Aggregation and Privacy in Multi-relational Databases

By

Yasser Jafer, BEng.

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MASc degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Most existing data mining approaches perform data mining tasks on a single data table. However, increasingly, data repositories such as financial data and medical records, amongst others, are stored in relational databases. The inability of applying traditional data mining techniques directly on such relational database thus poses a serious challenge. To address this issue, a number of researchers convert a relational database into one or more flat files and then apply traditional data mining algorithms. The above-mentioned process of transforming a relational database into one or more flat files usually involves aggregation. Aggregation functions such as *maximum, minimum, average, standard deviation, count* and *sum* are commonly used in such a flattening process.

Our research aims to address the following question: Is there a link between aggregation and possible privacy violations during relational database mining? In this research we investigate how, and if, applying aggregation functions will affect the privacy of a relational database, during supervised learning, or classification, where the target concept is known. To this end, we introduce the PBIRD (Privacy Breach Investigation in Relational Databases) methodology. The PBIRD methodology combines multi-view learning with feature selection, to discover the potentially *dangerous* sets of features as hidden within a database. Our approach creates a number of views, which consist of subsets of the data, with and without aggregation. Then, by identifying and investigating the set of selected features in each view, potential privacy breaches are detected. In this way, our PBIRD algorithm is able to discover those features that are correlated with the classification target that may also lead to revealing of sensitive information in the database.

Our experimental results show that aggregation functions do, indeed, change the correlation between attributes and the classification target. We show that with aggregation, we obtain a set of features which can be accurately linked to the classification target and used to predict (with high accuracy) the confidential information. On the other hand, the results show that, without aggregation we obtain another different set of potentially harmful features. By identifying the complete set of potentially dangerous attributes, the PBIRD

methodology provides a solution where the database designers/owners can be warned, to subsequently perform necessary adjustments to protect the privacy of the relational database.

In our research, we also perform a comparative study to investigate the impact of aggregation on the classification accuracy and on the time required to build the models. Our results suggest that in the case where a database consists only of categorical data, aggregation should especially be used with caution. This is due to the fact that aggregation causes a decrease in overall accuracies of the resulting models. When the database contains mixed attributes, the results show that the accuracies without aggregation and with aggregation are comparable. However, even in such scenarios, schemas without aggregation tend to slightly outperform. With regard to the impact of aggregation on the model building time, the results show that, in general, the models constructed with aggregation require shorter building time. However, when the database is small and consists of nominal attributes with high cardinality, aggregation causes a slower model building time.

Acknowledgements

I would like to express my sincere gratefulness to my supervisor Dr. Herna L. Viktor for guiding me through the process of completion of this thesis. Her insightful remarks, valuable advices, and continuous support are deeply appreciated. Also, I would like to sincerely thank my co-supervisor Dr. Eric Paquet for his insights and comments that added much value to my work.

My deepest gratitude goes to my parents for their continued love, support, and encouragement in pursuing my graduate studies. Endless thanks to my wife for her support and encouragement. And finally, special thanks to my sister Shafagh for her comments and recommendations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There is an exponential increase in the number of relational databases that store different types of data ranging from financial information, to medical records, manufacturing, and personal information. This fact, combined with an unprecedented increase in the amount of information stored in these databases, introduces new challenges to the data mining community.

One such challenge is related to mining relational databases. Most of the existing data mining approaches are propositional. These approaches tend to look for patterns in a single table and hence are applied on a single table. Mining multi-relational databases and multi-relational classification on the other hand, aims to discover patterns among multiple tables (relations). To shift from a propositional to a multi-relational paradigm, two strategies have been identified in the past few years [33]. The first strategy requires upgrading existing data mining algorithms in order to deal with the relational databases directly. The second strategy follows a two-step solution. Firstly, the relational database is converted into a single universal flat file (or multiple smaller flat files, in the case of multi-view learning). Secondly, conventional data mining algorithms are applied on the resulting flat file(s). As such, no upgrading or modification of the existing data mining algorithms is necessary.

A further challenge is with regard to preserving the privacy of data and protecting its confidential information, when mining such relational databases. Privacy-preserving data mining is a research area that emerged in 2000 [1]. Its core idea is to extend the traditional data mining techniques in order to work with modified data while masking confidential information. The main objective is to modify the original data while still being able to recover the data mining results from the modified data.

It follows that, considering privacy preserving in relational database mining (already challenging on its own) further increases the complexity of the issue. One of the difficulties is the ability to identify all the inter-relationships between the attributes in a relational database. For example, in a recent study [2] it was shown that, in a relational database which includes multiple linked relations, attributes which otherwise look harmless may be linked to confidential attributes and eventually lead to a privacy breach when building a model.

1.1 Motivation

As stated above, one major strategy in multi-relational database mining is to convert (i.e. flatten) the relational database into a single universal flat file or multiple smaller flat files (views), and then to apply existing data mining algorithms on the resulting dataset. This process of flattening usually involves joining different relations and applying aggregation functions.

Aggregation functions such as *max*, *min*, *avg*, *sum*, *stdv*, and *count* are used to summarize the information stored in multiple records into one record. They take a set of tuples as input, summarize the properties of the set, and finally produce a single value. During this process, new features (attributes) are constructed to contain the summarized information. As such, aggregation changes the set of attributes and the number of records in the resulting table/relation.

Although aggregation functions are extensively used in order to flatten relational databases, studying their direct (and possibly negative) impact on the privacy of a relational database has been overlooked. A recent survey presented in [3] highlights the role of aggregation operators in the context of two privacy preserving techniques, namely, microaggregation and disclosure risk measures. In this scenario, aggregation is considered

an important tool for privacy preservation. However no study, to the best of our knowledge, addresses the link between aggregation and privacy breach during relational database mining. In general, privacy leakage protection in data mining aims to prevent revealing sensitive data while preserving the validity of the data mining results [84; 85; 86]. In order to show such privacy leakage, consider the following example provided in [87]: Suppose that the purchasing directors of BigMart, a large supermarket chain, negotiate a deal with the Dedtrees paper company. If BigMart provides Dedtrees with access to its database (containing records of customers' purchases), in return, Dedtrees offers it a special discount. Access to this database, enables Dedtrees to track its inventory in order to allow "just-in-time" production and stocking. Eventually, this results in reducing its warehouse cost. After accepting this offer, Dedtrees starts mining BigMart's data and finds that customers who purchase cold remedies, later, purchase facial tissue (which allows them to stock us in advance). It also finds that, people who buy skim milk, most likely, purchase green paper. Therefore, it initiates a coupon marketing campaign, e.g. "50 cents off skim milk when you buy Dedtrees products", and improves its sale of green paper substantially. When BigMart tries to re-negotiate with Dedtrees, due to the reduced competition, Dedtrees will not be willing to offer it as low prices as before. As a result, BigMart start to lose business to its competitors (who were able to negotiate a better deal with green paper) [87]. Another work in [90] shows ways to determine the identity of individuals from the trail of information left behind when they use the Internet. For instance, IP addresses from online consumers were linked to publicly available hospital information which correlates to DNA sequence of disease [90]. Another study shows that, data collected as individuals use the World Wide Web to obtain health information, products, and services, also result in privacy risks [91].

In this research, we investigate the impact of applying aggregation functions on the privacy of a relational database, during supervised learning, or classification, where the target concept is known. We build our hypothesis on the fact that aggregation results in constructing new features and hence, changes the number of records and the number (and the type) of attributes in the resulting dataset.

Motivated by these observations, we propose a new methodology detailed in the **Privacy Breach Investigation in Relational Databases (PBIRD)** algorithm to investigate the possible negative impact of aggregation on the privacy of a relational database. This

algorithm consists of three major steps. Inspired by multi-view learning strategy [4] it first constructs different views that consists of subsets of the data, with and without aggregation. It then identifies the selected features associated with each view using the Correlation-based Feature Selection (CFS) [5] algorithm. Finally, it finds the logical relationship between selected features obtained in the previous step.

In this thesis, we also investigate the effect of applying aggregation functions on the classification accuracy and on the time required to build the models. It was mentioned earlier in this section that, aggregation summarizes the information stored in a relational database. With summarization, some of the details are ignored and the original data is modified. Also, due to aggregation, the list of attributes (usually, the number and the type of attributes) and the number of records in the resulting dataset is modified. To this end, given a flat file constructed via joining multiple relations (with and without applying aggregation), we compare the classification accuracy and the model building time and analyze our results.

1.2 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides a literature review on privacy preserving data mining and discusses the research that has been conducted in this area. Chapter 3 involves two main topics. It first provides a literature review on multi-relational database mining and outlines the main existing strategies. It then discusses the concept of aggregation and the challenges surrounding it. Chapter 4 outlines the experimental design of the thesis. It introduces the PBIRD algorithm and explains the techniques that are employed in this algorithm. This chapter also introduces our methodology to perform a comparative study to investigate the role of aggregation on classification accuracy and model building time. Chapter 5 presents the experimental results of applying the PBIRD algorithm to different benchmarking databases. It also presents the results of the comparative study obtained using different classifiers. Finally, Chapter 6 concludes the thesis and highlights possible future work.

Chapter 2

Privacy Preserving Data Mining

Data mining and knowledge discovery in databases aims to automatically extract previously unknown patterns in a large amount of data [6]. In the area of privacy preserving data mining, the data mining algorithms are analyzed for their impact on data privacy. The goal of privacy preserving data mining is to develop algorithms to modify the original dataset so that the privacy of confidential information remains preserved and as such, no confidential information could be revealed as a result of applying data mining tasks.

In this chapter, we first provide background information about privacy preserving data mining. Then, the major dimensions in privacy preserving data mining i.e. data modification, data distribution, and data or rule hiding, are discussed. This is followed by a detailed discussion of anonymization techniques along the three main privacy preserving techniques for data mining. Finally, we present some recent works addressing privacy preserving in multi-relational databases.

2.1 Background

In privacy preserving data mining, the following two concerns need to be addressed. Firstly, the private raw data such as names, addresses, identifiers, and so on should be either removed or modified from the original database. Therefore, the recipient of the data will not

be able to compromise the confidentiality of another individual. Secondly, the private and confidential knowledge that could be discovered using different data mining algorithms and techniques should also be removed or modified [7].

The work in [8] identifies four main characteristics of a PPDM (Privacy Preserving Data Mining) algorithm. A PPDM algorithm should prevent the discovery of sensitive information, be resistant to different data mining techniques, not compromise the access and the use of non-sensitive data, and should not have exponential computational complexity.

The key point in studying privacy preserving data mining is to define the level of privacy protection that is required. The fundamental question is: to what extent does privacy need to be protected? The work in [9] presented a strict definition of privacy protection. It stated that, access to the published data should not allow the attacker to learn anything extra about target victims compared to no access to the database, even if the attacker has access to background information from other sources. A later work [10] showed that, this type of strict privacy protection, due to inevitable existence of background knowledge, is not attainable. Background knowledge refers to domain specific information held by an attacker that could be used to infer confidential information. Most literature in Privacy Preserving Data Publishing (a.k.a. PPDP) consider another definition that is more practical. Rather than the assumption of not having any background knowledge, these literatures assume that the attacker has only limited background knowledge [11].

2.1.1 Privacy Models

It is possible to classify the privacy models into two broad categories based on their attack principles. In the **first category**, privacy is at risk if the attacker is able to link a record owner to a record in the published table (record linkage), or to a sensitive attribute in the published table (attribute linkage), or to the published table itself (table linkage).

In all of these three models i.e. record linkage, attribute linkage, and table linkage, the assumption is that, the attacker knows the Quasi-Identifier (QID) of the victim. Furthermore, in the case of record linkage and attribute linkage, we assume that, the attacker knows that the victim's record exists in the published table. However, in the case of table linkage the attacker tries to determine the presence (or absence) of the victim's record in the published

table. A given published data is considered to be privacy preserving if it is able to prevent the attacker from applying these linkages.

The **second category**, addresses the uninformative principle identified in [12]. The published table should only provide the attacker with little additional information, further than the background knowledge. If the variation between prior and posterior belief of attacker is large, the attack is called a probabilistic attack [11]. This model of privacy does not focus on which particular records, attributes, and tables the attacker is able to link to a target victim. Rather, the focus is on how the attacker would change his/her belief on the sensitive information after he/she accesses the published data.

2.1.2 Example of Record Linkage and K-Anonymity

To illustrate how privacy leakage can result from publishing data, consider the following example provided in [13]. A hospital releases patient records to enable researchers to study the characteristics of different diseases. The released information does not contain the names, IDs, and other individually identifiable attributes of the patients. There are, however, some attributes that, combined with external databases, enable an attacker to access confidential information. In our example, a hospital releases the data in Table 2.1 after removing the names of the patients. If an attacker obtains access to the (publicly available) voter's registration list shown in Table 2.2, he could easily discover the identity of the patients by joining the two tables on {Age, Sex, Zipcode}. These attributes are the Quasi-Identifier (QID) attributes.

To prevent record linkage, Samarati and Sweeney [14; 15] proposed the notion of k-anonymity. A table is considered *k-anonymous* if it satisfies the following condition: assuming that a given record in this table has a particular QID value, there are at least (k-1) other records which have the same QID. This indicates that, the probability of linking a victim to a particular record is at most 1/k. Furthermore, having the k value in the denominator implies that as the value of k increases, the probability of linkage decreases proportionally.

Table 2.1: Patients' Microdata **[13]**

ID	Attribute			
	Age	Sex	Zip code	Disease
1	28	M	83661	Headache
2	25	M	83634	Headache
3	30	M	83967	Cough
4	38	F	83949	Toothache

Table 2.2: Voter Registration List **[13]**

ID	Attribute			
	Name	Age	Sex	Zip code
1	Mark	28	M	83661
2	Joe	25	M	83634
3	Tom	30	M	83967
4	Judith	38	F	83949

Table 2.3 shows a 2-anonymous table corresponding to Table 2.1. Even though the attacker has access to the voter registration list, he could only infer that Tom may be the person in the last two records of the table. Therefore, the exact record is identifiable only with a probability of 50%.

Table 2.3: The 2-anonymous table corresponding to Table 2.1 **[13]**

ID	Attribute			
	Age	Sex	Zip code	Disease
1	2*	M	836**	Headache
2	2*	M	836**	Headache
3	3*	*	839**	Cough
4	3*	*	839**	Toothache

Contrary to the traditional privacy preserving techniques such as adding noise, or swapping, in k-anonymity, the information remains truthful. K-anonymity, however, does not provide enough protection against attribute linkage. If we consider a table that has no sensitive attribute, e.g. the Voter Registration List (Table 2.2), the attacker could possibly use its QID {Age, Sex, Zipcode} to link to the sensitive attribute in an external source. Two other limitations associated with the k-anonymity model are as follows. Firstly, it is difficult

for the owner of the database to know which attributes are or are not in the external table. Secondly, the k-anonymity model makes an assumption of a certain attack and in reality; there is no reason for the attacker not to choose another type of attack.

Some of these limitations are described in the context of our example. Even though Table 2.3 is anonymized, an attacker with background knowledge could infer sensitive attributes of individuals in the table. For example, if he knows Tom's age and zip code, he could conclude that Tom belongs to the last equivalence class. Then, with extra background knowledge, such as knowing that Tom has a very low risk of toothache, the attacker could infer that Tom most likely has a cough. In the case of homogeneity attack, assuming that the attacker knows that Mark is 28 years old and lives in zip code=83661, and his record is in the table, then the attacker could infer that Mark has headache.

Anonymization operations hide information in a way that multiple records become indistinguishable with regard to the QID' (or the anonymized QID). Therefore, a record will be linked to all records with the same QID' rather than to an individual record which has a given QID. Although k-anonymity is considered a popular anonymization approach, the evaluation of the actual re-identification probability was overlooked. A recent study by El Emam and Dankar [16] addresses this issue. The results of this study show that over-anonymization causes excessive distortion of the data and degrades the quality of data for subsequent analysis. In relational databases, it is possible to link a set of seemingly harmless attributes across multiple relations to confidential information [2]. Therefore, in these databases, anonymization of the private information, alone, does not guarantee privacy preservation.

The challenge of anonymization is to produce an anonymous table which could satisfy the privacy preserving requirement determined by a given privacy model and simultaneously retain the utility of the anonymized table. The fundamental question with regard to choosing a given anonymization operation is that, how much anonymization in a given table is needed? A given table is either *minimally* or *optimally* anonymous. The minimally anonymous table satisfies the given privacy requirement and it is not possible to reduce its sequence of anonymization operations without violating the requirements. On the other hand, the optimally anonymous table satisfies the given privacy requirement and has most information based on the chosen information metric among all other satisfying tables. It is

more reasonable to find a minimally anonymous table since finding the optimal anonymization is NP-hard [11].

To measure the utility of an anonymous data, an information metric is used which refers to the ability of retaining information in order to keep the published data practically useful [11]. Two broad categories of information metrics, namely, *data metric* and *search metric* are identified. *Data metric* measures the data quality in the anonymous table and compares it to the data quality in the original table. *Search metric* guides an anonymization algorithm step by step to find an anonymous table which has maximum information, and minimum distortion. The identified anonymous table is eventually evaluated using data metric.

In the next section, the main dimensions in privacy preservation are discussed.

2.2 Privacy Preserving Dimensions

The main approaches adopted to deal with privacy preserving data mining include data modification, data distribution and, data or rule hiding.

Data Modification: Usually a given table with private information is modified prior to publishing. Modification of data is performed via applying anonymization operations which include suppression, generalization, anatomization, permutation, and perturbation. Selective modification of data is another method that belongs to this dimension and tends to achieve a higher utility for the modified data while preserving its privacy [7].

Data Distribution: This dimension considers the distribution of data. Some approaches have been developed for centralized data while other methods address distributed data scenarios. Distributed data scenarios are also classified into horizontal or vertical data distribution. In horizontal data distribution, different database records reside in different locations. In vertical data distribution, all the values for different attributes reside in different locations [7].

*Data or Rule Hiding***:** This dimension refers to whether or not raw data or aggregated data should be hidden. Since the complexity of hiding aggregated data in the form of rules is high, heuristic approaches have been developed to address this problem [7].

2.3 Anonymization Operations

Typically, to satisfy privacy requirement, the original table with sensitive and confidential information needs to be modified before release. The goal of anonymization [17] is to hide the identity and/or the sensitive data of record owners and retain the sensitive data for data analysis purposes. K-anonymity is achieved using these anonymization operations. In order to achieve anonymization, a sequence of anonymization operations needs to be applied.

In the most basic form, the data owner has a table of the following format [11].

D(Explicit_Identifier, Quasi_Identifier, Sensitive_Attributes, Non-Sensitive_Attributes)

where *Explicit_Identifier* refers to a set of attributes such as name or SSN with information that explicitly identifies record owners; *Quasi_Identifier* (a.k.a. QID) is a set of attributes which could potentially determine record owners. *Sensitive_Attributes* refer to person-specific confidential attributes such as salary, disease, and so on. Finally, *Non-Sensitive_Attributes* contains the attributes that do not fall into any of the above categories.

The anonymous table satisfying the above privacy preserving constraints will have the following format:

T(QID', Sensitive_Attributes, Non-Sensitive_Attributes)

Where QID' is an anonymous version of the original QID which is obtained by applying anonymization operations to the attributes in QID.

In general, three major categories have been defined in order to perform the anonymization task on a given table. These categories include "*generalization and suppression"*, "*anatomization and permutation"*, and "*perturbation"*. A detailed description of each category is provided below:

2.3.1 Generalization and Suppression

In this category, values such as QID attributes with specific descriptions are replaced with less specific descriptions. Considering the taxonomy illustrated in Figure 2.1, **generalization** tends to replace some of the values with their parent values. The reverse operation of generalization is called specialization.

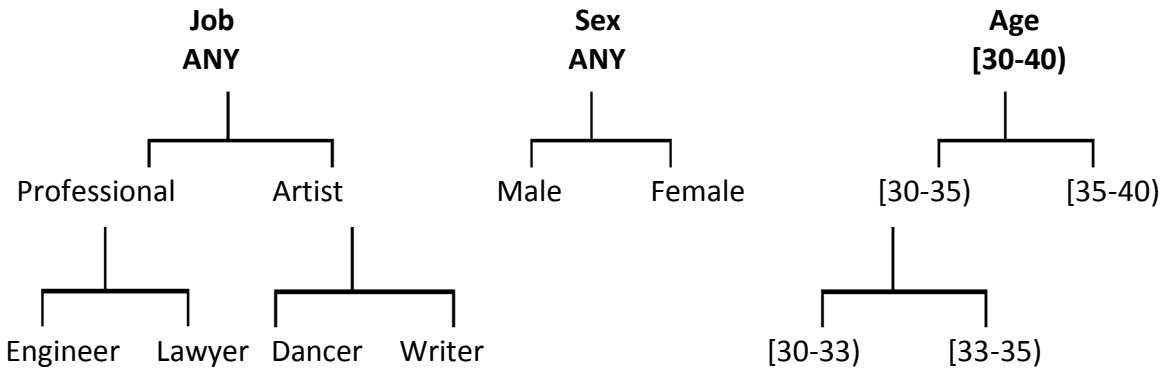


Figure 2.1: Taxonomy trees for Job, Sex, Age **[11]**

Generalization schemes are categorized as *full-domain generalization scheme, subtree generalization scheme, sibling generalization scheme, cell generalization scheme, and multidimensional generalization scheme* [11]. These schemes are described below.

Full-domain generalization scheme [18]: all values for a given attribute are generalized to the same level in the taxonomy tree. If Dancer and Writer are generalized to Artist, then it is required that Engineer and Lawyer be generalized to Professional. Because of this generalization, the search space required for this scheme is smaller than other schemes; however, data distortion is higher.

Subtree generalization scheme [18]: each branch of the taxonomy tree is considered separately. In other words, either all children are generalized or none of them. Referring to our taxonomy, if Dancer is generalized to Artist, other child i.e. Writer must be generalized as well, but Engineer and Lawyer remain intact.

Sibling generalization scheme [18]: in contrast to subtree generalization, some of the siblings can remain ungeneralized. An example would be generalizing Engineer to Professional and keeping the Lawyer unchanged.

Cell generalization scheme [18]: contrary to other previously mentioned schemes where all of the instances (records or tuples) in a given table are generalized, in cell generalization some instances may remain ungeneralized. This scheme is more flexible compared to other above mentioned schemes and generates less data distortion. However, data utility degrades.

Multidimensional generalization scheme [19; 20]: allows two QID groups which have the same values to be generalized into different parent groups independently. For example {Writer, Male} could be generalized to {Writer, ANY_sex} while {Writer, Female} could

be generalized to {Artist, Female}. The generalized table has both Writer, and Artist. This scheme results in less data distortion compared with full-domain and subtree generalization schemes.

In **suppression**, values are replaced by special values. The reverse operation of suppression is known as disclosure. Three different suppression schemes are defined; namely, Record suppression, Value suppression, and Cell suppression. In *Record suppression*, an entire record gets suppressed [18]. *Value suppression* refers to suppressing every instance of a given value [21], and finally, *Cell suppression* tends to suppress some instances of a given value in the table [17].

2.3.2 Anatomization and Permutation

These two methods tend to disassociate the correlation between sensitive attributes and QID. This is achieved via grouping and shuffling of confidential attributes in a given QID group.

Anatomization [22] does not modify the QID and the sensitive attributes; instead, it disassociate the relation between the two. In this method, data are released in two separate tables; one table containing the QID, and the other table containing the sensitive attributes. Both tables have a common attribute named GroupID. Records in the same group have the same GroupID and therefore, are linked together. If a group has x distinct sensitive values and each of those distinct values occurs once in the group, therefore the probability of linking a record to a sensitive value using GroupID, is 1/x.

Permutation [23] dissociates the correlation between QID and numerical sensitive attributes. It divides data records set into different groups and then shuffles their sensitive attribute (since they are numerical) within each group.

2.3.3 Perturbation

Perturbation is a simple and efficient method that preserves statistical information. It includes distortion of the data via adding noise, swapping values, aggregating values, blocking values, sampling, and generating non-real and synthetic data based on some statistical information obtained from the original data set.

An example of adding noise could be the alternation of an attribute value by another value, such as changing 1-value to 0-value and 0-value to 1-value. Swapping is interchanging values of individual records. Aggregation or merging refers to combination of several values into a coarser category. In the case of blocking, attributes values are replaced by "?". Sampling refers to releasing a sample of dataset rather than releasing it all.

The general idea in perturbation is to replace the original data values with other values such that there would be no significant difference between the statistical information of the perturbed data compared with the original data. What makes perturbation different from other anonymization operations is the fact that, the resulting datasets are synthetic, do not correspond to the original records, and have no real meanings. Therefore, instead of publishing the perturbed dataset, the data publisher releases the statistical information or the data mining results [24].

In choosing different generalization schemes as defined in Section 2.3.1, a trade off between search space and amount of distortion should be taken into consideration. Full domain generalization has the smallest search space and the highest distortion where cell generalization has the lowest distortion and largest search space. In generalization and suppression, domain values are lost and data is less precise but semantic consistency between the original data and the generalized data exists and therefore, the truthiness of data is preserved. In anatomization and permutation, data remain unmodified, the resulting anonymized table answers aggregate queries more accurately, and domain values are retained. However, because the data are published in two separate tables (one containing QID, and the other containing the sensitive attributes), it is not clear how standard data mining tools could be applied. Perturbation is a simple and efficient technique that provides good privacy and prevents an attacker from recovering sensitive information, performing sensitive linkage, and preserves statistical information. However, one drawback associated with perturbation is that, published records are "synthetic" and do not correspond to real-world values that exist the original data [11]. The characteristics of different anonymization operations are summarized in Table 2.4. In the next section, we introduce the main privacy preserving techniques used in data mining.

2. 4 Privacy Preserving Techniques

The three main privacy preserving techniques for data mining are, namely, heuristic-based techniques, cryptography-based techniques, and reconstruction-based techniques [7].

2.4.1 Heuristic-based Techniques

The goal of heuristic-based techniques is to hide sensitive rules in the original data and yet, to maintain the quality of the data. Heuristic-based approaches are mainly used in centralized database scenarios. They are similar to adaptive modification that modifies only selected values rather than all available values in order to minimize the utility loss. These techniques include "*centralized data perturbation-based association rule confusion", "centralized data blocking-based association rule confusion", and "centralized data blocking-based classification rule confusion".* These techniques are algorithm specific and have been developed for different data mining algorithms such as classification, association rule discovery, and clustering [7].

2.4.2 Cryptography-based Techniques

These techniques are mainly used in distributed environments and tend to address the Secure Multiparty Computation (SMC) problem. The nature of SMC problem is described as follows [7]. Assuming that two or more parties want to perform a computation based on their private inputs; none of the participating parties wants to disclose its own output to anybody else. In SMC, the computation is considered secure if, at the end of computation, none of the involved parties is able to obtain any extra information other than their own input and output. In particular, an SMC problem deals with computing a probabilistic function in a distributed network such that, each participant holds one of the independent inputs. Then, the computation is performed, and other than the participants own outputs, no further information is revealed to no other one.

Table 2.4: Characteristics of the anonymization operations

Technique	Pros	Cons
Generalization and Suppression	• Modified data is semantically consistent with original data • Preserve the truthiness of data • Suitable for continuous data publishing	• Domain values are lost
Anatomization and Permutation	• The data remain unmodified • Anonymized tables answer aggregate queries more accurately. • Domain values are retained.	• Data is published in two tables (one containing QID and the other sensitive attribute) and it is not clear how standard data mining tools could be applied • Not suitable for continuous data publishing
Perturbation	• Simple and efficient • provides good privacy and prevent attacker from recovering sensitive information and performing sensitive linkage • Preserves statistical information	• Published records are "synthetic" and do not correspond to real-world values in original data • Can only reconstruct the distribution of data not the original data value

2.4.3 Reconstruction-based Techniques

These techniques are applied on both centralized and distributed data. In order to preserve privacy, data is first perturbed and then, its distribution is reconstructed at an aggregated level in order to apply data mining. Depending on the type of data (being numerical, binary, or nominal), different approaches have been followed. For binary and nominal data, association rule mining is employed [25]. For numerical data (as presented in [1]), we build a decision tree classifier from training data where the values of individual records have been perturbed. Since it is not possible to estimate the original values of individual records accurately, a reconstruction procedure is employed to estimate the distribution of the original values. Then, the reconstructed distribution is used to build classifiers whose accuracy is close to the accuracy of the classifiers that were built from the original data.

The heuristic-based techniques are mainly used in centralized environments. These techniques hide sensitive information using perturbation techniques based on probability distribution. There are several heuristic-based approaches that hide both raw and aggregated data using different hiding techniques such as k-anonymization, data swapping, generalization, sampling, and adding noises.

The cryptography-based techniques are problem specific. They are applied on distributed data and in scenarios where more than one party is involved to securely and collaboratively perform a computation task based on their private inputs. One drawback of these techniques is that, they do not protect the output of the computation, and the privacy leakage is prevented only during the computation process [26].

Finally, reconstruction-based techniques address privacy preservation differently. These techniques perturb the data and reconstruct the distribution at an aggregate level and can be applied on both centralized and distributed data. The characteristics of these techniques are summarized in Table 2.5.

Table 2.5: Characteristics of privacy preserving data mining techniques

Technique	Characteristics
Heuristic-based	• Applied on centralized data • Modifies selected values rather than all values to minimize utility loss • Algorithm specific • Mostly to hide aggregated data in the form of rules
Cryptography-based	• Applied on distributed data • Problem specific: to address Secure Multiparty Computing (SMC) issue • Prevents privacy leakage in the process of computation and does not protect the output of computation.
Reconstruction-based	• Applied on both centralized and distributed data • Reconstructs the distribution of the data not the original data itself

2.5 Technique Selection Criteria

In choosing different privacy preserving techniques, a number of selection criteria should be taken into account. In general, no privacy preserving algorithm can outperform others on every aspect. The work presented in [7] identifies four evaluation criteria i.e. *performance*, *data utility*, *level of uncertainty*, and *resistance* in order to select a suitable privacy preserving technique.

Performance refers to the computational cost to achieve privacy preserving. It also refers to the time required and the average number of operations needed by each algorithm to achieve its goal. In the case of distributed data, where there are number of collaborating sites, the communication cost to exchange information between different sites will also have an impact on the overall performance.

Another important criterion is the utility of the data after applying privacy preserving algorithms. Privacy preserving data mining should be able to successfully perform data mining operations on a dataset without revealing the confidential information. The privacy preserving techniques should be able to minimize the privacy loss, the information loss, and the loss of functionality of the data. Measuring the data utility is not only associated with the modification of data. Even if the information stored in the database is not modified (e.g. sampling where a sample of dataset is released) the utility of the data degrades and eventually, the resulting data set and the original data set are different.

Uncertainty level is considered another evaluation parameter and refers to the level of uncertainty in predicting the sensitive information that is hidden. In general, the algorithm with maximum uncertainty level will be preferred over other algorithms.

The final evaluation criteria are the resistance of privacy preserving algorithm to different data mining techniques. Since the main goal of privacy preserving algorithms is to protect confidential information against attacks, the attacker could use different data mining algorithms to achieve his goal. Therefore, it is worthwhile to mention that, even though a given privacy preserving algorithm is protecting the private data against a particular data mining algorithm, it may not provide the same protection against other data mining techniques [7].

2.6 Privacy of the Data Mining Results

Two broad research directions to study the privacy threats caused by releasing data mining results or patterns are identified as follows.

The first approach is to anonymize the data in order to prevent generating sensitive data mining patterns. The work presented in [27] argues that the suppression of sensitive values is not enough, because it is still possible for an attacker to use association rules that are learnt from the data to predict the suppressed values. To prevent such an attack, a heuristic algorithm was proposed in order to suppress a minimal set of values. Another work [28] proposed a method to hide one rule at a time by either decreasing its support or confidence, which is achieved by removing items from the transactions. As such, if the rules satisfy a specified minimum support and a minimum confidence, they are removed.

The second approach anonymizes the data mining patterns directly. If the goal is not to disclose data, but rather the data mining result, sanitizing the mined patterns will provide us with a better quality than mining anonymized source data [29]. The pattern sanitization process focuses only on the portions of data relevant to the harmful patterns. Therefore, we obtain a better information utility compared with anonymizing data source and performing data mining on the anonymized data set. In [30], an evaluation method to measure the privacy loss caused by publishing data mining results was proposed.

2.7 Privacy Preserving in Multi-relational Data Mining

Privacy preserving in multi-relational database mining is a relatively recent field of study and hence, very few literature have addressed it. The major works done in this area are discussed as follows.

2.7.1 Multi-relational K-Anonymity

In general, k-anonymity approach deals with anonymization of a single table. To achieve k-anonymity in multi-relational databases, Nergiz et al. [31] introduced a privacy model called Multi-Relational k-anonymity (MultiR k-anonymity). It extends the k-anonymity from a

single relation to multiple relations and shows that the previous k-anonymity models fail to protect the privacy and cause reduction of utility of the data in multiple relations.

MultiR k-anonymity guarantees privacy when multiple tables are released. These released tables consist of a person-specific table PT which contains personal identifiable information and sensitive attributes, and also a set of tables (T_1 , T_2 , ... , T_n) which contain sensitive attributes, Quasi-Identifier attributes, and foreign keys that refer to attributes in PT. In Multi-relational k-anonymity the (X,Y)-anonymity condition is used and translated to operate on the join $PT \bowtie T_1 \bowtie T_2 \bowtie ... \bowtie T_n$. If in the result of the join, each group of tuples that share the same Quasi-Identifier value refers to at least k different respondents, the multi-relational k-anoymity is preserved.

2.7.2 Privacy Leakage in Multi-Relational Databases via Pattern Based Semi-Supervised Learning

The work in [32] introduces a data mining framework in order to identify potential privacy leakage in a multi-relational database. Based on this framework, it is possible to detect data leakage using different semi-supervised learning techniques such as K-Nearest Neighbour (KNN). This research defines a new approach for semi-supervised learning called Hyperclique Pattern-based Semi-supervised Learning (HPSL) which is different than the traditional semi-supervised learning methods. This is due to the fact that this technique considers the similarity amongst a set of objects instead of a pair of objects. It shows that either HPSL or KNN approaches have the potential to compromise the privacy of the relational database.

Figure 2.2 illustrates a view that is a content/context-dependent subset of one or more table. This view has *m* attributes and *n* tuples (objects) and this information is known to the user. There are also *p* attributes that exist in the base table but not in the view. Although the *p* values are hidden from the user, the research shows that using semi-supervised learning techniques, the user may predict the *p* attributes for those objects.

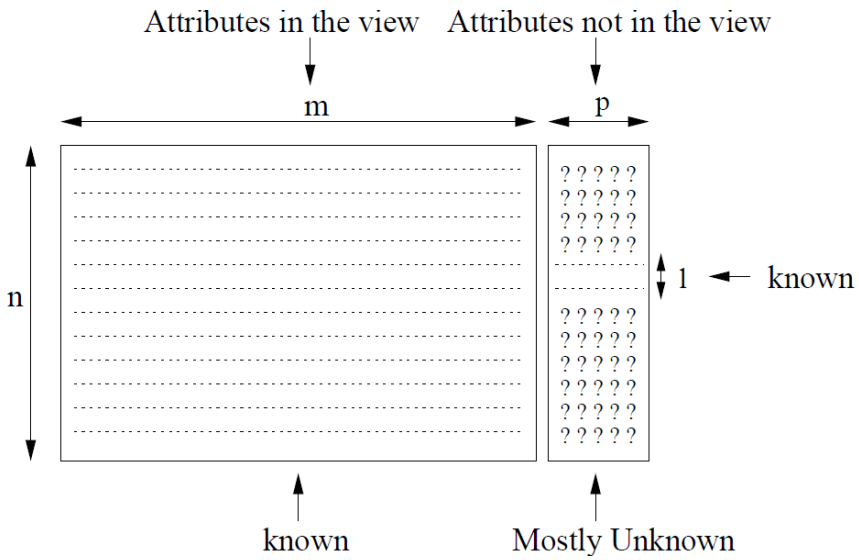


Figure 2.2: Illustration of the problem **[32]**

2.7.3 Identifying and Preventing Data Leakage in Multi-Relational Classification

The work presented in [2] shows the potential privacy leakage in multi-relational databases via linking seemingly harmless attributes to confidential information. Given a financial database, the goal is to determine whether a loan is considered to be high or low risk. The assumption is that, the database contains a confidential attribute e.g. income level which should not be disclosed. To protect this attribute, it might be decided to distort or eliminate it using existing privacy preserving techniques. However, even after removing or distorting this attribute, a learning model can accurately predict the income level. This research first, shows such privacy leakage and then, proposes a method to generate a ranked list of subschemas. These subschemas maintain high accuracy in predicting the class attribute while limit the disclosure of the private attribute and the predictive accuracy against it.

In multi-relational databases, in addition to the knowledge obtained from the target relation, the information related to the target relation (i.e. background information) stored in other relations also needs to taken into consideration as well. One major challenge is related to the very nature of relational databases and the fact that, existing data mining algorithms cannot be applied directly on them. Different approaches to deal with such issue are

discussed in details in Chapter 3. The fundamental question is the following: considering different approaches to mine relational database which approach is most suitable to preserve the privacy.

2.8 Summary

In this chapter, two broad categories of privacy models were studied. The first category addresses the privacy threat that occurs when an attacker is able to link a record owner to a record in the published data (record linkage), a sensitive attribute in a published data (attribute linkage), or to the published table itself (table linkage). The second category deals with probabilistic attack and aims to achieve the uninformative principle.

Three major privacy preserving dimensions i.e. data distribution, data modification, and data or rule hiding were addressed. Next, we discussed data modification, and in particular the anonymization concept, in more details. Three main groups of anonymization techniques , “Generalization and Suppression”, “Anatomization and Permutation”, and “Perturbation” were presented. Following, the three main privacy preserving techniques for data mining, namely, heuristic-based techniques, cryptography-based techniques, and reconstruction-based techniques were explained. Finally, three recent studies related to privacy preserving in multi-relational databases were discussed. These include, multi-relational k-anonymity [31], privacy leakage in multi-relational databases via pattern based semi-supervised learning [32], and identifying and preventing data leakage in multi-relational classification [2].

In the next chapter, a detailed discussion of multi-relational database mining and aggregation is provided.

Chapter 3

Multi-relational Database Mining and Aggregation

Relational databases contain multiple relations that typically consist of numerous rows or tuples. These repositories require specialized data mining techniques when aiming to mine them directly. To this end, a number of multi-relational database mining techniques have been developed. Many of these techniques employ some form of aggregation.

This chapter presents a review of multi-relational database mining and aggregation. We first introduce classification as one of the major data mining tasks and then focus on multi-relational classification. Next, we discuss the main strategies in mining relational databases and present the positive and negative aspects associated with each strategy. The role of aggregation in summarizing information and the properties of aggregation operators, are discussed in the following section. Finally, the applications of aggregation operators to privacy protection techniques as well as the main challenges associated with aggregation functions are investigated.

3.1 Classification

Classification, clustering, and association rule mining are considered the three typical data mining tasks [6]. In this thesis, we pay particular attention to the classification task. We

discuss classification and explain some of the classification algorithms which are employed in this research.

Classification and prediction are considered two forms of data analysis used to extract models that describe data classes, or to predict future data trends. Classification and prediction refer to supervised learning where the class label for each training sample is known. The attributes are split into independent (explanatory) variables and dependent variable(s) and the learning task is to specify the relation between the two [6].

Classification predicts categorical labels or class labels (such as categorizing bank loan applications to be good or bad) whereas prediction is concerned with modeling continuous-valued functions (predicting the expenditure of potential customer to purchase computers given their age and income) [6].

Han and Kamber [6] define classification as a two-step process following the data pre-processing stage. The first step includes building a model which describes a predetermined set of data concepts or classes (a.k.a. Learning) and the second step includes using that model for classification (a.k.a. Classification). In the first step, a classification algorithm is used to analyze the training data and to obtain a learning model in the form of classification rules against label class. In the second step, test data is used to estimate the accuracy of the obtained classification rules. If the accuracy is acceptable, the rules are subsequently used to classify new instances.

3.2 Classification in Multi-relational Database Mining

Following the general definition of data mining, relational database mining looks for useful patterns across multiple relations (tables) in a relational database. A relational database $\boldsymbol{R}$ includes a set of tables $\{R_1, R_2, \ldots, R_n\}$ where each table contains a set of tuples (records), a primary key, and some foreign keys. The foreign key attributes of a given table are linked to the primary keys of other tables.

In a multi-relational classification setting, a database $\boldsymbol{R}$, consists of a target table R_t, and a set of background tables $\{R_{b1,} R_{b2,}, \ldots, R_{bn}\}$ [33]. The target table R_t contains a target variable T and each of the tuples in the target table is associated with a class label that

belongs to *T*. The relational classification task as defined in [33] is to find a function *F*(x) that maps each tuple *x* from the target table R_t to the category *T*:

$$T = F(x, R_{b1\ldots bn}, R_t), x \in R_t \quad \text{(Equation 3.1)}$$

3.3 Mining Relational Databases

Recall that, when applying classification algorithms (and generally to apply data mining algorithms) on relational databases, we face the following challenge. Most of the existing data mining algorithms are applied on a single flat file and cannot be applied directly on relational databases. To address such a problem, two main strategies have been introduced: either to create new algorithms that deal with the relational database directly, or to solve the problem in two steps, i.e. to convert the relational database into a universal flat file and then, to apply traditional data mining algorithms on it. These two strategies are called "upgrade" and "propositionalization (flattening)" respectively. We will discuss these two categories next.

3.4 Upgrade Strategy

This category of algorithms includes techniques which extend (upgrade) existing propositional methods so they may be applied directly to relational databases. In other words, instead of searching for features in a single relation, the search for features includes multiple relations (target and background relations). For example, it is possible to search exhaustively or to search heuristically (such as greedy search, best-first search, etc.) [34].

Many of the existing upgrade methods use Inductive Logic Programming (ILP) techniques to search and explore information across different relations in the relational database. In the following sections, we will first introduce ILP and then discuss two of the proposed upgrade techniques.

3.4.1 Inductive Logic Programming (ILP)

Inductive Logic Programming is defined as the intersection between inductive learning and logic programming [35]. It uses techniques from machine learning as well as logic programming. The main goal of ILP, as explained in [36], is "to develop tools and

techniques to induce hypothesis from observations (examples) and to synthesize new knowledge from experience". ILP addresses two main problems which exist with the classical machine learning techniques. The first problem is related to the limited formalism for knowledge representation. The second problem arises from difficulties using substantial background knowledge during the learning process [36].

There have been different techniques which upgrade the existing propositional learners. Two of the well-known and state-of-the-art algorithms in this category are FOIL and CrossMine, which are described below.

3.4.2 First Order Inductive Learner (FOIL)

The FOIL algorithm is considered one of the best-known techniques that deal with structured data as contained in databases [37]. It upgrades the CN2 [38] method by enabling it to deal with first order representation. CN2 is a rule induction algorithm which allows the application of statistical methods in generating the if-then rules. It is designed to work efficiently, even when the data is noisy. The FOIL algorithm learns function-free Horn clause definition of a target relation [37] and is considered a top-down learner. (Horn clause refers to a clause with at most one positive literal used in logic programming and constructive logic). Its goal is to build a set of rules to cover many positive and few negative examples. The FOIL algorithm consists of two main stages, namely, *separate* and *conquer*. In the separate stage, the algorithm begins a new clause and in the conquer stage, it constructs a conjunction of positive or negative predicates serving as the body of the clause [39]. Figure 3.1 illustrates the FOIL algorithm.

```
Let Pred be the predicate to be learned.
Let Pos be the positive examples.
Until Pos is empty do:
Let Neg be the negative examples
Set Body to empty
Call LearnClauseBody
Add Pred ←Body to the rule
  Remove from Pos all examples that satisfy the Body.
Procedure LearnClauseBody
Until Neg is empty do:
Choose a literal L.
Conjoin L to Body.
Remove from Neg examples that do not satisfy L.
```

Figure 3.1: The FOIL algorithm [39]

We use the following two-class problem example (Figure 3.2) provided in [33] to describe the algorithm.

Account Table

account_ID	frequency	balance
132	monthly	3521
132	weekly	2000
240	monthly	20
100	weekly	50
98	weekly	10000

Loan Table

loan_ID	account_ID	amount	duration	payment	status
1	132	2000	12	100	Good
2	132	4000	12	200	Good
3	148	1000	24	80	Bad
4	148	30000	24	1000	Bad
5	98	10000	36	800	Good

Figure 3.2: Loan, Account schema adapted from PKDD 1999 Discovery Challenge [33]

The task of the FOIL algorithm is to construct a set of conjunctive rules to separate good and bad loans (i.e. positive and negative classes) in the target relation using information in both the Loan and the Account tables. The algorithm searches for relevant features across the two tables. It then evaluates possible predicates in order to find the best rule that separates positive and negative tuples in the target table. To define the predicate, let us consider "Account (A, f='monthly',-)". This predicate describes the Account (A) which has a frequency value set to 'monthly'. In general, predicates are constraints on attribute(s) (in this case frequency attribute) in a given table.

The final result of the learning process is to obtain a set of rules used to distinguish good loans from bad loans. An example of such rules could be "Loan (N, +):-Loan (N, A, A,-,-), A>10000, Account (A,-, B), B>10000". This rule says that if the amount of loan is more than $10,000 and the account balance is more than $10,000, the loan is considered to be good.

One drawback of this algorithm is that, it is very time-consuming especially with data that has many relations or predicates [40]. This was the motivation behind designing another ILP-based algorithm called CrossMine, which is discussed next.

3.4.3 CrossMine

CrossMine [40] is considered an extension of the FOIL approach. Its main focus is to enhance the scalability of FOIL with regard to the number of relations in the database. It thus addresses a general issue in the ILP-based approaches. This issue is related to the fact that most of the ILP-based systems evaluate many predicates separately and hence are not efficient [40].

CrossMine uses a sequential covering algorithm that constructs classes repeatedly and removes positive examples which are covered by each clause. Unlike other multi-relational classification approaches which only use simple positive and negative predicates, CrossMine also uses predicates which involve aggregations on attribute level. It uses a novel method of Tuple ID propagations. Tuple ID propagation is a method to virtually join target relation with non-target relations. It is an effective and flexible method which is much less expensive than physical join in terms of required time and space [40].

The idea of Tuple ID propagation is to pass and store information about the primary keys and classes of the target relation into background relations. The outcome of such approach is to calculate the FOIL gain in each local relation without a need to perform physical joins. Here, the FOIL gain refers to the goodness of a predicate with regard to the target relation (Loan in our example). We will use the example provided in Figure 3.3 to explain the algorithm.

As shown in Figure 3.3, after propagation, the Account table contains two new features (IDs, and class_labels). The information about the target relation (Loan) such as Loan.account_ID and Loan.class are added to their corresponding columns. Let us take account_ID = 45 as an example. There are two loans associated with this account ID, i.e. loan_ID = {4, 5}. Each of these two loans have a different class labels (‘+’ is associated with loan_ID = 4 and ‘-’ is associated with loan_ID = 5). The loan_IDs and their corresponding classes are added to the new columns. So, in Figure 3.3, for the IDs attribute of account_ID = 45 we obtain {4, 5} and in class_lables attribute we obtain {1+, 1-}. The information in

these two columns is then used to calculate the FOIL gain information of each predicate in the account column. This eliminates the need to physically join the two tables and hence results in a better accuracy and a faster model build time.

Loan

loan_ID	account_ID	amount	duration	payment	class
1	124	1000	12	120	+
2	124	4000	12	350	+
3	108	10000	24	500	-
4	45	12000	36	400	-
5	45	2000	24	90	+

Account

account_ID	frequency	date	IDs	class_labels
124	monthly	960227	1,2	2+,0-
108	weekly	950923	3	0+,1-
45	monthly	941209	4,5	1+,1-
67	weekly	950101	-	0+,0-

Figure 3.3: Example of Tuple ID propagation [40]

3.5 Flattening (Propositionalization) Strategy

The “flattening” (or propositionalization) techniques flatten multiple relations in a given multi-relational database into a universal single relation. Propositionalization is defined as the transformation of multi-relational data, which contains structured examples, into an attribute-value representation suitable for common data mining algorithms. This process could be thought of as summarizing data that is stored in multiple tables into a single table [41]. An important feature of propositionalization strategy is therefore, the flexibility of using any propositional learner once the universal flat relation is created. In other words, instead of upgrading the existing propositional learners and re-inventing the wheel, we can use the base algorithms “off the shelf”. The challenge, however, is how to transform multiple relations into a single relation efficiently, and to minimize all of the drawbacks resulting from such a transformation.

Propositionalization methods are further categorized into two families, namely Logic-oriented methods and Database-oriented methods. In general, none of the two outperforms the other in terms of classification accuracy [42].

3.5.1 Logic-Oriented Methods

Logic-oriented methods define propositional attributes using pure logic, i.e. existential features. They are able to handle complex background knowledge and to provide expressive first-order models. LINUS is one of the well-known logic-oriented methods and is described below.

LINUS

LINUS algorithm was first introduced by Lavrac [43] in 1990. It is considered one of the influential systems in the area of propositionalization. It is a transformational ILP learner in which the background knowledge is used to introduce new attributes during the learning process. LINUS algorithm involves three steps:

Step 1: Transforming the structural data into an attribute value, flat file format.

Step 2: Implementing one of the existing single table learning algorithms in order to construct a propositional hypothesis (if then rules).

Step 3: Transforming the induced if-then rules into ILP logic clauses.

We use the example provided in [44] in order to explain this technique in more detail. The original data in Table 3.1 consists of one target relation (daughter) and three background relations (parent, female, male). The target relation is daughter(X,Y) indicating that the person X is the daughter of person Y and the goal is to learn the target class using background relations. The variables (ann, eve, pat, sue, tom) are all of type person.

Table 3.1: A daughter family relationship problem in Prolog form [44]

Training Examples		Background Knowledge		
daughter(sue,eve).	pos	parent(eve,sue).	female(ann).	male(pat).
daughter(ann,pat).	pos	parent(ann,tom).	female(sue).	male(tom).
daughter(tom,ann).	neg	parent(pat,ann).	female(eve).	
daughter(eve,ann).	neg	parent(tom,sue).		

The following clauses are defined:

daughter(X,Y) :- female (X).
daughter (X,Y) :- female (Y).
daughter (X,Y) :- male(X).
daughter (X,Y) :- male(Y).
daughter (X,Y) :- parent(X,X).
daughter (X,Y) :- parent(X,Y).
daughter (X,Y):- parent(Y,X).
daughter (X,Y):- parent(Y,Y).

When the background knowledge predicates are applied in the form of the above clauses, the following flat file is obtained where f, m, and p represent female, male, and parent, respectively.

Table 3.2: Propositional form of the daughter relationship problem (1 for true, 0 for false) [44]

Variables		**Propositional Features**								
X	**Y**	**f(X)**	**f(Y)**	**m(X)**	**m(Y)**	**p(X,X)**	**p(X,Y)**	**p(Y,X)**	**P(Y,Y)**	**class**
sue	eve	1	1	0	0	0	0	1	0	pos
ann	pat	1	0	0	1	0	0	1	0	pos
tom	ann	0	1	1	0	0	0	1	0	neg
eve	ann	1	1	0	0	0	0	0	0	neg

Having the above propositional representation, an attribute value learner can induce the following rule [44]:

If [female(X) = 1] and [parent(Y, X) = 1] then class = pos

Thus, this rule could be transformed into the following clause as an output of LINUS:

daughter (X,Y) :- female(X), parent(Y, X).

3.5.2 Database-Oriented Methods

Database-oriented algorithms constitute the second major group of propositionalization methods. Their main characteristic is that they make use of aggregation functions (explained in details later) to define the propositional attributes. These approaches are more efficient when used on larger data sets. One of the main and well-known database-oriented propositionalization techniques is described below.

The RelAggs Algorithm

The RelAggs algorithm was introduced by Krogel and Wrobel [45]. This algorithm specifically addresses the following challenge that exists in the databases which store business related information. These databases are often structurally very simple, but have large sizes. Therefore, aggregate operators are employed and new features are constructed in order to summarize and finally store the information in the background relations.

We will discuss aggregation in the context of an example provided in [33]. Consider the Loan and the Account relations (Figure 3.2). We first join these two tables together. The result of this join is shown in Table 3.3 and indicates that each of the tuples in the target relation (Loan) is associated with zero to many tuples in the background relation (Account). Let us consider loan_ID = 2 which is associated with account_ID = 132. In the Account table, there are two attributes corresponding to each account_ID, namely, frequency (nominal attribute), and balance (numerical attribute). The goal is to create a flat file and to summarize the information in the Account table. To do so, aggregation functions *count, min, max, sum,* and *avg* are employed. This results in constructing new features in the final flat file as it is shown in Table 3.4.

Table 3.3: Flattening through joining of Loan and Account tables [33]

loan_id	account_id	amount	duration	payment	status	New_frequency	New_balance
1	132	2000	12	100	Good	monthly	3521
						weekly	2000
2	132	4000	12	200	Good	monthly	3521
						weekly	2000
3	148	1000	24	80	Bad	NULL	NULL
4	148	30000	24	1000	Bad	NULL	NULL
5	98	10000	36	800	Good	weekly	10000

For each of the unique values in the frequency field, using *count*, a new feature is constructed (e.g. *count* (monthly) and *count* (weekly)) and the value of these attributes for each of the tuples in the Loan table is recorded. If there is no corresponding record in the Account table, a NULL value is entered (e.g. in loan_ID = {3,4} case). The same procedure is performed for the balance attribute and since it is numerical, other aggregation functions such as *min, max, sum,* and *avg* could be used as well.

Table 3.4: The result of propositionalization using RelAggs [33]

loan_ID	...	status	Count (monthly)	Count (weekly)	Min (balance)	Avg (balance)	Sum (balance)	Max (balance)
1		Good	1	1	2000	2760.5	5521	3521
2		Good	1	1	2000	2760.5	5521	3521
3		Bad	NULL	NULL	NULL	NULL	NULL	NULL
4		Bad	NULL	NULL	NULL	NULL	NULL	NULL
5		Good	0	1	10000	10000	10000	10000

3.6 Upgrade versus Propositionalization

The following example [34] presents a comparison between a relational classification rule and a propositional classification rule. In this example, the propositional rule predicts if the person is a spendthrift based on the fact that, he/she has a high income. The relational rule on the other hand, predicts if a person spendthrift if he/she has a high income and if he/she is also married to a person having a high income. Relational patterns are usually expressed using subsets of first-order logic a.k.a. relational logic. The predicate logic in the relational rule (Figure 3.4) includes (MarriedTo) and variables (C1, C2) which are not represented in the propositional rule.

Customer Table

ID	Gender	Age	Income	TotalSpent	Spendthrift
C1	Male	30	214000	18800	Yes
C2	Female	19	139000	15100	Yes
C3	Male	55	50000	12400	No
C4	Female	48	26000	8600	No
C5	Male	63	191000	28100	Yes
C6	Male	63	114000	20400	Yes
C7	Male	58	38000	11800	No
C8	Male	22	39000	5700	No

MarriedTo Table

Spouse 1	Spouse 2
C1	C2
C2	C1
C3	C4
C4	C3
C5	C12
C6	C14

Propositional rule

If Income > 108000 THEN Spends_a_lot = Yes

Relational rule

Spendthrift (C1, Age1, Income1, TotalSpent1) ← married_to(C1,C2) and

customer(C2,Age2,Income2, TotalSpent2,BS2) ^ Income2 >= 108000

Figure 3.4: A relational database with two tables (Customer, and MarriedTo) and two classification rules (propositional and relational) [34]

An important feature of the propositionalization techniques is their ability to transform multiple relations into a universal, flat file which could then be used as an input into a wide range of existing propositional learners [34]. In order to obtain this universal flat file, however, an extensive pre-processing effort is needed. Summarizing all of the data which is distributed over multiple relations into one relation usually results in creating large number of attributes, introducing NULL values, and information loss due to aggregation.

Since the upgrade strategies directly search for features across multiple relations, they do not require extensive pre-processing efforts and therefore the drawbacks associated with transforming relations into a flat file could be eliminated. One of the main features of the upgrade methods is their expressive power [44]. These techniques have the ability to use first-order hypothesis space, and can describe more complex concepts using first-order logic clauses. The propositionalization methods are less powerful, in principle. However, in practice, it is shown that in many cases searching a fixed subspace which could be defined by feature transformation, is sufficient [42].

The propositionalization techniques show better scalability and enhanced efficiency in terms of computational cost when dealing with large applications and complex data sets [46]. The reduced scalability of the upgrade methods is due to the fact that, these techniques consume considerable amount of time on building hypothesis. This makes them more suitable for small datasets [47]. The propositionalization techniques outperform the upgrade methods in terms of handling regression, numerical values, and distance measures [41]. In dealing with numerical values, the ILP-based (upgrade) approaches are limited and only few ILP-based systems have the ability to handle numerical values provided that they are discretized [44]. In terms of handling noisy data, aggregation-based propositionalization techniques result in better predictive performance compared with the upgrade techniques [44]. Table 3.5 summarizes these pros and cons of the classes of strategies.

Table 3.5: Comparison of the upgrade and propositionalization strategies

Propositionalization	Upgrade
Advantage	**Advantage**
• Enhanced efficiency for large applications in terms of computational cost • Ability to use all traditional single-table propositional learning systems and therefore, flexibility of choosing different algorithms • Superior in handling regression, numeric values, and distance measure	• Good expressive power compared with the propositionalization techniques • The ability of directly searching the full first-order hypothesis space • Require less pre-processing efforts compared with the propositionalization techniques
Disadvantage	**Disadvantage**
• Require extensive pre-processing effort • Large number of missing values and exponential increase in the number of attributes • Limited expressive power compared with the upgrade techniques	• Reduced scalability when dealing with complex dataset • Inability to use existing single-file mining methods directly

The drawbacks associated the propositionalization strategy (such as extensive pre-processing efforts, redundancy, NULL values, and exponential increase in the number of attributes) and the upgrade strategy (such as their inability to directly re-use existing propositional algorithms) were the motivation behind the work introduced in [33] which is based on the multi-view learning concept. By constructing different flat files (views) instead of one universal flat file, less pre-processing effort is required. Furthermore, the same conventional propositional methods could be re-used without any need to upgrade them.

3.7 Multiple View Multi-relational Classification

In contrast to the first family of algorithms which is mainly about upgrading traditional learning algorithms to handle relational data or the second category of techniques which flattens multiple relations to a single universal flat file, the Multi-Relational Classification

(MRC) [33] algorithm (which is inspired by multi-view learning) learns directly from a relational database. The multi-view learning concept and the MRC algorithm are discussed next.

3.7.1 Multi-view learning

Multi-view learning has been the focus of different studies in recent years [48; 49; 50]. It is considered a new learning strategy which learns from different independent views (feature sets) of the data. It is usually applied when features can be divided into multiple subsets where any of those subsets is enough in order to approximate the hypothesis function [48].The work in [51] summarizes the multi-view paradigm as follows. In addition to the target attribute Y, the input attributes are partitioned into two different views say X_1 and X_2. The assumption is that, each of the views alone is enough to predict the target Y with high accuracy.

3.7.2 Multi Relational Classification (MRC)

The MRC technique enables learning from multiple views of a given relational database. The work in [33] states that, this method offers both efficiency gains and predictive performance over the existing models in mining various relational databases. Figure 3.5 illustrates the MRC algorithm framework. It consists of five stages, namely, *Information Propagation*, *Information Aggregation*, *View Construction*, *View Combinations*, and *View Validation*.

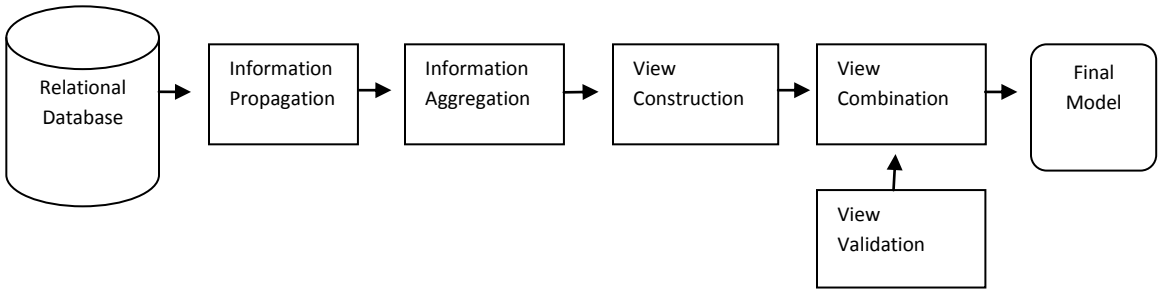


Figure 3.5: The MRC framework [33]

The *Information Propagation* stage propagates necessary information from the target relation to the background relations, using the foreign key links (as defined before in Section

3.2). Therefore, the resulting relations contain different information gathered from different tables in the database. This stage is followed by the *Information Aggregation* stage where, aggregation functions summarize the information which is embedded in multiple tuples into one row (record). Applying aggregation functions at this stage results in the newly constructed background relations. The third stage is *View Construction* where, existing single-table data mining methods are employed to learn the target concept from each view. In the next stage, i.e. *View Validation*, all constructed view learners get evaluated. First, we verify if the views are sufficiently able to predict the target concept. Second, we find strongly uncorrelated view learners. In the last stage, the multiple view learners (resulting from the *View Validation* stage) are incorporated into a meta-learner in order to build a final classification model. A meta-learner [52] learns from knowledge generated by base classifiers. In the MRC algorithm, the meta-learner is used to produce a function that controls the possible collaboration of the learners in order to obtain the best classification accuracy. The final model consists of this function in addition to the hypothesis that is constructed by each of the view learners [33].

Both the propositionalization database-oriented techniques and the MRC technique use aggregation in order to construct the universal flat file or multiple flat files respectively. In our work, we employ the first three stages of the MRC method in order to construct the views, as it is discussed in details in Chapter 4.

In the next section, we study the concept of aggregation and aggregate functions and operators in more details.

3.8 Aggregation

Aggregate functions (a.k.a. aggregation operators) are widely applied in databases. Aggregation is considered a significant feature of database query languages. In some specific application domains such as On Line Analytical Processing (OLAP), decision support, management of geographical data, and statistical evaluation the ability of applying aggregation becomes fundamental [53]. In fact, the functionality of OLAP is based on aggregation functions [54]. In statistics, aggregate functions are commonly used to describe population properties such as standard deviations and averages. Aggregation is considered to be an indispensible tool not only in mathematics or in physics, but in a majority of

economical, engineering, social and other areas of science. For example, one study [55] shows the main role of mathematical aggregation in reducing the huge amount of information into meaningful subset and their application for video. This study introduces a new method of browsing a video using words and looking for what users want in specific. It allows users to search within the content of a video and run queries about the actor, the character, what the actors are talking about, the camera techniques, amongst others. Despite these facts, there has been a slow progress in a systematic study of aggregation in the context of query languages [53].

With respect to database mining, aggregation is the key element in database-oriented propositionalization techniques and the flattening process. In relational databases, data are scattered over many tables and one has to deal with the non-determinacy of one-to-many relationships. Therefore, aggregation functions are considered powerful tools to handle this issue [56]. In this section (3.8), we will first provide a general definition of aggregate functions and list different proposed criteria to classify the aggregation operators. We then study the effective role of aggregation functions in summarizing the information in relational databases. This is followed by considering the role of aggregation functions when preserving the privacy of databases through microaggregation and disclosure risk measures. Finally, we discuss some of the existing challenges associated with aggregation functions.

3.8.1 General Definition

The work presented [53] defines an aggregate function over N as a total function from $\{\{N\}\}$ to N which maps each multiset of values to a value. In this definition, N represents an accountably infinite domain. The work in [44] adds two more assumptions. In the first assumption, N may also be a finite set and in the second assumption, the function values could come from a set of values different from N. For instance, consider counting a certain value of a nominal attribute.

3.8.2 Categorization of Aggregation Operators

Two families of properties have been defined for aggregation operators, i.e. mathematical properties and behavioural properties [57]. Mathematical properties include boundary

conditions, monotonicity (none decreasing), continuity, associativity, symmetry, bisymmetry, absorbent element, neutral element, indempotence, compensation, counterbalance, reinforcement, stability for a linear function, and invariance. Behavioural properties include decisional behaviour, interpretability of the parameters, and weights of the arguments. The same work provides a detailed description of each of these properties.

In this thesis, we have used the *count, min, max, sum, avg,* and *stdv* aggregation operators which are part of the standard SQL in order to perform aggregation. Different criteria have been considered to categorize the aggregation operators. In this section, we review some of these categorization criteria and see how our operators (i.e. *count, min, max, sum, avg,* and *stdv*) fall into those categories. The work presented in [41] defines general measures for an aggregation function based on the number of tables involved. It identifies three measures (variable-depth) for data models to characterize aggregates. An aggregate of variable-depth 0 involves only one table. Aggregates such as *count, min, sum, etc* have a variable-depth of 1 and variable-depth of greater than one represents multi-relational patterns such as benzene-rings in molecules, etc. The ranges of possibilities are listed below.

d_v(A) = 0:

- Propositions (adult == (age>18))
- Arithmetic functions (area == width.length)

d_v(A) = 1:

- Count, count with condition
- Count distinct
- Min, max, sum, avg
- Exists, exists with condition
- Select record (eldest son, first contract)

d_v(A) > 1:

- Exists substructure
- Count substructure
- Conjunction of aggregates (maximum count of children)

The complexity of aggregation functions is a main factor in determining the complexity of a relational concept. The work in [58] identifies three levels of aggregation complexity, namely, simple aggregation, multi-dimensional aggregation, and multi-type aggregation.

A simple aggregation refers to a mapping from a bag of zero or more atomic values to a numerical or categorical value. An example of simple aggregation operations used for numerical values are *max*, *min*, *count*, *mean* and an example for categorical values are the *mode* (the most common value) and the *count* of most common value. A multi-dimensional aggregation is a mapping with a bag of zero or more objects as input, of which *n* attributes in the form of feature vector (x_1, x_2, x_3, ..., x_n). It captures any relationship which exists between two or more attributes.

A multi-type aggregation refers to a mapping with two or more bags of objects of different types as input, with a possibility of having feature vectors of different lengths. An example of using such aggregation is finding the total value of the products which a given customer has returned. This type of aggregation involves two bags: the products which were bought by a customer, and their prices, in addition to the products which were returned.

With regard to aggregation complexity, the work presented in [58] empirically demonstrates that, on a noisy business related data, as a result of using more-complex aggregation methods, the generalization performance of relational learners could be increased.

Lenz and Thalheim [54] classify the aggregation functions according to their computational density and define two classes of aggregation functions as follows:

First Class: The simplest class of aggregation functions which uses **one-pass** aggregation. Functions which belong to this class include *count*, *avg*, *sum*, *min*, and *max*. This class is applied on linear transformed values. This study indicates that if these simple aggregations are applied to non-linear transformed values they could be misused. For instance, getting the average of pH values might refer to arithmetic average, metric average, or quadratic average.

Second Class: Constitutes more complex aggregation functions which are employed in moving or cumulative statistics that aims at relating data values to the whole data. It is referred to as **two-pass** aggregation. The first pass creates properties of a group as whole; and second pass generates the results corresponding to each unit. These functions, however, usually provide a weak and non-robust descriptive statistics because a minor data change implies a major effect on aggregation. The study in [54] further classifies two-pass aggregation functions into different categories such as *moving totals* which track changes

over time, *running differences* which track difference over time, *cumulative percentage* which refers to what percentage of the whole data set the current subset of data includes, *ranking* of the elements in a given set based on one of the values, *cross tabulation* which is similar to two-dimensional spreadsheet generation, *mode* which refers to the most frequently occurring value, *standard deviation*, *variance,* and *average deviation* which shows drift apart from the average.

In another study by Gray et al. [59], aggregation functions are classified into three categories namely Distributive, Algebraic, and Holistic.

- *Distributive*: Aggregation function F( ) is considered distributive if there is a function G( ) so that $\mathbf{F}(\{\boldsymbol{X}_{i,j}\}) = \mathbf{G}(\{\mathbf{F}(\{\boldsymbol{X}_{i,j} \mid i = 1,..., \mathbf{I}\}) \mid j = 1,...,\mathbf{J}\})$. Example is *count*, *min*, *max,* and *sum*. If the order is imposed, cumulative aggregation functions will also be considered distributive.
- *Algebraic*: Aggregate functions F( ) is considered algebraic if there are M-tuple valued function G( ) and a function H( ) so that $\mathbf{F}(\{\boldsymbol{X}_{i,j}\}) = \mathbf{G}(\{\mathbf{F}(\{\boldsymbol{X}_{i,j} \mid i = 1,..., \mathbf{I}\}) \mid j = 1,...,\mathbf{J}\})$. Example of algebraic functions are *avg*, *stdv*, *maxN*, *minN*, and *center_of_mass*.
- *Holistic*: Aggregation function F( ) is said to be holistic if there is no bound on the size of storage which is required to describe a sub-aggregate. Examples of such aggregation functions are *median*, *mode* and *rank*.

Based on the above criteria, the aggregation functions used in our work (*min, max, count, stdv, avg*, and *sum*) could be categorized as follows. These functions have a variable-depth of one since they involve more than one table. They are considered simple aggregation functions which map zero or more atomic values into a numerical or a categorical value. In terms of computational density, *count, avg, sum, min,* and *max* functions belong to one-pass aggregation class and are applied on linear transformed values. However, *stdv* function belongs to two-pass aggregation class. The operators *count, min, max,* and *sum* are distributive aggregation functions whereas; *avg* and *stdv* are considered algebraic functions.

In the next section the main goal of aggregation operators i.e. summarization is discussed in details.

3.8.3 Aggregation and Summarization

Given a set of records, it is possible to compute a feature (attribute) of that set which summarizes the set. Summarization of information in a set is considered the most important feature of aggregation which is used during the propositionalization process. In fact, the propositionalization process is a result of a series of steps where information in one table is projected onto records belonging to another table. We use the example provided in [41] to describe the relation between aggregation and summarization.

Consider two tables *P* and *Q* which are joined by association *A*. When we summarize over *A*, information about the structural properties of *A* and the data within *Q* could be added to *P*. The multiplicity of association *A* has an impact on the search space of multi-relational patterns that involves *A* [60; 41]. Moreover, the multiplicity of *A* affects the choice of aggregates. For example, when we summarize *P* over *A*, only the multiplicity on the side of *P* is relevant. The following categories describe four types of association which exist between two tables [41]:

1 Indicates that for every record in *P* there is only one record in *Q*. Therefore, no aggregation is required and a simple join will add all of the non-key attributes of *Q* to *P*.

0...1 A record in *P* may not have any corresponding record in *Q*. Therefore, an outer join is needed which substitutes NULL values for missing records.

1...n There is at least one record in *Q* for every record in *P*. Aggregation is required in order to capture the information in the group of records that belong to a single record in *P*.

0...n this is similar to the 1...n case; however, due to empty groups the value of certain aggregates may be undefined. Special attention is needed to deal with the resulting NULL values.

Other than the structural information of *A*, aggregation functions usually involve one or more attributes of *Q*. For every association, we can define an infinite number of aggregate functions [56].

In the next section, we will discuss some of the recent applications of aggregation functions in the privacy preserving of relational databases. This includes studying the role of

aggregation operators in two major privacy protection methods such as microaggregation and disclosure risk measurement.

3.8.4 Aggregation and Privacy

The application of aggregation operators on data privacy has been studied in a recent survey [3]. This work, defines the role of aggregation operators in the context of two main privacy protection techniques, i.e. microaggregation and disclosure risk measures. In both cases, aggregation is considered a useful tool in preserving data privacy.

3.8.4.1 Microaggregation

Microaggregation is a statistical disclosure control technique used for data protection at microdata level [61]. To protect sensitive information in microaggregation, raw microdata (i.e. individual records) are grouped into small aggregates (clusters) prior to publication. Each of the aggregates contains at least *k* data vectors in order to preserve the private information. The *k* parameter is a constant value determined by the data protector. When determining the *k* value, the trade off between loss of information and privacy is taken into consideration.

The process of creating clusters involves two steps, namely, partitioning and aggregation. In the partitioning phase, records are partitioned into clusters with at least *k* records and at most *2k* records. In the aggregation phase, aggregation operators are used and a representative or centroid for each of the clusters is computed.

We use the following example to describe the process in more details. Figure 3.6(a) shows microdata with four attributes i.e. *Name*, *Age*, *Wage*, and *Hours*. *Name* is the identifier attribute that should be removed and *Age*, *Wage*, and *Hours* are the Quasi-Identifiers that need to be anonymized. In the partitioning phase, the microdata records are partitioned into clusters. The value of *k* is set to be three, and therefore, each cluster consists of three records. The resulting clusters include {A,D,I}, {B,E,H}, and {C,F,G}.

In the aggregation phase, the *avg* aggregation function is used to calculate the centroid of the attributes in each cluster. Consider {A,D,I} corresponding to records Lenny, Sarge, and Shark in the original dataset. The average of the *Age*, *Wage*, and *Hours* attributes of these

records is calculated and then substituted in the resulting dataset. For example, the average of the *Age* attribute for these three records is (40+49+53)/3 = 47.333. Similarly, the average of the *Wage* attribute is (59+ 62+ 60)/3 = 60.333 and so on. The same procedure is repeated for the other clusters and the microaggregated dataset shown in Figure 3.6(b) is obtained.

(a)

Name	Age	Wage	Hours
Lenny	40	59	40
Woody	29	35	40
Etch	32	30	20
Sarge	49	62	30
Rex	42	65	35
Hamm	19	20	10
Wheezy	35	30	30
Mike	22	40	40
Shark	53	60	40

(b)

Name	Age	Wage	Hours
A	47.333	60.333	36.667
B	31	46.667	38.333
C	28.667	26.667	20
D	47.333	60.333	36.667
E	31	46.667	38.333
F	28.667	26.667	20
G	28.667	26.667	20
H	31	46.667	38.333
I	47.333	60.333	36.667

Figure 3.6: Original data (a) and microaggregated data with k=3 (b) **[3]**

The work presented in [3] summarizes the aggregators that are used in microaggregation applications according to the type of data they are applied on. These include numerical data, categorical data, heterogeneous data, and sequences.

Most of the microaggregation applications are developed for continuous numerical attributes and the arithmetic *mean* is the most frequently used aggregator for this type of data [62]. Microaggregation for categorical data was studied in [63]. Usually, for ordinal data, *median* or *convex median* is used where, for nominal data, *mode* is used [3]. Heterogeneous data includes several attributes of different types and, depending on the data type, different aggregators are applied. These aggregators are then combined in order to produce the final result [3]. For example, the aggregation of logs from a Web server is achieved using combined aggregation operators [64]. Data sequences include numeric time series and categorical spatial series. The approach for aggregating numeric time series and categorical series is described in [65] and [66], respectively.

3.8.4.2 Disclosure Risk Measures

Record linkage is one of the approaches for disclosure risk assessment [3]. Based on this approach, in order to measure the risk, the protected data file is linked with the original data file. Then, the number of correct links indicates a measure of the risk. If all of the records are identified correctly, then the risk is maximal. This indicates that the attacker is able to obtain sensitive information of all records. On the other hand, if no correct link is identified, the risk is minimal and the protection is maximal. Practically, a portion of original records could be re-identified. Determining this portion is considered a measure of the risk.

Two approaches for re-identification, namely, probabilistic record linkage and distance-based record linkage have been identified [3]. In the distance-based record linkage, using different distances such as Euclidean, Manhattan, Mahalanobis, and so on, the nearest record in the protected dataset is calculated. First, the weight of the attributes is determined. When the weights are selected appropriately, a better performance of the re-identification algorithm is expected and this lead to a better-estimated risk measure. For some distances such as Mahalanobis, the weight is estimated automatically. However, for some other distances such as Euclidean, it is not clear how to estimate the weight. To address this issue, aggregation operators may be used. The problem is formalized as follows.

Consider two sets of records i.e. A = (a_1, ..., a_N) and B = (b_1, ..., b_N) where a_i is the protected record corresponding to b_i. V_k (a_i) represents the value of the k^{th} variable of the i^{th} record. As a result, the sets of values $d(V_k(a_i), V_k(b_j))$ for all pairs of records a_i and b_j are obtained where d is the distance. Using aggregation operator C, the optimal performance of record linkage is obtained if for all k, the aggregation of the values $d(V_k(a_i), V_k(b_i))$ is larger than the aggregation of the values $d(V_k(a_i), V_k(b_j))$ for all $i \neq j$ [3].

$$C\left(d(V_1(a_i), V_1(b_i)), \ldots, d(V_n(a_i), V_n(b_i))\right) \geq C\left(d(V_1(a_i), V_1(b_j)), \ldots, d(V_n(a_i), V_n(b_j))\right)$$

for all $i \neq j$.

(3.1)

Let us consider the following example. Given two data files where each file has three records, we first obtain $V_1(b_1), V_1(b_2), \text{and } V_1(b_3)$ corresponding to the first file and

$V_2(b_1), V_2(b_2)$, and $V_2(b_3)$ corresponding to the second file. Their corresponding protected records are also determined as $[V_1(a_1), V_1(a_2), V_1(a_3)]$ and $[V_2(a_1), V_2(a_2), V_2(a_3)]$ respectively. Then, the distances between each of the records in the original dataset and all of the records in the protected dataset are computed. The results are shown in Table 3.6.

Table 3.6: data to be considered in the learning process **[3]**

d(V_1)	**d(V_2)**	**outcome**
$d(V_1(a_1), V_1(b_1))$	$d(V_2(a_1), V_2(b_1))$	link
$d(V_1(a_1), V_1(b_2))$	$d(V_2(a_1), V_2(b_2))$	no-link
$d(V_1(a_1), V_1(b_3))$	$d(V_2(a_1), V_2(b_3))$	no-link
$d(V_1(a_2), V_1(b_1))$	$d(V_2(a_2), V_2(b_1))$	no-link
$d(V_1(a_2), V_1(b_2))$	$d(V_2(a_2), V_2(b_2))$	link
$d(V_1(a_2), V_1(b_3))$	$d(V_2(a_2), V_2(b_3))$	no-link
$d(V_1(a_3), V_1(b_1))$	$d(V_2(a_3), V_2(b_1))$	no-link
$d(V_1(a_3), V_1(b_2))$	$d(V_2(a_3), V_2(b_2))$	no-link
$d(V_1(a_3), V_1(b_3))$	$d(V_2(a_3), V_2(b_3))$	link

This table represents the available information in the learning process of this example. From the information in this table, using Equation 3.1, we construct a model from which the weighting vector [3] is determined.

The work presented in [3] highlights the positive role of using aggregation operators in two privacy protection techniques, i.e. microaggregation and disclosure risk measures. In the next section, some of the challenges associated with aggregation functions are discussed.

3.8.5 Aggregation Functions Challenges

Aggregation functions are applied on a set of entries (tuples) in order to create an entry that implicitly represents the properties of that set. To this end, new attributes are introduced to store the summarized information. Therefore, aggregation is considered a form of feature construction [33]. In statistical relational learning, aggregation is regarded as a pre-processing step [67].

Recall that, depending on the type of attributes in the original dataset, different aggregation operators are employed. For nominal and binary attributes, aggregation function

count is applied. For numerical attributes, aggregation functions *max, min, stdv, sum, avg,* and *count* are used. Applying aggregation functions usually increases the number of attributes in the resulting dataset. When *count* is applied on either nominal and binary attributes, the type of the constructed features is changed into numerical. This issue is discussed in details in the following chapters.

Using aggregation functions in relational databases could lead to unexpected results [54]. The studies presented in [34; 67] highlight two drawbacks associated with aggregation. The first issue is described as follows. Aggregation almost always involves loss of information. When aggregation is applied, the information is summarized and therefore, some of the details are ignored. However, this usually leads to smaller datasets, minimized redundancy, and reduced computational complexity. The second issue is that, aggregation is adequate in some cases, and is inadequate in other cases. We investigate these two drawbacks in the context of the following example. The domain consists of two tables in a multi-relational database shown in Figure 3.7. "Customer" is the target table. "Transaction" is another background table which contains additional information about entities in the target table. The two tables are linked via the customer identifier (CID).

In order to show the loss of information in this example, we consider attributes TYPE and PRICE. They contain nominal and numerical values, respectively. After applying the appropriate aggregation functions, new features are constructed and listed in the Result table (Figure 3.7 c). When aggregation functions are applied, the information in multiple records of the Transaction table is summarized into a single record in the Result table. For example, there are three records associated with C2 in the Transaction table. These three records are summarized into a single record in the Result table. From this table, we are able to show that customer C2 has purchased three books; one fiction and two nonfictions. We also obtain information about the total number of books, the most expensive and the least expensive one. However, this summarization is at the expense of losing information about individual books. For instance, it is not possible to obtain the price of the fiction book customer C2 has purchased. Similarly, if the Transaction table consisted of another field, e.g. *Date* indicating the purchase date of a given book, with aggregation, it would not be possible to determine which date the books were purchased.

(a) Customer Table

CID	CLASS
C1	0
C2	1
C3	1
C4	0

(b) Transaction Table

CID	TYPE	ISBN	PRICE
C1	Fiction	523	9.49
C2	Non-Fiction	231	12.99
C2	Non-Fiction	523	9.49
C2	Fiction	856	4.99
C3	Non-Fiction	231	12.99
C4	Fiction	673	7.99
C4	Fiction	475	10.49
C4	Fiction	856	4.99
C4	Non-Fiction	937	8.99

(c) Result Table

CID	Fiction Count	NonFiction Count	Price Sum	Price Min	Price Max	Price Std	Price Avg	Total Count
C1	1	0	9.49	9.49	9.49	0	9.49	1
C2	1	2	27.47	4.99	12.99	3.27	9.16	3
C3	0	1	12.99	12.99	12.99	0	12.99	1
C4	3	1	32.46	4.99	10.49	2.01	8.12	4

Figure 3.7: Illustration of Customer, Transaction, and Result Table [67]

This example also shows that aggregation is suitable for some attributes such as TYPE, and PRICE while is it not ideal for others. For example, the ISBN attribute is a unique identification of a given book and it is meaningless to apply aggregation on it. Therefore, these types of attributes may be removed from the result set.

Aggregation functions present many advantages: they eliminate redundancy, hide unnecessary details, and reduce the size (the number of records) of the resulting dataset. However, these functions result in an exponential increase in the number of attributes specially when applied on nominal attributes with high cardinality, in which we choose to apply a number of aggregation functions. Aggregation functions also result in a loss of information due to summarization, and may not be suitable for all attributes in a given relation. Table 3.7 summarizes some of the advantages and disadvantages of using aggregation operators.

Table 3.7: Comparison of pros and cons of using aggregation operators.

Pros	Cons
• Eliminate redundancies resulted from joining multiple relations • Achieve abstraction and hide unnecessary details • Reduce the size of the resulting flat file • Key factor in the flattening process	• Exponential increase in the number of attributes specially with nominal attributes • Loss of information • Not being suitable for all attributes • May result in misleading correlation

In summary, aggregation functions are widely used in the propositionalization strategy in order to flatten a relational database. However, studying the impact of aggregation on the privacy of relational databases has been mostly overlooked. This issue is addressed in the following chapters. Aggregation is applied in order to construct flat files suitable for classification algorithms. Investigating the impact of aggregation on the classification accuracy and the time required to the classification models is another research area which has not been addressed extensively. This constitutes the second part of our research which is addressed in the remaining chapters.

3.9 Summary

In this chapter, classification was introduced. Then, two major strategies in classifying multi-relational databases, namely, upgrade and propositionalization were addressed. In addition to these methods, another relatively new strategy in mining relational databases, i.e. multi-view learning, was discussed. Multi-view learning aims at addressing the major shortcoming of the upgrade and the propositionalization methods, for classification. Instead of dealing with the whole relational database, it learns from different independent views (feature sets) of the data. With multi-view learning, on one hand, it is possible to use the existing propositional mining methods with less pre-processing efforts. On the other hand, we are able to eliminate the drawbacks associated with creating a universal flat file.

In the second part of this chapter, the concept of aggregation, and its role in data summarization was discussed. Different categories of aggregation operators were listed, and the role of aggregation operators in data privacy in the context of microaggregation and

disclosure risk measures was addressed. Finally, some of the challenges associated with aggregation were highlighted.

In the next chapter, the experimental design of our work is presented.

Chapter 4

Experimental Design

Aggregation functions play a key role during the flattening process of a relational database, as was pointed out in Chapter 3. These functions are used in either the propositionalization strategy or the multi-view learning approach. Our goal is to define a methodology to effectively study the impact when applying the well-known aggregation functions such as *max, min, sum, avg, stdv,* and *count* on the privacy of a relational database. To achieve this goal, we introduce the PBIRD (Privacy Breach Investigation in Relational Databases) algorithm which investigates the impact of aggregation on data leakage in a relational database. We also define a methodology to perform an aggregation-based comparative study. The main objective of this study is to examine the effect of applying aggregation functions on the classification accuracy and the model building time of different classification algorithms.

This chapter is organized as follows. First, the outline of the PBIRD algorithm and some of the background techniques are discussed. This is followed by a detailed description of the PBIRD algorithm. Then, we explain our strategy in order to investigate the effect of aggregation functions on the classification accuracy and model building time in a relational database. Finally, we discuss the environmental setup and describe the datasets which are used in this experimental study.

4.1 The PBIRD Algorithm

Identifying privacy leakage in relational databases has been recently investigated [2]. In that work, it was shown that harmless attributes could be linked to the private information and lead to data leakage when building a model. Our work is inspired by this study. We first investigate potential privacy leakage using feature selection. Next, we consider the potential impact of aggregation on such privacy leakage and show how aggregation affects the list of harmless attributes which could be linked to the private information.

In doing so, we introduce the PBIRD (Privacy Breach Investigation in Relational Databases) algorithm. The motivation behind this algorithm is as follows. Recall that aggregation functions are usually used to flatten a relational database, either in database oriented propositionalization or multi-view learning. This causes an increase in the number of attributes, a decrease in the number of rows, and a potential change of correlation between the set of attributes in the resulting flat file and the target class. The change of correlation in between attributes themselves and between the attributes and the target class could have a direct impact on the privacy of a database. In this investigation, we are interested in finding the set of attributes which are strongly correlated with the target class and may lead to privacy breach in the relational database. We consider how applying aggregation functions could affect such a list of attributes.

Before discussing the details of the PBIRD algorithm, it is useful to describe some of the background techniques which are employed.

4.1.1 Background Techniques

In our work, we use the multi-view learning strategy to achieve multi-view relational classification, as introduced in [4]. Therefore, instead of constructing a single universal flat file, we construct multiple views that are essentially smaller flat files and reflect the information in a subset of relations in a multi-relational database. In order to achieve attribute selection, we employ the CFS (Correlation based Feature Selection) algorithm [5].

4.1.1.1 Correlation based Feature Selection (CFS)

In general, feature subset selection refers to the process of identifying and removing redundant and irrelevant information in order to reduce the dimensionality of the data. The benefit of feature selection includes improved predictive accuracy, reduction in the amount of data which is required to perform learning, a more compact and easily understood learned knowledge, and less execution time. In our research, we employed the CFS algorithm [5]. In general, CFS is considered one of the fastest feature selection algorithms that chooses fewer features and produces smaller trees [68] that other feature selection methods. CFS is also one of the feature selection techniques that evaluate subsets of attributes, rather than individual attributes [68]. In our study, we use the multi-view learning approach in order to perform classification. In the multi-view learning scheme, each view consists of a subset of attributes and hence, CFS was considered a perfect candidate for our purpose. For instance, we were able to study the impact of aggregation on the selected feature subsets in each view.

Correlation-based Feature Selection uses a subset evaluation heuristic which takes into account the usefulness of individual features in predicting the class. Later (in Chapter 5), we show how this measure is used to study the impact of aggregation in changing the correlation of features with the classification target. CFS is based on a hypothesis stating that, a good feature subset contains features that are highly correlated with the target class and uncorrelated to one another. The concept of correlation [69] is used to measure the dependency between attributes. Correlation (first studied by Pearson [1857-1936]) refers to the departure of two variables from independence [69]. If we say that two variables *a* and *b* are correlated, this indicates the fact that, knowing *a* helps us to predict *b*. However, if knowing *a* does not tell us anything about *b*, we say that the two variables are uncorrelated. However, correlation does not imply causality [6]. That is, if *a* and *b* are correlated, it does not imply that *a* causes *b* or that *b* causes *a*. Using this definition of correlation, the CFS algorithm is concerned with finding a set of selected features (attributes) which are highly correlated with (predictive of) the class and yet not correlated to (predictive of) each others [5]. In the remaining parts of this thesis, we will use the term “correlation” in this context.

The heuristic goodness of feature set or M_s is obtained using the following formula:

$$M_s = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}} \qquad (4.1)$$

M_s refers to the heuristic "merit" of a selected feature subset s (selected attributes), K is the number of features in the selected subset, $\overline{r_{cf}}$ is the average of the correlation between the features and the class, and $\overline{r_{ff}}$ is the average of the inter-correlation in between features within the feature subset. The numerator of Equation 4.1 provides us with information about the predictiveness of a set of features with regard to the class and the denominator indicates the amount of redundancy which exists among the features.

In our experiment we use WEKA (Waikato Environment for Knowledge Analysis) [70] which is a well known suite of machine learning software, hosted at the University of Waikato, New Zeeland. In the CfsSubsetEval (the implementation of CFS in WEKA) with BestFirst search method, the value of M_s is referred to as "merit of best subset found". The higher the value of "merit of best subset found" the higher is the correlation in between the selected features and the class and the lower is the correlation in between the selected features themselves.

In discussing Equation 4.1, Hall [5] states that "as the number of components increase (assuming the additional components are the same as the original components in terms of their average inter-correlation with other components and with the class) the correlation between the composite and the class increases". We will refer to this statement later when we analyze the value of "merit of best subset found" for the constructed views.

4.1.1.2 View Construction

In multi-view relational classification, relational features are used in order to construct a set of n disjoint views $\{V^1, \ldots, V^n\}$ [71]. The process of view construction consists of two steps; namely, extracting join paths and constructing relational features [71]. These two steps are discussed below. We use the Thrombosis database (Figure 4.1) in order to explain the view construction process in more details. This database is from the PKDD 2001 Discovery Challenge [72]. Thrombosis is one of the most severe complications resulted from different collagen disease. This complication is related to anti-cardiolipin antibodies.

The Thrombosis database includes five background relations. The patient gender attribute in the Gender relation is considered the target attribute and the IgG concentration level attributes in the Antibody_Level relation is considered the privacy attribute.

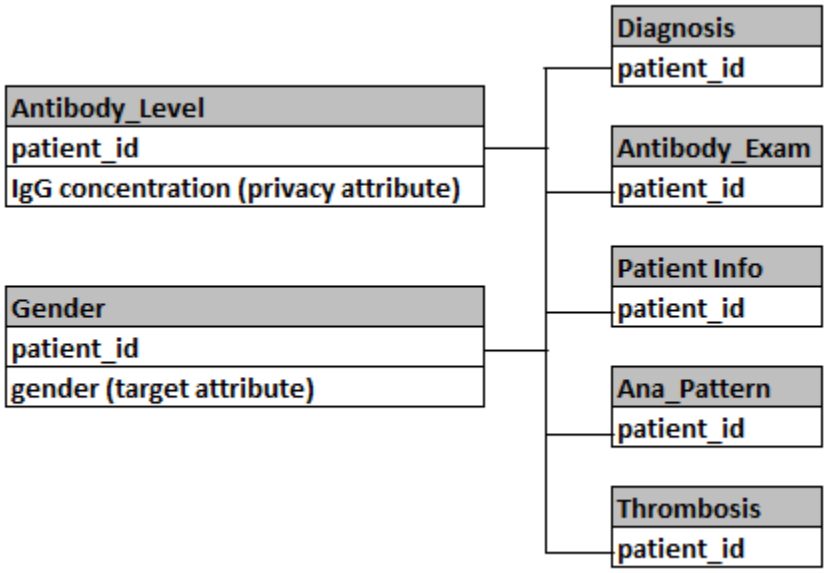


Figure 4.1: The PKDD 2001 Database Schema [73]

Extracting Join Paths

The first step is to convert the relational database schema into an undirected graph where each node refers to a table in the database while the edges refer to the joins. Each view corresponds to a unique join path which starts at the target relation and ends with one of the background relations. Following the technique presented in [71], we first collect unique join paths with two relations (one being the target relation) and subsequently add one more relation to the join path.

One constraint is imposed on each join path. That is, a relation in a join path is unique. The intuitive reason for this strategy is defined in [71]. Firstly, it is not feasible to exhaustively search all join paths since the number of join paths in a relational database is usually very large. Secondly, join paths with many relations can quickly reduce the number of entities related to the target tuples [71].

In the context of the Thrombosis database, the resulting extracted join paths are illustrated in Figure 4.2. For simplification purposes, the relations' names were replaced by their acronyms (Section 4.3.1.1).

a) Join Paths (w.r.t. target relation)
PS_PN_D
PS_PN_D_AnaP
PS_PN_D_AnaP_THR
PS_PN_D_AnaP_THR_AntiEN
PS_PN_D_AnaP_THR_AntiEN_AntiET
PS_D
PS_D_AnaP
PS_D_AnaP_THR
PS_D_AnaP_THR_AntiEN
PS_D_AnaP_THR_AntiEN_AntiET
PS_AnaP
PS_AnaP_THR
PS_AnaP_THR_AntiEN
PS_AnaP_THR_AntiEN_AntiET
PS_THR
PS_THR_AntiEN
PS_THR_AntiEN_AntiET

b) Join Paths (w.r.t. privacy relation)
AntiET_AntiEN
AntiET_AntiEN_D
AntiET_AntiEN_D_PN
AntiET_AntiEN_D_PN_AnaP
AntiET_AntiEN_D_PN_AnaP_THR
AntiET_AntiEN_PN
AntiET_AntiEN_PN_AnaP
AntiET_AntiEN_PN_AnaP_THR
AntiET_AntiEN_AnaP
AntiET_AntiEN_AnaP_THR
AntiET_D
AntiET_D_PN
AntiET_D_PN_AnaP
AntiET_D_PN_AnaP_THR
AntiET_PN
AntiET_PN_AnaP
AntiET_AnaP_THR
AntiET_THR

Figure 4.2: List of extracted join paths in the Thrombosis DB: (a) with regard to the target class and (b) with regard to the privacy class.

Constructing Relational Features

The work presented in [71] defines relational features as features that are used to extract information about a tuple (in the target relation) from the target relation itself and from other relations that are related to the target tuple. When a join path consists of a one-to-many join, an attribute derived from a join path is multi-valued [71]. Here, the role of aggregation operators comes into effect. These functions aggregate information from the multi-set in order to generate relational features.

The choice of selecting aggregation functions depends on the attributes on which they are applied and, in general, we are concerned with three types of attributes, namely nominal, binary, and numerical.

Since our study involves analyzing the effects of applying aggregation functions, we only focus on the join paths that include one-to-many relationships. Therefore, it is necessary to identify the relations which participate in one-to-many links. We call them the aggregation applicable relations. To this end, we are mainly interested in join paths which contain at

least one of these relations. In the Thrombosis database (Figure 4.1), three such relations are *Diagnosis*, *Ana Pattern*, and *Thrombosis*. Consider the *Diagnosis (D)* relation for example. Although it contains 1957 records in total, it has 1284 distinct records which implies that some patients are diagnosed with more than one disease. Table 4.2 lists the records corresponding to a given patient (ID=355009) in the *Diagnosis* table.

Table 4.1: Characteristics of the Diagnosis, Ana Pattern, and Thrombosis relations

Relation	Total Number of Records	Number of Distinct Records
Diagnosis	1957	1284
Ana Pattern	644	533
Thrombosis	193	75

Table 4.2: Records in the Diagnosis relation corresponding to ID=355009

ID	Confirm	Diagnosis	FromTest
355009	+	SLE	DT
355009	+	SJS	DT
355009	+	SLE	ST
355009	+	SJS	ST
355009	+	ITP	ST

For each of the join paths with at least one aggregation applicable relation, the corresponding view is constructed twice; with and without aggregation. The process is explained in the context of the following example. We consider the PS_PN_D_AnaP join path (Figure 4.2) to show how relational features are constructed. Table 4.3 and Table 4.4 show the records associated with patient ID = 355009 in *Patient_Info (PN)* and *Ana Pattern (AnaP)* relations, respectively.

Table 4.3: Records in the Patient_Info relation corresponding to ID=355009.

ID	Age	DescriptionDate	FirstDate	Admission
355009	26	91/08/13	89/09/07	+

Table 4.4: Records in the Ana Pattern relation corresponding to ID=355009.

ID	NAN_PA
355009	P
355009	S

The first flat file (view) represented in Table 4.5 is the result of joining the relations in the join path without applying aggregation functions.

Table 4.5: The flat file corresponding to the join path (PS_PN_D_AnaP) constructed without aggregation.

ID	Sex (PS)	Age (PN)	DescriptionDate (PN)	FirstDate (PN)	Admission (PN)	Confirm (D)	Diagnosis (D)	FromTest (D)	NAN_PA (AnaP)
355009	F	26	91/08/13	89/09/07	+	+	SLE	DT	P
355009	F	26	91/08/13	89/09/07	+	+	SLE	DT	S
355009	F	26	91/08/13	89/09/07	+	+	SJS	DT	P
355009	F	26	91/08/13	89/09/07	+	+	SJS	DT	S
355009	F	26	91/08/13	89/09/07	+	+	SLE	ST	P
355009	F	26	91/08/13	89/09/07	+	+	SLE	ST	S
355009	F	26	91/08/13	89/09/07	+	+	SJS	ST	P
355009	F	26	91/08/13	89/09/07	+	+	SJS	ST	S
355009	F	26	91/08/13	89/09/07	+	+	ITP	ST	P
355009	F	26	91/08/13	89/09/07	+	+	ITP	ST	S

The second flat file corresponding to PS_PN_D_AnaP view constructed with aggregation is illustrated in Table 4.6.

Table 4.6: The flat file corresponding to join path (PS_PN_D_AnaP) constructed using aggregation.

ID	355009
Gender (PS)	F
Age (PN)	26
Admission (PN)	+
DescriptionDate (PN)	910813
FirstDate (PN)	890907
Confirmed_count (D)	5
Diagnosis_SLE_count (D)	2
Diagnosis_SJS_count (D)	2
Diagnosis_ITP_count (D)	1
FromTest_ST_count (D)	3
FromTest_DT_count (D)	2
NAN_PA_S_pattern_count (AnaP)	1
NAN_PA_P_pattern_count (AnaP)	1

A comparison between the flat files presented in Tables 4.5 and 4.6 shows the role of applying aggregation in reducing the number of records associated with patient ID = 355009, and hence, reducing the redundancy. The number of records is reduced from ten to one in order to summarize the information in different background relations (i.e. *PN*, *D*, and *AnaP*). The information summarized over multiple records in Table 4.5 is represented in the newly constructed relational features in Table 4.6.

It was mentioned earlier in this section that different aggregation operators are used to summarize different types of data. In our example, *Confirm, Diagnosis,* and *FromTest* in the *Diagnosis* relation and *NAN_PA* in the *Ana_Pattern* relation are all nominal attributes. Therefore, aggregation operator *count* is used to perform the summarization.

Consider the *D.Diagnosis* attribute in Table 4.2. This patient is diagnosed with *SLE*, *SJS*, and *ITP* diseases. The information about this patient's diseases comes from two tables, namely, *DT* and *ST*. The *DT* table stores the basic information about patient diagnosis that is confirmed by the medical practitioner. The *ST* table stores information about patient diagnosis that is confirmed by a special laboratory examination. In the case of patient ID = 355009, the SLE and the SJS diseases are confirmed by both the medical practitioner and the laboratory examination. However, the ITP disease is confirmed only by the laboratory examination. In order to summarize this information, we count the total number of confirmations associated with each disease and represent the results in the newly constructed features (relational features) such as Diagnosis_SJS_count = 2, Diagnosis_ITP_count = 1, etc. The same procedure is followed for other nominal attributes and the final result is obtained (Table 4.6). In Table 4.6, these relational attributes are Confirmed_count, Diagnosis_SLE_count, Diagnosis_SJS_count, Diagnosis_ITP_count, FromTest_ST_count, FromTest_DT_count, NAN_PA_S_pattern_count, and NAN_PA_P_pattern_count. In addition to the relational attributes, there are other attributes in Table 4.6, such as ID, Sex, Age, Admission, and DescriptionDate. These attributes are not newly constructed. They are normal attributes that belong to some of the joined relations. We shall refer to them as the original attributes, versus relational attributes, throughout this thesis.

In the next section, we provide a detailed description of the PBIRD algorithm.

4.1.2 The PBIRD Algorithm

The PBIRD algorithm is summarized in Figure 4.3. The detailed description of the PBIRD algorithm is as follows:

Step 1: In order to construct the views, different join paths are extracted. The input is a relational database R which consists of a target relation R_t, a privacy relation R_p, and some background relations $\{R_b\}$ and the association between these tables via joins. The target

relation R_t contains the target attribute which needs to be classified and the privacy relation R_p contains the confidential attribute which needs to be protected.

Step 1:

- Extract join paths w.r.t. the target relation. For each of these join paths:
  - *Construct a corresponding Target View using aggregation and add it to Set A*
  - *Construct corresponding Target View without using aggregation and add it to Set B*
- Extract join paths w.r.t. the privacy relation. For each of these join paths:
  - *Construct corresponding Privacy View using aggregation and add it to Set C*
  - *Construct corresponding Privacy View without using aggregation and add it to Set D*

Step 2:

- For each of the sets (i.e. A, B, C, and D) identify their corresponding set of selected attributes {J}, {K}, {L}, {M} using CFS.

Step 3:

- Identify all intersections between {J}, {K}, {L}, {M} and construct the corresponding Venn diagram.

Figure 4.3: The PBIRD algorithm

Two sets of join paths with regard to the target class and the privacy class are identified. For example, consider the join paths corresponding to the Thrombosis database shown in Figure 4.2. From the list of extracted join paths, the join paths with one-to-many relationships (i.e. aggregation applicable join paths) are selected, the flat file are constructed, and added to the list of applicable views. As a result, two sets of views are identified and referred to as the *target views* and the *privacy views*. The *target views* correspond to the applicable join paths which start at the target relation. Similarly, the *privacy views* correspond to the applicable join paths which start at the privacy relation.

$$\textit{Target View} = R_t \cup \{R_b\} \cup R_p$$
$$\textit{Privacy View} = R_p \cup \{R_b\}$$

Since our work is specifically concerned with studying the effect of aggregation functions, each of the target views and the privacy views are constructed with and without aggregation. The outcome is made of four sets:

- A: target views constructed using aggregation
- B: target views constructed without using aggregation
  (for the same join paths used for set A)
- C: privacy views constructed using aggregation
- D: privacy views constructed without using aggregation
  (for the same join paths used for set C)

Step 2: Using the CFS algorithm (as discussed in Section 4.1.1.1), the sets of selected attributes that are highly correlated with the target class corresponding to A and B, and the privacy class corresponding to C and D, are identified. These sets are, namely, {J}, {K}, {L}, and {M}:

- {J}: selected attributes highly correlated to the target class corresponding to set A
- {K}: selected attributes highly correlated to the target class corresponding to set B
- {L}: selected attributes highly correlated to the privacy class corresponding to set C
- {M}: selected attributes highly correlated to the privacy class corresponding to set D

{K} and {M} consists of only original attributes while {J} and {L} include both original and relational attributes.

Step 3: The intersection (∩) of the set of attributes obtained in step 2 is identified. The list of all possible intersections is provided below:

{J} ∩ {K}: common list of attributes (corresponding to set A, and set B) that are correlated with the target class.

{J} ∩ {L}: list of attributes that are correlated with the target class (corresponding to set A) which are also correlated with the privacy class (corresponding to set C). These attributes may lead to a privacy breach.

{J} ∩ {M}: list of attributes that are correlated with the target class (corresponding to set A) which are also correlated with the privacy class (corresponding to set D). These attributes may lead to a privacy breach.

{K} ∩ {L}: list of attributes that are correlated with the target class (corresponding to set B) which are as well correlated with the privacy class (corresponding to set C). These attributes may lead to a privacy breach.

{K} ∩ {M}: list of attributes that are correlated with the target class (corresponding to set B) which are as well correlated with the privacy class (corresponding to set D). These attributes may lead to a privacy breach.

{L} ∩ {M}: common list of attributes (corresponding to set C, and set D) that are correlated with the privacy class.

Venn diagrams are used in order to illustrate the possible intersections in between the different sets (Figure 4.4). These diagrams illustrate all possible logical relations that exist in a finite collection of a set.

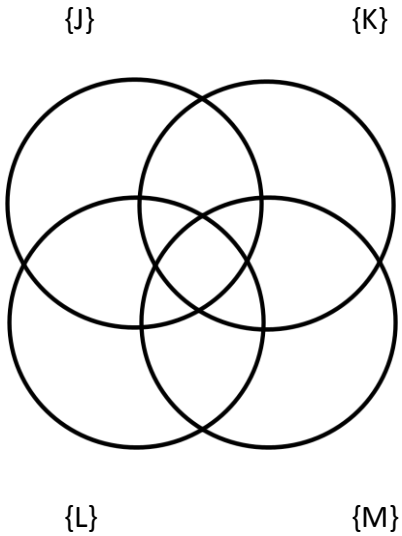


Figure 4.4: The Venn diagrams corresponding to the selected attribute sets.

In addition to the attributes that fall into the intersected (overlapping) sections, using the Venn diagrams, four non-overlapping regions are also defined, i.e. {J'}, {K'}, {L'}, and {M'}. The attributes in these regions do not belong to any of the overlapping areas. We discuss these regions in more details below:

{J'} = {J} – {K}: List of attributes which are uniquely correlated to the target class (excluding the common attributes between {J} and {K}) when the target views are constructed with aggregation.

{K'} = {K} – {J}: List of attributes which are uniquely correlated to the target class (excluding the common attributes between {K} and {J}) when the target views are constructed without aggregation.

{L'} = {L} – {M}: List of attributes which are uniquely correlated to the privacy class (excluding the common attributes between {L} and {M}) when the privacy views are constructed with aggregation.

{M'} = {M} – {L}: List of attributes which are uniquely correlated to the privacy class (excluding the common attributes between {M} and {L}) when the privacy views are constructed without aggregation.

This list provides us with important information about the correlation between the selected attributes and the target class or the privacy class. For example, having different sets of attributes in {J'} versus {K'} (or {L'} versus {M'}) indicates that aggregation changes the correlation between attributes and the amount of information obtained from a given view.

4.1.3 Evaluation Criteria

Using the list of intersections, we are able to find a set of selected attributes which are highly correlated with the target class that may lead to a privacy breach in a relational database. The following points illustrate the main evaluation criteria in examining the Venn diagrams.

1 Having different lists of attributes in {J'} versus {K'} (or in {L'} versus {M'}) indicates the fact that aggregation changes the set of selected attributes that are highly correlated with the classification target (either target class or privacy class) and uncorrelated to each other.

2 Even if we ignore the role of aggregation, using the logical relations in the resulting Venn diagram, we are able to identify a list of attributes that may lead to privacy leakage in the database. This refers to the list of selected attributes which are highly correlated to the target class (appear in {K}) and are correlated to the privacy class

(appear in {M}) as well. The attributes in the resulting intersection {K} ∩ {M} may lead to potential privacy breach in the relational database.

3 Investigating the potential privacy breach caused by applying aggregation is achieved in the following way. We investigate the set of selected attributes which are uniquely correlated with the target class i.e. appear in {J'} and are also correlated with the privacy class i.e. appear in {L} and/or {M}. This list contains the attributes which appear in the intersections {J'} ∩ {L} and {J'} ∩ {M}, respectively. Using this list, we can identify a set of attributes in the flat file constructed with aggregation that may potentially become harmful. The list of selected features in {J'} ∩ {L} or in {J'} ∩ {M} may include both the original attributes and the relational attributes and provides us with information about the impact of either type of attributes on the privacy breach.

4 Using the Venn diagrams, we also identify the list of selected attributes in {J} ∩ {K} ∩ {L} ∩ {M}. These attributes are highly correlated with the target class and the privacy class regardless of applying aggregation functions. The attributes in this region are considered harmful attributes and need attention.

4.2 Effects of Aggregation on the Classification Accuracy and Model Build Time

Our next goal is to study the effects of applying aggregation functions on the classification accuracy and the time required to build the models. Recall from Chapter 3 that, one main feature of aggregation is summarization of data which implicitly causes one to ignore some of the details in the relational databases. Aggregation also modifies the list of attributes (usually, the number and the type of attributes) and the number of records in the resulting dataset. We aim to see how, if, these changes, caused by aggregation, will impact the classification accuracy and the time required to build the models.

To this end, we re-use the target views constructed in Section 4.1.2 (sets A and B in particular). Each of the views in these sets are classified against the target attribute. Different classifiers have been employed in order perform classification. In the following section, a brief description of these classifiers is provided.

4.2.1 Classifiers Used

Four different classification algorithms were used to achieve our goal. These algorithms are the C.45, SVM, RIPPER, and Naive Bayes techniques which are briefly described as follows.

C4.5 (Quinlan [74]) is the best known system in order to induce decision trees from a set of training data. Decision tree is a flow-chart like tree structure. Each internal node represents a test on attribute, each branch denotes an outcome of the test, and each leaf node contains a class label [6]. C4.5 adopts a greedy approach where decision trees are constructed in a top-down recursive divide-and-conquer model. Rather than using the Information Gain measure used by ID3 (the predecessor of C4.5) [75], C4.5 uses Gain Ratio measure in order to perform attribute selection during the process of decision tree generation. Information Gain is biased towards tests with numerous outcomes and Gain Ratio, overcomes this bias.

In order to create a decision tree, this technique iteratively creates nodes and split the dataset into subgroups. The tree starts as a single node which represents the training samples. If all of the samples are of the same class, the node becomes a leaf. If not, the algorithm uses a selected attribute that will “best” split the samples into individual classes. The algorithm creates a branch for each value of the test attribute and the samples get partitioned. At each partition the algorithm uses the same process recursively to create a decision tree. If a given attribute becomes the test attribute at a node, it is not to be considered in any of the further nodes.

In addition to the ability of handling continuous attributes and missing values, one important feature of C4.5 compared to ID3 is tree pruning. When decision trees are built, many of the branches may reflect noise or outliers. Tree pruning addresses the problem of overfitting the data and uses statistical measures to remove least reliable branches [6].

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) is an inductive rule learner which was introduced by Cohen [76]. RIPPER is a sequential covering algorithm. In sequential covering techniques, IF-THEN rules can be extracted without generating decision trees. Rules are learned sequentially one at a time and each time a rule is learned, the tuples covered by that rule are removed. Then, the process is repeated on the remaining tuples [6].

With the RIPPER algorithm, rules are grown in a general-to-specific order. The algorithm starts off with an empty rule and then following a greedy depth-first strategy, keeps appending attribute tests to it. Therefore, each time it adds new attribute test to the current rule, it selects the attribute that mostly improves the rule quality. In other words, rather than finding rules with higher coverage, the algorithm aims to find rules that achieve highest accuracy. It follows that, it is possible to have more than one rule for a class where each rule covers different tuples. This algorithm is suitable for imbalanced class distribution and works well with noisy data. Also, the rules generated by this algorithm are easy to interpret and understand.

SVM (Support Vector Machine) [77] is a classification technique used for classifying both linear and nonlinear data. For datasets which are linearly separable, it finds a linear classification function which corresponds to a hyperplane (i.e. a decision boundary which separates the tuples of one class from another) that passes through the middle of two classes and separates them. After finding this function, a new data instance, x_n can be classified by testing the sign of the function $f(x_n)$. For example, if $f(x_n) > 0$, then x_n belongs to the positive class and vice versa. If the data is not linearly separable, the SVM algorithm should first transform the original data into a higher dimension space. Then, within this dimension, it searches for linear optimal separating hyperplane. The mapping used by SVM ensures that the dot product of the transformed vectors, i.e. the kernel function, may be computed easily. Although the training time of SVMs is very slow, due to their ability to model nonlinear decision boundaries, they are highly accurate. In addition to their role in classification, SVMs can be used for prediction as well [6].

The Naive Bayes algorithm is based on conditional probabilities. It uses Bayes' theorem formula. Bayes' theorem finds the probability that an event occurs given the probability that another event has already occurred. In general, it is better to have more than one feature in order to support the prediction of a given feature. However, some features may depend on one or more other features and those features may depend on others and so on. Including all of those dependencies into the model will make it very complicated. A Naive Bayesian classifier assumes that the presence or the absence of a particular feature is conditionally independent of the presence or the absence of other features in a dataset. It works well with

large data sets and result in high accuracy. It is an easy learner to construct and it is easily interpretable [78].

4.2.2 Procedure and Evaluation Criteria

We compare the classification accuracy and the model build time of different target views constructed with and without aggregation using the classifiers briefly mentioned in section 4.2.1.

By introducing relational features, aggregation operators increase the number of attributes in the resulting flat file. They also summarize the information and thus, decrease the total number of records in the resulting dataset. When applied on nominal and binary attributes, aggregation operators also modify the type of data in the resulting flat file. Re-consider join path PS_PN_D_AnaP discussed in Section 4.1.1.2 again. Although the type of most of the attributes in flat file constructed without aggregation is nominal (Table 4.5), the resulting flat file constructed with aggregation is mostly numerical (Table 4.6). We will specifically investigate this indirect role of aggregation in changing the overall type of a dataset on the classification accuracy of the models. Another evaluation criteria to consider is the time required to build the model, with or without aggregation.

4.3 Environmental Setup

We have used three datasets to perform our research. This section describes these databases and their characteristics. Also, at the end of this section, we discuss the experimental setup.

4.3.1 Datasets

4.3.1.1 Thrombosis DB – PKDD 2001 Discovery Challenge

Our first database is the previously introduced Thrombosis database from the PKDD 2001 discovery challenge [73]. Thrombosis is considered one of the most severe complications which arise from different collagen diseases. Studies have found that it is strongly related to anti-cardiolipin antibodies [73]. As was shown in Figure 4.1, there are seven relations associated with this database, namely, Antibody_Exam, Diagnosis, Patient_Info,

Ana_Pattern, Antibody_Level, Thrombosis, and Gender. Two classification targets have been defined for this database. The patient gender attribute in the Gender relation is considered the target attribute and the IgG concentration level attribute in the Antibody_Level relation is considered the privacy attribute. This attribute records the anti-cardiolipin antibody IgG level of the patient, indicates a personal health problem, and hence, needs to be protected [72]. The characteristics of this database are summarized in Table 4.7.

Table 4.7: Summary of the Thrombosis database

Target Attribute	Patient Gender
Target Distribution	361:200
Privacy Attribute	IgG Level
Privacy Distribution	421:349
Total Num of Tables	7

We use acronyms to refer to the relations names in this database as shown in Table 4.8.

Table 4.8: Acronyms for the relations in the Thrombosis database

Relation	Substitute Name
Gender	PS (target relation)
Patient_Info	PN
Ana_Pattern	AnaP
Diagnosis	D
Thrombosis	THR
Antibody_Level	AntiET (privacy relation)
Antibody_Exam	AntiEN

4.3.1.2 Swiss Insurance DB – ECML 1998 Discovery Challenge

Our second database was extracted from a data warehouse of a Swiss insurance company [79]. The schema of this database is depicted in Figure 4.5.

Two learning tasks i.e. Task A and Task B were included in this database [72]. Task_A classifies the partner category into class 1 or 2, and Task_B classifies the household category as a positive or a negative class. In this database, the partner category in Task A is considered the target attribute. On the other hand, the household category in Task B indicating partners' life style choices is considered the confidential attribute. In addition to

these two relations, eight other background relations were provided and were stored in *Eadr*, *Padr*, *Hhold*, *Part*, *Parrol*, *Vvert*, *Tfkomp*, and *Tfrol* tables. Table *Part* contains the personal information of all partners i.e. clients. Tables *Eadr* and *Padr* describe their electronic and postal addresses, respectively. *Hhold* contains partners' household information, and tables *Parrol*, *Vvert*, *Tfkomp*, and *Tfrol* include partners' insurance information. Table 4.9 summarizes the characteristics of this database.

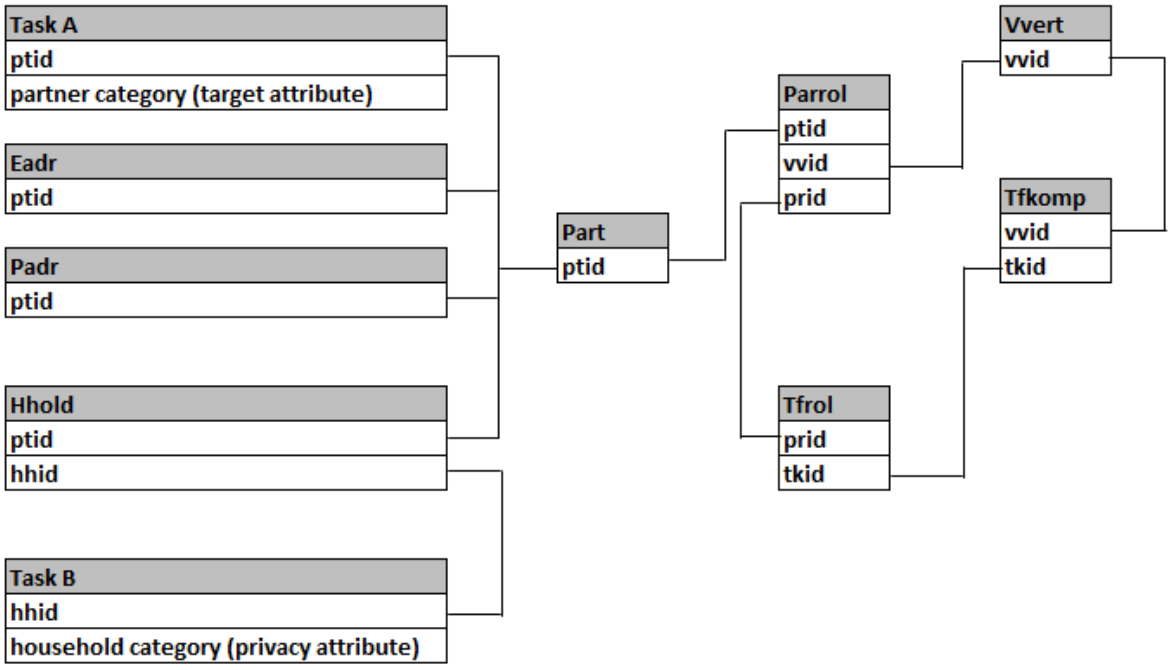


Figure 4.5: The schema of ECML 1998 database [79]

The characteristics of this database are summarized in Table 4.9.

Table 4.9: Summary of the Swiss Insurance database

Target Attribute	Partner Category
Target Distribution	5834:1498
Privacy Attribute	Household Category
Privacy Distribution	3705:3624
Total Num of Tables	10

4.3.1.3 Loan DB – PKDD 1999 Discovery Challenge

The third database used is a financial database which was published for PKDD99 discovery challenge [80]. This database, as illustrated in Figure 4.6, was offered by a Czech bank and contains typical financial data. The aim of multi-relational classification task for this database is to predict the risk of granting a personal loan to new customers.

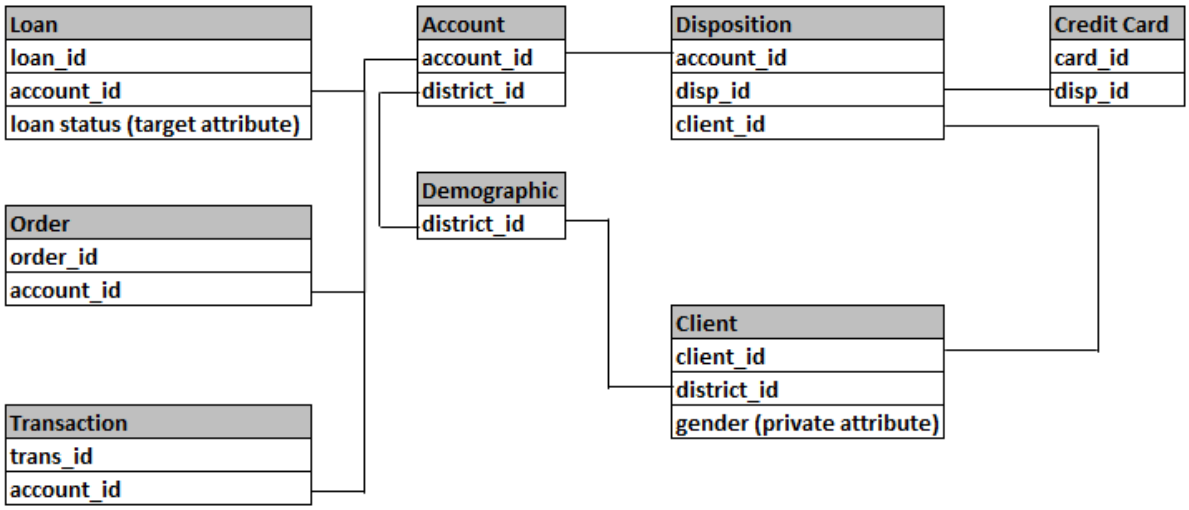


Figure 4.6: The PKDD99 Financial Database **[80]**

In our experiment, we used the data which was prepared by Yin et al. in [40]. This database consists of eight tables, i.e. *Loan, Account, Order, Transaction, Demographic, Client, Disposition,* and *Credit Card*.

The *Loan* table is the target relation and the status attribute in this table which indicates the status of a loan is considered the target attribute of the database. The Client table contains information about clients. In our experiment, the gender attribute in the Client table is considered the confidential attribute which needs to be protected [72]. The remainder of the tables are considered background relations.

Similar to the Thrombosis database, we have used abbreviations instead of relations' names and the list is shown in Table 4.10.

Table 4.10: Acronyms for the relations in the loan database

Relation	Substituted Name
Loan	L (the target relation)
Account	A
Disposition	D
Demographic	DG
Credit Card	CC
Order	O
Transaction	T
Client	CL (the privacy relation)

The characteristics of this database are illustrated in Table 4.11.

Table 4.11: Summary of the Loan database

Target Attribute	Loan Status
Target Distribution	324:76
Privacy Attribute	Gender
Privacy Distribution	2724:2645
Total Num of Tables	8

4.3.2 Experimental Setup

We have used the standard ten-fold cross validation method to conduct our experiment. In ten-fold cross validation, data is first divided into ten folds randomly. Each of the ten folds is then used as the test set, and the remaining nine folds are used as the training data. This process is repeated ten times where each of the folds is used exactly once as the test data. Finally, the results are averaged in order to get a final single estimation [6].

The MySQL database management system is the RDBMS used during the view construction process. In order to implement CFS, the previously introduced CfsSubsetEval function (the implementation of CFS algorithm in WEKA) with best first search strategy option was employed. To perform the comparative study, we used WEKA's JRip, SMO, J48, and Naive Bayes which correspond to RIPPER, SVM, C4.5, and Naive Bayesian classifiers, respectively. We run all experiments on a workstation with a 2.3 GHz AMD Dual-Core CPU, and 4 GBytes of RAM.

4.4 Summary

This chapter presented the details of the proposed PBIRD algorithm and the background information associated with it. We first introduced view construction and Correlation-based Feature Selection; the two main techniques employed by the PBIRD algorithm.

In order to implement the PBIRD algorithm, firstly, different join paths starting at the target and the privacy relations are extracted and their corresponding flat files with and without aggregation, are constructed. In the following step, the CFS algorithm is employed in order to identify a list of features that are highly correlated with the classification target, while being uncorrelated with one another. Finally, all possible logical relations between these features are identified. These logical relations are then used in order to identify a set of harmful features that may lead to a privacy breach during classification of relational databases.

In this chapter, we also introduced our methodology to perform a comparative study in order to investigate the impact of aggregation on the classification accuracy and the time required to build the models. We provided a description of the databases used in our work and presented the experimental setup.

In the next chapter, the experimental results of the proposed algorithm and the aggregation-based comparative study are presented.

Chapter 5

Evaluation and Results

In Chapter 4, we introduced the PBIRD algorithm, designed and implemented to investigate the impact of applying aggregation functions on the privacy of a relational database. Identifying the set of features which may lead to a privacy breach enables us to provide further protection for the sensitive information in a relational database. In this chapter, we apply our proposed algorithm to the three datasets discussed in Chapter 4 and evaluate the results.

In Chapter 4, we also discussed our methodology to assess the impact of aggregation on the classification accuracy and the time required to build the models. In this chapter, we evaluate this comparative study using the three databases introduced in Chapter 4.

This chapter is organized as follows. For each of the databases, we first perform the necessary pre-processing, apply the PBIRD algorithm, and analyze the obtained results. A comparison of the "merit of best subset found" for different constructed views (with and without aggregation) is presented next. Finally, we compare the classification accuracy and the model building time of different flat files constructed, again with and without aggregation.

5.1 Experimental Results

In the following sections, we present the experimental results of applying our methodologies on the Thrombosis database, the Swiss Insurance database, and the Loan database.

5.1.1 Thrombosis DB – PKDD 2001 Discovery Challenge

Recall from Section 4.3.1.1 that, two classification targets have been defined for this database. The patient gender attribute in the Gender (PS) relation is considered the target attribute. On the other hand, the IgG concentration level attribute in the Antibody_Exam_Target2 (AntiET) relation is considered the privacy attribute. This attribute records the anti-cardiolipin antibody IgG level of the patient and needs to be protected.

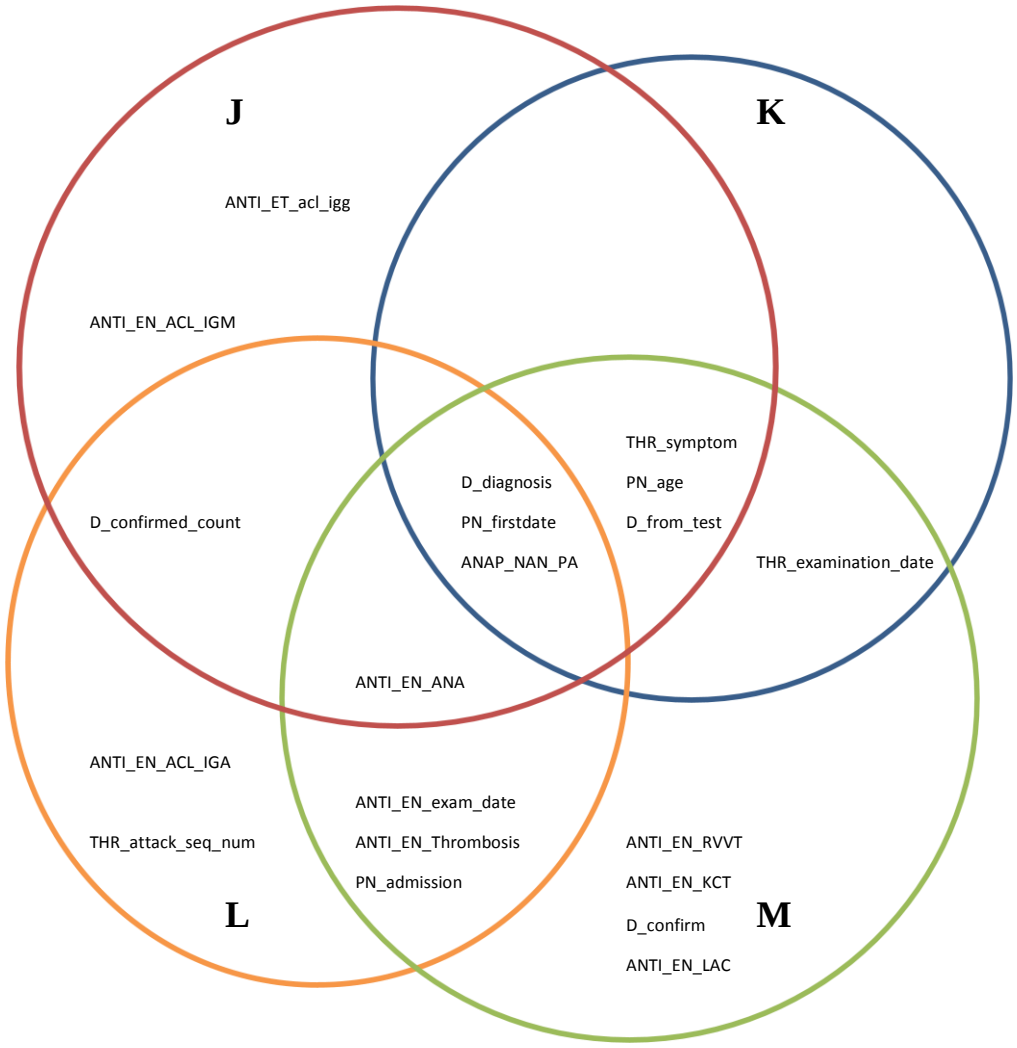
Following the PBIRD algorithm, we first extract two sets of join paths; one starting at the target relation PS, and another starting at the privacy relation AntiET (Figure 5.1) and construct the target views and the privacy views accordingly. In the next step, for each of these sets, the attributes that are highly correlated with the target class in {J} and {K}, and the privacy class in {L} and {M} are identified. Finally, we construct the Venn diagrams in order to determine a set of attributes which may lead to a privacy breach in the relational database.

a) Join Paths (w.r.t. Target class)
PS_PN
PS_PN_D
PS_PN_D_AnaP
PS_PN_D_AnaP_THR
PS_PN_D_AnaP_THR_AntiEN
PS_PN_D_AnaP_THR_AntiEN_AntiET
PS_D
PS_D_AnaP
PS_D_AnaP_THR
PS_D_AnaP_THR_AntiEN
PS_D_AnaP_THR_AntiEN_AntiET
PS_AnaP
PS_AnaP_THR
PS_AnaP_THR_AntiEN
PS_AnaP_THR_AntiEN_AntiET
PS_THR
PS_THR_AntiEN
PS_THR_AntiEN_AntiET

b) Join Paths (w.r.t. Privacy class)
AntiET_AntiEN
AntiET_AntiEN_D
AntiET_AntiEN_D_PN
AntiET_AntiEN_D_PN_AnaP
AntiET_AntiEN_D_PN_AnaP_THR
AntiET_AntiEN_PN
AntiET_AntiEN_PN_AnaP
AntiET_AntiEN_PN_AnaP_THR
AntiET_AntiEN_AnaP
AntiET_AntiEN_AnaP_THR
AntiET_D
AntiET_D_PN
AntiET_D_PN_AnaP
AntiET_D_PN_AnaP_THR
AntiET_PN
AntiET_PN_AnaP
AntiET_AnaP_THR
AntiET_THR

Figure 5.1: List of extracted join paths in the Thrombosis database: (a) with regard to the target class and (b) with regard to the privacy class

Recall from Section 4.1.1.1 that using the CFS algorithm, we obtain a list of attributes which are highly correlated with (predictive of) the classification target, and highly uncorrelated with (not predictive of) each others.



- Selected Attributes – Target Views – with aggregation (Group J)
- Selected Attributes – Target Views – without aggregation (Group K)
- Selected Attributes – Privacy Views – with aggregation (Group L)
- Selected Attributes – Privacy Views – without aggregation (Group M)

Figure 5.2: The Thrombosis database Venn diagrams

The tabular representation of the Venn diagrams is shown in Figure 5.3.

All Selected Attributes	(Group J)	(Group K)	(Group L)	(Group M)
ANTI_ET_acl_igg	•			
ANTI_EN_ACL_IGM	•			
D_confirmed_count	•		•	
ANTI_EN_ACL_IGA			•	
THR_attack_seq_num			•	
ANTI_EN_ANA	•		•	•
D_diagnosis	•	•	•	•
PN_first_date	•	•	•	•
ANAP_NAN_PA	•	•	•	•
ANTI_EN_exam_date			•	•
ANTI_EN_Thrombosis			•	•
PN_admission			•	•
THR_symptom	•	•		•
PN_age	•	•		•
D_from_test	•	•		•
THR_examination_date		•		•
ANTI_EN_KCT				•
ANTI_EN_RVVT				•
ANTI_EN_LAC				•
D_confirm				•

Figure 5.3: Tabular representation of the selected attributes associated with the Thrombosis database Venn diagrams

Consider *D_diagnosis*, *PN_firstdate*, *ANAP_NAN_PA*, *THR_symptom*, *PN_age*, and *D_from_test*. These attributes belong to the overlapping region of {J} ∩ {K} and are highly correlated with the target class either with or without applying aggregation during the flattening process. On the other hand, there are features that are correlated with the target class only when the target views are constructed with aggregation such as *ANTI_ET_acl_igg*, *ANTI_EN_ACL_IGM*, *D_confirmed_count*, and *ANTI_EN_ANA* in {J'}. Similarly, there is one selected feature i.e. *THR_examination_date* which appears in the {K'} list and is correlated with the target class only when the target views are constructed without aggregation.

Further investigation of features correlated with the privacy class validates this observation. Consider {L} and {M} in the Venn diagrams. In addition to the attributes that are commonly correlated with the privacy class in {L}∩{M}, i.e. *ANTI_EN_exam_date*, *D_diagnosis*, *PN_firstdate*, *ANAP_NAN_PA*, *ANTI_EN_ANA*, *ANTI_EN_Thrombosis*, and *PN_admission*, we are able to identify features which are correlated with the privacy class

only when the privacy views are constructed with aggregation in {L'} or without aggregation in {M'}.

Consider *D_confirmed_count*, *ANTI_EN_ACL_IGA*, and *THR_attack_seq_num* in {L'}. These attributes are highly correlated with the privacy class only when the views are constructed with aggregation. On the other hand, attributes such as *THR_symptom*, *PN_age*, *D_from_test*, *THR_examination_date*, *ANTI_EN_RVVT*, *ANTI_EN_KCT*, *D_confirm*, and *ANTI_EN_LAC* in {M'} are correlated with the privacy class when the views are constructed without applying aggregation functions.

Having different lists of attributes in {J'} versus {K'} and similarly in {L'} versus {M'} shows the impact of aggregation on changing these correlations. Recall from Section 4.1.1.2 that, aggregation results in constructing new features, the so-called relational features. As such, for a given join path, the flat file constructed with aggregation will have different attributes compared to the flat file constructed without aggregation. When we apply the CFS algorithm on the flat file constructed with aggregation, the list of selected attributes obtained by the CFS algorithm consists of both relational and original features. This implies that, in addition to the original features, relational features also play an important role in predicting the target class. Aggregation even changes the list of original features in {J'} versus {K'} and similarly in {L'} versus {M'}. To this end we conclude that, aggregation changes the correlation of the features with the classification target (either the target class or the privacy class).

Before discussing the role of aggregation in the privacy breach of this database, we first consider the potential privacy breach in this database, in general. Consider {K} ∩ {M} which includes *D_diagnosis*, *PN_firstdate*, *ANAP_NAN_PA*, *THR_symptom*, *PN_age*, *D_from_test* and *THR_examination_date*. This intersection provides us with a list of features which are highly correlated with the target class and are also correlated with the privacy class and therefore, may lead to a potential privacy breach in the database.

In studying the effect of aggregation on the privacy of this database, we notice the following. There are five attributes that belong to the overlapping region {J} ∩ {L}. They are namely *D_confirmed_count*, *ANTI_EN_ANA*, *D_diagnosis*, *ANAP_NAN_PA*, *PN_firstdate*. Out of these five attributes, two attributes i.e. *D_confirmed_count* and

ANTI_EN_ANA are correlated with the target class only when the flat files are constructed using aggregation. These two attributes are also highly correlated with the privacy class and may potentially lead to a privacy breach in the Thrombosis database. In fact, *D_confirmed_count* belongs to the overlapping region of {J'} and {L'}. It implies that it is highly correlated to the privacy class when the privacy view itself is constructed using aggregation.

In the Venn diagrams, attributes that belong to {J} ∩ {K} ∩ {L} ∩ {M}, namely, D_diagnosis, *PN_firstdate*, and *ANAP_NAN_PA* are of a considerable importance. These attributes are highly correlated with the target class and may lead to privacy leakage whether the views are constructed with or without aggregation.

In this database, aggregation introduces new attributes in the set of selected features such as *D_confirmed_count* and *ANTI_EN_ANA* that may potentially become harmful and negatively impacts the privacy of the database. However, looking at it from a different perspective, aggregation could eventually be used as an effective tool to provide more protection for the database. Constructing the views without aggregation provides us with a list of potentially dangerous attributes in {K} ∩ {M}. In addition to this list, if we identify and protect the potentially dangerous attributes obtained as a result of applying aggregation in {J'} ∩ {L} and {J'} ∩ {M}, we indeed provide a better privacy in our database.

Next, we evaluate the "merit of best subset found" (as discussed in Section 4.1.1) of the target views constructed with and without aggregation. When a given view is constructed with and without aggregation, as a result, two structurally different datasets are obtained.

Although these two datasets have some common attributes, most of their attributes are essentially different. Our goal is to investigate the impact of aggregation on the heuristic goodness of a feature set. The results of such comparison are shown in Figure 5.4. Recall from Equation 4.1 that, the heuristic goodness of feature subset depends on three factors. These factors are, namely, the number of features in the selected subset (k), the average of the correlation between the features and the class ($\overline{r_{cf}}$), and the average of inter-correlation in between features within the feature subset ($\overline{r_{ff}}$). In discussing the Venn diagrams of Figure 5.2, we noticed that aggregation changes the list of features that are highly correlated with the classification target, while being uncorrelated with one another. Furthermore, by constructing relational features, aggregation changes the number of attributes in the

constructed views. This impacts the number of components (*k*) in the selected subset. In general, the results show that with aggregation, better feature subsets are obtained. This thus implies that aggregation has a "positive role" in that it aids to identify features that have strong prediction ability against the classification target. These features are especially important when we investigate the potentially harmful features in the database, since they may lead to accurate privacy disclosure.

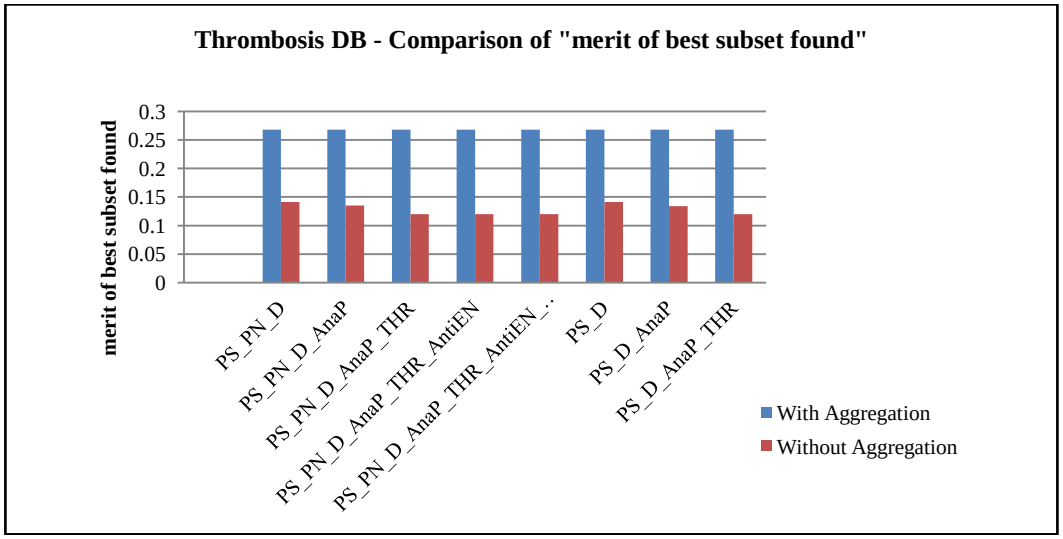


Figure 5.4: The comparison of "merit of best subset found" for different target views in the Thrombosis database

Classification Accuracy Measurement

When we compare the classification accuracy of different views constructed with and without aggregation in this database, (Figures 5.5, 5.6, 5.7, and 5.8), we observe that all classifiers return better classification accuracies when the views are constructed without aggregation. As it was mentioned earlier in this section, by constructing relational features, aggregation functions change the structure of the resulting flat files. The type and the number of attributes in the flat file constructed with aggregation depends on the type of the attributes in the joined relations.

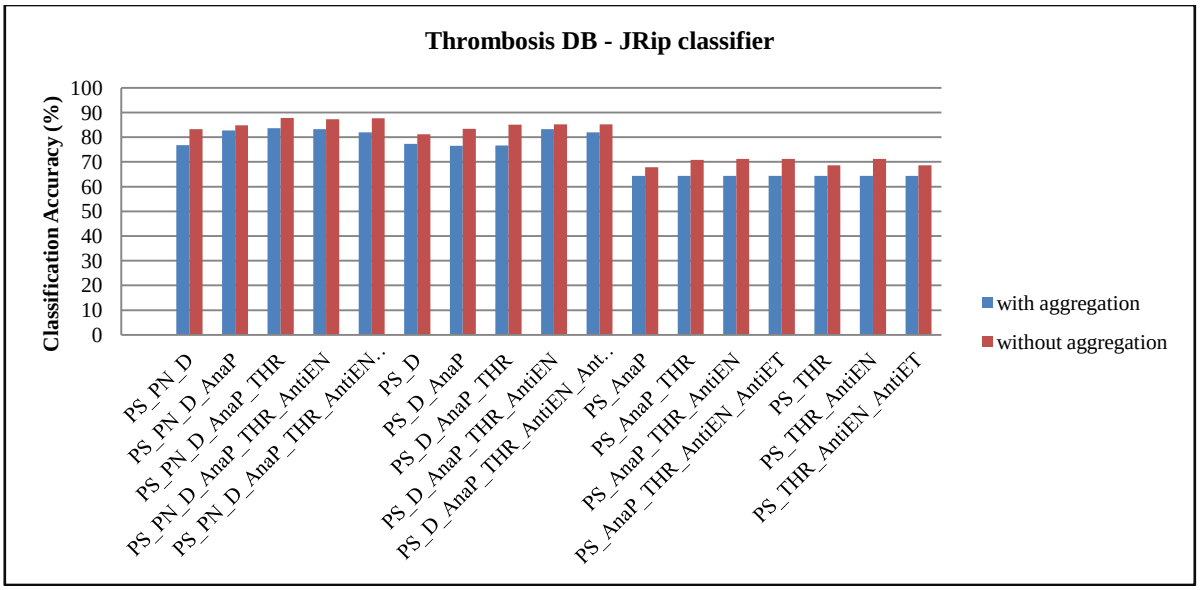


Figure 5.5: The comparison of classification accuracy of different target views (with and without aggregation) in the thrombosis database using JRip classifier

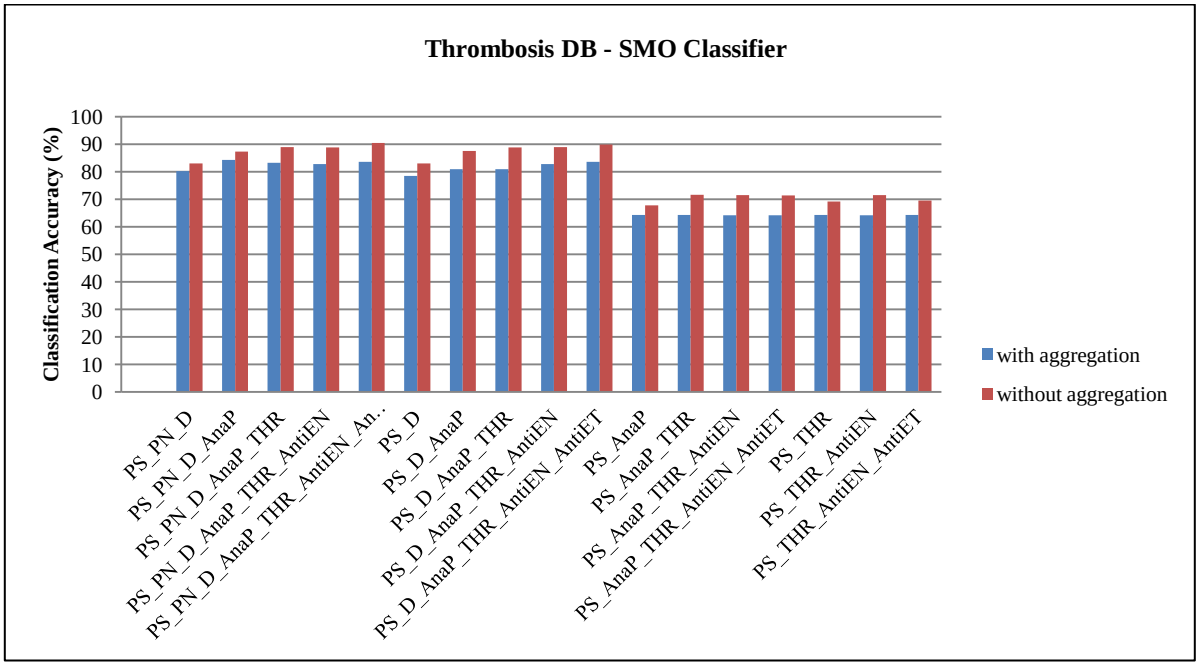


Figure 5.6: The comparison of classification accuracy of different target views (with and without aggregation) in the thrombosis database using SMO classifier

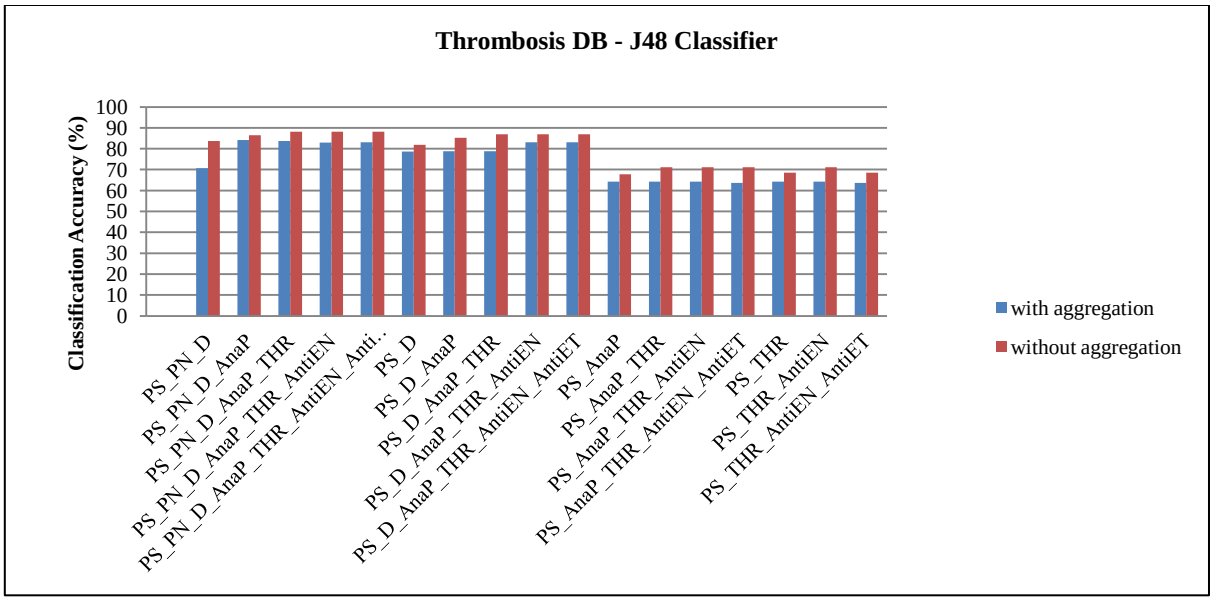


Figure 5.7: The comparison of classification accuracy of different target views (with and without aggregation) in the thrombosis database using J48 classifier

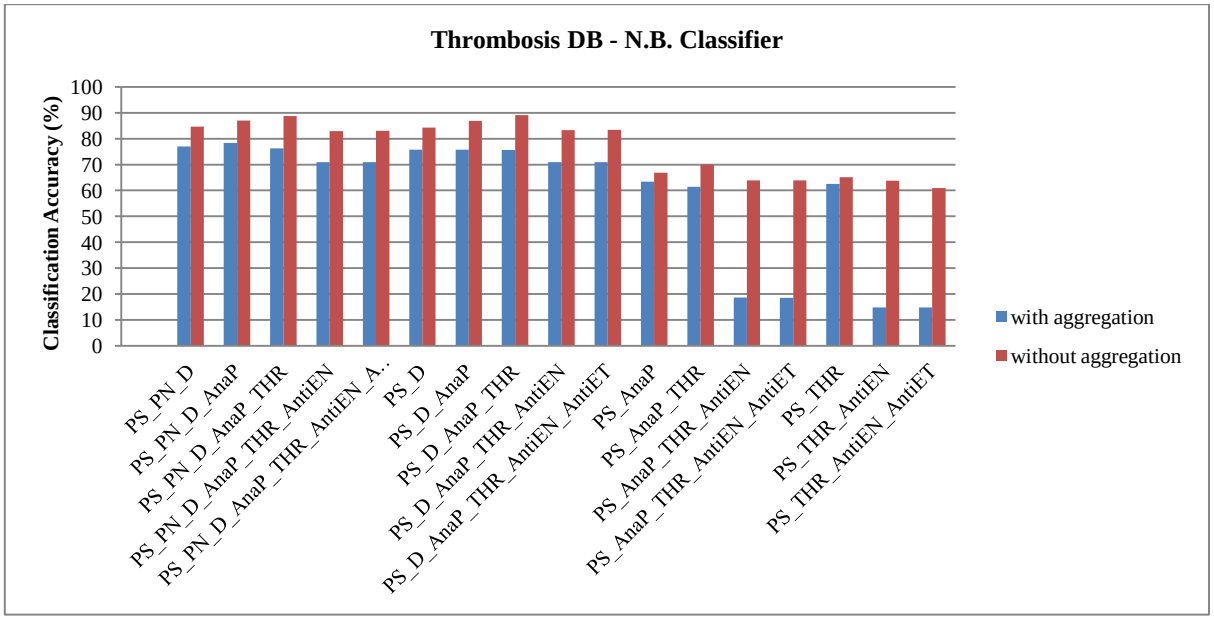


Figure 5.8: The comparison of classification accuracy of different target views (with and without aggregation) in the thrombosis database using N.B. classifier

In order to show how aggregation functions change the number and the type of attributes in the resulting dataset, we consider the PS_D_AnaP join path. Since the attributes of the relations in this join path are nominal, the *count* aggregation function is applied to calculate the number of occurrences of the given values [71]. These calculated numbers (essentially integer values) are stored in the relational features. Table 5.1 shows the records associated with patient ID 2788281 in the *Diagnosis (D)* table.

Table 5.1: Records in the Diagnosis relation corresponding to ID=2788281

ID	Confirm	Diagnosis	FromTest
2788281	+	SLE	DT
2788281	+	APS	DT
2788281	+	SLE	ST
2788281	+	APS	ST
2788281	+	PREG	ST

Attributes *Confirm*, *Diagnosis*, and *FromTest* are all nominal. In our example, the *Diagnosis* attribute includes three values: SLE, APS, and PREG. We calculate the number of occurrences of each value and obtain count(SLE) = 2, count(APS) = 2, and count(PREG)=1. Then, relational features Diagnosis_SLE_count, Dianosis_APS_count, and Diagnosis_PREG_count are constructed in order to store the above values. We notice that, although *Diagnosis* attribute consists of nominal values such as SL, APS, and PREG, the corresponding relational features are numerical. Similarly, in order to store the number of occurrences of the '+' value in the *Confirm* attribute, the relational feature Confirm_positive_count is constructed. Once more, although the type of the values in the *Confirm* attribute is nominal, the relational feature includes numerical values. We have the same scenario for the *FromTest* attribute and as such, two relational features i.e. FromTest_ST_count, and FromTest_DT_count are constructed. When the PS_D_AnaP view is constructed with aggregation, we obtain the resulting flat file as shown in Table 5.2.

Table 5.2: The flat file corresponding to join path (PS_D_AnaP) constructed with aggregation

ID	2788281
Sex (PS)	F
Confirmed_positive_count (D)	5
Diagnosis_SLE_count (D)	2
Diagnosis_APS_count (D)	2
Diagnosis_PREG_count (D)	1
FromTest_ST_count (D)	3
FromTest_DT_count (D)	2
NAN_PA_S_pattern_count (AnaP)	1
NAN_PA_P_pattern_count (AnaP)	1

When the same view is constructed without aggregation we obtain the flat file shown in Table 5.3.

Table 5.3: The flat file corresponding to join path (PS_D_AnaP) constructed without aggregation

ID	Sex (PS)	Confirm (D)	Diagnosis (D)	FromTest (D)	NAN_PA (AnaP)
2788281	F	+	SLE	DT	P
2788281	F	+	SLE	DT	S
2788281	F	+	APS	DT	P
2788281	F	+	APS	DT	S
2788281	F	+	SLE	ST	P
2788281	F	+	SLE	ST	S
2788281	F	+	APS	ST	P
2788281	F	+	APS	ST	S
2788281	F	+	PREG	ST	P
2788281	F	+	PREG	ST	S

Considering another join path such as PS_AnaP_THR, we observe a similar situation. In the THR relation, except the identification attribute, the other three attributes i.e. *THR_examination_date, THR_symptom*, and *THR_attack_seq_num* are nominal. In the AnaP relation, other than the identification attribute, the only other attribute i.e. *AnaP_NAN_PA* is nominal, and so on.

We notice that, although the overall data type of the flat files constructed without aggregation is nominal, the flat files constructed with aggregation (via applying *count*) consists of mostly numerical attributes. The attributes in the Thrombosis database are nominal. It follows that, a shift from a completely nominal dataset to a numerical dataset

reduces the classification accuracy across different views. One possible reason for this overall reduction in the classification accuracy is the possible loss of information caused by aggregation.

5.1.2 Swiss Insurance DB – ECML 1998 Discovery Challenge

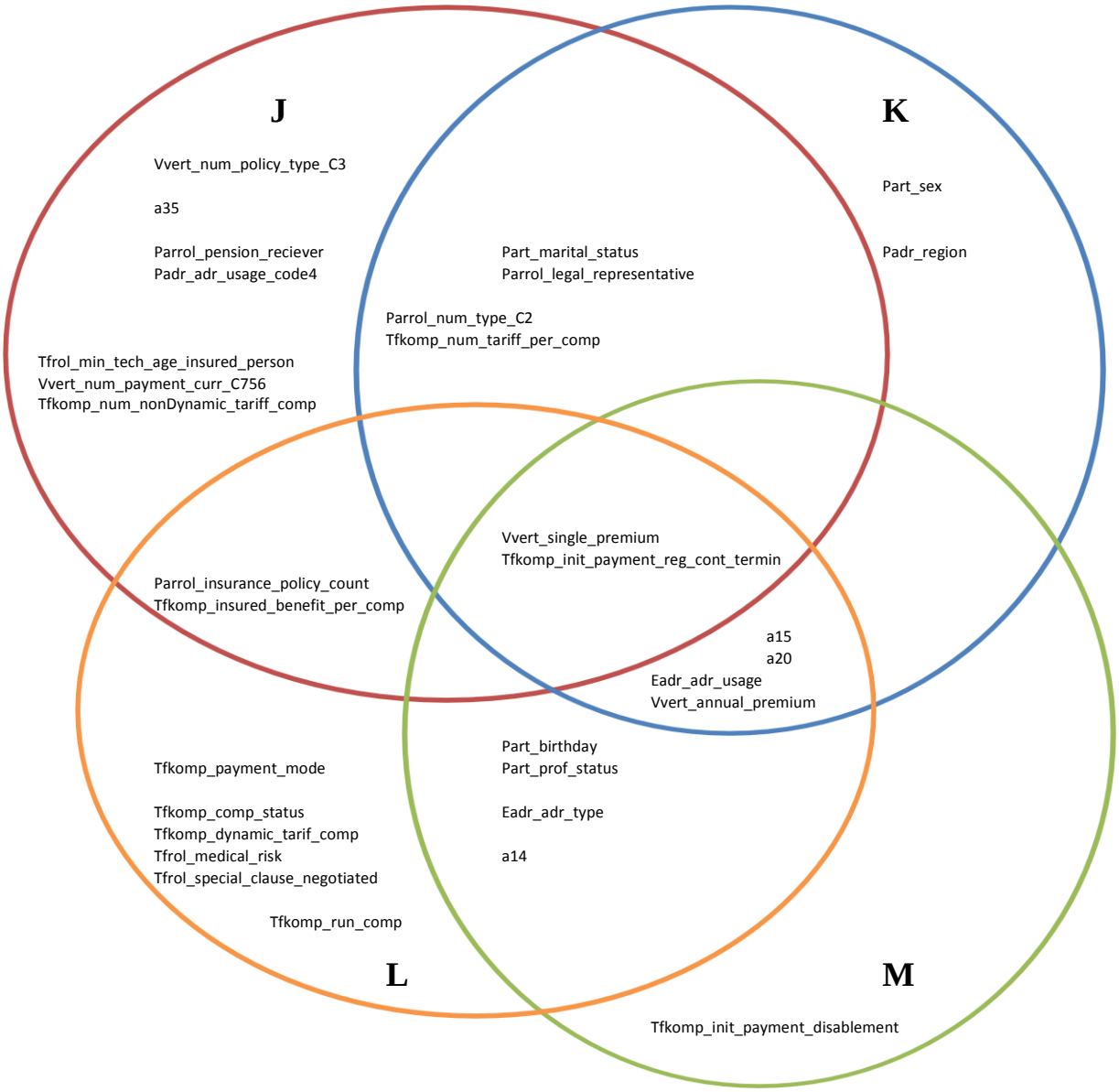
Recall from Section 4.3.1.2 that, in this database, partner category in Task_A and household category in Task_B are considered the target and the confidential attributes, respectively. Similar to the procedure explained in Section 5.1.1, we first extract two sets of join paths with regard to the target class and the privacy class as it is show in Figure 5.9.

a) Join Paths (w.r.t. Target class)
TaskA_Eadr
TaskA_Padr
TaskA_Part
TaskA_Part_Parrol
TaskA_Part_Parrol_Vvert
TaskA_Part_Parrol_Vvert_Tfkomp
TaskA_Part_Parrol_Tfrol
TaskA_Eadr_Padr
TaskA_Eadr_Padr_Part
TaskA_Eadr_Padr_Part_Parrol
TaskA_Eadr_Padr_Part_Parrol_Vvert
TaskA_Eadr_Padr_Part_Parrol_Vvert_Tfkomp
TaskA_Eadr_Padr_Part_Parrol_Tfrol
TaskA_Eadr_Padr_Part_Hhold_TaskB
TaskA_Eadr_Padr_Part_Hhold_TaskB_Parrol
TaskA_Eadr_Padr_Part_Hhold_TaskB_Parrol_Vvert
TaskA_Eadr_Padr_Part_Hhold_TaskB_Parrol_Vvert_Tfkomp
TaskA_Eadr_Padr_Part_Hhold_TaskB_Parrol_Tfrol

b) Join Paths (w.r.t. Privacy class)
TaskB_Hhold
TaskB_Hhold_Part
TaskB_Hhold_Part_Eadr_Padr
TaskB_Hhold_Part_Eadr_Padr_Parrol
TaskB_Hhold_Part_Eadr_Padr_Parrol_Vvert
TaskB_Hhold_Part_Eadr_Padr_Parrol_Vvert_Tfkomp
TaskB_Hhold_Part_Eadr_Padr_Parrol_Tfrol

Figure 5.9: List of extracted join paths in the Swiss insurance database: (a) with regard to the target class and (b) with regard to the privacy class

We then identify the relations that are involved in one-to-many links in order to find the aggregation applicable views. In this database, the identified relations are *Parrol*, *Vvert*, *Tfkomp*, and *Tfrol* with 62748, 20941, 43809, and 40213 records respectively. The target relation, *TaskA*, has 7332 records while the privacy relation, *TaskB*, includes 7329 records. After extracting the join paths, constructing the views, and obtaining the set of selected features, the Venn diagrams associated with the Swiss Insurance database are obtained. See Figure 5.10 below.



- Selected Attributes – Target Views – with aggregation (Group J)
- Selected Attributes – Target Views – without aggregation (Group K)
- Selected Attributes – Privacy Views – with aggregation (Group L)
- Selected Attributes – Privacy Views – without aggregation (Group M)

Figure 5.10: The Swiss insurance database Venn diagrams

The tabular representation of the Venn diagram for this database is shown in Figure 5.11.

All Selected Attributes	(Group J)	(Group K)	(Group L)	(Group M)
Vvert_num_policy_type_C3	●			
a35	●			
Parrol_pension_reciever	●			
Padr_adr_usage_code4	●			
Tfrol_min_tech_age_insured_person	●			
Vvert_num_payment_curr_C756	●			
Tfkomp_num_nonDynamic_tariff_comp	●			
Parrol_insurance_policy_count	●		●	
Tfkomp_insured_benefit_per_comp	●		●	
Tfkomp_payment_mode			●	
Tfkomp_comp_status			●	
Tfkomp_dynamic_tarif_comp			●	
Tfrol_medical_risk			●	
Tfrol_special_clause_negotiated			●	
Tfkomp_run_comp			●	
Part_marital_status	●	●		
Parrol_legal_representative	●	●		
Parrol_type_C2	●	●		
Tfkomp_num_tariff_per_comp	●	●		
Vvert_single_premium	●	●	●	●
Tfkomp_init_payment_reg_cont_termin	●	●	●	●
a15		●	●	●
a20		●	●	●
Eadr_adr_usage		●	●	●
Vvert_annual_premium		●	●	●
Part_birthday			●	●
Part_prof_status			●	●
Eadr_adr_type			●	●
a14			●	●
Tfkomp_init_payment_disablement				●
Part_sex		●		
Part_region		●		

Figure 5.11: Tabular representation of the selected attributes associated with the Thrombosis database Venn diagrams

Next, we study the role of aggregation in changing the correlation of the selected attributes with the classification target (either the target class or the privacy class) and among themselves. Consider {J'} versus {K'}. {J'} includes nine attributes, namely, *Vvert_num_policy_type_C3, a35, Parrol_pension_reciever, Padr_adr_usage_code4, Tfrol_min_tech_age_insured_person, Parrol_insurance_policy_count, Vvert_num_payment_curr_C756 ,Tfkomp_num_nonDynamic_tariff_comp,* and *Tfkomp_insured_benefit_per_comp*.

These attributes are highly correlated with the target class only when the target views are constructed using aggregation. On the other hand, {K'} includes *Part_sex, Padr_region, Hhold_a15, Hhold_a20, Eadr_adr_usage,* and *Vvert_annual_premium* which are correlated to the target class only when the views are constructed without applying aggregation functions.

We have the same comparison between the set of attributes in {L'} versus {M'}. In {L'}, eight attributes namely, *Parrol_insurance_policy_count,Tfkomp_insured_benefit_per_comp , Tfrol_medical_risk, Tfkomp_payment_mode,Tfkomp_dynamic_tarif_comp, Tfkomp_comp_ status, Tfrol_special_clause_negotieated,* and *Tfkomp_run_comp* are correlated with the privacy class only when the privacy views are constructed using aggregation. {M'} contains only the *Tfkomp_init_payment_disablement* attribute. In order to explain this behaviour, similar to the Thrombosis database, we consider the role of aggregation in constructing new features, reducing the number of records, and eventually changing the structure of the resulting datasets. Given a join path, and its corresponding views constructed with and without aggregation, the CFS algorithm returns different sets of selected attributes. From the Venn diagrams (Figure 5.10) we observe that, {J'} does not only include relational attributes. There, we identify both original attributes such as *a35, Parrol_pension_reciever* and relational attributes such as *Parrol_insurance_policy_count* and *Vvert_num_payment _curr_C756*. We also observe a different list of original attributes in {J'} compared with {K'}. There are original attributes such as *a35* and *Parrol_pension_reciever* which appear in {J'} and do not appear in {K'} and vice versa.

To this end, similar to the Thrombosis database, we conclude that aggregation changes the correlation between the selected attributes and the classification target (being the target attribute or the privacy attribute).

The intersection of {K} and {M} includes *Vvert_single_permium, Eadr_adr_usage, a15, a20, Tfkomp_init_payment_reg_cont_termin,* and *Vvert_annual_premium*. These selected attributes are obtained when both the target views and the privacy views are constructed without aggregation. It implies that, even if the role of aggregation is ignored, the PBIRD algorithm could still be used to identify potential privacy breach in relational databases.

We are able to identify two attributes, namely, *Vvert_single_premium* and *Tfkomp_init_payment_reg_cont_termin* in {J} ∩ {K} ∩ {L} ∩ {M}. These attributes are

highly correlated with the target attribute and the privacy attribute whether the views are constructed with or without aggregation. These attributes are of a special importance and may lead to a privacy breach in the database, and need to be taken into consideration.

With regard to studying the effect of aggregation on the privacy of the database, we are able to identify attributes such as *Tfkomp_insured_benefit_per_comp* and *Parrol_insurance_policy_count* in {J'} ∩ {L'} area. According to the Venn diagrams, {J'} and {L'} contain attributes which are only correlated with the target class and the privacy class when the views are constructed with aggregation.

Next, we examine the "merit of the best subset found" obtained as a result of applying the CFS algorithm on the target views (constructed with and without aggregation). The comparison result is displayed in Figure 5.12.

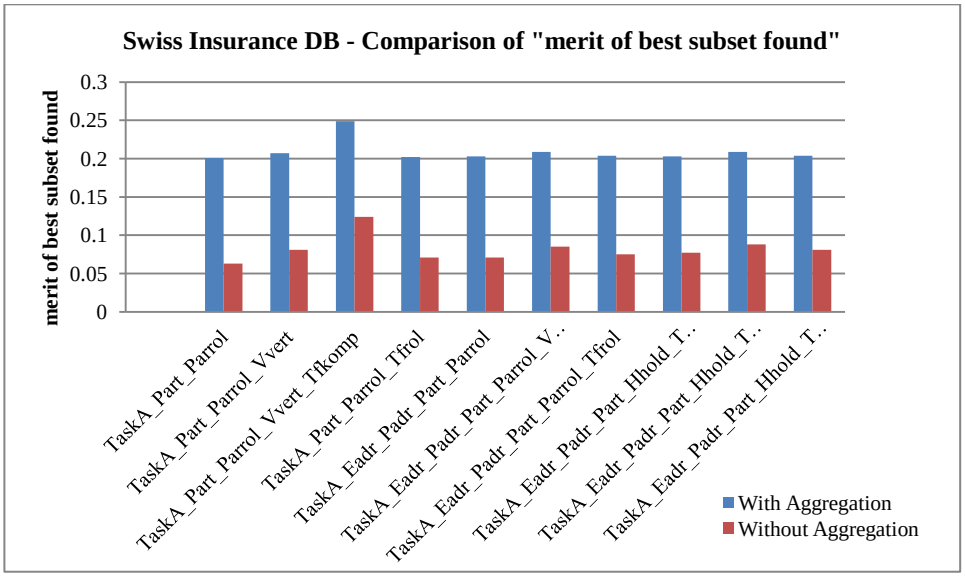


Figure 5.12: The comparison of "merit of best subset found" for different target views in the Swiss insurance database

Similar to the Thrombosis database, the target views constructed with aggregation result in higher "merit of best subset found". A justification similar to that discussed in Section 5.1.1 may be used to explain this behaviour. With aggregation we obtain feature subsets that have strong correlation with the classification target and are weakly inter-correlated with one another. Also, when aggregation is applied, relational features are constructed, which impacts the value of *k* in Equation 4.1. For example, consider join path

TaskA_Part_Parrol_Vvert_Tfkomp. Including the Tfkomp relation (with many attributes) result in a higher value of "merit of best subset found" compared with other views. Once again, the results show that, with aggregation better feature subsets are obtained.

Classification Accuracy Measurement

The size of this database is much larger than the Thrombosis database and the Loan database. This was especially evident when the views were constructed without aggregation. For example, there are 143,924 records associated with the TaskA_Part_Eadr_Padr_Parrol_Vvert_Tfkomp view. Similarly, the view TaskA_Part_Parrol_Vvert_Tfrol has 133,009 records. In this database, building the classification models using the SMO classifier was not possible and the models failed to converge.

A comparison of the classification accuracy of the constructed flat files for the three other classifiers is illustrated in Figures 5.13, 5.14, and 5.15.

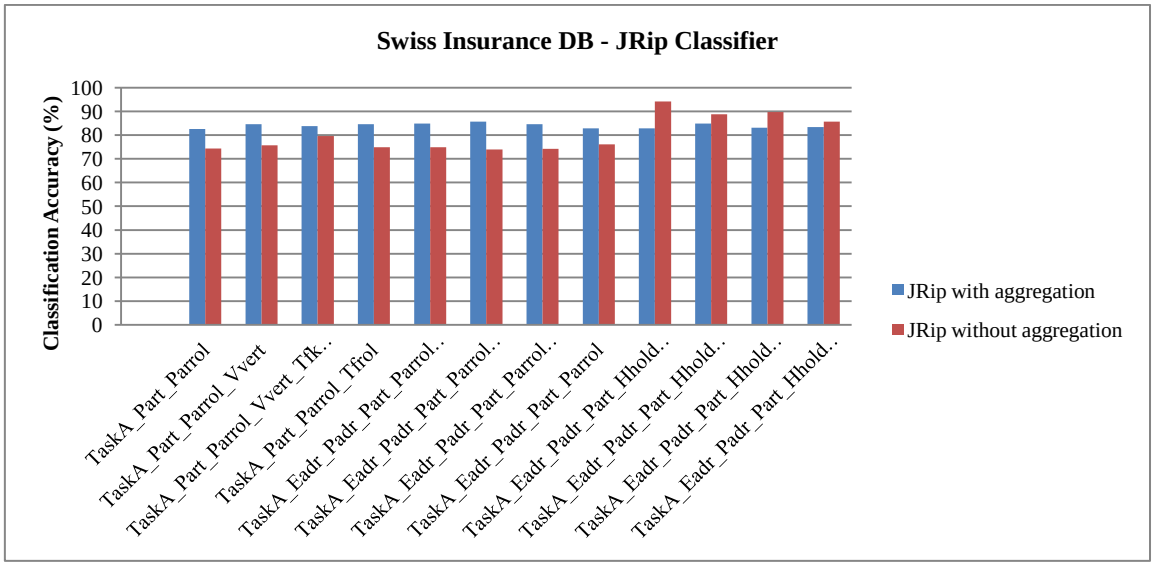


Figure 5.13: The comparison of classification accuracy of different target views (with and without aggregation) in the Swiss insurance database using JRip classifier

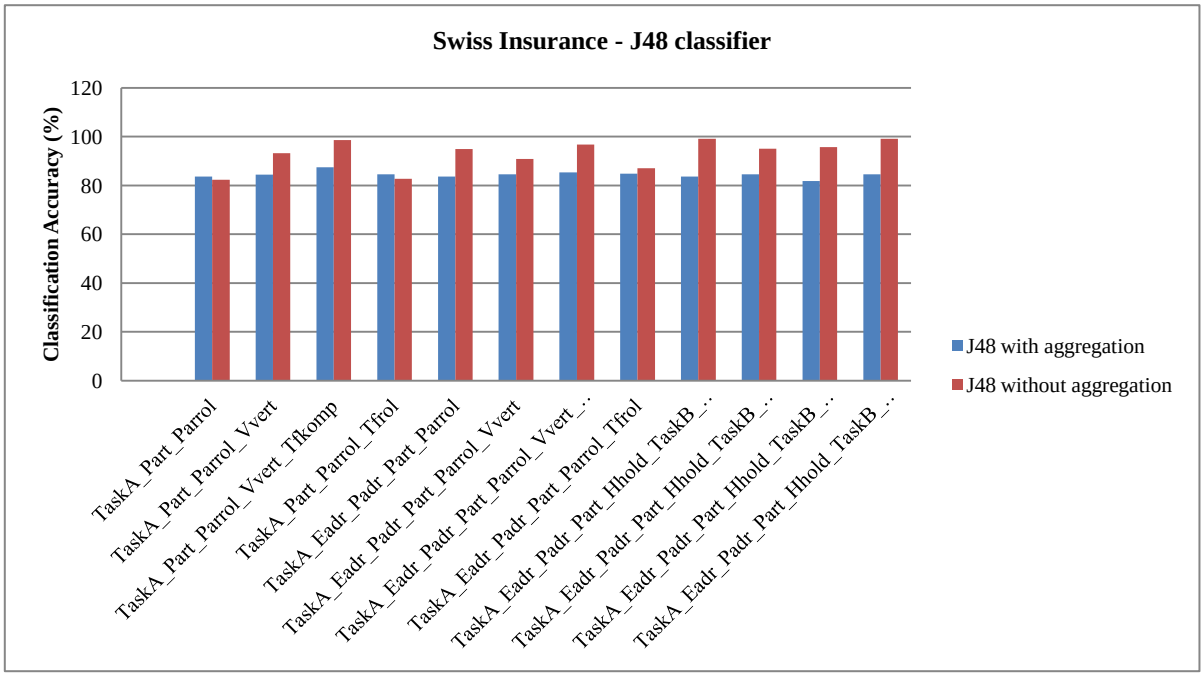


Figure 5.14: The comparison of classification accuracy of different target views (with and without aggregation) in the Swiss insurance database using J48 classifier

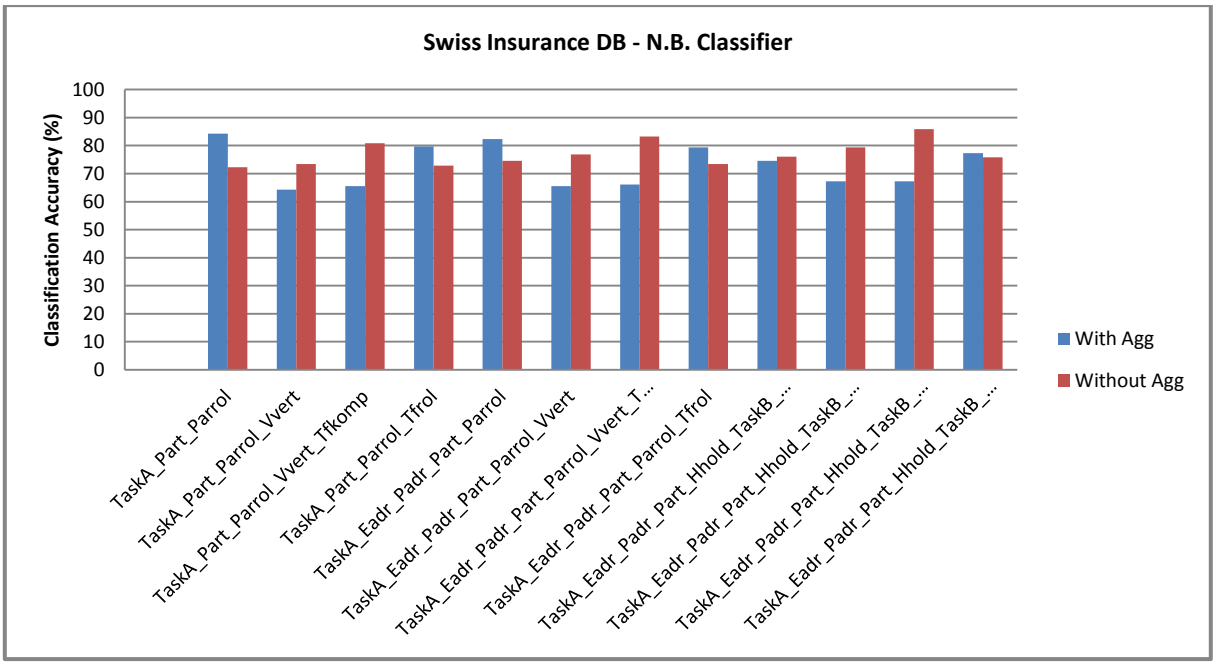


Figure 5.15: The comparison of classification accuracy of different target views (with and without aggregation) in the Swiss insurance database using N.B. classifier

In this database, unlike the Thrombosis database, we do not necessarily obtain better classification accuracies when the views are constructed without aggregation. Contrary to the Thrombosis database which only consists of categorical attributes, the Swiss insurance database has a combination of categorical and numerical attributes. It follows that, the relations existing in each view consist numerical and categorical attributes in various proportions. When these relations are linked, depending on their data type, the overall data type of the constructed views (with and without aggregation) is different.

Given the same classifier, some of the views have a better accuracy when they are built without aggregation and some others show the opposite behaviour. Similarly, different classifiers applied to the same view show different results. For example, consider the TaskA_Part_Parrol_Vvert view. JRip shows a better classification accuracy when this view is constructed with aggregation. However, both N.B. and J48 show better classification accuracy when the view is constructed without aggregation. Since we have the same input files, this difference is related to the way these classifiers handle different data types, NULL values, and number of attributes in the resulting flat files. Aggregation has an impact on all of these factors as it changes the number of records and the number (and type of) attributes.

When discussing the classification accuracies obtained for the Thrombosis database, we indicated the potential loss of information caused by applying aggregation on categorical data. In the Swiss insurance database, we notice that in some cases, aggregation, actually, increases the classification accuracy. We may argue that, if aggregation results in a loss of information, why do we (in some of the views) obtain better classification accuracies with aggregation? One possible reason is as follows. With aggregation, we inevitably lose some details. However, these details do not necessarily contain useful information when building a model. This is specially the case when we deal with numerical information. We lose details about individual records. However, instead, we obtain some other knowledge which is valuable when building the models. For instance, it is possible that, when aggregation is applied on numerical data, the count, sum, or average, may give us new insights. Recall that in data mining, we often are more interested in obtaining a global view (so-called knowledge nuggets) of the data, rather than a local view. It is also possible that some operators, such as

average, actually smooth the data and thus reduce the noise. Hence, it becomes easier to construct an accurate model with the aggregated data.

5.1.3 Loan DB – PKDD 1999 Discovery Challenge

Following the same procedure discussed in Section 5.1.1 and Section 5.1.2, our first task is to identify two classification targets. Recall from Section 4.3.1.3 that in this database, *loan_status* attribute in the Loan table is considered the target attribute and the *gender* attribute in the Client table is defined as the confidential attribute that needs to be protected.

The list of the extracted join paths with regard to the target class and the privacy class is shown in Figure 5.16.

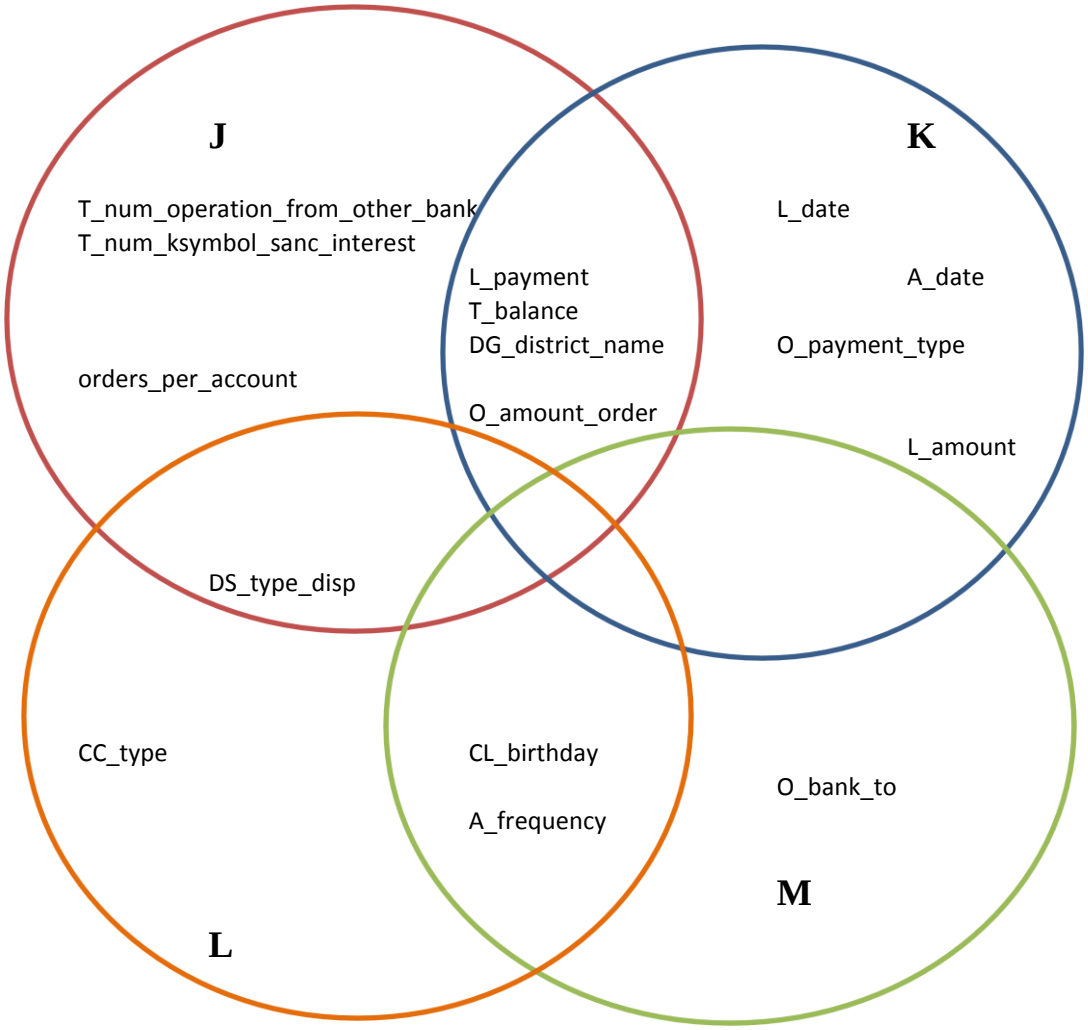
a) Join Paths (w.r.t. Target class)
L_T
L_T_A
L_T_A_DS
L_T_A_DS_CC
L_T_A_DS_CL
L_T_A_DG
L_A
L_A_DS
L_A_DS_CL
L_A_DS_CC
L_A_DG
L_A_DG_CL
L_O
L_O_T
L_O_T_A
L_O_T_A_DS
L_O_T_A_DS_CC
L_O_T_A_DG
L_O_T_A_DG_CL
L_O_A
L_O_A_DS
L_O_A_DG_CL
L_O_DS
L_O_DS_CL
L_O_DS_CC

b) Join Paths (w.r.t. Privacy class)
CL_DS
CL_DS_CC
CL_DS_A
CL_DS_A_O
CL_DS_A_T
CL_DS_T
CL_DS_O
CL_DS_A_T_O
CL_DS_CC_T
CL_DS_T_O
CL_DS_A_O_DG
CL_DS_A_O_T_DG
CL_DS_A_CC_DG_O_T

Figure 5.16: List of extracted join paths in the Loan database: (a) with regard to the target class and (b) with regard to the privacy class

After extracting the join paths, similar to the other two databases, the relations participating in one-to-many links are identified. In this database, two such relations are Transaction and Order. There are 4479 distinct account_IDs in the Transaction table while

the total number of transaction records is 52904. Similarly, the Order table contains 6471 records in total and only 3758 distinct account_IDs. Once more, using the CFS algorithm, we first obtain different sets of selected attributes that are highly correlated with the target class and the privacy class. Then, the logical relationships between these sets are identified and are represented in the Venn diagrams (Figure 5.17).



- Selected Attributes – Target Views – with aggregation (Group J)
- Selected Attributes – Target Views – without aggregation (Group K)
- Selected Attributes – Privacy Views – with aggregation (Group L)
- Selected Attributes – Privacy Views – without aggregation (Group M)

Figure 5.17: The Loan database Venn diagrams

The tabular representation of the Venn diagrams is presented in Figure 5.18.

All Selected Attributes	(Group J)	(Group K)	(Group L)	(Group M)
T_num_operation_from_other_bank	•			
T_num_ksymbol_sanc_interest	•			
DS_type_disp	•		•	
Orders_per_account	•			
L_payment	•	•		
T_balance	•	•		
DG_district_name	•	•		
O_amount_order	•	•		
A_frequency			•	•
CL_birthday			•	•
CC_type			•	
L_date		•		
A_date		•		
L_amount		•		
O_payment_type		•		
O_bank_to				•

Figure 5.18: Tabular representation of the selected attributes associated with the Loan database Venn diagrams

From the Venn diagrams, we are able to obtain a set of selected attributes that are highly correlated with the target class. Consider attributes *orders_per_account*, *T_num_ksymbol_sanc_interest*, *T_num_operation_from_other_bank*, and *DS_type_disp* in {J'}. These features are correlated with the target attribute only when the views are constructed using aggregation. There are other attributes such as *L_date*, *A_date*, *O_payment_type*, and *L_amount* in {K'} which are correlated with the target attribute only when the views are constructed without aggregation. We may use the same justification used for the Thrombosis database and the Swiss insurance database to explain this behaviour. A comparison between {L'} and {M'} once again shows that aggregation, indeed, changes the correlation between selected features and the classification target. Once more, we show that the lists of selected attributes in {J'} and {L'} include both the original and the relational features and both types of features can be used in order to predict the classification target.

In studying the impact of aggregation on the privacy breach of this database we are able to identify only one attribute, i.e. *DS_type_disp* in {J} ∩ {L}. This attribute is highly correlated with the target class and may lead to privacy breach. The overlapping region {J} ∩ {L} consists of a list of selected attributes returned when both the target views and the

privacy views are constructed using aggregation. Therefore, via this attribute, aggregation may negatively impact the privacy of this database.

Classification Accuracy Measurement

To see the effect of applying aggregation on the heuristic goodness of a selected feature subset, the value of "merit of best subset found" of different target views is compared. The results are shown in Figure 5.19.

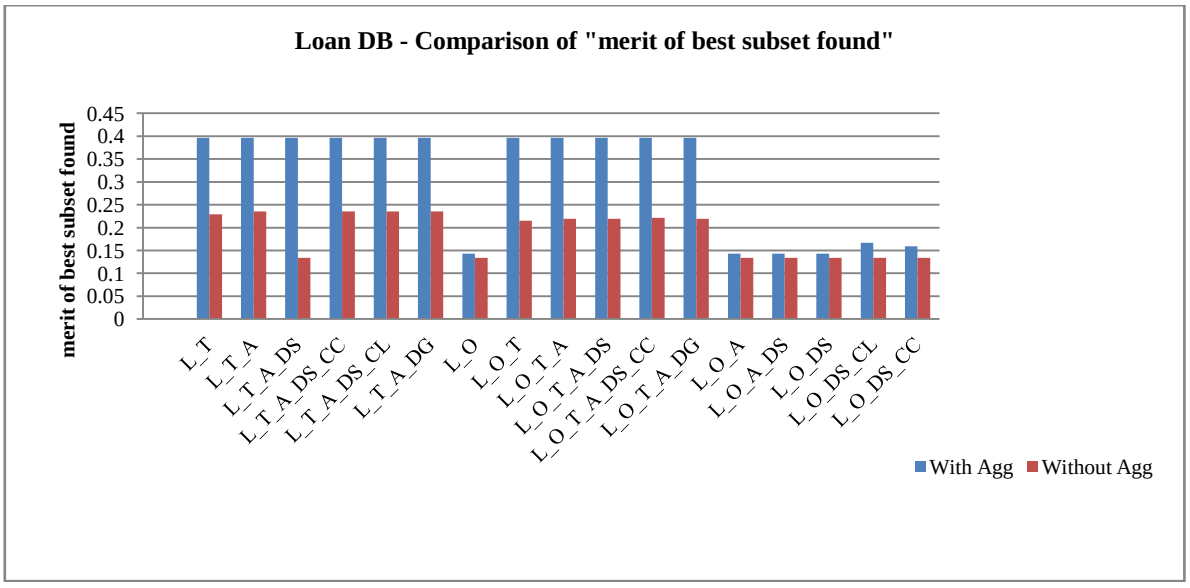


Figure 5.19: The comparison of "merit of best subset found" for different target views in the Loan database

Similar to the other two databases, with aggregation, we obtain better values of the "merit of best subset found" across different target views in the Loan database. To explain these results, once again, we refer to the impact of aggregation on the three factors in Equation 4.1; namely, $\overline{r_{cf}}$, $\overline{r_{ff}}$, and *k*. A higher "merit of best subset found" is obtained when we have a strong correlation between the features and the class ($\overline{r_{cf}}$), and weak inter-correlation in between features within the feature subset ($\overline{r_{ff}}$). Our results indicate that, with aggregation, we again obtain subsets of features that are strongly correlated with the classification target.

Next, the impact of aggregation on the classification accuracy and the model building time is investigated. The results are shown in Figures 5.20, 5.21, 5.22, and 5.23.

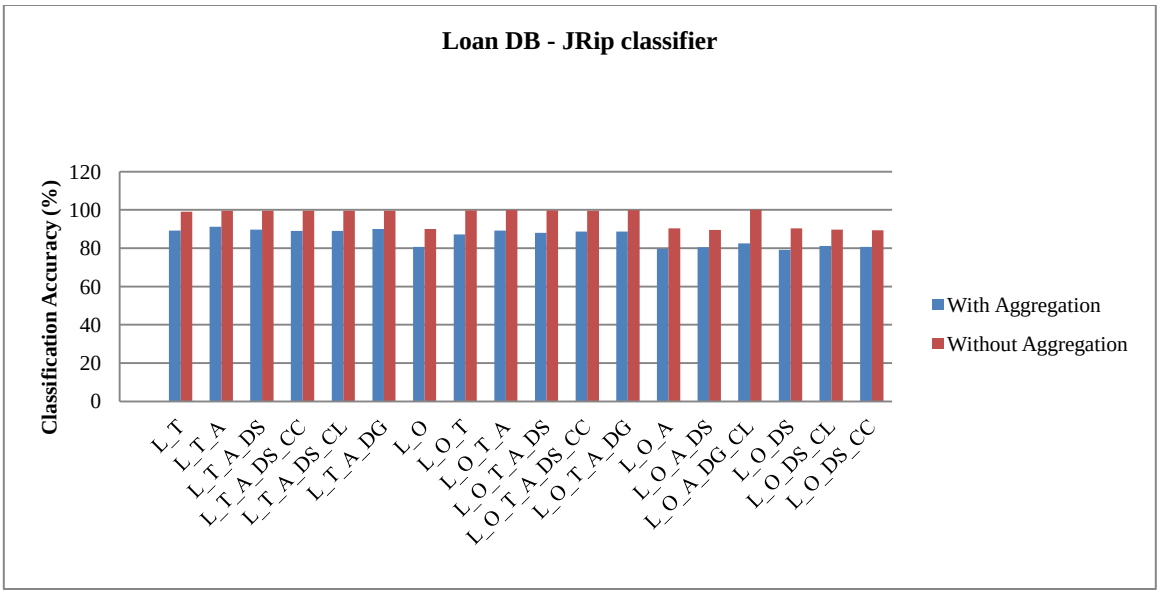


Figure 5.20: The comparison of classification accuracy of different target views (with and without aggregation) in the Loan database using JRip classifier

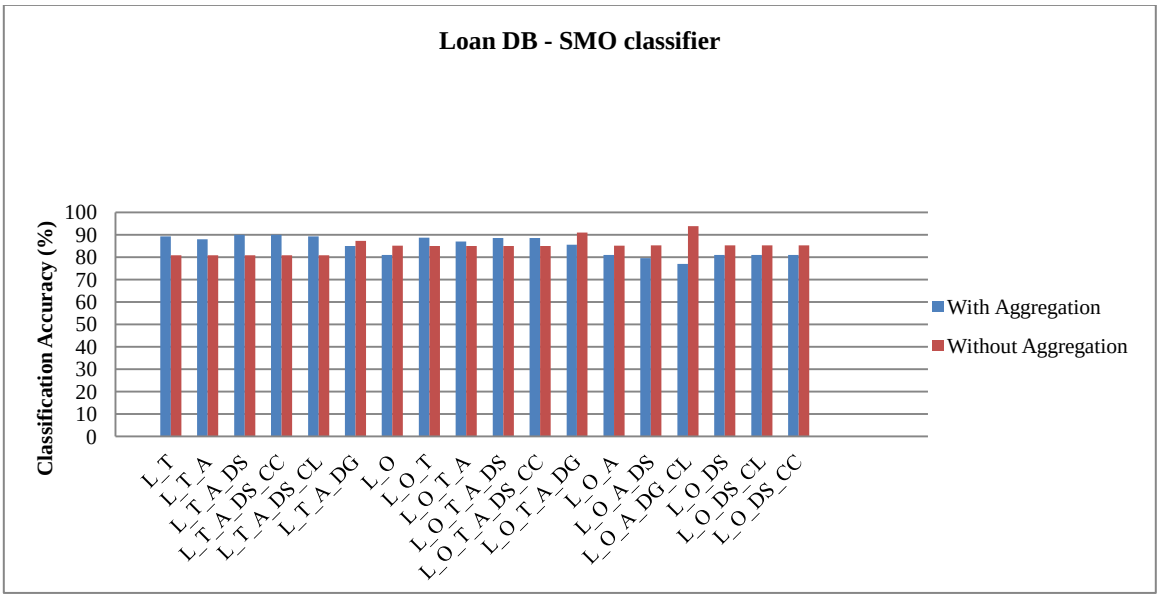


Figure 5.21: The comparison of classification accuracy of different target views (with and without aggregation) in the Loan database using SMO classifier

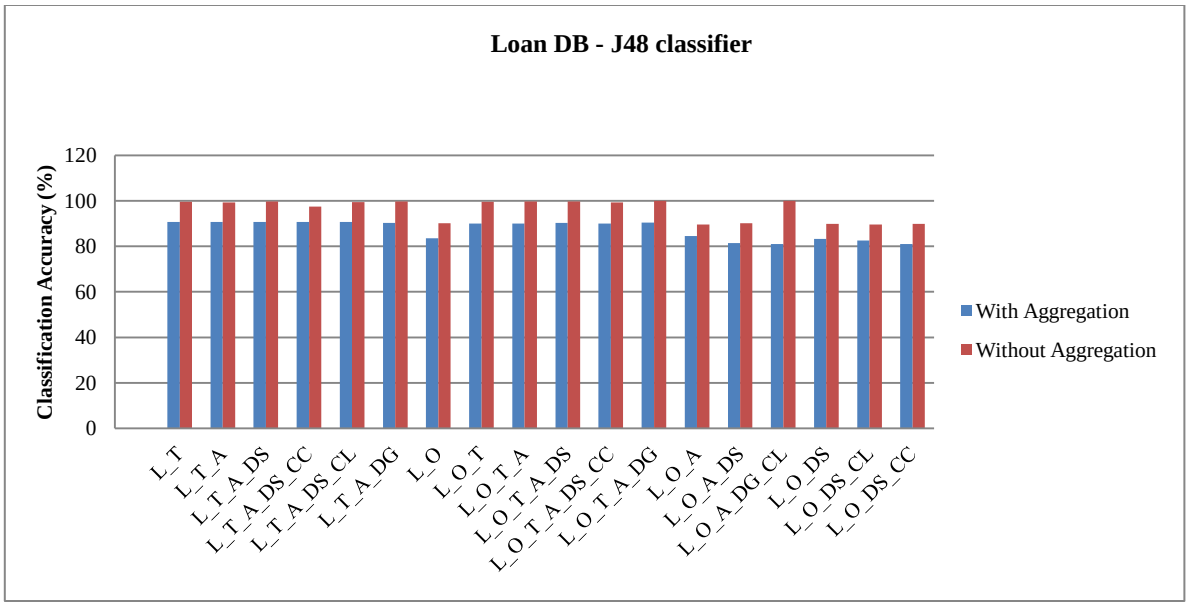


Figure 5.22: The comparison of classification accuracy of different target views (with and without aggregation) in the Loan database using J48 classifier

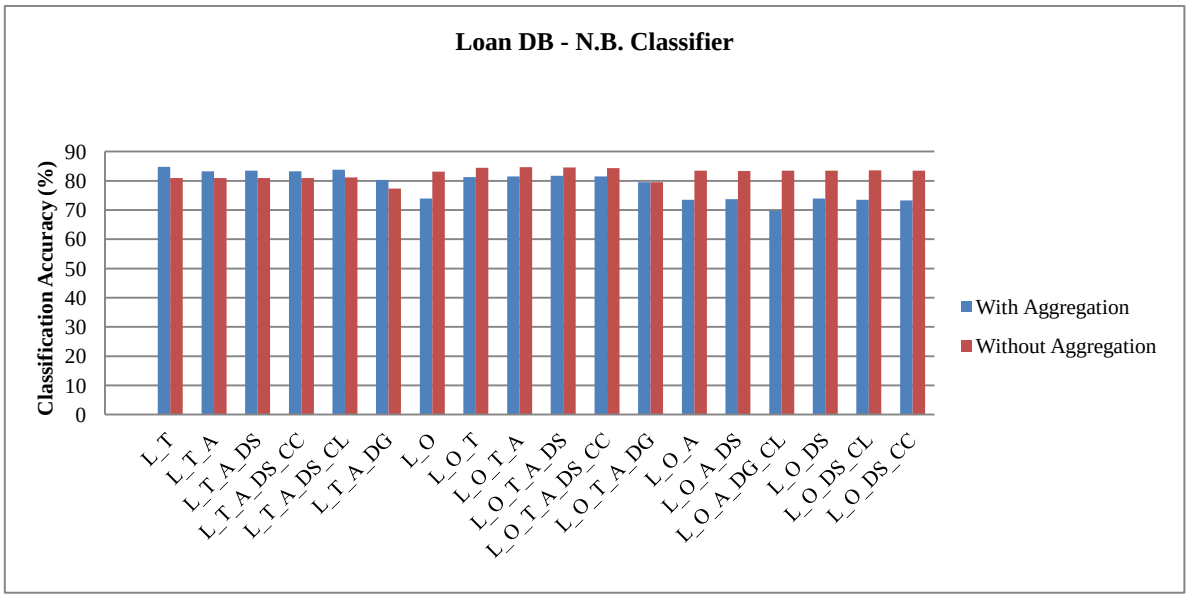


Figure 5.23: The comparison of classification accuracy of different target views (with and without aggregation) in the Loan database using N.B. classifier

An overview of these figures shows different behaviour than the other two databases. In this database, both the J48 and the JRip classifiers return better accuracies when the views are constructed without aggregation. The other two classifiers i.e. SMO and N.B. show

inconsistent results, i.e. some of the views constructed with aggregation result in a better classification accuracy, and vice versa.

By constructing relational features, aggregation changes the structure of the resulting datasets. Similar to the Swiss Insurance database, the tables in the Loan database have various proportions of numerical, binary and nominal values. Depending on the relations involved in a given join path, the proportion of numerical, nominal, and binary attributes in the resulting flat file is different and inconsistent results are obtained. For instance, consider two join paths such as L_O and L_T. In L_T, except the J48 and the JRip classifiers, the other two classifiers show a better classification accuracy when this view is built with aggregation. On the other hand, in L_O all classifiers show a better performance when L_O view is constructed without aggregation. These views have different proportions of nominal and numerical values as it is displayed in Table 5.4.

Table 5.4: Comparison of the number of attributes (L_O view versus L_T view)

	Nominal Attributes	**Numerical Attributes**
L_O (With Aggregation)	0	12
L_O (Without Aggregation)	1	5
L_T (With Aggregation)	0	32
L_T (Without Aggregation)	3	8

5.2 Discussion

In order to evaluate the PBIRD algorithm, we used three databases with different properties. The Thrombosis database consists of categorical attributes only, while the other two databases are mixed. In the Swiss Insurance database, the categorical attributes outnumber the numerical attributes, whereas in the Loan database, the numerical attributes outnumber the categorical attributes. Using databases with different overall data types was especially important for our research. We were able to study the impact of aggregation on the privacy, the classification accuracy, and the model building time as a function of data type.

After implementing the PBIRD algorithm, we were able to identify a list of features that are highly correlated with the target class and may lead to privacy breach in the examined relational databases. These features are listed in Table 5.5.

Table 5.5: List of potentially dangerous attributes in the Thrombosis DB, the Swiss Insurance DB, and the Loan DB (Common features are highlighted).

DB / Properties	With Aggregation	Without Aggregation
Thrombosis DB (Mostly Nominal Attributes)	**Potentially Dangerous Attributes: (J ∩ L)**	**Potentially Dangerous Attributes: (K ∩ M)**
	D_confirmed_count (Relational Att) *ANTI_EN_ANA* **D_diagnosis** **PN_firstdate** **ANAP_NAN_PA**	*THR_symptom* *PN_age* *D_from_test* *THR_examination_date* **D_diagnosis** **PN_firstdate** **ANAP_NAN_PA**
Swiss Insurance DB (Mixed Attributes Nominal > Numerical)	**Potentially Dangerous Attributes: (J ∩ L)**	**Potentially Dangerous Attributes: (K ∩ M)**
	Parrol_insurance_policy_count (Relational Att) *Tfkomp_insured_benefit_per_comp* **Vvert_single_premium** **Tfkomp_init_payment_reg_cont_termin**	*a15* *a20* *Eadr_adr_usage* *Vvert_annual_premium* **Vvert_single_premium** **Tfkomp_init_payment_reg_cont_termin**
Loan DB (Mixed Attributes Numerical > Nominal)	**Potentially Dangerous Attributes: (J ∩ L)**	**Potentially Dangerous Attributes: (K ∩ M)**
	DS_type_disp	*N/A*

Table 5.5 highlights the following points. Both with or without aggregation, we obtain a list of potentially harmful attributes. When aggregation is used, we obtain a set of harmful features and similarly, when aggregation is not used, we obtain another set of harmful features. Although some of the features commonly exist in both lists (highlighted), the remaining ones are different. To explain this variability, we should remind ourselves that these lists are obtained as a result of applying the CFS algorithm on the flat files constructed with and without aggregation. We frequently referred to the fact that, because of applying aggregation functions, the two flat files (constructed with and without aggregation) are structurally different. They have different number (and type of) of attributes, number of records, redundancy, and so on. Therefore, it is reasonable to get different lists of selected attributes.

Our results indicate that there is a link between aggregation and the privacy of a relational database. We illustrate that, there are dangerous attributes obtained only when the flat files are constructed with aggregation i.e. in (J ∩ L). We conclude that, we need both flat files constructed with and without aggregation in order to obtain a complete set of attributes that may become potentially harmful. By providing these lists, the PBIRD algorithm allows the designers/owners to implement necessary adjustments in order to protect the privacy of a database.

Recall that identifying all attribute interrelationships in relational databases is a difficult task. This is mainly due to the size and complexities of database schema which consists of multiple relations. In these databases, it is possible that seemingly harmless attributes may be linked to confidential information, leading to data leakage when building a model [2]. By obtaining the list of potentially dangerous attributes (as a result of applying the PBIRD algorithm), we, indeed, identify that list of seemingly harmless attributes which may potentially become harmful. After identifying these attributes, the database owner/designer may decide to remove (some of) them from the database, or may use one of the existing privacy preserving techniques (as discussed in Chapter 2) to modify these attributes. In either case, preserving the data utility and the validity of the data mining results should be taken into consideration. Recall from Section 2.2 that, data modification is considered one of the main approaches used in privacy preserving data mining. In general, modification of data is performed via anonymization [11].

To study this in more details, consider Table 5.5 which lists the potentially dangerous attributes corresponding to our three databases. We consider the Thrombosis database in particular. The following attributes constitute the complete set of attributes which may potentially be linked to confidential information and result in privacy violation. These attributes are *D_confirmed_count*, *ANTI_EN_ANA*, *D_diagnosis*, *PN_firstdate*, *ANAP_NAN_PA*, *THR_symptom*, *PN_age*, *D_from_test*, *THR_examination_date*. Recall that, these attributes are able to predict the patient gender, i.e. the target class, and the IgG concentration, i.e. the privacy class, with high accuracy. It follows that, having these attributes in the published (data mining) results may lead to disclosure of unwanted knowledge.

We may decide to apply one of the existing privacy preserving techniques in relational database mining such as multi-relational K-anonymity [31], or to publish a ranked list of subschema instead of publishing the full database [2]. Instead of protecting the privacy in the relational database directly, we may choose to employ the reconstruction-based technique after joining the relations. With this technique, as discussed in Section 2.4.3, firstly, we distort individual data values employing the perturbation method [1] or the k-anonymity model [88]. Then, we reconstruct the original distribution of the values of these attributes at an aggregated level. In this way, we may build a classification model which preserves the statistical information, while preventing the disclosure of confidential and potentially harmful attributes. Data distortion is achieved via adding noise, swapping values, blocking, sampling, amongst others, as discussed in details in Section 2.3.3. When selecting a particular perturbation technique, the type of the attributes (being numerical or. categorical) should be taken into consideration. Due to the absence of natural ordering in categorical values, the data perturbation techniques for categorical data and numerical data may be different [89].

In our example, (i.e. the Thrombosis database), by obtaining the list of potentially dangerous attributes, the database owner/designer may decide to perturb their values. For instance, since D_diagnosis, D_from_test, THR_symptom, amongst others, are categorical attributes, s/he may decide to use swapping, or to add noise using clustering among these values [89]. In the case of data swapping, values of individual records are interchanged.

Alternatively, instead of mining the anonymized source data, the database s/he may choose to anonymize the data mining patterns directly, as discussed in Section 2.6. In such solution, it is possible that, we obtain a better information utility compared with the previous option [29].

In this research, we have also shown that, multi-view learning is a useful technique to detect privacy breach in multi-relational databases. Compared to a universal flat file where all relations are joined together, in multi-view learning, each view consists only of a subset of relations/attributes. By focusing solely on a subset of attributes that are presented in each view, the role of these attributes in predicting the classification target becomes especially important. Therefore, we ensured that none of the attributes that are correlated with the classification target is ignored.

The impact of aggregation on the classification accuracy of the models is discussed next. Table 5.6 shows the average accuracy of all target views constructed, with and without aggregation.

The results show that when the database consists of only categorical data (e.g. the Thrombosis database), aggregation has a negative impact on the classification accuracy of the resulting models and should be used carefully. In such database, when the flat files are constructed without aggregation, no discretization is required and the classification algorithms can be applied directly. However, when the flat files are constructed with aggregation, we change the data type and obtain numerical datasets. It is likely that by applying aggregation on categorical data, information is lost and hence, the accuracy across all join paths is reduced.

Table 5.6: Comparison of classification accuracies of all databases using different classifiers with/without aggregation (for each classifier, the higher accuracy within a pair (with vs. without aggregation) is highlighted)

Database	Property	Number of Join Paths	J48 With Agg	J48 Without Agg	JRip With Agg	JRip Without Agg	SMO With Agg	SMO Without Agg	N.B. With Agg	N.B. Without Agg	Total number of numerical attributes	Total number of categorical attributes
Loan	Mixed (Mostly numerical)	16	87.3%	**96.3%**	85.8%	**96.4%**	**85.1%**	84.8%	78.6%	**82.5%**	23	12
Thrombosis	Completely nominal	17	73.8%	**79.5%**	73.7%	**78.8%**	74.7%	**80.5%**	58.6%	**76.9%**	0	21
Swiss Insurance	Mixed (Mostly nominal)	12	84.3%	**92.9%**	**83.9%**	80.1%	N/A	N/A	72.8%	**77.1%**	19	69

When the database is mixed (e.g. the Loan database and the Swiss Insurance database), the accuracies of the models with and without aggregation are comparable. However even in such a scenario, the results without aggregation seem to outperform.

A comparison of the average time required to build the classification models for the views constructed, with and without aggregation, is displayed in Table 5.7. In general, with aggregation, a shorter model building time is obtained. This is expected; since the flat files constructed with aggregation consist of substantially less records and therefore, less computational effort is required to build the models.

However, one of our databases, i.e. the Thrombosis database, shows different results. In this database, three classifiers (i.e. JRip, N.B., and J48) show a slightly shorter model building time when the flat files are constructed without aggregation. This is counter-

intuitive. The reason being that, the Thrombosis database is a relatively small database (in terms of number of records in the joined relations) and includes nominal attributes which have high cardinality. This high cardinality results in a large number of relational features in the flat files constructed with aggregation. As a result, due to the small size of this database, the impact of aggregation in reducing the number of records in the result sets is not substantial.

Table 5.7: Comparison of the time required to build the models using different classifiers with/without aggregation (for each classifier, the lower time within a pair (with vs. without aggregation) is highlighted)

Database	Property	J48 With Agg	J48 Without Agg	JRip With Agg	JRip Without Agg	SMO With Agg	SMO Without Agg	N.B. With Agg	N.B. Without Agg
Loan	Mixed (Mostly numerical)	**0.04s**	1.28s	**0.07s**	15.9s	**0.12s**	57.3s	**0.009s**	0.04s
Thrombosis	Completely nominal	0.09s	**0.02s**	**0.17s**	**0.07s**	**0.22s**	0.49s	**0.01s**	**0.001s**
Swiss Insurance	Mixed (Mostly nominal)	**3.67s**	19.00s	**19.08s**	13611s	N/A	N/A	**0.37s**	1.62s

The different behaviour of SMO (i.e. WEKA's implementation of SVM) is explained as follows. The SVM algorithm works only on numerical data. In order to handle categorical attributes, they need to be transformed into a set of binary attributes, one per category value. SMO implicitly performs this transformation. Since the Thrombosis database is completely categorical, it needs to be fully transformed in order to apply the algorithm and this transformation consumes time. On the other hand, due to the role of aggregation in changing the data type of this database, the flat files constructed with aggregation are numerical, no transformation is needed, and faster results are obtained.

It should be pointed out that, even though with aggregation a shorter model building time is obtained, a considerable pre-processing effort is required. Recall from Section 3.8.5 that in statistical relational learning, aggregation itself is regarded as a pre-processing step [67]. In studying the time required to build the models, these pre-processing efforts should be taken into consideration, as well.

Finally, when studying the role of aggregation, a trade-off between privacy, classification accuracy and time should be taken into consideration. Our research helps to achieve separation of concerns for the database owner/designer where it is applicable. With the

methods proposed in our research, the database owner/designer is, firstly, provided with the comprehensive list of potentially dangerous attributes. By identifying this list, s/he ensures that, no potentially harmful attribute is overlooked. In other words, by capturing the potentially dangerous attributes, the focus may be shifted towards the classification accuracy and the model building time. In such way, the database owner/designer should be less concerned that, by selecting one method over the other (i.e. aggregating over not aggregation and vice versa), some potentially harmful attributes may be ignored leading to privacy leakage.

5.3 Summary

In this chapter, we have evaluated the PBIRD algorithm on three different databases in order to investigate the possible link between aggregation and privacy breach in multi-relational databases. When a join path consists of one-to-many links, aggregation operators are employed to aggregate information from the multiset. Then, relational features are constructed in order to store these aggregated information.

The results obtained by the PBIRD algorithm showed that aggregation functions do, indeed, change the correlation between the selected features (returned by the CFS algorithm) and the classification target. We were able to identify a number of attributes that are highly correlated with (predictive of) the target class which may lead to privacy breach in the multi-relational database when the views were constructed using aggregation. Similarly, we were able to identify another list of harmful attributes obtained when the views were constructed without aggregation. We showed that these two lists are complementary, and a database designer needs both list in order to provide maximal privacy protection of the relational database.

The effect of applying aggregation on the heuristic goodness of a feature set was also investigated. Our results showed that, the flat files constructed with aggregation return a better values of the “merit of best subset found” across all examined databases.

We then studied the impact of aggregation on the classification accuracy and the time required to build the models. Our results showed that when the database consists of categorical attributes only, we obtain better classification accuracies across all views constructed without aggregation. However, when the database is mixed, the accuracies of

the flat files constructed with and without aggregation are comparable. For a mixed database, the extracted join paths consist of different relations. On the other hand, these relations have various proportions of categorical and numerical attributes. It follows that, the views constructed with and without aggregation have different overall data types and the classifiers return inconsistent results.

Finally, the impact of aggregation on the time required to build the models was investigated. The results showed that in general, the size of flat files constructed with aggregation is smaller and therefore, a shorter time to build the model is required.

Chapter 6

Conclusion

Privacy preserving in multi-relational database mining is an emerging research area. The challenges in this area originate, on one hand, from already existing challenges in privacy preserving data mining, and on the other hand, from multi-relational database mining.

Most of the conventional data mining algorithms are meant to be applied to a single table. To perform data mining tasks on multi-relational databases, different techniques have been introduced in the past few years. These techniques either upgrade the existing data mining algorithms in order to apply them on a relational database directly, or 'flatten' the relational database, i.e. convert it into flat file(s) suitable for traditional data mining algorithms [33]. The process of flattening usually involves aggregation and constructing new attributes.

With regard to privacy preserving in multi-relational databases, the very nature of relational database introduces new challenges. A recent study [2] showed that identifying all possible relationships between the attributes in different relations is difficult. Further, although some of those attributes look harmless, they may be linked to confidential data, leading to data leakage. That study further showed that even distortion or elimination of the confidential attributes may not be enough to prevent such privacy leakage.

Considering the above-mentioned concepts and challenges, this thesis investigated the role of aggregation in two areas, namely, its impact on privacy and the effects of aggregation

on the classification accuracy in multi-relational database mining. Our two major contributions presented in this work are summarized below.

6.1 Thesis Contribution

Firstly, our study introduces a new method to investigate privacy breach in relational databases. The PBIRD algorithm combines two relatively new techniques i.e. multi-view learning [4] and Correlation-based Features Selection [5]. The PBIRD algorithm shows that there is a link between aggregation and privacy breach in relational databases. In this research, we aimed to study the impact of aggregation on the privacy of a relational database. The results showed that, aggregation plays an important role in identifying potentially harmful attributes which could have not been identified if the relations were to be joined without aggregation. The results of the PBIRD algorithm showed that aggregation does, potentially, introduce new privacy violations. However, the potentially harmful attributes obtained with aggregation were different than the ones obtained when the relational database is not aggregated. Therefore, aggregation provides us with new insights and valuable information about these attributes. We showed that even when we enforce privacy on non-aggregated data, this may not be enough. This is due to the fact that, if the same relations are joined using aggregation new potentially dangerous attributes may be identified. When considering the broader view, the PBIRD algorithm provides us with a complete set of potentially dangerous attributes (obtained with and without aggregation) which may lead to privacy breaches in a relational database. By providing this information to the database designers/owners, they can be warned of the list of harmful features that may be linked to private information. Therefore, necessary adjustments can be put into place in order to protect the database. For example, the database owner/designer may decide to remove these features, or to apply one of the privacy preserving techniques on them, and so on.

Secondly, in our study we investigated the impact of aggregation on the classification accuracy and the time required to build the models. Our results showed that, when a database consists only of categorical data, aggregation decreases the overall accuracies of the resulting models. On the other hand, when the database is mixed, the accuracies with and

without aggregation are comparable. However, even in such scenario, the accuracy obtained without aggregation is slightly better.

With regard to the impact of aggregation on the time required to build the models, our results showed that, in general, with aggregation a faster model building time is obtained. However, when the database is small and consists of categorical attributes with high cardinality, with aggregation a slower model building time is obtained. Our results lead us to the following generalizations.

1- When the database is completely categorical and has a small size aggregation is not recommended, since it negatively impacts both the classification accuracy and the time required to build the models.
2- When the database is completely categorical and is large, given that without aggregation always results in a better accuracy, a trade off between accuracy and time should be taken into consideration.
3- When the database is mixed, since the classification accuracies (with and without aggregation) are comparable, aggregation is preferable, because it substantially improves the model building time. It is true that with aggregation, more pre-processing effort is required, but with large databases this overhead may be ignored.

For the comparative study, we were expecting to see a reduction in the classification accuracies and a shorter model building time across all views constructed with aggregation. This assumption was made, mainly, because of the role of aggregation in summarizing data. However, we found that, in some views, better classification accuracies were obtained with aggregation: especially when applied on numerical attributes. Similarly, we obtained unexpected results when we investigated the impact of aggregation on the time required to build the model. In general, by summarizing the data, aggregation eliminates redundancies, reduces the number of records in the resulting flat file, and results in a smaller dataset. This should eventually lead to less computational efforts and time required to build the classification model. To this end, we were expecting to obtain a shorter time with aggregation across all join paths. However, for the Thrombosis database, we observed different (and counter-intuitive) results. By investigating the nature of this database, we found that this database is small and consists of categorical data (with high cardinality). Because of the small size of this database, the impact of aggregation on reducing the number

of records in the resulting dataset is not significant. However, by constructing relational features, aggregation largely increases the number of attributes in the resulted dataset. Therefore, essentially with aggregation, a slightly larger dataset is obtained which explains the obtained results.

6.2 Future Work

Our future work includes testing the PBIRD algorithm on different databases with complex schema and on databases that include multiple confidential attributes residing in the same relation or in different relations. We will also test this algorithm in the case where the target attribute and the privacy attribute co-exist in the same relation.

It follows that, with different feature selection algorithms, the obtained set of dangerous attributes might differ. An interesting future direction would be to investigate this issue further. There are two main categories of feature selection algorithms, namely, the *filter model* and the *wrapper model* [68; 82; 70]. The filter model uses general characteristics of the data in order to evaluate attributes. The wrapper model uses a target learning algorithm in order to estimate the worth of attribute subsets. The algorithms are further divided into those which evaluate individual attributes and those which evaluate subsets of features [68]. Because wrappers are computationally expensive, with large number of attributes (which is usually the case when we deal with relation databases), the filter model is preferred [82]. Recall that, in this work, we used the CFS algorithm to implement the PBIRD algorithm. CFS belongs to the category of the filter model feature selection algorithms. This algorithm uses a subset evaluation heuristic which takes into account the usefulness of individual features for predicting the class [68]. It would be worthwhile to conduct a comparative study of different categories of feature selection algorithms. With such a study, we will be able to investigate the impact of different feature subset selection techniques on the privacy of the relational database. This comparative study would include comparing different algorithms within the filter model category, algorithms in the wrapper model category, algorithms which evaluate individual attributes, or those which evaluate subset of features.

Another idea is to automate the process, i.e. to build a system to apply the PBIRD algorithm automatically. This system will take the relational database, the target attribute, and the privacy attribute as inputs, then applies the algorithm, and finally returns the list of

attributes which may lead to privacy breach. Using this system, the database designer is able to identify the potentially dangerous attributes and change the schema to minimize the privacy breach in the database accordingly. This system can also be used to monitor the privacy of the database over time. Our research shows that aggregation changes the list of selected attributes that are highly correlated with the classification target (target class and privacy class). In real databases, records are added, removed, or modified over time. This may change the list of selected attributes returned by the CFS algorithm and thus, change the list of attributes in the overlapping regions of the Venn diagrams. It follows that with time, due to the change in the dataset, some of the potentially dangerous attributes become harmless while some of the harmless attributes may potentially become harmful. Therefore, this system can be used by the database owner to regularly monitor the relational database. This is closely related to concept drift [81] which refers to the fact that in the real world concepts are changing over time. Therefore, the model built on old data becomes inconsistent with the new data which requires that the model be regularly updated.

Another research direction is to use the PBIRD algorithm to investigate if there is any link between privacy breach and the type of attributes in a relational database. If such a link is found, the database designer will be advised (when it is applicable) to choose the appropriate data type that minimizes data leakage when he/she designs the database.

Finally, in a recent study by Paquet et. al [83], it was shown that making the implicit assumption of having Gaussian data distribution (and that the standard limit theorem holds), is not always correct. Therefore, it is important to consider the distribution of data during any data mining exercise. This study argues that, during any data mining exercise, data with a Lévy distribution should be handled with caution, especially during data pre-processing and aggregation. To this end, a future direction is to apply the PBIRD algorithm on databases with different data distributions and to see how this may impact the results.

Bibliography

[1] **R. Agrawal and R. Srikant.** *Privacy-preserving data mining.* SIGMOD Record (ACM Special Interest Group on Management of Data), Vol. 29, No. 2, pages 439-450, 2000.

[2] **H. Guo, H. L. Viktor, and Eric Paquet.** *Privacy disclosure and preservation in learning with multi-relational databases.* Journal of Computing Science and Engineering, Vol. 5, No. 3, pages 183-196, 2011.

[3] **V. Torra, G. Navarro-Arribas, and D. Arbil.** *On the applications of aggregation operators in data privacy.* Integrated Uncertainty Management and Applications, AISC 68, pages 479-488, Berlin, Heidelberg, Springer-Verlag, 2010.

[4] **H. Guo and H. L. Viktor.** *Mining relational databases with multi-view learning.* Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, pages 15-24, IL, 2005.

[5] **M. Hall.** *Correlation-based feature selection for machine learning*. PhD Dissertation, Waikato University, Hamilton, New Zealand, 1998.

[6] **J. Han and M. Kamber.** *Data mining: concepts and techniques*. 2nd Edition. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[7] **V. Verykios, E. Bertino, I. N. Favino, L. P. Provenza, Y. Sayging, and Y. Theodoridis.** *State-of-the-art in privacy preserving data mining.* SIGMOD Record, Vol. 33, No. 1, 2004.

[8] **E. Bertino, D. Lin, and W. Jiang.** *A survey of quantification of privacy preserving data mining algorithms.* Privacy Preserving Data Mining. US : Springer, 2008, pages183-205.

[9] **T. Dalenius.** *Towards a methodology for statistical disclosure control.* Statistik Tidskrift. Vol. 15, pages 429-444, 1977.

[10] **C. Dwork.** *Differential privacy.* Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, pages 1-12, 2006.

[11] **B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu.** *Privacy preserving data publishing: a survey of recent developments.* ACM Computing Surveys, Vol. 42, No. 4, article 14, 2010.

[12] **A. Machanavajjhala, D. Kiffer, J. Gehrke, and M.Venkitasubramaniam.** *l-diversity: privacy beyond k-anonymity.* ACM Transactions on Knowledge Discovery from Data, Vol.1, No. 1, article 3, 2007.

[13] **J. Wang, Y. Luo, Y. Zhao, and J. Le.** *A survey on privacy preserving data mining.* Proceedings of the 1st International Workshop on Database Technology and Applications, pages 111-114, 2009.

[14] **P. Samarati and L. Sweeney.** *Generalizing data to provide anonymity when disclosing information.* Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART (PODS), New York, 1998.

[15] **P. Samarati and L. Sweeney.** *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.* Tech.rep., SRI International, 1998.

[16] **K. El Emam and F. Dankar.** *Protecting privacy using k-anonymity.* The Journal of the American Medical Information Association, Iss. 15, pages 627-637, 2008.

[17] **L. H. Cox.** *Suppression methodology and statistical disclosure control.* Journal of the American Statistical Association., Vol. 75, No. 370, pages 377-385, 1980.

[18] **K. Leferve, D. J. Dewitt, and R. Ramakrishnan.** *Incognito: efficient full-domain k-anonymity.* In Proceedings of ACM SIGMOD, ACM, New York, pages 49–60, 2005.

[19] **K. Lefervre, D. Dewitt, and R. Ramakrishnan.** *Mondrian multidimensional k-anonymity.* Proceedings of the 22nd IEEE International Conference on Data Engineering, 2006.

[20] **K. Lefervre, D. Dewitt, and R. Ramakrishnan.** *Workload-aware anonymization.* Proceedings of the 12th ACM SIGKDD, New York, 2006.

[21] **K. Wang, B. C. M. Fung, and P.S. Yu.** *Handicapping attacker's confidence: an alternative to k-anonymization.* Knowl. Inform. Syst. Vol. 11, No. 3, pages 345–368, 2007.

[22] **X. Xiao and Y. Tao.** *Anatomy: simple and effective privacy preservat*ion. Proceedings of the 32nd Conference on Very Large Data Bases (VLDB), 2006.

[23] **Q. Zhang, N. Koudas, D. Srivastava, and T. Yu.** *Aggregate query answering on anonymized tables.* Proceedings of the 23rd IEEE International Conference on Data Engineering, 2007.

[24] **J. Domingo-Ferrer,** *Privacy-preserving data mining: models and algorithms.* Berlin: Springer, 2008, pages 53-80.

[25] **A. Evfimievski, R. Srikant, R. Agrawal and J. Gehrke.** *Privacy preserving mining of association rules.* Proceedings of the 8th ACM SIGKDD, pages 217–228, New York, 2002.

[26] **K. Wang, B. C. M. Fung, and P. S. Yu.** *Template-based privacy preservation in classification problems.* Proceedings of the 5th IEEE International Conference on Data Mining (ICDM), pages 466–473, 2005.

[27] **G. Aggrawal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu.** *Achieving anonymity via clustering*. Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART(PODS) Conference, New York, 2006.

[28] **V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni.** *Association rule hiding.* IEEE Transactions on Knowledge and Data Engineering. Vol. 16, No. 4, pages 434–447, 2004.

[29] **M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi.** *Anonymity preserving pattern discovery.* International Journal on Very Large Data Bases Vol. 17, No. 4, pages 703–727, 2008.

[30] **M. Kantarcioglu, J. Jin, and C. Clifton.** *When do data mining results violate privacy?* In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and DataMining, pages 599–604, New York, 2004,.

[31] **M. E. Nergiz, C. Clifton, and A. E. Nergiz.** *Multirelational k-anonymity.* Proceedings of the 23rd International Conference on Data Engineering, pages 1417–1421, 2007.

[32] **H. Xiong, M. Steinbach, and V. Kumar.** *Privacy leakage in multi-relational databases via pattern based semi-supervised learning.* ACM 14th Conference on Information and Knowledge Management, Bremen, Germany, 2005.

[33] **H. Guo.** *Learning from multirelational data through multiple views.* PhD Dissertation, University of Ottawa,Ottawa, Canada, 2007.

[34] **S. Dzeroski.** *Multi-relational data mining: an introduction.* ACM SIGKDD Explorations, Vol. 5, No. 1, pages 1-16, 2003.

[35] **S. Muggleton.** *Inductive Logic Programming*. MIT Press. 1999.

[36] **S. Muggleton and L. D. Raedt.** *Inductive Logic Programming: theory and methods.* Journal of Logic Programming, Vol. 19, No. 20, pages 629–679, 1994.

[37] **J. R. Quinlan and R. M. Cameron-Jones.** *Foil: A midterm report*. Proceedings of the European Conference on Machine Learning, pages 3–20, 1993.

[38] **P. Clark and T. Niblett.** *The CN2 induction algorithm.* Mach. Learn., Vol. 3, No. 4, pages 261–283, 1989.

[39] **M. Pazzani and D. Kibler.** *The utility of knowledge in inductive learning.* Mach. Learn., Vol. 9, No.1, pages 57-94,1992.

[40] **X. Yin, J. Han, J. Yang, and P. S. Yu.** *CrossMine: Efficient classification across multiple database relations.* Proceedings of the 20th International Conference on Data Engineering, Boston, 2004.

[41] **A. J. Knobbe, M. Hass, and A. Siebes.** *Propositionalization and aggregates.* Lecture Notes in Computer Science, Vol. 2168, pages 277-288, Springer, Heidelberg, 2001.

[42] **M.-A. Krogel, S. Rawles, F.Zelezny, P. A. Flach, N. Lavrac, and S. Wrobel.** *Comparative evalution of approaches to propositionalization.* Proceedings of the 13th International Conference on Inductive Logic Programming, pages 197–214, 2003.

[43] **N. Lavrac.** *Principles of knowledge acquisition in expert systems.* PhD Dissertation, Faculty of Technical Sciences, University of Maribor, 1990.

[44] **M.-A. Krogel.** *On propositionalization for knowledge discovery in relatinal databases.* PhD Dissertation, The Faculty of Computer Science, Otto-von-Guericke-Universitt Magdeburg, 2005.

[45] **M. -A. Krogel and S.Wrobel.** *Transformation-based learning using multi-relational aggregation.* Proceedings of the 11th International Conference on Inductive Logic Programming, pages 142-155, 2001.

[46] **B. Pfahringer and G. Holmes.** *Propositionalization through stochastic discrimination.* In Proceedings of the Work-in-Progress Track at the 13th International Conference on Inductive Logic Programming, pages 60-68, 2003.

[47] **H. Blockeel, L. D. Raedt, N. Jacobs, and B. Demoen.** *Scaling up inductive logic programming by learning from interpretations.* Data Mining and Knowledge Discovery, Vol. 3, No. 1, pages 59-93, 1999.

[48] **A. Blum and T. Mitchell.** *Combining labeled and unlabeled data with co-training* . Proceedings of the Workshop on Computational Learning Theory, pages 92-100, 1998.

[49] **V. R. Sa. and D. H. Ballard.** *Category learning through multimodality sensing.* Neural Computation, Vol. 10, No. 5, pages 1097-1117, 1998.

[50] **I. A. Muslea.** *Active learning with multiple views.* PhD Dissertation, Department of Computer Science, University of Southern California, 2002.

[51] **K. Sridharan and S. M. Kakade.** *An information theoretic framework for multi-view learning.* Proceedings of the 21st Annual Conference on Learning Theory, Helsinki, Finland, pages 403-414, 2008.

[52] **P. K. Chan and S. J. Stolfo.** *Experiments on multistratege learning by meta-learning.* Proceedings of the 2nd International Conference on Information and Knowledge Management, pages 314-323, New York, 1993.

[53] **L. Cabibbo and R. Torlone.** *A framework for investigation of aggregate functions in database queries.* Proceedings of the 7th International Conference on Database Theory, pages 383-397, London, UK, 1999.

[54] **H.-J. Lenz and Bernhard Thalheim.** *OLAP databases and aggregation functions.* Proceedings of the 13th International Conference Scientific and Statistical Database Managment, pages 91-100 , 2001.

[55] **M. Detyniecki.** *Mathematical aggregation operators and their application to video querying.* PhD Dissertation, l'Universite Paris VI, Paris, 2000.

[56] **A. Knobbe, A. Siebes, and B. Marseille.** *Involving aggregate functions in multi-relational search.* Lecture Notes in Computer Science, Vol. 2431, pages 145-168, Springer, Heidelberg, 2002.

[57] **M. Detyniecki.** *Fundamentals on aggregation operators*. Proceedings of International Summer school on Aggregation Operators and their Applications '2001, Asturias, 2001.

[58] **C. Perlich and F. J. Provost.** *Aggregation-based feature invention and relational concept classes.* Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 167-176, 2003.

[59] **J. Gray, S. Chudhuri, A. Bosworth, A. Layman, D. Reichart, and M. Venkatrao.** *Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals.*Data Mining and Knowledge Discovery Vol. 1, No. 1, pages 29-53,1997.

[60] **A. J. Knobbe, H. Blockeel, A. Siebes, and D. Van Der Wallen.** *Multi-relational data mining.* Proceedings of the 9th BelgianDucth Conference on Machine Learning, 1999.

[61] **J. Domingo-Ferrer, J. M. Mateo-Sanz.** *Practical data-oriented microaggregation for statistical disclosure control*. IEEE Transactions on Knowledge and Data Engineering. Vol. 14, No. 1, pages 189-201, 2002.

[62] **G. Sande.** *Exact and approximate methods for data directed microaggregation in one or more dimensions.* International Journal of Uncertainty, Fuzziness and Knowledge Based Sytems, Vol. 10, No. 5, pages 459-476, 2002.

[63] **V. Torra.** *Microaggregation for categorical variables: a median based approach.* In J. Domingo-Ferrer, V. Torra (eds.) Privacy in Statistical Databases 2004. Lecture Notes in Computer Science, Vol. 3050, pages 162–174, Springer, Heidelberg, 2004.

[64] **G. Navarro-Arribas and V. Torra.** *Towards microaggregation of log files for Web usage mining in B2C e-commerce.* Proceeding of the 28th North American Fuzzy Information Processing Society Annual Conference Conference, pages 1-6 , 2009.

[65] **J. Nin and V. Torra.** *Extending microaggregation procedures for time series protection.* In S. Greco, Y. Hata, S.Hirano, M. Inuiguchi, S. Miyamoto, H. S. Nguyen, R. Słowi´nski (eds) Rough Sets and Current Trends in Computing 2006, Lecture Notes in Computer Science, Vol. 4259, pages 899–908. Springer, Heidelberg, 2006.

[66] **A. Valls, J. Nin, and V. Torra.** *On the use of aggregation operators for location privacy.* Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference, pages 489-494, 2009.

[67] **C. Perlich and F. Provost.** *Distribution-based aggregation for relational learning with identifier attributes.* Mach. Learn., Vol. 62, No. 1-2, pages 65–105, 2006.

[68] **M. Hall and G. Holmes.** *Benchmarking attribute selection techniques for discrete class data mining.* IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 6, pages 1437-1447, 2003.

[69] **T. H. Wonnacott and R. J. Wonnacott.** *Introductory statistics.* Wiley, 1977.

[70] **I. H. Witten and E. Frank.** *Data mining: practical machine learning tools and techniques.* 2nd edition, San Francisco, CA: Morgan Kaufmann, 2005.

[71] **H. Guo and H. L. Viktor.** *Mining relational data through correation-based multiple view validation.* Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 567-573, 2006.

[72] **H. Guo, H. L. Viktor, and E. Paquet.** *Privacy leakage in multi-relational learning via unwanted classification models.* Proceedings of the 21st Annual Inernational Conference on Computer Science and Software Engineering**,** Toronto, 2011.

[73] **I. Coursac, N. Duteil, and N. Lucas.** *PKDD 2001 discovery challenge - medical domain.* the PKDD Discovery Challenge 2001, Vol. 3, No. 2, 2002.

[74] **J. R. Quinlan.** *Induction of decision trees.* Mach. Learn., Vol. 1, No. 1, pages 81-106, 1986.

[75] **J. R. Quinlan.** *Uknown attribute values in induction.* Proceedings of the 6th International Workshop on Machine Learning, pages 164-168, 1989.

[76] **W. Cohen.** *Fast effective rule induction.* In A. Prieditis and S. Russell (eds), Proceedings of the 12th International Conference on Machine Learning, Tahoe City, CA, pages 115-123, 1995.

[77] **V. Vapnik.** *The nature of statistical learning theory.* New York : Springer, 1995.

[78] **P. Domingos and M. Pazzani.** *On the optimality of the simple Bayesian classifier under zero-one loss.* Mach. Learn. Vol. 29, pages 103-130, 1997.

[79] **M. -A. Krogel and S. Wrobel.** *Facets of aggregation approaches to propositionalization.* Proceedings of the Work-in-Progress Track at the Inductive Logic Programming, 2003.

[80] **P. Berka.** *Guide to financial data set.* In A.Siebes and P.Berka, editors, PKDD2000 Discovery Challenge, 2000.

[81] **G. Widmer and M. Kubat.** *Learning in the presence of concept drift and hidden contexts*. Mach. Learn., Vol. 23, pages 69-101, 1996.

[82] S. Das. *Filters, wrappers and a boosting-based hybrid for feature selection.* Proceedings of the 18th International Conference on Machine Learning, San Francisco, CA, USA, pages 74–81, 2001.

[83] **E. Paquet, H. L. Viktor, H. Guo.** *To aggregate or not to aggregate: that is the question*. Proceedings of International Conference on Knowledge Discovery and Information Retrieval, Paris, France, 2011.

[84] **D. Agrawal and C. C. Aggarwal**. *On the design and quantification of privacy preserving data mining algorithms*. Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, NY, USA: ACM, pages 247–255 , 2001.

[85] **B.-C. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala**. *Privacy-preserving data publishing Found*. Trends databases, Vol. 2, No. 1–2, pages 1–167, 2009.

[86] **A. Gkoulalas-Divanis and V. S. Verykios**. *An overview of privacy preserving data mining.* ACM Crossroads, Vol. 15, No. 4, 2009.

[87] **C. Clifton and D. Marks.** *Security and Privacy Implications of Data Mining*. In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pages 15-19, 1996.

[88] **P. Samarati**. *Protecting respondents identities in microdata release*. IEEE Transactions on Knowledge and Data Engineering. Vol. 13, No. 6, pages1010-1027, 2001.

[89] **M. Z. Islam and L. Brankovic**. *DETECTIVE: A decision tree based categorical value clustering and perturbation technique for preserving privacy in data mining*. Proceedings of the 3rd International IEEE Conference on Industrial Informatics, pages 701-708, 2005.

[90] **J. Goldman and Z. Hudson**. *Perspective Virtually Exposed: Privacy and E-Health.* Health Affairs, Vol. 19, No. 6, pages 140-148, 2000.

[91] **B. Malin, L. Sweeny, and E. Newton**. *Trail Re-identification: Learning Who You Are from Where You Have Been.* Technical Report - LIDAP-WP12, Carnegie Mellon University, Pittsburgh, 2003.