Implementing Privacy-Preserving Filters in the MOA Stream Mining Framework

Bachelor Degree Thesis

Author:
David Martínez Rodríguez

Supervisor:
Dr. Jordi Nin Guerrero
Department:
Computer Architecture

Bachelor Degree in Informatics Engineering
Major in Computer Science

Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

April 2015

"Al fin y al cabo, somos lo que hacemos para cambiar lo que somos."

Eduardo Galeano

Abstract

Implementing Privacy-Preserving Filters in the MOA Stream Mining Framework

by David Martínez Rodríguez

Data mining enables a better understanding of human and natural processes by analyzing massive amounts of data with machine learning algorithms. Stream mining is a process that allows us to discover knowledge in data when it comes in the form of a continuous stream. MOA, initials for Massive Online Analysis, is an open source data stream mining framework, developed at the University of Waikato, New Zealand. One of the available features in MOA is the use of filters, which can process streaming data before or after being fed to other subsystems, such as machine learning algorithms.

Although data science has brought us many benefits, because the data being analyzed is often personal and sensitive, we face the threat of losing our privacy. Statistical Disclosure Control (SDC) deals with controlling that information about specific individuals is not extracted from released datasets, whilst maintaining the statistical significance of the masked data. By applying SDC techniques to data, disclosure is prevented, thus effectively protecting the privacy of the data owners.

Four MOA *privacy-preserving filters* have been developed in this project, which implement the following SDC methods: **noise addition**, **microaggregation**, data **rank swapping** and **differentially private microaggregation**. Each of the algorithms has been adapted from well-known solutions in order to enable their utilization for stream processing tasks. Finally, the filters have been benchmarked to assess their quality in terms of two important SDC measurments: disclosure risk and information loss.

Resum

Implementing Privacy-Preserving Filters in the MOA Stream Mining Framework

per David Martínez Rodríguez

La mineria de dades ens ajuda a entendre millor els processos antropogènics i naturals, analitzant quantitats massives de dades, mitjançant algorismes d'aprenentatge automàtic. La mineria de *fluxos* de dades és un paradigma d'anàlisi que ens permet extreure coneixement de les dades quan són rebudes en forma d'un flux continu. El paquet de programari MOA, de l'anglès *Massive Online Analysis*, és un entorn de mineria de fluxos de codi obert, desenvolupat a la Universitat de Waikato, a Nova Zelanda. Una de les funcionalitats de MOA és la possibilitat d'utilitzar filtres, els quals processen les dades en flux abans o després de ser redirigides cap a altres sub-sistemes, com ara algorismes d'aprenentatge automàtic.

Tot i que la mineria de dades ens aporta molts beneficis, les dades que s'analitzen són, sovint, personals i sensibles. Ens trobem, doncs, davant d'un escenari en el que la nostra privacitat està en perill. L'SDC, de l'anglès *Statistical Disclosure Control*, és un camp que estudia mecanismes per controlar que la informació d'un individu específic no sigui extreta dels conjunts de dades publicades, alhora que s'intenta preservar la utilitat estadística de les dades emmascarades. Aplicant tècniques d'SDC, s'impedeix la re-identificació dels individus, protegint, per tant, la privacitat dels mateixos.

En aquest projecte s'han desenvolupat quatre *filtres de preservació de la privacitat* per l'entorn MOA, que implementen els següents mètodes d'SDC: **addició de soroll**, **microagregació**, **intercanvi de rangs** i **microagregació de privacitat diferencial**. Cadascun dels algorismes ha estat adaptat d'algun mètode ja conegut, en ús, per habilitar la seva utilització per a tasques de processament de fluxos. Finalment, tots quatre filtres han estat avaluats respecte de dues mesures molt importants en l'àmbit de l'SDC: el risc de revelació i la pèrdua d'informació.

Resumen

Implementing Privacy-Preserving Filters in the MOA Stream Mining Framework

por David Martínez Rodríguez

La minería de datos nos ayuda a entender mejor los procesos antropogénicos y naturales, analizando cantidades masivas de datos, mediante algoritmos de aprendizaje automático. La minería de *flujos* datos es un paradigma de análisis que nos permite extraer conocimiento de los datos, cuando éstos son recibidos en forma de un flujo continuo. El paquete de software MOA, del inglés *Massive Online Analysis*, es un entorno de minería de flujos de código abierto, desarrollado en la Universidad de Waikato, Nueva Zelanda. Una de las funcionalidades de MOA es la posibilidad de utilizar filtros, los cuales procesan los datos de los flujos antes o después de ser redirigidos hacia otros subsistemas, como los algoritmos de aprendizaje automático.

Aunque la minería de datos nos aporta muchos beneficios, los datos que se analizan son frecuentemente personales y sensibles. Nos encontramos, pues, ante un escenario en el que nuestra privacidad está en peligro. El campo de SDC, del inglés Statistical Disclosure Control, estudia los mecanismos para controlar que la información de un individuo específico no se extraiga de los conjuntos de datos publicados, a la vez que se intenta maximizar la utilidad estadística de los datos enmascarados. Aplicando técnicas de SDC, se impide la re-identificación de los individuos, protegiendo, por lo tanto, la privacidad de los mismos.

En este proyecto se han desarrollado cuatro *filtros de preservación de la privacidad* para el entorno MOA, que implementan los siguientes métodos de SDC: **adición de ruido**, **microagregación**, **intercanvio de rangos** y **microagregación de privacidad diferencial**. Cada uno de los algoritmos ha sido adaptado de algun método ya conocido y en uso, para habilitar su utilización para tareas de procesamiento de flujos. Finalmente, los cuatro filtros se han evauado respecto dos medidas muy importantes en el ámbito del SDC: el riesgo de revelación y la pérdida de información.

Acknowledgements

First and foremost, I would like to thank Dr. Jordi Nin for his advice and guidance throughout the entire development of this project, as well as his continued support to help me achieve my present and future goals. To him and all the professors I ever had, I am grateful of their will to teach and share their knowledge and sapience with all of us students; it is a great task you carry on your shoulders and for that you have my heartfelt gratitude.

Thanks to all inLab *'ers* too. There, I have grown as an engineer and, most importantly, as a person, surrounded by amazing people. Thanks to Rosa, Albert and Jaume for their guidance, and thanks to all with whom I have had the most enthralling experiences at the faculty: Albert, Anna, Arnau, Carlos, Eli, Germán, Jaume, Joel, Maki, Marc, MJ, Natxo, Néstor, and sure I forget someone.

I am most thankful to my dearest friends Bernat, Borja, Laura, Natàlia, Pau; all these years have been amazing by your side. With you, life is brighter and the future is always awaiting on the horizon.

This project would not be without my family. All my gratitude and love is for my parents and my brother, who support me and raise me and teach me the most wonderful things. Thank you.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

What follows is a brief introduction to the project under review in this report and its context, in a broad sense. The structure of this document is also outlined in the last section of this chapter.

1.1 Context

An overview is now given of the two main concepts that, when combined, drive the motivation behind the inception and development of this project.

1.1.1 Data mining

Information society produces vast amounts of data all over the world. This data comes from innumerable sources and in diverse formats, and has been stored for years in data warehouses, waiting to be processed. With the continuous increase in computing power, due to the recent advances in software and hardware technologies, the machine learning field, also known as *data science*, has arisen, allowing us to exploit this stored data and extract knowledge from it.

Data science is, indeed, a holistic process, where many different disciplines are involved, from data acquisition and storage, through its selection, filtering and analysis up to knowledge extraction, visualization and discovery.

Data science enables a better understanding of human or natural processes and provides us with means to identify trends, predict future events or discover useful patterns. Its uses range from scientific and medical applications to social sciences or business administration [1].

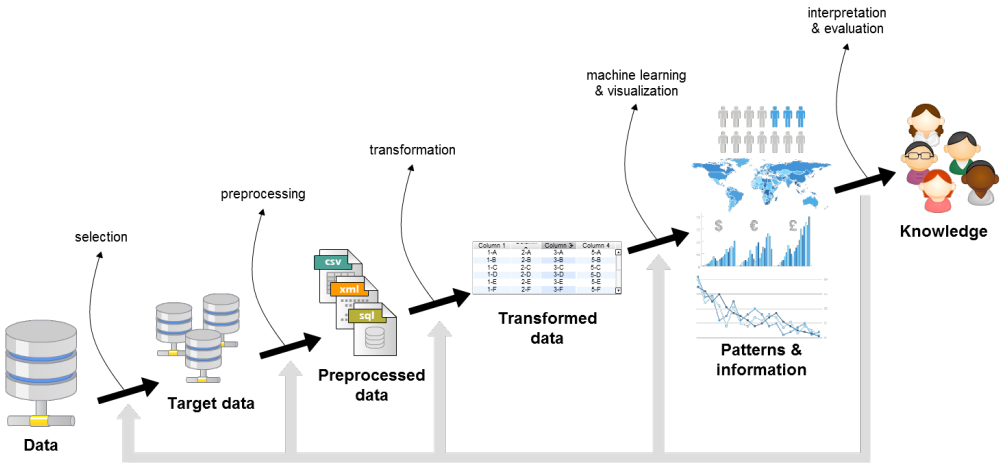

FIGURE 1.1: Data mining as a *process*. Adapted from Fayyad et al. [1]

1.1.1.1 Facing the limits

Despite lots of efforts are put into enhancing different data mining processes, there still are many cases where these techniques fail to perform well, mainly because of the scale of the problems that we face nowadays.

On the one hand, traditional data mining workflows cannot cope with the really massive data sets that are available nowadays, if performed on a common infrastructure. To solve this issue, clusters of hundreds or thousands of computers are used to run such analysis. It is costly and complex but, doing so, we can mine amounts of data in a way that was unthinkable some time ago.

On the other hand, we face another type of scaling problem. In some situations, data acquisition throughput is so high that it cannot be stored anyway, so another approach is needed to avoid this loss of information, because we still want to analyze it to extract knowledge from it. Both these scenarios are addressed with a series of techniques known as *stream mining*.

1.1.1.2 Stream mining

Stream mining or *data stream mining* is a process that allows us to still discover knowledge and patterns in data, even when it comes in the form of a continuous stream, or many of them [2]. Instead of processing all statically stored data, like traditional data mining does, a relatively small portion of it is kept during the analysis, and it is updated when needed - either because more resources are available to the system or because new data is acquired. A more deeper review of this research area is given in 2.1.

1.1.2 Privacy

Privacy is a concept that can be defined as the ability of an individual or group to seclude[1] themselves, or information about themselves, and thereby express themselves selectively. It is understood differently depending on the social and cultural background of each individual, but it is in fact recognised as one of the most fundamental rights of our human nature. The Universal Declaration of Human Rights' 12th article [4] states that:

> No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks.

This right has been continuously violated ever since information exchange and advanced communication technologies have been developed. Despite this did not begin with the spread of the Internet, its adoption has greatly magnified both the ability to breach people's privacy and the impact that these breaches have. A more thorough analysis of privacy and its interrelations with society and technology is given in Section 4.1.

1.1.3 Privacy Preserving Data Mining

Nowadays, data mining technologies have become a relevant debate topic, concerning what information is collected from individuals, who owns it and what are the purposes behind its gathering. Information technologies deliver us many benefits at many levels - safer streets, cheaper communications, better health systems, more convenient shopping - but at the high cost of losing our privacy.

Knowledge discovery processes need data to work and, in most cases, it is sensitive and personal. Moreover, it is massively collected, stored and analyzed without the users consent. Besides this lack of consent in the data acquisition stage of the process, data mining poses a bigger thread on individuals: information disclosure. Sensitive data must be treated accordingly, which involves not only good IT security practices to avoid information leaks, but a responsible treatment when research results are published.

1.1.3.1 Statistical Disclosure Control

Statistical Disclosure Control (SDC) is the name that the statistical community has given to what the data mining community calls Privacy Preserving Data Mining (PPDM). This

[1] "Seclusion is the act of placing or keeping someone away from other people." [3]

field, whatever its preferred name is, deals with controlling that information about specific individuals is not extracted from statistical summary results. Also, if full datasets are to be released, SDC methods should be applied to data in order to preserve user's privacy, whilst maintaining the statistical significance of it, i. e., the amount of information - knowledge - that this data can provide.

1.2 The project: moa-ppsm

Having reviewed the main concepts to which this project is related to, we can now outline its main purpose, once we take a closer look to the technical environment in which it will be developed.

1.2.1 MOA

MOA, initials for **M**assive **O**nline **A**nalysis, is an open source framework for data stream mining [5], originally developed at the University of Waikato, New Zealand. It includes several machine learning algorithms[2] to perform the analysis and tools to evaluate the quality of the results. It also deals with a problem known as *concept drift*[3]. It is related to the well known and commonly used Weka[4] package, but it is built to perform at a greater scale for more demanding problems.

FIGURE 1.2: Massive Online Analysis logo.

1.2.1.1 MOA filters

One of the available features in MOA is the use of *filters*, which can process streaming data before or after being fed to other systems or algorithms, such as learners or file

[2]Algorithms used to perform the actual data mining analysis (the "machine learning & visualization" step on Figure 1.1) belong to the field of machine learning. In MOA, clustering, classification, regression, outlier detection and recommender systems are available.

[3]It is said of statistical properties of a target variable being analyed, when they change over time in unforeseen ways.

[4]Weka is a popular software package including classical data mining algorithms, this is, not stream mining. It is also developed at the University of Waikato. [6]

writers. However, few filters are currently shipped within the latest MOA distribution, namely a filter to replace *missing values*[5] and a filter that adds noise to data.

1.2.1.2 MOA extensions

When working with MOA, the environment consists of the core library, but *extensions* can be used to enhance the existing methods or to provide additional features, based on the core tools that MOA already provides. A series of extensions have been developed and can be found on MOA's website, at `http://moa.cms.waikato.ac.nz/moa-extensions`.

1.2.2 The project in a nutshell

Summing up, the aim of this project is to **implement privacy preserving filters for the MOA stream mining framework**. This is, adapt some well-known SDC methods to a stream mining environment and, more precisely, to the MOA software framework, in the form of a MOA extension.

1.3 Report structure

The structure of this report gives an overview of the development process of the project, from the theoretical foundations that are necessary to understand the work to the final results and conclusions.

Chapter 2 covers the theory basis behind the SDC methods implemented in the project and provides some insights on different stream mining approaches. Chapter 3 discusses state of the art solutions concerning SDC for static databases (not streaming data). Chapter 4 analyzes more thoroughly the motivation behind privacy-preserving data mining by discussing *practical* questions like the relationship between society and privacy or the legal framework that applies to the context of this project. Project management is layed out in Chapter 5 and then the report turns to more technical related topics, such as implementation details and desgin decisions, covered in Chapter 6, as well as benchmarking, in Chapter 7. Finally, the report and project conclusions are given in Chapter 8, covering both achievements and possible future work.

[5] In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Chapter 2

Theoretical framework

Before reviewing the state of the art and compare existing solutions to this project, to better understand the contribution of this work, we will provide now an introductory overview of the theoretical concepts related to this project on *stream mining* and *privacy preserving* data mining mechanisms.

2.1 Stream Mining

Data stream mining is a relatively new field. Even though its theoretical foundation is based in well-established statistical and computational approaches, it has not been until recent years that this research area has experimented a great growth in interest.

The main problem when dealing with streaming data is the high throughput of data being analyzed, under computational resources constraints. Variable data rates is another problem that has to be addressed too. Once these problems are resolved, the same kind of data mining analysis as in the case of batch data processing are available: classification, regression or clustering tasks, as well as outlier detection and recommendation systems. We will not cover these techniques here, because they are not related to this project, by themselves. Instead, we will have a look at some different stream mining solutions, because their working principles do affect the way the project's algorithms will be implemented.

2.1.1 Stream mining approaches

Solutions provided in this field can be categorized into *data-based* and *task-based* ones [7], depending on their approach.

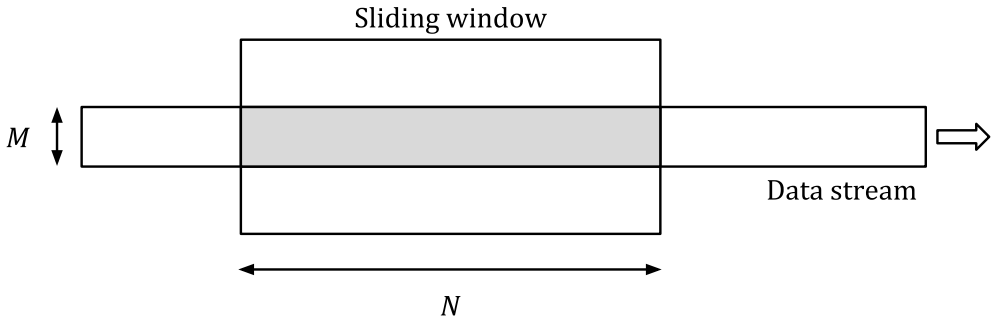

FIGURE 2.1: Processing a data stream using a *sliding window* approach. In the figure, N is the *size* of the sliding window, in terms of the number of samples of the stream being stored, whereas M is the number of *attributes* of the samples in the stream.

Data-based stream mining solutions

The idea behind these solutions is to use a subset of the original dataset to perform the required analyses. Diverse techniques that have been used in this sense can further be split into two more categories:

- **Sampling methods:** either by randomly picking samples of the data stream or by randomly selecting chunks (subsets) of the stream, sampling methods discard part of the incoming data, while performing the knowledge discovery processes with the sampled data. The main problem with this approach is that is hard to know when to pick a sample or which records should be stored, because there is no previous knowledge of the dataset size or its information structure.
- **Summarizing methods:** they use aggregated data or calculated statistical measures (that are continuously recalculated) to provide the information needed for the data mining algorithms. In this case, it is the loss of information and accuracy and the inability to control data distribution fluctuations what renders these methods not so usable as it was desired.

Task-based stream mining solutions

The solutions that fall into this category are based not on performing data transformations, but on changing the data mining methods to enable their use on data streams.

- **Approximation algorithms:** these are a kind of algorithms that are designed to solve computationally hard problems, by giving an approximate result. Instead of computing exact solutions, they just guarantee a certain error bound. The problem with these methods is, again, the high received data throughput, which

they cannot cope as well. Additional tooling is therefore needed if one wishes to use them.

- **Sliding window method:** this method, a common pattern in many online[1] applications, maintains a *sliding window* in which the most recent data is kept. As data is received from the incoming streams, this window "advances" so new observations are kept inside, as can be seen in Figure 2.1. The data mining analyses are then performed using the data available inside the window and summarized versions of the older records, in the form of statistical measures or aggregated data. This particular method is the one that the MOA package uses - thus its name: Massive **Online** Analysis. This solution scheme enables dealing with concept drift, which would not be possible if just aggregated data was used.

- **Algorithm output granularity:** this method is a resource-aware data analysis approach that can perform the local analysis on resource constrained devices, by adapting to resource availability and data stream rates - when resources are completely running out, the results are merged and stored.

2.2 Statistical Disclosure Control

As was already introduced in Section 1.1.3, the purpose of Statistical Disclosure Control (SDC) is to prevent confidential information from being linked to specific individuals to whom this data belongs. We will review now some concepts related to data disclosure and SDC methods and some theoretical foundations.

2.2.1 Privacy preserving algorithms

As a quick and superficial review, the algorithms[2] being used nowadays to achieve effective privacy preserving in datasets can be categorized into the following groups [8]:

- *Non-perturbative data masking:* these kind of methods do not perform data values transformations. Instead, they are based in partial suppressions of records or reductions of detail of the datasets. Some examples are:
  - Sampling
  - Global recoding

[1] In computer science, an *online algorithm* is one that can process its input piece-by-piece in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the start.

[2] We will not cover every algorithm in detail, because some of them are not included in the scope of this project.

  - Top and bottom coding
  - Local suppression

- *Perturbative data masking:* these methods do release the whole dataset, if required, but it is perturbed, this is, values are changed by adding them noise. This way, records are diffused and reidentifying individuals is harder. Some examples are:
  - Noise masking
  - Micro-aggregation
  - Rank swapping
  - Data shuffling
  - Rounding
  - Re-sampling
  - PRAM
  - MASSC

2.2.2 Definitions of disclosure

When assessing the disclosure risks of a given dataset (or data stream) we must have a look at the different kind of variables this data is composed of. We will stick to a classic [9] categorization of such attributes into three groups, which need not be disjunctive, as follows:

- **Identifiers:** variables that precisely identify individuals, e.g., social insurance numbers, person names, or addresses.
- **Quasi-identifiers:** a set of variables that, when considered together, can be used to identify individual units. It might be possible to, for example, identify people by combining variables such as gender, age, region and occupation.
- **Non-identifying variables:** these are neither *identifiers* nor *quasi-identifiers*.

Concerning *disclosure*, it is also defined differently depending on the type of privacy breach that has occurred:

- We talk about *identity disclosure* when a specific individual record can be recognised in a dataset, i.e., when linkage with external available data is possible. Identity disclosure is performed using direct identifiers, rare combinations of values in quasi-identifier attributes and exact knowledge of variable values in external databases.

- In the case of *attribute disclosure*, the intruder is able to gather sensitive information about a specific unit from the released data, where it is directly available. For example, if no perturbation is applied to the original values of the *wages* variable, one could learn how much a person is earning if its identity is disclosed too.
- *Inferential disclosure*, the most general case, occurs when an intruder is able to, with some uncertainty, predict or *infer* confidential information about an individual from the statistical properties of data.

It is important to remark that a subset of critical variables might be exploited to disclose every information about a single unit in a dataset. Thus, we are bound to carefully select which variables of the dataset might be released to further users of the data, while trying to maximize its statistical utility. More concretely, it is extremely important to **not release identifiers** and to analyze quasi-identifiers closely, in order to avoid information leaks and privacy breaches.

2.2.3 Disclosure Risk

Concerning the safety of the released data, **Disclosure Risk** (DR) is a common way to measure and assess the risk of re-identification of particular individuals. Re-identification happens when some sensitive and confidential data that have been released are subsequently linked to a particular individual, which results in a confidentiality breach. There are a number of different approaches in how to assess disclosure risk and whether to measure it *per record* or globally, taking into account the whole dataset.

As noted in [10], there is not much literature on disclosure risk that can be used for a broad class of perturbative methods; disclosure risk measures tend instead to be method-specific. Therefore, empirical methods are most used to assess disclosure risk for these kind of methods.

2.2.3.1 Record linkage

Most notably, the mechanisms used to measure disclosure risk follow a *record linkage* approach. This is, after an SDC method has been used to anonymize data, a record linkage procedure is applied to the original and released (masked, anonymized) datasets. This *linkage* attempts to identify, for each record in the masked dataset, which is the corresponding record in the original dataset. If such correspondance is verified, the record is labeled as *correctly linked*. A generic measure for disclosure risk is the percentage of correctly linked records from the total amount in the dataset.

- **Distance-based record linkage:** provided that a *distance* measure can be defined between the original and the masked datasets, linkage is performed as follows: for each record in the anonymized dataset, a distance to each record in the original dataset is calculated. The nearest record, in terms of this distance measurement, is assumed to be the corresponding record, thus establishing a *link* between them. This linkage is then verified to assess how many of these guesses are true re-identifications.

- **Probabilistic record linkage:** in this case, the matching algoritm works a little different. For each possible pair of original and masked records, a *coincidence vector* is defined. This vector holds, for each attribute, whether or not the values of the considered records are equal. An index is computed afterwards over these vectors and, using such index, the records pairs are classified as *linked* or *not linked*. Again, this linkage is verified to assess the number of true re-identifications.

2.2.4 Information Loss

Another key measurement concerning data protection is **Information Loss** (IL) or *data utility*, which could be defined as the amount of useful statistical information that is lost along the data masking process. A good SDC method should try to minimize IL, in order to provide optimally useful data to the legitimate users of such data, while also keeping a low disclosure risk. It is important to note that these two properties are inversely proportional: the lower disclosure risk is, the higher information loss will occur. This trade off between these two parameters is often a difficult and challenging task and should be taken into very careful consideration, depending on the release policies that apply, the kind of data being released and the sensitivity of the information contained in such data. This evaluation should be performed not only from a purely quantitave and numerical point of view, but from an ethical and privacy concerned one too.

As well as with disclosure risk, a number of methods and approaches are taken to assess information loss when releasing privacy protected datasets, ranging from unbounded [11] to probabilistic (bound to the $[0, 1]$ interval) measurements [12].

Unbounded Information Loss

An example framework to assess IL was given in [11], which evaluates some key statistical properties of the released data. More concretely, it computes three *discrepancy* measurements for a series of pairs of matrices (correlation, covariance, etc. of the original and masked datasets), namely the *mean square error*, the *mean absolute error* and the *mean variation*.

Probabilistic Information Loss

The aim of measuring IL in a probabilistic manner is to bound this measurement to the $[0, 1]$ range, thus allowing its comparison with DR, which is also generally expressed within this range. This way, a *score* could be calculated from both normalized measures for an SDC method, easing parameters selection to data protectors, for example.

2.2.5 Privacy guarantees

Many different methods have been developed to help prevent information disclosure when data mining datasets or results are released. These algorithms pursue the generation of results or data that have particular properties concerning privacy preservation. Some of the desirable properties of privacy-protected data are described in the following sections, but no formal definition is provided for some of them (please refer to the original papers and publications to understand them better).

2.2.5.1 *k*-Anonymity

First described in 2002, by Latanya Sweeney, a release of data is said to have the *k-anonymity* property if the information for each person contained in the release cannot be distinguished from at least $k - 1$ individuals whose information also appears in the release [13]. A more formal definition uses the previously reviewed concept of *quasi-identifiers* (see Section 2.2.2).

Definition 2.1. (k-Anonymity)
A dataset is said to satisfy k-anonymity for an integer $k > 1$ if, for each combination of values of quasi-identifiers, at least k records exist in the dataset sharing that combination. [14]

An intruder trying to use a k-anonymous dataset to do, for example, record linkage against an external source of information will find that at least k records in the dataset match any value of the quasi-identifiers that he or she is trying to use to perform the linkage. Thus, re-identification is limited to *groups*, this is, no individual records can be linked, just groups of size at least k.

2.2.5.2 *l*-Diversity

The evolution of the concept of k-anonymity is *l-diversity* and adds further privacy preservation by adding intra-group diversity, so to avoid the flaws of the k-anonymity privacy model [15].

2.2.5.3 *t*-Closeness

Further on, the *t-closeness* property definition adds attribute-based privacy enforcement to the *l*-diversity model: to better preserve privacy, all values (all observations) from a particular attribute must not be too much different - instead, they should be close up to a certain threshold [16]. This is needed to preserve the privacy of those records that are more easily identifiable because their attribute values are more distinguishable.

2.2.5.4 Differential Privacy

Described in Dwork [17], *differential privacy* is a condition *on the release mechanism* (not the dataset) that guarantees a strong privacy preservation level for some particular data uses contexts. Differential privacy is introduced in an *interactive* setting, i.e., in a query-response data retrieval environment, and offers probabilistic guarantees that the contribution of any single individual to thenquery response is limited.

Definition 2.2. (ε-Differential privacy)
A randomized mechanism[3] $\mathcal{M}$ gives ε-differential privacy if, for all datasets X_1, X_2 such that one can be obtained from the other by modifying a *single* record, and all $S \subset Range(\mathcal{M})$, it holds

$$P(\mathcal{M}(X_1) \in S) \leq \exp(\varepsilon) \times P(\mathcal{M}(X_2) \in S) \tag{2.1}$$

This definition, cited from Soria-Comas et al. [14], easier to understand than the original one given in Dwork [17], states that, given an ε-differential privacy mechanism $\mathcal{M}$ and any possible output r, the presence or abscence of a participant (in terms of the dataset, a *row*) will cause at most a multiplicative e^{ε} change in the probability of the mechanism to output a response r.

2.3 SDC methods

We will describe now some of the most common methods and mechanisms used in SDC applications to anonymize data or provide privacy preserving data releases.

Notation

We assume the following notation for the subsequent method descriptions:

[3] By *mechanism*, we refer to any kind of function or system used to query for data.

- The original dataset is the matrix X, with n rows (samples) and m attributes or variables. Therefore, the x_{ij} element of the dataset denotes the value that the j-th attribute takes in the i-th row for any $1 \leq i \leq n$ and $1 \leq j \leq m$.
- The anonymized (protected) dataset is named X'.

2.3.1 Noise Addition

Noise addition or *additive noise masking* is a fairly simple method that is based on the addition of gaussian noise to data, thus randomly distorting its values and difficulting re-identification of individuals. The main additive noise algorithms in the literature are [8, p. 54]:

- Uncorrelated noise addition.
- Correlated noise addition.
- Noise addition and linear transformation.
- Noise addition and non-linear transformation.

We will only cover the first couple of methods, because of the inherent difficulty of the latter, both in its theoretical basis and its practical implementation, which renders them not suitable for the needs of this project.

2.3.1.1 Uncorrelated noise addition

Masking by additive noise the j-th variable of an original dataset X yields an anonymized dataset X' such that

$$x'_{ij} = x_{ij} + \epsilon \quad \text{for } 1 \leq i \leq n \tag{2.2}$$

where ϵ is drawn from a random variable $\varepsilon_j \sim N(0, \sigma^2_{\varepsilon_j})$. The general assumption is that the variances of each ε_j are proportional to those of the original variables, this is, if $\text{Var}(X_j) = \sigma^2_j$ is the variance of the j-th attibute of the dataset X, then $\sigma^2_{\varepsilon_j} := \alpha\sigma^2_j$.

While this method preserves means and covariances, it is, unfortunately, not able to preserve variances nor correlation coefficients.

2.3.1.2 Correlated noise addition

This method is aimed to also preserve correlation coefficients, with respect to *uncorrelated* noise addition. The main difference with the previous mechanism is that the covariance matrix of the errors is now proportional to the covariance matrix of the data: $\varepsilon \sim N(0, \Sigma_\varepsilon)$, where $\Sigma_\varepsilon = \alpha\Sigma$.

Masking by correlated noise addition provides data with higher analytical utility than masking using uncorrelated noise, as long as α is revealed to the data user. However, the low level of protection yielded by this method and the previous one render them as not very useful for truly important SDC applications.

2.3.2 Microaggregation

Originally described for continuous (numerical) data, microaggregation is a family of SDC methods that, in the most general form, consist of making homogeneous groups of k or more individuals (rows) from within the X dataset to later replace their values with aggregated ones, this is, averages, computed on the groups themselves. These grouped and aggregated records conform the resulting X' release dataset.

Two main approaches are taken when considering microaggregation techniques: *univariate* and *multivariate* microaggregation. The difference remains in the number of variables used to perform the *clustering* phase of the method: a single variable and multiple attributes, correspondingly. As can be assessed in the literature, the univariate approach causes either a very high information loss or a very high disclosure risk, thus not being appropriate for normal SDC uses [8, p. 63]. On the other hand, multivariate microaggregation, proposed by Domingo-Ferrer and Mateo-Sanz [18], is considered an excellent protection method and, as such, we will focus on this approach.

It is important to note that this family of techniques are directly related to k-anonymity, as proved in Domingo-Ferrer and Torra [19].

2.3.2.1 Partition

The first and most computational complex task to do in a microaggregation method is to partition the dataset into g groups of size at least $k > 1$, which is, indeed, a *clustering* task. This proves to be quite difficult, but an optimal solution approximation with respect to information loss was already given in [19] and further refined in Domingo-Ferrer et al. [20].

The aim of these partition methods is to find the optimal k-partition that maximizes within-group homogeneity. Following Domingo-Ferrer and Mateo-Sanz [18], a practical

information loss measure for microaggregation, relatively common in the clustering literature, is the ratio of within-group homogeneity over the total sum of squares (the sum of *within* and *between* group homogeneity)

$$L = \frac{SSE}{SST} \tag{2.3}$$

The within-group homogeneity (SSE) is defined as

$$SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (x_{ij} - \bar{\mathbf{x}}_\mathbf{i})^2 \tag{2.4}$$

where g denotes the total number of groups of n_i elements each and $\bar{\mathbf{x}}_\mathbf{i}$ denotes the i-th group centroid. The between-groups sum of squares, SSA, is

$$SSA = \sum_{i=1}^{g} n_i (\bar{\mathbf{x}}_\mathbf{i} - \bar{\mathbf{x}})^2 \tag{2.5}$$

where $\bar{\mathbf{x}}$ is the average vector over the whole dataset. The total sum of squares is, then, $SST = SSE + SSA$.

Because microaggregation replaces values in a group by the group centroid, if we recall Equation 2.3, it follows that the higher the within-group homogeneity, the lower the information loss is. Both the MDAV [19] (Maximum Distance to Average Vector) and μ-Approx [20] algorithms are built to partition the dataset into groups, while minimizing information loss, exploiting the previous theoretical result.

2.3.2.2 Aggregation

The aggregation step is the simplest of the ones that take place in a microaggregation setting: for each group g of at least k records and for each attribute $1 \leq j \leq m$, an *aggregate* γ is computed among the values of the j-th variable for the records in the group. This aggregate is then imputed to each record for its j-th attribute.

Concerning the types of variables that are aggregated [19]:

- **Continuous attributes:** the aggregated value correponds with the arithmetical mean of the selected values.
- **Categorical attributes:** the aggregated value should either be the median or the mode of the selected values.

2.3.3 Rank Swapping

Also a fairly simple SDC method, the basic idea behind data swapping and its refinement, *rank swapping*, is to transform a dataset by exchanging values of confidential variables in such a way that marginals are maintained. The method works as follows:

First, values of a variable j are ranked in ascending order, this is, they are *sorted*. Each ranked value is then swapped with another ranked value, randomly chosen within a restricted range. This range is controlled by an input parameter p, normally denoting that swapped values cannot differ more than $p\%$ of the total number of records. This procedure is applied for every variable in the dataset.

2.3.4 Laplace Mechanism

We recall now the context of differential privacy to discuss a relatively extended method that is designed to achieve this privacy preserving guarantee. However, this technique is restricted to a particular family of data release functions. More precisely, it can only be applied to functions that provide a *numerical* answer, like counting queries, for example. To understand this method, called *Laplace mechanism*, we must review first the concept of *global sensitivity* of a function.

Definition 2.3. (Neighbour datasets)
Given two datasets from a universe of datasets, $D_1, D_2 \in \mathcal{D}$, we call them *neighbours* if they differ in just one record, which we indicate using the notation $|D_1 \Delta D_2| = 1$.

Definition 2.4. (Global Sensitivity of a function)
We define the global sensitivity of a numerical function $f : \mathcal{D} \to \mathbb{R}^w$, with $w \in \mathbb{N}^+$, over the universe of datasets $\mathcal{D}$, as

$$\Delta(f) = \max_{\substack{D_1, D_2 \in \mathcal{D} \\ |D_1 \Delta D_2| = 1}} \|f(D_1) - f(D_2)\|_1 \tag{2.6}$$

As we will now see, the Laplace mechanism is just a noise addition masking method, where the sensitivity of the release function f drives the amount of noise being added to the response of f: the higher the sensitivity of the function, the higher the amount of noise added. If f is applied to a dataset D_1 and then to a neighboring dataset D_2, if f changes a lot, it means that we will have to add more noise to probably obtain the same output.

Definition 2.5. (Laplace mechanism)
Given a database $D \in \mathcal{D}$ and a function $f : \mathcal{D} \to \mathbb{R}^w$, with $w \in \mathbb{N}^+$ and global sensitivity

Δ, an ε-differential privacy mechanism $\mathcal{M}$ for releasing f is to publish

$$\mathcal{M}(D) = f(D) + L \tag{2.7}$$

where L is a vector of random variables each drawn from a Laplace distribution $Lap(0, \frac{\Delta(f)}{\varepsilon})$.

This mechanism ensures that ε-differential privacy is achieved for the release function f, as can be assessed in Leoni [21] and Soria-Comas et al. [14].

Laplace distribution

On a quick note to understand the kind of noise being added through the Laplace mechanism, a Laplace distribution $Lap(\mu, b)$ has *location* μ (which could be understood as the mean or the location of the *peak* of the PDF of the distribution) and *scale* parameter b. This last parameter is the one used to adjust the amount of perturbation the data or release method will receive.

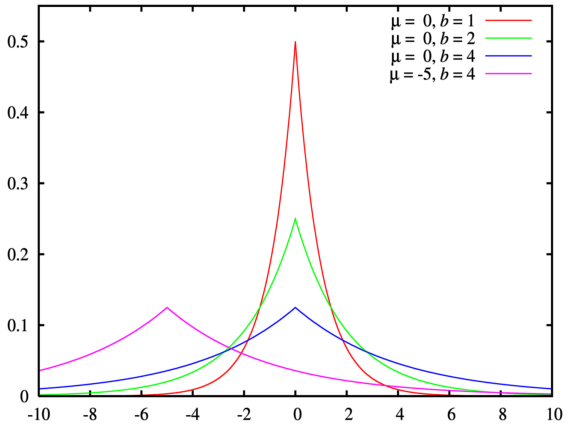

FIGURE 2.2: Laplace distribution probability density function (PDF). Source: Wikipedia [22]

The density function of the Laplace noise, also called *double exponential*, is

$$P(x|\mu, b) = \frac{1}{2b} \exp(-\frac{|x - \mu|}{b}) \tag{2.8}$$

Chapter 3

State of the art

This chapter gives further insights concerning the latest discoveries and cutting-edge technological solutions related to the main knowledge fields that affect this project: *data stream mining* and *statistical disclosure control*.

3.1 Stream mining software

Data stream mining is a relatively new field. Even though its theoretical foundation is based in well-established statistical and computational approaches, it has not been until recent years that this research area has experimented a great growth in interest Gaber [7].

Because it is an incipient field, stream mining software packages are quite uncommon. Even though specific applications have been developed (see Kargupta [41]), MOA remains as one of the few generic, free and open sourced systems. One example of a commercial solution that includes support for data stream mining is RapidMiner, through the use of plugins.

MOA is currently the most complete framework for data stream clustering research and it is an important pioneer in experimenting with data stream algorithms. MOA's advantages are that it interfaces with WEKA, provides already a set of data stream classification and clustering algorithms and it has a clear Java interface to add new algorithms or use the existing algorithms in other applications.

Related to MOA, a new project called SAMOA (from Scalable Advanced Massive Online Analysis) is being developed too, based on MOA itself and a couple of streaming processing engines: Apache S4 [46] and Apache Storm [47], developed by the Apache Software Foundation.

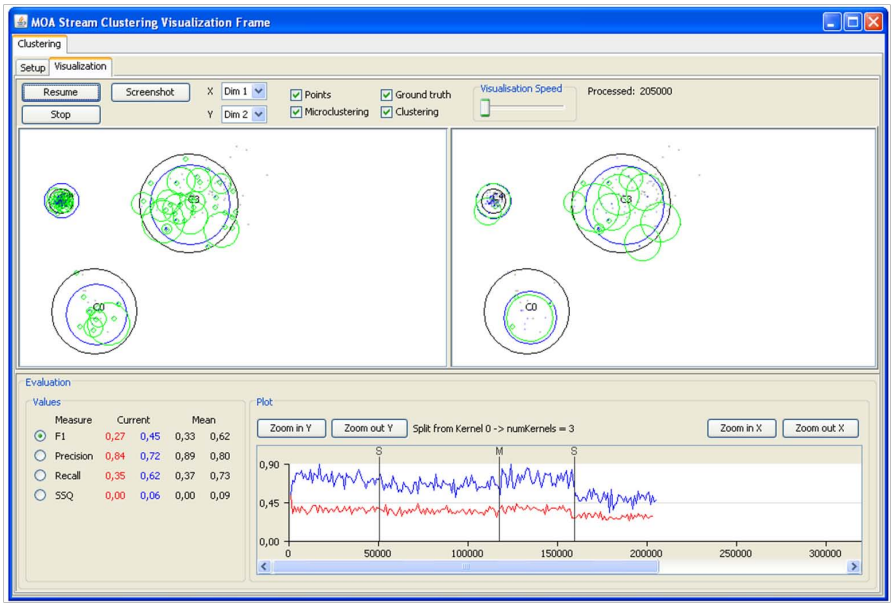

FIGURE 3.1: MOA's Graphical User Interface, showing the clustering visualization capabilities of the software.

Finally, an R package called `stream` was released into the CRAN repository[1] in 2013. It allows to do real time analytics on data streams and is currently focused on clustering algorithms available in MOA.

3.1.1 The MOA framework

Massive Online Analysis (MOA) is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. MOA is designed to deal with the challenging problems of scaling up the implementation of state of the art algorithms to real world dataset sizes and of making algorithms comparable in benchmark streaming settings.

MOA contains a collection of offline and online algorithms for both classification and clustering as well as tools for evaluation. Researchers benefit from MOA by getting insights into workings and problems of different approaches, practitioners can easily compare several algorithms and apply them to real world data sets and settings.

MOA supports bi-directional interaction with WEKA, the Waikato Environment for Knowledge Analysis, which is an award-winning open-source workbench containing implementations of a wide range of batch machine learning methods. WEKA is also written in Java. The main benefits of Java are portability, where applications can be run on any

[1]The capabilities of the R language are extended through user-created packages. Most of these packages are available at the Comprehensive R Archive Network (CRAN), on the following web address: `http://cran.r-project.org`.

platform with an appropriate Java virtual machine, and the strong and well-developed support libraries. Use of the language is widespread, and features such as the automatic garbage collection help to reduce programmer burden and error.

The MOA framework provides a graphical user interface (GUI), which eases its use, when experiments can be carried out using the algorithms already included in the framework. However, for more complicated analysis or industry-scaled uses, MOA offers the possibility to be used using a command line interface, which is extremely powerful and flexible. Also, due to its open source nature and the fact that it is built in Java, custom procedures and integration techniques can be developed to meet the data analysis requirements. Last but not least, when the core features are not sufficient for the user's needs, MOA can be extended with new mining algorithms, new stream generators or evaluation measures, like the SDC filters that we will implement in this project.

3.2 Statistical Disclosure Control software

With the advent of new technologies and the Internet widespread, concepts like Open Data[2] are beginning to arise. Information exchange within the Internet is a very powerful way to share knowledge and allow others — researchers, statistical agencies and any other user — get insights from the analysis of this data. However, the released data must be protected against disclosure attacks, to enhance data owners privacy.

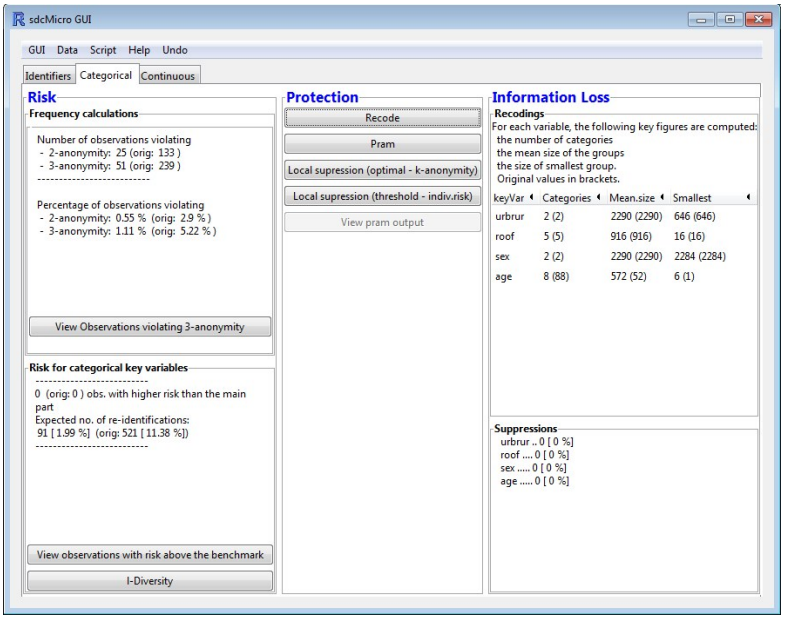

FIGURE 3.2: `sdcMicro` package graphical user interface.

[2]Open data is the idea that certain data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control.

Some software suites have been developed to provide the SDC tools needed to effectively anonymize the released datasets. The most prominent of these is the `sdcMicro` package. It is a free R-based open source suite for the generation of protected data for researchers and public use. It can be used for the generation of anonymized data, i.e. for the creation of public and scientific-use files. In addition, various risk estimation methods are included. Moreover, the `sdcMicro` package it is bundled with a graphical user interface for some of the SDC methods it offers.

The `sdcMicro` includes all the methods of another popular software, called μ-ARGUS[3], along with some more new methods. A series of documents can be found on the official website of the package concerning its usage.

[3]The μ-ARGUS suite can be found on `http://neon.vb.cbs.nl/casc/mu.htm`, but it seems to be quite an outdated software.

Chapter 4

Practical aspects

This chapter addresses the *practical* aspects of this project, this is, those that are related to the *praxis*[1], rather than to technology or theory. An analysis of the concept of privacy and the need to protect it is given in the first section of the chapter, followed by a short review of the legal framework that applies to this project and, finally, a brief note on the environmental impact of the present work.

4.1 Privacy & society

Privacy has become a hot topic in debates nowadays, concerning *what* information is collected from individuals, *who* owns it and with *which* purposes. It is a matter of great importance and certainly worth to be examined carefully. Information technologies have brought us many benefits at many levels — safer streets, cheaper communications, better health systems, more convenient shopping — but many times at the high cost of losing our privacy. With the rapid adoption of the Internet and all sorts of digital telecommunications as the basis of our modern communication relationships, a vast capacity of interception, storage and analysis of such information exchanges has been reached. This potential has been used by companies in the private sector to, for example, analyze the population consuming profiles, target marketing campaigns more accurately and offer much more customized products and services. In order to apply these techniques and mechanisms, corporations collect private data from users, excusing that these same users accept privacy terms and conditions. It seems clear that data mining is highly related to privacy: knowledge discovery processes need data to work and, in most cases, sensitive personal data is at stake.

[1] *Praxis* is the process by which a theory, lesson, or skill is enacted, embodied, or realised. *Praxis* may also refer to the act of engaging, applying, exercising, realizing, or practicing ideas.

We have already outlined in Section 2.2 that the aim of SDC and this project in particular is to protect users privacy by avoiding information disclosure from released datasets and real time analysis processes that require sensitive data. The question, however, is: why do we *need* to protect privacy? What urges us to preserve our right to privacy? It is not a simple and mere question; indeed, the answer is related to our understanding and interpretation of the term "privacy" itself. Therefore, we will review the definition of privacy and provide an argument that is the basis to justify privacy protection.

In the introductory chapter of the report (see Section 1.1.2) an introduction to the concept of privacy was given by literally reproducing a dictionary definition: "*Privacy is a concept that can be defined as the ability of an individual or group to seclude themselves, or information about themselves, and thereby express themselves selectively*". We also saw that privacy is recognised as one of our most fundamental rights, as it is enshrined in the Universal Declaration of Human Rights. Going further on, *privacy*, understood not only as the mechanism that allows us to keep our opinion and ideas private, but also the rest of our *praxis*, enables us to develop a *particular* personality, yet when we are within a social structure. Without the right to keep certain aspects of our life private, the *individuation* process is compromised and many consequences of this individual diversity are endangered — thought heterogeneity, for example, cultural heritage and, above all, individuals *emancipation*, all because the individuation process does not happen in a context of complete freedom.

We must not forget that when organizations such as enterprises or governments acquire massive amounts of private information about particular individuals, a certain *control* capacity on these individuals is gained too. This power, on the contrary of what ultimate defendants of data gathering hold, does not liberate people nor make them safer. The true consequence of such an increase in control power is that all equitable bonds between individuals and these organisms are torn apart: people become *dominated* by social institutions, be them governments or any kind of structured association, and their freedom is, thus, canceled. There is no possible emancipation nor conviviality of people in a social context if the individuals-society relationships are domain based.

Finally, from a more pragmatic point of view, not only ethical concerns are addressed by protecting users privacy, but economical issues too. Industrial-scale information theft has a huge impact on enterprise economies, because of distrust and because disclosed sensitive data can be used to make profit of it. Identity theft, for example, was estimated to have a cost in the order of billions of dollars, back in 2005, as shown by Romanosky [23].

4.1.1 Impact of this project

The motivation of the project is now well-founded: privacy is a relevant concern for any data analysis related field, whether it is statistics, data mining or data stream processing. Of course, this project addresses just a small portion of the broader picture of privacy protection, but it is indeed another effort taken towards its effective achievement.

Together with good IT security practices, a reasonable usage of data and information and ackowledged consent from the data owners, the application of SDC techniques — like the ones implemented by the privacy filters which conform the goal of this project — enables the preservation of the inalienable right to privacy.

To provide further examples of the impact of the project, potential users of the MOA privacy filters are both companies and government statistical agencies, which handle vasts amounts of sensitive and personal data. Using SDC methods, they would be able to exploit the intrinsic knowledge of these data, while preserving privacy and protecting their users against disclosure attacks. Not only they could carry more interesting experiments, but they could also release this information, sharing it with third parties to promote collaboration with researchers and, last but not least, as an exercise of transparency.

4.2 Legal framework

One of the aspects to bear in mind when developing a technological project is the legal environment in which it is framed. To this respect, efforts are being carried out to develop legal frameworks to help protect people's privacy, at many levels. One such example is the spanish LOPD[2], a law that aims, among other things, to define different data privacy levels and mandatory proceedings associated to each - no matter the medium used to transfer it or store it. The full text of the law can be consulted at the BOE [24].

There are some pitfalls to these legislative efforts, though. Firstly, it is really hard to assess their accomplishment in the IT sector and, thus, it is sometimes a matter of confidence in the developer's good practice. Another important drawback is that online services, such as social networks, can be accessed globally, but, on the other hand, their legislative framework is that of the country to which the backing company offering the

[2]LOPD stands for L*ey* O*rgánica* de P*rotección* de D*atos*, a law that was approved by the spanish courts in 1999. It has been modified several times, being the law enforcement regulation approved in 2007.

service belongs to - jurisdiction definition in the Internet is still a matter of intense debate nowadays[3].

We have not detected any kind of legal consequences or regulations bound to this project's development, besides intellectual property protection measures — no personal data has yet been used to perform any benchmarking process nor to assess the quality of the developed methods: random data generators are being used instead (see Chapter 7).

As we already stated before, concerning the code base of the project, we must implement all necessary copyright protection mechanisms. Because this is an *open source* project, an internationally recognised software license is included in the public code repository, hosted at GitHub[4]. The chosen license is the MIT License, which has proven to be easy to understand, relatively widespread and quite permissive in terms of its commercial applicability.

4.3 Environmental issues

No relevant direct environmental impact is related to this project, neither tied to its development nor its further deployment. No use of massive resources is done and the results of the work do not, presumably, result in a significant environmental change of any kind.

It is still true, however, that *data mining* as a discipline, does consume a lot of resources, in terms of technological infrastructure and energy. We cannot forget that collecting, storing and processing data at the scale that we have reached needs entire data centers fully dedicated to the data mining process. Power consumption is a big concern with nowadays information technology, as it is the huge amount of rare materials that electronic devices contain. These highlights are indeed indirect effects of the data mining process.

[3]Proof of this debate is the emergence of initiatives like the Internet & Jurisdiction Project, which was launched in 2012 to address the tension between the cross-border nature of the Internet and the patchwork of national jurisdictions. To enable the digital coexistence of different norms in shared cross-border online spaces, it facilitates a neutral multi-stakeholder dialogue process, which brings together governments, civil society groups, major Internet platforms, technical operators and international organizations [25].

[4]The project is available at `https://github.com/necavit/moa-ppsm`.

Chapter 5

Project management

This chapter discusses all the aspects concerning the management of the project: *scope*, *schedule* and *budget*. However, we must stress that this classical approach of management analysis is not really suited for our needs. Instead, a more *Agile*[1] methodology will be applied. We cover this on the Methodology section, but there is an important conceptual change to be taken into account: the different driving force of the project. Whereas in classical project management the scope-schedule-budget triad is what must be controlled, in an Agile project management approach it is *value*. Indeed, *quality* must be ensured so maximum value is delivered to the project's stakeholders, thus being scope, cost and schedule just *secondary* constraints to these primary goals.

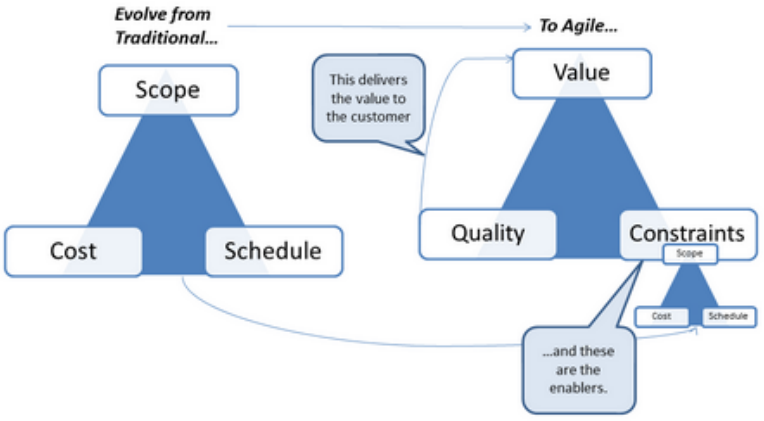

FIGURE 5.1: Traditional to Agile project management evolution. Source: Agile Australia - Opening keynotes [27]

[1] Agile software development is based on the *Agile manifesto* [26].

5.1 Goals & scope

One of the first things to do when beginning any project is delimiting its **scope**, this is, deciding *what* will be done and *how*, in terms of resources and methodology.

We already stated in Section 1.2 what the main *goal* of this project is:

Main goal:

Implement privacy preserving filters for the Massive Online Analysis (MOA) stream mining framework.

5.1.1 Requirements analysis

For the sake of completeness and verbosity, a more detailed list of the project's *requirements* is given in the next couple of sections, categorized into *functional*[2] and *non-functional*[3] ones. Together, they comprise the formal scope of the project.

Functional requirements

R1 Implement privacy preserving stream mining *filters*[4] for the MOA stream mining framework. The *suggested* algorithms to be implemented correspond with the following requirements:

- **R1-1** Noise addition [8, p. 54]
- **R1-2** Multiplicative noise [8, p. 57]
- **R1-3** Microaggregation [8, p. 60]
- **R1-4** Rank swapping [8, p. 73]
- **R1-5** Differential privacy [17]

R2 Evaluate technological alternatives prior to the implementation of the privacy filters.

R3 Benchmark the performance of the filters in terms of *disclosure risk* and *information loss*.

[2]Functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished.

[3]Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors.

[4]Within the MOA context, *filters* are procedures applied to data prior to their analysis using machine learning algorithms.

Non-functional requirements

NFR1 *Correctness:* privacy protection is at stake in this project, so algorithms must be implemented correctly, from the theoretical point of view, in order to not ease information disclosure when they are used.

NFR2 *Efficiency:* given that no data mining process can scale well if its algorithms are slow, effort will be put in making them the most efficient we can.

NFR3 *Test coverage:* measures and tests will be performed to assess the quality of the developed software, as well as its scalability and performance, which is paramount in this project's context.

NFR4 *Documentation:* MOA is an *open source* data mining framework, which means that its community can assess how is it built and how to improve it. One of the benefits of the open source development model is that software can be safer, more robust and efficient, by receiving contributions from different developers. If people are to continue improving the work done, it has to be well documented.

5.1.2 Scope deviations

There have been no major changes in the scope of the project along its development. Both the functional and non-functional requirements sets remain the same as the ones defined in the final report of the Project Management course (and also listed above).

However, concerning its completion, we have to admit that not all requirements have been achieved. We provide now an enumeration of the functional requirements and their final status:

R1 [Mostly completed] Implement privacy preserving stream mining *filters* for the MOA stream mining framework.

- **R1-1** [Completed] Noise addition [8, p. 54]
- **R1-2** [Not completed] Multiplicative noise [8, p. 57]
- **R1-3** [Completed] Microaggregation [8, p. 60]
- **R1-4** [Completed] Rank swapping [8, p. 73]
- **R1-5** [Completed] Differential privacy [17]

R2 [Completed] Evaluate technological alternatives prior to the implementation of the privacy filters.

R3 [Completed] Benchmark the performance of the filters in terms of *disclosure risk* and *information loss*.

Even though the **R1-2** requirement could not be finished, and further work would be possible, as will be discussed in the Conclusions section, the Agile approach for this project has enabled us to avoid a sense of failure at the end of its development.

5.2 Methodology

The methodology approach used in this project will be based on Agile principles. Some of the key concepts and practices related to Agile software development are:

- **Iterative** development versus the classical *waterfall* development model.
- Short to mid range development **sprints** (phases), in order to keep track of the project's evolution and to be able to react to changes, unforeseen constraints or scope drifts.
- **Constant meetings** with the project's stakeholders, in which the progress and deviations of the project are assessed.
- Usage of **burndown charts** - a graphical model of work left to do versus time - and other visual representations of the project's track
- Reduced documentation generation, to alleviate the potential loss of time that changes in the requirements would cause.

Among many other approaches and Agile methodological frameworks, *Scrum* is one of the most well-known due its flexibility, its proved resilience against requirements rapid changes and easy adoption by software development teams.

5.2.1 Scrum

Scrum is an iterative and incremental agile software development methodology for managing product development. It challenges assumptions of the *traditional, sequential approach* to product development, and enables teams to self-organize by encouraging physical location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project [28].

This methodology is based on the adoption of certain roles, as well as some *artifacts* and predefined processes, all of which can be adapted as necessary by the team to suit their specific needs and resources. However, a central concept forms the basis for the rest of the framework: the **sprint**. A sprint or iteration is the basic unit of development in Scrum. The sprint is a timeboxed effort, this is, it is restricted to a specific duration, which is fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common.

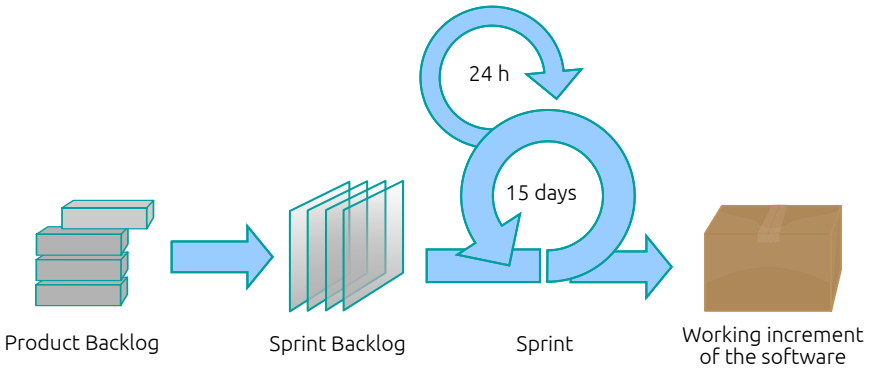

FIGURE 5.2: *Scrum* methodology process overview. Adapted from Wikimedia [29]

Roles

The follwing are the relevant roles that emerge in a Scrum developed project:

- **Product owner:** the product owner represents the stakeholders and is the voice of the customer. He or she is accountable for ensuring that the team delivers value to the business. The product owner writes *user stories* (tasks) and adds them to the *product backlog*, prioritizing them.
- **Scrum master:** Scrum is facilitated by a scrum master, who is accountable for removing impediments to the ability of the team to deliver the product goals and deliverables. The scrum master is not a traditional team lead or project manager, but acts as a buffer between the team and any distracting influences. The scrum master ensures that the scrum process is used as intended.
- **Development team:** the development team is responsible for delivering potentially shippable increments of product at the end of each *sprint*. A team is made up of 3–9 individuals with cross-functional skills who do the actual work: analyse, design, develop, test, document, etc. Finally, it is important to emphasize that the development team in Scrum is self-organizing.

Events

A series of *events* take place during the Scrum process, configuring the actual workflow of the team. We will provide an overview of some of them:

- **Sprint planning:** at the beginning of a sprint, the team holds a sprint planning event, in which the work to be done is selected from the product backlog and transferred to the sprint backlog.

- **Daily Scrum:** a stand-up, timeboxed and short meeting takes place every day during each sprint. In these meetings, every member of the team explains the work carried out the previous day, discusses any impediment or blocking situation he or she has encountered and decides which tasks will do in the following day.
- **Retrospective:** at the end of each sprint, a review of the work that has been completed is made, and the team reflects on the past sprint to identify and agree on any process improvement, which requires actions to be taken in the upcoming sprint.

Artifacts

Even though we have given an overview of some of them, the following artifacts are the remaining pieces that shape up the Scrum process and methodology:

- **Product backlog:** the product backlog is an ordered list of requirements that is maintained for a product. It consists of features, bug fixes, non-functional requirements, etc., i.e., whatever needs to be done in order to successfully deliver a viable product. The items in this backlog are ordered by the product owner based on considerations like risk, business value, dependencies or date needed, for example.
- **Sprint backlog:** The sprint backlog is the list of work the development team must address during the next sprint. The list is derived by selecting product backlog items from the top of the product backlog until the development team feels it has enough work to fill the sprint. The development team should keep in mind its past performance assessing its capacity for the new sprint, and use this as a guide line of how much *effort* they can complete.

5.2.2 Agile in this project

The methodology chosen for this project will be based upon Scrum, but major modifications will have to be made, for a number of reasons. Firstly, there is no such *development team*: a single developer will take care of the implementation of the project. Moreover, there is no possibility of having a Scrum master either. The project director will take a role between a technical coordinator and a product owner, although no real concept of *product* exists in the project, either way.

5.2.2.1 Practices

The adopted Agile practices for this project include:

- The usage of Trello[5] as a task tracking tool, to prioritize them similarly to the Scrum backlogs.
- **Sprint**-based development cycles with a sprint duration of one week.
- Constant (re)-evaluation of constraints and requirements, to forsee changes and take preventive action (similar to retrospectives, but less formal and certainly shorter).

5.2.2.2 Scope

Adopting Agile methodologies involves several decisions on how to manage the project and its requirements. In this particular project, if we are to examine the classical constraints (of which we talked about at the beginning of this chapter), we must be aware that the *schedule is fixed* (perhaps not the planning, but the final milestone) and this forces us to let the **scope opened**. This means that we will implement as much features as we can, assessing their quality, but no feature list will drive the success or failure of the project. Because we will be working on the basis of such an *open scope*, deviations in this field are likely to happen. These, however, will not result in a project failure in any case, because an agreement has been reached to work this way.

5.3 Schedule

The following subsections provide some details about the initial project planning (Section 5.3.1), as well as the changes it has suffered over time (Section 5.3.2). There have been *significant* deviations concerning the original project schedule. Not only the global duration has been lengthened, but more phases have been layed out, as was needed. As a positive contrast, early detection of such alterations has been sometimes possible.

5.3.1 Initial schedule

In this section, we cover the original analysis that was reported during the Project Management module[6], at the beginning of this project's development.

[5] Its description, along with other resources and tools used, can be found on Section 5.4.1

[6] The Project Management module is a compulsory course that all students have to undertake when beginning their Bachelor's Degree Final Project, concerning project management concepts and techniques, as well as documentation.

5.3.1.1 Overall duration

Taking a general look at the project's schedule, we can estimate it to have a total duration of about 5 months. Even though it was registered on July, 2014, the project did not begin until September, because August is the only month I can have holidays, due to job restrictions. Considering the next possible project's lecture shifts, we believe that the one taking place in December is too close in time. Thus, the project will endure until January the 26th, 2015. This should give us time enough to develop the project and document it without too much pressure, which is key to fulfill one of the main established goals: high quality results.

5.3.1.2 Schedule slack

The project schedule we present herein does not fill up the total amount of time available - more than two weeks are left blank, with no assigned tasks. This is intended because of the following reasons:

- The amount of time needed to develop the proposed algorithms is uncertain. It is hard to estimate the time it may take, because I have no previous knowledge on the area. Therefore, we opted for, in one hand, an *open scope* approach, and, on the other, leaving a considerable time gap between the last planned task and the project's final milestone: its defense. Being conservative, if the development of any proposed method is delayed, we still have some leeway to introduce schedule changes, without risking the project's success.
- We have estimated the project's report confection and the defense presentation rehearsals to be 35 and 7 days, respectively, but depending on how much development is finally carried out, it might not be time enough to write down the report. Extra time for doing it can be then borrowed from the schedule slack time.

5.3.1.3 Schedule monitoring & changes

For the development phase of the project, the most suitable way to monitor the schedule we have found is applying an Agile approach to the process. We will work in one week long sprints, meeting every week to assess the quality of the solutions, the proper progress of the project and to plan what will be done during the following sprint.

Sprint planning meetings are where the main goals of the project will be sliced in small tasks, which can be tracked and implemented better, because they are not so complex. Thanks to this constant fine-grained planning process, schedule or scope deviations are detected earlier and can be managed efficiently, reacting before they affect deeper the

overall success of the project. Given that no fixed features list is assigned to each sprint of the development phase, if the completion of either of those features is delayed, it can be made to span for some more time.

Within each of the development sprints, burndown charts[7] will be used to monitor the progress of the sprint. These charts are helpful in identifying patterns of work (sprint-end rushes, for example) and can help developers maintain a constant rate of finished features.

Besides burndown charts and sprint planning meetings, the use of velocity charts will also be helpful to increase the predictability of the following sprint plannings. The more predictable they are, the less deviations will occur and the schedule will be more likely to be fulfilled.

5.3.1.4 Project phases

The project is divided in 4 main phases, besides of the undertaking of the Project's Management module. Each phase has an estimated duration and a risk evaluation in terms of schedule deviation. The amount of hours is an approximated calculation from the number of days in each phase: 4 hours a day are estimated to be spent, because I am currently working part-time and also taking some subjects. A more detailed task granularity can be seen in the Gantt chart (on Figure 5.3 and Figure 5.4). Task dependencies are shown in the chart too. Those phases, chronologically ordered are:

[**Phase 1**] **Contextualization**: it is intended to perform a deeper bibliographic research and a study of the main subjects concerning the project, at the theory level - no practical skills or technological research will be done.

- **Duration estimation:** 11 days (44 hours).
- **Risk:** this phase has a medium to high risk of being delayed, due to lack of effective time (a wrong estimation), and also because more insight than planned might be needed, consuming more time.

[**Phase 2**] **Environment setup**: during this phase, all necessary tools and material resources will be gathered and configured. The concrete developing workflow will be decided, too.

[7] A burn down chart is a graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal. That is, it is a run chart of outstanding work. It is useful for predicting when all of the work will be completed. It is often used in agile software development methodologies such as Scrum.

- **Duration estimation:** 8 days (32 hours).
- **Risk:** this phase has a low risk of being delayed, because the technology that is to be used is, a priori, well known to us.

[**Phase 3**] **Development**: all of this project coding will be performed during this phase. As said before, a sprint methodology will be used during this phase, being one week each.

- **Duration estimation:** with an initial planning of 7 sprints, 49 days will be used (196 hours).
- **Risk:** there is a medium risk of this phase to be delayed. Even with the use of Agile methodologies, if a fundamental feature was needed and there was no more time left, another sprint (or at most a couple of them) could be introduced, to finish the remaining tasks.

[**Phase 4**] **Documentation**: the project's report will be written after the development phase, along with any deployment documentation that was required and the final presentation, which will also be rehearsed then.

- **Duration estimation:** 42 days (168 hours).
- **Risk:** this phase has a medium risk of being delayed too. Reviews of the report will be made and writing in English might take up more time than expected.

5.3.1.5 Detailed schedule: Gantt chart

A detailed Gantt chart of the schedule can be seen in Figure 5.3 and Figure 5.4. The chart was generated with the *Project management* free software package, available online on the Ubuntu 12.04 Software Center. Please note that there is no way the chart could fit in a single page (not even if it was landscape).

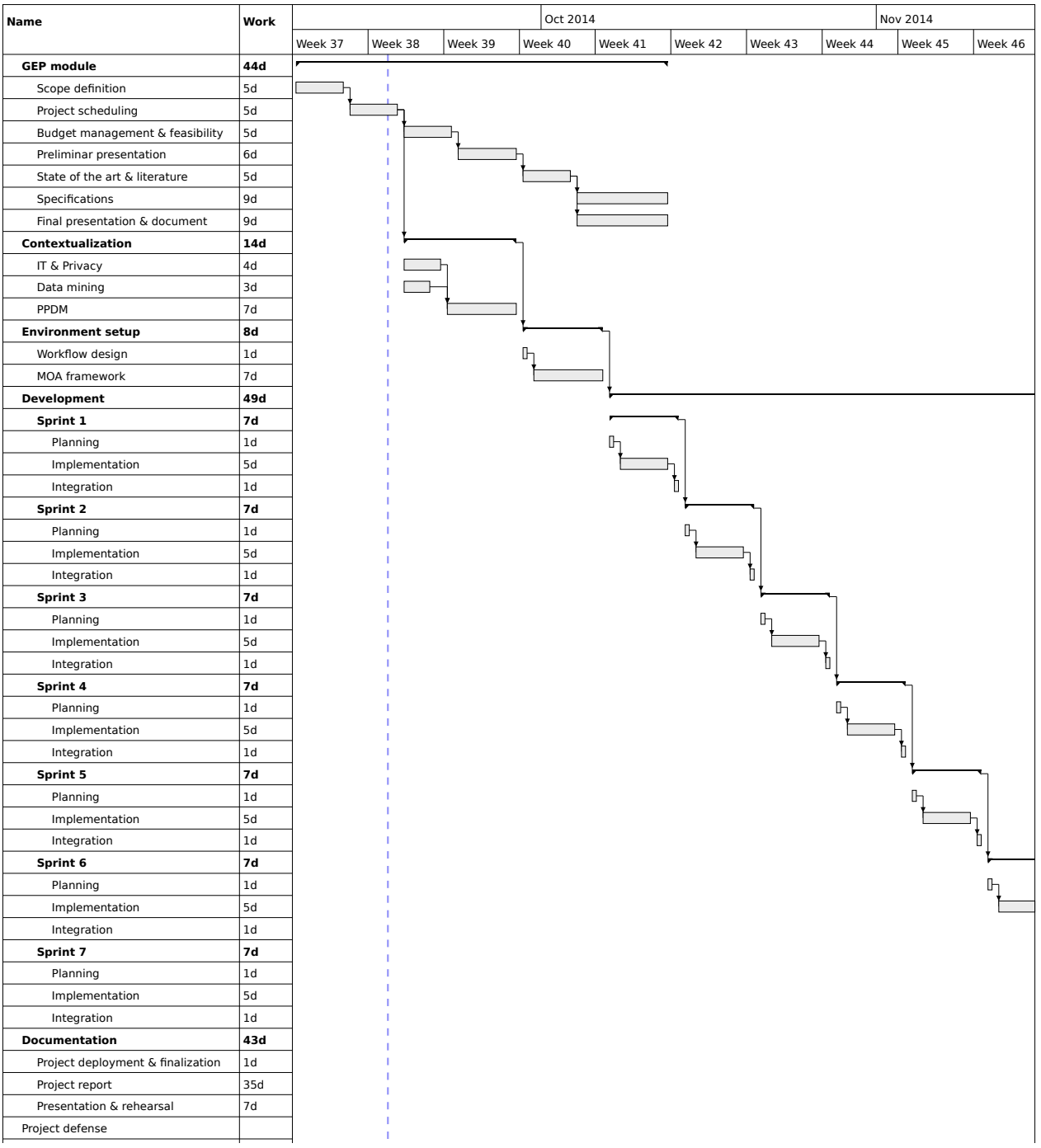

FIGURE 5.3: Initial project schedule Gantt chart (part 1).

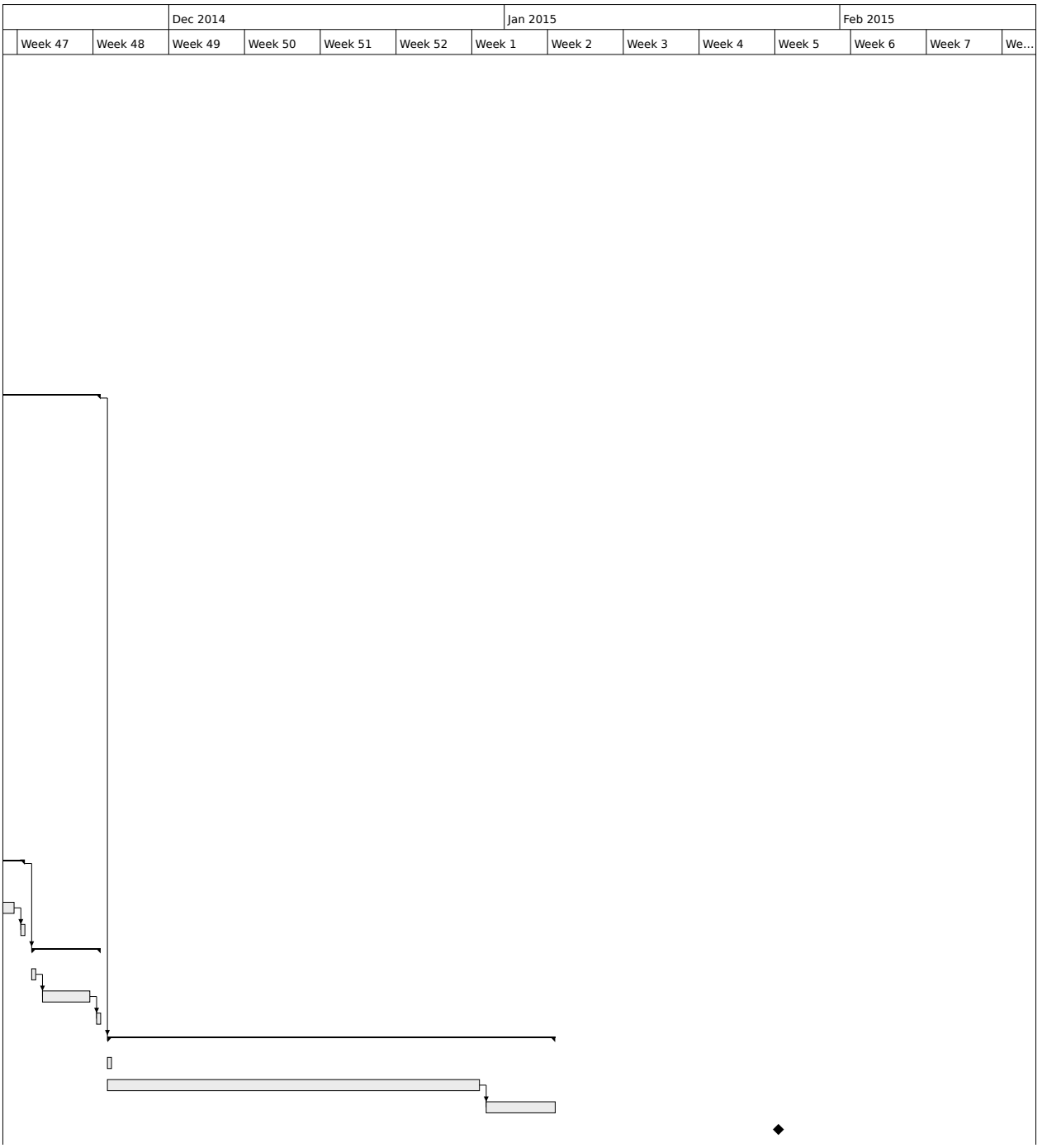

FIGURE 5.4: Initial project schedule Gantt chart (part 2).

5.3.2 Schedule deviation

We will cover now the changes that have occurred in the schedule of the project and analyze its causes.

5.3.2.1 Overall duration

The original total duration has been extended from 5 months to 8 months, approximately. Thus, the final report and its defence is now scheduled to be in April, which is the next available lecture shift in the Faculty. We believe that this extended duration will allow us to fulfill all requirements defined in the scope of the project.

5.3.2.2 Deviation analysis

There are several possible reasons behind this schedule deviation:

- The Project Management module lasted longer than expected, forcing the development phase of the project to begin later.
- During the definition of the project initial schedule, we expected to begin developing it while the Project Management module endured, which was, definitely, a planning error. Such tasks concurrency was not possible at that time.
- At the beginning of the development phase, we explored different technological alternatives, before deciding which approach was mostly suited to our needs, but this exploration delayed the actual development process for a couple of weeks.
- As was already stated in the Project Management report, some of the requested features have posed to be more complicated than was expected, consuming some more time than that assigned to them.
- For personal reasons, no work could be carried out during the Christmas vacations, which lasted two more weeks, furtherly delaying the project's development.

5.3.2.3 Current detailed schedule

Considering the previous analysis, a new Gantt chart has been built, with the new project's schedule, which is detailed in Figure 5.5 and Figure 5.6.

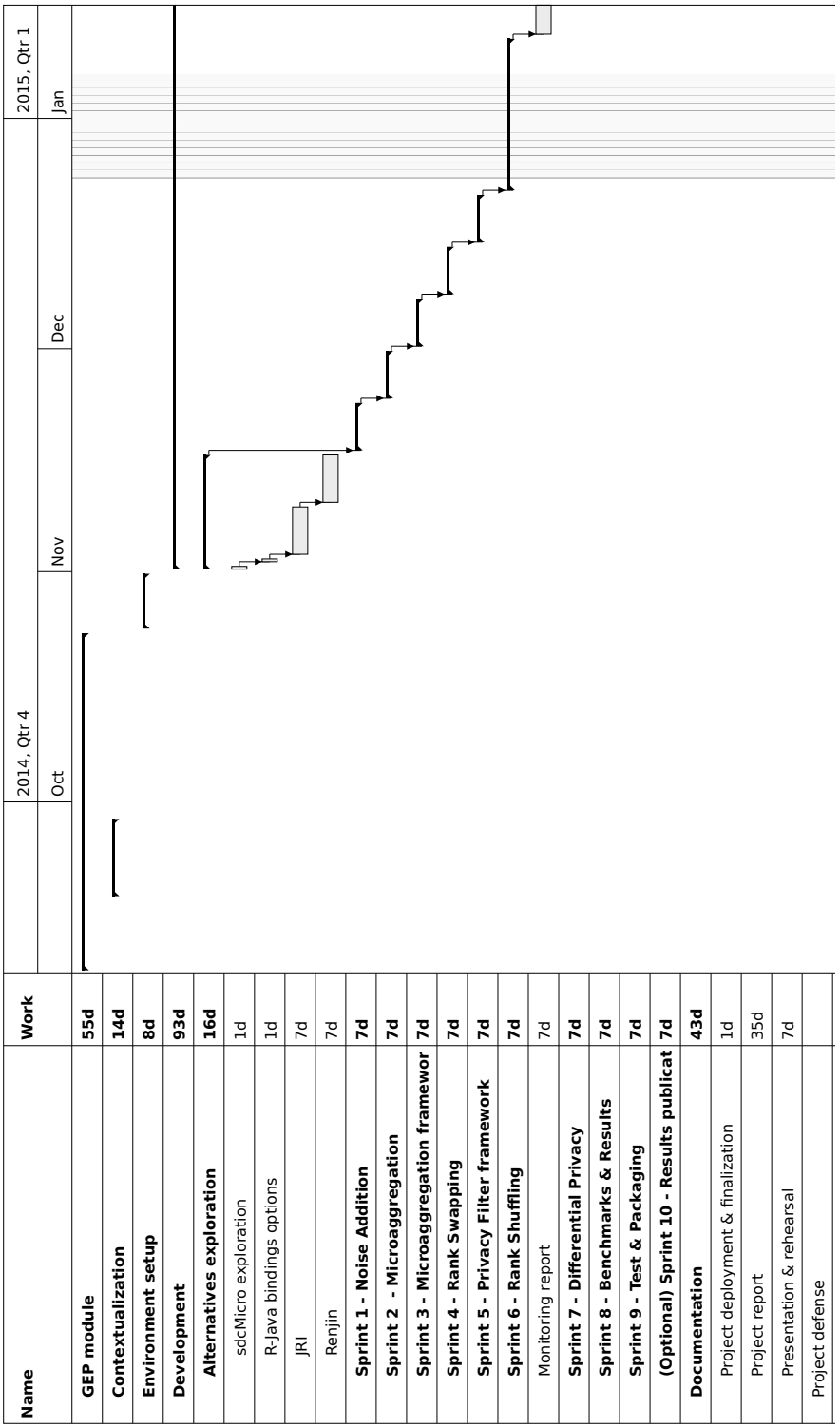

FIGURE 5.5: Final project schedule Gantt chart (part 1).

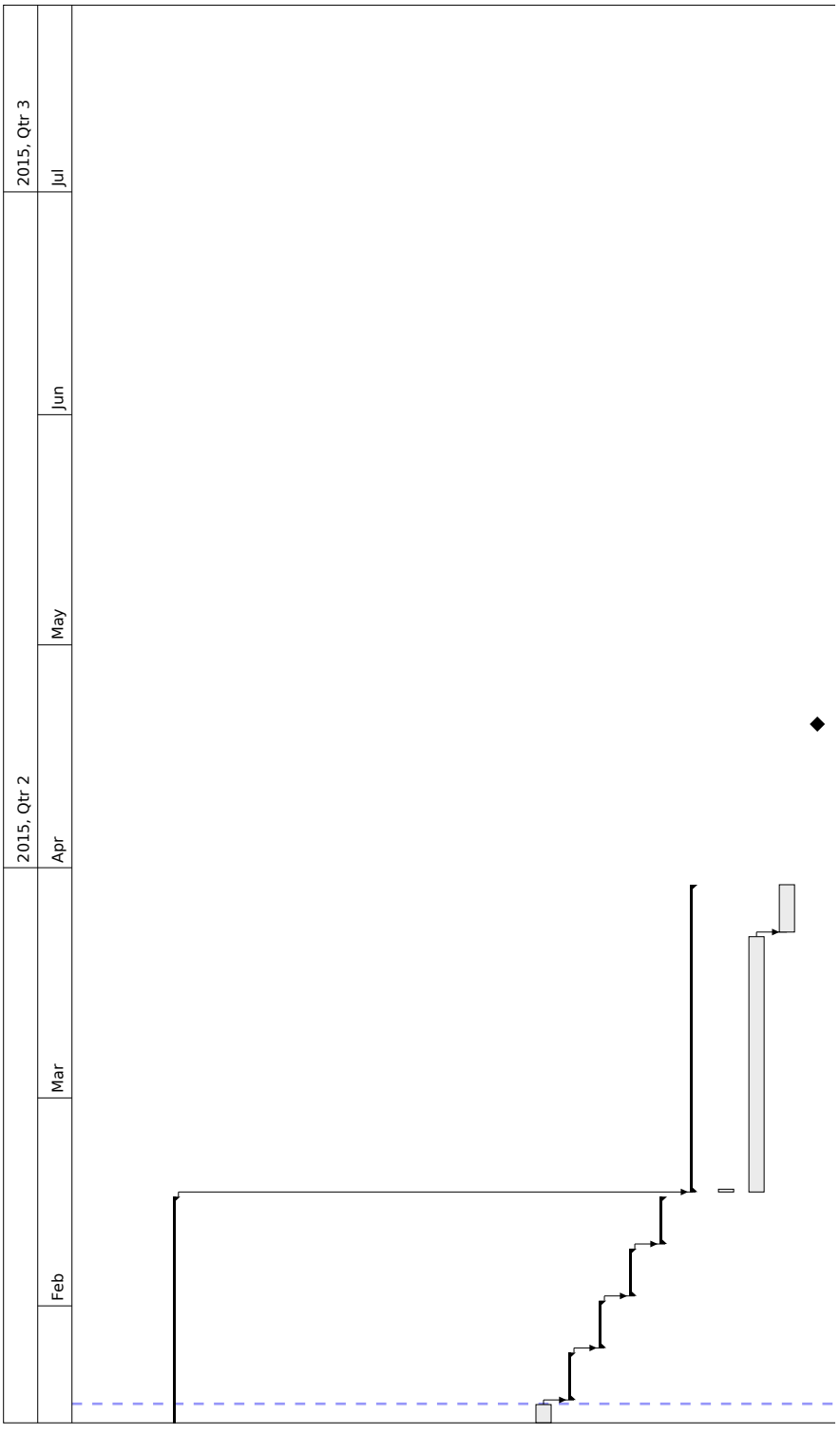

FIGURE 5.6: Final project schedule Gantt chart (part 2).

5.4 Budget

The initial budget and resources analysis, performed during the Project Management module, corresponds with Section 5.4.1, Section 5.4.2 and Section 5.4.3. Due to schedule deviations during the project, a final budget estimation is given in Section 5.4.4.

The project's budget is entirely based on an estimation of human, hardware and software resources costs. No real income is perceived, besides the salary of the project's supervisor, who is a tenure-track lecturer at the Barcelona School of Informatics, and an associate researcher at the Barcelona Supercomputing Center. No third parties are involved in the project - no companies or organizations are providing any funds. Moreover, even though the work is to be integrated into the MOA framework, it is indeed an open-source project, to which we will be contributing, meaning contributions are expected from any kind of source, be it funded or not.

All other associated costs are *externalized*, either by people involved in the project or by the university, where the development of the project will be held.

5.4.1 Resources & budget estimation

Resources consumed in this project only fall in one of the following categories: *human resources*, *hardware*, *software* and *other expenses*. For a detailed description of what will be needed in the project, please see the following subsections. It is important to keep in mind that *all* resources will be consumed equally througout the entire project duration.

5.4.1.1 Human resources

Human resources are summarized in Table 5.1.

All expenses included here are related to people's salaries. Only one developer will be working on this project, but a number of hours involving supervision tasks is also imputed to the project's supervisor, so its corresponding cost is added too. Taxes are included in all of the following items. The price is also an estimation: on the developer's side, it is based on a salaries comparison webpage (*Glassdoor* [30])[8]; on the supervisor side, the price is based on his own estimation.

- **Developer:** an average of 20 hours a week are estimated, spanning for about 21 weeks, summing up a total of 420 hours.

[8] As of date 12th October, 2014, the average salary for a software engineer in Barcelona is 32000€ per year (including taxes). Considering 12 monthly instalments and an average of 160 hours per month, this yields a total of 16.66€ per hour.

- **Supervisor:**
  - **Project's take off:** 8 hours, between meetings and initial planning.
  - **Sprints:** 8 hours each sprint, taking into account both face to face meetings and other supervising tasks. There are 7 sprints scheduled so far, making a total of 56 hours.
  - **Documentation:** during the project's final stage, an estimation of 20 hours is taken from the corresponding supervision of the project's report.

Role	Price (per hour)	Working hours	Total
Supervisor	35€	84	2940€
Developer	16.66€	420	6997.2€
		Total	**9937.2€**

TABLE 5.1: Human resources associated costs. All taxes are included in the Price per hour column.

5.4.1.2 Hardware resources

Hardware resources are summarized in Table 5.2.

All hardware needed resources are shown in the corresponding table. Their cost is calculated by estimating its amortization, spanned over 5 years (it is a personal laptop). To calculate its amortized cost per hour, we will take into account that this equipment is used throughout the course too, and estimating that 2500 hours of work are carried each year.

Product	Price	Units	Amortized price per hour	Work time (hours)	Total
Asus k53sv	650€	1	0.052€	420	21.84€
				Total	**21.84€**

TABLE 5.2: Hardware amortization costs. All taxes included.

5.4.1.3 Software resources

All software needed to undertake this project is free and, most of it, is open sourced. Despite this, we will include a list of it here, to show what will be used at a finer grain.

- **Ubuntu 12.04**: operating system. Available at: `http://www.ubuntu.com/download`.
- **Trello**: online task management tool. Available at: `https://trello.com/`.
- **Google Drive**: online, collaborative office software suit, used to create burndown charts (spreadsheets). Available at: `https://drive.google.com`.

- **Java SDK**: Java language Software Development Kit. Available at: `http://openjdk.java.net`.
- **Eclipse IDE**: integrated development environment package. Available at: `https://www.eclipse.org/home/index.php`.
- **Git**: source version control system. Available at: `http://git-scm.com/`. Remote code repositories will be hosted at GitHub (`https://github.com`) for free.
- **MOA**: Massive Online Analysis, a stream mining framework. Available at: `http://moa.cms.waikato.ac.nz`.
- **LaTeX**: document preparation system. Available at: `http://www.latex-project.org`.

5.4.1.4 Other expenses

All expenses not covered in the previous sections are detailed in Table 5.3.

Please note that the cost of each item of this section is an estimation. Moreover, even though they are displayed, since no budget is really available, they will be *absorbed* by the university, where most of the work will be carried out.

Product	**Price per month**	**Months**	**Total**
Energy	35€	4	140€
Water	25€	4	100€
Heat & air	30€	4	120€
Internet connection	40€	4	160€
		Total	**520€**

TABLE 5.3: Uncategorized resources estimated costs. All taxes are included.

5.4.2 Total budget estimation

The sum of the subtotals of the previous sections is shown in Table 5.4. Please note that, since taxes are already included in each item appropriately, there is no need to add them here.

Concept	**Total**
Human resources	9937.2€
Hardware	21.84€
Software	0€
Other expenses	520€
Total	**10479.04€**

TABLE 5.4: Total budget: summation of budget estimations.

All costs are just estimations and are not covered in any way, with the exception of the supervisor's salary. This means that, in fact, there is no possible way this project is feasible. However, given that the developer has no salary at all and that all other extra costs are assumed by the university or the developer, the project can be developed normally.

5.4.3 Budget control mechanisms

Any budget deviations related to material equipment or software purchases will be monitored in the sprint planning meetings at the beginning of each of those phases during the project. These possible extra costs will be assumed by the developer, since no other source of funds is available.

Another source of budget deviations can be found on the project's duration. If the schedule is not fulfilled and the project is delayed, extra cost in terms of human resources, hardware amortizations and other expenses would have to be added. They still would be treated as they are in the present analysis, meaning no significant change would occur.

5.4.4 Final budget estimation

Due to the deviation in the project's schedule, that was already analyzed in Section 5.3.2, an increment in the human resources, external expenses and hardware amortization budget contributions has arised. It is important to note that, given that no proprietary software package has been used, no additional costs might be derived from the lenghtening of the project duration. We will now cover this budget deviation and provide a final estimation of the project cost, which is summarized in Table 5.8.

Human resources: deviation

Following the analysis from 5.4.1, we just have to add the corresponding increment of working hours for both the developer and supervisor.

- **Developer:** an average of 20 hours a week are estimated, spanning for about 32 weeks, summing up a total of 640 hours. However, given that no work was carried during Christmas holidays, the total number of hours should be lowered to, at most, **600 hours**.
- **Supervisor:**
  - **Project's take off:** 8 hours, between meetings and initial planning.

- **Sprints:** 8 hours each sprint, taking into account both face to face meetings and other supervising tasks. With 10 sprints of final work, this yields a total of **80 hours**.
- **Documentation:** during the project's final stage, an estimation of **20 hours** is taken from the corresponding supervision of the project's report.

Considering the previous estimation and keeping the same prices per hour of the initial estimation, the following total human resources cost is calculated (see Table 5.5).

Role	**Price (per hour)**	**Working hours**	**Total**
Supervisor	35€	108	3780€
Developer	16.66€	600	9996€
		Total	**13776€**

TABLE 5.5: Human resources associated costs (final estimation).

Hardware resources: deviation

The only change in the hardware related costs is the number of working hours devoted to the project, which have a direct impact on the amortization of the equipment.

Product	**Price**	**Units**	**Amortized price per hour**	**Work time (hours)**	**Total**
Asus k53sv	650€	1	0.052€	600	31.2€
				Total	**31.2€**

TABLE 5.6: Hardware amortization costs (final estimation).

Other expenses: deviation

Given that the amount of months dedicated to the project's development has increased, the estimated cost for the expenses related to the developer's accomodation has to reflect the changes as well.

Product	**Price per month**	**Months**	**Total**
Energy	35€	7	245€
Water	25€	7	175€
Heat & air	30€	7	210€
Internet connection	40€	7	280€
		Total	**910€**

TABLE 5.7: Uncategorized resources estimated costs. All taxes are included.

Final estimation

The following is the final estimation of the project's budget, taking into account all deviations from the particular budget contributions.

Concept	**Total**
Human resources	13776€
Hardware	31.2€
Software	0€
Other expenses	910€
Total	**14717.2€**

TABLE 5.8: Total budget final estimation.

Chapter 6

Implementing the filters

We will cover now the results of the main development phase of the project, concerning the implementation of the MOA privacy preserving filters and the design decisions taken for each of them.

6.1 Alternatives exploration

The Massive Online Analysis stream mining framework is built in the Java language, thus providing some benefits in terms of portability, ease of maintenance and development, but also exposing some drawbacks, mainly due to the lack of easily parallelizable code, like is the case with C or Fortran by using the OpenMP[1] language extensions. Given the language enforcement MOA imposes and the existence of well-known SDC toolsuites, like the `sdcMicro` R package (reviewed in Section 3.2), an analysis of possible alternatives was taken during the first weeks of the project's development phase.

6.1.1 `sdcMicro` & Java

The most direct alternative, besides actually implementing the filters, was to use the `sdcMicro` library to perform the necessary calculations over the streaming data originated in MOA and take the results back to the framework. This approach can be better understood in Figure 6.1: a bi-directional connection between the Java runtime (the Java Virtual Machine or JVM) and the R process would be needed to be able to use the SDC methods of the `sdcMicro` library. The results of the exploratory analysis of this type of solution are summarized in Table 6.1.

[1] OpenMP (Open Multi-Processing) is a programming interface that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior. [31]

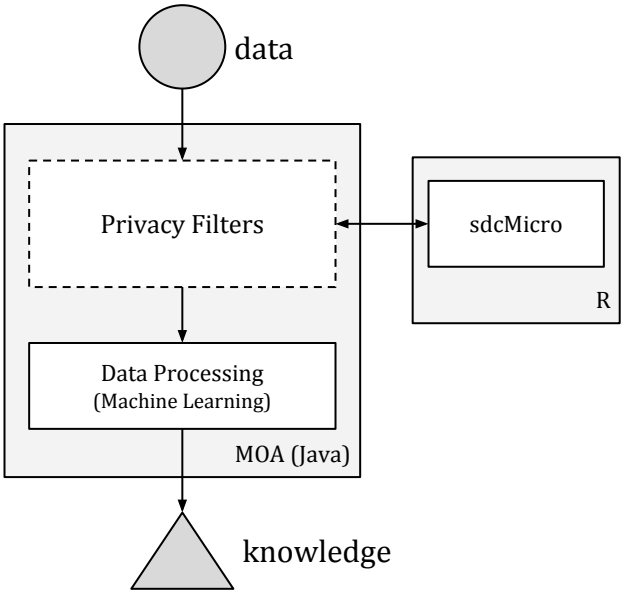

FIGURE 6.1: R/Java hybrid solution using the `sdcMicro` package.

This interconnection could be achieved by using some existing technologies that perform the inter-process communication based on different approaches:

- **rJava/JRI:** the `rJava` and `JRI` counterparts are a couple of libraries designed to provide low-level communication between the Java Virtual Machine (JVM) and an R process. `rJava` provides a low-level bridge between R and Java via the Java Native Interface (JNI)[2]. It allows to create objects, call methods and access fields of Java objects from R [33]. On the other side, `JRI` is a Java/R Interface, which allows to run R inside Java applications as a single thread. Basically, it loads R dynamic library into Java and provides a Java API to R functionality [34].
- **Rserve:** it is a TCP/IP server which allows other programs to use facilities of R from various languages without the need to initialize R or link against an R library. A typical use is to integrate R backend for computation of statstical models, plots etc. in other applications [35].

Due to performance related to networking protocols against native interface communication, `Rserve` was discarded as an option to implement filters for MOA: a streaming environment requires the maximum throughput possible for its algorithms and, thus, the overhead associated with TCP-based IPC is considered to be excessive.

[2]The Java Native Interface is a standard programming interface for writing Java native methods and embedding the Java Virtual Machine into native applications. The primary goal is binary compatibility of native method libraries across all Java virtual machine implementations on a given platform [32].

Benefits	**Drawbacks**
Faster development	No algorithms are indeed developed
Easily extensible	Strong dependencies
SDC methods are right	Depends on external installed software
	Needs system libraries to work
	Maintanability is harder
	Reduced performance due to marshalling

TABLE 6.1: Evaluation of the R/Java hybrid solution.

Anyway, either of such solutions imply that marshalling and unmarshalling techniques would have to be applied, in order to transform the data structures that are differently used by R and Java. Moreover, even though that no SDC algorithm would need to be implemented, the interconnect code would not be easy to maintain.

Finally, there is another important argument against the R/Java hybrid approach: its strong reliance in external dependencies. These dependencies not only make the installation of the SDC-enabled MOA framework more difficult, but are directly linked to third-party software and *system* libraries, making the environment less stable and robust, from the software user point of view. These dependencies are shown in Figure 6.2

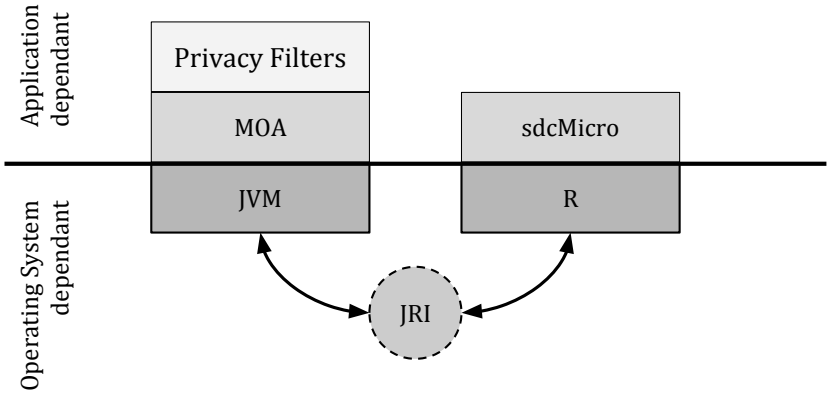

FIGURE 6.2: JRI based R/Java hybrid solution architechture: strong dependencies.

6.1.1.1 Renjin

Yet another altenative was explored that was meant to interconnect MOA with the `sdcMicro` package: the Renjin project [36]. **Renjin** is a JVM-based interpreter for the R language: all computations of any R package can be executed upon the JVM, instead of a separate R process. This way, the dependency that this project could have had on R and some system libraries disappeared. However, it is worth noting that, even with Renjin, data structures conversion would have to be performed, rendering its use as impractical as the use of the `JRI` library. Moreover, the `sdcMicro` package was still not

available in its JVM *port* at the time of the evaluation due to some internal dependencies and errors, and we could not wait for it to be solved.

6.1.2 Chosen alternative

As a simple remark, the final decision was to actually develop the filters for the MOA framework by extending it, in the form of a **pure Java** implementation.

6.2 MOA & Privacy Filters

Notation: from now on, a text stylized with a monospaced font like `this example` will refer to an actual programming artifact: a variable, class, file name, etc.

We have already showed along this report that *filters* are a feature of the MOA framework. Filters are, actually, a particular form of *stream*. Whenever a filter should be applied to a stream to perform a posterior analysis, a `FilteredStream`[3] is built. This class takes a generic `Stream` object as the input stream and a list of `StreamFilter`s, which are also `Stream`s, if we examine their type hierarchy. Taking advantage of the existence of both the `FilteredStream` and `StreamFilter` classes, we can begin designing the privacy filters that will implement the actual SDC methods.

6.2.1 `PrivacyFilter`

Thanks to the object-oriented capabilities of the Java language, we can design and implement a generic abstract data type for all proposed SDC algorithms. By doing so, we will be able to centralize some of the common logic behind them. The abstract type of the privacy filters is the `PrivacyFilter` class. An incomplete UML diagram of the specification of this class and its most relevant parent types can be seen in Figure 6.4.

Concerning the responsibility[4] of this class, there is a main task that the `PrivacyFilter` is meant to address: the measurement or *evaluation* of the **disclosure risk** (DR) and the **information loss** (IL). The approach is to let the SDC method (the *concrete* subclass) anonymize the *instances*[5] of the stream and collect them, along with the original instances that have been processed. The evaluation of both magnitudes, DR and IL,

[3] All documentation of the MOA API can be found on `http://www.cs.waikato.ac.nz/~abifet/MOA/API/index.html`

[4] In object-oriented programming, the *single responsibility principle* states that every class should have responsibility over a single part of the functionality provided by the software, and that responsibility should be entirely encapsulated by the class (Martin [37]).

[5] In the MOA context, records in a dataset (in a stream) are called *instances* and are represented using the `Instance` interface.

is performed by *estimators* using these *pairs* of instances (see Estimators, below). The mechanism is best understood with the schematic presented in Figure 6.3.

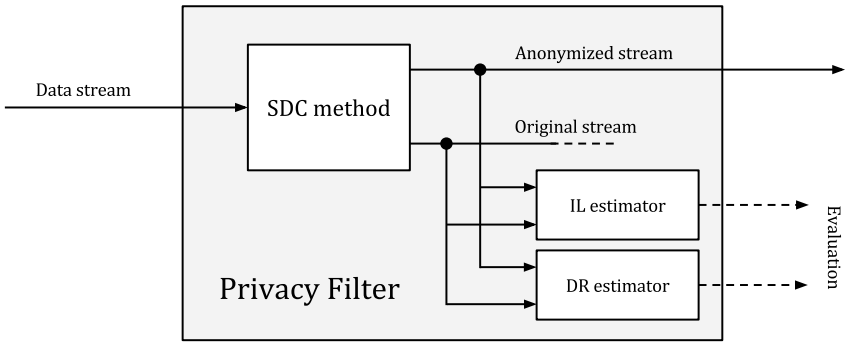

FIGURE 6.3: A schematic of the tasks performed by the `PrivacyFilter` class, showing the stream data flow.

Implementation details

The `PrivacyFilter` class is, by construction, an `InstanceStream` and a `StreamFilter` (see Figure 6.4) but also, and most importantly, an `AnonymizationFilter`. This last *interface*, which the `PrivacyFilter` implements, allows to encapsulate all privacy-preserving behaviour in a single module.

The most important of the methods defined in the `AnonymizationFilter` type is

```
nextAnonymizedInstancePair() :  InstancePair
```

which is left to be implemented (it is *abstract* at the `PrivacyFilter` level) by any subclass. This way, we can use *inversion of control*[6] to force a subtype define the concrete behaviour of the function, while still conforming to a precise *contract*. The `InstancePair` returned by this abstract method is just a *pair* structure containing both the original and anonymized instances that should be streamed next. If we say that x is an instance and x' its anonymized counterpart, the `InstancePair` class would simply be the tuple $\langle x, x' \rangle$.

Estimators

The estimators used by the `PrivacyFilter` class are desgined to be modular and, most of all, easily modifiable: they are just interfaces defining a contract that all estimators

[6]The *Hollywood principle* or *inversion of control* pattern is a software design methodology that takes its name from the cliché response given to amateurs auditioning in Hollywood: "Don't call us, we'll call you". It is a useful paradigm that assists in the development of code with high cohesion and low coupling that is easier to debug, maintain and test.

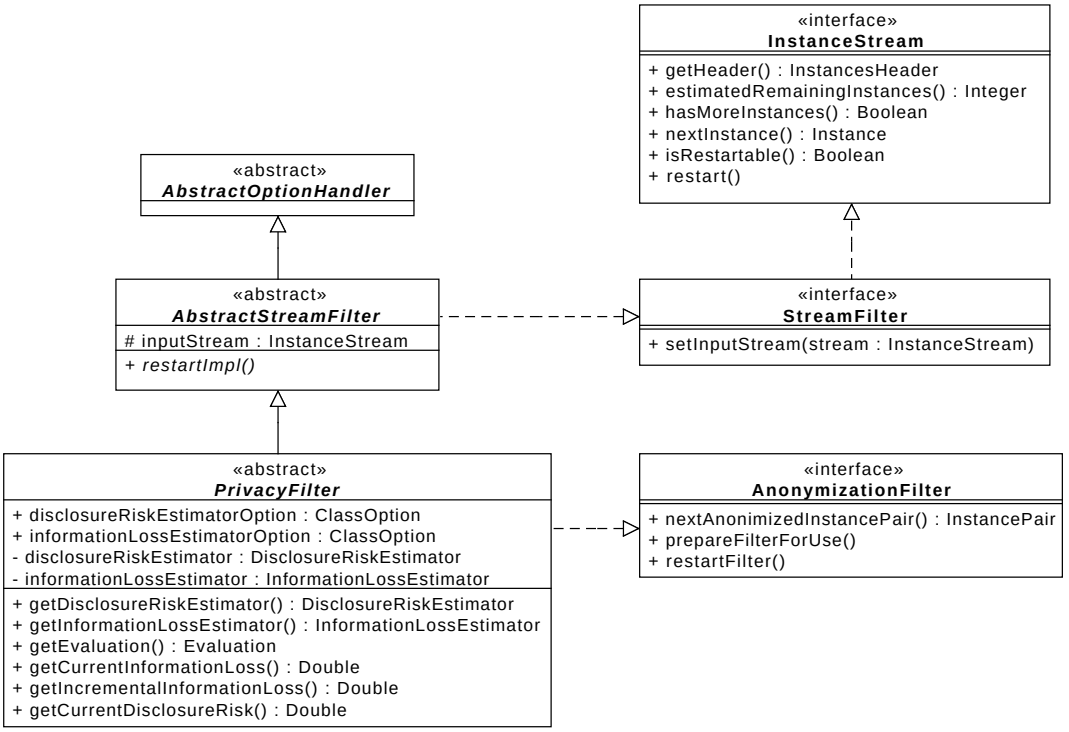

FIGURE 6.4: UML class diagram of the relevant types in the `PrivacyFilter` class hierarchy. Notice that not all the involved types are shown.

must implement. The methods belonging to such contracts can be seen in Figure 6.5 (the concrete estimators implementation is explained in Section 6.3). Again, there is one particular method that is most important in the estimators context:

```
performEstimationForInstances(instancePair :  InstancePair)
```

This method is the generic way for the `PrivacyFilter` to feed the estimators with $\langle x, x' \rangle$ tuples (`InstancePairs`). The estimators have the responsibility of performing the necessary calculations using this stream of pairs of instances.

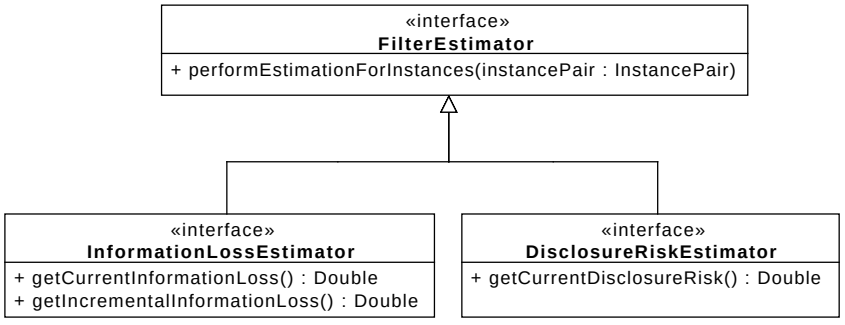

FIGURE 6.5: Class diagram of the `FilterEstimator` type hierarchy.

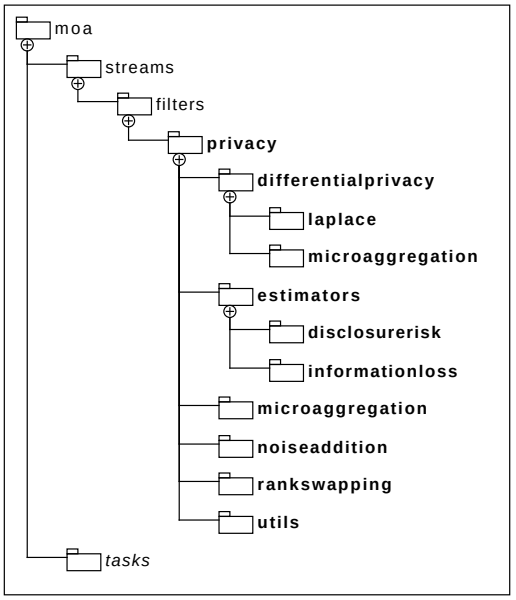

FIGURE 6.6: Package organization of the privacy filters. **New** packages (not existing in the MOA framework) are shown in bold. *Existing* packages that have been extended with new types are shown in italics.

Finally, the DR and IL estimators used by a `PrivacyFilter` can be configured at runtime by setting the appropriate *options*[7] of the filter.

6.2.2 Filters ecosystem

Having reviewed the basic `PrivacyFilter` generic type, we can now provide a couple of figures that introduce the final structure of the privacy filters class ecosystem. The *package* encapsulation of the methods can be seen on Figure 6.6. An incomplete[8] class diagram of the filters is shown in Figure 6.7.

6.3 Estimators

We have implemented a pair of disclosure risk and information loss estimators, conforming to the corresponding interfaces described in Figure 6.5. These concrete implementations are the default estimators of the `PrivacyFilter`, but can be configured as necessary, being able to plug in different methods.

[7]The MOA framework makes extensive use of configurable *options*, which can be set either on a command line execution or via the GUI that MOA provides.

[8]Most of the classes shown in the diagram depend on others for their internal implementation, but, for the sake of concreteness, they are not shown, as are not relevant for the purpose of this report.

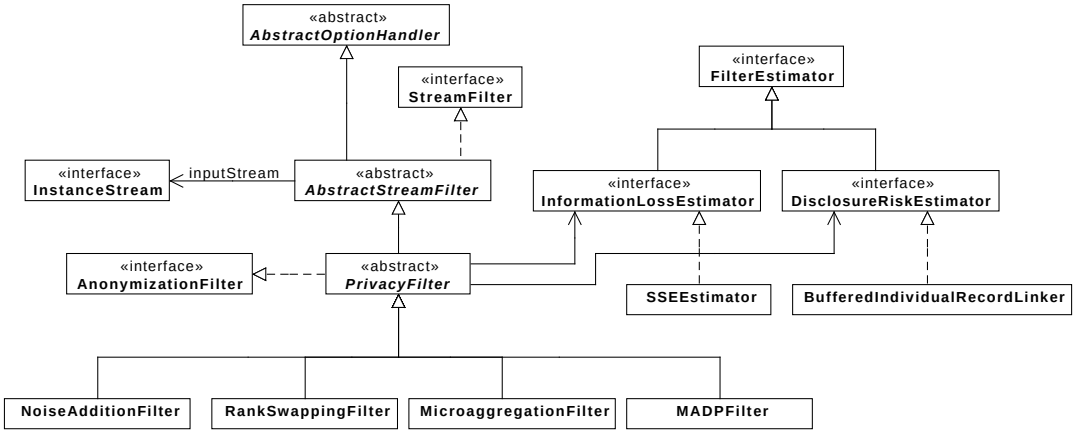

FIGURE 6.7: Class diagram of the privacy filters ecosystem. Only the relevant types are shown and no method or member specifications have been included.

6.3.1 `BufferedIndividualRecordLinker`

The disclosure risk estimator, called `BufferedIndividualRecordLinker`, uses a distance-based record linkage approach (see Section 2.2.3.1) to estimate the risk of records re-identification.

The estimator holds a buffer W, thus its name, of the last b original instances, this is, non-anonymized records, with size $|W| = b$ as an input parameter. Each time that a $\langle x, x' \rangle$ pair is passed in to the estimator, it adds the original instance x to the buffer, deletes the oldest seen one, and performs a record linkage trying to re-identify x' with any instance in the buffer.

The re-identification works as follows: for each instance w_i, with $0 \leq i \leq b-1$, we store it a set G if its distance to x' is the minimum one recorded, named δ. Whenever an instance is found at distance $d < \delta$, all the instances are removed from G and both this set and δ are updated accordingly. At the end of the buffer traversal, the target original instance is checked to see if it is in the set G. The *linkage probability* for an anonymized instance x' is calculted as

$$P(x') = \begin{cases} 0 & \text{if } x \notin G \\ \frac{1}{|G|} & \text{if } x \in G \end{cases} \tag{6.1}$$

Being X the set of all the instances already processed and $|X| = n$, the disclosure risk is estimated in a $[0, 1]$ range as

$$DR = \frac{\sum_{x \in X} P(x')}{n} \tag{6.2}$$

Finally, the distance measure used in the estimator follows a modification of the Euclidean distance which also takes into account categorical variables. It is best explained with the pseudo-code representation shown in Procedure 6.1.

Procedure `distance(`x`,`y`)`

Data: x, y instances

Result: the distance measure, d

```
begin
    d ← 0;
    for i ∈ attributes(x) do
        if isNumeric(i) then
            d ← d + (x_i − y_i)^2;
        else
            if x_i ≠ y_i then
                d ← d + 1;
    d ← √d;
    return d;
end
```

6.3.2 `SSEEstimator`

The information loss estimator implemented as a default for the `PrivacyFilter` class uses an unbounded approach (see Section 2.2.4) to measure the amount of useful information that is lost with the application of such privacy filters.

The aim of the implementation given in this project is to provide a way to compare the diverse privacy filters, this is, we do not intend to achieve a reliable and precise IL measurement. Therefore, the estimation is simply based on the *sum of square errors* or SSE between the original and anonymized instances, x and x', respectively. If we call X to the set of original instances already processed and X' to its anoymized counterparts, the SSE is calculated as

$$SSE = \sum_{x \in X} \sum_{x' \in X'} (\text{dist}(x, x'))^2 \tag{6.3}$$

where the distance metric used is the same that was defined for the DR estimator in Section 6.3.1 (see Procedure 6.1). The main drawback in using this approach, besides it being more difficult to make comparisons due to not being a bounded measure, is that categorical attributes are overweighted, thus distorting the validity of the estimation.

6.4 `NoiseAdditionFilter`

The *uncorrelated* noise addition mechanism that was reviewed in Section 2.3.1 to protect microdata is implemented in the `NoiseAdditionFilter` class. Given the low level of data protection that this family of algorithms are capable of [38], we have not implemented any further sofistication, such as estimating correlated noise or using (non)-linear transformations to obtain it.

6.4.1 Design

The `NoiseAdditionFilter` adds uncorrelated noise to the values of the attributes of an instance x, whether they are numerical or categorical. This is achieved using an array of *observers*, one for each variable. If an attribute is numerical, its associated observer is a `GaussianEstimator`, an existing class in the MOA framework that allows us to incrementally (thus best suited to streaming data) estimate the properties of a gaussian distributed variable: the *mean* μ and the *variance* σ^2 (or standard deviation, if desired). On the other hand, for each categorical attribute, its observer stores a set of all the different values that previously processed instances had.

The filter has two input parameters: a and c, both real numbers in the $[0, 1]$ range, which act as a scaling factor of the noise being applied to *attributes* and to the *class* variable, respectively.

We denote by x_i the value of the i-th attribute of the instance x and by x'_i its masked (distorted) counterpart. For a numeric variable, the noisy values are calculated as

$$x'_i = x_i + \beta \cdot \sigma \cdot \epsilon \tag{6.4}$$

where $\beta \in [0, 1]$ is one of the input parameters a or c, σ is the standard deviation estimate, obtained from the attribute's `GaussianEstimator` observer and, finally, ϵ is drawn from a gaussian random variable $\varepsilon \sim N(0, 1)$.

For a categorical variable i, its value for a given instance, x_i, is replaced by another value $x'_i \in \text{Range}(i)$. Given that MOA encodes the values of categorical attributes as natural numbers, we can simply select x'_i from a uniform discrete random variable bound to the range of the attribute as it is estimated by its observer. In order to preserve the scale of the amount of noise being added, this replacement only takes place if $\epsilon < \beta$, with β being either the a or c parameter and ϵ drawn from a random variable $\varepsilon \sim N(0, 1)$.

Finally, because no complex processing is needed to implement this filter, its computational cost bounded by $O(n)$, with n being the number of instances anonymized by the algorithm.

6.4.2 Summary

The `NoiseAdditionFilter` implements an uncorrelated noise addition scheme to the instances of the filtered stream. Table 6.2 summarizes the main properties of the filter.

NoiseAdditionFilter	
Parameters	a, c scaling factors of the noise added for $a$ttributes and $c$lass variable
Type of data	Heterogeneous (both numeric and categorical attributes)
Cost	$O(n)$

TABLE 6.2: `NoiseAdditionFilter` summary.

6.5 `MicroAggregationFilter`

The `MicroAggregationFilter` class is an implementation of the *microaggregation* SDC method that was reviewed in Section 2.3.2. It is one of the best performing filters in terms of both speed and disclosure risk versus information loss trade off.

6.5.1 Design

There are three main issues that are involved in the design of the microaggregation filter: the need of a *sliding window* and the *partition* and *aggregation* steps.

6.5.1.1 Buffered filter

The first issue to address when designing the microaggregation implementation was the adaptation of existing well-known algorithms to a streaming environment. It is obvious that no partition can be made by just processing a single instance at a time: we *need* some kind of historical knowledge of the previous or future records that the algorithm will process in order to cluster them into groups. Given that MOA uses a sliding window (see Section 2.1.1) technique to perform most of the machine learning tasks, we decided to follow the same approach: we use a historical instance buffer to perform the partition and aggregation steps. We say that it is *historical*, because it holds the last b instances of the stream, being $b \in \mathbb{N}^+$ an input parameter.

Because the contract defined in the `AnonymizationFilter` interface requires that the result of an anonymization step is a pair of an original and an anonymized instances, it

is not the only buffer we need. Therefore, a second vector is used to hold the actually modified instances. A third list (containing boolean values) is used to control which of the instances in the buffers have been already anonymized.

We say that a MOA filter implementation of an SDC method using this processing scheme (a sliding window) is a *buffered filter*.

Notation: from now on, when discussing implementaiton details of buffered filters, we will use the following notation and symbols:

- The *original* instances buffer is named W and has length $|W| = b$ [9] By w_i, we denote the i-th instance stored in the buffer, being w_0 the oldest one and w_{b-1} the most recently added.
- The *anonymized* instance buffer is named W' and has the same length than the previous buffer: $|W'| = b$. The i-th instance of the buffer is denoted by w'_i.
- For convenience, A denotes the set of already anonymized instances. A generic instance x is said to be anonymized if $x \in A$.
- The value of the j-th attribute of an instance w_i is denoted as w_{ij}.
- *Named* instances are those that are denoted using greek letters: τ, σ, ρ. These instances *must* be named by explicitly denoting which position they are in a buffer; for example: $\tau \leftarrow w_0$.
- The value of the j-th attribute of a named instance is denoted as τ_j, for example. Notice that, since a named instance is already well defined in terms of its position in the buffer, a single subscript index is needed to reference an attribute.
- The instance to be anonymized is called the *target* and is always referred to by a the named instance τ.

Procedure 6.2 shows the common implementation of the `nextAnonymizedInstancePair()` abstract method (defined in the `AnonymizationFilter` interface) for any buffered filter: the buffer is filled with instances of the input stream S and, if the target instance ($\tau = w_0$) has not already been processed, its anonymization is requested via the `processNextInstance()` method. After this procedure has been called, the instance

[9] The size of the historical buffer is a common parameter to all buffered filters.

pair $\langle x, x' \rangle$ is built, the target instance is removed from all necessary buffers and the tuple is returned.

Procedure `nextAnonymizedInstancePair(`*void*`)`

Data: buffers W, W', A and stream S

Result: an instance pair $\langle x, x' \rangle$

begin

```
while S.hasMoreInstances() and |W| < b − 1 do
    s ← S.nextInstance();
    W ← W ∪ s;
    W' ← W' ∪ s;
x ← w_0;
if x ∉ A then
    processNextInstance();
x' ← w'_0;
W ← W − {w_0};
W' ← W' − {w'_0};
A ← A − {x};
return ⟨x, x'⟩;
```

end

6.5.1.2 Partition

Concerning the clustering step of microaggregation, we have seen that the MDAV and μ-Approx algorithms are best suited to achieve the lowest information loss possible (see Section 2.3.2), but a more thorough evaluation forced us to discard them as they are rather too computationally complex, given the streaming context we are in.

Domingo-Ferrer et al. [20] show that both methods (MDAV and μ-Approx) are bounded to a $O(n^2)$ complexity time, where n is the number of records (instances) processed. With such a high cost, a sensible implementation would do the clustering step just once, when the window was full of instances, thus getting a complete partition (*all* instances would belong to a cluster.) and then returning the whole window as a block. This is, no real streaming scheme would be used; instead, we would be doing *block* processing.

Our proposal is to use a k-Nearest Neighbours (KNN) algorithm to *continuously* partition the sliding window and be able to provide anonymized instances much faster, by building just *one* cluster each time a new instance is requested to the filter. The records in this single cluster are then aggregated and the target instance is returned. The computational cost of this approach is quite lower than that of the MDAV and μ-Approx heuristics, as long as the sliding window size remains relatively small.

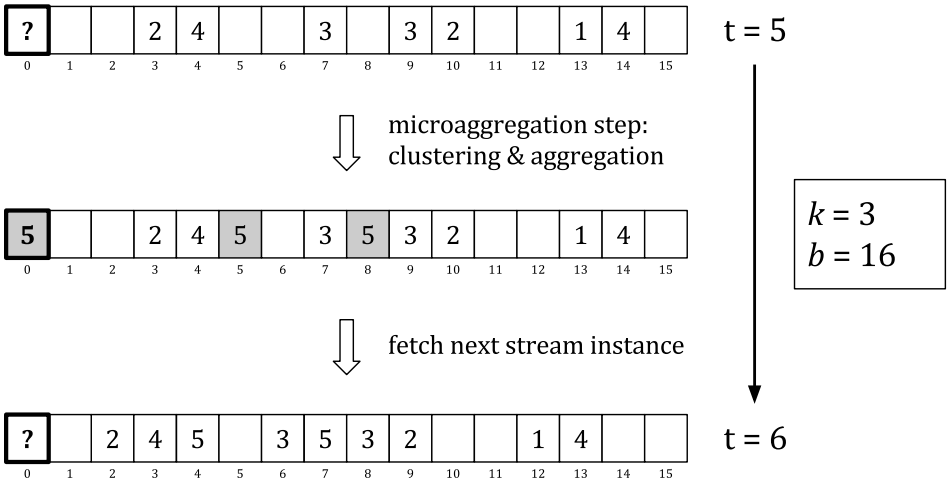

FIGURE 6.8: k-Nearest Neighbours based microaggregation schematic of a whole processing step (from time $t = 5$ to $t = 6$). The *target* instance at the top of the buffer (position 0) is anonymized by aggregating the values of its attributes with the other instances of the cluster, highlighted with a gray background. Afterwards, it is flushed out of the buffer and a new instance is received.

The idea of this partition procedure, formally explained in Algorithm 6.3, is to calculate, for all instances not yet anonymized, the distance to the target (τ), keeping track of the $k - 1$ nearest ones. If an instance is closer than the current furthest, the latter is removed from the cluster and the former is added. At the end of the buffer traversal, together with τ, the instances that have been kept will form the next cluster of the stream partition. The distance metric used is the same than that used by the default disclosure risk estimator (see Section 6.3.1 and Procedure 6.1).

The KNN algorithm uses a *priority queue*[10] to hold the $k - 1$ nearest neighbours to the target, τ. The queue works over *distance-instance* pairs $\langle d, i \rangle$, but is actually indexed by d, holding the greatest value on the top of the queue. This way, it is very cheap (constant time) to know whether a given instance x is closer than the current furthest instance from τ. Insertions are also cheap, with an upper bound cost of $O(\log(k))$.

The overall cost of the procedure, when implemented with a priority queue, depends on the amount of instances n being processed, the size b of the sliding window and the size k of the clusters and its upper bound is $O(n \cdot b \cdot \log(k))$.

[10] A *max-heap* is used to keep the greatest element on top of the queue.

Algorithm 6.3: KNN-based Clustering

```
Data: W', A
Result: a cluster C of k instances
begin
    τ ← w'_0;
    C ← ∅ ∪ τ;
    Q ← PriorityQueue⟨DistanceInstancePair⟩();
    for x ∈ W', x ∉ A do
        d ← dist(x, τ);
        p ← DistanceInstancePair(d, x);
        if |Q| < k then
            Q ← Q ∪ p;
        else
            if p < Q.peek().distance() then
                Q.poll();
                Q ← Q ∪ p;
    for q ∈ Q do
        C ← C ∪ q.instance();
    return C;
end
```

6.5.1.3 Aggregation

After a cluster has been obtained from the previous partition step, the instances of the cluster are aggregated, this is, the values of their attributes are imputed with the values of the *centroid* of the cluster. For each attribute, the arithmetic mean (in the case that the attribute is numeric) or the mode (if the attribute is nominal) are calculated over the instances of the cluster. We do not provide any figure or algorithm concerning this step, due its simplicity.

The computational cost of the aggregation step is directly related to the size k of the clusters and can be approximated to $\Theta(n \cdot 2 \cdot m \cdot k)$, where n is the number of instances processed and m is the number of attributes. This is: for each of the m attributes, a first traversal over the k instances in the cluster is done to compute the averages and a second one to impute the values of the centroid found. If we add together both steps, we found the total cost of the algorithm: $O(n \cdot (b \cdot \log(k) + 2 \cdot m \cdot k))$. However, given that $m \ll n$ and $k \ll b, n$, the overall cost of this microaggregation implementation is actually dominated by the cost of the clustering step: $O(n \cdot b \cdot \log(k))$.

6.5.2 Summary

The `MicroAggregationFilter` implements a microaggregation algorithm based on a KNN clustering for the partition step and a basic centroid aggregation scheme. Figure 6.9 shows a complete execution for a given target instance and Table 6.3 summarizes

the main properties of this SDC method.

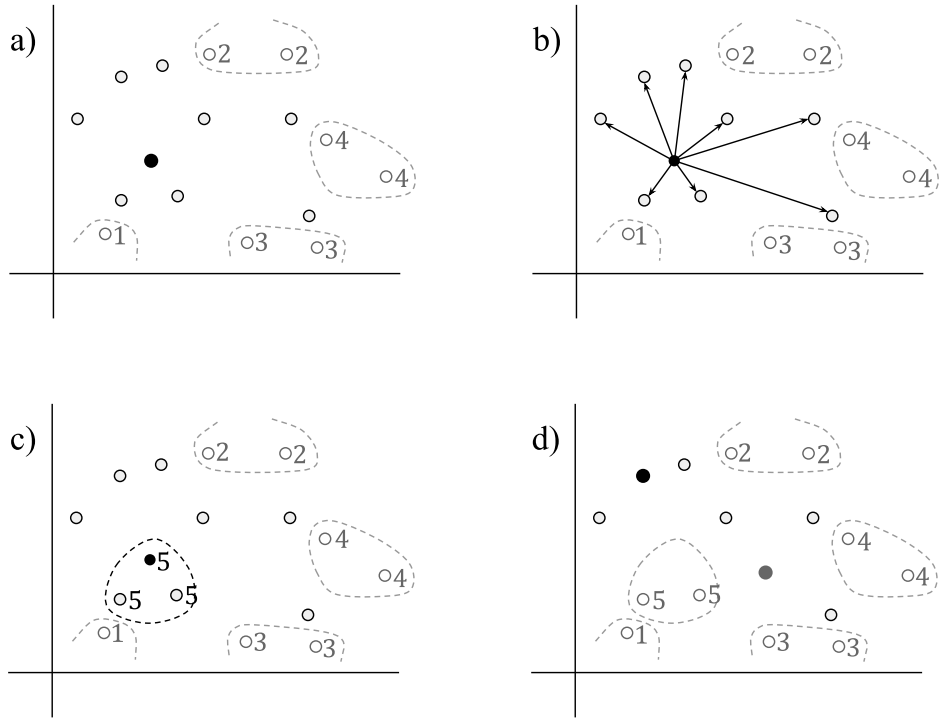

FIGURE 6.9: KNN-based microaggregation ($k = 3$, $b = 16$). Firstly, the target instance (black) is not anonymized (a). Distances to the remaining non-anonymized instances are calculated (b) and a cluster is formed with the $k-1$ nearest ones (c). After aggregating the records in the cluster, the target instance is streamed out, a new instance (grey) is received and a new target is selected (d).

MicroAggregationFilter	
Parameters	k (cluster size), b (buffer size)
Type of data	Heterogeneous (both numeric and categorical attributes)
Cost	$O(n \cdot b \cdot \log(k))$

TABLE 6.3: `MicroAggregationFilter` summary.

6.6 `RankSwappingFilter`

The rank swapping SDC method described in Section 2.3.3 is implemented by the `RankSwappingFilter` class. We must notice that it is a very naïve implementation and certainly not the fastest of the filters. As we will discuss later, future work is needed to enhance the performance of this filter.

6.6.1 Design

The `RankSwappingFilter` is the second *buffered filter* that has been implemented in this project (see Section 6.5.1.1). Almost the same data structures (the W, W' and A buffers)

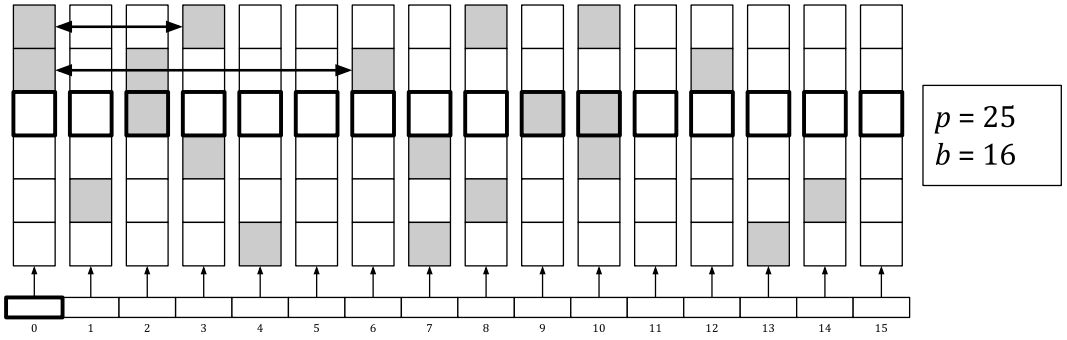

FIGURE 6.10: Rank swapping algorithm schematic. Two attributes of the target record (position 0) have already been *rank swapped* with other values from instances in the buffer. The variable being now processed is highlighted.

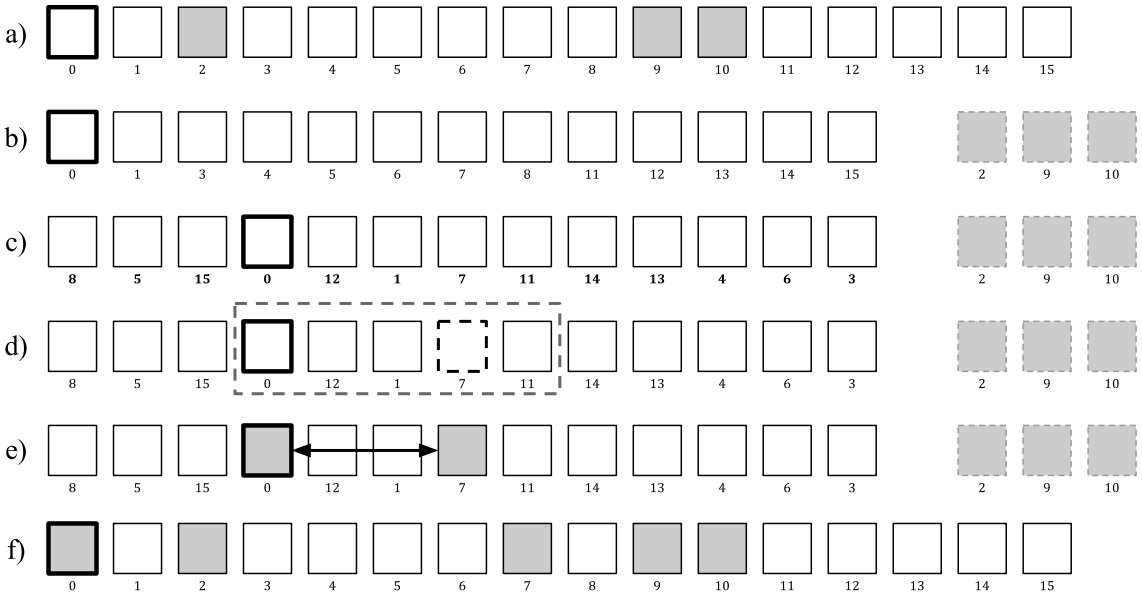

FIGURE 6.11: Rank swap of a single attribute for a target instance τ. First, the non already swapped values of the attribute are filtered from the instances in the buffer W (b) and are ranked, i.e., sorted (c). A maximum swap range is calculated using the p parameter (d) and a value within this range is selected to perform the swap (e). Finally, the vector of values is returned in the original order they were in the buffer (f).

used by the `MicroAggregationFilter` are used by the rank swapping algorithm. The main difference with respect to the microaggregation algorithm is that the set A, used to know whether or not an *instance* has already been anonymized, is now used to know whether or not a *value* w'_{ij} (this is, the j-th attribute value of the i-th instance in W') has been *swapped* or not. Summarizing, we say that w'_{ij} has been swapped if $w'_{ij} \in A$.

The design of this SDC method follows almost exactly the explanation given in the theoretical background chapter (Section 2.3.3): for each instance processed from the stream, values of each variable j are ranked in ascending order, this is, they are *sorted*.

Each ranked value is then swapped with another ranked value, randomly chosen within a restricted range, controlled by the input parameter p, which denotes that swapped values cannot differ more than $p\%$ of the total number of records. A more formal description of its implementation is given in Algorithm 6.4, along with its auxiliar procedure `selectSwap()` (see Procedure 6.5). Also, a more visual explanation is shown in Figure 6.11.

Algorithm 6.4: Rank Swapping

Data: W', A, p

Result: the target instance τ is anonymized

begin

- $\tau \leftarrow w'_0$;
- **for** $j \in$ attributes(τ) **do**
  - $\gamma \leftarrow$ selectSwap(W', p, j);
  - $\sigma \leftarrow w'_\gamma$;
  - swap(τ_j, σ_j);
  - $A \leftarrow A \cup \tau_j \cup \sigma_j$;

end

Procedure `selectSwap`(W', p, j)

Data: W' buffer, p parameter and j attribute

Result: the index γ of the instance with which the swap will be done

begin

- $V \leftarrow \text{Vector}(\{\langle w'_{ij}, i\rangle \mid 0 \leq i \leq b-1,\ w'_{ij} \notin A\})$;
- `// Notice that, by construction,` $\langle \tau_j, 0\rangle \in V$
- $V^* \leftarrow \text{sort}(V)$ `// sort by` ***value***`, not by index`
- $t \leftarrow V^*.index(\langle \tau_j, 0\rangle)$;
- $r \leftarrow 1 + \text{Random}.uniform()$ mod $(p \cdot b \ / \ 100)$ `//` $r \in [1, p\% \cdot b]$
- $s \leftarrow 0$;
- **if** $t + r < V^*.size()$ **then**
  - $s \leftarrow t + r$;
- **else**
  - $s \leftarrow V^*.size() - 1$;
- `//` s `is the index of the selected value-index pair to be swapped`
- $\langle \cdot, \gamma\rangle \leftarrow V^*[s]$;
- **return** γ;

end

If we examine the previous rank swapping algorithm in detail, we can estimate its computational complexity. The cost of swapping a value of an attribute is, basically, that of sorting all of its values: $O(b \cdot \log(b))$, where b is the size of the buffers W and W'. Each of the n instances of the stream will have its m attributes rank swapped with those

of another record; therefore, the total complexity of the algorithm can be approximated to be $O(n \cdot m \cdot b \cdot \log(b))$.

6.6.2 Summary

The `RankSwappingFilter` class implements rank swapping algorithm to anonymize streaming data by exchanging values of the same attribute between close instances. Table 6.4 summarizes the main properties of this SDC method.

RankSwappingFilter	
Parameters	p (maximum swap range, as a percentage of the buffer size), b (buffer size)
Type of data	Heterogeneous (both numeric and categorical attributes)
Cost	$O(n \cdot m \cdot b \cdot \log(b))$

TABLE 6.4: `RankSwappingFilter` summary.

6.7 `DifferentialPrivacyFilter`

The concept of *differential privacy* is introduced in Section 2.2.5.4 and on Section 2.3.4 a data release *mechanism* that achieves this privacy guarantee is described: the Laplace mechanism. The drawback of differential privacy is that its definition relies on an interactive *query-response* environment, which is definitely not the one we encounter in stream data mining. However, we can find in the literature some efforts to bring differential privacy to non-interactive settings, such as in Leoni [21] and Soria-Comas et al. [14]. The `DifferentialPrivacyFilter` is devised to provide a differentialy private release method in such a setting, namely, in the context of MOA privacy filters.

6.7.1 Design

Following the idea presented in Soria-Comas et al. [14], we have built an SDC method which combines microaggregation with the Laplace mechanism. We recall now the definition of this mechanism:

Definition 6.1. (Laplace mechanisnm)
Given a dataset X and a function $f : X \to \mathbb{R}^d$, with $w \in \mathbb{N}^+$, an ε-differential privacy mechanism $\mathcal{M}$ for releasing f is to publish

$$\mathcal{M}(X) = f(X) + L$$

where L is a vector of d random variables each drawn from a Laplace distribution $Lap(0, \frac{\Delta(f)}{\varepsilon})$.

We must remember that the amount of noise introduced by the addition of L to the application of f depends on the *sensitivity* of f, denoted by $\Delta(f)$, this is, the maximum variation in the result of f when computed over two neighbour datasets, i.e., sets differing in at most one record. For a fixed ε, the higher the sensitivity of f, the more noise is added.

Let $I_r(X)$ be the function that returns the attribute values corresponding to the r-th record (instance) of a stream X, this is, the "identity" function that returns instances from a stream. It is clear that I_r, formally defined as

$$\begin{aligned} I : X \times \mathbb{N}^+ &\to \mathbb{R}^d \\ (X, r) &\mapsto (x_{r1}, x_{r2}, ..., x_{rd}) \end{aligned} \tag{6.5}$$

where $d \in \mathbb{N}^+$, X a *dataset* and $r \in \mathbb{N}$, is a good candidate to be fed into the Laplace mechanism to obtain an ε-differential private data release method.

The idea is now to compose I_r with a microaggregation function M, this is, $I_r \circ M$ in order to reduce the sensitivity of the results, thus increasing the analytical utility of data released by the mechanism $\mathcal{M}(X) = (I_r \circ M)(X) + L$. If we are able to lower the sensitivity of the function captured by the Laplace mechanism, the information loss due to the noise added will also be lower.

As we will see in Section 6.7.1.3, the `DifferentialPrivacyFilter` implements the mechanism $\mathcal{M}$ described in the previous paragraph using the $I_r \circ M$ composition.

6.7.1.1 Insensitive microaggregation

Soria-Comas et al. [14] prove that, by using an *insensitive microaggregation* function M, the global sensitivity of its composition with I_r is $\Delta(I_r \circ M) \leq \Delta(I_r)/k$, being k the minimum size of the clusters returned by M. The condition that such an *insensitive* algorithm must fulfill is:

Definition 6.2. (Insensitive microaggregation [14])
Let X be a dataset, M a microaggregation algorithm, and let $\{C_1, ..., C_n\}$ be the set of clusters that result from running M on X. Let X^* be a neighbour dataset of X, differing in a single record, and $\{C_1^*, ..., C_n^*\}$ the clusters that result from running M on X^*. We say that M is insensitive to the input data if there is a bijection between the set of clusters $\{C_1, ..., C_n\}$ and the set of clusters $\{C_1^*, ..., C_n^*\}$ such that each corresponding pair of clusters differs at most in a single record.

Microaggregation algorithms are, however, very sensitive to the input data, this is, concerning the previous definition, they mostly do not accomplish it, because a minimum

change in a single record can cause the generation of completely different clusters. In order to correct this behaviour, Soria-Comas et al. [14] prove that the design of an insensitive microaggregation algorithm is possible by using a an *order relation consistent* distance metric in the partition step.

Definition 6.3. (Order relation consistent distance [14])
A distance function $d : X \times X \to \mathbb{R}$ is said to be consistent with a order relation $\leq_X$ if $d(x, y) \leq d(x, z)$ whenever $x \leq_X y \leq_X z$.

One way to achieve such a consistent distance function is to define a total order relation among the elements of a dataset X as follows: given a *reference point* $R \in X$, for a pair of elements $x, y \in X$, we say that $x \leq y$ if $d(R, x) \leq d(R, y)$, where d is a function such that $d : Dom(X) \times Dom(X) \to \mathbb{R}$ (the Euclidean distance between records of X, for example). Furthermore, in order to increase the *within-cluster* homogeneity (see Section 2.3.2), this reference point R should be located at the boundaries of $Dom(X)$.

6.7.1.2 Sensitivity estimation

In the previous section, we discussed how to achieve a reduction in the sensitivity of I_r by composing it with an insensitive microaggregation function such that the following result holds: $\Delta(I_r \circ M) \leq \Delta(I_r)/k$.

The problem, however, remains in determining the actual sensitivity of I_r. By definition of sensitivity, the maximum change that occurs in I_r, as a result of a single record being different in the dataset X on which I_r is applied, can be estimated as the range of $Dom(X)$, when the attributes of X are numerical. For example, if an attribute a in a dataset represents the height of a person, and $Dom(a) = [a_{min}, a_{max}]$, the difference in the result of I_r that the presence or abscence of an individual in this dataset causes is bounded by the range $[a_{min}, a_{max}]$. This is, however, a very naïve estimation, because it assumes that this attribute values in the dataset, are representative of the attribute values of the *population*. Particularly, it means that the population outliers are represented in the dataset.

Despite being a very rough estimate, we will use it to scale the amount of noise added to the data that will eventually be released. More accurately, we will estimate the sensitivity of an attribute j for the function I_r over a dataset X as

$$\Delta_j\big(I_r(X)\big) = 1.5 \times \Big(\max\big(Dom(X_j)\big) - \min\big(Dom(X_j)\big)\Big) \tag{6.6}$$

Notice that a scaling factor of 1.5 is applied, in order to make the estimation more reasonable. The idea of this estimation was also drawn from Soria-Comas et al. [14].

6.7.1.3 Putting it all together

Given a dataset X with m attributes, the `DifferentialPrivacyFilter` class implements the Laplace mechanism

$$\mathcal{M}(X) = (I_r \circ M)(X) + L$$

where M is an insensitive k-microaggregation function and L is a vector of random variables l_j, for $1 \leq j \leq m$, each drawn from a Laplace distribution $Lap_j(0, b_j)$, with b_j being the scale parameter, estimated by

$$b_j = \frac{\Delta_j\big(I_r(X)\big)}{\varepsilon}$$

A general and high level description of the complete mechanism, adapted to a streaming environment, is given in Algorithm 6.6. Notice that a single instance is processed in the algorithm pseudo-code: the privacy filter executes the mechanism for each instance in the input data stream. All the involved classes and modules of the actual implementation can be seen in Figure 6.12. The general cost of this method can be approximated to the same as the `MicroAggregtionFilter`: $O(n \cdot b \cdot \log(k))$.

Algorithm 6.6: Microaggregation-based Laplace Mechanism

Data: an instance x from a stream, an insensitive microaggregation function M and a Laplace noise adder $\mathcal{L}$
Result: an anonymized instance x'
begin
  $\mu \leftarrow M(x)$;
  $x' \leftarrow \mathcal{L}(\mu)$;
  return x';
end

The adaptation of the insensitive microaggregation algorithm to a stream processing environment follows the same scheme presented for the `MicroAggregationFilter` (see Section 6.5.1), the only difference being the use of a *reference* point in order to achieve a total ordering relation between the instances of the stream and, thus, fulfilling the insensitivity condition. The reference "point", denoted by $\mathcal{R}$, is incrementally[11] built as *new* instances are processed by the filter, this is, it is updated independently of the

[11] Remember that, in a stream processing environment, and more precisely, in the context of a *buffered filter* (see Section 6.5.1.1), only a portion of the complete dataset (the stream) is visible to the algorithm at a given moment.

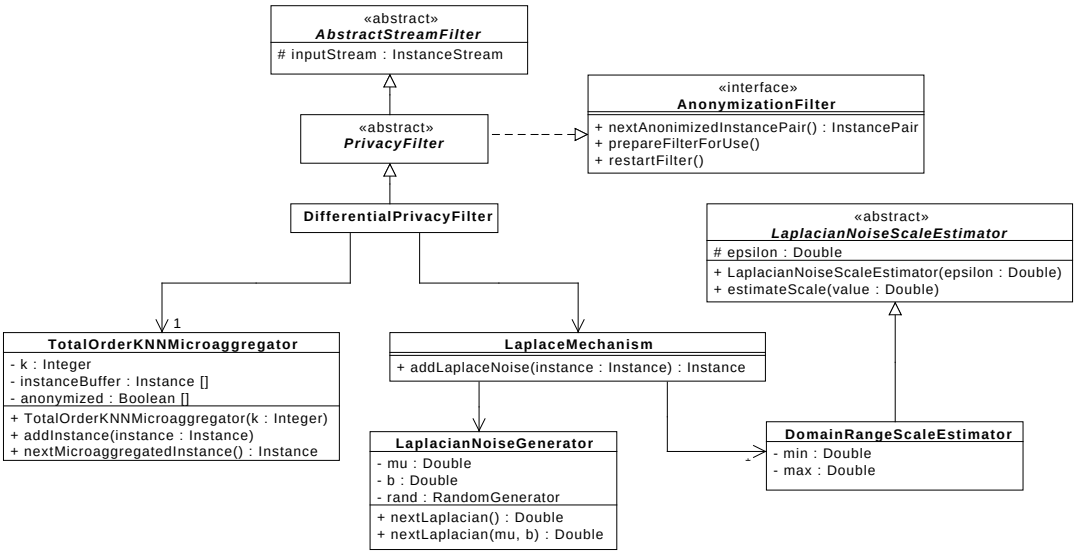

FIGURE 6.12: `DifferentialPrivacyFilter` and the related classes and types it uses. Notice that the `LaplaceMechanism` uses many noise scale estimators, one for each attribute in the stream.

clustering step, when a new instance is added to the buffer. The necessary modifications are presented in Algorithm 6.7.

Algorithm 6.7: KNN-based Insensitive Clustering

Data: W', A buffers and $\mathcal{R}$, the current reference point
Result: a *cluster* $\mathcal{C}$ of k instances
begin
  $\mathcal{C} \leftarrow \emptyset$;
  $Q \leftarrow \text{PriorityQueue}\langle\text{DistanceInstancePair}\rangle()$;
  for $x \in W', x \notin A$ **do**
    $d \leftarrow \text{dist}(x, \mathcal{R})$;
    $p \leftarrow \text{DistanceInstancePair}(d, x)$;
    if $|Q| < k$ **then**
      $Q \leftarrow Q \cup p$;
    else
      if $p < Q.peek().distance()$ **then**
        $Q.poll()$;
        $Q \leftarrow Q \cup p$;
  for $q \in Q$ **do**
    $\mathcal{C} \leftarrow \mathcal{C} \cup q.instance()$;
  return $\mathcal{C}$;
end

The Laplace-distributed noise addition step of the mechanism is performed by a noise adder (called $\mathcal{L}$ in Algorithm 6.6) that works in a very similar fashion to the `NoiseAdditionFilter`,

with the addition of the scale parameter estimation, already discussed before. The complete description of the procedure is given in Algorithm 6.8 and, finally, the generation of a random variable Λ following a Laplace distribution is shown in the equation below:

$$\Lambda \sim Lap(\mu, b) \iff \Lambda = \mu - b \operatorname{sgn}(U) \ \ln(1 - 2|U|) \tag{6.7}$$

where U is another random variable drawn from a uniform distribution constrained to the $(-0.5, 0.5]$ interval.

Algorithm 6.8: Laplace Noise Adder

Data: an instance x and a vector B of scale estimators
Result: an anonymized instance x'
begin
  for $i \in$ attributes(x) **do**
    $b \leftarrow B_i.estimate(x_i)$;
    $x'_i \leftarrow x_i + \text{Random}.laplace(0, b)$;
  return x';
end

6.7.2 Summary

The `DifferentialPrivacyFilter` class implements a microaggregation-based Laplace mechanism to achieve ε-differential privacy sanitization of released data. Table 6.5 summarizes the main properties of this SDC method.

DifferentialPrivacyFilter	
Parameters	k (cluster size), ε (differential privacy), b (buffer size)
Type of data	Numerical data only
Cost	$O(n \cdot b \cdot \log(k))$

TABLE 6.5: `DifferentialPrivacyFilter` summary.

Chapter 7

Benchmarking

This chapter shows the benchmarks that have been performed with the privacy filters which implementation was detailed in Chapter 6. A brief introduction is given to the software and hardware resources involved in the benchmark process in 7.1. The final results are displayed in Section 7.2.

7.1 Experimental setup

A simple experimental setup has been used to assess the performance of the MOA privacy filters in terms of *disclosure risk* and *information loss*. To do so, a MOA *task*[1] was specifically designed to retrieve the results of the evaluation measures of the filters.

7.1.1 MOA generators

In order to test the filters, streams of synthetically generated data have been used, rather than actual datasets, mainly because, this way, we avoid the complex and time consuming preprocessing of real datasets.

The MOA framework offers a rich set of stream data generators, of which we chose the `RandomRBFGenerator` and the `WaveformGenerator`. Both streams consist of numerical variables only. Even though most of the implemented filters are capable of dealing with heterogeneous data, some previous tests had shown that the information loss evaluation, based on the SSE estimation, pondered too much categorical attributes differences, thus getting disturbed IL measurements.

[1] Within the MOA framework, tasks define a procedure to run as the main program, like a classifier or regression learning task.

The `RandomRBFGenerator` outputs a stream of 10 attributes and 1 class variable, drawing values for those attributes from radial basis functions (RBFs). The class variable is indeed a categorical one, indicating to which RBF an instance belongs, this is, the intended machine learning task of this generator is *classification*.

The `WaveformGenerator` generates values by combining two or three base wave functions, which form a numerical stream of 21 attributes and 1 class variable. The machine learning task intended for this generator is, again, *classification*.

7.1.2 Experimental design

The experiments undertaken during the benchmarking stage of the project consist in generating streams of synthetic data and pipe those streams through each of the privacy filters, for each of the parameters permutations that we decided, taking 100000 instances from the generators. A summary of paramaters values used in those experiments is shown in Table 7.1.

	Parameters		
Privacy filter	**Name**	**Range**	**Selected values**
`NoiseAdditionFilter`	a	$[0, 1] \in \mathbb{R}$	0.1, 0.25, 0.5, 0.75, 1.0
	c	$[0, 1] \in \mathbb{R}$	0.0
`MicroAggregationFilter`	b	$\mathbb{N}^+$	100, 250, 500, 1000
	k	$\mathbb{N}^+,\ k \leq b$	3, 5, 10, 15, 20, 25, 50, 100
`RankSwappingFilter`	b	$\mathbb{N}^+$	100, 250, 500, 1000
	p	$[1, 100] \in \mathbb{N}^+$	10, 25, 50, 75, 80
`DifferentialPrivacyFilter`	b	$\mathbb{N}^+$	100, 250, 500, 1000
	k	$\mathbb{N}^+,\ k \leq b$	3, 5, 10, 15, 20, 25, 50, 100
	ε	$\mathbb{R}^+$	0.01, 0.1, 1, 10, 100

TABLE 7.1: Privacy filters benchmark parameterization.

The optimal situation would have been to execute each filter configuration (each parameters permutation) a number of times, in order to statistically validate the results. However, due to the lack of time and resources available, a single execution of each filter was performed during the benchmark phase.

7.1.3 Hardware

The executions of the filters were executed in the following hardware environment[2]:

Category	**Description**	
Computer model	Asus k53-SV	
CPU	Model	Intel(R) Core(TM) i5-2430M, 64 bit
	Frequency	2.40GHz
	Cores	2 (4 *virtual* threads available)
	Cache	32KB data L1
		32KB instructions L1
		256KB L2 per core
		3072KB shared L3
Memory	Capacity	8GB
	Frequency	1333MHz

TABLE 7.2: Hardware benchamark setting.

7.1.4 Software

The following software was involved in the execution of the privacy filters benchmarks:

- **Operating System:** Ubuntu 12.04 LTS with the Linux kernel 3.2.0-80 version
- **Java:** Java 1.7.0_75 (OpenJDK Runtime Environment, version IcedTea 2.5.4)
- **Python:** Python 2.7 (to execute the scripts running the actual MOA tasks)

7.2 Results

We provide now the results of the execution of the privacy filters, with the configuration specified in the previous section. For each filter, we present summary tables with the disclosure risk and information loss estimates for each parameterization, as well as plots showing the evolution of both measurements against the number of instances processed.

[2]Only the relevant specifications to the execution of the filters are included.

7.2.1 Noise addition

All the considered parameterizations for the `NoiseAdditionFilter` are reflected on Table 7.3 and Table 7.4. The evolution of disclosure risk against the number of instances processed is shown in Figure 7.1 and information loss is shown in Figure 7.2.

a	c	DR	IL
0.1	0.0	0.956	1123.97
0.25	0.0	0.889	7024.83
0.5	0.0	0.774	28099.33
0.75	0.0	0.596	63223.50
1.0	0.0	0.430	112397.34

TABLE 7.3: Noise addition IL & DR estimations for all considered parameterizations with the `RandomRBFGenerator`.

a	c	DR	IL
0.1	0.0	1.000	50741.41
0.25	0.0	1.000	317133.83
0.5	0.0	0.996	1268535.34
0.75	0.0	0.919	2854204.53
1.0	0.0	0.742	5074141.39

TABLE 7.4: Noise addition IL & DR estimations for all considered parameterizations with the `WaveformGenerator`.

As was anticipated in the theoretical introduction chapter, the `NoiseAdditionFilter` is not able to protect data as much as other methods do. The results of its application on the `RandomRBFGenerator` data prove to be much better than on the `WaveformGenerator`, on which has almost no effect at all. Even the high amount of noise introduced, it is not really protecting data against disclosure — although the a control parameter of the filter is set to its maximum value ($a = 1$), the disclosure risk is quite high.

The effect of the noise scaling factor controlled by the a parameter can cleary be seen on Figure 7.2: the higher the value of the parameter, the higher the information loss. Another important thing to notice is that the amount of noise added for each instance remains constant, this is, the increment in the total information loss measure grows linearly.

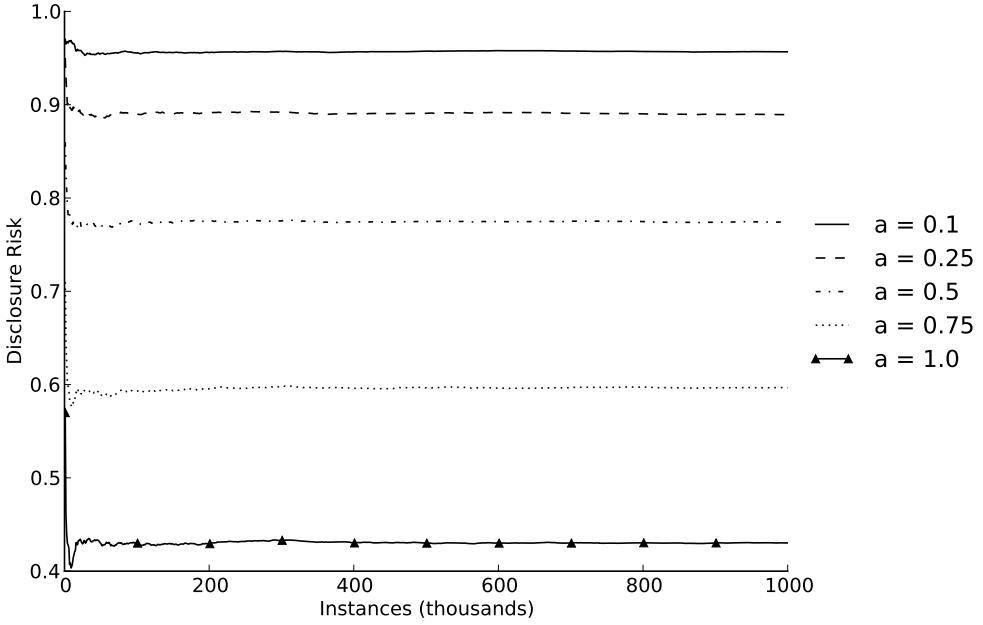

FIGURE 7.1: `NoiseAdditionFilter` DR evaluation using the `RandomRBFGenerator` with fixed parameter $c = 0$.

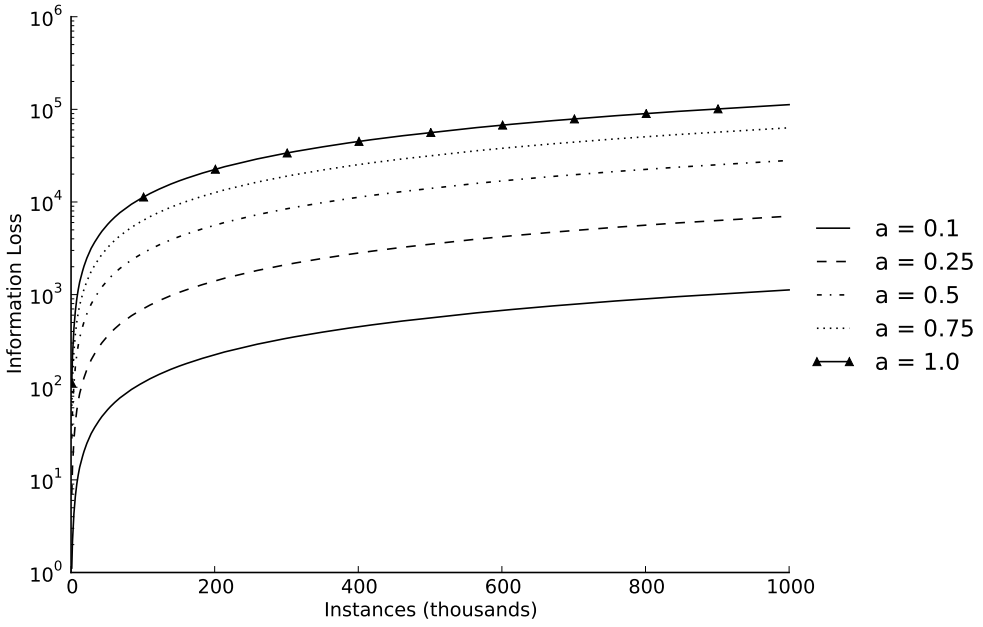

FIGURE 7.2: `NoiseAdditionFilter` IL evaluation using the `RandomRBFGenerator` with fixed parameter $c = 0$, on a logarithmic scale.

7.2.2 Microaggregation

Tables 7.5 and 7.6 show the final values for both information loss and disclosure risk for increasing values of the cluster size of the partition step of the microaggregation algorithm (the k parameter) and a fixed historical buffer size of $b = 100$. Additional tables are provided in Appendix A for the rest of the considered buffer sizes. The evolution of disclosure risk against the number of instances processed for the `MicroaggregationFilter` is shown in Figure 7.3 and information loss is shown in Figure 7.4.

b	k	DR	IL
100	3	0.232	74917.30
100	5	0.144	89953.68
100	10	0.089	101193.72
100	15	0.069	104952.79
100	20	0.061	106800.51
100	25	0.055	107937.42
100	50	0.041	110179.72
100	100	0.030	111313.78

TABLE 7.5: Microaggregation IL & DR estimations for increasing k (cluster size) and fixed buffer size $b = 100$ with the `RandomRBFGenerator`.

b	k	DR	IL
100	3	0.241	3375656.78
100	5	0.135	4057649.13
100	10	0.076	4566550.54
100	15	0.061	4739204.47
100	20	0.054	4822716.29
100	25	0.049	4872587.52
100	50	0.038	4976718.32
100	100	0.029	5027574.64

TABLE 7.6: Microaggregation IL & DR estimations for increasing k (cluster size) and fixed buffer size $b = 100$ with the `WaveformGenerator`.

As shown in the tables above, the `MicroaggregationFilter` disclosure risk performance is almost the same for both streams (unlike the `NoiseAdditionFilter`). The total disclosure risk diminishes as the size of the clusters formed increases, which is a direct consequence of the theoretical privacy guarantee that microaggregation offers: because k-anonymity is implemented through this algorithmic scheme, the maximum disclosure risk for a given k is $DR \leq 1/k$.

The couple of figures included below show an interesting result of the microaggregation filter. Even though the size of the clusters does not affect too much the amount of noise introduced in the data (see Figure 7.4), the increase of the of the k parameter offers much better privacy protection than that achieved with lower values. We can anonymize data, obtaining good (low) disclosure risk and not losing too much utility in the process with respect to configurations that yield poorer disclosure risk results.

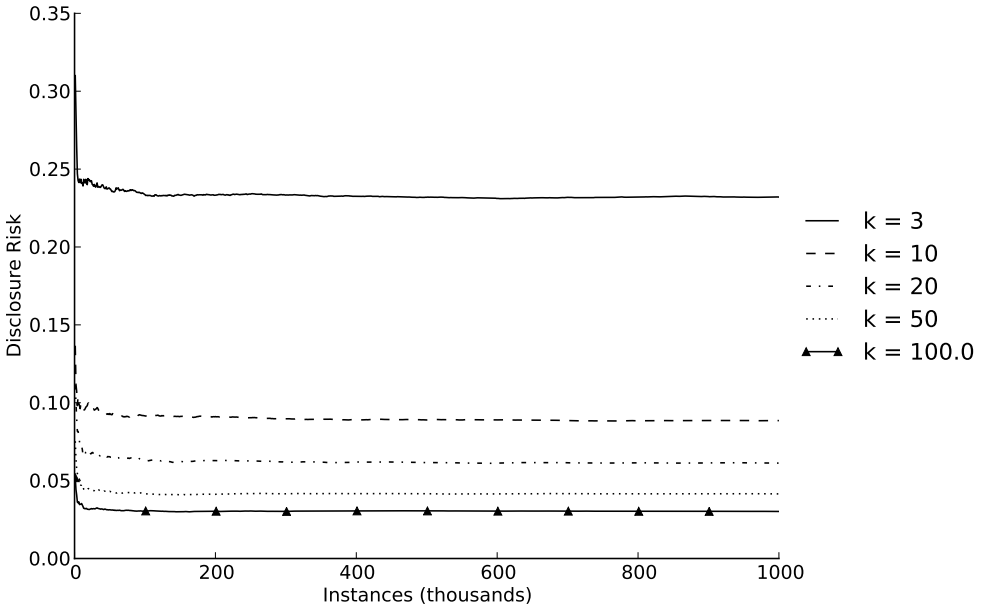

FIGURE 7.3: `MicroaggregationFilter` DR evaluation using the `RandomRBFGenerator` with fixed buffer size $b = 100$, for increasing k.

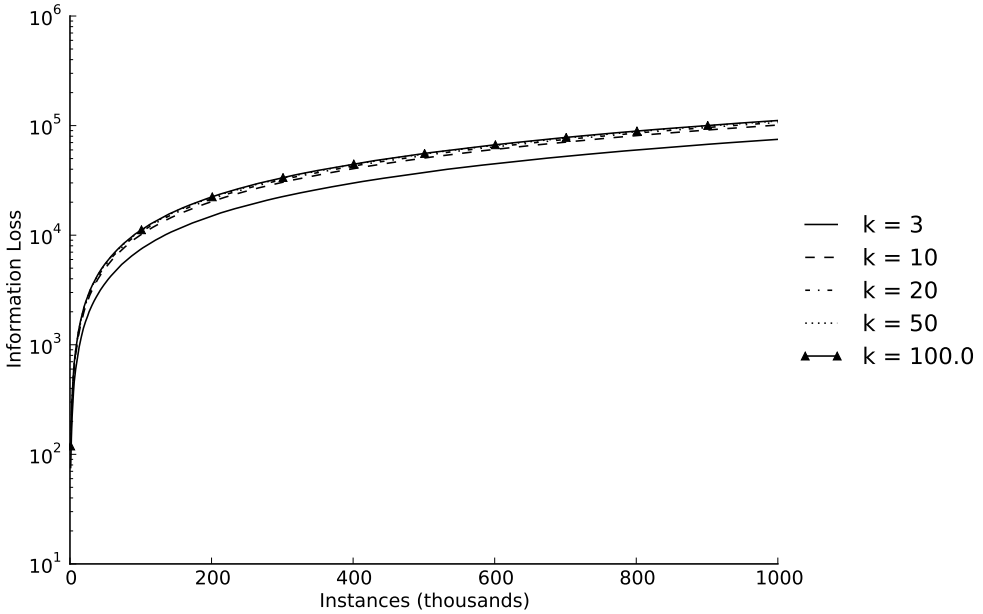

FIGURE 7.4: `MicroaggregationFilter` IL evaluation using the `RandomRBFGenerator` with fixed buffer size $b = 100$, for increasing k, on a logarithmic scale.

7.2.3 Rank swapping

A comparison of the performance of the `RankSwappingFilter` on the streams produced by the `RandomRBFGenerator` and `WaveformGenerator` can be seen in Table 7.7 and Table 7.8. The remaining tables of results can be consulted in the Appendix A. Figure 7.5 shows the disclosure risk estimation against the number of instances processed, for increasing values of the p parameter and two buffer sizes ($b = 100$ and $b = 500$). The information loss estimate of the corresponding parameterizations is shown in Figure 7.6.

b	p	DR	IL
100	10	0.838	19136.62
100	25	0.469	57753.05
100	50	0.070	136791.17
100	75	0.036	178200.11
100	80	0.037	178685.81

TABLE 7.7: Rank swapping IL & DR estimations for increasing p (maximum swap range) and fixed buffer size $b = 100$ with the `RandomRBFGenerator`.

b	p	DR	IL
100	10	0.997	705196.07
100	25	0.776	2464673.56
100	50	0.112	6195304.18
100	75	0.046	8088473.27
100	80	0.045	8097619.70

TABLE 7.8: Rank swapping IL & DR estimations for increasing p (maximum swap range) and fixed buffer size $b = 100$ with the `WaveformGenerator`.

As we see in both Table 7.7 and Table 7.8, the maximum swap range, defined as the percentage p of the size of the buffer, is indeed a key factor to achieve good results, concerning the disclosure risk of the anonymized data. Both tables show that, for swap ranges shorter than half the size of the window, the risk or fe-identification is too high.

The remaining parameter of this filter, the size of the buffer, is also, in this case, an important factor to leverage when using the `RankSwappingFilter`. For bigger buffer sizes, the information loss incurred decreases, but the disclosure risk grows too much. The cause behind this increase in the revelation risk could be the usage of synthetical data: for bigger buffer sizes, because generated attribute values are sensibly close to each other, the difference between the swapped values is smaller, thus generating lower noise and exposing more information to an attacker.

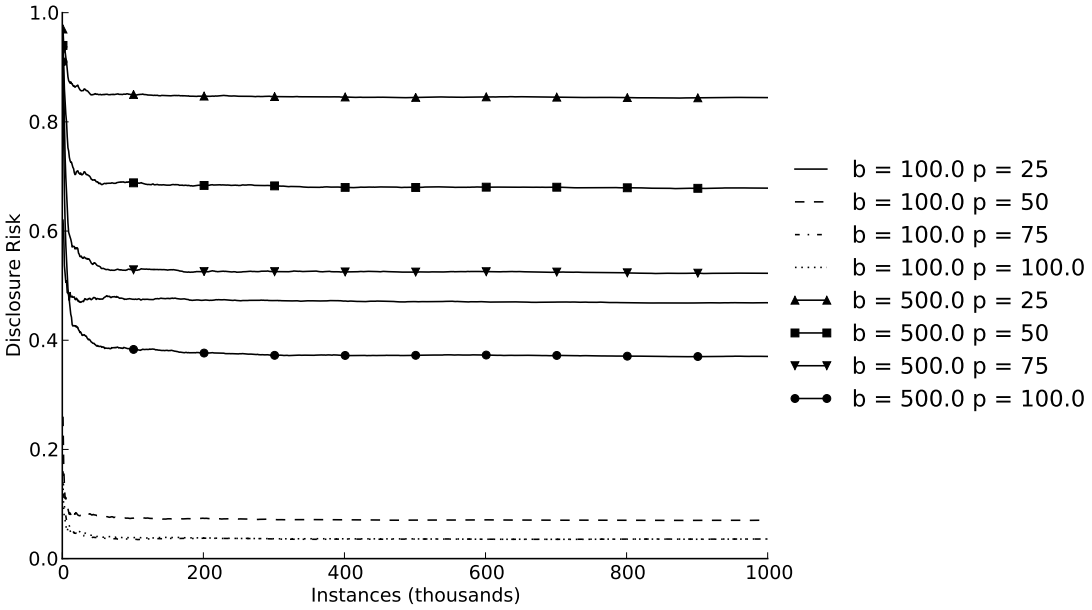

FIGURE 7.5: `RankSwappingFilter` DR evaluation using the `RandomRBFGenerator` with buffer sizes $b = 100$ and $b = 500$, for increasing p.

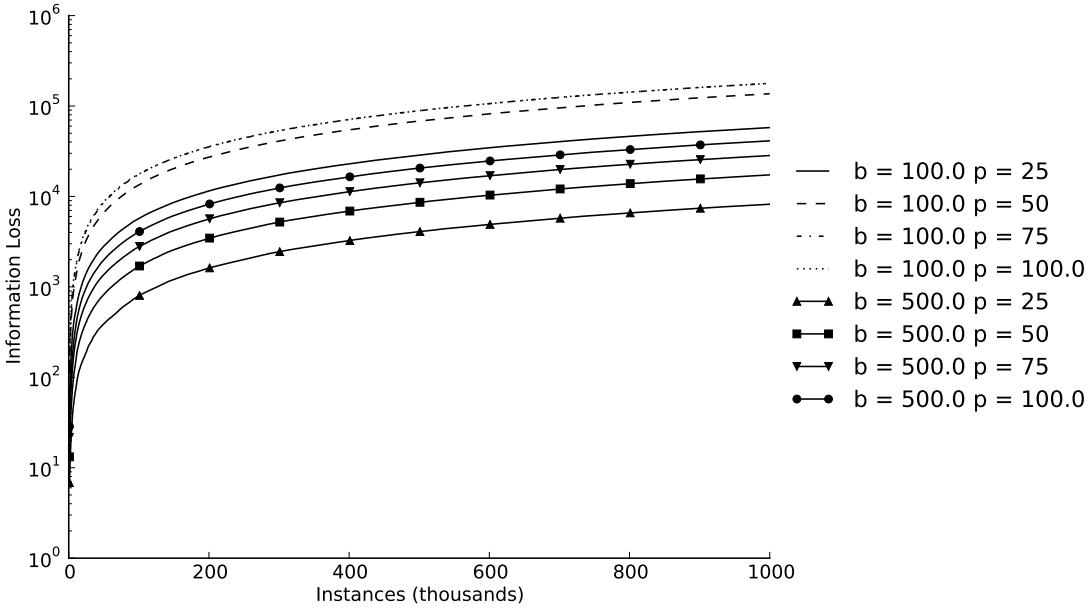

FIGURE 7.6: `RankSwappingFilter` IL evaluation using the `RandomRBFGenerator` with buffer sizes $b = 100$ and $b = 500$, for increasing p, on a logarithmic scale.

7.2.4 ε-Differential private microaggregation

An aggregate of five "subtables" is shown in Table 7.9: the disclosure risk and information loss results are assessed for increasing values of cluster size k, increasing differential privacy scale factors, controlled by the ε parameter, and fixed historical buffer size $b = 250$. The complete set of results tables is located in Appendix A.

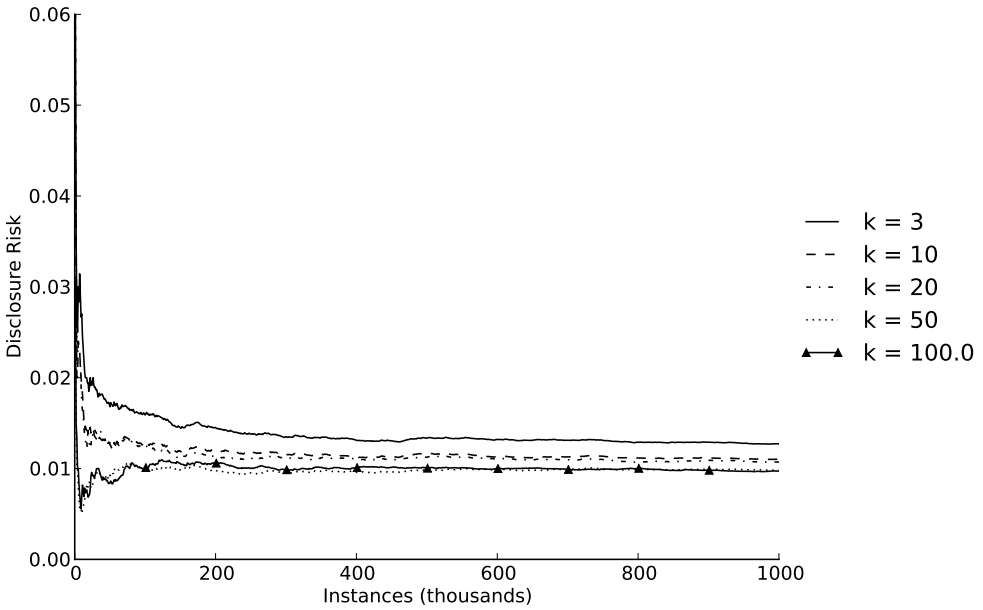

FIGURE 7.7: `DifferentialPrivacyFilter` DR evaluation using the `RandomRBFGenerator` with fixed buffer size $b = 100$ and fixed differential privacy scale parameter $\varepsilon = 1$, for increasing k.

The `DifferentialPrivacyFilter` performs really well in terms of disclosure risk, for any of the parameters combinations displayed. However, for small values of ε, the amount of noise added to data is really high. Confirming the hipothesis presented in Section 6.7.1, increasing the size of the clusters k achieves a reduction in the sensitivity of the release function of the data, thus reducing the Laplacian noise introduced, thus reducing information loss. This effect can be seen for values of ε up to 10, but is not visible for $\varepsilon = 100$, because, for such a high ε, the majority of the added noise is indeed caused by the microaggregation function, not by the Laplace mechanism. This effect can also be assessed in Figure 7.9, where the value of ε is fixed and k increases, effectively reducing information loss.

b	ε	k	DR	IL
250	0.01	3	0.004	3.298E+11
250	0.01	5	0.004	2.483E+11
250	0.01	10	0.004	2.033E+11
250	0.01	15	0.005	1.885E+11
250	0.01	20	0.004	1.880E+11
250	0.01	25	0.004	1.881E+11
250	0.01	50	0.005	1.502E+11
250	0.01	100	0.005	1.050E+11

b	ε	k	DR	IL
250	0.1	3	0.005	3.298E+09
250	0.1	5	0.005	2.483E+09
250	0.1	10	0.004	2.033E+09
250	0.1	15	0.005	1.885E+09
250	0.1	20	0.005	1.880E+09
250	0.1	25	0.004	1.881E+09
250	0.1	50	0.005	1.502E+09
250	0.1	100	0.005	1.050E+09

b	ε	k	DR	IL
250	1.0	3	0.006	3.305E+07
250	1.0	5	0.005	2.492E+07
250	1.0	10	0.005	2.043E+07
250	1.0	15	0.005	1.896E+07
250	1.0	20	0.005	1.890E+07
250	1.0	25	0.005	1.892E+07
250	1.0	50	0.005	1.513E+07
250	1.0	100	0.005	1.062E+07

b	ε	k	DR	IL
250	10	3	0.032	3.985E+05
250	10	5	0.019	3.345E+05
250	10	10	0.012	3.021E+05
250	10	15	0.011	2.917E+05
250	10	20	0.010	2.934E+05
250	10	25	0.009	2.950E+05
250	10	50	0.006	2.600E+05
250	10	100	0.005	2.164E+05

b	ε	k	DR	IL
250	100	3	0.155	7.152E+04
250	100	5	0.079	8.824E+04
250	100	10	0.041	1.004E+05
250	100	15	0.027	1.047E+05
250	100	20	0.022	1.069E+05
250	100	25	0.019	1.083E+05
250	100	50	0.010	1.110E+05
250	100	100	0.006	1.122E+05

TABLE 7.9: Differential privacy DR & IL estimations for increasing ε and k values and fixed buffer size ($b = 250$). Results from the execution with the `RandomRBFGenerator`.

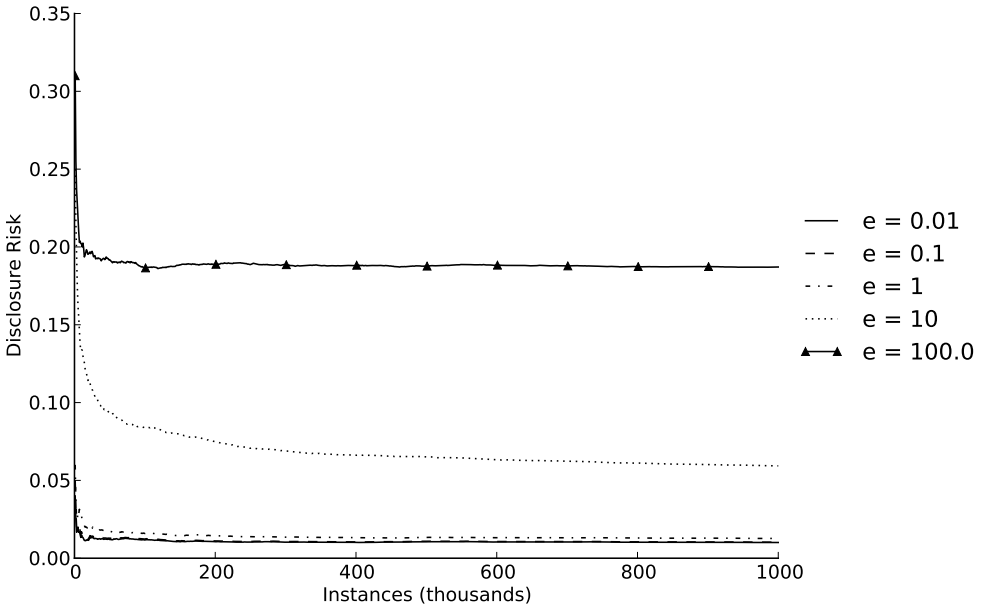

FIGURE 7.8: `DifferentialPrivacyFilter` DR evaluation using the `RandomRBFGenerator` with fixed buffer size $b = 100$ and fixed cluster size $k = 3$, for increasing ε differential privacy scale factor.

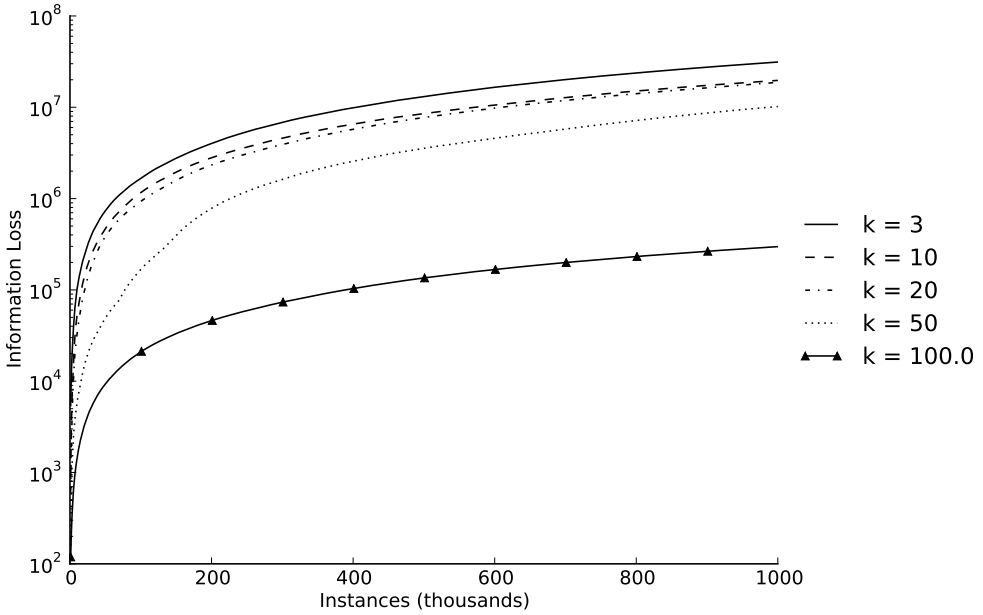

FIGURE 7.9: `DifferentialPrivacyFilter` IL evaluation using the `RandomRBFGenerator` with fixed buffer size $b = 100$ and fixed differential privacy scale parameter $\varepsilon = 1$, for increasing k, on a logarithmic scale.

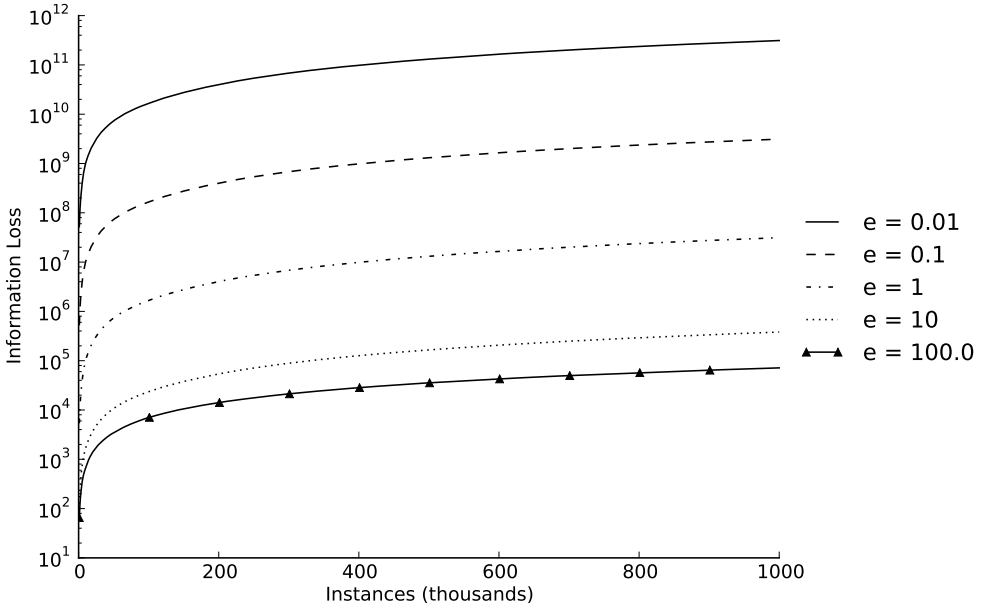

FIGURE 7.10: `DifferentialPrivacyFilter` IL evaluation using the `RandomRBFGenerator` with fixed buffer size $b = 100$ and fixed cluster size $k = 3$, for increasing ε differential privacy scale factor, on a logarithmic scale.

Chapter 8

Conclusions

8.1 Achieved goals

The present report assesses the inception, development and benchmarking of the implementation of several Statistical Disclosure Control (SDC) methods, adapted to the MOA data stream mining framework. Concerning the actual goals of the project, we can say that it has ultimately been successful: the main requirements have been fulfilled and the resulting work has proven its performance and utility for the data science community.

First, a thorough theoretical background analysis had to be done in order to select the approaches best suited to the implementation of the filters on the MOA environment. Moreover, the state of the art of existing solutions was reviewed and some technology alternatives were considered before the actual development of the algorithms.

Four MOA privacy preserving filters have been developed, implementing the following SDC methods: *noise addition*, *microaggregation*, *data rank swapping* and a microaggregation based *differential privacy* mechanism. Each of the algorithms has been adapted from well-known solutions, already in use in non-streaming data analysis settings, in order to enable their utilization in stream processing tasks. Special emphasis has been put in easing the filters customization, either by setting the appropriate parameterizations or by actually modifying parts of their behaviour exploiting the extensibility that the MOA framework offers us. Finally, all four filters have been benchmarked to assess their quality in terms of two important SDC measurments: *disclosure risk* and *information loss*. These quality parameters are evaluated using estimators, which implementation has also been adapted to the streams processing setting from existing alternatives. While some of the methods perform better than others, they all offer results that conform to their theoretical limits.

Even though this work can be improved and extended by implementing some more SDC mechanisms, it is most useful to protect people's privacy, which was the main goal of this project. It is yet another contribution aimed to enhance the preservation of the universal right to privacy, which definitely deserves more attention from the IT community than it actually gets.

On the personal side, the development of this project has brought me a closer understanding of SDC, a field that was almost unknown to me, and has reassured the interest I have for two of the main concerns of the project: data science and the relation between technology and society, since privacy preservation is, indeed, the project's most important outcome. Beyond this, the possibility of working in an open source project and being able to contribute to its extension has proved to be really engaging.

Finally, I am certainly happy of having found a project in which all the skills and knowledge acquired throughout these years as an undergraduate student could be put in practice. Not only algorithmical theory and statistics-related concepts have been used, but also analytical rigour and good practices in terms of software architechture and development were necessary for the project's success.

8.2 Future work

Different aspects of this project deserve to be considered for future enhancements, ranging from algorithmic details to benchmarking methods.

First and foremost, there is a particular non-functional requirement that could be emphasized in the future: documentation. The confection of a user manual for the MOA extension that the privacy filters represent would be a great complement to the toolsuite. In addition, the developed package should be released, as a binary distribution, to a central repository, in order to ease access not only to the source code, that is already public, but to ready for use bundles.

Concerning the algorithmic facet of the project, there is room for performance improvement, in terms of computational complexity and execution time. Advanced and more customized data structures could be used, as well as some other design approaches. Moreover, a complete code base refactoring should be carried out to adapt the SDC methods to the changes introduced by the latest MOA release, that was published while developing the filters for this project.

Finally, a more rigorous benchmark process could also be set up, running multiple executions to assess statistically valid results and applying the filters to real world dataset, for example, all of which could not be done due to the lack of time available.

Appendix A

Results tables

The following are the complete tables of the benchmark process performed using each of the privacy filters, for all the parameterizations defined in the experiments design.

`NoiseAdditionFilter`

a	c	DR	IL
0.1	0.0	0.956	1123.97
0.25	0.0	0.889	7024.83
0.5	0.0	0.774	28099.33
0.75	0.0	0.596	63223.50
1.0	0.0	0.430	112397.34

TABLE A.1: Noise addition IL & DR estimations for all considered parameterizations with the *RandomRBF* generator.

a	c	DR	IL
0.1	0.0	1.000	50741.41
0.25	0.0	1.000	317133.83
0.5	0.0	0.996	1268535.34
0.75	0.0	0.919	2854204.53
1.0	0.0	0.742	5074141.39

TABLE A.2: Noise addition IL & DR estimations for all considered parameterizations with the *Waveform* generator.

`MicroAggregationFilter`

b	k	DR	IL	b	k	DR	IL
100	3	0.232	74917.30	250	3	0.153	74917.30
100	5	0.144	89953.68	250	5	0.089	89953.68
100	10	0.089	101193.72	250	10	0.051	101193.72
100	15	0.069	104952.79	250	15	0.039	104952.79
100	20	0.061	106800.51	250	20	0.034	106800.51
100	25	0.055	107937.42	250	25	0.031	107937.42
100	50	0.041	110179.72	250	50	0.025	110179.72
100	100	0.030	111313.78	250	100	0.018	111313.78

b	k	DR	IL	b	k	DR	IL
500	3	0.109	74917.30	1000	3	0.078	74917.30
500	5	0.060	89953.68	1000	5	0.040	89953.68
500	10	0.033	101193.72	1000	10	0.021	101193.72
500	15	0.025	104952.79	1000	15	0.015	104952.79
500	20	0.021	106800.51	1000	20	0.013	106800.51
500	25	0.019	107937.42	1000	25	0.011	107937.42
500	50	0.015	110179.72	1000	50	0.009	110179.72
500	100	0.012	111313.78	1000	100	0.008	111313.78

TABLE A.3: Microaggregation IL & DR estimations for all considered parameterizations with the *RandomRBF* generator.

b	k	DR	IL	b	k	DR	IL
100	3	0.241	3375656.78	250	3	0.186	3375656.78
100	5	0.135	4057649.13	250	5	0.092	4057649.13
100	10	0.076	4566550.54	250	10	0.045	4566550.54
100	15	0.061	4739204.47	250	15	0.035	4739204.47
100	20	0.054	4822716.29	250	20	0.030	4822716.29
100	25	0.049	4872587.52	250	25	0.027	4872587.52
100	50	0.038	4976718.32	250	50	0.022	4976718.32
100	100	0.029	5027574.64	250	100	0.017	5027574.64

b	k	DR	IL	b	k	DR	IL
500	3	0.152	3375656.78	1000	3	0.125	3375656.78
500	5	0.068	4057649.13	1000	5	0.050	4057649.13
500	10	0.030	4566550.54	1000	10	0.020	4566550.54
500	15	0.022	4739204.47	1000	15	0.014	4739204.47
500	20	0.019	4822716.29	1000	20	0.011	4822716.29
500	25	0.017	4872587.52	1000	25	0.010	4872587.52
500	50	0.013	4976718.32	1000	50	0.007	4976718.32
500	100	0.012	5027574.64	1000	100	0.007	5027574.64

TABLE A.4: Microaggregation IL & DR estimations for all considered parameterizations with the *Waveform* generator.

`RankSwappingFilter`

b	p	DR	IL
100	10	0.838	19136.62
100	25	0.469	57753.05
100	50	0.070	136791.17
100	75	0.036	178200.11
100	80	0.037	178685.81

b	p	DR	IL
250	10	0.910	6875.94
250	25	0.749	17803.36
250	50	0.473	41889.27
250	75	0.211	70737.65
250	80	0.176	76772.86

b	p	DR	IL
500	10	0.953	3402.06
500	25	0.844	8210.11
500	50	0.679	17322.47
500	75	0.523	28339.42
500	80	0.493	30792.56

b	p	DR	IL
1000	10	0.980	1670.47
1000	25	0.916	4048.45
1000	50	0.801	7993.85
1000	75	0.699	12328.09
1000	80	0.681	13204.72

Table A.5: Rank swapping IL & DR estimations for all considered parameterizations with the *RandomRBF* generator.

b	p	DR	IL
100	10	0.997	705196.07
100	25	0.776	2464673.56
100	50	0.112	6195304.18
100	75	0.046	8088473.27
100	80	0.045	8097619.70

b	p	DR	IL
250	10	1.000	185228.04
250	25	0.996	643029.21
250	50	0.872	1729458.77
250	75	0.491	3077876.57
250	80	0.415	3362868.67

b	p	DR	IL
500	10	1.000	69572.08
500	25	1.000	233509.06
500	50	0.995	621635.78
500	75	0.955	1118353.33
500	80	0.937	1229266.47

b	p	DR	IL
1000	10	1.000	27149.30
1000	25	1.000	86497.92
1000	50	1.000	225324.49
1000	75	0.999	401418.85
1000	80	0.998	440628.83

Table A.6: Rank swapping IL & DR estimations for all considered parameterizations with the *Waveform* generator.

`DifferentialPrivacyFilter`

b	ε	k	DR	IL	b	ε	k	DR	IL
100	0.01	3	0.010	3.122E+11	250	0.01	3	0.004	3.298E+11
100	0.01	5	0.010	2.411E+11	250	0.01	5	0.004	2.483E+11
100	0.01	10	0.010	1.959E+11	250	0.01	10	0.004	2.033E+11
100	0.01	15	0.010	1.867E+11	250	0.01	15	0.005	1.885E+11
100	0.01	20	0.010	1.861E+11	250	0.01	20	0.004	1.880E+11
100	0.01	25	0.010	1.771E+11	250	0.01	25	0.004	1.881E+11
100	0.01	50	0.010	1.007E+11	250	0.01	50	0.005	1.502E+11
100	0.01	100	0.010	1.860E+09	250	0.01	100	0.005	1.050E+11

b	ε	k	DR	IL	b	ε	k	DR	IL
500	0.01	3	0.002	3.222E+11	1000	0.01	3	0.001	3.332E+11
500	0.01	5	0.002	2.557E+11	1000	0.01	5	0.001	2.618E+11
500	0.01	10	0.002	2.011E+11	1000	0.01	10	0.001	2.000E+11
500	0.01	15	0.002	1.876E+11	1000	0.01	15	0.001	1.724E+11
500	0.01	20	0.002	1.949E+11	1000	0.01	20	0.001	1.922E+11
500	0.01	25	0.002	1.917E+11	1000	0.01	25	0.001	1.995E+11
500	0.01	50	0.002	1.693E+11	1000	0.01	50	0.001	1.744E+11
500	0.01	100	0.002	1.454E+11	1000	0.01	100	0.001	1.616E+11

TABLE A.7: Differential privacy filter IL & DR estimations for $\varepsilon = 0.01$ with the *RandomRBF* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	0.1	3	0.010	3.122E+09	250	0.1	3	0.005	3.298E+09
100	0.1	5	0.010	2.411E+09	250	0.1	5	0.005	2.483E+09
100	0.1	10	0.010	1.960E+09	250	0.1	10	0.004	2.033E+09
100	0.1	15	0.010	1.867E+09	250	0.1	15	0.005	1.885E+09
100	0.1	20	0.010	1.861E+09	250	0.1	20	0.005	1.880E+09
100	0.1	25	0.010	1.771E+09	250	0.1	25	0.004	1.881E+09
100	0.1	50	0.010	1.007E+09	250	0.1	50	0.005	1.502E+09
100	0.1	100	0.010	1.871E+07	250	0.1	100	0.005	1.050E+09

b	ε	k	DR	IL	b	ε	k	DR	IL
500	0.1	3	0.002	3.222E+09	1000	0.1	3	0.001	3.332E+09
500	0.1	5	0.002	2.558E+09	1000	0.1	5	0.001	2.618E+09
500	0.1	10	0.002	2.011E+09	1000	0.1	10	0.001	2.000E+09
500	0.1	15	0.002	1.877E+09	1000	0.1	15	0.001	1.724E+09
500	0.1	20	0.002	1.949E+09	1000	0.1	20	0.001	1.923E+09
500	0.1	25	0.002	1.917E+09	1000	0.1	25	0.001	1.995E+09
500	0.1	50	0.002	1.693E+09	1000	0.1	50	0.001	1.744E+09
500	0.1	100	0.002	1.454E+09	1000	0.1	100	0.001	1.616E+09

TABLE A.8: Differential privacy filter IL & DR estimations for $\varepsilon = 0.1$ with the *RandomRBF* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	1.0	3	0.013	3.129E+07	250	1.0	3	0.006	3.305E+07
100	1.0	5	0.012	2.420E+07	250	1.0	5	0.005	2.492E+07
100	1.0	10	0.011	1.970E+07	250	1.0	10	0.005	2.043E+07
100	1.0	15	0.011	1.878E+07	250	1.0	15	0.005	1.896E+07
100	1.0	20	0.011	1.872E+07	250	1.0	20	0.005	1.890E+07
100	1.0	25	0.011	1.782E+07	250	1.0	25	0.005	1.892E+07
100	1.0	50	0.010	1.019E+07	250	1.0	50	0.005	1.513E+07
100	1.0	100	0.010	2.977E+05	250	1.0	100	0.005	1.062E+07

b	ε	k	DR	IL	b	ε	k	DR	IL
500	1.0	3	0.003	3.229E+07	1000	1.0	3	0.002	3.339E+07
500	1.0	5	0.003	2.567E+07	1000	1.0	5	0.002	2.627E+07
500	1.0	10	0.003	2.022E+07	1000	1.0	10	0.001	2.010E+07
500	1.0	15	0.002	1.887E+07	1000	1.0	15	0.001	1.735E+07
500	1.0	20	0.002	1.960E+07	1000	1.0	20	0.001	1.933E+07
500	1.0	25	0.002	1.928E+07	1000	1.0	25	0.001	2.006E+07
500	1.0	50	0.003	1.704E+07	1000	1.0	50	0.001	1.755E+07
500	1.0	100	0.002	1.466E+07	1000	1.0	100	0.001	1.627E+07

TABLE A.9: Differential privacy filter IL & DR estimations for $\varepsilon = 1.0$ with the *RandomRBF* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	10	3	0.059	3.809E+05	250	10	3	0.032	3.985E+05
100	10	5	0.038	3.271E+05	250	10	5	0.019	3.345E+05
100	10	10	0.025	2.948E+05	250	10	10	0.012	3.021E+05
100	10	15	0.020	2.900E+05	250	10	15	0.011	2.917E+05
100	10	20	0.018	2.919E+05	250	10	20	0.010	2.934E+05
100	10	25	0.016	2.843E+05	250	10	25	0.009	2.950E+05
100	10	50	0.011	2.111E+05	250	10	50	0.006	2.600E+05
100	10	100	0.011	1.132E+05	250	10	100	0.005	2.164E+05

b	ε	k	DR	IL	b	ε	k	DR	IL
500	10	3	0.022	3.907E+05	1000	10	3	0.015	4.019E+05
500	10	5	0.012	3.421E+05	1000	10	5	0.008	3.480E+05
500	10	10	0.008	2.997E+05	1000	10	10	0.005	2.986E+05
500	10	15	0.006	2.907E+05	1000	10	15	0.004	2.755E+05
500	10	20	0.006	3.004E+05	1000	10	20	0.004	2.976E+05
500	10	25	0.006	2.985E+05	1000	10	25	0.004	3.062E+05
500	10	50	0.004	2.789E+05	1000	10	50	0.003	2.841E+05
500	10	100	0.003	2.567E+05	1000	10	100	0.002	2.729E+05

TABLE A.10: Differential privacy filter IL & DR estimations for $\varepsilon = 10$ with the *RandomRBF* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	100	3	0.187	7.148E+04	250	100	3	0.155	7.152E+04
100	100	5	0.103	8.819E+04	250	100	5	0.079	8.824E+04
100	100	10	0.053	1.004E+05	250	100	10	0.041	1.004E+05
100	100	15	0.035	1.049E+05	250	100	15	0.027	1.047E+05
100	100	20	0.027	1.072E+05	250	100	20	0.022	1.069E+05
100	100	25	0.023	1.086E+05	250	100	25	0.019	1.083E+05
100	100	50	0.012	1.111E+05	250	100	50	0.010	1.110E+05
100	100	100	0.011	1.113E+05	250	100	100	0.006	1.122E+05

b	ε	k	DR	IL	b	ε	k	DR	IL
500	100	3	0.139	7.136E+04	1000	100	3	0.125	7.149E+04
500	100	5	0.067	8.840E+04	1000	100	5	0.058	8.848E+04
500	100	10	0.037	1.003E+05	1000	100	10	0.034	1.003E+05
500	100	15	0.023	1.047E+05	1000	100	15	0.020	1.045E+05
500	100	20	0.019	1.070E+05	1000	100	20	0.017	1.068E+05
500	100	25	0.016	1.083E+05	1000	100	25	0.015	1.082E+05
500	100	50	0.009	1.110E+05	1000	100	50	0.008	1.110E+05
500	100	100	0.005	1.124E+05	1000	100	100	0.004	1.125E+05

TABLE A.11: Differential privacy filter IL & DR estimations for $\varepsilon = 100$ with the *RandomRBF* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	0.01	3	0.010	8.511E+12	250	0.01	3	0.004	8.293E+12
100	0.01	5	0.010	7.466E+12	250	0.01	5	0.004	7.621E+12
100	0.01	10	0.010	6.583E+12	250	0.01	10	0.004	6.561E+12
100	0.01	15	0.010	5.583E+12	250	0.01	15	0.004	5.672E+12
100	0.01	20	0.010	5.270E+12	250	0.01	20	0.004	5.226E+12
100	0.01	25	0.010	5.202E+12	250	0.01	25	0.004	5.289E+12
100	0.01	50	0.010	3.307E+12	250	0.01	50	0.004	4.564E+12
100	0.01	100	0.010	8.695E+10	250	0.01	100	0.004	2.640E+12

b	ε	k	DR	IL	b	ε	k	DR	IL
500	0.01	3	0.002	8.370E+12	1000	0.01	3	0.001	8.331E+12
500	0.01	5	0.002	7.711E+12	1000	0.01	5	0.001	7.569E+12
500	0.01	10	0.002	6.765E+12	1000	0.01	10	0.001	6.751E+12
500	0.01	15	0.002	5.900E+12	1000	0.01	15	0.001	6.017E+12
500	0.01	20	0.002	5.471E+12	1000	0.01	20	0.001	5.606E+12
500	0.01	25	0.002	5.420E+12	1000	0.01	25	0.001	5.296E+12
500	0.01	50	0.002	4.578E+12	1000	0.01	50	0.001	4.511E+12
500	0.01	100	0.002	3.559E+12	1000	0.01	100	0.001	3.703E+12

TABLE A.12: Differential privacy filter IL & DR estimations for $\varepsilon = 0.01$ with the *Waveform* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	0.1	3	0.010	8.512E+10	250	0.1	3	0.004	8.293E+10
100	0.1	5	0.010	7.467E+10	250	0.1	5	0.004	7.621E+10
100	0.1	10	0.010	6.584E+10	250	0.1	10	0.004	6.562E+10
100	0.1	15	0.010	5.584E+10	250	0.1	15	0.004	5.673E+10
100	0.1	20	0.010	5.270E+10	250	0.1	20	0.004	5.227E+10
100	0.1	25	0.010	5.203E+10	250	0.1	25	0.004	5.289E+10
100	0.1	50	0.010	3.307E+10	250	0.1	50	0.004	4.564E+10
100	0.1	100	0.010	8.746E+08	250	0.1	100	0.004	2.640E+10

b	ε	k	DR	IL	b	ε	k	DR	IL
500	0.1	3	0.002	8.370E+10	1000	0.1	3	0.001	8.331E+10
500	0.1	5	0.002	7.712E+10	1000	0.1	5	0.001	7.569E+10
500	0.1	10	0.002	6.765E+10	1000	0.1	10	0.001	6.751E+10
500	0.1	15	0.002	5.901E+10	1000	0.1	15	0.001	6.017E+10
500	0.1	20	0.002	5.472E+10	1000	0.1	20	0.001	5.607E+10
500	0.1	25	0.002	5.420E+10	1000	0.1	25	0.001	5.296E+10
500	0.1	50	0.002	4.579E+10	1000	0.1	50	0.001	4.511E+10
500	0.1	100	0.002	3.560E+10	1000	0.1	100	0.001	3.703E+10

TABLE A.13: Differential privacy filter IL & DR estimations for $\varepsilon = 0.1$ with the *Waveform* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	1.0	3	0.014	8.542E+08	250	1.0	3	0.006	8.323E+08
100	1.0	5	0.012	7.505E+08	250	1.0	5	0.005	7.660E+08
100	1.0	10	0.011	6.629E+08	250	1.0	10	0.005	6.607E+08
100	1.0	15	0.010	5.631E+08	250	1.0	15	0.004	5.719E+08
100	1.0	20	0.010	5.318E+08	250	1.0	20	0.004	5.275E+08
100	1.0	25	0.010	5.251E+08	250	1.0	25	0.004	5.338E+08
100	1.0	50	0.010	3.357E+08	250	1.0	50	0.004	4.614E+08
100	1.0	100	0.010	1.373E+07	250	1.0	100	0.004	2.690E+08

b	ε	k	DR	IL	b	ε	k	DR	IL
500	1.0	3	0.003	8.400E+08	1000	1.0	3	0.002	8.361E+08
500	1.0	5	0.003	7.750E+08	1000	1.0	5	0.001	7.607E+08
500	1.0	10	0.002	6.810E+08	1000	1.0	10	0.001	6.796E+08
500	1.0	15	0.002	5.948E+08	1000	1.0	15	0.001	6.064E+08
500	1.0	20	0.002	5.520E+08	1000	1.0	20	0.001	5.655E+08
500	1.0	25	0.002	5.469E+08	1000	1.0	25	0.001	5.345E+08
500	1.0	50	0.002	4.629E+08	1000	1.0	50	0.001	4.561E+08
500	1.0	100	0.002	3.610E+08	1000	1.0	100	0.001	3.754E+08

TABLE A.14: Differential privacy filter IL & DR estimations for $\varepsilon = 1.0$ with the *Waveform* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	10	3	0.123	1.148E+07	250	10	3	0.085	1.127E+07
100	10	5	0.075	1.126E+07	250	10	5	0.050	1.142E+07
100	10	10	0.040	1.103E+07	250	10	10	0.027	1.100E+07
100	10	15	0.030	1.024E+07	250	10	15	0.021	1.033E+07
100	10	20	0.023	1.004E+07	250	10	20	0.016	9.988E+06
100	10	25	0.020	1.005E+07	250	10	25	0.013	1.012E+07
100	10	50	0.012	8.284E+06	250	10	50	0.008	9.526E+06
100	10	100	0.011	5.116E+06	250	10	100	0.005	7.668E+06

b	ε	k	DR	IL	b	ε	k	DR	IL
500	10	3	0.062	1.134E+07	1000	10	3	0.047	1.130E+07
500	10	5	0.037	1.151E+07	1000	10	5	0.028	1.136E+07
500	10	10	0.021	1.120E+07	1000	10	10	0.016	1.119E+07
500	10	15	0.016	1.055E+07	1000	10	15	0.012	1.067E+07
500	10	20	0.013	1.023E+07	1000	10	20	0.010	1.036E+07
500	10	25	0.011	1.025E+07	1000	10	25	0.008	1.012E+07
500	10	50	0.007	9.540E+06	1000	10	50	0.005	9.472E+06
500	10	100	0.004	8.585E+06	1000	10	100	0.003	8.725E+06

TABLE A.15: Differential privacy filter IL & DR estimations for $\varepsilon = 10$ with the *Waveform* generator.

b	ε	k	DR	IL	b	ε	k	DR	IL
100	100	3	0.262	3.053E+06	250	100	3	0.221	3.047E+06
100	100	5	0.131	3.867E+06	250	100	5	0.108	3.867E+06
100	100	10	0.059	4.500E+06	250	100	10	0.049	4.494E+06
100	100	15	0.039	4.709E+06	250	100	15	0.032	4.703E+06
100	100	20	0.030	4.821E+06	250	100	20	0.024	4.807E+06
100	100	25	0.024	4.890E+06	250	100	25	0.020	4.874E+06
100	100	50	0.013	5.003E+06	250	100	50	0.011	5.000E+06
100	100	100	0.010	5.029E+06	250	100	100	0.006	5.049E+06

b	ε	k	DR	IL	b	ε	k	DR	IL
500	100	3	0.194	3.052E+06	1000	100	3	0.178	3.043E+06
500	100	5	0.096	3.868E+06	1000	100	5	0.090	3.861E+06
500	100	10	0.044	4.498E+06	1000	100	10	0.042	4.496E+06
500	100	15	0.029	4.702E+06	1000	100	15	0.028	4.702E+06
500	100	20	0.022	4.806E+06	1000	100	20	0.021	4.806E+06
500	100	25	0.018	4.873E+06	1000	100	25	0.017	4.874E+06
500	100	50	0.009	4.997E+06	1000	100	50	0.008	4.997E+06
500	100	100	0.005	5.054E+06	1000	100	100	0.005	5.051E+06

TABLE A.16: Differential privacy filter IL & DR estimations for $\varepsilon = 100$ with the *Waveform* generator.

Bibliography

[1] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Advances in knowledge discovery and data mining. chapter From Data Mining to Knowledge Discovery: An Overview, pages 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996. ISBN 0-262-56097-6. URL `http://dl.acm.org/citation.cfm?id=257938.257942`.

[2] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011. ISBN 1107015359, 9781107015357.

[3] Merriam Webster Inc. Seclusion, October 2014. URL `http://www.merriam-webster.com/dictionary/seclusion`.

[4] UN General Assembly. Universal declaration of human rights, December 1948.

[5] New Zealand University of Waikato. Moa - overview, October 2014. URL `http://moa.cms.waikato.ac.nz/overview`.

[6] New Zealand University of Waikato. Weka 3 - data mining with open source machine learning software in java, October 2014. URL `http://www.cs.waikato.ac.nz/ml/weka`.

[7] Mohamed Gaber. Mining data streams: a review. *ACM SIGMOD Record*, 34, June 2005.

[8] Anco Hundepool et. al. *Statistical Disclosure Control*. John Wiley & Sons, Ltd.

[9] Matthias Templ, Bernhard Meindl, and Alexander Kowarik. Introduction to statistical disclosure control (sdc). URL `http://cran.r-project.org/web/packages/sdcMicro/vignettes/sdc_guidelines.pdf`.

[10] Josep Domingo-Ferrer and Vicenç Torra. Disclosure risk assessment in statistical data protection. doi: 10.1016/S0377-0427(03)00643-5.

[11] Josep Domingo-Ferrer and Vicenç Torra. Disclosure control methods and information loss for microdata. In *Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies*, pages 91–110. Elsevier Science, 2001.

[12] JosepM. Mateo-Sanz, Josep Domingo-Ferrer, and Francesc Sebé. Probabilistic information loss measures in confidentiality protection of continuous microdata. *Data Mining and Knowledge Discovery*, 11(2):181–193, 2005. ISSN 1384-5810. doi: 10.1007/s10618-005-0011-9. URL `http://dx.doi.org/10.1007/s10618-005-0011-9`.

[13] Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), October 2002. doi: 10.1142/S0218488502001648. URL `http://dx.doi.org/10.1142/S0218488502001648`.

[14] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. Enhancing data utility in differential privacy via microaggregation-based k-anonymity. *The VLDB Journal*, 23(5):771–794, 2014. ISSN 1066-8888. doi: 10.1007/s00778-014-0351-4. URL `http://dx.doi.org/10.1007/s00778-014-0351-4`.

[15] Ashwin Machanavajjhala. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 2007.

[16] Li Ninghui. t-closeness: Privacy beyond k-anonymity and l-diversity. *Proc. of IEEE 23rd Int'l Conf. on Data Engineering*, 2007.

[17] Cynthia Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[18] J. Domingo-Ferrer and J.M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *Knowledge and Data Engineering, IEEE Transactions on*, 14(1), Jan 2002. doi: 10.1109/69.979982.

[19] Josep Domingo-Ferrer and Vicenç Torra. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11 (2):195–212, 2005. ISSN 1384-5810. doi: 10.1007/s10618-005-0007-5. URL `http://dx.doi.org/10.1007/s10618-005-0007-5`.

[20] Josep Domingo-Ferrer, Francesc Sebé, and Agusti Solanas. A polynomial-time approximation to optimal multivariate microaggregation. *Computers & Mathematics with Applications*, 55(4):714 – 732, 2008. doi: http://dx.doi.org/10.1016/j.camwa.2007.04.034. URL `http://www.sciencedirect.com/science/article/pii/S0898122107005044`.

[21] David Leoni. Non-interactive differential privacy: A survey. In *Proceedings of the First International Workshop on Open Data*, WOD '12, pages 40–52, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1404-6. doi: 10.1145/2422604.2422611. URL http://doi.acm.org/10.1145/2422604.2422611.

[22] IkamusumeFan. Laplace distribution pdf, 2005. URL http://commons.wikimedia.org/wiki/File:Laplace_distribution_pdf.png.

[23] Sasha Romanosky. Do data breach disclosure laws reduce identity theft? In *Workshop on the Economics of Information Security*, 2008.

[24] BOE. Ley orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal, 1999. URL https://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf.

[25] Internet & Jurisdiction Project. Internet & jurisdiction project - a global multi-stakeholder dialogue process, 2015. URL http://www.internetjurisdiction.net.

[26] Martin Fowler et. al. Agile manifesto for software development, October 2014. URL http://agilemanifesto.org.

[27] Shane Hastie. Agile australia - opening keynotes, October 2010. URL http://www.infoq.com/news/2010/10/agile-australia-keynotes.

[28] Wikipedia. Scrum (software development), 2015. URL http://en.wikipedia.org/wiki/Scrum_%28software_development%29.

[29] Lakeworks. Scrum process, 2009. URL http://commons.wikimedia.org/wiki/File:Scrum_process.svg.

[30] Glassdoor Inc. Company salaries - glassdoor, October 2014. URL http://www.glassdoor.com/Salaries/index.htm.

[31] Wikipedia. Openmp, 2015. URL http://en.wikipedia.org/wiki/OpenMP.

[32] Oracle Corporation. Java se 7 java native interface-related apis and developer guides, 2014. URL http://docs.oracle.com/javase/7/docs/technotes/guides/jni/.

[33] Simon Urbanek. rjava - low-level r to java interface, 2015. URL https://www.rforge.net/rJava.

[34] Simon Urbanek. Jri - java/r interface, 2015. URL https://www.rforge.net/JRI.

[35] Simon Urbanek. Rserve - binary r server, 2015. URL `https://rforge.net/Rserve/index.html`.

[36] BeDataDriven Inc. Renjin, 2015. URL `http://www.renjin.org`.

[37] Robert C. Martin. The principles of ood, 2015. URL `http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod`.

[38] Ruth Brand. Microdata protection through noise addition. In Josep Domingo-Ferrer, editor, *Inference Control in Statistical Databases*, volume 2316 of *Lecture Notes in Computer Science*, pages 97–116. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-43614-0. doi: 10.1007/3-540-47804-3_8. URL `http://dx.doi.org/10.1007/3-540-47804-3_8`.

[39] Wikipedia. Inversion of control, 2014. URL `http://en.wikipedia.org/wiki/Inversion_of_control`.

[40] Office for National Statistics. Statistical disclosure control, 2014. URL `http://www.ons.gov.uk/ons/guide-method/method-quality/general-methodology/statistical-disclosure-control/index.html`.

[41] H. Kargupta. Minefleet®: The vehicle data stream mining system for ubiquitous environments. *Ubiquitous Knowledge Discovery*, 2010.

[42] Matthias Templ. *sdcMicro: Statistical Disclosure Control methods for anonymization of microdata and risk estimation.* CRAN R package repository. URL `http://cran.r-project.org/web/packages/sdcMicro/index.html`.

[43] Shane Richmond. Millions of internet users hit by massive sony playstation data theft, April 2011. URL `http://www.telegraph.co.uk/technology/news/8475728/Millions-of-internet-users-hit-by-massive-Sony-PlayStation-data-theft.html`.

[44] Arthur Charles. Naked celebrity hack: security experts focus on icloud backup theory, September 2014. URL `http://www.theguardian.com/technology/2014/sep/01/naked-celebrity-hack-icloud-backup-jennifer-lawrence`.

[45] Yahoo. Samoa by yahoo, 2014. URL `http://samoa-project.net`.

[46] The Apache Software Foundation. S4: Distributed stream computing platform, 2014. URL `http://incubator.apache.org/s4`.

[47] The Apache Software Foundation. Storm, distributed and fault-tolerant realtime computation, 2014. URL `https://storm.incubator.apache.org`.

[48] Cullen (Director) Hoback. Terms and conditions may apply. Documentary, 2013. URL `http://tacma.net`.

[49] Jordi Nin, Javier Herranz, and Vicenç Torra. Rethinking rank swapping to decrease disclosure risk. doi: 10.1016/j.datak.2007.07.006. URL `http://www.sciencedirect.com/science/article/pii/S0169023X07001498`.

[50] Vicenç Torra and Josep Domingo-Ferrer. Record linkage methods for multidatabase data mining. In Vicenç Torra, editor, *Information Fusion in Data Mining*, volume 123 of *Studies in Fuzziness and Soft Computing*, pages 101–132. Springer Berlin Heidelberg, 2003. ISBN 978-3-642-05628-4. doi: 10.1007/978-3-540-36519-8_7. URL `http://dx.doi.org/10.1007/978-3-540-36519-8_7`.