models due to the data it is trained on.

1.3 Contributions

To summarize the contributions of this thesis can be divided into two main parts:

1. Assisting users in data selection. For this first part, we propose two main contributions:

    - First, we aim at separating the tuples the user is looking from all the other tuples in the database, by considering this problem as imbalanced classification. To this end, we propose to exploit the database setting, by investigating whether or not the relation's dependencies can be used to better classify between the minority and majority classes. We study the influence of the distance between the two classes, using a distance measure that is based on the functional dependencies satisfied by each class. More precisely, we perform an undersampling of the majority class based the closure systems of each class. To this end, we generate synthetic datasets corresponding to this setting, and compare the performances of various classification algorithms on such datasets compared to random sampling. This work has been presented at ISIP international workshop, and published in their post-proceedings [Gui+18a].

    - We then question the formulation of a SQL query to help the user select the desired data, by proposing an iterative approach based on query extensions. These are additional selection predicates, that can be used by the user to better understand the data selected by her initial query, and to refine it in order to only select the tuples of interest. We expose a use case that motivated the creation of such extensions, and develop an algorithm to compute extensions for a given SQL query, as well as vizualisations to assist the user in choosing an extension. We propose a web application to use this approach, validated with scaling experimentations, as well as a user validation with 70 participants. This work has been presented at the BDA2018 national conference [Gui+18b], and a demo has been proposed at CIKM2019 [Gui+19].

2. Evaluating the adequacy of the selected data with the predictive task. For this, we rely on our approach based on the existence of a function between the features and the class in the training dataset, based on the satisfaction of the corresponding functional dependency. We focus on classification problems, and explain some leads to generalize the approach for regression cases. For this, we propose three contributions:

    - Based on the G_3 measure [KM95] of the functional dependency, we point out how it can be used to qualify the adequacy of the data for the considered classification task. We show that this value can be used as an upper bound to classifier's accuracy on the considered data. We validate

this approach on both synthetic and real classification dataset. These results have been exposed at BDA2019 [LG+19]. We also discuss how to deal with data that require to use non-crisp functional dependencies, and how this impact our results.

- When the data is judged adequate to go on with the classification task, or if data selection cannot be refined, even though the dependency might not be entirely satisfied, we propose to use the analysis of the tuples preventing the dependency from being satisfied, in order to better understand the limitations of the model. We illustrate this approach with a use case on the data from Cemafroid, a company dealing with refrigerated transport vehicles. The results from this collaboration showed how important it is to discuss counterexamples with domain experts, i.e to analyze what are the tuples that prevent the FD from being satisfied. A paper presenting the results of this study has been accepted at ISMIS2020 [Leg].
- When data selection has to be refined, we propose to use the current data and the tuples preventing the dependency's satisfaction to better define what is the data that should be kept, because it is coherent with the existence of a function in the data. We propose a systematic methodology to identify a coherent *context* in the data, in which the dependency is more satisfied, while allowing to select as many tuples as possible. We illustrate this contribution with a use case on Airbus Helicopters data, on which we compared our contextualization approach to the one manually performed by Airbus' experts: these results will be presented at ADBIS2020 [LG+20].

1.4 Outline

We start by introducing all necessary preliminaries in chapter 2. Chapter 3 exposes our proposition to consider data selection as an imbalanced classification problem. Then, we develop how to write the SQL query selecting the desired data using query extensions in chapter 4. We move on to the adequacy of the selected data for the predictive task in chapter 5, before concluding this thesis in chapter 6.

2 Preliminaries

Chapter's outline

In this chapter, we introduce the notions that will be used in the following chapters of this thesis. These notions are used in the different contributions, in order to propose new solutions regarding data selection for predictive models.

We start by introducing general notations related to relational databases, in section 2.1. We then move on to functional dependencies, and to all the concepts related to them (closure systems, Armstrong relations, etc), and we discuss how to evaluate the satisfaction of an FD in section 2.2. Finally, we introduce the preliminaries regarding predicting models, and discuss the evaluation of their performances and limitations in section 2.3.

2.1 General databases notations

We first recall basic notations and definitions on relational databases. It is assumed that the reader is familiar with databases notations (see [LL12]).

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 $D^{|R|}$. Let $X \subseteq R$. The restriction of a tuple t to X is denoted by $t[X]$.

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.

In the sequel, letters from the beginning of the alphabet $(A, B, C, ...)$ denote a single attribute, while letters from the end of the alphabet denotes union of attributes. When clear from context, set $\{A, B, C\}$ is refered to as ABC, while XY is equivalent to $X \cup Y$.

We consider the SQL and the relational algebra query languages without any restriction. We will switch between both languages when clear from context. A query Q is defined on a database schema R and $ans(Q, d)$ is the result of the evaluation of Q against d. To define the extensions of any query Q, we will use two operators: π_X the projection defined as usual with $X \subseteq \mathcal{U}$, and σ_F the selection, where F is a conjunction of atomic formulas of the form $A\theta B$ or $A\theta v$, with $A, B \in \mathcal{U}$, $v \in \mathcal{D}$ and θ a binary operator in operation in the set $\{<, >, \leq, \geq, =, \neq\}$.

2.2 Functional dependencies

2.2.1 General definition

Definition 1. *We now define the syntax and the semantics of a Functional Dependency (FD). Let R be a relation schema, and $X, Y \subseteq R$.*

Syntax *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.

Semantic *$X \rightarrow Y$ is satisfied in r, denoted by $r \models X \rightarrow Y$, if and only if:*

$$\text{for all } t_1, t_2 \in r, \text{ if } \forall A \in X, t_1[A] = t_2[A] \text{ then } \forall B \in Y, t_1[B] = t_2[B]$$

2.2.2 Closure systems

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 $IRR(F)$, such that:

$$\text{for all } X, Y, Z \in IRR(F), \text{ if } X \cap Y = Z, \text{ then } X = Z \text{ or } Y = Z.$$

Note that we have $IRR(F) \subseteq CL(F) \subseteq P(R)$, where $P(R)$ denotes the powerset of R.

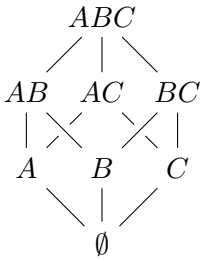


Fig. 2.1: Poset of closures for schema $\mathcal{R} = ABC$

Example 1. *Let's consider a schema $\mathcal{R} = ABC$. Then $|R| = 3$. The poset of closures for $\mathcal{R}$ is presented on figure 2.1: it consists of 6 elements ($2^3 + \mathcal{R} + \emptyset$) distributed over 4 levels.*

We also denote for a set S of attributes over R its closure by intersection $S^{\cap}$ as:

$$S^{\cap} = \{X \in P(R) | X = \bigcap S', S' \subseteq S\}$$

Finally, the concept of Armstrong relations [Arm74] allows to define relations satisfying a set of functional dependencies, and only those dependencies.

Definition 2. *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 [Bee+84]. It requires agree sets 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, its 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 [Bee+84]:

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:*

$$IRR(F) \subseteq ag(r) \subseteq CL(F)$$

2.2.3 FD-based distance between relations

It is possible to use the dependencies that hold in two relations to define the distance between them: this was proposed by Katona and Sali in [Kat+10]. Formally, the distance between two databases, instances of the same schema, is defined as follows:

Definition 3. *[Kat+10] Let r_1 and r_2 be two relations over 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, and defined as follows:

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

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 in the instances of the database, this distance characterizes the similarity of two sets of functional dependencies. From definition 3, it follows that distant relations tend to have opposite FDs, while close ones will have similar or even shared FDs. The symmetric distance is used here to identify how different the two closure systems are: the more distant they are, the less elements they have in common, and therefore the higher the symmetric distance.

Moreover, it is possible to know the largest possible distance between two relations based on the size of their schemas. This is based on the following property:

Property 1. *[Kat+10] Let $|R| = n$. Then, for any two instances of schema R.:*

$$d(r_1, r_2) \leq 2^n - 2$$

Example 2. *Let's take the two closure systems from relations defined on a schema schema $R = ABC$ defined as follows:*

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

Then, for two relations r_1 and r_2 with respective closure systems CL_1 and CL_2:

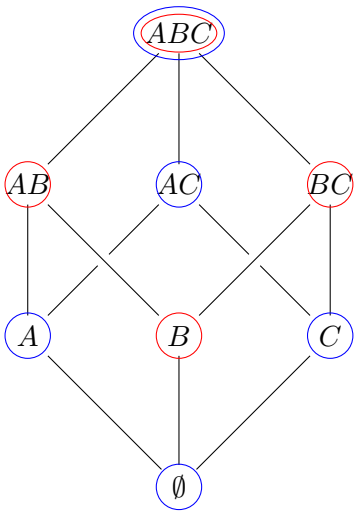


Fig. 2.2: Illustration of two closure systems with a maximum symmetric difference, defined on the same schema

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

This symmetric difference is illustrated on figure 2.2, that shows how the two closures systems as complementary different, allowing for a maximum symmetric difference, and therefore a maximum distant between relations satisfying such closure systems. As a result, r_1 *is a far as possible from* r_2 *and vice-versa. Moreover, they respect the following sets of functional dependencies:*

- $r_1 \models \{A \to B, C \to B\}$
- $r_2 \models \{B \to AC\}$

This illustrates this intuition of "opposites" functional dependencies in distant relations. For example, the two following relations would be consistent with these constraints:

r_1	*A*	*B*	*C*
	0	*0*	*0*
	0	*0*	*1*
	2	*0*	*0*
	3	*3*	*3*

r_2	*A*	*B*	*C*
	0	*0*	*0*
	0	*1*	*0*
	0	*2*	*2*
	3	*3*	*0*

This notion of distance based on closure systems 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.

2.2.4 Evaluating FDs satisfaction

Essentially, given a relation, an FD is either satisfied or not. Rather than comparing all the pair of tuples in the dataset, it is possible to verify the satisfaction of an FD using the following simple yet powerful property [Huh+99]:

Property 2. *Let r be a relation over $\mathcal{R}$. Then:*

$$r \models X \rightarrow C \iff |\pi_{X \cup \{C\}}(r)| = |\pi_X(r)|$$

But when the FD is not satisfied, it is possible to evaluate how far the relation is from satisfying the considered FD. Indeed, the unsatisfaction of the FD comes from pairs of tuple that violate the constraint modeled by the FD. Such pairs of tuples are called counterexamples, and are defined as follows:

Definition 4. *Let r be a relation over R and $X \rightarrow Y$ a FD f on R. The set of counterexamples of f over r is denoted by $CE(X \rightarrow Y)$ and defined as follows:*

$$CE(X \rightarrow Y, r) = \{(t_1, t_2) | t_1, t_2 \in r, \textit{ for all } A \in X, \\ t_1[A] = t_2[A] \textit{ and there exists } B \in Y, t_1[B] \neq t_2[B]\}$$

The counterexamples are therefore tightly linked to agree sets.

The problem is then to evaluate what is the importance of counterexamples in the relation: if there are just a few of them, then the FD is not far from being satisfied, and on the opposite, too many counterexamples makes the FD far from being satisfied. As a result, it is necessary to quantify their number, a problem that is more subtle than it may seem. Indeed, as counterexamples involve pairs of tuples, it means that one tuple may be involved in several counterexamples, so the distribution of counterexamples among the tuples should be taken into account.

This problem has been addressed in [KM95], that proposes three measures to evaluate the satisfaction of a functional dependency $X \rightarrow Y$ in a relation r. Other measures, based on information theory, are presented in [DR00], but are out of the scope of this paper.

The first measure, G_1, gives the proportion of counterexamples in the relation:

$$G_1(X \rightarrow Y, r) = \frac{|\{(u, v) | u, v \in r, u[X] = v[X], u[Y] \neq v[Y]\}|}{|r|^2}$$

Using definition 4, this can be rewritten as:

$$G_1(X \rightarrow Y, r) = \frac{|CE(X \rightarrow Y, r)|}{|r|^2}$$

Following this first measure, it is also possible to determine the proportion of tuples involved in counterexamples. This measure G_2 is given as follows:

$$G_2(X \rightarrow Y, r) = \frac{|\{u | u \in r, \exists v \in r : u[X] = v[X], u[Y] \neq v[Y]\}|}{|r|}$$

These two metrics are designed to evaluate the importance of counterexamples in the relation. Similarly, measure G_3 computes the size of the set of tuples in r to obtain a maximal new relation s satisfying $X \rightarrow Y$. Contrary to [KM95] that present this measure as an error, we propose it as follows:

$$G_3(X \rightarrow Y, r) = \frac{max(\{|s| | s \subseteq r, s \models X \rightarrow Y\})}{|r|}$$

2.2.5 Non-crisp FDs

All the previous concepts introduced only consider crisp functional dependencies, which are based on the comparison of values using the strict equality. However, there exists many applications in which this assumption is not representative of the field reality, because two values might be considered as equal even though they are not strictly equals. In this context, we introduce the necessary concept to take into account *similarity* when dealing with an FD's satisfaction, and assess how it impacts the notions previously introduced to evaluate it. In this context, it is therefore necessary to introduce a form of *similarity* measure to compare the values: depending on the desired result, it is therefore possible to relax the equality on the left-hand side, or on both sides of the dependency.

A similarity operator $\approx_A$ on an given attribute A is defined on $dom(A) \times dom(A) \rightarrow \{true, false\}$, such that the output is $true$ if the two compared values are similar, and false otherwise. Given $a, b \in dom(A)$, we denote by $a \approx_A b$ if a and b are similar, and $a \not\approx_A b$ otherwise.

Reflexivity $a \approx_A a$

Symmetry if $a \approx_A b$ then $b \approx_A a$

It should be noted that a similarity measure is not necessarily transitive, meaning that if for $a, b, c \in dom(A)$, if $a \approx_A b$ and $b \approx_A c$, there is no guaranty for the similarity of a and c.

Similarity can be extended to a set X of attributes: $a \approx_X b \iff \forall A \in X, a \approx_A b$. Based on this, the satisfaction of non-crisp FD ($\models_s$) is a straightforward extension of crisp FD.

Definition 5. $r \models_s X \rightarrow Y$ *iff for all* $t_1, t_2 \in r$, *if for all* $A \in X, t_1[A] \approx_A t_2[A]$, *then for all* $B \in Y, t_1[B] \approx_B t_2[B]$

It is therefore possible to redefine some of the previous concepts, this time for non-crisp functional dependencies:

- $CE^s(X \rightarrow Y, r) = \{(t_1, t_2) | t_1, t_2 \in r,$ for all $A \in X, t_1[A] \approx_A t_2[A]$ and there exists $B \in Y, t_1[B] \approx_B t_2[B]\}$
- $G_1^s(X \rightarrow Y, r) = \frac{|CE^s(X \rightarrow Y, r)|}{|r|^2}$
- $G_2^s(X \rightarrow Y, r) = \frac{|\{u | u \in r, \exists v \in r : u[X] \approx_X v[X], u[Y] \not\approx_Y v[Y]\}|}{|r|}$
- $G_3^s(X \rightarrow Y, r) = 1 - \frac{Max\{|s| : s \subseteq r, s \models_s X \rightarrow Y\}}{|r|}$

All this relates directly to alternative FDs that have been defined. For example, metric functional dependencies [Kou+09] allow similarity on the right-hand part (Y), while using the equality on X. In a complimentary manner, matching dependencies [Fan+09] relax the left-hand part with similarity measure, while requiring an exact match for the Y values.

2.3 Predictive models

2.3.1 General definition

Predictive models are used to predict an outcome, based on statistics: their purpose is to use the available data to predict future values. In machine learning such models are used especially for supervised prediction: the model is trained on data for which the outcome is known. It can be a classification model (the number of values to predict is limited and known) or regression (the value to predict is continuous). In this thesis, we mostly consider classification problem, and latter on discuss how to extend our approach to regression ones.

Traditionally, a supervised classification problem (see [Moh+18]) consists in a set of N training examples, of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$ where x_i is the feature vector of the i-th example and y_i its label. The number of label (also known as class), k, is limited and usually much smaller than the number of examples ($k = 2$ in binary classification problems). Given the training examples, classification is the task of learning a target function g that maps each example x_i to one of the k classes.

The function g, known as a the model (or classifier), is an element of some space of possible functions G, usually called the *hypothesis space*. The objective of a learning algorithm is to output the classifier with the lowest possible error rate, which is the portion of misclassified examples according to their ground truth label.

It is sometimes convenient to represent g using a scoring function $f : X \times Y \to \mathbb{R}$ such that g is defined as returning the y value that gives the highest score:

$$g(x) = \arg\max_y f(x, y)$$

This optimal function can take different forms, depending on the learning algorithm used to define it: polynomial, exponential, sometimes not even expressible using simple formulas (black boxes). This function g is often referred to as the *model* of the classification task.

For sake of simplification, we express a classification problem using relational databases notations. In the sequel, we will therefore consider a relation $r_0(A_1, \ldots, A_n, C)$ with N tuples, where for any tuple t_i, $t_i[A_1 \ldots A_n] = x_i$ and $t_i[C] = y_i$. In addition, we consider that traditional feature selection methods (see [AA+11]) have been applied and consider the subset $X \subseteq \{A_1 \ldots A_n\}$ of attributes selected for the classification process.

2.3.2 Evaluating the model's performances

In order for a predictive model to be useful, it is necessary that it performs satisfyingly with good performances: this means that he should make as many correct prediction as possible, and as few errors as possible. There as therefore been many measures developed to evaluate a model's performances. In this thesis, we mainly use *accuracy*, a simple yet useful measure, measuring, for a model M and a relation r to evaluate it on, the proportion of correct prediction. More precisely:

$$accuracy(M, r) = \frac{number\ of\ correct\ predictions}{|r|}$$

It should be noted that in order to train a predictive model, the available data is usually divided into at least two sets:

Training set It consists in the data used to fit the initial model

Testing set It is the data used to first assess the model's performances, and consists in data that is not in the training set, to provide an unbiased evaluation.

2.3.3 Estimating the performances from the data

In machine learning, estimating the performances of a classifier is a problem that has been studied, because it gives an objective to reach for when training a model. The main proposition to estimate this is *Bayes error* [Fuk13], which is defined as the lowest possible error for any classifier, and computed as follows:

Definition 6. *The Bayes error E_{Bayes} for a multiclass classifier from X to C is calculated as follows:*

$$E_{Bayes} = 1 - \sum_{C_i \neq C_{max,x}} \int_{x \in H_i} P(C_i|x)p(x)\,dx$$

where x is an instance, C_i is a class into which an instance is classified, H_i is the area/region that a classifier function h classifies as C_i.

Given a data distribution, Bayes error is the probability for a tuple to be misclassified by a model, if this model knows the true class probabilities. Bayes error is therefore a very nice theoretical tool, but turns out to be very difficult to estimate in practice, because it requires to estimate integrals over distribution functions that are most likely unknown. As a result, solutions to approximate Bayes error and to compute it in very specific situations have been proposed such as in [TG96]. But because of the definition of this error itself, it is difficult to compute it without making asssumptions and hypothesis on the data, or whithout having additional information on the data distribution.

Similarly, the Vapnik–Chervonenkis (VC) dimension [Vap+94] is also a decription of a classifier's abilities, but is also hard to compute in practice. It measures the "capacity" of a space of functions that can be learned by a statistical classification algorithm: intuitively, it corresponds to the number of points for which a function can be found to predict the class, such that the function does not make any error.

3 Data selection and imbalanced classification

Chapter's outline

In this chapter, we explain how data selection can be seen as an imbalanced classification problem, where the tuples to be selected are much fewer than the volume of available ones in the database. We therefore propose an undersampling solution, based on the distance between two relations relying on the dependencies of these relations and there closure systems. We test this approach by generating synthetic datasets, and propose an evaluation using several classification algorithms.

We introduce the problem addressed in this chapter in section 3.1, and related work in section 3.2. Our approahch based on the generation of synthetic data is presented in section 3.3. Based on this, experimentations are detailed in section 3.4. We discuss the persectives of this work and conclude in section 3.6.

3.1 Introduction

3.1.1 Guided data selection

The selection of the desired data is the task of finding the relevant subset of tuples among all the ones available in the considered database. At first sight, this problem can seem trivial, but in practice it can turn out to be extremely complex. Indeed, the reality of how relational databases are currently used is often distant from the theoretical concepts of databases foundations. The database's schema complexity can make it more difficult to locate the desired attribute, if there are too many columns and/or table, if the name of columns is not explicit, etc. This is especially true nowadays, as because of the vast amount of available data, domain experts do not necessarily work on raw data, but rather on concatenation of several data sources, and therefore on results that might already have been obtained from an

unknown SQL query. They therefore have to do their best with what is given to them, and in this context, writing a SQL query can appear anecdotal: the main problem is to sort between the data of interest and unrelevant tuples, among all the ones that the experts has been given access to.

The user's qualification also has an important impact on the data selection process: depending on her knowledge of the database and of SQL, the time required to select the required data can greatly vary. As a result, the solution proposed to address these various problems should rely on minimal knowledge from the user.

In this context, one possible approach is to ask the user what are the tuples of interest, and to determine, based on her assessment, what are the tuples of interest, and what are the ones that can be discarded. This can be done with a minimal user implication in the form of a binary input, with a simple *yes/no* answer to indicate if a given tuple is interesting or not: such techniques are called *example-based* [Lis+18]. From this labeling, it is possible to constitute a basic labeled dataset, that can then be used to apply well-known classification methods to determine if a tuple should be returned to the user or not. As a result, once the training set labeled by the user is available, all the other tuples from the database can be automatically processed.

3.1.2 Data selection as imbalanced classification

In this chapter, we focus on the sorting of the tuples, to help a database's user to identify the one that are of interest to her. Essentially, we aim at distinguishing what the user wants from the rest of the database, to help her select the appropriate data. Whether it is by asking the user to manually label part of the tuples, or by inferring what are the tuples of interest, the final result is a form of binary dataset with all the tuples of the database labelled as *interesting* or not. This is a form of virtual dataset, where only one class should be explicitly specified, which is the one containing the tuples of interest. The other class can then be constructed by inference, considering all the other tuples in the database as part of it.

As presented before, such a dataset is of high interest to assist the user in writing her queries. Whatever the method or the purpose, all these different tasks can be summed up as learning, from this binary dataset, to classify the tuples in two classes. As a result, in this chapter, we propose to consider the data selection problem as a classification one.

Clearly, this classification problem is an imbalanced one: the distribution of the two classes is skewed, as the set of tuples required by the user is usually much

smaller than the amount of tuples available in the database. This is also why the data selection problem is so difficult: identifying the tuples is sometimes like looking for a needle in a haystack. Additionally, like for most imbalanced problem, the most important class is the minority one, that is also the most difficult one to predict due to the lack of training example with respect to the size of the minority one. In the case of the data selection problem, it is especially important to have a good recall for the minority class, to make sure that all the relevant tuples are retrieved, and to not miss any relevant tuple: this is related to the problem of *why not queries* [Bid+14], that aim at explaining why a tuple is not selected by a given query. Such tuples can be considered as false negatives: they should be selected, but they weren't. A good precision is also important, to not have too many false positives: this would overload the user with useless tuples, and give her too many tuples that she does not need.

3.1.3 Problem statement

Imbalanceness is a well-known problem in machine learning, but looking at it from the prism of data selection opens new possibilities. As a result, we propose to consider a specific aspect of relation data, that is the data dependencies that exists between the columns of a given database. Functional dependencies have proven to be a key concept for databases design [Abi+95], or for data quality and data cleaning [Boh+07]. There exists many algorithms do discover the dependencies that hold in given relation, such as TANE [Huh+99]: it is also reasonable to assume that these dependencies can be directly given by a domain expert, as she usually known what are the domain's constraints that the data should satisfies.

Functional dependencies give powerful global constraints on the database structure, by describing relationships between the columns. It is therefore very interesting to notice that dependencies are a key concept in databases, but is much less considered in machine learning, especially when building predictive models.

Additionally, one can make the assumption that the set of tuples that are interesting for the user is not a random collection of tuples, but rather a set of tuples that are likely to share some common characteristics. It is therefore likely that the set of *interesting* tuples satisfies some functional dependencies, as the different tuples are more likely to follow similar trends. Based on this assumption, the intuition is that in opposition, the majority class, that is the set of *non-interesting* tuples does not satisfy the same dependencies, or even satisfies opposite ones. Ideally, it is then possible to estimate the difference between the two classes, in terms of functional

dependencies, using some sort of *distance* to be able to estimate how different they are.

Considering the overall majority class, it might not be very distant from the minority one, especially due to the diverse data it contains: but as we are in an imbalanced setting, it is possible to only consider a subset of the majority class. The objective would therefore be to identify the subset of data, in the majority class, that is, in terms of dependencies, as far as possible from the minority class. This would therefore identify a subpart of the majority class that is very different from the minority one, and that would therefore be the most "opposite" class for a classification problem. As a consequence, in this chapter, we propose to study a new form of undersampling, and to see if it can be used to better classify the tuples of interest for a user.

To summarize, the general idea is to consider a set of functional dependencies satisfied by the minority class, and to define the set of functional dependencies that are as distant 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 second set of dependencies. Traditional classification techniques can then be applied on this new dataset, to solve the various tasks at hand related to data selection.

Our approach is therefore centered around one main conjecture: it should be easier to classify between two sets that are *distant*[1]. Therefore, before thinking about the general solution, which is not trivial and raises several combinatorial challenges, we propose to study this conjecture, and to see how it applies in the specific context of imbalanced datasets. To this end, we ask the two following questions:

(1) Is it easier to classify imbalanced datasets with *distant* classes?

(2) Can functional dependencies help to identify better balanced classification datasets, by undersampling the majority class?

The objective of this chapter is 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. This makes it possible to fix the distance between the considered classes, and to control the various parameters of the experimentations. We can then express 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"?

[1]For the rest of this chapter, when clear from context, we will use the words *distant* and *distance* to refer to the distance in terms of functional dependencies

After considering this first question, we can move to the one closer to data selection and imbalanced classification:

> Given a synthetic imbalanced dataset, is classification easier when the majority data for the training set is undersampled in order to have two *distant* classes?

To answer this, the contributions of this chapter are as follows :

- The use of a semantic distance based on functional dependencies and closure systems, as described in [KS12].
- The construction of synthetic datasets such that the distance between the minority and the majority class is maximum.
- The construction of imbalanced classification datasets, such that there exists a subset of the majority class that is as *distant* as possible from the minority one.
- Experimentations, applying various classification models, to compare classifiers performances when discriminating between the different datasets generated.

The purpose is first to point out if synthetic datasets generated as *distant* are easier to classify than random datasets. Then, further experimentations verify how well classifier trained on such datasets adapt when tested on the imbalanced dataset.

3.2 Related Work

3.2.1 Data Selection with example tuples

Being able to classify tuples as relevant or irrelevant is very useful for data selection problems, for which it is not clear how to select the desired tuples, and therefore for which the query to pose is not clear. For this reason, many solutions have been proposed to automatically retrieve tuples and/or write queries that return the desired tuples, that are based on a subset of tuples labelled by the user according to different techniques. This is why, most of the time, these methods have two objectives: identify the tuples of interest, but also design the query returning such tuples. Indeed, the query is usually necessary to characterize the set of tuples, so that it can be reused later, or on different instances of the database.

This is exactly the purpose of *example-based* methods: they use a few input from the user, that are given as examples of what she finds relevant, in order to derive all the tuples that are of interest, and therefore the query she is looking for. An overview of such methods can be found in [Lis+18]. Historically, *query-by-example* has been introduced in [Zlo75] as a language, in order to specify a query using a sort of skeleton, so that SQL is not necessary to write the query. However, it just allows to specify the different columns of the query in a simpler manner, but is not example-based in the sense that it does not retrieve tuples that are related to the given example: in this case, the example are used to give the general pattern for all the tuples that should be retrieved, as a representation of the desired query output.

Globally, the purpose of example-based methods is to find the query returning a given set of tuples (labeled as interesting), a problem called the *query-by-output* problem, or *query reverse engineering*. Sometimes, there might be several possible queries for the same set of examples. In [Tra+09], this problem is treated as a binary classification one: the approach is strict, as all tuples must be returned by the identified queries. In comparison, [She+14] allows for query returning approximately the example tuples, so that the queries can be found more efficiently.

But in addition to the tuples labeled as *interesting*, it is also important to retrieve tuples similar to these, that the user might not have labeled. The key to *example based* methods is therefore similarity: their purpose is, from a few tuples given or labeled by the user, to retrieve all the tuples that are similar enough to these examples. This requires to be able to define the similarity between the tuples, and to define a threshold. This is also the purpose of classification algorithms in this setting: define conditions to group the tuples that are similar, based on different algorithmic strategies. To help the user select among all the possible queries, [Li+15] offers to determine what distinguishes one query from another, returning the labeled examples, so that user understands what is the most appropriate query for her need. Similar approaches include [WC17] for which users have to specify both positive and negative examples, an approach also used in [Bon+14]. All these approaches usually use a form of binary classification. The main problem is usually the complexity of the queries, a problem addressed in [Zha+13] that allows to consider queries more complex than the usual SPJ (select-project-join) used by other approaches. In [Mar+18], the authors compare three different methods to derive a SQL query from examples (greedy algorithm, decision tree and genetic programming), showing that decision trees usually produce the most acurate queries from example, even though there are sometimes too long to summarize the user's needs efficiently. [Dim+14]

introduces AIDE, a framework that ask relevance feedback from the user, in order to predict the query that retrieves the tuples she is looking for.

If we take the example of query reformulation for data exploration [Cum+17], there is a direct imbalanceness problem: in order to reformulate a given query, the authors classify between the tuples from this query and other tuples from the database. For this second set, the number of candidate tuples is much bigger that the result set of the considered query. The authors bypass this imbalanced problem by proposing a heuristic that essentially select only a subset of the available tuples from the pool of *non-interesting* ones. This is therefore a form of guided undersampling of the majority class, that uses the SQL query given by the user to create the set of *interesting tuples*. In this work, the authors therefore exploit the relational language used to identify the minority class in order to constitute a balanced classification dataset. However, this approach only cares about the size of the two classes, but does not consider the data itself or its structure: the selected tuples might therefore not be the most interesting ones to perform the best possible classification possible.

To summarize, our approach is similar to example-based methods, in the sense that we focus on the data to perform the data selection, rather than starting with the query writing. Our main caracteristic is to consider this problem as imbalanced classification, and to therefore propose an alternative method taking advantage of the database's setting.

3.2.2 Imbalanced datasets classification

Imbalanced dataset classification is a well-known and recurrent problem in machine learning. It occurs every time one of the classes of a classification dataset is much smaller than the others: in practice, it occurs in many real-life application, such as medical diagnosis or fraud detection. In the setting with only two classes (binary classification), the class with few examples is denoted as the minority one, and by contrast, the other is the majority one. 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.

Because of its importance and recurrence, the imbalanced dataset problem has received a lot of attention from the machine learning community. They have

proposed various approaches to deal with it [Kot+06], in order to improve the performances of classification algorithms over such datasets. These approaches can be roughly divided into three families: data-centered , algorithm-centered, and hybrid [Kau+19].

Data-centered solutions aim at reshaping the dataset to make it more balanced, by modifying the data distribution among the classes. Such approaches are usually based on resampling of the data. It is for example possible to undersample the majority class, by removing some of its tuples, to give it a size similar to the one of the minority class. This undersampling can be random [KP03], or guided by the constitution of the minority class [KM+97]. It is also possible to oversample the minority class, by generating synthetic samples in the minority class, to get it closer to the size of the majority one. The most common appraoch to oversampling is the synthetic minority oversampling technique (SMOTE) [Cha+02], that interpolates new instances based on the nearest neighbors in the minority class. Due to the success of this method, many variants have been proposed [Fer+18]. Some also propose to use the information from the majority class to synthetize the minority one [Sha+18]. Of course, it is even possible to combine oversampling and undersampling on the same dataset. In this chapter, our solution is based on the undersampling on the majority class.

At the algorithmic level, the purpose it to adapt existing classification algorithms, so that they take into account the specificity of imbalanced datasets. One-class learning on the minority class has for example shown good performances on imbalanced data [RK04]. Cost-sensitive approaches can also be of interest [Dom99], as they give more importance to the misclassification of a sample based on its class. Additionally, boosting algorithms [FS95] rely on the output of several classifiers to produce a final decision.

Finally, some solutions have been proposed to combine the resampling of the dataset to an adapted algorithm. This is the case of the mixture-of-experts approach [EJ01] that combine boosting with resampling of the data given to each of the different classifier.

3.2.3 Distances between relations

The approach developed in this chapter is based on the distance between two relations. 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 [Han+11]). 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 [Mül+06], 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 duplicated and modified at different places. In this chapter, we use the distance between relations as defined by Katona and Sali [Kat+10], that relies on the closure systems of the relations.

3.3 Imbalanced datasets generation

3.3.1 Proposed solution

The semantic distance based on closure systems from [KS12] gives an indication of how two relations are distant regarding their functional dependencies. It is therefore valuable to propose a new data selection approach, if it can allow to better classify imbalanced datasets with constraints from relation databases.

In order to evaluate if functional dependencies can be used for undersampling the majority class of non-interesting tuples for the data selection problem, we propose to experimentally evaluate two conjectures:

1. In a balanced setting, is it easier to classify when the two classes are *distant*? The purpose of this question is to see if, even without the imbalanceness, the *distance* between the classes could be helpful for the classification.

2. In an imbalanced setting, is an FD-based undersampling strategy helpful to improve the classifiers performances? This second question evaluates if it is worth developing an undersampling approach for the imbalanced data selection problem.

To this end, we therefore need a imbalanced dataset. We therefore consider one table representing this dataset, such that it contains a minority class r^-, and a majority class M: the tuples are labeled with the class they belong to. Such a dataset is easy to generate. However, because of the specific setting of this chapter, it is necessary to have guaranties on the majority class: there should exists a sample of it, r^+, such that it is as *distant* as possible from r^-. We also need to have a random sample from

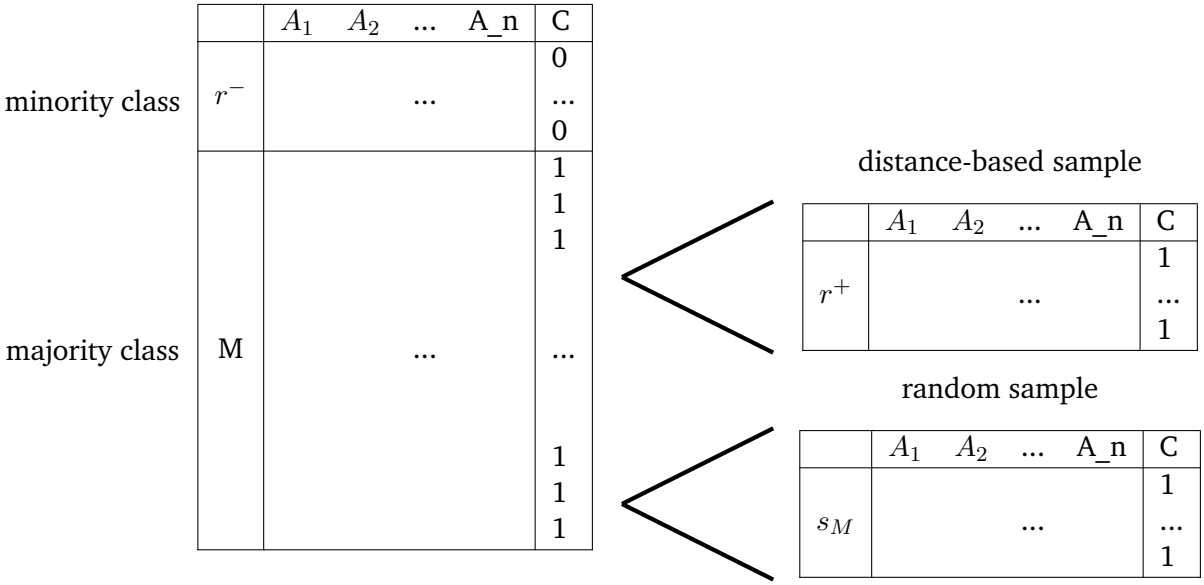


Fig. 3.1: Structure of the different relations for the imbalanced dataset

M that we denote by s_M. The structure of those different relations is outlined on figure 3.1. This way, it is possible to answer our first question by comparing the classification between r^+ and r^- to the one between s_M and r^-. We can then move to the second question by training the classifiers in the same two settings, but using tuples from M to test their performances, in order to see if the undersampling for the training set improves the performances of algorithms when generalizing to an imbalanced dataset.

We follow the following five steps process:

Step 1: For a given schema $R = \{A_1, ..., A_n\}$ of size n, create two closure systems CF^+ and CF^-, such that their symmetric difference is maximum and their size comparable (of the same order).

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

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

Step 4: Create a synthetic majority relation M over R, from which r^+ is supposed to be a "semantic" sample based on the FD-distance.

Step 5: Sample M to get s_M, such that $|s_M| = |r^-|$, and train classification models on s_M and r^-.

Step 6: Test the models on the testing sets corresponding to the two initial conjectures.

This approach raises several questions and sub-problems, that are addressed in the following sections. We first discuss the generation of the closure systems so that they are distant. We then discuss the generation of relations from such closure systems, and specifically address the problem of the data values taken to build r^-, r^+ and therefore M. 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. Finally, we discuss the classification process that shall be used for the experimentations.

3.3.2 Synthetic closure systems generation

Let's first consider the closure systems generation: they should satisfy some properties in order to be consistent with the setting of this chapter:

Property 3. *Let CF^- and CF^+ be the two closure systems of two relations over R, as distant as possible and of similar size, and closed by intersection. Then:*

- $|CF^- \triangle CF^+| = 2^{|R|} - 2$
- $|CF^-| \approx |CF^+|$
- *CF^- and CF^+ are closed by intersection*

Generating such closure systems is not a trivial problem: given a schema $\mathcal{R}$ of size n, many different closures systems can be obtained satisfying this condition, but they are not easy to find, especially as the schema's size grows.

This problem is interesting and difficult, and the main objective is to be able to automate the generation of such systems, so that many different configurations can be tested, on instances with more than two or three attributes. Indeed, the design of such closure systems quickly becomes impossible to do manually. To be able to generate automatically two closure systems with a maximized symmetric difference, we therefore 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 CF^- and CF^+ are both initialized with $\mathcal{R}$.

- At a given level i in $\mathcal{P}(\mathcal{R})$, all elements of size i that do not belong in either CF^- or CF^+ are considered as available candidates for insertion in one of the closure systems. The candidates are selected one by one in a random order, so that each execution of the algorithm can produce a different result. This way, we can obtain various pairs of closure systems to work with.

- Before inserting a candidate 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 ensure that the closure system is closed by intersection, and to therefore add all the missing elemnts corresponding to the required intersections.

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 the properties given in property 3. 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.

Example 3. *Let's take once again the example of a schema $\mathcal{R} = ABC$. The different steps of the algorithm are illustrated on figure 3.2. On this figure, each step of the algorithm is indicated by a number next to each element of the closure system, CF^- is represented in red and CF^+ in blue. The different steps correspond to the following actions:*

1. *After initialization, the two closure systems both contain the same two elements, that are the top and the bottom from $\mathcal{P}(\mathcal{R})$. After this step, we have:*

$$CF^- = \{ABC, \emptyset\}$$
$$CF^+ = \{ABC, \emptyset\}$$

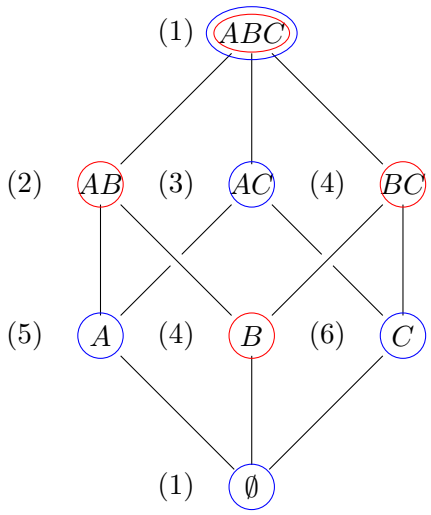


Fig. 3.2: Illustration of algorithm 1 applied to a schema of size 3

2. *We then move on to the second level of $\mathcal{P}(\mathcal{R}$ that contains all elements of size 2. At random, we can start with element AB. At this stage of the algorithm, it can be added to the two closure systems, as it does not cause any intersection problem, and as the two sytems are of same size. Let's add it to CF^-. We now have:*

$$CF^- = \{ABC, \emptyset, AB\}$$
$$CF^+ = \{ABC, \emptyset\}$$

3. *Let's move on to element AC: it does not have any intersection problem with any of the closures, but as $|CF^+| \leq |CF^-|$, it is automatically added to CF^+. As a result:*

$$CF^- = \{ABC, \emptyset, AB\}$$
$$CF^+ = \{ABC, \emptyset, AC\}$$

4. *Then there is element BC to consider. There is no intersection problem. As they are of same size, let's add BC to CF^-: automatically, it is necessary to also add B to CF^- because $BC \cap AB = B$. The closure systems are now as follows:*

$$CF^- = \{ABC, \emptyset, AB, BC, B\}$$
$$CF^+ = \{ABC, \emptyset, AC\}$$

5. *Let's then move to the next level of $\mathcal{P}(\mathcal{R})$: there are only two candidates left, A and C, let's start with A. It does not pose any intersection problem, but because $|CF^+| \leq |CF^-|$, A is added to CF^+.*

6. *For the same reasons, C is also added to CF^+. We therefore obtain the two following closure systems:*

$$CF^- = \{ABC, \emptyset, AB, BC, B\}$$
$$CF^+ = \{ABC, \emptyset, AC, A, C\}$$

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 (CF⁻ ∪ e)∩ ⋂ CF+ ≠ ∅ then
8              CF⁺ = (CF⁺ ∪ e)∩
9          end
10         else if (CF⁺ ∪ e)∩ ⋂ CF− ≠ ∅ then
11             CF⁻ = (CF⁻ ∪ e)∩
12         end
13         if e ∉ CF⁻ and e ∉ CF⁺ then
14             if |CF⁻| < |CF⁺| then
15                 CF⁻ = (CF⁻ ∪ e)∩
16             end
17             else
18                 CF⁺ = (CF⁺ ∪ e)∩
19             end
20         end
21     end
22 end
23 return CF⁺, CF⁻
```

3.3.3 Data generation from closure systems

Once the two closure systems are generated, the objective is to generate two relations corresponding to each of them. It is quite easy to do so using Armstrong relations, as explained in [Arm74]. Indeed, the structure of an Armstrong relation for a set of

functional dependencies (and thus a closure system) is a problem that has already been addressed (see [Bee+84]). It relies on the results of theorem 1 (see page 15.

The general idea is to encode each element from the closure system into a tuple of the relation. Additionally, because some of these elements can be obtained by the intersection of others, it is only necessary to consider the irreducible set of the closure system.

The generation of relations works as follows: given a closure system CF, an Armstrong relation is defined with a reference tuple t_0: $\forall X \in CF$, add a tuple t encoding the element X, $t[X] = t_0[X]$ and $t[A] = i$, with $A \in \mathcal{R} \setminus X$, where i is a value used to encode this tuple. This generation procedure is completely independent from the domain of the attributes.

Example 4. *Let's take the first closure systems from example 3:*

$CF^- = \{ABC, AB, BC, B, \emptyset\}$
Therefore $IRR^- = \{ABC, AB, BC, \emptyset\}$

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

r^-	A	B	C	*Encodes*
	0	0	0	*reference*
	0	0	1	AB
	2	0	0	BC
	3	3	3	$\emptyset$

Similarly $CF^+ = \{ABC, AC, A, C, \emptyset\}$
Therefore $IRR^+ = \{ABC, AC, A, C\}$

Thus r^+ *is derived from* IRR_+*:*

r^+	A	B	C	*Encodes*
	0	0	0	*reference*
	0	1	0	AC
	0	2	2	A
	3	3	0	C

With this basic example, several important questions arise. Indeed, when generating such relations, it is necessary to decide what values the different tuples will take: if we used integers in the example, it could be possible to use continuous values, or even textual values. Additionally, in this example, each encoding of a new element is done by increasing the previous encoding value of 1. Different strategies could be applied to generate relations just as valid as these ones: they could use values at random, increase the encoding values differently, etc.

Having two closure systems also adds some complexity to the generation of the relation: indeed, in the previous example, the two relations are generated using the same encoding values. But it is possible to use completely different ones, to have an overlap between the two, etc.

Finally, the future use of those relations, i.e for classification problems, should also be taken into account. Each of these relations will correspond to a class in a classification dataset. There is therefore a balance to be found, in order to obtain a convincing classification dataset. There are two possible pitfalls to avoid:

- If the values of the two relations are completely different, then the classification problem becomes trivial: it is easy to discriminate between two classes if they don't have any values in common (for example one with only positive values, and the other with only negative ones). As a consequence, the values taken by the two classes should overlap in some way.
- If the values used to generate the two relations are the same, then it is not representative of a classification problem: indeed predictive models are build on the assumptions that there exists specific characteristics in each class, and are mostly based on distinguishing the classes based on the value over the different features. Each relation should therefore have its specificities that characterize its data, that can be identified by classifiers to discriminate between the two classes. To this end, it is necessary for the two classes to be partially distinct in terms of values used.

In these consideration, the choice of the reference values for each relation is especially important:

- If it is the same for both, it 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 very similar active domains.
- 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 whether or not this specific reference value appears in a tuple, and therefore discriminate between the classes solely based on their respective reference value.

Finding balance is a crucial point of our process. Based on all this questioning, the following choices where made:

- The values used for the data generation are integers: numerical values are easier to handle for classifiers. Moreover, this reduces the number of possible values in contrast with continuous values, while still allowing to mark a real difference between the two generated relations. Only values in $\mathbb{N}$ are therefore considered. This choice is important, as the predictive models are based on the values available to classify: we therefore discuss alternatives at the end of this chapter.
- For each relation, the reference value used to encode a given tuple is based on the previous tuple that was inserted in the relation. This way, there is not too much redundancy of a single reference value that would appear in every tuple. More specifically, to encode an element X of the closure system, a tuple t_i is generated based on its predecessor t_{i-1}, $t_i[X] = t_{i-1}[X]$ and $t[A] = random\_value$, with $A \in \mathcal{R} \setminus X$, and $random\_value$ the random value selected to encode t_i.
- The two relations do not share any common reference value: this avoids too much overlapping between the two relations.
- Other values that are not used as references are selected at random, to ensure a diversity in the dataset, and to allow some overlapping between the two classes in order to make the classification process less trivial.

We formalize our approach in algorithm 2, that details how the relations are created in order to respect the given constraints. 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^-.
- In `ArmstrongRelation`, a reference value is selected at random in the pool of possible values. It is used to construct the initial reference tuple, required to encode the first element of the closure system. The value used at this step is

removed from the pool of available values, so it is then no longer a possible new value (line 10).

- Then for each element in the considered 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.

- `ArmstrongRelation` ends when all the elements of the considered closure system have been considered.

- Once r^- is complete, the same procedure applies for r^+, using the remaining values in the pool of available values.

Example 5. *Following example 4, applying algorithm 2, the following relations could be obtained:*

r^-	A	B	C	*encodes*
	1	*1*	*1*	*reference*
	1	*1*	*6*	*AB*
	4	*1*	*6*	*BC*
	5	*5*	*5*	$\emptyset$

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

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
```

3.3.4 Random data generation for the majority class

Once the two relations derived from closure systems are generated, we obtain the two relations r^+ and r^-. There are therefore other relations to generate. According to the approach presented in section 3.3.1, it is now necessary to generate relations the relation M, which is the majority class, and to obtain a random sample s_M from it. We recall that r^+ is supposed to be a sample from M, to recreate an undersampling strategy to the imbalanced dataset problem. The generation of the

majority class M therefore has to be designed carefully, and to be coherent with the generation technique previously applied. One M has been generated, it is trivial to generate s_M.

First, in order to be coherent with the generation of r^+ and r^-, only values in $\mathbb{N}$ are considered. Then, the most complex problem is to generate a relation M such that is is believable to say that r^+ is a sample of M. To this end, it is necessary that M and r^+ share many common values. The definition of the active domain of M is therefore crucial, and various strategies are possible. We propose to compare the two following ones:

- $ADOM(M) = ADOM(r^+)$
- $ADOM(M) = ADOM(r^+ \cup r^-)$

The first one proposes the exact same conditions when comparing between r^- versus r^+ and r^- versus ${}_M s$, as s_M will therefore have an active domain similar to r^+. The second possibility introduces some noise as s_M and r^- could have values in common, which would provide a more challenging dataset. These two options are explored in the experimentations (section 3.4).

Once the active domain of M is set, its generation is not complicated, and the number of possible tuples is bounded. Indeed, given a schema $\mathcal{R}$ of size n, there is a limited number of tuples (yet big), that can be generated:

Property 4. *Let $\mathcal{R}$ be a schema of size n, and M a relation over R such that $|ADOM(M)| = m$. The maximal number of tuples for M denoted by $max(|M|)$ is the number of permutations of size n from an alphabet of m elements, i.e $max(|M|) = m^n$.*

This grows considerably fast, and clearly the size of M can be arbitrarily large. Therefore, to limit the size of M, we consider the scaling factor sf, which is the ratio of size difference between r^- and M:

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

This allows to adapt the "imbalanceness" of the dataset, as it is possible to play with the size difference between the majority and minority classes. Based on this scaling factor, $|r^-| * sf$ tuples are generated, corresponding to the majority class M.

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

3.3.5 Classification problem

At this point, all the generation strategies for the different required relations have been defined. We have the minority class r^-, and the majority one M. r^+ and s_M are samples from M, such that r^+ is as distant as possible from r^- in terms of FDs, while s_M is just a random sample from M without any FD consideration.

The purpose is now to apply classification algorithms on different datasets and compare their performances, and to vary the training and testing sets to address our two initial conjectures. Classification datasets are constituted by adding one additional *class* attribute.

Example 6. *Let's take relations r^- and r^+ from example 5. They constitute the following classification dataset:*

	A	*B*	*C*	*class*
r^-	*1*	*1*	*1*	-
	1	*1*	*6*	-
	4	*1*	*6*	-
	5	*5*	*5*	-
r^+	*3*	*3*	*3*	+
	3	*7*	*3*	+
	3	*2*	*2*	+
	9	*9*	*2*	+

The same process is applied by adding the label "+" to tuples from M (and therefore of s_m).

For the first set of experimentations, the 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_M. Using the available datasets, we build two datasets:

- r^- versus r^+
- r^- versus s_M

For our second experimentation, we study the conjecture that the training on distant sets is beneficial for imbalanced classification. 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 an imbalanced classification problem, with r^+ and s_M being two different undersampling of a

bigger relation M. 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 M, as in a real imbalanced dataset scenario.

Finally, in order to assess the performances of the classification algorithms in these different settings, it is necessary to use some metrics. We decided to use *accuracy* as a first line metric. For our second conjecture, it should be noted that in general, accuracy should be manipulated very carefully when dealing with imbalanced datasets, as high accuracy can be reached while misclassifying the entire minority class. In the setting of this chapter, accuracy as a first-line metric is useful to determine if it is worth going further with more elaborate measure: indeed, if the accuracy is bad, then it means that other metrics will be even worse, and that other elements of the process should be improved before moving to more specific metrics. If accuracy is good, then it is worth going further to see how the different classes are classified, using for example precision, recall, geometric mean [Bar+03], etc.

3.4 Experimentations

The objective of the experimentations conducted in this chapter is to evaluate two conjectures, presented in section 3.3.1. To this end, we first present the implementation of these experimentations, before presenting the results for the different scenarii conducted.

3.4.1 Implementation

All the generation algorithms presented in section 3.3 algorithms have been implemented using Python 3. All classification algorithms are from the scikit-learn machine learning library [Ped+11]. Several classification algorithms were selected for the experimentations (see [Han+11] for details), with a fixed parametrization that is usually the default setting of the library. The algorithms and the parameters used are 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.
- Linear Support vector machine: classic SVM with linear kernel
- Radial Basis Function (RBF) kernel Support vector machine: RBF kernel.

3.4.2 Semantic distance and classification

The first set of experimentations aims at studying the first conjecture on which this paper is based: it is easier to classify between *distant* sets. For experimentations, the size of the schema is fixed to $n = 12$. Then:

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

Moreover, sf is fixed to 100: $|M|$ is therefore around 200 000 tuples, and sampled to get $|s_M| = 2096$. The two necessary classification datasets are then built, and divided for training and testing, with an even distribution among the classes. Training test is composed of 80% of a dataset, the remaining 20% are for testing.

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

Table 3.1 presents the average accuracy score obtained for each algorithm over the ten iterations, comparing the two possible choices for $ADOM(Z)$. The main take-out from these results is that the accuracy is better for models that are based on *distant* classes: for every configuration, the result is either equal or better for models trained and tested on $r^+ \cup r^-$. Moreover, if the observed difference is anecdotal for some algorithms such as Random Forest and Adaboost, it is pretty important for others like neural networks. Finally, the active domain of the majority class does not

	$ADOM(M) = ADOM(r^+)$		$ADOM(M) = ADOM(r^+ \cup r^-)$	
Classifier	r^- **vs** r^+	r^- **vs** s_M	r^- **vs** r^+	r^- **vs** s_M
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

Tab. 3.1: Accuracy of each classifier for each data generation strategy. Both models are evaluated on their own testing sets.

seem to affect our observations, as results are also good with noisy data, and even better in some cases.

Those results are very encouraging, as it appears that classifiers perform better for the distant instances, supporting the conjecture that it is easier to classify between distant sets.

3.4.3 General imbalanced dataset problem

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 M 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 3.2. They are more mitigated than the ones observed in 3.1: the difference between the two models is less pronounced, and seems to be more algorithm-specific. In general, the results between the two undersampling strategies are closer than for the first strategy, and are slightly more often in favor of the random undersampling. However, the training on *distant* classes seems to improve the results for some algorithms such as random forest, naive Bayes and RBF SVM, showing that their might be situations in which it is worth investigating a *distance*-based undersampling for an imbalanced dataset.

	$ADOM(M) = ADOM(r^+)$		$ADOM(M) = ADOM(r^+ \cup r^-)$	
Classifier	r^- **vs** r^+	r^- **vs** s_M	r^- **vs** r^+	r^- **vs** s_M
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
RBF SVM	**0,83**	0,56	**0,77**	0,55
Linear SVM	0,99	0,99	0,99	0,99

Tab. 3.2: 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.

3.4.4 Experimentations take-out lessons

The results of these experimentations are both encouraging and mitigated. First, the result on a general classification approach are satisfying, and validate our initial conjecture: it does seem easier to classify between distant set based on their FDs. This is a very interesting experimental result, but difficult to apply in practice: given a general classification dataset, its classes have a fixed *distance*, and it is not possible to modify it to make them more *distant* and easier to classify.

Additionally, the application of our approach to imbalanced dataset does not seem to bring a significant advantage compared to random sampling in our experiments. It seems to improve the results for some specific algorithms, but in other situations the results are not better. The few positive results might not be significant enough to be worth going further.

3.5 Discussion on FDs and classification

From this study, one of the main lessons appears to be the gap there is between the database constraint and the machine learning models: indeed, while models are mainly based on the value, the satisfaction of constraints is independent of them, and only cares about comparing the equality of two values. The choices made in the proposed strategy therefore have a tremendous impact. We here considered the options of using integer values, but many other would be tested: for example, using strings would remove the notion of "order" introduced by number, and could pose new challenges to build a predictive model. Overall, this shows the real bridge

there is between DB and ML, in which the data is clearly not considered from the same angle. In this chapter, we tried to see if DB concepts could be beneficial to ML, even though the results were mitigated. But the question of the values to consider turns out to be the key challenge, in order to define how to generate a convincing classification dataset.

Finally, 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) [Jek+17]. It is also reasonable to assume that as such constraints are also derived from keys and domain's requirements, they can be given by domain experts.

The application of this approach to real data is therefore 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?

3.6 Conclusion and perspectives

In this chapter, we considered the data selection problem as imbalanced classification between tuples of interest (minority class) and the other. Whereas many solutions exists to deal with imbalanceness in classification, none of them uses constraints such as functional dependencies to guide the undersampling strategy. We therefore started from the following conjecture: it is easier to classify a dataset when its classes are *distant* in terms of dependencies? We proposed an undersampling of the majority class such that the the sample is as *distant* as possible from the minority class. We proposed to assess these conjectures by generating synthetic datasets based on *distant* closure systems.

The proposed solution was evaluated experimentally on synthetic datasets. The first conclusion is that it does appear easier to classify on *distant* datasets. Second, we evaluated the approach on imbalanced datasets, for which the results are more mitigated.

Several leads could be exploited to go further in this study. Other generation strategies could be designed to better mimic real classification datasets, other types of values could be used, etc. The results of this study also open the way for

other research questions, as it appeared that the results can greatly vary from one algorithm to another. One could wonder 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.

3.6.1 Towards chapter 4

The overall purpose in this chapter was to identify the relevant tuples for the user: classification of the tuples is therefore enough, in the sense that it returns a set of tuples that has been judged as interesting by the algorithm. However, a complementary approach is naturaly to give a query returning the data of interest. To this end, the method developed in this chapter should be enriched with one of the existing methods presented in section 3.2.1 so that the classification results can be used to produce a query.

For all these reason, in the upcoming chapter, we address the problem of writing the query that returns the data of interest. Moreover, because the manual labeling can be tiresome for the user, we study how this can be done without asking for manual labeling, but rather by exploring the database until reaching the desired data.

4 Data selection and exploration in SQL for imprecise queries

Chapter's outline

The objective of this chapter is to address the problem of writing SQL queries in a exploratory context, by helping user write their final SQL query while exploring the database's content: this way, they can answer their question while gaining knowledge on the data, helping them to perform a more complete data selection.

We present the related work in section 4.2. We then motivate the query extensions with an industrial collaboration in section 4.3. The formal definition of query extensions is then given in section 4.4, and the algorithm to compute them in section 4.5. We then present its implementation and the related web prototype in section 4.6. Finally, the scaling experimentation are presented in section 4.7, and the user one in section4.8. We conclude and discuss the perspectives of this work in section 4.9.

4.1 Introduction

4.1.1 From data selection to data exploration

The data selection problem is central, and in this chapter, we address the writing of the SQL query that actually selects the data: this query is very useful for the user, to both be able to understand the content of the select dataset, and to be able to reuse the obtained results at other times or on other instances.

In chapter 3 we mentioned data selection approaches based on manual labeling of tuples. If such solutions are valuable, as they can be efficient and allow to involve the user in the process, they also suffer from several drawbacks. First, the manual

labeling can soon be tiring and tedious for the user: it is therefore necessary to ask a limited number of labeling to keep the user involved in the data selection task. Moreover, such a labeling is subject to the own biases of the user, that might make assumptions on the tuples she is looking for, therefore missing other that could yet be of interest to her. In such cases, it is interesting to help the user to go beyond her prejudice, by exploring other elements of the database that might be relevant for her task. This is why it is often necessary to combine data selection with data exploration [Idr+15], in order to make sure all the elements offered by the database have been considered before making the final data selection. It is also important to propose data exploration tools to assist user that don't really know what they are looking for, and that are having troubles to start their data selection process, because they are not aware of what they can obtain from the database, or just have a rough idea of what data is interesting without knowing how to get it.

4.1.2 Exploration of relational databases

Combining data selection with data exploration is getting more and more mandatory, because of several factors impacting relational databases, making it more and more difficult to define and fine what the user is looking for.

First, data selection is complicated by the size of databases that keeps increasing continuously, as the volume of data stored is doubling in size every two years (and should reach 44 zettabytes (10^{21}) in 2020). As a result, relational databases are also getting bigger, with more tuples, more columns, more tables, etc. In these conditions, identifying the required data is getting more complicated than ever. Nowadays, it is not unusual to see databases with several hundred of tables, some of them having several hundreds of attributes. For instance, the database used by the LSST[1] (Large Synoptic Survey Telescope) contains tables with hundreds of attributes (table *Objects* has 229 attributes for example), and a look at it is convincing to see that writing queries on such schemas is not trivial.

In addition, because of the deluge of data, there is sometimes less time spent on the definition if the schema: the table and columns might therefore not be very informative or meaningful of their contents, making it harder to locate the desired information. This is even more difficult if the naming is not consistent in the database, a problem that is gaining more importance as the data often come form several different sources [SI18].

[1] `http://lsst-web.ncsa.illinois.edu/schema/index.php`

Another factor explaining the need for assistance in query writing is the profile of users accessing the databases. Indeed, more and more people are in contact with data and databases: SQL and relational databases are widely used to store and access datasets in most commercial data management systems. For example, data scientists often use SQL to fetch and explore data. Many companies, as well as many scientific applications, such as chemical research, pharmaceutical applications, and astronomy research, rely on the declarative nature of SQL to explore these massive datasets, to find insights or to select subsets to feed into their models. This rush on data happens mostly because the data they are trying to access has a potential *value* they can exploit: typically, they can use such data to train classification algorithms in order to better predict some business factors. This is exactly the scenario considered in this thesis, that requires to be able to define what the users need, and what is the query that can retrieve it. They therefore have to explore the data in order to understand it, and to determine what is useful for their applications.

Data exploration tools are therefore crucial as they help users to go to interesting regions of their dataspace, and to identify relevant information or patterns in the data, as argued in [HV18]. There is a real need for exploration systems that can assist the discovery of relevant data, as well as the writing of the queries returning the tuples of interest.

4.1.3 Imprecise relational queries

The purpose of data selection and exploration is to allow users to better understand and reach the data that is required to answer their considered problems. This step is difficult, because it requires to transform a problem, usually expressed with several sentences in natural language, to a single SQL query, which is a constrained language, that is supposed to identify all the necessary tuples. This is especially true for the selection conditions (in the *where* clause of the SQL query), that are a form of characterization of the anwser set of the query: the tuples should be lower than a value on a attribute, but above another one on another attribute, etc. Such conditions can be hard to come up with, especially because usually, they are first given in natural language, making them vague and imprecise. Indeed, the analyst might first think of the necessary conditions in term of adjectives, such as *low, bigger than average, surprisingly high, etc*, that are at first impossible to translate into exact numerical conditions. This is even more complicated if the analyst is not familiar with the database, as such description can have very different translations depending on the database's content and the distribution of the values for the different attributes.

In this chapter, we are therefore interested in assisting user that have to translate such queries that are defined in natural language, but difficult to translate into precise selection conditions for a SQL query. We call such queries *imprecise queries*: this problem is recurrent when performing data selection, because it is the bridge between what the user has in mind, and the query that allows to allow the relevant data. Such a problem is a clear example of how finding the relevant SQL query is a combination of query inference and data exploration: the purpose is to make the user more knowledgeable of the database's content, so that she is able to write the most appropriate query for her problem.

To address imprecise queries in a traditional database setting, an analyst usually has to start with a general query, and to try and refine it until reaching the desired output. This iterative process can be tiresome and require many iterations, as the analyst can be overwhelmed with the initial results, and not know what direction to take. Indeed, the initial query is likely to be too general and to return many tuples. The analyst therefore has to find a way to understand it, in order to to be able, afterwards, to eventually modify the query. Sometimes, this is even harder because the analyst has no control of the initial query, for example if the data is managed externally, so she only has access to a CSV file containing results she has not been able to initially select.

The analyst then compares the results she obtains against her expectations: if they are not reached, the original query needs to be revised. This modification step can be hard, as the analyst has to understand how the given results differ from the expected one. And once the source of the difference is identified, the problem is to find how to modify the initial query to correct it.

One solution to reduce the query's output size is to add more selection predicates to the query, to filter some tuples out. Data analysts often adopt a trial and error approach: they try different combinations of attributes and thresholds for a selection predicate, and adapt and modify it according to the result they obtain. The question of queries returning too many tuples has already been addressed in other research papers, offering various solutions. The interactive query refinement solution exposed in [MK09] addresses the *too many tuples* problem by transforming the selection predicates of a query with respect to a cardinality objective. The `STOP AFTER` SQL operator proposed in [CK97] is also a solution to this problem. Finally, *top-k* approaches are also available [Fag+03].

In this chapter, our objective is to propose a solution that refines the initial query given by the user so that it returns less tuples, by suggesting additional selection clause for the considered query that assist users with imprecise queries. The objective

is to turn the imprecise query into SQL progressively by identifying interesting selection conditions that, one by one, refine the initial query until reaching the desired result set. The proposed solution should be based on SQL so that users can stay in a familiar setting, ans so that it is easily integrable in any DBMS.

4.1.4 Query extensions

To summarize, it appears that to assist users in writing imprecise query, the main objective is to propose a guided refinement process to take them from a very general query to the one they actually need. This process has too main objectives:

Refine the query In the case of data selection and imprecise queries, this means suggesting additional selection predicates that can catch the imprecise needs of the user, and that can be added to the current query so that it returns more tuples. This is related to the query writing part of the problem.

Better understand the results It is necessary to help the user better understand the database's content, so that she can make an informed choice when choosing an additional selection predicate.

Based on these two objectives, in this chapter, we propose, for each refinement step, to summarize the current query by describing its results set using a group of selection predicates that each describe a part of this result set. This helps the user to better understand the current query. Additionally, she can then choose one of these selection predicates to refine her current query, therefore moving to the next refinement step. As a results, for a given query we introduce its *query extensions*: it consists in a set of different selection predicates. The role of these extensions is to provide options to refine the initial query and, when considered together, to understand how the initial query's results can be divided, and therefore what are the different groups of data it consists in.

Using such query extensions, a user can start by writing a query that contains all the knowledge she has about the data she is looking for: if she does not know anything, she can start with a query returning all attributes from all tables. The extensions can then help her to refine her query until she reaches her data of interest. More specifically, given an initial SQL query Q, we propose to a data analyst a list of SQL queries, that addresses two sides of the problem. First, the queries are extension of Q: the beginning of each query is the same, and equal to Q. Then each query has its own set of additional selection predicates, so that each query returns a smaller subset of tuples. Moreover, the extension's results form a partition of Q's results:

EmpID	LastName	Gender	Salary	Commission
e10	SPEN	F	41160	1300
e20	THOMP	M	41250	7400
e30	KWAN	F	39850	5200
e40	SMITH	F	40525	1400
e50	GEYER	M	40175	1100
e60	STERN	M	39560	6200
e70	PULASKI	F	40120	800
e80	FREY	M	40625	6600
e90	HENDER	F	39450	6700
e100	SPEN	M	41560	900

Tab. 4.1: Result set of query Q

each extension summarize part of Q, helping the user in understanding what lies into the initial bigger result set.

Example 7. *Assume that Alice, a data analyst, has access to the database of a company, which contains several tables, among which the Finance and HR tables (with a join attribute EmpID). She is asked to find the gender of employees with a low income: this is typically an imprecise question. Not knowing how to translate the notion of "low" into SQL, and only knowing the existence of a salary attribute in the database, she starts with the following query:*

```
Select * From Finance, HR
Where Finance.EmpID = HR.EmpID
```

She hoped to be able to refine this first query query by looking at the results it returned. We propose a subset of such results in table 4.1, but the full results (several hundreds of tuples) make it hard for Alice to assess what a good threshold would be to get only low salaries. With the solution proposed in this paper, Alice could get help from the three following extensions of her initial query:

```
Select * From Finance, HR
Where Finance.EmpID = HR.EmpID
  and commission ≥ 6200
```

Extension 1 (4 tuples)

```
Select * From Finance, HR
Where Finance.EmpID = HR.EmpID
  and commission < 6200
  and sex = 'F'
```

Extension 2 (4 tuples)

```
Select * From Finance, HR
Where Finance.EmpID = HR.EmpID
  and commission < 6200
  and sex ≠ 'F'
```

Extension 3 (2 tuples)

These extensions contain several pieces of information that can be valuable for Alice. First, they summarize the tuples that were returned by her first query, as they are divided into three queries, and described by the additional selection predicates of these extensions of Q. It gives her directly a selection condition based on the commission attribute, which is part of an employee's income. it shows Alice an attributes she had not considered at first, but that is relevant for her question. Moreover she now has a numerical threshold to start from. Finally, the gender selection predicate can also draw her attention to a discrimination she had not considered. Therefore, those extensions are a way to highlight information and patterns that could be pertinent for Alice, and to help her find numerical threshold that can be hard to come up with.

4.1.5 Problem statement

Because the extensions should summarize and refine the considered query at the same time, we propose the following problem statement: given a query Q, and a number of extensions n, return n extensions of Q such that:

1. The results of the extensions are a partition of those of Q.
2. Each extension consists in query Q and one or several additional predicate.
3. Each predicate is on one attribute from the *Select* clause of Q.

Based on such a definition, the problem is then to be able to compute such extensions, so that they can be useful for query writing when the question is imprecise. As a result, the general question of this chapter can be summarized as follows:

> How can we complete a set of extensions that are informative and help the user refine her initial query to select the tuples of interest?

It is thus necessary to propose a solution to address this question, and to evaluate how well it is answered. The contributions of this chapter are therefore as follows:

- Present a concrete industrial problem in which query extensions can be useful

- A formal definition of the set of query extensions for a given query
- Given a database, a query and the number of desired extensions, an algorithm to compute such extensions that does not require any laborious user input;
- Visualizations to assist the extension selection;
- An implementation of the algorithm with the design of a web application that can be used by users to test the query extension solution;
- Experimentations on the scaling of our algorithms so that extensions can be obtained in a reasonable amount of time.
- User experimentations to validate whether or not extensions are helpful to answer imprecise queries on a relational database.

4.2 Related work

Over the last decades, and because of the new usage and shapes of relational databases, there has been a growing solution for exploratory solutions in the database community. First, all the works mentioned in section 3.2.1 are also solutions that can be seen as exploratory solutions: by helping the user to write her query and by presenting tuples to label, these solutions are a way for the user to be confronted to part of the database's content while getting a query returning tuples of interest. However, all these methods rely, in some way, on a form on tuple labeling. We will now cover other solutions for assisting query writing, and present other approaches more oriented toward data exploration.

When it comes to query writing, there has recently been a growing interest for methods relying on natural language processing (NLP). The idea is simple but very promising: the user only has to specify her query using natural language, and it is transformed into a SQL query. An overview of such methods can be found in [Aff+19]. Among these solutions, some are keyword-based, that use an index to associate words from the natural language to the database's schema: we can mention for example the soda system (Search over Data Warehouse) [Blu+12], that enable "Google-like experience" to generate a valid SQL query. SQLSUGG [Fan+11] is an alternative that suggest several SQL queries based on users keywords. In [ZP09], the authors offer to extend keyword-search to find minimal group-by queries that fit the user's search, in order to aggregate several tuples. In [Yu+07], the problem of keyword search is extended to ditributed databases.

But keywords are limited in the interpretation they have of the initial sentence, as they do not analyze its structure nor grammar. For this reason, new solutions have been proposed to capture more complex patterns: ATHENA [Sah+16] proposes an ontology-based approach to take the phrasing of the query into account, while NaLIR [LJ14] builds a parser to formulate complex SQL queries. The most recent approaches have taken an interest on recurrent neural networks, that show very good performances on the parsing on natural sentences [DL16; JL16; Uta+18] such approaches are very efficient, and do not require too make many domain-specific assumptions to build the model.

These natural language approaches are very interesting, because the user does not need much SQL knowledge, and does not need to no too much about the database's schema. However, it means that she might have trouble to say if the returned query is really the one she is looking for. Additionally, such systems "only" translate the natural language query to a SQL one: this means that the query will be limited by the knowledge of the user, that will only ask for what she knows of. For this reason, more exploratory solutions are interesting the attract the user's to regions of the database he might have overlooked, and to complete the initial results she was searching for.

More exploratory solutions include visualizations, a very useful tool when it comes to exploring a dataset. Solutions have been specifically designed for relational databases, as advocated in [Wu+14] that underlines the need for integrated visualization systems in relational databases. For example, Polaris [Sto+08] designs a visual query language, that aims at combining traditional SQL with drawing commands, to assist user in the exploration of their database. However, producing visualizations when the result set is huge can cause scaling issues and produce visualizations that are too big to ba valuable: for this reason, ScalaR [Bat+13] offers a dynamic solution to efficiently select a subset of tuples that are relevant for visualizations. Another example of visual solutions is SeeDB, that explores a set of interesting and useful visualization related to the considered query [Par+13]. The dbTouch system [IL13] represents the data visually with different shapes, that are used to interact with the user: instead of writing a SQL query, she can interactively refine her results and adjust the obtained tuples. The purpose of all these solutions is to help the user explore her database, by giving her tools that are more intuitive than SQL: in [Nan+13], the authors propose to let the user interact with different gestures that are filmed and interpreted, rather than having to use a keyboard.

However, there also exists other exploration that do not rely as much on visualization, but rather use SQL and the data itself to help the user better understand the data,

and get interesting suggestion of where to look next. Among these systems, one family of solutions is faceted search and exploration [Kas+10], that allow a user to explore and narrow her results with facets conditions that work as restrictions on attribute values: for example *FleXplorer* [Tzi+08] proposes a framework for faceted search, that can be used for relational databases. An other solution,*YMALDB* [DP13], offers to explore databases by recomanding "you may also like" results, that where not part of the user's initial query. Theses additional items are computed by identifying interesting sets of attributes and values (facets), and really allow the user to explore beyond her initial intended results.

All the aforementioned solutions try to relieve the user from having to use SQL for their exploratory processes, by providing complementary solutions. However, other solutions have been proposed that rely more on SQL, but try to make it easier to write SQL queries, while suggesting some exploration solutions, and without relying on example tuples. DataPlay [Abo+12] is an example of such systems, that rely partly on visualizations, and encourages the user to look at the result of her query, but also at tuples that are not returned by it, to better understand the tuples, and better refine the considered query. Alternatively, [QR14] offers to help users refine their imprecise queries by using a probability-based framework.

4.3 Industrial motivation

In addition to all the elements exposed in the introduction of this chapter, this work on query extensions was also motivated by several observations made during an industrial collaboration with Dodin Campenon Bernard (Vinci). It indeed clearly appears that the use that is made of relational databases on a daily basis by some companies, is quite distant from the intended ones from database designers. This considerations were presented at the BDA2017 conference [LG+17].

4.3.1 Context

This considered industrial collaboration involved a civil engineering company, and more specifically its branch dedicated to tunnel boring machines. They are gigantic machines, with a diameter up to 9 meters, and 100m long: an example of such a machine is shown on figure 4.1. They are extremely costly, as well as delicate to maneuver: for these reasons, they are equipped with thousands of sensors, used to guide the machine, monitor the boring, drive the excavation or rocks, etc.

Fig. 4.1: Picture of a tunnel boring machine

Thanks to all these sensors (more than 1200), a lot of data is recorded on the boring machine. They are used to give indications to the machine's driver, as well as to domain experts that can remotely monitor the process, and raise alerts if something odd is detected. The data is usually kept "row", in the sense that only domain expert with acquired knowledge are able to process it based on their experience: it is therefore harder to formalize and convey such knowledge. This data was therefore mainly used for monitoring, limited to specific patterns identified manually by domain experts.

The purpose of the collaboration with this company was therefore to see if this vast quantity of data would be exploited to propose new usage that could go beyond the monitoring tools that were then in place. We focused on the prediction of a value that is measured regularly on the route of the boring machine. Due to confidentiality reasons, the exact purpose of the prediction task cannot be given: we will therefore called the variable to predict $var1$ in the rest of this section.

4.3.2 Database's description

The first step to build a predictive model for $var1$ was to list all the available data sources that existed and that could be used to retrieve valuable features for the model, in order to build a classification dataset. It soon appeared that there were many different ones: files in a proprietary format, pdf files containing instructions

for the machine's pilot, textual description of the soil, snapshots of internal reports about the progress of the boring and, finally, the dump of a relational database with the records of the machine's sensors. For obvious reasons, all these data sources could not be exploited easily: we therefore decided to focus on the database, as it was supposed to contain all the necessary information for the prediction of $var1$, especially, the data from the 1250 sensors.

The data from the 1250 sensors had been stored in a single table, therefore containing 1250 columns: the table was simply a dump for the files in the proprietary format generated by the boring machine. This structure was puzzling, as that applying a simple SQL query on it would be a more difficult task than anticipated. But the trickiest problem came from the name of these columns: it started from DATA0001, DATA0002, ... all the way until DATA1250: these names were therefore completely useless to understand the content of each column, making it almost impossible to identify the column of interest for the construction of the classification dataset. Additionally, there was no constraint specified for the database, therefore no primary or candidate key, making it difficult to join the sensor data to two other tables, such that the one containing the values for $var1$.

In these conditions, querying the database was almost impossible, because identifying the relevant data in could not be done using the column names. According to the domain experts of the company, 93 attributes add been identified as important for the prediction of $var1$. The sensor table of the database had to be entirely rebuilt from the files generated by the boring machine. Thanks to the experts knowledge, the 93 attributes were directly identified in the files, making it possible to only use the sensors of interest. In the end, it was therefore possible to build a classification dataset. But it is important to underline that the techniques employed to make the database exploitable were extremely time consuming, because they required many back-and-forth exchanges with the company. Moreover, this showed that their was no chance for this database to be useful for the company, because the design choices made it impossible to use the data from the most important table. They were therefore loosing a lot of potential value by not being able to easily query their data using all the powerful tools of DBMS and SQL.

4.3.3 Take-out lessons

If the initial goal of the study was focused on the development of a classification model, one of the most important lessons of this study was the importance of the quality of the relational database from which the classification dataset is extracted.

In this specific industrial context, the database was just a dump of the available datafiles, but its design made it impossible to query the data efficiently, therefore depriving the company from valuable possibilities: during the collaboration, much more time was spent on the data selection than on the classification. It is also a vicious circle: because the database is not practical to query, it is not used to its full potential, and there is therefore no effort made to improve it.

In such a context, the ideal solution would of course be to completely redesign the database, to make it more compliant with the standard practices, and therefore more easily usable. But, for many reasons, this is not simple to do for many companies. First, because there are ways to more or less bypass the problems, it might not be a priority for the company. In addition, it requires time and effort, as well as human resources that are not necessary available: it might require to hire someone, or to pay to externalize the process, which is costly. Moreover, changing the database's design is likely to impact other upstream and downstream applications, because they rely on the current design: as a result, this means changing much more than a schema, and the impact on the company's processes can be huge.

In this setting, it appears that from the company point of view, the ideal solution would keep the same database structure, but the DBMS would provide tools that would make it easier to query the data even with such ill-specified tables. If such tools are integrated in the DBMS, it does not change much from the company's side, but is develop new querying possibilities.

4.4 Query extensions definition

Let's consider an initial query Q, for which query extensions have to be defined. The first characteristics of these extensions is that they aim at refining the initial query Q. Hence, an extension Q_{ext} of Q should diminish the size of the result set. We therefore propose the following definition:

Definition 7. *An extension Q_{ext} of Q is defined by:*

$$Q_{ext} = \sigma_{c_1 \wedge \ldots \wedge c_n}(Q)$$

where c_i is an atomic formula, for every $i \in 1..n$

The following property therefore follows:

Property 5. *If Q_{ext} is an extension of a query Q then:*

$$ans(Q_{ext}, d) \subseteq ans(Q, d).$$

The second objective of these extensions is to summarize the result of the initial query Q: as one extension will only contain part of the result of Q, it is necessary to propose a set of different extensions, to make sure all of $ans(Q, d)$ is contained in one of the extensions. Therefore, we define the notion of a k-extensions set, containing k extensions of a query Q such that:

- The union of the results of extensions in the set of size k is equal to the set of tuples from the initial query: this way, the initial query is fully represented in the extensions.

- Each extension returns tuples that are not returned by any of the other extensions: this way the options offered to the user are very different from one another, giving us a wide variety of options to choose from.

More formally:

Definition 8. *A* k-extensions set *of Q over d, denoted by C_Q, is defined as: $C_q = \{Q_1, Q_2, ..., Q_k\}$ such that:*

- *Q_i is an extension of Q, for all $i \in 1..k$*

- *$ans(Q_i, d) \cap ans(Q_j, d) = \emptyset$, , for all $i, j \in 1..k, i \neq j$*

- $\bigcup_{i=1}^{k} ans(Q_i, d) = ans(Q, d)$

Clearly, a k-extensions set forms a partition of $ans(Q, d)$. An example of a 3-extensions set is given in example 7. Being able to parameter the number of extensions for the considered query give more freedom to the user, who can explore with different size of extensions sets, and eventually use her domain knowledge to fix the most appropriate size. We will therefore see in the following section how such sets can be computed, in order to provide useful extensions to users.

4.5 Solution

4.5.1 General approach

Based on definition 8, it clearly appears that an extension set is a partition of the results of the initial query, where each subset has to be described by a set of selection predicates. Given a query, there exists many different possible partitionings: as a result, it is necessary to decide which one to choose to present to the user, so that it is as useful as possible, taking into account the exploratory context. Given definition 8, there is a first indication of which possible partition to select: as the number of subset k to compute is given in the definition, it therefore removes a vast number of candidate partition. However, it still leaves many possible partitions: the definitions itself is therefore not enough, it is necessary to design a strategy to compute an extension set, that is a coherent as possible with the intended used of such extensions: in this setting, the solution used to compute the extensions has therefore a huge impact on the result presented to the user, and is a key element of the data exploration strategy proposed to assist the user in query writing.

In addition to deciding which partition of the results to use, it is also important to keep in mind that for each subset option through partitioning, it will be necessary to define the selection predicates returning such a subset of tuples. Given an initial query Q and the number k of extensions, we have to:

- First, divide $ans(Q, d)$ in k disjoint subsets: this is not trivial, and requires to define the division strategy. This first part is related to partioning, as well as to clustering in machine learning [Jai08].
- Then, for each subset, find a query returning all of its tuples, that is an extensions of Q with respect to definition 7. It should be as short as possible to reduce the user's effort, and to help him easily understand the description of the tuples contained in the considered extension. It should also be as informative as possible, explaining how the considered extension is different from the others. This problem is very similar to *query reverse engineering* [Tra+14] or *redesciption mining* [PR05].

Example 8. *Figure 4.2 illustrates the extensions set computation process: given the results of an initial query Q (the outer envelop), the tuples are partioned into several subsets. Then, for each subset, an extensions of Q is defined returning the tuples of the subset.*

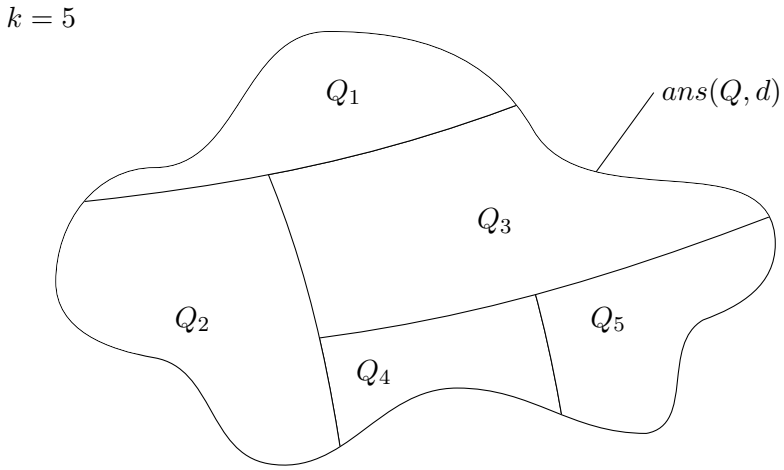


Fig. 4.2: Illustration of the query extensions computation process for a 5-extensions set: partition the initial results, and find an extension for each of them

4.5.2 Partitioning of the initial set of tuples

Using clustering

Let's first consider the partitioning of the initial set of tuples $ans(Q, d)$: the partitions should have a sense for the user, and return tuples that are meaningful with respect to the exploration process. As a result, we propose to identify tuples that make sense together, and that identify regions of $ans(Q, d)$ that share some common characteristics. These regions are more likely to be of interest because they are more likely to correspond to the need of the user, who is usually not looking for sets or random tuples. In addition, these common characteristics are likely to help the formulation of the selection predicates for the extensions, as they can rely on these common characteristics of the tuples.

As a consequence, we propose to divide $ans(Q, d)$ by grouping together *similar* tuples. More specifically, we propose to divide the initial set of tuples using a clustering algorithm (see [Han05] for an overview), as it corresponds to all of our requirements: The clustering algorithm will group *close* or *similar* tuples together; Clusters are pairwise distinct, so the sets will not overlap; and the clusters cover $ans(Q, d)$. Regardless of the initial query Q, the answer set of the query will always be a unique table, that can then directly be sent to the clustering algorithm, along with k, which directly corresponds to the number of clusters to be produced.

Challenges of clustering

Clustering is one solution to address the considered problems, but it yields intrinsic challenges that we ought to address. The main issues is that because we propose a general approach that should be applied to many different databases and queries, the data to cluster is initially unknown: it is therefore necessary to take some precautions. First, it is important to normalize the data to cluster, to avoid the features with higher values to take more importance than the other. We also have to make the assumption that the number of features stays reasonable: indeed, clustering can then be less significant, due to the curse of dimensionality [Fri97]. Finally, most clustering algorithms require to be able to define the distance between two tuples, which can be more or less easy depending on the feature's types. From all of this, it is clear that clustering has to be manipulated carrefully: in our setting, it however provides an efficient solution to group together similar tuples, that are more likely to have interest for the user.

It is also necessary to choose the specific clustering algorithm. Given the input dataset, a clustering algorithm groups together tuples that are *close* to one another, which requires to have some sort of distance between tuples. If this is straightforward for numerical values, with for example the well-known euclidean distance, a database if likely to contain different data-types: therefore our solutions requires a clustering algorithm that is able to handle mixed datatypes.

Addtionally, one key element in choosing the clustering algorithm is that the number of clusters to be produced in known: in our extension setting, we consider this parameter to be given by the user with respect to her exploration requirements: this therefore allows her to explore different extensions possibilities with different values for this parameter.

Considering all these parameters, we propose to compute the k sets of tuples using the *k-means* clustering algorithm [Llo06], that can take the number of clusters to produce as an input: this solution was also used in [SK16] to propose visual data exploration of query results. Moreover, to handle both numerical and categorical attributes, we propose to use its variant *k-modes* [Hua98].

Moreover, as the user might not always know the size of the extensions set she desires, we offer to automatically try different ones: she can therefore specify an lower and upper bound (by default for k = 2 to k = 10), so that different values of k can be tested, and therefore several extensions sets proposed. As among all the computed ones, some extensions sets might be more pertinent than other, they are ranked according the clustering quality, based on the well-known clustering score

of the silhouette coefficient [Rou87]: let's consider a tuple i assigned to cluster C_i. Then we define:

- $a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, i \neq j} d(i,j)$
- $b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$

The silhouette score for i is then defined as:

$$s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$$

Considering all the tuples that have been clustered, the silhouette coefficient is defined as:

$$S_{sil} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|I_k|} \sum_{i \in I_k} s(i)$$

Intuitively, it is the average of how close the points in the same cluster are, and how far they are from he others cluster: the better the clustering, the more separated the clusters.

Preparation for extensions construction

After the clustering, each tuple can be assigned to a single cluster, that can be added as an additional attribute to $ans(Q,d)$. This cluster column acts as a form of tuple labeling, indicating which tuples are likely to form an interesting subset, because they share some common characteristics. Compared to other existing labeling techniques, this clustering-based approach has several advantages. First, it drastically simplifies the role of the user, who does not have to endure the boring task of manual labeling. Additionally, this solution goes beyong the simple binary labeling that is usually used: instead of just considering a tuple as interesting or not, it says whether or not it is interesting to consider it with other tuples or not. This way, instead of evaluating individually the relevance of a tuple, this allows to consider groups of tuples that might be relevant for a given task, together. In practice, a new attribute, called `cluster`, is added to the schema of the query to keep track of the cluster corresponding to each tuple.

Example 9. *Table 4.2 presents tuples from table 4.1, with the additional attribute* cluster *that is the cluster tuples have been assigned to, for a 3-extensions set.*

EmpID	LastName	Sex	Salary	Commission	*Cluster*
e10	SPEN	F	41160	1300	*2*
e20	THOMP	M	41250	7400	*1*
e30	KWAN	F	39850	5200	*2*
e40	SMITH	F	40525	1400	*2*
e50	GEYER	M	40175	1100	*3*
e60	STERN	M	39560	6200	*1*
e70	PULASKI	F	40120	800	*2*
e80	FREY	M	40625	6600	*1*
e90	HENDER	F	39450	6700	*1*
e100	SPEN	M	41560	900	*3*

Tab. 4.2: Tuples from table 4.1 labelled by clustering ($k = 3$)

4.5.3 Construction of extensions for each subset of tuples

Using a binary decision tree

The second part of the process aims at finding, for each cluster, a set of selection predicates that can describe it. Once again, in an exploration setting, there are some specific considerations to take into account: there is a trade-off to find between the accuracy of the clusters description, and the *utility* of the extensions. Indeed, they should be concise enough, so that the user can understand the data they describe quickly, and as informative as possible, in order to underline what separated a specific cluster from the others.

Taking this constraints into consideration, we propose to use a decision tree to construct the query extensions, using the tuples as a classification dataset, and the cluster column as the class to predict. The general idea is to learn what distinguishes a cluster from another: the decision tree will identify relevant attributes and discriminating values to describe concisely a cluster with respect to the other ones. Each leaf of the tree can then be considered as an extension, by using the selection conditions that lead to it from the root of the tree.

Using decision trees to generate SQL queries is a technique that has already been exploited [Cum+17]. To be able to reach the objectives defined for our SQL extensions, we follow the same path but with *binary decision trees* (BDT) [Bre+84], which is a tree splitting at each node on exactly two opposite conditions.

One specificity of our approach is the fixed number of extensions to be computed: we need exactly k extensions. For the clustering, this meant computing k clusters:

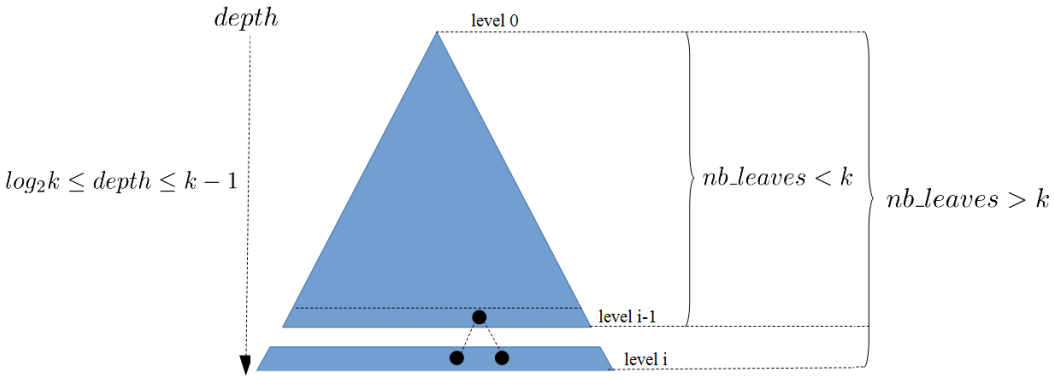


Fig. 4.3: Construction of a binary decision tree given a fixed number of leaves

however, the decision tree might produce more leaves than classes: if we want to produce exactly k extensions, two solutions are considered:

- As they are k classes, it is possible, for each class, to consider all the leaves of the tree that correspond to it: the different conjunction of selection conditions leading to each leaf can then be disjoined, to give only one selection predicate for the considered class. In this case, the produced extensions might be longer and less easy to understand directly because of disjunctions and conjunctions, but maybe also more informative.
- The other possibility is to limit the growth of the tree, to only allow it to have k leaves: by considering each leaf as an extension, k extensions are therefore obtained. In this case the extensions might be shorter and easier to understand, but also less discriminating.

We therefore propose to let the two possibilities, and to let the user decide which strategy he wants to use. For the second one, it requires to be able to produce a decision tree with a constrained number of leaves, a problem we address in the following section.

Obtaining a constrained BDT with k leaves from a given data partition

Constrained generation of BDT given a specific number of leaves has been studied in [Wu+16]. In our case, we just need to explore levelwise the search space (breadth-first search) and stop as soon as the number of leaves exceeds k. To do so, we rely on the following property of BDT: if N is the number of classes to classify, the depth of the BDT is bounded between $\lceil \log_2(N) \rceil$ and $N-1$. Both bounds are attainable:

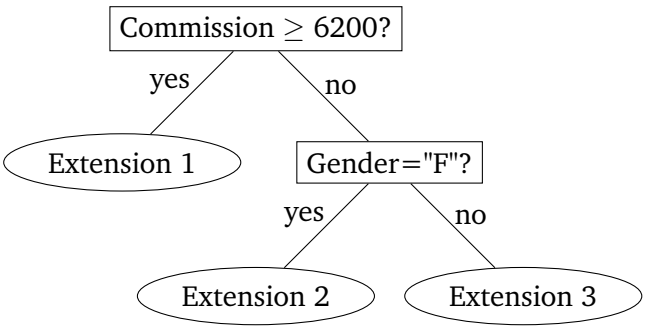


Fig. 4.4: Binary decision tree from Table 1

the first one with a *full binary tree* and the second one with a *right deep tree*. For the query extensions, it requires to stop somewhere in-between.

To reach exactly the k leaves constraint, the construction of the tree starts as usual, level by level. At each new level, there are then three possible scenarios:

- The total number of leaves in the tree is inferior to k: then the construction of the tree can continue.
- The total number of leaves in the tree is equal to k: then it is perfect, the process should stop there.
- The total number of leaves in the tree is greater than k: it means the last level has added to many leaves, and some should be removed.

To figure out how to deal with the last scenario, and how to remove leaves from the tree, let's consider figure 4.3: assume the number of leaves in the BDT is less than k at level $i-1$, and greater than k at level i. The process to remove the unnecessary leaves works as follows: while the number of leaves remains greater than k, replace two leaves at level i from the same parent by turning this parent into a leaf at level $i-1$ (using a majority vote to assign a class to this new leaf). As two leaves are replaced by one, this process removes the leaves one by one, until the total number of leaves in the tree is equal to k.

Example 10. *From the clustering in table 4.2, the binary decision tree of figure 4.4 can be obtained. In this running example, the decision tree leaves match exactly with the clusters.*

The example points out that the clustering and the binary decision tree may coincide. However, this is not always true since some tuples may fall into the wrong cluster or some clusters could be lost by the binary decision tree because, as previously explained, we do not grow a full tree and limit the number of leaves. It is possible

to obtain a correspondence between clustering and decision tree by using our alternative strategy, which allows to make disjunctions and therefore to fully grow the tree.

Obtaining SQL statements from a BDT with k leaves

Once the binary tree has been constructed, each leaf can be reached through a unique decision path. The decision path from the root of the tree to a specific leaf can be written as the conjunction of each decision encountered along the road. It is therefore straightforward to go from a decision tree to a SQL query. After exploring all the path in the tree, each conjunction can be directly injected in the *where* clause of a SQL query, and gives a new extension.

Example 11. *From the decision tree on figure 4.4, every extension from example 7 can be obtained easily.*

4.5.4 Algorithm proposal

In order to combine all the steps described previously, and to specify how the k-extensions set of a given query is to be computed, algorithm 3 is proposed hereafter. It takes a query Q and a database d as an input. In addition, it takes also an lower and upper bound for the size of the extensions, and allows to select the strategy for the decision tree: by default (when $tree_{strategy} = false$), the extensions only contain conjunctions, and the number of leaves in the tree is therefore limited. It works as follows:

- The algorithm is centered around function `Extend`, that computes an extension set for a given query on a database, given the number of extensions to compute and the selected strategy for the tree:
  - Line 12, we compute the result of the initial query. Clearly, if the size of the result is expected to be large, the online computation of extensions might take some time, and sampling could speed it up. Details on scaling and sampling are discussed in section 4.7.
  - Line 13, the clustering transforms the dataset wd into a labelled dataset lwd in which each tuple is labelled with the cluster it was assigned to. The quality of the clustering is evaluated line 14.

  - The decision tree is then built from lines 15 to 21, depending on the selected strategy. The conditions are extracted from the tree.
  - The conditions are sorted by length. They are then turned into extensions, and stored.
  - Finally, the considered $k - extensions$ set is returned along with the corresponding clustering quality.

- The main algorithm (lines 2-10) computes the extensions sets for the different values of k. The different extensions sets are ordered according to the quality of their clustering.

Thanks to the parameters of the algorithm, the user can have more influence on the extensions. However, the default parameters are already a good start for a novice user. This flexibility makes the extensions useful for different user profiles and different kind of data selection tasks.

The conditions from definition 8 are satisfied by the proposed algorithm as stated in the following property.

Property 6. *Let d a database over R, Q a query over R, and k an integer. $extension(Q, d, k)$ is a k-extensions set of Q, i.e for $\{Q_1, Q_2, ...Q_k\}$ in $Extend(Q, d, k)$, and for all $i, j \in 1..k$, $i \neq j$:*

$$Q_i \text{ is an extension of } Q \tag{4.1}$$

$$ans(Q_i, d) \cap ans(Q_j, d) = \emptyset \tag{4.2}$$

$$\bigcup_{i=1}^{k} ans(Q_i, d) = ans(Q, d) \tag{4.3}$$

Proof. (1) By construction, and with respect to definition 7, Q_i is an extension of Q

(2) At each node of the binary decision tree, there is one split leading to the creation of two child nodes. This split is done on one attribute A, for one threshold value t if A is numeric or a value v otherwise. The first child node takes all tuples in the dataset for which $A \leq t$ (respectively $A = v$), the second child node takes the rest, i.e tuples for which $A > t$ (respectively $A \neq v$)[2]. As a consequence, any tuple in the

[2]In case of null values on A, one of the two conditions of a split should integrate a test of the form A `IS NULL`, not to miss any tuple.

Algorithm 3: Query extensions sets configuration procedure

```
1  procedure Extension (Q, d, k_min = 2, k_max = 10, tree_strategy = false);
   Input  : A query Q over R,
            d a database over R,
            k_min the lower bound of query extensions set,
            k_max the upper bound of query extensions set,
            tree_strategy the strategy for the decision tree (true if it is different
   from default one)
   Output : ext_list a list of k extensions sets of Q, with sizes from k_min to k_max
2  ext_list = [] // the final list of extensions sets
3  scores = [] // list of clustering quality for each extensions set
4  for i = k_min; i ≤ k_max; i++ do
5  |  S_c, quality = Extend(Q, d, i, tree_strategy)
6  |  ext_list.append(S_c)
7  |  scores.append(quality)
8  end
9  ext_list = sort(ext_list, scores); // sort the extensions sets by
     silhouette score
10 return ext_list;
11 Function Extend(Q, d, k, tree_strategy = false):
12 |  wd = ans(Q, d) // wd: working data
13 |  lwd = kmeans(wd, tree_depth) // lwd: labelled wd
14 |  quality = silhouette(wd, tree_depth) // silhouette: clustering
   |   quality
15 |  if tree_strategy then // allow any depth for the tree
16 |  |  tree = DecisionTree(lwd)
17 |  |  conditions = getDisjunctionConjunction(tree) // get one
   |  |   extension per cluster
18 |  else
19 |  |  tree = DecisionTree(lwd, k)
20 |  |  conditions = getConjunction(tree) // get one extension per
   |  |   leaf
21 |  end
22 |  S_c = {}
23 |  conditions = sort(conditions); // (sort by length)
24 |  foreach c in conditions do
25 |  |  S_c = S_c ∪ σ_c(Q)
26 |  end
27 |  return S_c, quality;
```

dataset reaches one and only one node at each split, and therefore one and only one leaf of the tree. As each extension corresponds to one leaf, they all contain different tuples from $ans(Q, d)$.

(3) By property of extension we have $ans(Q_i, d) \subseteq ans(Q, d)$, so we obtain $\bigcup_{i=1}^{n} ans(Q_i, d) \subseteq ans(Q, d)$. Moreover, as a node splits on opposite conditions, any tuple from $ans(Q, d)$ satisfies one and only one condition at each split. Therefore any tuple t from $ans(Q, d)$ necessarily ends up in the result set of an extension, so there exists $i \in 1..n$ such that $t \in ans(Q_i, d)$ and therefore $\bigcup_{i=1}^{n} ans(Q_i, d) \supseteq ans(Q, d)$. □

4.6 ExplIQuE: presentation of the web prototype

4.6.1 Implementation

Our objective was to develop a prototype to query a relational database with the help of query extensions. To this end, we started by implementing all necessary algorithms using Python 3, with the help of the scikit-learn library [Ped+11]. In order for these algorithm to be easy to use, we developed a web application named ExplIQuE: Exploration Interface with Query Extensions.

The backend of the application relies on Flask[3] framework, and is therefore implemented using Python 3. It performs all the necessary computations to produce the extensions, query the database, etc.

The frontend, that is the interface between the user and the backend, is implemented using the React[4] javascript library.

Data is stored in a relational database that can be either MySQL or Oracle. Optional external files can be stored outside the database in a dedicated folder.

We will now present the different features offered by the interface.

[3] `http://flask.pocoo.org/`
[4] `https://reactjs.org/`

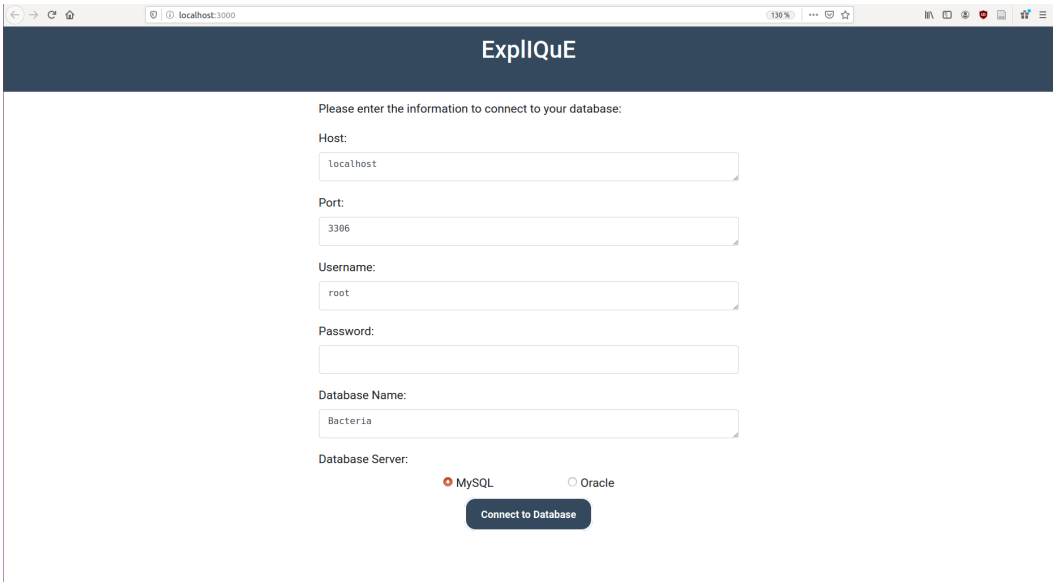


Fig. 4.5: Snapshot of ExplIQuE connection page

4.6.2 ExplIQuE's features

ExplQuE has been designed in order to connect to any database that the user has access to. As a result, it opens on a connection page, a snapshot of which is presented on figure 4.5: this page only ask for all the necessary information required to connect to the database.

After connection, the application opens its main page, presented on figure 4.6. On this second snapshot, all the different features of ExplIQuE are presented. The interface is divided into two main zones, that are clearly identified visually:

- The left zone of the interface is a standard SQL editor: the results are printed in a table below a given SQL statement. There is an is the additional button allowing to ask for the extensions sets of the current query.

- The right zones concerns all the extensions-related features. When a user asks for extensions in the SQL editor, they are printed in this zone.

In addition, to printing the extensions, ExplIQuE offers all the necessary tools to use the various configuration options to specify the parameters from algorithm 3 they can be accessed by clicking on the ⚙ symbol at the top of the extensions zone, therefore opening a configuration panel presented on figure 4.7, allowing to enter the parameters necessary to algorithm 3.

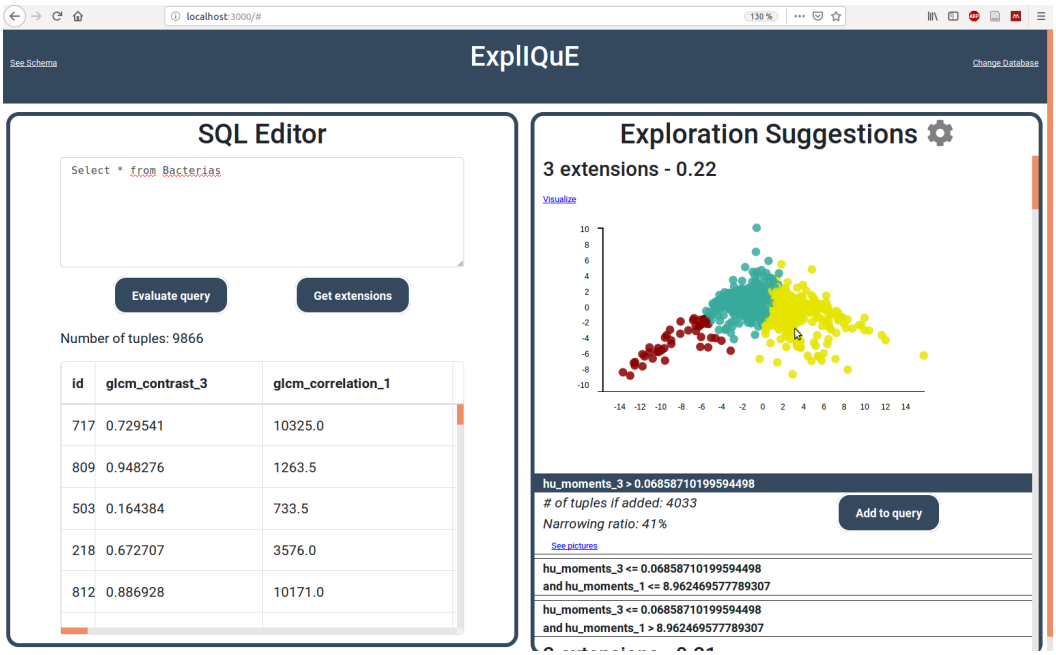


Fig. 4.6: Snapshot of ExplIQuE main page

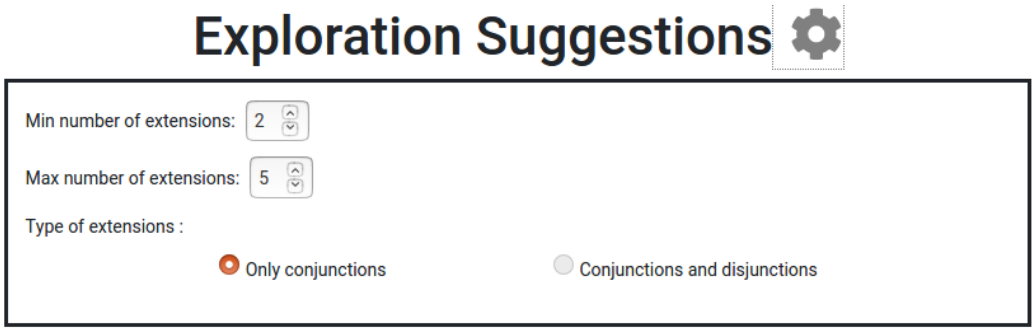


Fig. 4.7: Snapshot of ExplIQuE configuration panel

On figure 4.6, it can also be noticed that the silhouette coefficient used to rank the extensions sets is indicated, right next to the size of the set. Inside a given set, the extensions are also ranked, and to make it more explicit to the user, the user is given two indications to help her understand how restrictive a given extension is:

- The first one is how many tuples would be returned, if the considered extension is added to the current query.

- The second one is the narrowing ratio, which is the proportion of tuples from the initial query are removed by the considered extension. For a given query Q and one if its extensions Q_{ext}, the narrowing ratio is therefore defined as follows:

$$narrowing\_ratio = \frac{|ans(Q,d)| - |ans(Q_{ext},d)|}{|ans(Q,d)|}$$

Finally, the web interface offers two visualizations, aiming at facilitating the extension's selection:

- The first is available for any database. It is a scatterplot of the results of the query being extended, where the tuples are grouped by extension: this visualization is clearly visible on figure 4.6. The data is projected on two dimensions using principal component analysis (PCA), and each extension is presented using a different color. Moreover, the visualization is interactive, as the user can see the extension corresponding to the datapoints by moving the mouse over the scatterplot. The purpose of this visualization is to show in one glance to the user the size and dispersion of an extension, as well as how separated each extension is with respect to the others.

- The second visualization is only valid for databases in which the tuples can be associated to images: this happen for some classification, and these images can be particularly useful to chose an extension. In this particular situation, the images associated to the results of an extension can be displayed as a mosaic in ExplIQuE. This is useful for the user to easily identify the diversity of data that an extension represents: she might see visually that the extension contains homogeneous images, or on the opposite easily identify outliers. An example of such a visualization is given on figure 4.8, on a database containing pictures of bacteria.

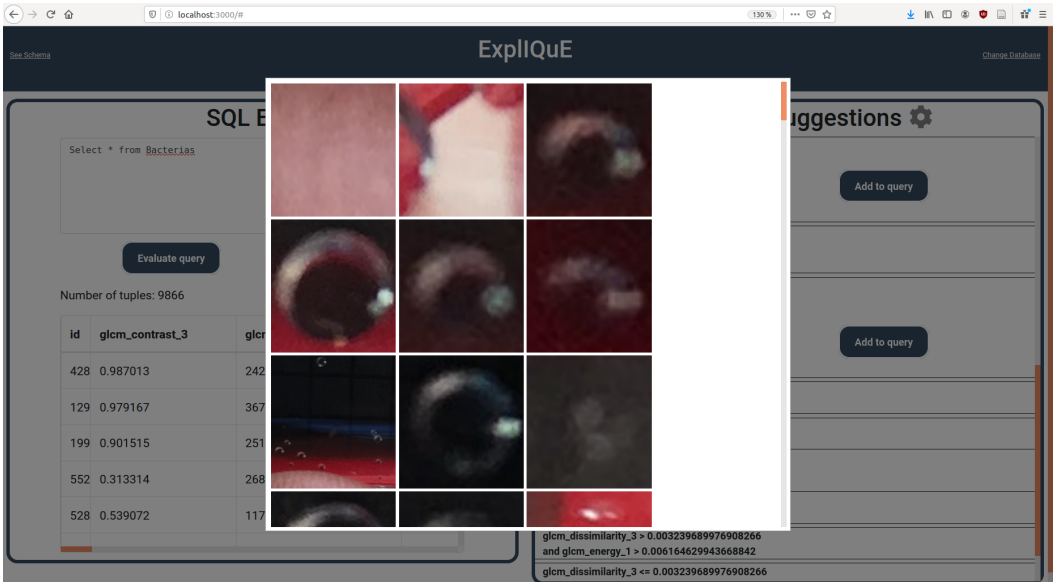


Fig. 4.8: Snapshot of ExplIQuE images visualization

4.7 Scaling experimentations

Whether it is on the web application or in an existing DBMS, one important feature of the extensions is that they should be computed quickly, so that the user does not have to wait to obtain the suggestions. Indeed these extensions can be seen as a form of query completion, similar to the completion offered by most search engines: the user therefore wants to get these results quickly.

The main bottleneck is the tuples clustering, because it requires to compute the distance for each pair of tuples in the result set, therefore having a polynomial complexity. As mentioned in section 4.5.4, one possible solution to limit the response time of the algorithm is to sample the tuples before computing the extensions: this allows to obtain a trade-off between user waiting time and the extension's precision. It is therefore necessary to evaluate what balance is acceptable, to have relevant extensions in a reasonable amount of time. It is also important to see if other parameters of the algorithms influence the extensions computation time. To this end, we performed an experimental evaluation to assess the correct sampling ratio. These experiments were run using a machine with an Intel Core i7-7600U (2.8 GHz) CPU and 16GB of memory.

Several parameters can influence the time necessary to compute the extensions:

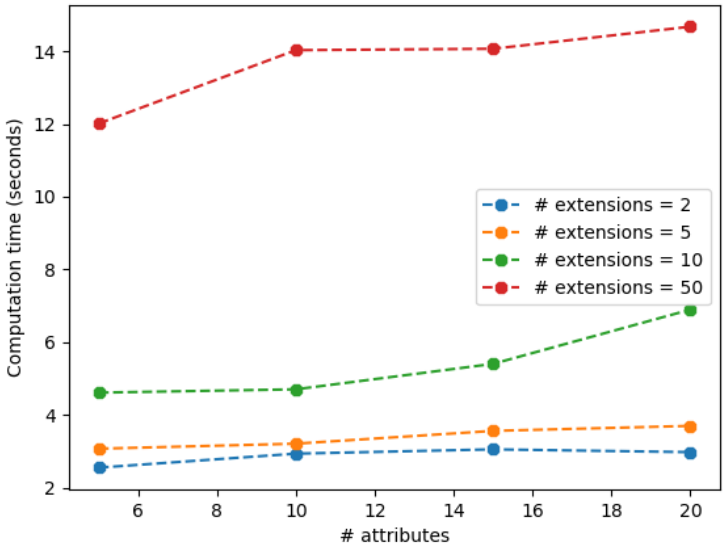


Fig. 4.9: Extensions computation time vs number of attributes and extensions (10 000 tuples)

- The number of attributes: this mostly influences the decision tree construction, as each node of the tree has to test all attributes to chose the best splitting condition.
- The number of tuples: as explained, this mostly influences the clustering
- The number of extensions to compute: the more there are, the more cluster there is, and the more classes there are for the decision tree, which is therefore bigger.

To determine what parameters influence the response time, we designed an experimentation to explore the influence of the parameters on the extension's computing time. The scaling experimentations were conducted on a database with data from the Large Synoptic Survey Telescope[5] containing 500 000 tuples over 25 attributes. It only contains one table, as the number of tables only influences the evaluation of the initial query, but not the extension process *per se*. The influence of the three aforementioned parameters was studied, with respect to the extension's computation time: number of tuples, number of attributes, and number of extensions to compute on figure. These parameters were studied two by two, to study every aspect of their influence, and are presented on figure 4.9, 4.10 and 4.11.

[5] `https://www.lsst.org/`

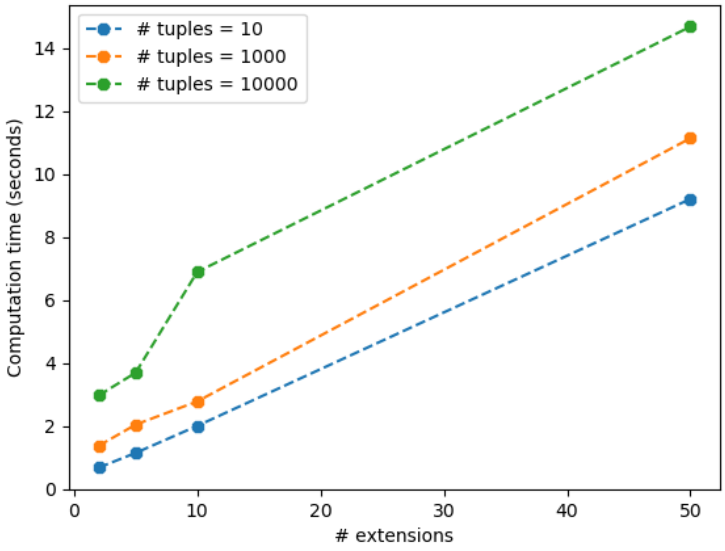


Fig. 4.10: Extensions computation time vs number of extensions and tuples (20 attributes)

From figure 4.9, it follows that the number of attributes barely influences the computation time. The number of extensions does seem to slightly increase it, which is also visible on figure 4.10, but it is negligible compared to the influence of the number of tuples: on figure 4.11, it is obvious that it is the most influential parameter. It is also clear that for a high number of tuples, the time necessary to produce the extensions is not acceptable for an online system, as users will not accept to wait more than a few seconds to obtain their results. This confirms the necessity of sampling the initial query's results, in order to provide the extensions in a reasonable amount of time.

From this first experimentation, it appears that up to 50 000 tuples, the extensions are computed in less than 20 seconds, and below 5 seconds for 30000 tuples. But this is not enough to determine if this is a good sampling size. Indeed, the quality of the extensions should also be taken into account: even if they are computed quickly, they should still be a good representation of the extensions obtained when taking all the tuples into account: to this end, a second round of experimentation was conducted, to evaluate the impact of sampling over the extension's results. First, the extensions for the query returning the entire database (500 000 tuples) were computed, using the full answer set. Then, for the same number of tuples, the extensions were computed, but over a sample of the initial query's result, using different sampling sizes (100, 1000, 5000, 10 000, 25 000, 50 000). Finally, the obtained extensions were compared to the ones obtained without sampling, by comparing how they

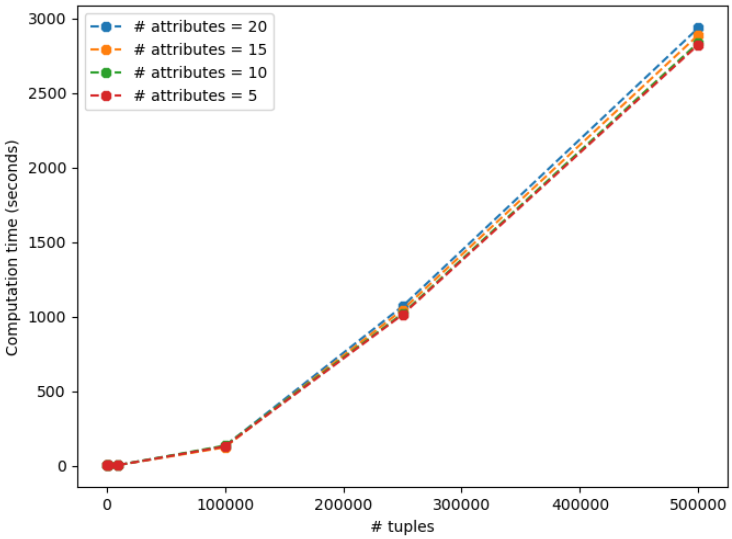


Fig. 4.11: Extensions computation time vs number of tuples and attributes (10 extensions)

	Y_1	Y_2	$\cdots$	Y_s	sums
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
sums	b_1	b_2	$\cdots$	b_s	

Tab. 4.3: Example of a contigency table

partition the initial tuples, using the adjusted rand index [Ran71]: given two partitions of the same set S of n elements, namely $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$, the two partitionings are compared using a contingency table such as represented on table 4.3. Each cell n_{ij} denotes the number of objects in common between X_i and Y_j : $n_{ij} = |X_i \cap Y_j|$. From this table, the adjusted rand index (ARI) is defined as:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}$$

In addition, two sampling strategies were compared: random sampling, and systematic sampling (see [Coc46]). The results of these experimentations are presented on figure 4.12. Systematic sampling requires slightly more tuples to reach the same

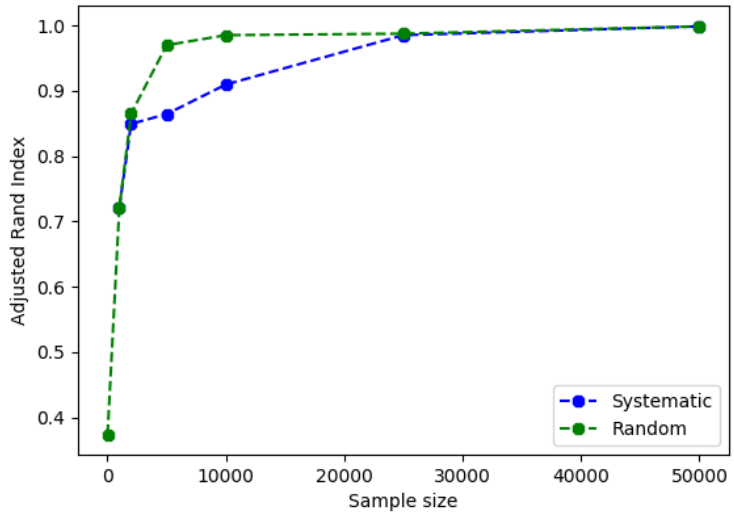


Fig. 4.12: Comparison of extensions quality against the sampling size

behavior as random sampling. However, both samplings rapidly reach good results in comparison to the full data extensions, as with a sampling of 1000 tuples, the adjusted rand index is already above 0.8 (1 meaning results identical to non sampled dataset). With 10 000 tuples, the score reaches 0.98 with random sampling, and the computation time with respect to figure 4.11 is around 5 seconds. These results show that sampling can considerably speed up the extension computation while keeping a very acceptable quality. Those results were confirmed on several other databases not presented in this paper, showing that sampling provides an excellent trade-off between computation time and extension's quality.

It should also be noted that in order to speed up more the extensions computation, it is possible to store the results obtained for one query, in order to reuse it later to avoid recomputing extensions that had already been obtained before. As a result, for each query, it is possible to look in the query log to make sure its extensions are not already saved in the system. Finally, as the most expensive part of the process is the clustering, that requires to compute distances between each pair of tuples, we also propose to save the distance matrices, so that they can be directly reused in other extensions computations. Using these two methods, even when an extensions set takes some time to compute, it will only happen once, and it will also speed up extensions computation for following similar queries.

4.8 User experimentation

In addition to being able to return the extensions in a reasonable amount of time, it appeared necessary to assess whether or not such extensions are useful, for a user with an imprecise query and/or in a exploratory setting. It was therefore decided to perform a user experimentation, to answer the two following main questions:

1. In terms of writing time, is it faster to reach a desired set of tuples using extension?
2. How well is the extension tool accepted by users?

4.8.1 Organization

To answer these questions, an experimentation was designed, in the form of a SQL competition for 70 computer science students (last year bachelor students and master students). It should be noted that ExplIQuE had not yet been developed at the time of this experimentation, that therefore took place using a previous version on the interface, visible on figure 4.13. As a consequence, this previous version did not have the possibility to fully configure the interface further than choosing the number of extensions to compute. The features visible on figure 4.13 are the following ones:

(A) SQL editor to write the queries

(B) Ask for an extensions set, and choose its size

(C) See the query results

(D) See the extensions, and the number of tuples returned if it is added to the current query

All students from the computer science department were welcome to participate, as long as they had at least basic knowledge in SQL and data management. They were initially only told that they would have to address several SQL-related challenges, and agreed to participate to a one-hour experimentation.

Prior to the experiment, participants were randomly divided into two groups. The division was however balanced in terms of number of students from each level (bachelor, first year and second year master students): this was done to avoid a

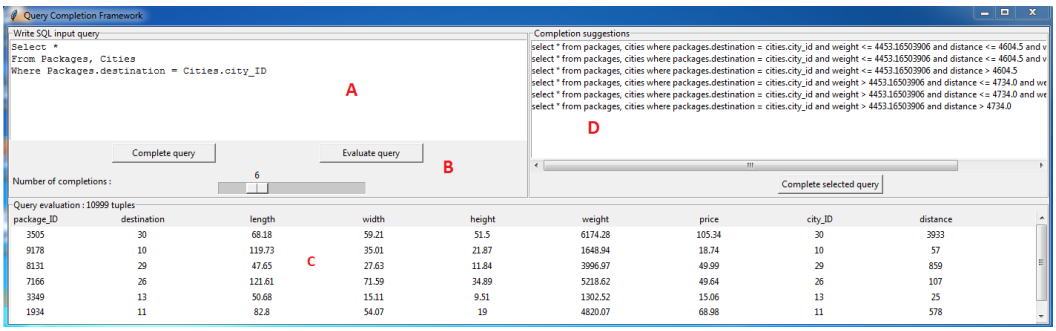


Fig. 4.13: Snapshot of the interface used for the experimentations (ExplIQuE's ancestor)

bias because of an uneven distribution that could cause a group to perform better because it has more experienced students.

The experiment required to evaluate SQL queries on a database. For this, the first group (referred to as group EXT from now on) had access to the extension tool, from figure 4.13. The second group (group NoEXT from now on) had a tool similar to the one of group EXT visually, and with the same classic SQL features, but without the extension possibility. Thanks to this setting, it was therefore possible to compare the results of the two groups, i.e to see the difference between groups with and without extension possibility, while working under similar conditions. Because the design of such questions was the most delicate part of the experiment setup, we will discuss their creation in the following section.

4.8.2 Design of the test

To test the query extension interface, we had to design *imprecise* questions, to put the students in the situation for which the extensions were designed. The design of these questions was delicate, to find the right level of difficulty. When conceiving the questions, our purpose was to propose a fair situation for groups EXT and NoEXT. For this reason, we eliminated two types of questions :

- Questions that were trivial with extension, but impossible to do without it: this would have given an unfair advantage for group EXT, and any result on such question would have been meaningless.

- Questions for which extension has no interest: queries with empty result sets, dates comparison, specific operators from DBMS, etc. Such questions are out of the scope of what the experiment is meant to evaluate.

All questions were designed according to the same pattern: they exposed a scenario,asked a question on it, and then asked to find out the SQL query to solve it. The questions were separated into two categories:

- The first three were classic SQL queries, that are directly and easily transformable into SQL queries, similar to the ones that can be found in an "introduction to SQL" exercise. On such queries, the extensions are not necessary nor useful: these questions were used to verify that each participant really had basic SQL skills, and that groups EXT and NoEXT had similar results, and were therefore well balanced. As a result, if both groups had similar results on these questions, any difference that could appear for the following ones were not due to a bias from the participants' level.

- The other questions (number 4 to 10) were the imprecise ones: their conversion into an SQL query was not straightforward: the questions required some fumbling and several trials to reach the expected results, as in an exploratory context. The questions' specification was less strict, as selection conditions were not specified in terms of numbers, but rather as imprecise queries, using adjectives such as *higher, bigger, lower, above average, low, etc*. However, the final expected SQL query required to identify numerical conditions to discriminate between the tuples, and to translate the description of data given in the question.

Even with those specifications, the level of difficulty for the second group of questions was not easy to settle: formulating those *vague* questions requires to choose carefully the vocabulary used in the question. Moreover, even though the test aimed at reproducing an exploratory context, it was necessary to give the participants some clues, so that they could have some idea of when their query was precise enough. We therefore indicated the approximate number of tuples the query was supposed to return.

In addition, we also proposed a question based on a visualization (presented on figure 4.14), asking participants to formulate queries that would return a part of these visualizations. The objective was to transform a visual pattern into a query, so they had to identify the pertinent conditions to characterize the given pattern. This part of the experimentation then inspired the additional functionalities for the interface.

4.8.3 Test's questions

Based on all these requirements, the 10 questions were designed, as well as the scenario and database that surrounded these questions. The participants were given the following scenario:

> "You're a new member of a post office, in charge of packages. When you're not at the front desk taking care of customers, you have access to data recorded about the packages sent from your post office. For simplification, we will focus on the packages leaving the post office to other destinations".

They were also given the SQl code that created the database's schema, which is as follows:

```
CREATE TABLE Cities(
    city_ID DECIMAL,
    distance DECIMAL,
    PRIMARY KEY (city_ID)
)

CREATE TABLE Packages(
     package_ID DECIMAL,
     destination DECIMAL,
     length DECIMAL,
     width DECIMAL,
     height DECIMAL,
     weight DECIMAL,
     price DECIMAL,
     PRIMARY KEY (id_colis)
     FOREIGN KEY (destination)
          references Villes(id_ville)
)
```

Table `Cities` contains 30 tuples, and `Packages` 11000 tuples. In addition, the participants were given this final indication:

> Table `Packages` has one entry per package that left your post office. From the destination of a package, you can see how far it was sent, by joining tables `Packages` (11000 tuples) and `Cities` (30 tuples) on attributes `destination` and `city_ID`.

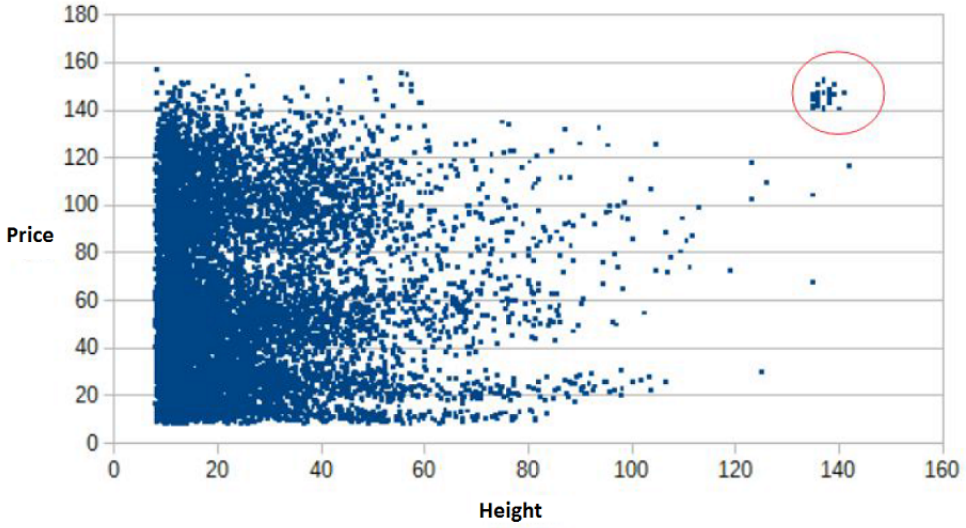


Fig. 4.14: Data visualization for question 4 of the user experimentation

Questions are ordered from easiest to hardest :you should therefore answer them in the given order. First three questions are simple, while the others are voluntary more complex, and finding the required SQL query in questions 4 to 10 required more exploration.

The questions were then as follows:

1. This first question is here so that you can get familiar with the data and the tools at your disposal. Please test the two tools (SQL software and online form for answers) with the following query, that is a join between the two tables (Expected result size: 10 999 tuples):

```
Select *
From Packages, Cities
Where Packages.destination = Cities.city_ID
```

2. Maximum size limit authorized for a package is 9000 grams. However, some exceed this limit without being detected. Give the query to obtain the ID of packages whose weight exceed this limit. (Expected result size: 73 tuples)

3. What query can you write to obtain the average length of packages sent less than 100 kilometers from your post office? (Expected result size: 1 tuple)

4. A little bit interested by data analysis, a colleague of yours had, with a spreadsheet, visualized some curves from the database. By plotting packages prices against their height, he/she had noticed a group of packages very distinct and well separated from the others, which is presented on figure 4.14, and circled in red. Can you find the query that returns all packages belonging to this group? (Expected result size: 33 tuples)

5. According to some colleagues who've been working here for years, heaviest packages are the ones going to very distant destinations. The intuition behind this is that sending a package far away is expansive, so customers put many things in one package to compensate. Can you identify packages that do not comply with this, i.e that are not heavy but are sent far away? (Expected result size: 13 tuples)

6. Once at the regional sorting center, packages go through a machine that automatically sorts them according to their destination. However, this machine is sometimes defective. Indeed, when a package is less than 480g, the machine does not always detect it, and an operator has to take it and process it manually. This phenomenon is marginal, but more likely to happen if in addition to its light weight, the packages is small regarding its length and width. On all packages registered in your database, 12 have caused a problem. Which query can identify those 12 packages?

7. Some packages are sent to a city that is very close to your post office, less than 10km away. Moreover, some are very light (less than 550g), and you wonder why people pay the post office to transport them while they would quite easily do it themselves. One of your colleagues has an hypothesis : maybe those packages are cumbersome and therefore hard to transport. Can you identify packages validating this hypothesis? (Expected result size: 8 tuples)

8. A customer arrives at the post office, because he needs the ID of a package he had send, but isn't able to find. In order to help him, he gives you a few informations: the package was light, less than 450g and its dimensions (mainly length and width) were surprisingly big in regard to its weight. Can you give the query returning such a package? (Expected result size: 1 tuple)

9. When working at the front desk, one of your colleagues made a mistakes on four on the packages he registered. Luckily, he remembers their length was above 140cm, and he therefore applied a special tarification, as those kind of packages are more complicated to deliver due to their size. But he applied the wrong tarification, and those packages have therefore an abnormally elevated price. Can you identify those packages? (Expected result size: 4)

10. At question 2, you showed that 73 packages are above the weight limit. But your colleagues in charge of putting packages in the trucks say that a third of packages are really heavy, and require two employees to be lifted, in order to avoid back pains. Can you modify the query for question 2 in order to identify those packages? (Expected result size : 3073 tuples)

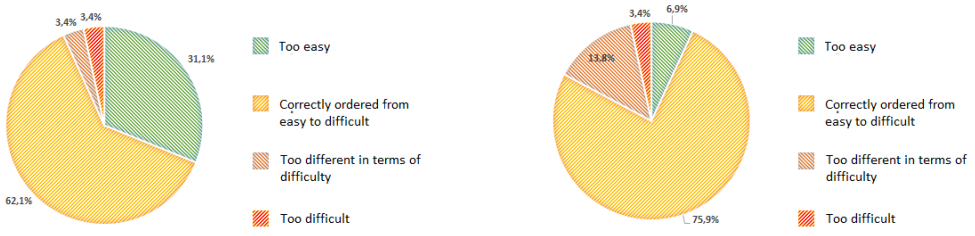
4.8.4 Test's setup

Participants had one hour to answer the 10 questions. They used the tool to write queries and evaluate them on the database, and once they thought they had the right query, had to submit it online. They were not told whether their answer was right or not, as in real-life where only the data analyst can know if she obtained the data she wanted, and to avoid participants that would just try several possibilities to bypass the difficulty of the question.

During the hour of experimentation, we were able to monitor the time each participant spent on each question. After the experiment, we checked whether the answers they submitted were correct or not. Moreover, we were able to say, for each question, if participants from group EXT had used an extension or not to select the data.

At the same time, group EXT had to adapt to the extension tool, and they did not receive any specific training on how to use the tool before the test. They were only given a one-page instruction sheet on how query extension worked. But they did not get any additional time, and had to use the hour to both answer the questions and master the extensions (even though they were not forced to use it). This was done to avoid influencing them on their use of the extension tool, and to see how they would adapt to this new functionality.

Finally, after completing the 10 questions, participants were asked to answer a quick survey to collect their opinions and feelings on the experiment.



(a) Difficulty of questions perceived by group EXT

(b) Difficulty of questions perceived by group NoEXT

Fig. 4.15: Participants' feedback on questions difficulty

4.8.5 Results

Validation of experimental setup

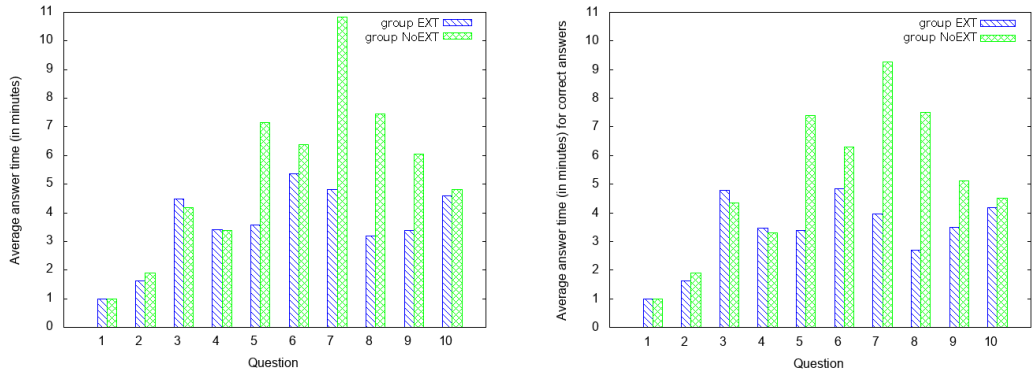
First, we validated that the assumptions that had been made regarding the experimental setup were correct, by analyzing the feedbacks given by the participants. The feeling of participants regarding the difficulty of the questions is given on figures 4.15a and 4.15b. In both groups, only 3.4% of participants felt like questions were too difficult. In both cases, the majority of participants felt like questions were correctly ordered from easiest to more difficult. The only difference is that more people in group EXT felt like questions were too easy (31.1% against 6.9%), as the extension tool helped them in answering the questions that were supposed to be really difficult.

On this first evaluation, we met our objectives with respect to our questions difficulty. This is also an indication that query extension can make answering SQL questions easier for users.

Query writing time

In a second phase, we evaluated the time spent by each participant on the questions, to see if the extensions allowed to identify the desired results faster. The first result that is interesting to look at is how much time each group spent on average answering each question: those results are presented on figure 4.16a. There are several interesting points to notice on this figure :

- For questions 1 to 3, the results of the two groups are similar, which was the initial objective. When extension was not necessary, the performance of both groups were equivalent.



(a) All answers **(b)** Only correct answers

Fig. 4.16: Histogram of average answering time per question, for group EXT and group NoEXT

- Question 4 was still easy for both group, as could be expected as the visualization was here to help . Even though the use of extension could have helped on this question, it does not seem to have made a difference, as the average answering time is very similar for both groups. This means that a good and efficient visualization, when efficient, can also be useful.

- For questions 5 to 10, the difference between the two groups is much more important and it is clear that group EXT performed considerably faster than group NoEXT. This is a strong argument to support the fact that SQL query extension can indeed make the SQL query writing faster. The difference is stronger for questions 7 and 8, which seem to have been the most difficult questions for participants.

However, figure 4.16a takes into account all answers from participants, which means that some of those answers might be wrong. And a participant who did not give a good answer might have spent a lot of time on a question looking for the answer without finding it, or on the contrary given up quickly as he did not know how to find the answer. For this reason, figure 4.16a was recomputed, taking into account only the answering time from participants who had given the correct answer for the considered question: such results are presented on figure 4.16b. The tendency is similar, and group EXT still performs considerably faster than group NoEXT. Actually, results from group EXT are even slightly better, especially for more complex queries like for question 7 and 8.

To understand the behavior of participants, it is possible to look at the boxplot of answering time per question for each group, on figure 4.17, that only takes into account correct answers. The main observation is that results of group EXT are

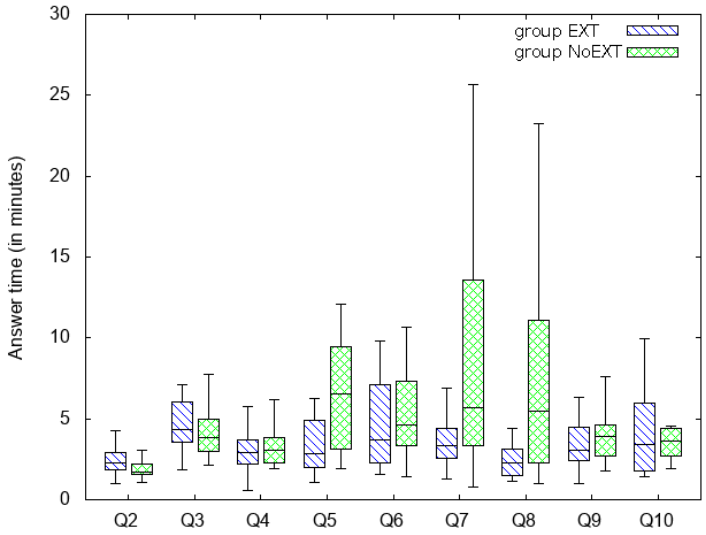


Fig. 4.17: Boxplot of answering time per question, for groups EXT and NoEXT, only for correct answers

much more packed than for group NoEXT: participants who had access to extension had a way to help them if they were stuck on a question, contrary to group NoEXT participants who had to search by themselves until they identified the answer. This is flagrant once again for question 7, where someone spent more than 25 minutes looking for the answer.

On this second evaluation, it is therefore possible to draw the following conclusion: when evaluated in similar conditions, the group with access to extensions performed faster than the group with only classic SQL tools. This therefore answers one of our initial questions. To answer the second question, it is now necessary to evaluate if the participants got used to the extensions.

Extension tool acceptance

As mentioned previously, it was also possible to say whether a participant had used extension for a given question or not. The proportion of extension use per question is presented on figure 4.18. We only presented question 4 to 10 on which extension was possible. It can be seen on this figure that participants did not always use the extension tool. In total, 70% of participants from group EXT used query extension at least once, while the others completed the test without using it. On average, on the seven questions for which completion was possible with the extensions, participants used it 4.2 times.

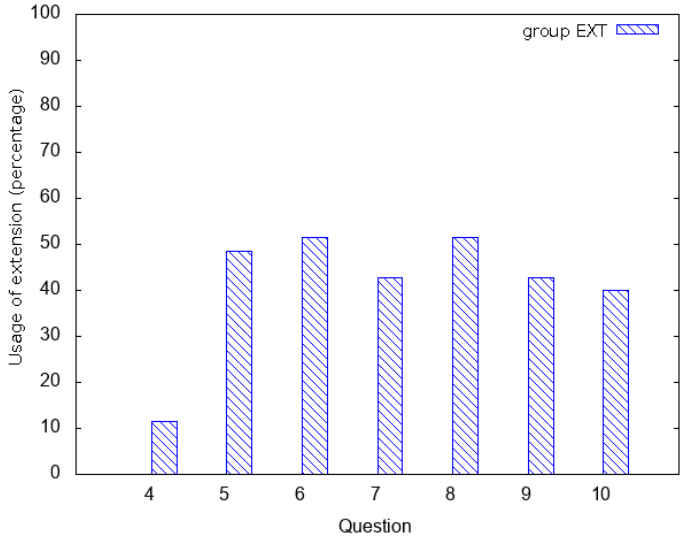


Fig. 4.18: Percentage of extensions usage for questions 4 to 10 for group EXT

Additional results were analyzed to understand those observations. We first analyzed the way participants had used extension: on figure 4.19, interesting patterns can be observed. The main observation to do is that once participants have used query extension for a question, they are very likely to use again in the next question. This is indicated by the continuous blue lines on this figure. This is a really important result, as it showed that once a user has understood the utility of extension, she will use it again. This observation is particularly true for participants number 1 to 13, which in addition did not make many mistakes. Participants 14 to 19 also used extensions a lot after their first use, but made more mistakes: when looking at their answering time, it seems that they did not have much time to complete the last questions, and therefore might have been in a rush and did not give correct answers. Finally, participants 20 to 24 seem to have tested extension, but preferred to finish the test without using it.

On figure 4.20, we divided group EXT into two groups for each question: participants who had submitted a query generated with extension (group EXT1), and others participants from group EXT in group EXT2. We then compared their average answering time for each questions, as well as for group NoEXT. It should be noticed that for each question groups EXT1 and EXT2 might be different as participants who used extension are different from one question to another. First, on question 4, group EXT1 is slower: as it is the first question on which extension could be used, we interpret this as the time necessary for participant to get familiar with the extension tool. But for all the other questions, group EXT1, that used the extensions answers the query faster than the two other groups.

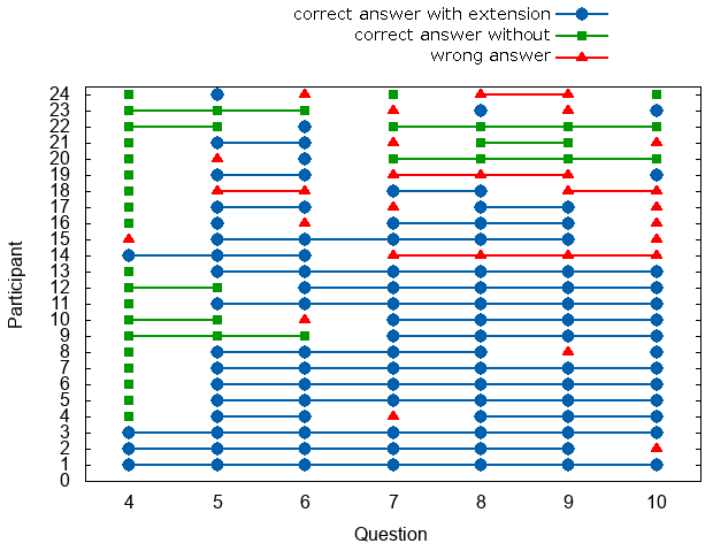


Fig. 4.19: Type of answer per question, for participants who used the extensions at least once

Moreover, even though group EXT2 answered without extension, its behavior is different from group NoEXT on question 4 to 10. Indeed, except for question 10 where it is the slowest group (but on previous figures, question 10 always has specific behaviors), the tendency of group EXT2 is closer to the one of group EXT1 than to the one of group NoEXT. This is explained by the fact that participants who did not use extension in group EXT were students good enough in SQL to be able to answer the question quickly: for them, taking the time to understand the extension tool would have been a waste of time as they were comfortable enough in SQL and had enough information, to succeed the test without it.

Take-out lessons from results

These experimentations have allowed to answer our two initial question on the proposed query extensions tool. The first question concerned the query writing time: based on the results, it does appear faster to reach the desired tuples using the extensions, when the initial query is imprecise.

We also answered our second question, showing the tool is well accepted by users, if they feel the need to use it: they are likely to use it again once they have tested the extensions. Therefore, the use of extensions was not a single isolated try by participants, but that a first use encouraged them to use it again.

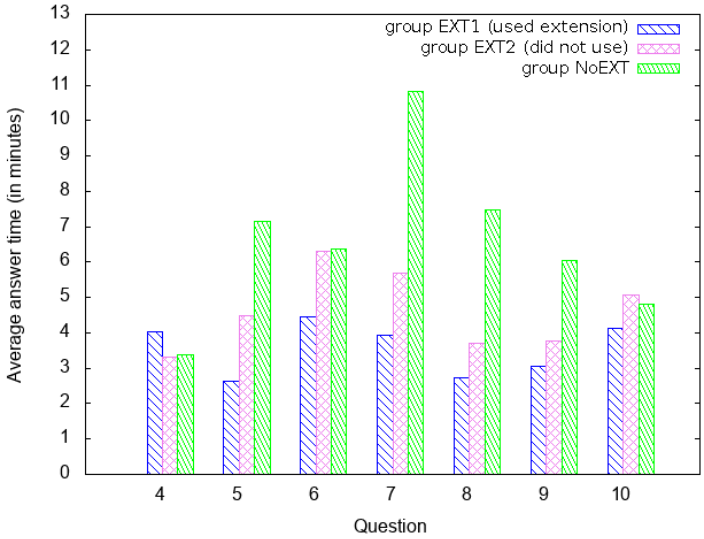


Fig. 4.20: Histogram of average answering time for questions 4 to 10, for groups EXT1, EXT2 and NoEXT

4.9 Conclusion and perspectives

In this chapter, we considered the problem of selecting the relevant data in a relational database, by making two assumptions: the user might not exactly know what she is looking for, and her manual work should be kept minimal. To this end, we therefore proposed an exploratory solution, but that relies on SQL and returns a SQL query, so that the user is kept in a familiar environment and can reuse the query in other contexts, or on other instances of the database. We also tried to limit the necessary user input, and for the required ones, we proposed solutions that could be applied to make suggestions of what values to used for the parameters.

The proposed solution consist in, for a given SQL, the suggestion of an extensions-set, that have a twofold objective: they give refinement solutions for the considered query, while also helping the user understand what is contained in the current answer set. Such extensions are meant to assist user confronted to imprecise queries, when they know the data they are looking for, but are having a hard time translating it into a formal SQL query. In this setting, the user can engage in an iterative process, by starting with a general query, and refining it using the extensions. Moreover, the utility of such extensions has been demonstrated through our user experimentation, showing that even if the process does not involve any complex algorithm, it does provide interesting suggestions, by simply grouping tuples that are more likely to be selected together by a user, and computing the selection conditions that characterize them.

Such solutions are very valuable to identify the desired data in a relational database: contrary to chapter 3, the proposed solution does provide a SQL query, rather than just a set of tuples, therefore summarizing the whole set of tuples in one query, and making it reusable. These extensions is also a way to integrate knowledge usually provided by machine learning techniques externally, directly into the database, therefore keeping the user in the database context for the data selection.

Several extensions of this work could be envisioned, to allow for more complex queries, and to remove even more work from the user: for example, it could be interesting to adapt the extensions to directly identify queries to start from, rather than having to ask the user for a first query input. Alternatively, it would be relevant to extend other parts of the query: for example suggest other tables to join, other columns to select, or even aggregate functions for the `group by` clause.

4.9.1 Towards chapter 5

The extensions-based solution in this chapter is a practical answer to the problem of identifying and selecting the relevant data in a relational database. But the data is selected to answer a question, to solve a problem, to perform a specific task. As a result, the data selection is often only the first step of a bigger process, even though it can be a time consuming process. And because the user selects this data with a given goal in mind, she is usually the only one able to say whether or not she has selected the required data: that can also be a difficult burden, especially if the number of tuples to select is high. For this reason, after data selection and before moving on to the central task of the process, it might be interesting to consider an intermediary step, that would evaluate for the user if the selected data is appropriate for the considered task. We will therefore consider this step, in the case of predictive models, in the following chapter.

5 Evaluating data adequacy with the predictive task

Chapter's outline

In this chapter, we try to assess, given the data selected for building a predictive model, how much it "fits" the considered classification task. To this end, we propose an upper bound on the accuracy of classifiers, based on the satisfaction of the functional dependency between the features and the class to predict. We propose experiments on synthetic and real data, and discuss how to adapt to the different domains of the features. We also expose two industrial collaborations with Cemafroid and Airbus Helicopters.

We start with the related work in section 5.2. We then present our FD-based approach in section 5.3. We discuss how to adapt to different features domains in section 5.5. We then explain how to discuss models limitation in section 5.6. Finally, we propose to refine the data selection using contextualization in section 5.7, before concluding in section 5.8.

5.1 Introduction

5.1.1 Data selection for predictive models

Usually, data selection is just the first step necessary to move to the main part of the process, which is the building of a predictive model. For sake of clarity, in this chapter, we will mainly focus on classification to produce, therefore on dataset where there is a class to predict, with a small number of possible values. We will discuss later on how to deal with regression datasets. After data selection, there is a fundamental question to answer: how can the user know that the selected data is enough, or is adequate, for her considered problem? Other than her own instinct and knowledge, are there assurances that can be given to assess the pertinence of the data for a given problem? These questions are crucial, as databases users ask

them every time they select data, in order to know if they should keep querying and exploring the database, or if they can move on to the next step in their process. Such questions relate to what Lise Getoor calls "the smell test" [Get19]: it should feel like there is a plausible connection between the features and the class to predict, otherwise there is no reason to train a classifier.

Answering these questions can be costly: if the user spends too much time on the data selection, she is delaying the process that is actually using the data, and that is usually what she can get value from. On the opposite, if she moves to the next step too early, she might get poor results, meaning she will eventually have to reperform the data selection, loosing even more time in total. In this chapter, the general question can thus be summarized as follows:

> Can we evaluate whether or not the selected data fits the predictive task?

Of course, this is an extremely large and difficult question: depending on the considered task, the answer and the evaluation tools can greatly differ. It is also extremely difficult to propose a general answer, as such questions are necessarily tightly intertwined with the application domain related to the data. Predictive models are widely used nowadays, in many industrial contexts, and they are are a real source of potential value for many companies. Moreover, the previous research question is even more true in this context, as the data used for the construction of such predictive models has a direct impact on the performances of such models. As a result, the research question of this chapter can be refined as follows:

> How to evaluate the adequacy between the data selected for a predictive model and the predictive task?

Indeed, one of the main objective of predictive models is to have accurate ones, in the sense that their prediction should be correct as many times as possible. This can seem trivial, but reaching this objective can turn out to be difficult in practice, because it highly depends on the data available in practice. Based on this data, it is important to be able to decide whether or not it can be used to build a predictive model, or if it would just be like trying to find something that does not even exists in the data.

If the adequacy is evaluated , it can then be used to decide if the training set should be modified (i.e refine the data selection), or if the predictive model can be trained on the considered data. Ideally, when the quality is not satisfactory, this evaluation of the dataset should also give an indication of what could be done to improve it, or at least explain what is currently wrong with the selected data, with respect to

the predictive task. This would be extremely helpful for the user, that instead of just knowing she has to reselect her data, could have a better understanding of what is wrong with the initial selection.

5.1.2 Trusting predictive models

Over the last decade, building classification models, using machine learning algorithms, has become completely mainstream. it has never been more easy to build a model on any dataset, thanks to the development of many machine learning libraries, that allow to train and test a model in a few steps, sometimes without having to write a single line of code. This is extremely practical, but also pretty perilous, as it is tempting to just dive into the model's building, without really questioning the data that is used to train it. In the worst cases, this means that it is possible to build models using data on which there is nothing to learn from: of course, the scores used to evaluate the model's performances will therefore be low, if not disastrous. But it is unfortunate to wait for the whole model to be trained and tested before realizing something is wrong with the initial dataset: it therefore appears necessary to be able to take a step back, and to spend some time to assess the data's adequacy with the predictive task, in most of machine learning applications.

Another problem that might arise during the construction of a predictive model is the *trust* user have in the model: in order for it to be used, it is important to be able to understand how it works, and on what the predictions it makes are based. However, many algorithms can look like black boxes to users, especially if they are not familiar with machine learning algorithms. Indeed, if some algorithms like decision trees can be interpreted, for others, it is much more difficult to understand their decision process (for example neural networks). Nonetheless, it is crucial for domain experts to be able to trust the models, so that they are not reluctant to use them: these models are sometimes used to take crucial business decisions, with a possibly use impact for a company. Domain experts therefore have to be 100% confident in the predictions made by the models. As a result, it is necessary to develop solutions to explain, interpret and evaluate how and why a model makes given predictions.

Being able to explain the prediction of a given model is not only a matter of being able to explain an algorithm: essentially, the output of a given model almost entirely depend on the data used to train it. Additionally, the data is also what matters for a domain expert, because it is what she understands, and what she is interested in. The dataset selected for the predictive task is therefore the heart of the process, and should be used to explain to the domain expert why or why not the considered

dataset can satisfactorily be used to train a training model, and therefore why or why not she can trust the prediction made by such model.

5.1.3 Existence of a function and dependencies

If we recap the previous observations, the objectives of this chapter can be summarized into the two following main points:

- Given a dataset, assess if it will allow to train a predictive model for a given predictive task.
- Explain what makes a dataset a good fit (or bad) for the considered task.

These two tasks are intricately correlated, and each bring a complementary information to a domain expert, with respect to her dataset. Ideally, it would therefore be interesting to propose a solution that can address the two problems simultaneously. In this chapter, we propose to consider these two research problems through the simple mathematical notion of a function: ultimately, a predictive model is an algorithm seeking to define the function that maps the input data to the class/values to predict. However, some datasets are more likely to correspond to a function than others, because there are some conditions that have to hold in order to be able to define such a function. As a result, the question becomes :

> How can we evaluate how likely it is for a predictive dataset to follow a function, and how can we identify the blocking points to the existence of such a function?

The notion of *function* is a widely used one, so this question can be tackled using many different approaches. In the specific case of predictive datasets, we propose to consider the notion of data dependencies, by considering that data satisfying a dependency can be characterized by a function. We especially consider functional dependencies, and more specifically the FD between the features used for the prediction, and the class/values to predict: if the FD $\{features\} \rightarrow class$ is satisfied (or almost satisfied), then there exists a function from the features to the class: therefore, a machine learning algorithm might be able to find such function. If the dependency is not satisfied, then it is necessary to find the data that prevents its satisfaction, to understand what is going on, and to explain it to the domain expert. To summarize, we argue that a functional dependency characterizes the existence of a function that a predictive algorithm seeks to define. To the best of our knwoledge this relationship is new, despite its simplicity.

Considering a dataset, such an approach is thus a way to determine if the selected data is likely to produce a coherent predictive model, because there exists a function in the data that can be defined and approximated, or if the dataset does not have a satisfying quality for the considered task. In addition, we propose to reuse the well-known notion of counterexamples, that can identify pairs of tuples that are in contradiction with the existence of the FD. As a result, such counterexamples can identify the blockages that can lower the model's performances, and therefore be used for discussion with the domain expert, and eventually to refine the initial data selection.

5.1.4 Problem statement

The purpose of this chapter is therefore to use the FD between the features and the class to assess the adequacy of the selected data with the predictive task. The objective is to propose a methodology that can be used to determine if the selected data is coherent with the predictive task considered, to help the domain expert decide whether or not to spend more time on the data selection. In this setting, this chapter is based on the following simple yet powerful observation:

> Given a dataset r over $\{A_1, \ldots, A_n, C\}$ where C is the class to be predicted, classification algorithms seek to find out a function to predict an output (C value) based on a given input ($A_1, \ldots, A_n$ values) ; the satisfaction of the functional dependency $A_1, \ldots, A_n \rightarrow C$ in r expresses the existence of that function

As a consequence, we have to quantify the satisfaction of the functional dependency: if it is not satisfied, how far is the data from the satisfaction? Based on this, the idea is to use this satisfaction to estimate the performances of any classifier on the considered dataset. The counterexamples preventing the satisfaction of the DF should also be used to explain these results to domain experts.

Example 12. *Let's take a small dataset from table 5.1 as an example. This is the dataset about passenger of the famous Titanic, with their ticket class (first or second), their age range (child or adult), their gender, and whether or not they survived. The purpose of this problem is to predict if a passenger has survived or not. Such an analysis can then be used to determine if some passengers were more likely to survive than the other.*

In this dataset, the available attributes are not enough to determine the class. For example, tuples t_2 and t_5 both concern male children in second class, however one

id	Ticket	Age	Gender	Survived
t_1	1st	Child	Female	no
t_2	2nd	Child	Male	yes
t_3	1st	Adult	Male	no
t_4	2nd	Adult	Female	yes
t_5	2nd	Child	Male	no
t_6	2nd	Child	Male	yes
t_7	1st	Adult	Male	no
t_8	1st	Adult	Male	yes
t_9	2nd	Child	Male	yes
t_{10}	1st	Child	Female	yes

Tab. 5.1: Toy dataset: Titanic relation

survived while the other did not. Similarly, the two adult males in first class from tuples t_7 and t_8 had two different outcomes. Whatever the classifier, it will irremediably misclassify at least one of them.

This very simple example shows how the satisfaction of the functional dependency $Ticket, Age, Gender \rightarrow Survived$ in a classification dataset highlights the limits a classifier reaches on a dataset, and how counterexamples are a way to materialize such limitations.

The contributions of this chapter can therefore be summarized as follows:

- First, we expose our approach to estimate an upper bound of a classifier given a dataset, based on the satisfaction of the functional dependency between the features and the class to predict. We propose experimentations on synthetic data to validate the relationship between the accuracy of a classifier and such a dependency. These synthetic datasets are generated in order to be "as difficult as" possible for a classifier, and we propose an algorithm for their generation.
- We then discuss how apply this approach on real datasets, that often contain different datatypes, and or which the evaluation of the satisfaction of FDs might therefore have to be adapted.
- Thirdly, we examine how to use this approach to discuss the limitations of the models with domain experts, especially using the notion of counterexamples. We present the results and feedbacks from an industrial collaboration with Cemafroid, a company dealing with the prediction of the ageing of thermic engines for refrigerated vehicles.
- Finally, we explore the case when the dataset does not appear satisfying to be used for the predictive task: in this setting we propose a methodology

to improve the data selection, by taking into account the predictive task to perform: we call this method *contextualization*, as the data selection is specifically performed for a specific predictive context. We present another industrial collaboration with Airbus Helicopters, for which we applied our methodology on their data to predict the oil pressure in the main gear box of an helicopter.

5.2 Related work

This chapter is at the intersection of several important research problem, as we try to both evaluate the feasibility of classification from the training data, use such result to better explain the model to domain experts, and devise strategies to better identify the tuples of interest. We will now give an overview of the different domains related to these questions, to show how our propositions relate to other existing problems:

Classification feasibility One of the main contribution of this chapter is the proposition of an upper bound for the accuracy of classifier. As explained in section 5.3.2, this upper bound is limited to the available data, and does not take into account the data ditribution, contrary to Bayes error: this error is hard to compute i practice, and for this reason, several approaches have been proposed to approximate it: for example in [FH87], a k-nearest-neighbors approach is used to estimate this error. Alternatively, [Ant+99] how an upper bound of Bayes eror can be found. Bu appart from Bayes error that give a numerical evaluation of classification's feasibility, other solutions have looked at the factors that can improve or lower a classifier's performance: in this chapter, we showed that the functional dependency between the features and the class has an impact on the accuracy; in [KS13], the authors showed that if there is a functional dependency between features, it is likely to affect the classifer negatively. Looking at the features used for classification is of high importance, and is therefore an important research question: in this chapter, we mainly focused on adding/removing tuples, but it is also important to consider if more features should be added (as appeared in the study with Cemafroid), or if some should be removed. Therefore, many approaches have been devised to determine what are the features that allow to obtain the best classification performances, a problem called feature selection [Tan+14]. In [Ghi+15], the authors explain that the choice of the decriptors (attributes) is crucial, and propose a list of characteristic a good set of attributes should have. Usually, the

purpose of feature selection algortihms is to find the best subset of features, so that it maximizes the performance's measure [DL97]. As it is a combinatorial problem, it is time costly, and heuristics have been proposed to address this problem (see for example [LS+96; YL03; KR92]).

Model's interpretability In addition to giving an upper bound for the classification, we also used our methodology to help domain experts in better understanding their classification models: indeed, they can often appear as black boxes, and better explaining their decision process is an important research question. There is therefore a trade-off to find between achieving a satisfying accuracy, while being able to interpret the decisions made by such models. The problem of interpretability of predictive models has been addressed from different angles (see [Car+19] for an overview). First, there are two kinds of models: some are interpretable by nature, such as decision trees, while others require to develop methods to interpret their decision, for example neural networks that can seem like black boxes. It is also possible to focus on the data rather than on the features, as was done in this chapter: this mostly consist in exploring the dataset [Tuk77], similarly to the exploration techniques proposed for data selection in chapter 4. Most interpretation methods are algorithm specific, as they exploit the algorithms characteritics to extract interpretations: many focus on the interpretation of neural network, because they often show good accuracy [Mon+17; Kim+17; Sel+17].

Data dependencies and machine learning There has been several attempts to use key concepts from data dependencies to tackle various problems related to machine learning. In [AK+18], the authors propose to perform in-database learning, and use functional dependencies to tackle optimization problems. More generally, there is a raising interest for integrated key database theory concepts into machine learning, such as in [Sch+16] that builds least squares regression models over training datasets defined by arbitrary join queries on database table. It is also worth mentioning [Zou+06] where an entire machine learning library has been adapted so that it is compatible with a storage of data in a DBMS. It therefore appears that approaches combining database notions into the learning process are promising, as they allow to combine the data storing to the data processing.

Data dependencies and data cleaning The analysis of counterexamples, and the eventual removal of some of them, for example through contextualization, is a form of data cleaning, as it aims at removing the data that is not fitted for the considered task. Data cleaning is a crucial part in most of data science

application, as data scientist actually spend around 80% of their time on cleaning the data [Zha+03]. As a consequence, many research has be done on addressing this problem [RD00], many of them relying on data dependencies and constraints. For instance, [WL18] proposes a semantic data profiler that can compute samples that satisfy the same constraints than a given dataset. As the limited expressiveness of functional dependencies did not always adapt well to the need of data cleaning on real datasets, specific dependencies have been proposed to identify inconsistencies in a dataset, and eventually repair it. Conditional dependencies [Boh+07] are functional dependencies that hold only on a subset of the dataset. Matching dependencies [Fan08] for data repairing uses matching rules to relax the equality on functional dependencies and assign values for data repairing. In Holoclean [Rek+17], dependencies are used to clean automatically a dataset. In [Sa+19], a formal framework is proposed to bridge the gap between database theory and learnability theory, and is applied to three applications: data cleaning, probabilistic query answering, and learning. It can even be used to clean dataset in order to provide *fairness* [Sal+19]. [Chu+13] introduces denial constraints, allowing to declaratively specify logical formulae to exclude counterexamples. This work acknowledges the importance of counterexamples for data cleaning, in collaboration with domain experts. It can be noticed that there is a tight relationship between denial constraints and our counterexamples. Indeed, counterexamples of functional dependencies are no more than a special case of denial constraints, i.e. $\nexists t_1, t_2$ such that $t_1[X] = t_2[X]$ and $t_1[Y] \neq t_2[Y]$. Interestingly, we do not rely on expert users to specify logical statements for defining denial constraints, and thus counterexamples. Our proposition is fully automatic, and the counterexamples we provide at the end are complex and do not require any user input. The price to be paid is that we cannot express the induced denial constraints.

5.3 Estimating the model's performances from the data

5.3.1 Existence versus determination of a function

To evaluate the adequacy of a dataset with respect to a classification task, we rely on the mathematical notion of function:

Definition 9. *A function f is a mapping of each element x of a set X (the domain of the function), to a* unique *element y of another set Y (the codomain of the function).*

According to the core definition of a classification problem, a classifier is itself a function: for any input vector, it predicts a unique output value. As a results, classifiers rely on the strong assumption that there exists -or almost exists- a function from the attributes to the class in the dataset.

Functional dependencies also rely heavily on this notion of unique output. Indeed, a relation r satisfies a FD $X \rightarrow C$ if and only if all tuples that are equal on X are associated with the same unique value on C. As a result, $r \models X \rightarrow C$ if and only if there exists a function f from X to C, on the active domain of r.

If we combine these results, it appears that a classifier determines a function over a relation, whereas a satisfied functional dependency guaranties the existence of a partial function from the active domain of X to the active domain of Y. Therefore, when writing a classification problem using relational databases notations, it follows:

Property 7. $r \models X \rightarrow C \iff$ *there exists a function* f *from* $\bigcup_{A \in X} adom(A, r)$ *to* $adom(C, r)$

This problem is clearly easier than the associated classification problem: determining the function itself requires more investigation than proving its existence. However, it provides a simple solution that indicates the likelihood of finding a function following the data proposed to train the classifier. As a result, if the FD $X \rightarrow C$ is satisfied, then it means the relation is coherent with the classification task. If it is not, then it is necessary to study the counterexamples, to determine if their proportion is acceptable, allowing to still perform the classification task, or if the data selection should be refined to select more adequate data.

To summarize, our first proposition is, given a classification dataset, to verify the degree of satisfaction of the functional dependency $X \rightarrow C$.

5.3.2 Upper bound for accuracy of classifiers

Upper bound definition

If the satisfaction of the functional dependency $X \rightarrow C$ is interesting, the key element is the proportion of counterexamples. Based on this proportion, it is possible to estimate how far the DF is from being satisfied. Ideally, it would therefore be interesting to make a direct relationship between the proportion of counterexamples, and the performances of classifiers on the relation. Indeed, intuitively, the more counterexamples there is, the worst the performances should be.

In this chapter, we propose to give a bound, similar to Bayes error, but to make it more specific to a given relation, in order to be able to compute exactly. More specifically, we propose to estimate the error only taking into account the available data, by using the evaluation of the satisfaction of the FD $X \rightarrow C$ over the considered dataset. Additionally, rather than proposing it as en error, we propose it as an upper bound of the accuracy from which the error can very easily be derived, as the accuracy is equal to 1 minus the error).

As a result, if we only take into account the available data to train the classifier, measure G_3 can be used as an upper bound for the accuracy of a classifier on the considered data, for the FD $X \rightarrow C$. Indeed, G_3 represents the proportion of tuples in the dataset satisfying the considered functional dependency. In such a subset of the original data, there is therefore no counterexample. This means that in the subset s defined for G_3, there exists a function between the left and right hand side of the dependency. Theoretically, it is thus possible for a classifier to reach a perfect score if it identifies the correct underlying function, independently of its capabilities to generalize from it. On the opposite, the counterexamples to $X \rightarrow class$ are blocking points for any classification algorithms, as they introduce pairs of tuples for such that the classifier will misclassify at least one of them. As a result, we propose the following result:

Theorem 2. *Let $X \subseteq R$ be a set of features, $C \in R$ the class to be predicted, r a relation over R, and M a classifier from X to C. Then:*

$$accuracy(M, r) \leq G_3(X \rightarrow C, r)$$

Proof. Let s be a maximum subset of r such that $s \models X \rightarrow C$. For all $(t_i, t_j) \in s$, if $t_i[X] = t_j[X]$ then $t_i[C] = t_j[C]$.

Let $t_h \in r \setminus s$. Then there exists $t_i \in s$, such that $(t_i, t_j) \in CE(X \rightarrow C, r)$, otherwise s is not maximal. Even if only the tuples from $r \setminus s$ are misclassified, and all the tuples from s correctly classified, $accuracy(M, r) = \frac{|s|}{|r|} = G_3(X \rightarrow C, r)$.

If some tuples are misclassified due to to the algorithm itself, this can only lower the accuracy, and thus the result follows.

□

This result is simple but powerful, as it can be applied to any classification dataset, and offer guaranties on the feasibility of classification over it. Indeed, G_3 can then be seen as a first indicator for the user, to decide whether or not the considered data

is adequate for the classification task: if the upper bound is lower than the accuracy the domain is hoping for, then it is necessary to spend more time on data selection. For example, in a medical application, it is likely than only a very high accuracy is acceptable, because human's lives are at stake: therefore if G_3 is low, then there is no model than will be able to achieve an acceptable accuracy.

It can be argued that measure G_3 is much simpler than other existing measures such as Bayes error, because it only considers the data available to train the model. But this simplicity is what make it possible to compute it, and what makes it easier to understand for domain experts, because it allow to discuss on the data they actually have, rather than considering data that they can't even consider yet. We will now discuss how to compute G_3 in practice.

Upper bound computation

Now that G_3 has been established as an upper bound for any classifier on a considered classification dataset, it is necessary to explain how to concretely compute it. Contrary to G_1 and G_2 that can be quite easily computed by looking at each pair of tuples, G_3 requires to identify the tuples to be removed from the dataset so that is satisfies the dependency. In addition, the minimum possible number of tuples should be removed.

Following the definition of G_3, in order for s to be maximal, it should keep as many tuples as possible, while removing all the counterexamples of the given functional dependency. As a result, this can be done by grouping all the tuples that share the same left hand side, and then selecting among them the ones that share the same right hand side, and that are the majority. This allows to remove all the counterexamples, while removing the minimum number of counterexamples. The size of s is therefore the sum, for each different left hand side, of the size of the majority right hand side. Therefore, G_3 can be computed with the following proposition:

Property 8. *Let r be a relation over $\mathcal{R}$. Then:*

$$G_3(X \to Y, r) = \frac{\sum_{x_i} \max_{y_i} |\pi_{XY}(\sigma_{X=x_i \wedge Y=y_i}(r))|}{|r|}$$

where $x_i \in \pi_X(r)$ and $y_i \in \pi_Y(\sigma_{X=x_i}(r))$.

Note that $X = x_i$ is a simplification for $A_1 = v_1 \wedge \ldots \wedge A_n = v_n$ for $X = \langle A_1..A_n \rangle$ and $x_i = \langle v_1..v_n \rangle$.

Algorithm 4: G_3 computation algorithm

```
1  procedure ComputeG3 (r);
   Input  : r the classification dataset,
            A_1 ... A_n → C a functional dependency
   Output : G_3(A_1 ... A_n → C, r)
2  map = {}
3  for row ∈ r do
4      if row[A_1..A_n] ∈ map then
5          if row[class] ∈ map[row[A_1..A_n]] then
6              map[row[A_1..A_n]][row[class]] += 1
7          else
8              map[row[A_1..A_n]][row[class]] = 1
9          end
10     else
11         map[row[A_1..A_n]] = {}
12         map[row[A_1..A_n]][row[class]] = 1
13     end
14 end
15 maxsum = 0
16 foreach key ∈ map do
17     maxfrequent = max(map[key])
18     maxsum += maxfrequent
19 end
20 return maxsum / |r|
```

To compute this measure, we propose algorithm 4. It relies on a specific data structure, presented on figure 5.1, with tuples from table 5.1 as an example. It is a hash map, with the values over the attributes as key, and another hash map as value. For the second map, the key is the class, and the value the number of times this class appears (for these given attributes). The construction of this map is explained from line 3 to line 14 of algorithm 4: for each row in the dataset, the corresponding values in the map are filled or created when necessary. Once this data structure is complete, the algorithm looks at each key in the map: it will then retrieve the number of occurrences for the class that has the highest value in the second map. All these maximum values are added to one another, as they correspond to the maximum number of tuples that can be kept, among the ones that share the same attributes value, in order to satisfy the functional dependency. This process is explained though line 15 to 19 in algorithm 4.

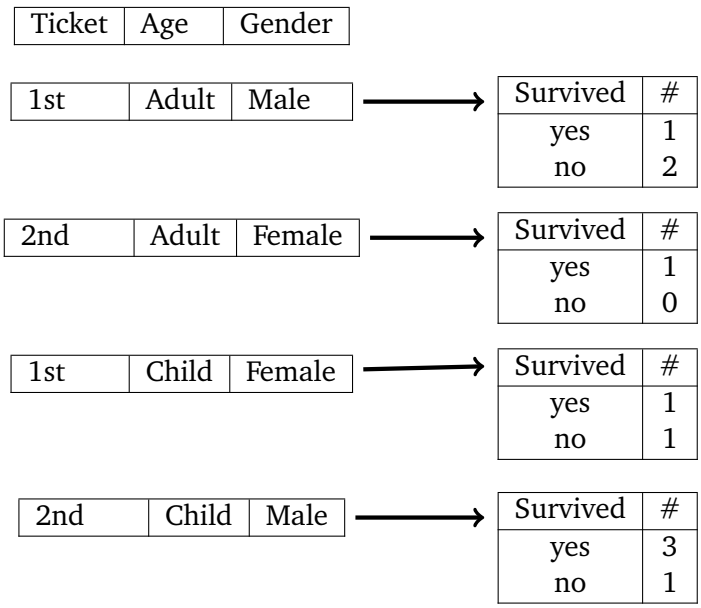


Fig. 5.1: Data structure for G_3 computation

Example 13. *Using figure 5.1, measure G_3 can be computed as follows for the Titanic dataset:*

$$G_3(Ticket, Age, Gender \rightarrow Survived, Titanic) = \frac{2+1+1+3}{10} = \frac{7}{10} = 70\%$$

5.4 Generation of *difficult* synthetic classification datasets

From the previous results, it appears that the counterexamples directly control part of the accuracy results of classifiers. It is therefore possible to influence the *difficulty* of classification datasets, by modifying their proportion of counterexamples. We therefore decided to generate datasets with a controlled value of G_3, and to evaluate the accuracy of several classifiers over such synthetic classification datasets. Moreover, the datasets were generated in the worst possible scenario for classification, by introducing more and more counterexamples, and see the impact of the augmentation of counterexamples on the classifiers performances: the generated datasets therefore have a low G_3 value, that decreases with the number of counterexamples. We call such datasets *difficult*, in the sense that they are designed to be as hard as possible for a given classifier. We first explain how the synthetic

id	A_1	A_2	...	A_{n-1}	A_n	Class	Scaling Factor
1	13	9	...	21	16	1	
2	58	13	...	18	5	2	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
...	...	...	...	...	...	k	1
...	...	...	...	...	...	1	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
N	35	9	...	21	11	4	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$(j-1)\times N+1$	13	9	...	21	16	$(t_{((j-1)\times N+1)-N}[Class]+1)\%k$	
...	...	...	...	...	...	...	
i	$t_{i\%N}[A_1]$	$t_{i\%N}[A_2]$	...	$t_{i\%N}[A_{n-1}]$	$t_{i\%N}[A_n]$	$(t_{i-N}[Class]+1)\%k$	j
...	...	...	...	...	...	...	
$j\times N$	35	9	...	21	11	$(t_{(j\times N)-N}[Class]+1)\%k$	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$(sf-1)\times N+1$	13	9	...	21	16	$(t_{((sf-1)\times N+1)-N}[Class]+1)\%k$	
...	...	...	...	...	...	...	sf
$sf\times N$	35	9	...	21	11	$(t_{(sf\times N)-N}[Class]+1)\%k$	

Tab. 5.2: Generation of a difficult dataset

data was generated, and then detail the result of the evaluation of classifiers over such data.

5.4.1 Synthetic data generation

The idea to generate datasets with a very low G_3 measure, such that any classifier will not be able to perform efficiently on it, and to push the scenario to the limits, in the worst possible case. Generating such datasets can then be used to test new classifier, or to improve existing one so that they get as close as possible to the theoretical maximum accuracy (i.e. they only misclassify counterexamples).

To generate such datasets, it is necessary to create counterexamples, and to play with their number to increase G_3 error and lower the maximum accuracy. To get very difficult classification datasets, we propose to start with an initial classification relation that does not have any identical tuples, and to duplicate them, by associating each new tuple to a different class at each duplication. This will naturally introduce counterexamples, and their number will increase with the number of duplication.

Algorithm 5: Difficult dataset generation algorithm

```
1  procedure GenerateDifficult (n, N, k, sf);
   Input  : n the number of attributes, N the number of tuples before
            duplication, k the number of classes, and sf the scaling factor
   Output : d a difficult dataset for classification
2  class = 1
3  relation = [N][n+1]
4  for i ∈ 1..N do
5  |  for j ∈ 1..n do
6  |  |  relation[i][j] = random([0:ln(N)]
7  |  end
8  |  relation[i][n+1] = class
9  |  if class == k then
10 |  |  class = 1
11 |  end
12 |  class + = 1
13 end
14 if A_1..A_n is not a key ;                    // Check all rows are unique
15  then
16 |  go to line 4
17 end
18 dataset = relation
19 copy = relation
20 for i ∈ 2..sf do
21 |  copy = Duplicate(copy, k);
22 |  dataset = dataset ⋃ copy;
23 end
24 return dataset

25 Function Duplicate(r, k):
26 |  duplicate = []
27 |  for row ∈ r do
28 |  |  new[A_1...A_n] = row[A_1...A_n]
29 |  |  new[class] = (row[class] + 1)%k
30 |  |  duplicate += new
31 |  end
32 |  return duplicate
```

The generation strategy is illustrated in table 5.2. It requires the size N of the initial relation, the number n of attributes of the relation, the number k of classes, and the scaling factor sf (the number of duplication of the initial relation) used to produce counterexamples. The strategy works as follows: let r be a relation over $R = A_1 \dots A_n$. Then:

1. Insert N unique tuples t_i for $i \in 1..N$. For each tuple $t_i, i \in 1..N$, add a value for the $class$ attribute as follows: $t_i[Class] = i\%k$. This corresponds to the rows 1 to N in table 5.2.

2. The duplication process is repeated $sf - 1$ times as follows:
   Let j be the current duplication, $2 \leq j \leq sf$. The initial relation is duplicated, generating N new tuples, numbered $t_{(j-1)\times N+1}$ to $t_{j\times N}$. For each duplicated tuple $t_i, (j-1)\times N+1 \leq i \leq j\times N$, the values do not change over attributes A_1 to A_n, i.e. $t_i[A_1...A_n] = t_{i\%N}[A_1...A_n]$. However, the class value is shifted by one with respect to the previous duplicate, i.e. $t_i[Class] = (t_{i-N}[Class]+1)\%k$

Algorithm 5 give the details of the generation process. Let us mention a few important point not detailed here.

- First, the domain of attributes is to be defined. In table 5.2, we use integers for the sake of clarity, but any other type of attribute would work exactly the same. However the data types will have an impact on the classifiers, as for example non-numerical values would require some pre-processing to be used with most classifiers.

- The only limitation on the attribute domain is to have enough values to generate unique rows, at least $\frac{ln(N)}{ln(n)}$ values[1]. Using a really high number of different values will only increase the difficulty for a classifier, as there will be very little redundancy between the values. This is a parameter than can be used to tune the difficulty of the classification dataset. In Algorithm 2 and the experiments, we used $\ln(N)$ different values (see line 6 in algorithm 5.

- In addition, whenever $sf > k$, the dataset contains duplicates that share the same class values, as all values for the class have already been used for duplicates. This introduces redundancy in the data, but does not remove any counterexample. Given the parameters of algorithm 5, it is possible to compute the exact value of G_3 for the produced dataset:

[1] Given $|dom|$ different value, there exists $n^{|dom|}$ different vectors of size n. Therefore it is necessary that $N < n^{|dom|}$ and thus $|dom| > \frac{ln(N)}{ln(n)}$

Proposition 1. *Let r_{hard} be a relation generated using algorithm 5. Then:*

$$G_3(A_1..A_n \rightarrow Class, r_{hard}) = \frac{1 + \frac{sf - sf\%k}{k}}{sf}$$

Proof. Let $i \in [1..sf]$ denote the i-th duplication of the initial relation. While $i \leq k$, it only introduces counterexamples. Therefore, for each duplicated tuple, there are i different classes, for each of the N original tuples. As a consequence, if $sf < k$, $G_3 = \frac{N}{N*sf} = \frac{1}{sf}$.

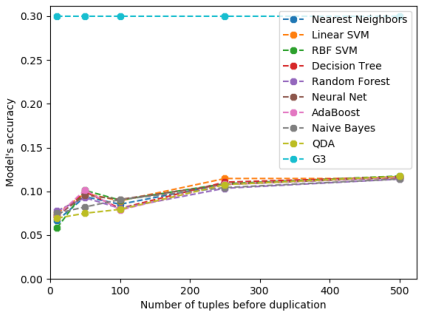
For $i > k$, there is redundancy for each duplicate: the duplicated tuples agree with the ones already produced. Therefore, there are as many agreeing tuples as the number of times $i\%k = 1$. The size of the set of agreeing tuples therefore depends of how many times an initial tuple is associated with the same initial class during the sf duplications, which is exactly the quotient of the euclidean division of sf by k, i.e $\frac{N + N*\frac{sf - sf\%k}{k}}{sf*N}$. And the result follows. □

5.4.2 Experimentations

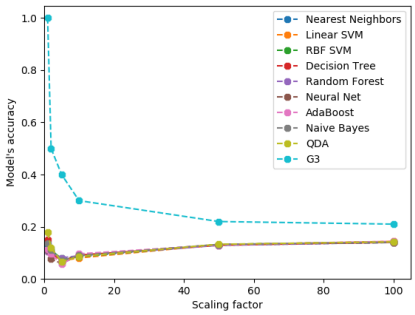
We implemented the generation algorithm, in order to measure the influence of its different parameters. To do so, we generated datasets with different parameters, and used ten different classifiers from the scikit-learn library [Ped+11] to estimate their accuracy over the datasets. All results presented below are averaged over 10 different instances randomly generated using algorithm 5. The computing being below one second, they are not discussed in this paper.

First, the influence of the number of tuples in the original relation (before duplication), was tested. The results are shown on figure 5.2a. It is worth noticing that the accuracy is in any case really low as expected: the maximum accuracy reached is 12%. However, as the initial number of tuples increases, so does the accuracy. Indeed, adding more tuples introduces some redundancy among the values for each attribute, allowing the classifier to find some sort of generalization for some cases. But the number of counterexamples is way too high to reach good classification measure, and $G3$, which is constant as the number of tuples does not influence it, is also low: it can be seen that the model is far from reaching it.

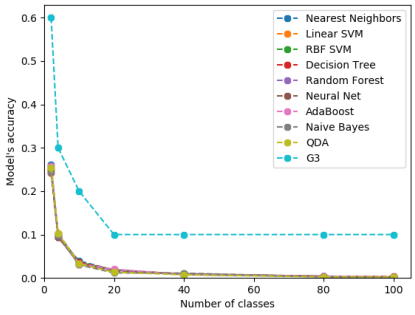
Then, the influence of the scaling factor sf was tested, and results are presented on figure 5.2b. As it influence G_3, this measure slowly decrease with the scaling factor, as more and more counterexamples are introduced. For this parameter, the accuracy



(a) Evolution of classifiers accuracy against the number of tuples the dataset, with respect to G_3 ($k = 5$, $sf = 10$)



(b) Evolution of classifiers accuracy against the scaling factor of the dataset, with respect to G_3 ($N = 100$, $k = 5$)



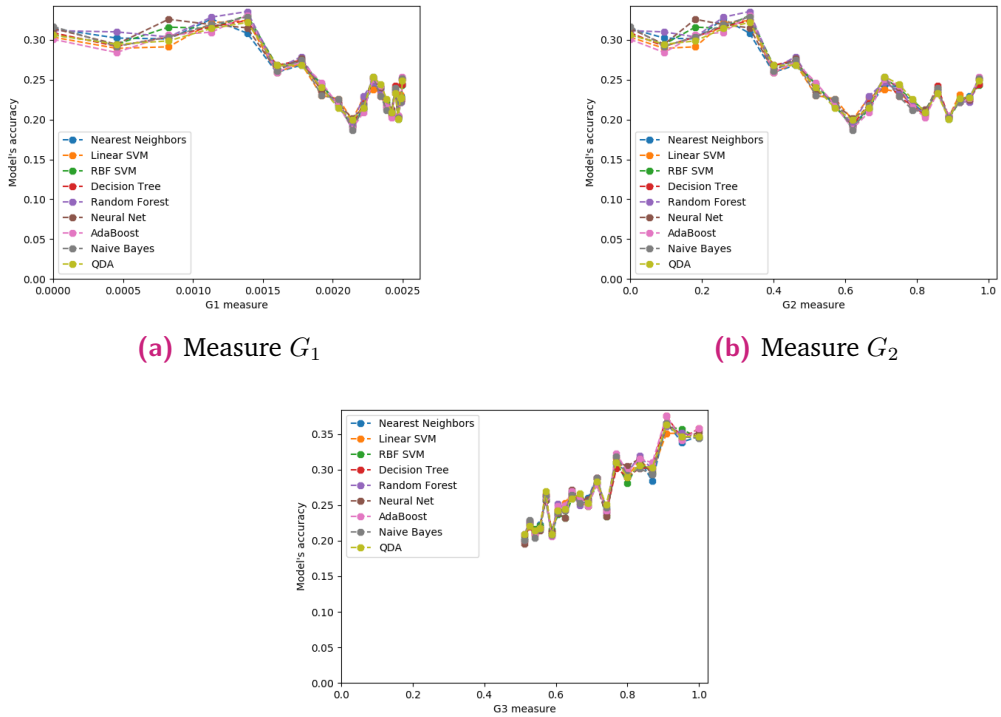
(c) Evolution of classifiers accuracy against the number of classes in the dataset, with respect to G_3 ($N = 100$, $sf = 10$)

Fig. 5.2: Classifiers accuracy given the parameters for generating difficult classification datasets, compared to G_3

first drops, before slowly increasing with the scaling ratio. This increase starts as soon as $sf > k$, as explained previously, because there is then some redundancy allowing the classifier to make correct predictions for some tuples.

In addition, the influence of the number of classes is exposed on figure 5.2c. Once again, G_3 decreases with the number of classes, as their are more counterexamples. As expected, the accuracy only drops with the number of classes, as the classifier has then fewer examples for each class, and therefore fewer possibilities to find patterns and generalize.

Finally, the accuracy of the classifiers was evaluated with respect to the error measures G_1, G_2 and G_3, for the functional dependency $\{features\} \rightarrow class$. The results are shown on figure 5.3. For measures G_1 and G_2 the accuracy drops as the error increases, as the number of counterexamples also increases. On the opposite,



(a) Measure G_1

(b) Measure G_2

(c) Measure G_3

Fig. 5.3: Evolution of classifiers accuracy given the FD error measure of the dataset

when G_3 increases, so does the accuracy, as it means that the set of tuples that would satisfy the functional dependency is getting bigger, allowing the classifier to reach a higher accuracy.

To summarize, these experimentations have confirmed in practice the impact of counterexamples on the accuracy of classifiers, as they prevent the functional dependency between the features and the class to be satisfied. It is now necessary to test this approach on real datasets, in order to compare classifier's accuracy in practical situations, to our theoretical upper bound estimated using G_3.

5.5 Taking attributes domains into account

Up until this point, we mainly considered intergers values for the feature's domain: they are very practical, because they fit well with crisp functional dependencies, and allow to use the strict equality for the comparison of values. However, in real life, there are many different possible domains. We now discuss the limitations of crisp FDs and G_3 depending on the values domain.

Dataset	# tuples	# classes	Average Accuracy (%)	Max Accuracy	G_3 (%)	$G_3 - max$
Titanic	2201	3	76.6	79.1	79.1	0.0
Breast Cancer	286	9	71.2	76.2	97.8	21.6
Abalone	4177	5	60.1	67,4	100	32.6
Adult	48842	13	81.8	86,2	99.9	13.7
Bank	4521	16	88.4	90,5	100	9.5
Car	1728	6	86	99,2	100	0.8
Contrac	1473	9	49.6	57,2	95,5	38.3
Ecoli	336	7	77.6	90,9	100	9.1
Iris	150	4	89.4	99,3	100	0.7
Led-display	1000	7	60.3	74,8	76	1.2
Lenses	24	4	74	95,8	100	4.2
wine-quality-red	1599	11	55.6	69	100	31
yeast	1484	8	52.5	63,7	100	36.3
zoo	101	16	86.5	99.0	100	1.0

Tab. 5.3: Comparison of accuracy and G_3 measure over classification datasets

5.5.1 Limitations of crisp FDs

Experimental observations

After the initial experimentations on synthetic data, it appeared necesssary to test the approach on real data. To this end, we ran experiments on well-known classification datasets, to compare the state-of-the-art accuracy results for these datasets with their G_3 measure. The datasets and the accuracy measures come from [FD+14], a thorough study on the accuracy of 179 classification algorithms over 121 datasets.

The results are presented in table 5.3. As expected, for all the datasets, the maximum accuracy found by [FD+14] is always below our measured G_3 value: the difference between the two values is indicated in the last column of the table, showing some significant differences for some datasets. For some datasets, the G_3 measure is 100%, which is reassuring, as it means that their exists a function between the attributes and the class, and that it does actually make sense to perform classification. It is also interesting that despite the existence of a function, the average accuracy is still very low sometimes, such as for the Contract dataset. Finally, the Titanic and the Led display are very interesting datasets, as their G_3 measure is pretty low compared to the other ones. Therefore it is worth questioning the interest of performing classification on these datasets. For the Titanic dataset, the maximum accuracy is strictly equal to G_3, meaning their exists a *perfect* classifier on this dataset, that only failed on tuples from counterexamples.

X	Y	$X \to Y$
0	0	1
0	1	1
1	0	0
1	1	1

Tab. 5.4: Possible outcomes for the comparability of two values for the satisfaction of a functional dependency

These results are very interesting, as they confirm how G_3 can be used to indicate to domain experts wether or not it is worth performing classification, or if there are simply too many counterexamples for the results to be interesting. In addition, once a model has been trained, it is possible to evaluate the quality of classification by comparing the obtained accuracy to the estimated upper bound.

However, these results also highlights some limitations for G_3: indeed, for several datasets, it appears that the upper bound is 100%, even though for many of them, the average accuracy reached is far from perfect. One of the main reason is that, for the computation of G_3, we make use of the satisfaction of a crisp functional dependency: therefore, the counterexamples retrieval is based on the strict equality, meaning two tuples form a counterexamples only if all of the values over the features are strictly equal, but differ on the class attribute. As a result, depending on a feature's domain, it can be more or less likely for two values to be equal: for continuous values, because there is an infinity of possible values, there might not be two tuples with the same value. This directly relates to how values our compared, and how it impacts the satisfaction of a functional dependency. The different possibilities are presented in table 5.4:when comparing two tuples, they are two possibilities for each side of the FD, meaning four possibilities in total. Among these four, only one prevents the FD from being satisfied, which is when the left-hand side is equal, but not the right-hand one. Additionally, because of the infinite number of possible values, when the domain of an attribute is continuous, it is less likely to be equal. As a result, with continuous domains, there are less counterexamples. This means that in this situation, it is necessary to adapt the evaluation of FD satisfaction for example using non-crisp FDs, to better take into account the data. Indeed, there are often field-specific knowledge that can be taken into account, to consider two values as *similar*. It is therefore also necessary to modify how G_3 is evaluated and analyzed.

Example 14. *On figure 5.5, the table presents data from a meteorological problem: given the temperature, pressure and humidity of a place, will it be raining the next hour? Based on this initial table, by considering crisp functional dependencies, there are*

id	Temperature	Pressure	Humidity	Rain
t_1	27.2	1004.5	98.7	yes
t_2	26.5	1018.4	42.5	no
t_3	15.7	1008.6	78.9	yes
t_4	16.1	1016.9	76.7	no
t_5	25.9	1017.5	43.8	yes
t_6	28.1	1021.7	41.7	no
t_7	4.1	1007.2	74.3	yes
t_8	15.9	1022.3	79.1	no
t_9	27.3	1019.8	39.5	no
t_{10}	3.8	1006.7	73.4	yes

Tab. 5.5: Toy example from the problem of continuous values

no counterexamples, and therefore $G_3 = 100\%$. However, for meteorologists, there are counterexamples, especially if the following measurement errors are taken into account:

- *Temperature measurement uncertainty is $\pm 0.5°C$*
- *Pressure measurement uncertainty is $\pm 1hPa$*
- *Humidity measurement uncertainty is $\pm 2\%$*

Based on this, for example, tuples t_7 and t_{10} should form a counterexamples, but are missed because only the strict equality is considered for now.

As a result, it appears necessary to be able to take into account these continuous values, and to propose solutions to be able to take domain constraints into account when evaluating the satisfaction of the functional dependency.

Solutions overview

In order to deal with the problem of the continuous values for which the strict equality shows some limits, it is possible to consider two different directions:

- Adapt the functional dependency evaluation: as part of the problem comes from the satisfaction of the FD, it is possible to modify how the values are compared, for example by relaxing some of its constraints. This mean using non-crisp functional dependencies, and for example use G_3^s instead of G_3.
- Modify the data: as the other part of the problem comes from the domain of the features (i.e continuous values), it is possible to modify the data to make it

adequate with the evaluation of traditional FDs, for example by discretizing the data.

Both methods require to use a similarity measure, either to group the data for the discretization, or to relax the FD's satisfaction. Such similarity measures have to be reflexive and symmetric. However, for discretization, it also has to be transitive, in order to be able to form the bins required by such technique. This additional constraint is the main difference between the two approaches, and explains why G_3 applies and can be computed when the data is discrete, but has to be adapted in the general case.

Indeed, when the data is not discrete, the computation of G_3 is an NP-complete problem [Car+20]. Indeed, this problem is equivalent to the minimum vertex cover (see [Son10] for the proof), a well-known problem in graphs. G_3 computation can indeed be modelized as a graph problem: let a graph G be a pair $(V(G), E(G))$. The elements in $V(G)$ are the vertices of the graph, and E(G) its edges, between two vertices in $V(G)$. The size of the graph is $|G| = |V(G)$. From this, given a relation r and a FD, we propose to build the counterexample graph as follows:

Definition 10. *The counterexample graph G_c for an FD $r \models X \rightarrow Y$ is a graph for which:*

- $V(G_c) = t \in r$
- *for $(t_1, t_2) \in r \times r, (t_1, t_2) \in E(G_c)$ iff for all $A \in X, t_1[A] = t_2[A]$ and $\exists B \in Y, t_1[B] \neq t_2[B]$*

From this graph, computing G_3 is equivalent to removing the minimal number of vertices such that there are no edges in the graph: indeed, this means removing the minimum number of tuples, such that there are no counterexamples.

Property 9.

$$G_3(X \rightarrow Y, r) = 1 - \frac{max\{|G'| : E(G') \subseteq E(G_c), |V(G')| = 0, \nexists (u, v) \in E(G') \times E(G') | (u, v) \in V(G_c)\}}{|r|}$$

Example 15. *Figure 5.4 presents such a graph for the toy Titanic dataset. The degree of t_5 (resp. t_8) is 3 (resp. 2). Clearly t_5 and t_8 cause more counterexamples than the others. By changing the class value for these two tuples, the number of counterexamples drops to 1 (instead of 6).*

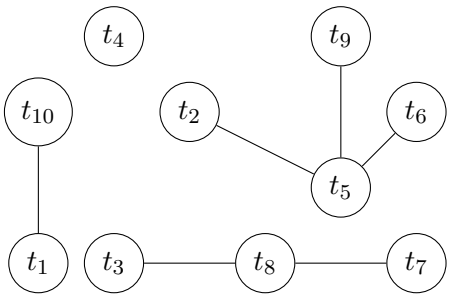


Fig. 5.4: Couterexamples interaction graph for the Titanic dataset

As a result, computing G_3 is NP-complete. When the data is discrete, the values comparison is transitive, and it is then possible to use algorithm 4 to compute it. But it the general case, for example with G_3^s, only heuristics can be designed.

As a result, we compare the two proposed approaches, one relying on transitivity and allowing to compute G_3, and the other non-transitive.

5.5.2 Similarity-based solution

The first possibility is to compare each value of a given domain using a similarity function. This is equivalent to relaxing the equality that is used for crisp FDs. Because FDs have been used for many different and important problems (data cleaning, database's schema, query relaxing, etc), there has already been the need to relax their constraints to better adapt to the specificities of each application. All these alternative are grouped under the term *relaxed functional dependencies* (see [Car+15] for an overview). Among all the proposed alternatives, there are two main families of approaches:

- The first group of solutions relax the FD by allowing it to be *almost* satisfied: this means that even if a few tuples form counterexamples, the FD can hold for almost all the data. This is especially useful to allow the data to contain some missing values, measurement errors, or outliers. Approximate functional dependencies [Huh+99] are an example of such relaxed FDs. These approaches are very related to measure G_3, as it quantifies how far is the FD from being satisfied: therefore, when a domain experts decides whether or not the considered data is adequate for classification based on G_3, she basically decides if the number of counterexamples is small enough to say that the FD almost holds. Therefore, our evaluation criteria is a form of approximate FD. However, this does not solve the problem of continuous values that prevents some counterexamples from being identified.

- The second type of solutions aims at modifying how the values are compared, to propose alternatives to the strict equality, in order to consider to values as equal, based on a given criteria. Fuzzy functional dependencies [Bos+98; RM88] are an example of such an approach, by introducing a *fuzzy* resemblance measure to compare two values of a given domain. Other approaches propose a comparison based on an order relation, such as ordered functional dependencies [Ng99; Ng01]. These approaches can be very interesting to deal with the problem of continuous values, because they allow the relax the strict equality, and to propose adequate values comparison that can be defined in accordance with the application domain.

From all these possibilities, it therefore appears that relaxing FDs is an very interesting possibility, if used to allow a new form of value comparison on the domain values. The center of this solution relies on the existence of the similarity functions that are necessary to compare the values of a given domain. These similarities can be given by domain experts, if they have specific knowledge. It should also be noted that because such similarities require to compare each pair of tuple using the similarity function, it is most costly to evaluate the FD's satisfaction, as there is an exponential complexity ($\mathcal{O}(|r|^2)$). It should also be noted that in these case, G_3^s cannot be easily computed: however it still possible to evaluate G_1^s and G_2^s, and to retrieve the associated counterexamples.

5.5.3 Transitive similarity function and discretization

Data discretization

Alternatively, it is possible to use measures that are also transitive, meaning that all the values that are *similar* can be grouped together. In this case, it is possible to discretize the data in bins of similar values, and to obtain a discrete dataset on which it is possible to compute G_3.

Once again, the biggest question is what similarity measure to use, in order to perform a discretization that makes sense with respect to domain constraints. This approach allows to keep exactly the same approach for the FD evaluation, and to only transform the data once and for all. Additionally, because the number of different values after discretization is much smaller, this mean that the number of different tuples to compare can be reduced.

id	Temp.	Pres.	Hum.	Rain
t_1	27.2	1004.5	98.7	yes
t_2	26.5	1018.4	42.5	no
t_3	15.7	1008.6	78.9	yes
t_4	16.1	1016.9	76.7	no
t_5	25.9	1017.5	43.8	yes
t_6	28.1	1021.7	41.7	no
t_7	4.1	1007.2	74.3	yes
t_8	15.9	1022.3	79.1	no
t_9	27.3	1019.8	39.5	no
t_{10}	3.8	1006.7	73.4	yes

(a) Original data

→

id	Temp.	Pres.	Hum.	Rain
t_1	1	1	4	yes
t_2	1	2	1	no
t_3	3	1	2	yes
t_4	2	2	2	no
t_5	1	2	1	yes
t_6	1	2	1	no
t_7	3	1	2	yes
t_8	2	2	2	no
t_9	1	2	1	no
t_{10}	3	1	2	yes

(b) Discretized data

→

id	Temp.	Pres.	Hum.	Rain
t'_1	1	1	4	yes
t'_2	1	2	1	no
t'_3	2	2	2	no
t'_4	1	2	1	yes
t'_5	3	1	2	yes

(c) Reduced data

Fig. 5.5: Data reduction process

As a consequence, we propose a discretization solution that first groups similar values together, so that they are all assigned to a unique same value: this allows to define similarity. Second, as the number of different values is likely to be much smaller, it is possible to "reduce" the dataset by only keeping unique rows. The detailed process is illustrated on figure 5.5, following example 14. It works as follows:

- First, each attribute of the original data is discretized, to group similar values. It should be noted that this can be adapted and fine-tuned to each application, and that specific similarity measures can be defined in this step if necessary.
- Once an attribute is discretized, each of its value is replaced by the value representing the interval it belongs to. As functional dependencies do not care about the order between values, the value itself does not really matter, as long as all values in the same interval are replaced by the same new value.
- Once the data is discretized, as the number of different values for each attribute is equal to the number of clusters, there might be identical rows in the dataset. It is therefore possible to dramatically reduce the size of the original data, by only keeping unique rows, and adding an additional attribute to memorize how many times this row appears in the clustered data. This step allows to reduce the number of necessary comparisons to evaluate the satisfaction of the FD $X \rightarrow C$: if n is the number of tuples in the relation, $\frac{n*(n-1)}{2}$ comparisons are necessary: the smaller the n, the faster the evaluation.

To perform this process, we propose algorithm 6: the clustering process is described from line 3 to 9, and the data reduction is performed on line 12.

Algorithm 6: Discretization and reduction algorithm

```
1  procedure Reduce (r) over attributes A_1..A_n;
   Input  : r the classification dataset
   Output : A discretized and reduced dataset of integers
2  d = []
3  for A ∈ {A_1..A_n} do
4      if A is continuous then
5          intervals = discretize(r[A])
6          d[A] = intervals
7      else
8          d[A] = r[A]
9      end
10 end
11 d[class] = r[class]
12 r_reduced =Select A_1 ... A_n, C, count(*) as '#' From d Group By A_1 ... A_n, C
13 return r_reduced
```

Example 16. *In the reduced dataset given in table 5.5c, let us consider the counterexample (t'_2, t'_4) of the functional dependency $Temp, Pres, Hum \rightarrow Rain$. Using table 5.6b, it concerns in fact 4 tuples: t_2, t_6, t_9 and t_5 (cf previous example). It is then possible to go back to the original data in table 5.6a to see what does that mean on the real values. For example, values $\langle 25.9, 1017.5, 43.8\rangle$ of t_5 and values $\langle 28.1, 1021.7, 41.7\rangle$ of t_6 are considered similar and then, both tuples form a counterexample.*

In this data-centered process, it is still necessary to be able to define a strategy for the data discretization: however, it is possible to use existing techniques to propose the best discretization, for example clustering algorithms.

Impact of discretization on dataset size

The main advantage of the discretization is the reduction of the number of tuples, that allow to perform less comparisons to identify the counterexamples. In order to validate this, we performed experimentations to validate this data reduction, and to validate the impact on the time necessary to retrieve the counterexamples.

We first looked at how the data is reduced. We took the 4 biggest datasets from table 5.3, and computed their reduction ratio, when applying our algorithm. defined as follows:

$$ratio = \frac{|r| - |r_{reduced}|}{|r|}$$

Dataset	# tuples before	#tuples after	reduction ratio
Titanic	2201	24	98.9%
Abalone	4177	242	94.2%
Adult	48842	10176	79.2%
Bank	4521	4031	10.8%

Tab. 5.6: Reduction ratio for some datasets from table 5.3

Where $r_{reduced}$ is the reduced dataset.

To perform the discretization, we used the k-means clustering algorithm, coupled with the the silhouette coefficient [Rou87] to determine the most appropriate number of clusters for each attribute. The results are presented in table 5.6. It shows that the ratio differs from one dataset to another, with some very significant drops for some datasets such as Titanic, indicating their must be many redundant values. We then evaluated how well the data reduction technique improves the counterexamples retrieval time, to see how the approach would scale on large datasets. We evaluated it on astrophysical data from the Large Synoptic Survey Telescope[2] containing 500000 tuples over 25 attributes. For different sizes of datasets, we compute:

- The reducing ratio
- The computation time for counterexamples retrieval on the original data.
- The computation time for counterexamples retrieval on the reduced data.

The results of these experimentations are presented on figure 5.6. The first observation is that on the original data, the computing time is quickly too long even on relatively small instances. On the opposite, on the reduced data, it increases very slowly with the number of tuples, allowing for a reasonable time for retrieving the counterexamples. This is tightly linked with the reducing ratio, that increases significantly with the number of tuples: the more tuples there are, the more the original data is reduced with respect to its original size.

On datasets on which our approach would not be sufficient to scale, it is always possible to apply blocking [Bil+06] or parallelization [Chu+16], but such solutions have not been considered in the context of this thesis.

[2] `https://www.lsst.org/`

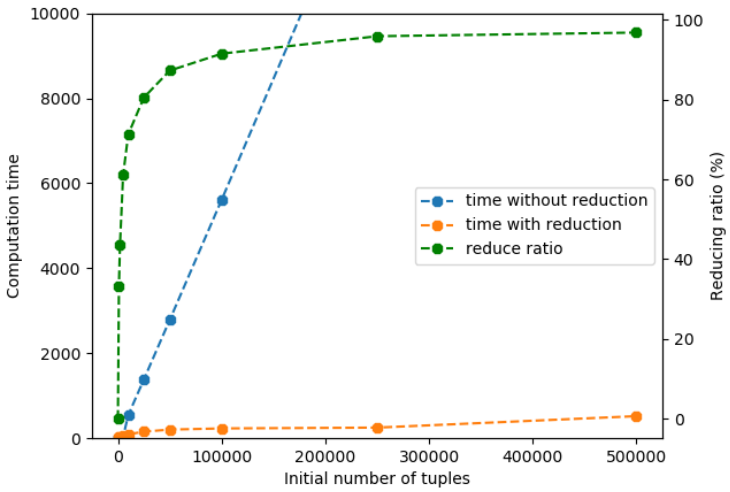


Fig. 5.6: Validation of G_3 computing time, on both original and reduced data, in parallel with the reducing ratio

5.5.4 Choosing the best solution

We have presented two different approaches, one modifying the evaluation of the FD, the other modifying the data. The question is therefore to decide which one to chose. There is of course no simple answer, as this decision should mainly be guided by the domain application, and the size of the dataset. Indeed, as the tuples comparison is the main bottleneck of the evaluation of the FD, the data discretization is an interesting solution to reduce the number of tuples, and to therefore be able to evaluate the satisfaction of the FD in a reasonable amount of time. Additionally, discretizing the data is a solution to take into account regression dataset, by discretizing the values to predict, the problem being then equivalent to a classification one. On the opposite, relaxing the FD satisfaction is an interesting solution if similarity measures have been defined that do not allow discretization, for example because they have to be evaluated on the fly during the tuples comparison.

Finally, if such solutions are interesting because they allow to get more counterexamples, that make more sense with respect to the application domain, the counterpart is that the link between G_3 and the accuracy's upper bound is lost: indeed, the proof from algorithm 2 does not hold, because it is not possible to predict the behavior of any classifier when the features are not strictly equal. Therefore, the data and the FD evaluation should not be changed if the domain experts really wants an upper bound on the initial data. Alternatively, the upper bound can be estimated on the original data, and the counterexamples retrieved on the transformed data: indeed, G_3 in

itself is usually insufficient, and the counterexamples are extremely informative for domain experts. Such counterexamples can be used to better understand G_3, and to better decide the strength and weaknesses of the dataset for the considered classification task. Ultimately, if data selection has to be refined, the counterexamples are the key to understanding what is wrong with the current data, and therefore what should be modified: such problems are discussed in the following sections.

5.6 Understanding the model's limitations

5.6.1 Importance of counterexamples

As mentioned previously, there are several situations for which measure G_3 is not sufficient to assess the adequacy of a dataset for a classification task:

- If measure G_3 is too low, then it is necessary to understand why there are so many counterexamples, in order to perform a better data selection if possible.
- If the link between G_3 and the accuracy is lost because the data had to be modified to compute a more coherent G_3, then it is necessary to look at the counterexamples, in order to check if they really are counterexamples with respect to the domain constraint, so that an informed choice can be made.

But even outside these specific situations, analyzing the counterexamples is an important exercises, with many potential benefits for the classification process. The counterexamples are very important to understand the value of G_3, to see the data that causes conflicts and lowers the accuracy of the classifier. Thus, by looking at them, a domain expert can seek an explanation to their presence in the dataset, and eventually remove them to improve the classification results. The counterexamples are a powerful notion to avoid the domain expert from being overwhelmed by the data when exploring a dataset, as she then only have a small but meaningful subset of tuples to study.

As a result, the counterexamples are a perfect starting point for a discussion between data scientist and domain expert: the first can use the counterexamples to better explain the limitations of the predictive model on the consider data ; the other can explore more easily the dataset by only accessing the data that causes conflicts, and can therefore better point out relevant information on the data, that can be used to refine the data selection, or to better understand the model.

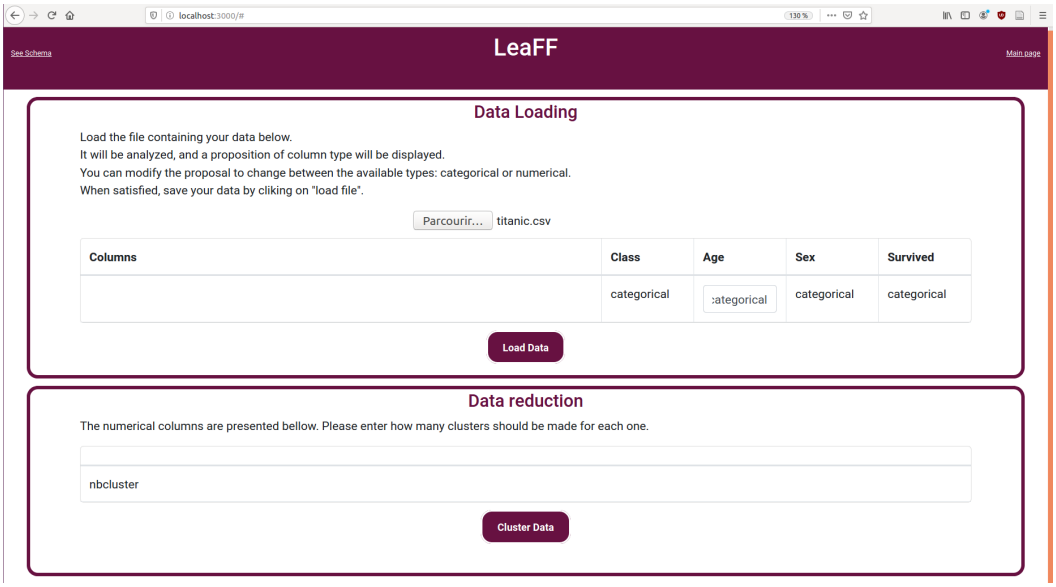


Fig. 5.7: LeaFF page for the configuration of the classification dataset

In this context, even if no additional data selection is performed, it appears necessary to look at the counterexamples, to gain a better understanding about the predictive model. The counterexamples help domain experts to better understand the limitations of such models, because they will be aware of the contradictions it has been trained on. This is beneficial, as some of the future misclassifications of the model might be explained by the counterexamples the model has been trained on: as a result, this removes part of the "black-box" aspect of the model, by showing that some of its mistakes simply come from the data it has been trained on. Additionally, being aware of the limitations of the considered model can help to perform better data selection for the training of other future models.

5.6.2 Leaff: a system for counterexamples exploration

Based on these considerations, and the importance for domain expert to explore their counterexamples, we developed a web application that can be used for the exploration of counterexamples. This application if named LeaFF, for "Learning Feasibility using Functional Depedencies". It was developed using React[3] for the frontend, and Python 3 and Flask[4] for the backend.

Leaff has two main functionalities:

[3] `https://reactjs.org/`
[4] `http://flask.pocoo.org/docs/0.12/`

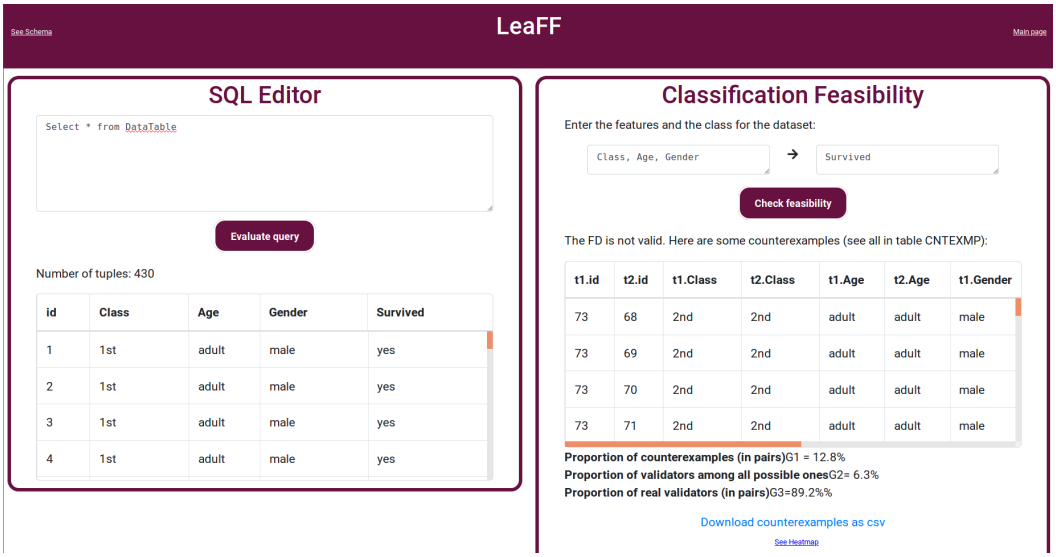


Fig. 5.8: LeaFF page for the exploration of counterexamples

- The first one is presented on figure 5.7: this page can be used to load a new classification to analyze. In addition, if the considered dataset contains continuous values, they can be discretized, by specifying the number of bins to produce for each column to discretize.

- The second and most important page is presented on figure 5.8: this is the page used to retrieve and explore the counterexamples. The interface on this page is divided in two zones. The first one on the left is, like for ExplIQuE (see section 4.6) a simple SQL editor: it can be used to explore the dataset using SQL. Once the counterexamples have been retrieved from the considered data, they are stored in a relational table, and can also be queried. The second part of the interface, on the right, is the one to be used to specify what are the features and what is the class to consider in the dataset: once they have been entered by the domain expert, she can ask to assess the feasibility of classification over the dataset. To this end, measures G_1, G_2 and G_3 are computed. Finally, the counterexamples are retrieved, and presented in a table.

Using LeaFF, it is therefore possible to obtain all the primary necessary information on the dataset to assess its adequacy with the classification task. Additionally, it is possible to propose some additional functionalities to simplify the exploration of counterexamples. First, since they are stored in a relational database, it is possible to query the counterexamples tables, therefore allowing to order them, select only part of them, etc. Indeed, for large datasets with many counterexamples, it appears necessary to allow the domain experts to explore these counterexamples by reorganizing the way they are presented: using a SQL query, they can implement filters to

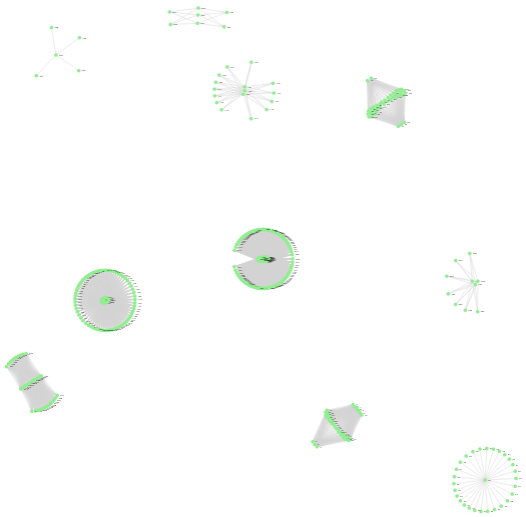


Fig. 5.9: LeaFF visualization of counterexample's for the real Titanic dataset: each green node is a tuple, and two tuples are connected if they form a counterexample

directly select the counterexamples of interest. Alternatively, the counterexamples can also be dowloaded in a csv file, to process them externally. It is also possible to display the counterexample graph for the considered relation and FD. The degree of a node indicates the number of counterexamples it is involved in, and is a first criterion to sort out the tuples to focus on. Such representations are really interesting to better understand and explore the counterexamples, because they allow to better grasp the interactions between the tuples. Moreover, this can be used to refine the data selection, for example by removing the tuples that are at the center of many counterexamples. An example of such a visualization offered by LeaFF is given on figure 5.9, for the real Titanic dataset: it clearly show clusters of counterexamples, usually centered around a few tuples. This visulization is interactive, as the user can drag the nodes of the graph to reshape a cluster, and get information on each node by pointing the mouse on it: it is therefore useful to understand what is the data behind each node.

5.6.3 Case study: prediction of the ageing of refrigerated vehicles

The approach developed in this section was tested on real data, during a collaboration with Cemafroid, a french company issuing attestations for refrigerated transport vehicles. We will now develop the lessons learned from this study.

Context

CEMAFROID[5] is a company with a french delegated public service, delivering conformity attestations of refrigerated transport vehicles. The main objective of these vehicles is to supply consumers with good quality and safe perishable products. In this context, CEMAFROID offers testing and calibration services, and has been designated as an approved body for issuing conformity attestations for refrigerated transport vehicles. One of the main characteristics of these vehicles if their "insulation coefficient", denoted by K, that evaluates well the vehicle is at preventing heat gain from the outside of the refrigerated enclosure. This coefficient is therefore very important, and may be controlled as required by the ATP (the international Agreement for the Transport of Perishable foodstuff). The lower the K coefficient the better, but due to the ageing of the vehicle, it increases over time. To control it and allow the vehicle to continue to transport perishable food, its thermal insulation is measured again after 12 years of service. The ratio between this value and the initial value allows too define the *ageing* of the vehicle.

To explain the variability in the ageing of refrigerated vehicles, several studies have looked at the problem to identify factors to explain it, and propose physical models for the ageing. [Cap+19] presented the factors playing a role in the ageing process. As stated by [Pan+95], the ageing of an insulated enclosure for a refrigerated vehicle is mostly due to the permeability of the foam to the gases, to the condensation of water into the foam cells and to the increase of the percentage of the broken cells. Ageing of refrigerated vehicles also has a mechanical component due to the movements on the road, the routes covered and the payload. A simple statistical analysis carried out in [Pan+99] highlighted the influence in the ageing rate due to the rails and to the refrigerating units.

But whereas all these studies rely on physical models to analyse the problem, the amount of data recorded on the refrigerated transport vehicles, makes it possible to build a predictive model, using machine learning techniques: such a new approach is a way to tackle the problem from a different angle, to possibly identify new causes for the ageing of vehicles, and to confront new results to the one obtained with physical models. Shifting from a physical to a numerical model is indeed a big change of paradigm, especially for the domain experts that mainly design and work with physical models. In this context, it was therefore important to be able to explain the performances of the trained model, and to analyze what were its limitations. To this end, we proposed to carry an analysis of the counterexamples in collaboration with Cemafroid's experts.

[5] `http://www.cemafroid.fr/`

Data selection

The ageing of a refrigerated vehicle used in the carriage of perishable foodstuffs is quantified through the evolution over the time of the K coefficient. The French Regulation requires that this coefficient is measured for all the in-service vehicles after twelve years of service. As a result, the *ageing* evaluation may be made by comparing the K coefficient value of the in-service vehicle at the twelfth year of life, K_{12} to the initial K value of the prototype equipment, K_p, through their ratio:

$$ageing = \frac{K_{12}}{K_p}$$

To predict this ageing, thermal engineers had access to the Datafrig® database, that records data for the attestations of refrigerated transport vehicles. At 31 December 2017, based on the Datafrig® data, the French fleet counted 110 000 refrigerated transport equipment with a valid ATP certification. These equipments are divided into different categories, the main ones are: vans (vehicles with a total weight allowed in charge smaller than 3.5 tons), trucks (vehicles whose total weight allowed in charge varies from 3.5 to 29 tons) and semi-trailers (vehicles with total weight allowed in charge higher than 29 tons). The Datafrig® database is managed by Cemafroid and contains all the ATP in-service equipment in France.

The data selection for the ageing prediction from Datafrig® was performed by Cemafroid, and only a very small subset of the database was selected: only 1158 highly curated data has been extracted from the 109 122 refrigerated equipment registered in the Datafrig® database. These 1158 data represent different in-service vehicles of different firms, tested after twelve years in the laboratory of Cemafroid. The selection criteria for the sample of 1158 vehicles were based on data quality requirements for the available data: no null values, no outliers, and no duplicates. It clearly appears that these simple criteria drastically lowered the number of tuples. We had to deal with this very limited number of tuples, because the data was processed by an external service provider, and we therefore did not have the control on the data selection. It was therefore not possible to ask for the query performing this selection, and we were not able to have access to more data.

For each of these vehicles more than 80 features are known. Ten of the most important features were selected by Cemafroid's experts, including the type of vehicle, the body manufacturer, the nature of the refrigerated enclosures and their isolation, the use of the vehicles and the different types of transport to which they may be subjected. Most of the available features are therefore categorical. A statistical analysis of this dataset is available in [Cap+19].

Classification model

Before studying the feasibility of classification using the available data, a first classification model had been built, by dividing the ageing into two classes: low (≤ 0.36) or high (> 0.36). This dicretization was done in collaboration with Cemafroid, based on the distribution of the ageing values.

Based on this, a first model had been built, using a decision tree [Bre17], in order to obtain an interpretable model, that could easily be discussed with thermal engineers. To improve the results, a boosted version of the algorithm (see [Sch90]) was used. The results were very encouraging, with a precision of 0.81 and a recall of 0.78 for the high ageing, which is the class of importance, as it is important to detect it early to anticipate maintenance issues. Additionally, the overall accuracy of the classifier was 0.69, which leaves room for improving the model.

As a result, before trying to improve the model itself, we decided to explore the classification dataset, and to explain its limitations the the domain experts, based on the satisfaction of the functional dependency between the features and the class. The purpose was to:

- Help Cemafroid's experts to better understand the limitation to the model's performances: is the obtained accuracy satisfying with respect to the available training data?
- Explore the counterexamples, and see if they could be explained by domain experts: in other terms, does the existence of such counterexamples make sense with respect to domain constraints?
- Try to find solutions to remove the most problematic counterexamples in the data selection phase, to be able to build more robust models in future works.

Counterexamples exploration

Using LeaFF, the classification dataset was analyzed to discuss with the experts. A toy sample of counterexamples is presented in table 5.7. For sake of clarity, only five attributes are represented in this example. Each line represents a counterexample, which is two tuples. As they both share the same values over the classification features, only the ageing column, on which they differ, is represented for both(Ageing t_1 and Ageing t_2).

id	Manufacturer	Cell Type	Insulation type	Vehicle type	Products	Ageing t_1	Ageing t_2
1	Firm 1	Integrated	Polyuréthane	Truck	Meat	High	Low
2	Firm 2	Integrated	Polyuréthane with cylopentane	Van	Fruits	High	Low
3	Firm 3	Rapportee	Polyuréthane with cylopentane	Panel truck	Frozen food	High	Low
4	Firm 4	Integrated	Polyuréthane with cylopentane	Trailer	Vegetables	High	Low
5	Firm 5	Rapportee	Polyuréthane with cylopentane without CFC	Truck	Cheese	High	Low
6	Firm 6	Rapportee	Polyuréthane with cylopentane	Remorque	Dairy products	High	Low

Tab. 5.7: Subset of counterexamples from the classificationd dataset, on the ageing of refrigerated transport vehicles.

In addition to the counterexamples, we started by computing the three main metrics for the satisfaction of the FD between the features and the class. The proportion of counterexamples, $G_1 = 9.02\%$, is low, showing that the pairs of tuples in the dataset are not a big proportion. However, as $G_2 = 100\%$, all tuples from the dataset are involved in at least one counterexamples: this is likely because a few tuples are in conflicts with almost all the other. This is confirmed by measuring $G_3 = 86.73\%$: this shows that to obtain a counterexamples-free dataset, the vast majority of the data can be saved. Additionnaly, it shows that G_3 is much higher than the obtained accuracy on the first model (69%): as a result, there are two possible actions to improve the performances of the model: remove the counterexamples to improve G_3, and improve the model's parameters to get the accuracy closer to the theoretical upper bound.

We then moved on to the analysis of counterexamples with Cemafroid's experts: we simply retrieved the list of counterexamples, and examined some of them when they appeared important to the experts. Based on the discussion that followed, it appeared that the presence of counterexamples in the dataset could be explained by several causes, that can be organized in the following categories:

Dirty Data A considerable amount of time had been spent on preparing and cleaning the dataset in order to use it for classification. However, when looking at counterexamples, it appeared that some tuples contained data that appeared to be incorrect. For counterexample 1 in table 5.7, it appeared that the vehicle with a low ageing was actually a van instead of a truck. Similarly, for

counterexample 2, the vehicle with a high ageing actually transported meat instead of vegetables. Such counterexamples are easy to fix by correcting the wrong values. Alternatively, it is possible to take into account additional attributes that are more precise than the dirty ones: for example, the weight of the vehicle can better define the vehicle's type than the given label, so mistakes can be avoided if it is taken into account: with the weight, vans and trucks could be more easily differentiated.

Missing information For other counterexamples, it appeared that the attributes selected for classification were not enough to explain their existence. However, other attributes, that had not been kept for classification, allowed to discriminate between the two tuples involved in the counterexample. For instance, in table 5.7, counterexample 3 can be removed if the number of food cases in the vehicle is considered. For counterexample 4, a specific characteristic of the cooling unit differed between the two tuples. As a result, taking those additional attributes, that at first had not been considered relevant for the classification, allowed to remove counterexamples, that will then improve the classifier's performances. This is a way to do feature engineering in collaboration with the domain experts.

Human Factor Finally, for a last group of counterexamples such as number 5 and number 6 from table 5.7, it appeared that the only explanation was a human factor, such as how the driver operates the vehicle. Indeed, this is susceptible of influencing the ageing of the vehicle, but it is hard to quantify, and may pose ethical issues. As a result, this last class of counterexamples is very difficult to fix, as data cannot be cleaned or completed. However, being aware that such counterexamples exist helps the data expert in understanding the limitations of the classification model. Finally, it also indicates other values that could be interesting to record: in this case for example the average speed of the vehicle. Similarly, during its life, the vehicle is subject to accidents of which nothing is known. Information about the nature and severity of such accidents could be useful for the study of ageing, and could also explain some

This categorization of counterexamples is very interesting: it allows to better understand what can be improved in the data recording process to produce less counterexamples in future classification datasets, but also show that in the considered one, some counterexamples can already be removed. However, further test were not conducted for the classification model, as it was then decided to build regression ones, to obtain more precise values for the ageing of the refrigerated vehicles. Moreover, because of many missing values and the low number of available

tuples in the study, it was not possible to compare exactly the models produce later on, because they were not trained on the exact same features.

Take-away lessons

The exploration of the counterexamples from Cemafroid's classification dataset was a very enriching experience, that created fruitful discussions and results. First, the counterexamples allowed to identify limitations in the dataset, and to take concrete actions to get higher quality data, and therefore improve the results for a future new model. The counterexamples are a powerful notion to avoid the domain expert from being overwhelmed by the data, as she then only have a small but meaningful subset of tuples to study. The counterexamples are therefore a perfect starting point for a discussion between data scientists and domain experts: while the first gain knowledge on data they are not expert on, the others can point out important information more easily. The counterexamples are a way for domain experts to read a concrete information that have an impact on their day-to-day work. By helping to clean and improve the dataset, domain experts stay in the loop, and better understand the data used in the learning process. Last but not least, they gain some evidence regarding whether or not it makes sense to infer a numerical model from their data. Despite its simplicity, this study also shows that in classification, it is not possible to ignore the field reality, and to only see the problem as a matrix of data: the physical model is as important as the data itself, and bridges have to be built between physical and numerical models.

This study with Cemafroid was interesting, mainly because it showed the interest of the counterexamples to better understand how the selected data impact the quality of the classifiers. However, because we did not have the control on the initial data selection, it was not possible to go further, for example by considering other refrigerated transport vehicles, or by modifying the initial query that is handle by the external service provider.

In the next section, we will discuss what can be done, starting from the counterexamples, to improve the data selection, for the cases in which it is possible, contrary to our collaboration with Cemafroid.

5.7 Data contextualization

Until now, we have seen how to evaluate the adequacy of a classification dataset with the predictive task using emasure G_3, and how to use the counterexamples related to this measure to better understand a predictive model limitations. This steps are important and necessary, because they allow to decide if the considered data is acceptable, and if the obtained model can be used for the considered application. But if the answers to this question is negative, then the natural question is: what to do next? What can be done to select data more appropriate for the considered classification problem?

5.7.1 The contextualization problem

As mentioned previously, when training a predictive model on a dataset, it relies on the assumption that there exists a function from the features to the class, that the model is trying to approximate. Because of noisy data, outliers, etc, there can be some counterexamples that do not allow the data to perfectly follow a function, but it is acceptable as long as these counterexamples are a minority, and this can be evaluated using measure G_3. The existence of such a function is also important because in many applications, the predictive model is a way to approximate a function that has a concrete reality: for example to predict physical phenomenon, there often exists an underlying physical equation/modelization, that the predictive model is trying to infer from the data: as a result, if the training data does not follow a function, then it is likely not representative of the underlying function that should exists in reality. In this scenario, it is therefore necessary to identify the data that is the most representative of the phenomenon that the predictive model is trying to grasp: as a result, the data selection should be refined to select the data that is the most coherent with the classification problem, in order to build a model that follows a function that is as close as possible to the phenomenon being modeled.

As a result, when a dataset does not appear to be adequate with the considered classification problem, it appears necessary to refine the data selection to obtain tuples that better fit the problem. The question is therefore to define what is the data that is the most appropriate to model the considered problem? We refer to this problem as *contextualization*, as among all the available data, the problem is to select the subset that correspond to a coherent context, which is the one in which the classification problem takes place. As a result, a context is described with respect to an initial set of available tuples r, by the number of tuples it selects from r, and

by its proportion of counterexamples. Contextualization corresponds to identifying the context that is the most coherent with the task at hand, and therefore to refining the data selection until having the adequate data. This contextualization problem can seem as a simple problem at first hand (mainly data selection), but turns out to be a nightmare in practice. Identifying the appropriate data is clearly not an easy task, and depends on the final objective for the classification model.

From a theoretical point of view, this problem appears clearly related to measure G_3. If we refer to the definition of this measure, then the subset of tuples s from the available data in r is the biggest possible number of tuples that does not contain any counterexample:

$$max(\{|s||s \subseteq r, s \models X \rightarrow Y\})$$

Indeed, such a subset s allows to select as many tuples as possible without introducing any counterexample: this way, it guaranties that there exists a function from X to C. However, from a practical point of view, this approach reaches its limits, for several reasons:

- It is true that there should be as few counterexamples as possible: however, removing all of them might not make sense with respect to the domain's reality, because real data is likely to contain outliers, noisy data, etc, that can make sense in a given context.
- Such an approach is difficult to explain to domain experts, because there is no concrete explanation for the removal of the tuples, other than "it was in a counterexample": this is a local explanation that can be given for each tuple, but there is no overall explanation to describe the subset of tuples in the obtained context. Therefore, it does not allow to describe what are the characteristics of the removed tuples (or of the selected one), meaning a context cannot be described with more elements than its size with respect to the initial relation.

From these two observations, it appears that contextualization should be able to remove counterexamples but also explain why they are removed, in order to be able to describe the context with respect to some characteristics: this way, the context can be understood and analyzed by domain experts, that can assess whether or not they make sense with the domain constraints. In addition, if the contexts can be characterized, then it means that these characteristics can be used to reperform the data selection even if the initial set of available tuples is modified.

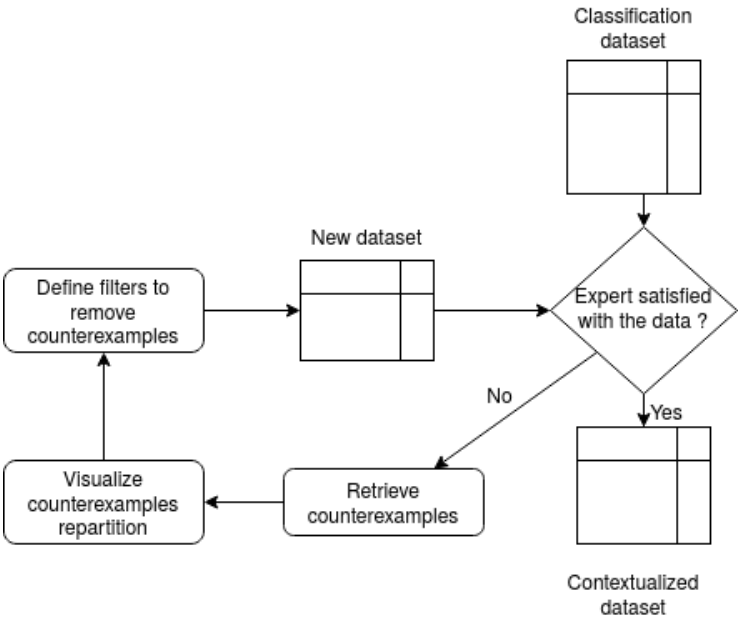


Fig. 5.10: Overview of the solution proposed to contextualize a classification dataset

As a consequence, we developed a methodology to characterize and select a context to refine a data selection, to make it more coherent with the considered classification problem. This methodology, that aims at keeping domain experts in the loop of the data selection, is presented in the following section.

5.7.2 Data selection of a context for classification

Methodology

The philosophy of the proposed methodology is to remove several counterexamples at a time, and to be able to characterize the ones that are removed. This way, G_3 can be improved over the classification datasets, and the removed counterexamples can be characterized. The selection conditions used to remove groups of counterexamples can also be used by domain experts to determine if such conditions make sense with respect to their domain constraints.

To contextualize a dataset, we therefore propose an iterative approach, that is summarized on figure 5.10. The process starts with an initial classification dataset. Then, $G3$ is computed, and a classifier is trained and tested, to obtain an accuracy measure: this is a first indication to see if the classification feasibility is satisfying, and if the initial model's performances are far from our theoretical upper bond. If the measures are satisfying for domain experts, then it is not necessary to contextualize

the considered dataset. In the opposite case, the counterexamples are retrieved from the dataset.

Starting from the counterexamples, the objective is to identify filters that can be applied to the considered dataset, so that they remove the tuples that cause too many counterexamples. The key is to find balance between removing regions of the data while keeping enough tuples and to improve G_3 and the accuracy. This filters can be seen as some sort of extensions from chapter 4, in the sense that they should help to reduce the selected data, by identifying pertinent selection conditions, that make sense with respect to the considered problem. They are therefore a way to refine the data selection by continuing the work of extensions outside of the database.

Context aware data selection

To define what tuples should be removed to define a context that is more coherent with the considered classification problem, we propose a two step process:

1. Identify the counterexamples to remove
2. Characterize these counterexamples to define the selection clauses to remove them. We call such selection *filters*, as they are use to filter the counterexamples from the data to consider for classification.

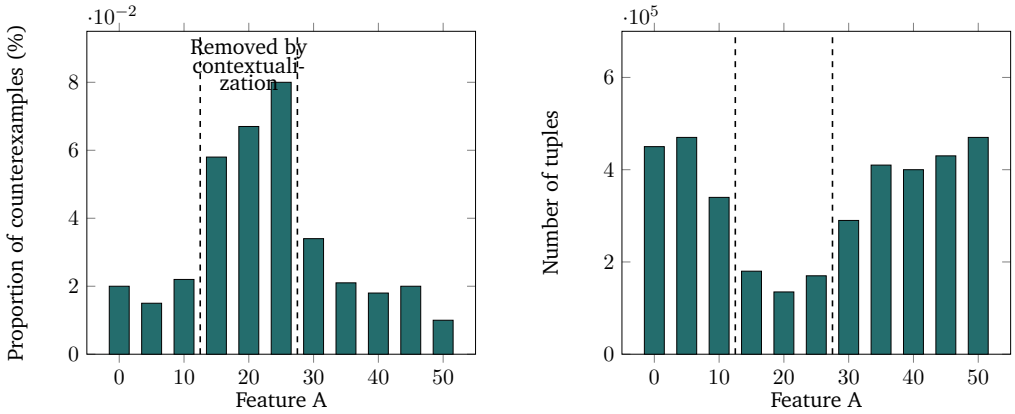
Such filters that can be applied to the dataset, to remove tuples and thus lower the number of counterexamples in the dataset, therefore improving $G3$. It is then necessary to design these filters. Ideally, they should remove as few tuples as possible, while removing as many counterexamples as possible. Sometimes, removing only one tuple might remove many counterexamples if the tuples was in conflict with many other tuples, that are in themselves only involved in one counterexample, as explained in example 15. We define the filters as follows:

Definition 11. *A filter F over a relation r to contextulized is a cunjunction of selection predicates, such that:*

$$F = \sigma_{c_1 \wedge \ldots \wedge c_n}$$

where c_i is an atomic formula, for every $i \in 1..n$

Applying the filters, we there obtain a new relation r_c^F:



(a) Counterexamples proportion for each value of a given feature

(b) Histogram of values frequency for a given feature

Fig. 5.11: Toy example for filter design

Definition 12. *Given a relation r, r_c^F is the result of the contextualization of r by filter F:*

$$r_c^F = \sigma_{c_1 \wedge \ldots \wedge c_n}(r)$$

Defining the filters to remove the tuples that cause too many counterexamples is a problem for which many solutions could be considered: algorithms could be defined to remove automatically, arbitrary cuts could be made, etc. Our main objective is to involve domain experts in the loop, as they are the most aware of the context they are looking for in the data.

We therefore consider easily interpretable filters, in the form of conjunction of conditions that are then applied to the dataset. These filters should define, in simple terms, regions of the dataset that contain more counterexamples that others, while not representing many tuples in the dataset: this is a form of anomaly detection based on counterexamples. This can be performed using visualization, that for each feature, propose histograms showing the distribution of values among counterexamples, and the number of tuples that have a given value. The histogram can then be used to identify values that, on a given feature, have few tuples that are involved in many counterexamples. The filters can then integrate a condition removing such values from the dataset. Visualization are usually very useful because they can be understood easily, and such filters are interpretable by domain experts, that can refine them using there domain knowledge, and analyze whether or not these filters make sense with the desired context.

Example 17. *Figure 5.11 presents an example of visualizations involving tuples and counterexamples, to show how they can be used to define filters. For a given feature*

A, figure 5.11a shows for each different value taken by this feature, the proportion of tuples involved in counterexample, and therefore how much they contribute to the value of $G3$. Figure 5.11b shows the number of tuples that take a a given value for the considered feature. By comparing these two visualizations, it appears that there is a zone that does not contain many tuples, but that is involved in many counterexamples. As a result, one condition for a contextualization filter could be to remove all tuples for which $A \geq 15$ and $A \leq 25$. This allows to obtain an interpretable filter, that does not remove too many tuples, but improves measure $G3$. Similar work can be performed for each feature of the dataset, creating a filter that is a conjunction of conditions over all features.

5.7.3 Case study: predictive maintenance for helicopters

To test the contextualization methodology, we had the opportunity to use it during a collaboration with Airbus Helicopters (AH), a company for which the contextualization problem is central for many predictive models. We will now present AH problems, and detail how we applied our contextualization solution to their datasets.

Predictive models for AH

AH is a company providing civil and military equipment in more than 150 countries around the world. For systems as complex as helicopters, predictive maintenance is critical, as ensuring safety is the first concern. AH is therefore dedicated to anticipating any failure as soon as possible. It has indisputable safety benefits as well as a significant economic impact, of high value for AH customers: they can plan any necessary maintenance operation ahead of time, order replacement part in advance, adapt mission planning, avoid last minutes cancellations, etc. From a mechanical perspective, early failure detection of one component can also prevent damaging to neighbouring parts. In this context, an important part of predictive maintenance solutions proposed by AH rely on predictive models, to be able to anticipate failures and to predict any incident beforehand.

As most of medium and heavy helicopters are equipped with flight recorders that record hundreds of parameters, there is a large volume of available data to build such predictive models: over the last decade, AH has gathered data on hundreds of thousands flight hours, over hundreds of helicopters operated by different customers worldwide, on many different types of missions. To face such a huge amount of data,

a Big Data platform has been deployed at AH to enable the storing and processing of large quantities of data [Mec+19]. All the data used for the predictive models of AH is selected using this platform.

AH made the choice to produce a specific type of predictive models, named *digital twins*: such model are used as virtual sensors, to predict a value that is also measured by a physical sensor, and an alert is raised is the difference between the real and the predicted value is to high. Such digital twins are therefore devised to model the *normal behavior* of different helicopter's systems and to identify as soon as possible small variations on core physical sensors.

In this context, and for AH predictive models, contextualization of the training dataset is very important: helicopters go through many different flight phases captured in the data, and only a subset of them is usually relevant for a given model, as they are the only one for which the laws of normal behavior of the system can apply. It is therefore necessary to identify the correct context for the considered task, which is the subset of data corresponding to the desired phases on which the model is applied. In the specific application to helicopters, contextualization therefore consists in determining the flight phases where considered parameters have lesser variability and are less subject to pilot maneuvers as well as to external parameters not recorded by the system. Additionally, helicopters are systems governed by physical laws, that apply only in specific contexts: the purpose of predictive models is therefore to produce outputs coherent with the physical laws governing the systems on which they are applied. To this end, these models have to be trained on data consistent with the physical model they represent. For all these reasons, AH already has to perform contextualization before being able to produce predictive models: this crucial step is usually dealt with by relying on experts knowledge who specify how to filter flight data.

During our collaboration with AH, we therefore applied our own contextualization methodology, in order to assist AH experts in the contextualization of their dataset, by providing a more systematic process, and to provide a solution that takes the predictive task into account. We will now detail the classification dataset on which we worked with AH, and the results that came out from this study.

Flight data from helicopters

The data generated by helicopters is recorded continuously during each flight, and transferred to be processed by the different AH services. This data is used to develop the virtual sensors, that aim at monitoring the aircraft health and usage. Virtual

	Baseline		Filter 1		Filter 2	
Dataset	# tuples	accuracy	# tuples	accuracy	# tuples	accuracy
Raw	1969533	53.97%	607248	57.28%	468630	61.71%
Expert-contextualized	541342	73.94%	281947	76.02%	100165	78.61%

Tab. 5.8: Accuracy of random forest models on the oil pressure datasets

sensors use the historical flight data to learn a predictive model (estimator) for a given parameter. This model is used to predict what the parameter value should be: this prediction is compared to the one given by the physical sensor measuring the actual value. If the difference between the virtual and real values is too significant, then the aircraft is not behaving according to its normal state, and a alert is raised to indicate the problem. An example of such a virtual sensor has been proposed by AH for the oil pressure of the helicopter Main Gear Box (MGB) [Dao+19]. We used the data from this virtual sensor to test our contexutalization approach, and to compare it to the one that was done manually on the initial AH study. Indeed, for this sensor, a first contextualization had been done by AH domain experts, identifying some stable flight phases based on their domain knowledge.

Based on this study, we were able to use and compare two different dataset, giving us a baseline against which the G_3-based contextualization can be compared, but also two datasets on which we can try our methodology:

Raw dataset It corresponds to the flight data without any contextualization, for a given period of time, randomly mixing tuples from several flights.

Expert-Contextualized dataset It is a subset of the raw one: it contains tuples filtered by AH experts (around 50% of the raw data). The context corresponds to stabilized flight phases, during which the helicopter altitude, speed and direction are maintained around restricted values.

Both datasets share the same 10 features chosen by experts. They include aircraft speed, altitude, etc, in addition to the MGB oil pressure to predict. All those features are continuous values, but have been discretized based on AH experts. The same discretization process was applied to the two datasets, so that the raw and expert-contextualized data are discretized in the same manner. All the results are therefore presented on the discretized data.

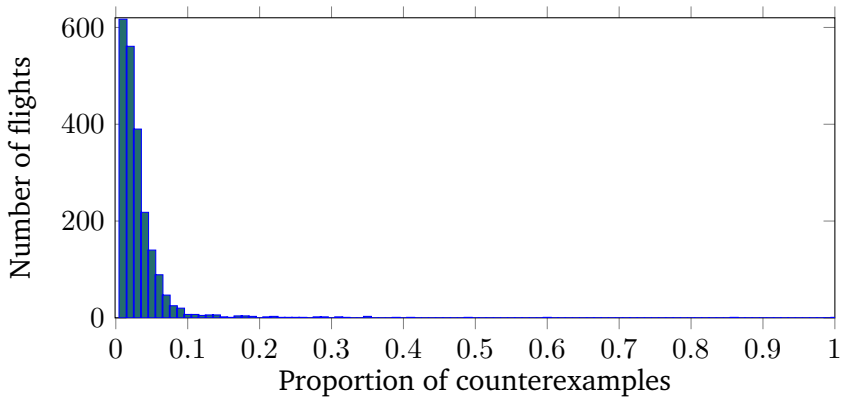


Fig. 5.12: Distribution of flights for each proportion of counterexamples

Comparison of AH datasets

We first compared how a classifiers performs depending on which dataset it is trained on, in order to see if the expert-contextualization performed by AH improved the quality of the datasets (in terms of counterexamples), and if it add an impact on the quality of classification (in terms of accuracy). With therefore trained two models, one for each dataset, using a random forest algorithm. The models were trained on 80% of the data, and tested on the remaining 20%, using cross-validation. The results of this test are presented in the *baseline* column of table 5.8. There is a huge difference as the accuracy for the expert-contextualized dataset is much higher than for the raw one. This is therefore a confirmation of the domain expert contextualization pertinence.

We computed G_3, which is $G_3 = 95.53\%$ for raw dataset and $G_3 = 95.51\%$ for expert-contextualized one. These results show that the proportion of counterexamples is reasonable, so the accuracy of classifiers is not too limited by them. In addition, the two datasets present very similar G_3 values. The domain expert contextualization seems to have preserved the proportion of counterexamples: they have decreased in absolute number, but with respect to the size of the dataset, there are still in the same proportions. New contextualization might here help to decrease their number and therefore increase the model's accuracy: after this initial analysis of AH data, we applied our contextualization methodology based on counterexamples.

Contextualization of AH data

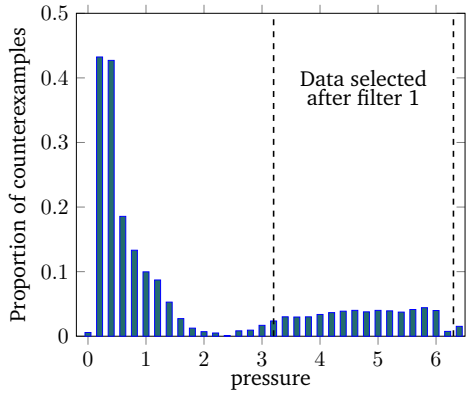
Starting from the two dataset, we applied our proposed methodology from figure 5.10. We first verified that the counterexamples were evenly distributed among the

flights, to make sure our process would not remove entire flights at once. Figure 5.12 show histogram of the percentage of counterexamples for the flights in the raw dataset. It appears that most flights share the same very low rate of counterexamples, so that any removal of large parts of counterexamples is very likely to affect a large number of flights. This ensures that even with additional contextualization, all flights will still be represented in the dataset. This guaranties that the model will consider a vast majority of flights with enough diversity, so that it is robust and does not overfit on a subpart of the flights.

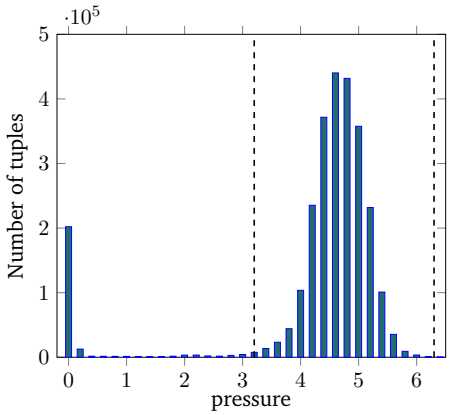
We then analyzed into details the distribution of values with respect to G_3. We made two plots for each feature and for each dataset: one showing the values distribution, and another showing for each different value the proportion of tuples corresponding to it and involved in a counterexample, exactly as in example 17. By looking at figures 5.13a and 5.13b, it appears that the highest counterexamples rates can be found for low pressure values, for which there seem to be a much higher proportion that for others. In addition, these values are not the most frequent, but still represent a non negligible fraction of this dataset. It can also be noted that the domain contextualization removes a significant part of counterexamples, as shown on figures 5.13c and 5.13d. But other regions remain that could be cleaned from counterexamples with additional contextualization, for example for pressure values over 5.6. A similar trend was observed for another feature of the dataset (rotor speed).

To perform a $G3$-based contextualization, it was decided to apply a filter to the dataset, that is named Filter 1 in table 5.8. For the pressure, this filter removes all the data for which the pressure is below 3.2 and above 6.4, as these regions have few tuples but highly contribute to counterexamples, as shown on figure 5.13. Similar rules were applied for the other features involved, such as rotor speed. The results in table 5.8 show the positive effect of such a filter on classifier's accuracy: for both the raw and expert-contextualized datasets it is increased significantly by the application of the filter.

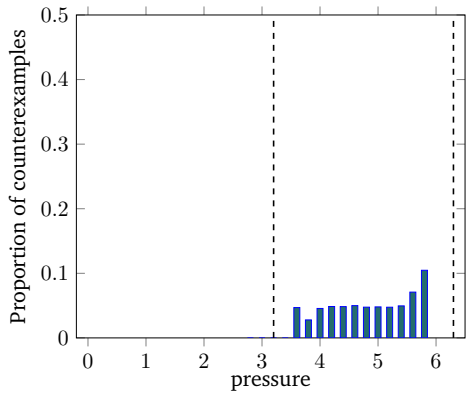
To see if the results could be improved even more, we identified additional regions with a higher counterexamples rate, and obtained Filter 2, that contains the rules of filter 1 and additional ones. This second filter includes the indicated air speed (IAS), for which the visualisations are provided on figure 5.14. On this figure, the contextualization appears clearly on the dataset, as there is a clear difference between the two datasets. Indeed, in the raw dataset, there are many counterexamples that appear for the low IAS values: they are removed by contextualization. However, there are still a few values with an elevated counterexample rate. Filter 2 therefore



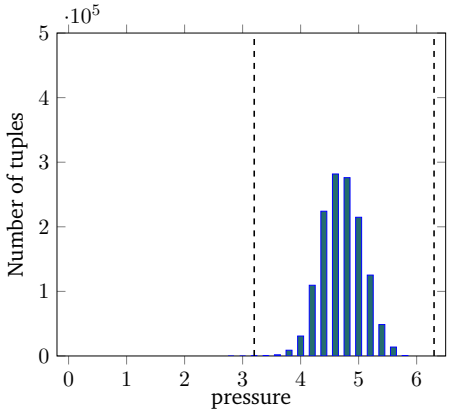
(a) Raw: counterexamples proportion against pressure



(b) Raw: histogram of pressure values



(c) Expert-contextualized: counterexamples proportion against pressure

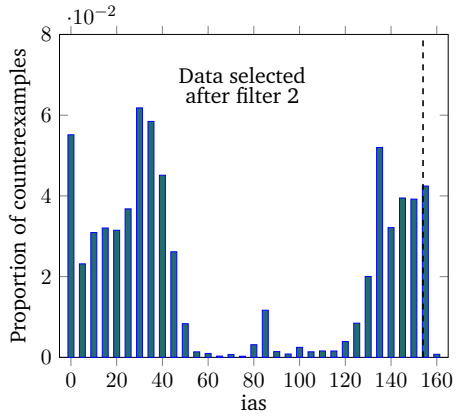


(d) Expert-contextualized: histogram of pressure values

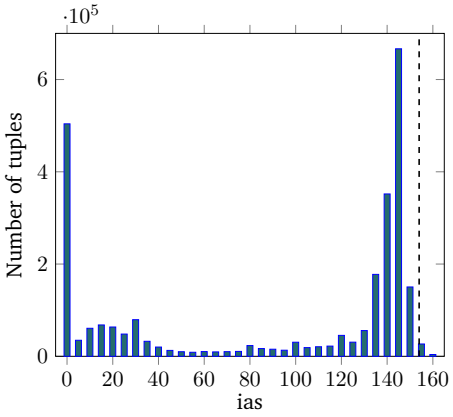
Fig. 5.13: Counterexamples and distribution for pressure values

takes IAS into account, as well as other parameters related to the torque of the helicopter's MGB. Based on figure 5.14, it was decided to remove the data for which $IAS > 155$. This conditions correspond to a small regions with many few tuples involved with many counterexamples, and also makes sense with respect to domain knowledge with respect to the normal behavior of the aircraft. Once again, table 5.8 shows that the accuracy results are improved by this new filter, as the context is more defined, and allows the data to make more sense for classification, while allowing the classifier to generalize well over the data despite the removal of tuples.

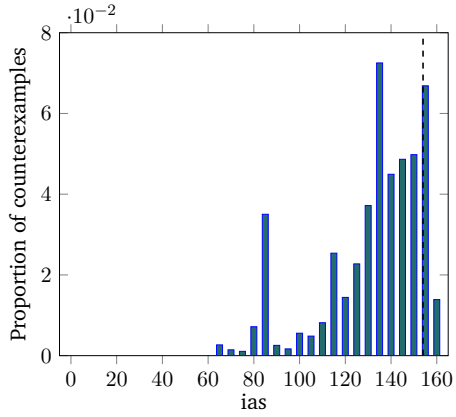
The selection conditions by the filter can here be seen as a refinement of the expert's contextualization, guided by the removal of counterexamples. In addition, the counterexamples can correspond to *dirty* data, it is a way to remove easily several counterexamples without having to look at them one by one, while still improving the dataset overall quality for classification. Finally, it should be noted



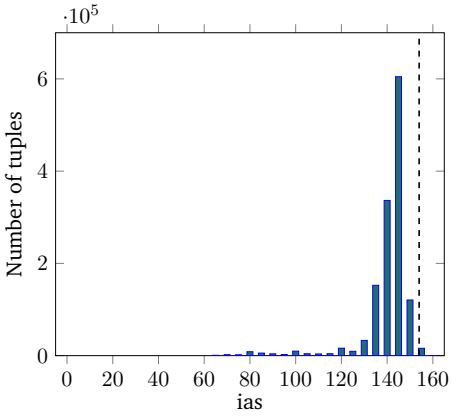
(a) Raw dataset: counterexamples proportion against IAS



(b) Raw: histogram of IAS values



(c) Expert-contextualized: counterexamples proportion against IAS



(d) Expert-contextualized: histogram of IAS values

Fig. 5.14: Counterexamples and distribution for IAS values

from table 5.8, that there is a significant gap between the highest accuracy on the raw dataset (61.71%), and the lowest accuracy for the expert-contextualized one (73.94%). Indeed, even with the best filter, it is not possible to reach the result that can be obtained using expert knowledge: the best approach consists in taking the valuable domain expert knowledge into account, before refining it using tools such as counterexamples and G_3: this combination is what gave the best results in this study.

Overall, these experiments have allowed to apply the framework presented on figure 5.10. For each dataset, we computed metrics and provided an additional $G3$ based contextualization, that allowed to give context elements for AH domain experts, and improved the classification performances of the dataset.

Take away lessons

These experimentation have demonstrated how the fine-tuning of data selection for contextualization can be used successfully to improve the accuracy of classification models for the digital twins and virtual sensors developed by AH, by using counterexamples to thee function the model is seeking to define. The methodology developed in this paper allows to identify context on which it makes sense to learn a model, by improving the initial context defined by domain experts. In addition, this method identifies healthy flight data, by removing counterexamples, therefore cleaning the data and improving the overall quality of the dataset. Contextualization is an important problem in the industry, but it is not easy to address because the proposed solution are often domain-specific, or included in the "data preparation" steps that our left to data scientists judgment: our solution could in comparison be applied for other types of application and involves domain experts in the loop.

There is also a qualitative aspect to this approach, that aims at taking a step back from the model, to understand what is being done, and understand the limitations. This is why the counterexamples are very important, to show and discuss why the proposed model is limited by the data used for the training. G_3 also gives a numerical upper bound for the accuracy, that explain very easily if (or if not) the model has reached its maximum performances. This is directly related to the explicability of the model, a crucial notion in aeronautics: every choice can have a tremendous impact on safety, and in case of a problem, it should be possible to explain every decision made and every choice. Indeed, the prediction of what can be seen as a simple classification algorithm output can put into question human lives getting back into an aircraft or not. As a result, for AH, any procedure that can explain better what is happening in the prediction process is valuable, and it will be taken into consideration when deciding if the model should be put into production and used in real time monitoring of helicopters.

In conclusion, this collaboration with AH turned out successful to show that our contextualization methodology can be used on classification dataset, to better assist domain experts in the data selection refinement for their predictive models.

5.8 Conclusion and perspectives

After questioning the data selection process in chapters 3 and 4, in this chapter, we sought to evaluate if the selected data is adequate with the task for which it

has been selected, especially for the training of predictive models. To this end, we started from the simple yet interesting remark that while predictive models seek to approximate a function from the features to the class to predict, the satisfaction of the functional dependency between such features and class can be used to say whether or not this function even exists. As the FD is likely to not be satisfied, we showed how measure G_3 can be used how far the data is from satisfying the FD, and how this measure can by extension be used to give a theoretical upper bound to classifier's accuracy on the considered data.

To go further in the evaluation of the dataset's adequacy, and to provide domain experts all the necessary information on their model's limitation, we showed how the analysis of counterexamples can be complementary to measure G_3, by allowing to better understand what prevents the dataset to be coherent with the classification task. We illustrated this process by the results from a collaboration with Cemafroid.

Finally, considering that the initial data selection might not provide the most adequate dataset for the considered classification task, we proposed to explore how the initial selection can be refined. To this end, we proposed to perform a contextualization of the dataset, by adding new selection clauses that take into account the classification problem, and therefore seek to improve the satisfaction of the functional dependency between the features and the class. We successfully applied our proposed methodology in a collaboration with Airbus Helicopters.

The three steps of this process are complementary, while each exploring a classification dataset from a different angle, to assess its adequacy with the classification task. This assessment is important, especially because with the democratization of machine learning libraries and associated tools, it is easy to build a model without questioning the quality of the data on which it is built. As a consequence, the various solutions proposed in this chapter are a way to take a step back from the model, and to better understand the data and its limitations before diving into the training process. Finally, it also provides solution to better perform the data selection, by taking into account the classification task for which it is selected.

Of course, it should be kept in mind that the proposed approach makes some assumptions, especially because it only takes into account the available data used for the training, and does not extrapolate for example by taking into account the distribution of the data: we argue that this is still an interesting approach because it allows to discuss data experts on the data they are familiar with, by providing information on the data that is maybe simple, but can turn out to be very useful in practice.

6 Conclusion

6.1 Research summary

Databases and machine learning are two distinct research domains, but because they focus on the same main object, i.e data, they are intimately intricated, and there are many possibilities for the two domains to benefit from one another, and to propose new solutions at their intersection. In this thesis, we focused on what happens when, to build a predictive model, the data transitions from the database world to the machine learning one.

To this end, we started by focusing on the data selection: to extract the data, it requires that the user is able to identify it, and to write the SQL query that can exactly select her desired data. We thus started in chapter 3 by studying the identification of the required data, with respect to all the available tuples in the database: we proposed to model this problem as an imbalanced classification one, and provided synthetic experimentations, to assess if it is easier to classify between distant relations. The results were mitigated, but this first chapter allowed to combine a machine learning approach with a key concept from relational databases (functional dependencies).

In chapter 4, we questioned how to write a SQL query to capture exactly the desired data, especially when it is not necessary obvious for the user how to write such query. We therefore propose our query extensions solutions, and performed a user evaluation to support the utility of such extensions.

We then focused on evaluating the selected data, by assessing its adequacy with the predictive task to perform, with a focus on classification problems. In this setting, we started by simply pointing out how, while predictive model seek to define a function in data, functional dependencies can assess the existence of a function. Based on this, we proposed to evaluate the adequacy of a dataset by the satisfaction of the functional dependency between the features and the class. From this, we proposed to use a measure to estimate how far a dataset is from satisfying this dependency, and used this measure as an upper bound for the accuracy of classifiers on the considered dataset. Then, we proposed solutions for two scenarios: first, if data

selections does not have to or cannot be refined, we proposed to understand the model's limitation by analysing the tuples preventing the dependency from being satisfied; then, we proposed a methodology for the cases where data selection should and can be refined. These two scenarios were backed up by application on real data in the context of industrial collaborations.

6.2 Future work and perspectives

Based on the contributions exposed in this thesis, several improvements and research directions can be considered to build up on this work. First, in the short term, the main priority seems to be the generalization of the work presented in chapter 5, first to also take into account regression problems with a more systematic approach and, more generally, by being able to consider more datatypes. Indeed, the study conducted during this thesis highlighted how the specificities of each dataset impacts greatly the evaluation of the adequacy of the dataset for the predictive model. Our main leads for this problem focus on being able to define custom comparability functions for each feature, to be able to provide a personalized comparison of tuples, and therefore retrieve counterexamples that make sense with respect to the domain constraint. Based on this, it is even possible to consider the comparison of several comparability scenarios, and to evaluate in which one the dataset is adequate, to see how strict the comparability can be.

In the longer term, it would be interesting to get the works from chapter 4 and 5 even closer, in order to take the predictive task into account directly into the initial data selection process. The idea would therefore be to characterize directly the results of a SQL query with respect to the task it is being selected for. This can mean evaluating G_3 directly in the database, providing visualizations like in section 5.7, and other mixed approach that allow to address the two main research questions of this thesis at the same time. Such an approach would be in line with a database incorporating machine learning features, and it would make it possible to perform everything directly in the database, by relying on optimizations from the DBMS.

Finally, more generally, this thesis and especially the industrial collaborations during which the proposed solutions were tested, highlighted the need, during a data science process, to put the domain experts at the center of the process: the fanciest approaches are useless is they cannot make sense to the people that best know the content of the data. It therefore appears crucial to devise solutions that allow a smooth collaboration between data scientists and domain experts.

Bibliography

[Abi+95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases: the logical level*. Addison-Wesley Longman Publishing Co., Inc., 1995 (cit. on p. 25).

[AK+18] Mahmoud Abo Khamis, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. "In-database learning with sparse tensors". In: *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM. 2018, pp. 325–340 (cit. on pp. 3, 104).

[Abo+12] Azza Abouzied, Joseph M Hellerstein, and Avi Silberschatz. "Playful query specification with DataPlay". In: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 1938–1941 (cit. on p. 59).

[Aff+19] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. "A comparative survey of recent natural language interfaces for databases". In: *The VLDB Journal* 28.5 (2019), pp. 793–819 (cit. on pp. 4, 9, 57).

[Ant+99] András Antos, Luc Devroye, and Laszlo Gyorfi. "Lower bounds for Bayes error estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.7 (1999), pp. 643–645 (cit. on p. 103).

[AA+11] Antonio Arauzo-Azofra, José Luis Aznarte, and José M Benítez. "Empirical study of feature selection methods based on individual feature evaluation for classification problems". In: *Expert Systems with Applications* 38.7 (2011), pp. 8170–8177 (cit. on p. 21).

[Arm74] William Ward Armstrong. "Dependency structures of database relationship". In: *Information processing* (1974), pp. 580–583 (cit. on pp. 15, 36).

[Bar+03] Ricardo Barandela, José Salvador Sánchez, V Garca, and Edgar Rangel. "Strategies for learning in class imbalance problems". In: *Pattern Recognition* 36.3 (2003), pp. 849–851 (cit. on p. 44).

[Bat+13] Leilani Battle, Michael Stonebraker, and Remco Chang. "Dynamic reduction of query result sets for interactive visualizaton". In: *2013 IEEE International Conference on Big Data*. IEEE. 2013, pp. 1–8 (cit. on p. 58).

[Bee+84] Catriel Beeri, Martin Dowd, Ronald Fagin, and Richard Statman. "On the structure of Armstrong relations for functional dependencies". In: *Journal of the ACM (JACM)* 31.1 (1984), pp. 30–46 (cit. on pp. 15, 37).

[Bid+14] Nicole Bidoit, Melanie Herschel, and Katerina Tzompanaki. "Query-based why-not provenance with nedexplain". In: 2014 (cit. on p. 25).

[Bil+06] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. “Adaptive blocking: Learning to scale up record linkage”. In: *Sixth International Conference on Data Mining (ICDM'06)*. IEEE. 2006, pp. 87–96 (cit. on p. 125).

[Blu+12] Lukas Blunschi, Claudio Jossen, Donald Kossmann, Magdalini Mori, and Kurt Stockinger. “Soda: Generating sql for business users”. In: *Proceedings of the VLDB Endowment* 5.10 (2012), pp. 932–943 (cit. on p. 57).

[Boh+07] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. “Conditional functional dependencies for data cleaning”. In: *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE. 2007, pp. 746–755 (cit. on pp. 25, 105).

[Bon+14] Angela Bonifati, Radu Ciucanu, and Slawomir Staworko. “Interactive inference of join queries”. In: 2014 (cit. on pp. 3, 28).

[Bos+98] Patrick Bosc, Didier Dubois, and Henri Prade. “Fuzzy functional dependencies and redundancy elimination”. In: *Journal of the American Society for Information Science* 49.3 (1998), pp. 217–235 (cit. on p. 122).

[Bre17] L Breiman. *Classification and regression trees*. Routledge, 2017 (cit. on p. 133).

[Bre+84] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984 (cit. on p. 68).

[Cap+19] C Capo, J-M Petit, R Revellin, G Bonjour, and G Cavalier. “Ageing of in-service refrigerated transport vehicles: a statistical analysis”. In: *25th IIR International Conference on Refrigeration*. 2019 (cit. on pp. 131, 132).

[CK97] Michael J Carey and Donald Kossmann. “On saying “enough already!” in sql”. In: *ACM SIGMOD Record*. Vol. 26. 2. ACM. 1997, pp. 219–230 (cit. on p. 53).

[Car+20] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. “Mining relaxed functional dependencies from data”. In: *Data Mining and Knowledge Discovery* 34.2 (2020), pp. 443–477 (cit. on p. 120).

[Car+15] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. “Relaxed functional dependencies—a survey of approaches”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.1 (2015), pp. 147–165 (cit. on p. 121).

[Car+19] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. “Machine Learning Interpretability: A Survey on Methods and Metrics”. In: *Electronics* 8.8 (2019), p. 832 (cit. on p. 104).

[Cha98] Surajit Chaudhuri. “Data mining and database systems: Where is the intersection?” In: *IEEE Data Eng. Bull.* 21.1 (1998), pp. 4–8 (cit. on p. 3).

[Cha+02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357 (cit. on p. 30).

[Chu+13] Xu Chu, Ihab F Ilyas, and Paolo Papotti. “Discovering denial constraints”. In: *Proceedings of the VLDB Endowment* 6.13 (2013), pp. 1498–1509 (cit. on p. 105).

[Chu+16] Xu Chu, Ihab F Ilyas, and Paraschos Koutris. “Distributed data deduplication”. In: *Proceedings of the VLDB Endowment* 9.11 (2016), pp. 864–875 (cit. on p. 125).

[Coc46] William G Cochran. “Relative accuracy of systematic and stratified random samples for a certain class of populations”. In: *The Annals of Mathematical Statistics* (1946), pp. 164–177 (cit. on p. 81).

[Cod02] Edgar F Codd. “A relational model of data for large shared data banks”. In: *Software pioneers*. Springer, 2002, pp. 263–294 (cit. on p. 2).

[Cum+17] Julien Cumin, Jean-Marc Petit, Vasile-Marian Scuturici, and Sabina Surdu. “Data exploration with sql using machine learning techniques”. In: 2017 (cit. on pp. 29, 68).

[DR00] Mehmet M Dalkilic and Edward L Roberston. “Information dependencies”. In: *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2000, pp. 245–253 (cit. on p. 18).

[Dao+19] Nassia Daouayry, Ammar Mechouche, Pierre-Loic Maisonneuve, Jean-Marc Petit, and Marian Scuturici. “Data-Centric Helicopter Failure Anticipation: The MGB Oil Pressure Virtual Sensor Case”. In: *International Conference on Big Data*. IEEE. 2019, 10 pages (cit. on p. 144).

[DL97] Manoranjan Dash and Huan Liu. “Feature selection for classification”. In: *Intelligent data analysis* 1.1-4 (1997), pp. 131–156 (cit. on p. 104).

[DR02] Luc De Raedt. “A perspective on inductive databases”. In: *ACM SIGKDD Explorations Newsletter* 4.2 (2002), pp. 69–77 (cit. on p. 3).

[Dim+14] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. “Explore-by-example: An automatic query steering framework for interactive data exploration”. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014, pp. 517–528 (cit. on p. 28).

[Dom99] Pedro Domingos. “Metacost: A general method for making classifiers cost-sensitive”. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 1999, pp. 155–164 (cit. on p. 30).

[DL16] Li Dong and Mirella Lapata. “Language to logical form with neural attention”. In: *arXiv preprint arXiv:1601.01280* (2016) (cit. on p. 58).

[DP13] Marina Drosou and Evaggelia Pitoura. “Ymaldb: exploring relational databases via result-driven recommendations”. In: *The VLDB Journal—The International Journal on Very Large Data Bases* 22.6 (2013), pp. 849–874 (cit. on p. 59).

[EJ01] Andrew Estabrooks and Nathalie Japkowicz. “A mixture-of-experts framework for learning from imbalanced data sets”. In: *International Symposium on Intelligent Data Analysis*. Springer. 2001, pp. 34–43 (cit. on p. 30).

[Fag+03] Ronald Fagin, Amnon Lotem, and Moni Naor. “Optimal aggregation algorithms for middleware”. In: *Journal of computer and system sciences* 66.4 (2003), pp. 614–656 (cit. on p. 53).

[Fan+11] Ju Fan, Guoliang Li, and Lizhu Zhou. "Interactive SQL query suggestion: Making databases user-friendly". In: *2011 IEEE 27th International Conference on Data Engineering*. IEEE. 2011, pp. 351–362 (cit. on p. 57).

[Fan08] Wenfei Fan. "Dependencies revisited for improving data quality". In: *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2008, pp. 159–170 (cit. on p. 105).

[Fan+09] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. "Reasoning about record matching rules". In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 407–418 (cit. on p. 20).

[Fer+18] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary". In: *Journal of artificial intelligence research* 61 (2018), pp. 863–905 (cit. on p. 30).

[FD+14] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. "Do we need hundreds of classifiers to solve real world classification problems?" In: *The journal of machine learning research* 15.1 (2014), pp. 3133–3181 (cit. on p. 117).

[FS95] Yoav Freund and Robert E Schapire. "A desicion-theoretic generalization of on-line learning and an application to boosting". In: *European conference on computational learning theory*. Springer. 1995, pp. 23–37 (cit. on p. 30).

[Fri97] Jerome H Friedman. "On bias, variance, 0/1—loss, and the curse-of-dimensionality". In: *Data mining and knowledge discovery* 1.1 (1997), pp. 55–77 (cit. on p. 66).

[FH87] K. Fukunaga and D. M. Hummels. "Bayes Error Estimation Using Parzen and k-NN Procedures". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.5 (1987), pp. 634–643 (cit. on p. 103).

[Fuk13] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013 (cit. on pp. 10, 22).

[Get19] Lise Getoor. "Responsible Data Science". In: *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. 2019, p. 1 (cit. on p. 98).

[Ghi+15] Luca M Ghiringhelli, Jan Vybiral, Sergey V Levchenko, Claudia Draxl, and Matthias Scheffler. "Big data of materials science: critical role of the descriptor". In: *Physical review letters* 114.10 (2015), p. 105503 (cit. on p. 103).

[Gui+18a] Marie Le Guilly, Jean-Marc Petit, and Marian Scuturici. "A First Experimental Study on Functional Dependencies for Imbalanced Datasets Classification". In: *Information Search, Integration, and Personalization - 12th International Workshop, ISIP 2018, Fukuoka, Japan, May 14-15, 2018, Revised Selected Papers*. 2018, pp. 116–133 (cit. on p. 11).

[Gui+19] Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici, and Ihab F. Ilyas. "ExplIQuE: Interactive Databases Exploration with SQL". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*. 2019, pp. 2877–2880 (cit. on p. 11).

[Gui+18b] Marie Le Guilly, Jean-Marc Petit, Vasile-Marian Scuturici, and Ihab F. Ilyas. "Partitioning queries for data exploration using query extensions". In: *BDA 2018 34ème conférence sur la Gestion de Données: Principes, Technologies et Applications*. Bucarest, Romania, 2018 (cit. on p. 11).

[Han05] Jiawei Han. *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005 (cit. on p. 65).

[Han+11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011 (cit. on pp. 30, 44).

[HV18] Melanie Herschel and Yannis Velegrakis. "On Data Exploration in the era of Big Data". In: *ACM SIGMOD Blog* (2018) (cit. on p. 52).

[Hua98] Zhexue Huang. "Extensions to the k-means algorithm for clustering large data sets with categorical values". In: *Data mining and knowledge discovery* 2.3 (1998), pp. 283–304 (cit. on p. 66).

[Huh+99] Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. "TANE: An efficient algorithm for discovering functional and approximate dependencies". In: *The computer journal* 42.2 (1999), pp. 100–111 (cit. on pp. 18, 25, 121).

[IL13] Stratos Idreos and Erietta Liarou. "dbTouch: Analytics at your Fingertips." In: *CIDR*. 2013 (cit. on p. 58).

[Idr+15] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. "Overview of data exploration techniques". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, pp. 277–281 (cit. on pp. 9, 51).

[IM96] Tomasz Imielinski and Heikki Mannila. "A database perspective on knowledge discovery". In: *Communications of the ACM* 39.11 (1996), pp. 58–64 (cit. on p. 3).

[Jai08] Anil K Jain. "Data clustering: 50 years beyond k-means". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2008, pp. 3–4 (cit. on p. 64).

[Jek+17] L Jekov, P Cordero, and M Enciso. "Fuzzy functional dependencies". In: *Fuzzy Sets and Systems* 317.C (2017), pp. 88–120 (cit. on p. 48).

[JL16] Robin Jia and Percy Liang. "Data recombination for neural semantic parsing". In: *arXiv preprint arXiv:1606.03622* (2016) (cit. on p. 58).

[Kas+10] Abhijith Kashyap, Vagelis Hristidis, and Michalis Petropoulos. "Facetor: cost-driven exploration of faceted query results". In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM. 2010, pp. 719–728 (cit. on p. 59).

[KS12] Gyula OH Katona and Attila Sali. “On the distance of databases”. In: *Annals of Mathematics and Artificial Intelligence* 65.2-3 (2012), pp. 199–216 (cit. on pp. 27, 31).

[Kat+10] Gyula OH Katona, Anita Keszler, and Attila Sali. “On the distance of databases”. In: *International Symposium on Foundations of Information and Knowledge Systems*. Springer. 2010, pp. 76–93 (cit. on pp. 16, 31).

[Kau+19] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. “A systematic review on imbalanced data challenges in machine learning: Applications and solutions”. In: *ACM Computing Surveys (CSUR)* 52.4 (2019), pp. 1–36 (cit. on p. 30).

[Kim+17] Been Kim, Martin Wattenberg, Justin Gilmer, et al. “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)”. In: *arXiv preprint arXiv:1711.11279* (2017) (cit. on p. 104).

[KR18] Benny Kimelfeld and Christopher Ré. “A relational framework for classifier engineering”. In: *ACM Transactions on Database Systems (TODS)* 43.3 (2018), pp. 1–36 (cit. on p. 3).

[KR92] Kenji Kira and Larry A Rendell. “A practical approach to feature selection”. In: *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 249–256 (cit. on p. 104).

[KM95] Jyrki Kivinen and Heikki Mannila. “Approximate inference of functional dependencies from relations”. In: *Theoretical Computer Science* 149.1 (1995), pp. 129–149 (cit. on pp. 11, 18, 19).

[KP03] Sotiris Kotsiantis and Panayiotis Pintelas. “Mixture of expert agents for handling imbalanced data sets”. In: *Annals of Mathematics, Computing & Teleinformatics* 1.1 (2003), pp. 46–55 (cit. on p. 30).

[Kot+06] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. “Handling imbalanced datasets: A review”. In: *GESTS International Transactions on Computer Science and Engineering* 30.1 (2006), pp. 25–36 (cit. on p. 30).

[Kou+09] Nick Koudas, Avishek Saha, Divesh Srivastava, and Suresh Venkatasubramanian. “Metric functional dependencies”. In: *2009 IEEE 25th International Conference on Data Engineering*. IEEE. 2009, pp. 1275–1278 (cit. on p. 20).

[KM+97] Miroslav Kubat, Stan Matwin, et al. “Addressing the curse of imbalanced training sets: one-sided selection”. In: *Icml*. Vol. 97. Nashville, USA. 1997, pp. 179–186 (cit. on p. 30).

[KS13] Ohbyung Kwon and Jae Mun Sim. “Effects of data set features on the performances of classification algorithms”. In: *Expert Systems with Applications* 40.5 (2013), 1847–1857 (cit. on p. 103).

[LG+20] Marie Le Guilly, Nassia Daouayry, Maisonneuve Pierre-Loic, et al. “Contextualisation of Datasets for better classification models: Application to Airbus Helicopters flight data”. In: *24th European Conference on Advances in Databases and Information Systems (ADBIS)*. Lyon, France, 2020 (cit. on p. 12).

[LG+19] Marie Le Guilly, Jean-Marc Petit, and Vasile-Marian Scuturici. “Evaluating classification feasability over datasets using functional dependencies”. In: *BDA 2019 35ème conférence sur la Gestion de Données: Principes, Technologies et Applications*. Lyon, France, 2019 (cit. on p. 12).

[LG+17] Marie Le Guilly, Jean-Marc Petit, and Vasile-Marian Scuturici. “Retour d’expérience sur l’analyse des données d’un tunnelier”. In: *BDA 2017 33ème conférence sur la Gestion de Données - Principes, Technologies et Applications*. Nancy, France, Nov. 2017 (cit. on p. 59).

[LL12] Mark Levene and George Loizou. *A guided tour of relational databases and beyond*. Springer Science & Business Media, 2012 (cit. on p. 13).

[LJ14] Fei Li and HV Jagadish. “Constructing an interactive natural language interface for relational databases”. In: *Proceedings of the VLDB Endowment* 8.1 (2014), pp. 73–84 (cit. on p. 58).

[Li+15] Hao Li, Chee-Yong Chan, and David Maier. “Query from examples: An iterative, data-driven approach to query construction”. In: *Proceedings of the VLDB Endowment* 8.13 (2015), pp. 2158–2169 (cit. on p. 28).

[Lis+18] M. Lissandrini, D. Mottin, T. Palpanas, Y. Velegrakis, and H. V. Jagadish. *Data Exploration Using Example-Based Methods*. 2018 (cit. on pp. 7, 24, 28).

[Liu+15] Henry Liu, Mingbin Xu, Ziting Yu, Vincent Corvinelli, and Calisto Zuzarte. “Cardinality estimation using neural networks”. In: *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*. IBM Corp. 2015, pp. 53–59 (cit. on p. 4).

[LS+96] Huan Liu, Rudy Setiono, et al. “A probabilistic approach to feature selection-a filter solution”. In: *ICML*. Vol. 96. Citeseer. 1996, pp. 319–327 (cit. on p. 104).

[Llo06] S. Lloyd. “Least Squares Quantization in PCM”. In: *IEEE Trans. Inf. Theor.* 28.2 (Sept. 2006), pp. 129–137 (cit. on p. 66).

[MP18] Ryan Marcus and Olga Papaemmanouil. “Deep reinforcement learning for join order enumeration”. In: *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 2018, pp. 1–4 (cit. on p. 4).

[Mar+19] Ryan Marcus, Parimarjan Negi, Hongzi Mao, et al. “Neo: A learned query optimizer”. In: *Proceedings of the VLDB Endowment* 12.11 (2019), pp. 1705–1718 (cit. on p. 4).

[Mar+18] Denis Mayr Lima Martins, Gottfried Vossen, and Fernando Buarque de Lima Neto. “Discovering SQL Queries from Examples using Intelligent Algorithms”. In: *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE. 2018, pp. 1–6 (cit. on p. 28).

[Mec+19] A. Mechouche, N. Daouayry, and V. Camerini. “Helicopter Big Data Processing and Predictive Analytics: Feedback and Perspectives”. In: *Proceedings of the 45th European Rotorcraft Forum*. Warsaw, Poland, 2019, 7 pages (cit. on p. 143).

[MK09] Chaitanya Mishra and Nick Koudas. “Interactive query refinement”. In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. ACM. 2009, pp. 862–873 (cit. on p. 53).

[Moh+18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. “Foundations of machine learning”. In: (2018) (cit. on p. 21).

[Mon+17] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. “Explaining nonlinear classification decisions with deep taylor decomposition”. In: *Pattern Recognition* 65 (2017), pp. 211–222 (cit. on p. 104).

[Mül+06] Heiko Müller, Johann-Christoph Freytag, and Ulf Leser. “Describing differences between databases”. In: *Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM. 2006, pp. 612–621 (cit. on p. 31).

[Nan+13] Arnab Nandi, Lilong Jiang, and Michael Mandel. “Gestural query specification”. In: *Proceedings of the VLDB Endowment* 7.4 (2013), pp. 289–300 (cit. on p. 58).

[Ng01] Wilfred Ng. “An extension of the relational data model to incorporate ordered domains”. In: *ACM Transactions on Database Systems (TODS)* 26.3 (2001), pp. 344–383 (cit. on p. 122).

[Ng99] Wilfred Ng. “Ordered functional dependencies in relational databases”. In: *Information Systems* 24.7 (1999), pp. 535–554 (cit. on p. 122).

[Ort+19] Jennifer Ortiz, Magdalena Balazinska, Johannes Gehrke, and S Sathiya Keerthi. “An Empirical Analysis of Deep Learning for Cardinality Estimation”. In: *arXiv preprint arXiv:1905.06425* (2019) (cit. on p. 4).

[Pan+95] G. Panozzo, Boldrin B., G. Minotto, B. Toniolo, and N. Biancardi. “Ageing of insulated vehicles: theoretical model and experimental analysis”. In: *Proc. 19th Int. Congr. Refrig., Den Hague, Netherlands, Vol. II* II (1995), pp. 583–589 (cit. on p. 131).

[Pan+99] G. Panozzo, O. Alberti, B. Toniolo, A. Barizza, and B. Boldrin. “Parameters affecting the ageing of insulated vehicles”. In: *Proc 20th International Congress of Refrigeration* IV (1999) (cit. on p. 131).

[Par+13] Aditya Parameswaran, Neoklis Polyzotis, and Hector Garcia-Molina. “Seedb: Visualizing database queries efficiently”. In: *Proceedings of the VLDB Endowment* 7.4 (2013), pp. 325–328 (cit. on p. 58).

[PR05] Laxmi Parida and Naren Ramakrishnan. “Redescription mining: Structure theory and algorithms”. In: *AAAI*. Vol. 5. 2005, pp. 837–844 (cit. on p. 64).

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 44, 74, 114).

[QR14] Bahar Qarabaqi and Mirek Riedewald. “User-driven refinement of imprecise queries”. In: *2014 IEEE 30th International Conference on Data Engineering*. IEEE. 2014, pp. 916–927 (cit. on p. 59).

[RD00] Erhard Rahm and Hong Hai Do. “Data cleaning: Problems and current approaches”. In: *IEEE Data Eng. Bull.* 23.4 (2000), pp. 3–13 (cit. on p. 105).

[RM88] KVSVN Raju and Arun K Majumdar. “Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems”. In: *ACM Transactions on Database Systems (TODS)* 13.2 (1988), pp. 129–166 (cit. on p. 122).

[Ran71] William M Rand. “Objective criteria for the evaluation of clustering methods”. In: *Journal of the American Statistical association* 66.336 (1971), pp. 846–850 (cit. on p. 81).

[RK04] Bhavani Raskutti and Adam Kowalczyk. “Extreme re-balancing for SVMs: a case study”. In: *ACM Sigkdd Explorations Newsletter* 6.1 (2004), pp. 60–69 (cit. on p. 30).

[Rek+17] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. “Holoclean: Holistic data repairs with probabilistic inference”. In: *Proceedings of the VLDB Endowment* 10.11 (2017), pp. 1190–1201 (cit. on p. 105).

[Roh+19] Yuji Roh, Geon Heo, and Steven Euijong Whang. “A survey on data collection for machine learning: a big data-ai integration perspective”. In: *IEEE Transactions on Knowledge and Data Engineering* (2019) (cit. on p. 5).

[Rou87] Peter J Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65 (cit. on pp. 67, 125).

[Sa+19] Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. “A Formal Framework for Probabilistic Unclean Databases”. In: *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*. 2019, 6:1–6:18 (cit. on p. 105).

[Sah+16] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, et al. “ATHENA: an ontology-driven system for natural language querying over relational data stores”. In: *Proceedings of the VLDB Endowment* 9.12 (2016), pp. 1209–1220 (cit. on pp. 4, 58).

[Sal+19] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. “Interventional fairness: Causal database repair for algorithmic fairness”. In: *Proceedings of the 2019 International Conference on Management of Data*. ACM. 2019, pp. 793–810 (cit. on p. 105).

[Sch90] R Schapire. “The strength of weak learnability”. In: *Machine learning* 5.2 (1990), pp. 197–227 (cit. on p. 133).

[Sch+16] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. “Learning linear regression models over factorized joins”. In: *Proceedings of the 2016 International Conference on Management of Data*. ACM. 2016, pp. 3–18 (cit. on pp. 3, 104).

[Scu+15] David Sculley, Gary Holt, Daniel Golovin, et al. “Hidden technical debt in machine learning systems”. In: *Advances in neural information processing systems*. 2015, pp. 2503–2511 (cit. on p. 4).

[SK16] Thibault Sellam and Martin Kersten. “Cluster-driven navigation of the query space”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.5 (2016), pp. 1118–1131 (cit. on p. 66).

[Sel+17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626 (cit. on p. 104).

[Sha+18] Shiven Sharma, Colin Bellinger, Bartosz Krawczyk, Osmar Zaiane, and Nathalie Japkowicz. “Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance”. In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2018, pp. 447–456 (cit. on p. 30).

[She+14] Yanyan Shen, Kaushik Chakrabarti, Surajit Chaudhuri, Bolin Ding, and Lev Novik. “Discovering queries based on example tuples”. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014, pp. 493–504 (cit. on pp. 3, 28).

[Son10] Shaoxu Song. “Data dependencies in the presence of difference”. PhD thesis. Hong Kong University of Science and Technology, 2010 (cit. on p. 120).

[Sto+08] Chris Stolte, Diane Tang, and Pat Hanrahan. “Polaris: a system for query, analysis, and visualization of multidimensional databases”. In: *Communications of the ACM* 51.11 (2008), pp. 75–84 (cit. on p. 58).

[SI18] Michael Stonebraker and Ihab F Ilyas. “Data Integration: The Current Status and the Way Forward.” In: *IEEE Data Eng. Bull.* 41.2 (2018), pp. 3–9 (cit. on p. 51).

[Tan+14] Jiliang Tang, Salem Alelyani, and Huan Liu. “Feature selection for classification: A review”. In: *Data classification: algorithms and applications* (2014), p. 37 (cit. on p. 103).

[Tra+09] Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. “Query by output”. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 2009, pp. 535–548 (cit. on pp. 3, 28).

[Tra+14] Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. “Query Reverse Engineering”. In: *The VLDB Journal* 23.5 (Oct. 2014), pp. 721–746 (cit. on pp. 7, 64).

[Leg] “Transport Vehicles (industrial paper)”. In: (cit. on p. 12).

[Tru+19] Immanuel Trummer, Junxiong Wang, Deepak Maram, et al. “SkinnerDB: regret-bounded query evaluation via reinforcement learning”. In: *Proceedings of the 2019 International Conference on Management of Data*. 2019, pp. 1153–1170 (cit. on p. 4).

[Tuk77] John W Tukey. *Exploratory data analysis*. Vol. 2. Reading, Mass., 1977 (cit. on p. 104).

[TG96] Kagan Tumer and Joydeep Ghosh. “Estimating the Bayes error rate through classifier combining”. In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 2. IEEE. 1996, pp. 695–699 (cit. on p. 22).

[Tzi+08] Yannis Tzitzikas, Nikos Armenatzoglou, and Panagiotis Papadakos. “FleXplorer: A framework for providing faceted and dynamic taxonomy-based information exploration”. In: *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*. IEEE. 2008, pp. 392–396 (cit. on p. 59).

[Uta+18] Prasetya Utama, Nathaniel Weir, Fuat Basik, et al. “An end-to-end neural natural language interface for databases”. In: *arXiv preprint arXiv:1804.00401* (2018) (cit. on p. 58).

[VA+17] Dana Van Aken, Andrew Pavlo, Geoffrey J Gordon, and Bohan Zhang. “Automatic database management system tuning through large-scale machine learning”. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. 2017, pp. 1009–1024 (cit. on p. 4).

[Vap+94] Vladimir Vapnik, Esther Levin, and Yann Le Cun. “Measuring the VC-dimension of a learning machine”. In: *Neural computation* 6.5 (1994), pp. 851–876 (cit. on p. 22).

[Wan+03] Haixun Wang, Carlo Zaniolo, and Chang Richard Luo. “ATLAS: A small but complete SQL extension for data mining and data streams”. In: *Proceedings 2003 VLDB Conference*. Elsevier. 2003, pp. 1113–1116 (cit. on p. 3).

[Wan+16] Wei Wang, Meihui Zhang, Gang Chen, et al. “Database meets deep learning: Challenges and opportunities”. In: *ACM SIGMOD Record* 45.2 (2016), pp. 17–22 (cit. on p. 3).

[WL18] Ziheng Wei and Sebastian Link. “DataProf: semantic profiling for iterative data cleansing and business rule acquisition”. In: *Proceedings of the 2018 International Conference on Management of Data*. ACM. 2018, pp. 1793–1796 (cit. on p. 105).

[WC17] Yaacov Y Weiss and Sara Cohen. “Reverse engineering spj-queries from examples”. In: *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 2017, pp. 151–166 (cit. on p. 28).

[Wu+16] Chia-Chi Wu, Yen-Liang Chen, Yi-Hung Liu, and Xiang-Yu Yang. “Decision tree induction with a constrained number of leaf nodes”. In: *Applied Intelligence* 45.3 (2016), pp. 673–685 (cit. on p. 69).

[Wu+14] Eugene Wu, Leilani Battle, and Samuel R Madden. “The case for data visualization management systems: vision paper”. In: *Proceedings of the VLDB Endowment* 7.10 (2014), pp. 903–906 (cit. on p. 58).

[Yu+07] Bei Yu, Guoliang Li, Karen Sollins, and Anthony KH Tung. “Effective keyword-based selection of relational databases”. In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007, pp. 139–150 (cit. on p. 57).

[YL03] Lei Yu and Huan Liu. “Feature selection for high-dimensional data: A fast correlation-based filter solution”. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*. 2003, pp. 856–863 (cit. on p. 104).

[Zha+13] Meihui Zhang, Hazem Elmeleegy, Cecilia M Procopiuc, and Divesh Srivastava. “Reverse engineering complex join queries”. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 2013, pp. 809–820 (cit. on p. 28).

[Zha+03] Shichao Zhang, Chengqi Zhang, and Qiang Yang. “Data preparation for data mining”. In: *Applied artificial intelligence* 17.5-6 (2003), pp. 375–381 (cit. on p. 105).

[Zho+17] Victor Zhong, Caiming Xiong, and Richard Socher. “Seq2sql: Generating structured queries from natural language using reinforcement learning”. In: *arXiv preprint arXiv:1709.00103* (2017) (cit. on p. 4).

[ZP09] Bin Zhou and Jian Pei. “Answering aggregate keyword queries on relational databases using minimal group-bys”. In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. 2009, pp. 108–119 (cit. on p. 57).

[Zlo75] Moshé M Zloof. “Query by example”. In: *Proceedings of the May 19-22, 1975, national computer conference and exposition*. 1975, pp. 431–438 (cit. on p. 28).

[Zou+06] Beibei Zou, Xuesong Ma, Bettina Kemme, Glen Newton, and Doina Precup. “Data mining using relational database management systems”. In: *Pacific-asia conference on knowledge discovery and data mining*. Springer. 2006, pp. 657–667 (cit. on pp. 3, 104).

List of Figures

List of Tables

UNIVERSITÉ DE LYON

INSA

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : LE GUILLY
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 24/09/2020

Prénoms : Marie Morgane Coralie

TITRE : Guided data selection for predictive models

NATURE : Doctorat

Numéro d'ordre : 2020LYSEI072

Ecole doctorale : N°512 InfoMaths

Spécialité : Informatique

RESUME :
Databases and machine learning (ML) have historically evolved as two separate domains: while databases are used to store and query the data, ML is devoted to predictive models inference, clustering, etc. Despite its apparent simplicity, the "data preparation" step of ML applications turns out to be the most time-consuming step in practice. Interestingly this step encompasses the bridge between databases and ML. In this setting, we raise and address three main problems related to data selection for building predictive models. First, the database usually contains more than the data of interest: how to separate the data that the analyst wants from the one she does not want? We propose to see this problem as imbalanced classification between the tuples of interest and the rest of the database. We develop an undersampling method based on the functional dependencies of the database. Second, we discuss the writing of the query returning the tuples of interest. We propose a SQL query completion solution based on data semantics, that starts from a very general query, and helps an analyst to refine it until she selects her data of interest. This process aims at helping the analyst to design the query that will eventually select the data she requires. Third, assuming the data has successfully been extracted from the database, the next natural question follows: is the selected data suited to answer the considered ML problem? Since getting a predictive model from the features to the class to predict amounts to providing a function, we point out that it makes sense to first assess the existence of that function in the data. This existence can be studied through the prism of functional dependencies, and we show how they can be used to understand a model's limitation, and to refine the initial data selection if necessary.

MOTS-CLÉS : databases, data selection, predictive models

Laboratoire (s) de recherche : LIRIS

Directeur de thèse: Jean-Marc Petit et Vasile-Marian Scuturici

Président de jury : Nicole Bidoit-Tollu

Composition du jury : Antoine Cornuejols, Sebastian Link, Nicole Bidoit-Tollu, Mohand-Saïd Hacid, Jean-Marc Petit, Vasile-Marian Scuturici