WARSAW UNIVERSITY OF TECHNOLOGY

Faculty of Electronics and Information Technology

Ph.D. THESIS

Piotr Andruszkiewicz, M.Sc.

Privacy Preserving Classification and Association Rules Mining over Centralised Data

Supervisor
Professor Marzena Kryszkiewicz, Ph.D., D.Sc.

Warsaw, 2011

Abstract

This thesis is devoted to privacy preserving classification and association rules mining over centralised data distorted with randomisation-based methods which modify individual values at random to provide an expected level of privacy. It is assumed that only distorted values and parameters of a distorting procedure are known during the process of building a classifier and mining association rules.

In this thesis, we have proposed the optimisation MMASK, which eliminates exponential complexity of estimating an original support of an itemset with respect to its cardinality, and, in consequence, makes the privacy preserving discovery of frequent itemsets and, by this, association rules feasible. It also enables each value of each attribute to have different distortion parameters. We showed experimentally that the proposed optimisation increased the accuracy of the results for high level of privacy. We have also presented how to use the randomisation for both ordinal and integer attributes to modify their values according to the order of possible values of these attributes to both maintain their original domain and obtain similar distribution of values of an attribute after distortion. In addition, we have proposed privacy preserving methods for classification based on Emerging Patterns. In particular, we have offered the eager ePPCwEP and lazy lPPCwEP classifiers as privacy preserving modifications of eager CAEP and lazy DeEPs classifiers, respectively. We have applied meta-learning to privacy preserving classification. Not only have we used bagging and boosting, but also we have combined different probability distribution of values of attributes reconstruction algorithms and reconstruction types for a decision tree in order to achieve higher accuracy of classification. We have proved experimentally that meta-learning gives higher accuracy gain for privacy preserving classification than for undistorted data.

The solutions presented in this thesis were evaluated and compared to the existing ones. The proposed methods obtained better accuracy in privacy preserving association rules mining and classification. Moreover, they reduced time complexity of discovering association rules with preserved privacy.

Streszczenie

Praca jest poświęcona zagadnieniu zapewniania prywatności w odkrywaniu reguł asocjacyjnych oraz klasyfikacji z wykorzystaniem danych scentralizowanych, które zostały zmodyfikowane przy użyciu metody zakłócania oryginalnej wartości polegającej na modyfikacji poszczególnych wartości w losowy sposób. Zgodnie z założeniami tej metody w trakcie budowy klasyfikatora i odkrywania reguł asocjacyjnych znane są jedynie dane zmodyfikowane i parametry opisujące procedurę modyfikacji.

W pracy zaproponowana została optymalizacja MMASK dla algorytmu odkrywania reguł asocjacyjnych z zapewnianiem prywatności, która w zadaniu szacowania wsparcia zbioru eliminuje wykładniczą ze względu na liczbę elementów tego zbioru złożoność obliczeniową i czyni proces odkrywania zbiorów częstych z zapewnianiem prywatności, a w konsekwencji reguł asocjacyjnych, możliwym w praktycznych zastosowaniach. Optymalizacja ta pozwala również na wykorzystanie różnych parametrów modyfikacji poszczególnych wartości atrybutów i umożliwia poprawę jakości uzyskiwanych rezultatów przy zadanym poziomie prywatności. Zaproponowana w pracy metoda modyfikacji oryginalnych wartości atrybutów porządkowych pozwala na przeprowadzenie tego procesu zgodnie z porządkiem i dziedziną tych atrybutów. Dodatkowo zaproponowano metody klasyfikacji oparte na wzorcach wyłaniających i zapewniające prywatność. Przedstawione metody zostały wykorzystane w gorliwym klasyfikatorze ePPCwEP i leniwym lPPCwEP. Klasyfikatory te są zapewniającymi prywatność modyfikacjami algorytmów CAEP i DeEPs. W pracy zaproponowano także wykorzystanie metauczenia. Zastosowane zostały nie tylko metody bagging i boosting, ale także połączono rezultaty pochodzące z różnych algorytmów rekonstrukcji oryginalnego rozkładu wartości atrybutów oraz typów rekonstrukcji dla drzewa decyzyjnego w celu poprawy jakości klasyfikacji. Eksperymentalnie wykazano, że metauczenie pozwala na uzyskanie większego wzrostu jakości klasyfikacji przy wykorzystaniu zmodyfikowanych danych niż w przypadku braku zapewniania prywatności.

Zaproponowane w pracy metody zostały przetestowane i porównane z dotychczas istniejącymi. Nowe metody pozwoliły na uzyskanie lepszych rezultatów w odkrywaniu reguł asocjacyjnych i klasyfikacji z zachowaniem prywatności, a także na zmniejszenie złożoności czasowej procesu odkrywania reguł asocjacyjnych.

Acknowledgements

I wish to express my deepest gratitude to my supervisor, Professor Marzena Kryszkiewicz, for her guidance in the world of science and help in determining research objectives.

I am most grateful to my family and friends who have contributed to this work with their unwavering patience and heartwarming encouragement.

Last but not least, financial support for this research has been provided by the grants No 3 T11C 002 29 and N N516 070437 from Ministry of Science and Higher Education of the Republic of Poland.

Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

Since privacy concerns related to a possible misuse of knowledge discovered by means of data mining techniques have been raised [33], many attempts have been made to provide privacy preserving techniques for data mining [121, 7, 98]. Thus, a new (sub)domain of data mining, privacy preserving data mining, emerged in the last decade. In order to provide sufficient privacy protection in data mining, several methods for incorporating privacy have been developed. *Privacy* itself is not an easy term to define and can be preserved on different levels in different scenarios [118, 1]. In spite of enormous diversity in privacy aspects of data mining, three main approaches can be distinguished: heuristic-based, reconstruction-based and cryptography-based [121].

In the first approach, the heuristic algorithms are used to hide knowledge an organisation does not want to reveal, for instance, individual values in data are changed according to a heuristic algorithm to hide sensitive knowledge such as important rules in the case of association rules mining.

The reconstruction-based approach is used to incorporate privacy on an individual level by changing original individual values (for instance, users' answers) in a random way by means of a randomisation-based method and revealing only modified values. The distorted data as well as parameters of a randomisation-based method used to distort them can be published or passed to a third party. Knowing distorted individual values and parameters of a randomisation-based method, one is able to perform data mining tasks. To this end, first original distributions of values of attributes are reconstructed (estimated) based on the distorted values and the parameters of the distortion method, and a data mining model is built based on the distorted data. The creation of a model is carried out without the need to access original individual data.

The third approach, which is based on cryptography, uses secure multiparty computations (SMC) to carry out data mining tasks based on distributed data, that is, data possessed by different organisations that do not want to disclose their private input. Furthermore, encryption techniques which enable one to perform computations over encrypted data without being able to decrypt can be used in privacy preserving.

The heuristic approach is designed for centralised data. The cryptography-based approach

is used for the distributed data, while the reconstruction-based approach can be applied to both distributed and centralised data.

The performance is one of the advantages of the reconstruction-based approach compared to cryptography-based solutions [44]. Nevertheless, this technique achieves it at the cost of accuracy of the results. As the reconstruction-based approach uses individual randomised values, there is a trade-off between the level of privacy and obtained accuracy of the data mining results. The higher level of privacy is used, the higher loss of accuracy of the data mining results is obtained [50].

The cryptography-based techniques do not suffer from accuracy loss, but their main drawback is high performance cost, especially when many parties are involved in the process [127].

Both the reconstruction-based and cryptography-based approaches can be used to preserve privacy on an individual level. The latter assumes that there are many information providers which need to interact to perform a data mining task. Thus, each time a data mining task is performed, information providers should be ready to interact. The reconstruction-based approach is designed to be used for both distributed and centralised data. The distorted records can be stored in a centralised database and can be used many times in a process of building different models without interactions with information providers. Furthermore, the reconstruction-based approach enables organizations to provide data that one can use to discover hidden knowledge from it without revealing individual characteristics of objects.

Considering Internet surveys, which are very popular nowadays and very likely will be still popular in the future, we may say that the reconstruction-based approach fits such applications better because, contrary to the cryptography-based approach, data providers do not participate in the process of building a model, i.e., they do not obtain the results or partial results of a data mining task. Users' concerns about a possible misuse of provided data can be reduced by means of a randomisation-based method, which distorts original data and stores only distorted values in a centralised database. The distortion procedure, which modifies individual values of the object, does not depend on any information about other objects, thus it can be performed at the object site. In the case of Internet surveys, the distortion can be done at a user machine, which makes a randomisation-based method easy to incorporate into a web browser.

Moreover, in the reconstruction-based approach additional objects provided by new users can be added to a centralised database and a data mining model can be rebuilt without the need to interact with other data providers.

In this thesis, we investigate the use of randomisation-based methods and the reconstruction-

based techniques in developing privacy preserving data mining for centralised data. In this work, we focus on the following approach to privacy preserving data mining: 1) data is distorted by means of randomisation-based methods to preserve privacy and is stored in a centralised database; 2) the reconstruction-based techniques are used to perform a mining task on stored distorted data, and, in the case of classification, on original values of objects to be classified.

The main goal of this thesis is to find solutions for privacy preserving classification and association rules mining over centralised data distorted with randomisation-based methods. The proposed solutions should take into account the following:

— basic kinds of attributes, namely continuous, integer, nominal, and ordinal attributes,
— high level of privacy,
— time complexity of privacy preserving data mining tasks,
— a level of accuracy of the results.

The main challenge is to reduce or eliminate the loss of accuracy of the results caused by applying the randomisation-based methods. Second task of this thesis is to make privacy preserving data mining viable in practice from time complexity point of view.

1.1. Statements of the Thesis

The following statements are made in this work:

1. In privacy preserving classification and association rules mining, continuous, integer, nominal, and ordinal attributes can be distorted by means of the randomisation-based methods according to attributes characteristics.
2. The accuracy loss which comes from incorporating privacy on an individual level by means of the randomisation-based methods for centralised data in privacy preserving classification and association rules mining can be reduced or eliminated.
3. The exponential complexity of estimating support of an itemset with respect to its cardinality in the process of privacy preserving association rules mining over centralised data distorted by means of the randomisation-based methods can be reduced to the constant complexity.

1.2. Contribution

We made the following contribution:

1. Our proposed optimisations of support estimation of an itemset in the process of privacy preserving association rules mining can substantially reduce the time complexity of this process and make it viable in practice.
2. We showed that the randomisation-based perturbation of data can be applied for ordered attributes according to an order and domains of attributes.
3. We presented that privacy preserving classifiers based on (Jumping) Emerging Patterns can reduce the trade-off between privacy and accuracy. Both the eager and lazy approach to classification with Emerging Patterns can be used in an effective way.
4. We proposed how to use meta-learning to combine results obtained for different reconstruction types and algorithms of reconstructing an original distribution of values of an attribute in order to significantly reduce the accuracy loss caused by incorporating privacy by means of the randomisation-based methods.

1.3. Content Outline

In Chapter 2, we describe theoretical foundations of association rules and classification. Chapter 3 shows the background for privacy preserving association rules and classification. In this chapter, the solutions proposed in literature, for instance, the algorithms for reconstructing an original distribution of values of an attribute and Mining Associations with Secrecy Konstraints (MASK) Scheme, are also presented and analysed. In Chapter 4, we propose the MMASK optimisation for estimating support of an itemset, which is critical operation in privacy preserving association rules mining over centralised data distorted by means of randomisation-based methods. This optimisation eliminates exponential complexity of estimating original support of an itemset with respect to its cardinality. In Chapter 5, we propose how to randomise ordered attributes according to their order and domain. In Chapter 6, we propose the privacy preserving eager ePPCwEP and lazy lPPCwEP classifiers which use Emerging Patterns and are based on CAEP and DeEPS, respectively. In Chapter 7, we show how to utilise meta-learning concept in privacy preserving classification to combine results for different reconstruction types and algorithms of reconstructing an original distribution of values of an attribute. We conclude our work and present future avenues to explore in Chapter 8.

2. Theoretical Foundations of Association Rules and Classification

This chapter presents basic concepts of association rules mining and classification. In Chapter 3, these theoretical foundations are used while incorporating privacy in data mining.

2.1. Association Rules

The concept of association rules was proposed in [3]. To define an association rule, we introduce basic notation:

Let $I = \{i_1, i_2, ..., i_k\}$ be a set of items. Any subset X of items in I is called an *itemset*. An itemset X is called a *k-itemset* when X consists of k items. k is the *length* of the itemset X. A *transaction database* $\mathcal{D}$ is a set of itemsets. An itemset T in a transaction database $\mathcal{D}$ is a *transaction*. A transaction T supports X if all items in X are present in T.

An association rule is defined as follows:

Definition [6]. An *association rule* is an expression of the form $X \Rightarrow Y$, where $X \subseteq \mathcal{I}$, $Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$.

An association rule is characterised by means of a *support* and a *confidence* measures.

Definition [3]. A *support* of an itemset X, denoted as $sup(X)$, is the number (or the percentage) of transactions in $\mathcal{D}$ that contain X.

A support of an association rule $X \Rightarrow Y$ ($sup(X \Rightarrow Y)$) in a transaction database $\mathcal{D}$ is the number (or the percentage) of transactions in $\mathcal{D}$ that contain $X \cup Y$ and is equal to the support of the set $X \cup Y$, i.e., $sup(X \cup Y)$.

Definition [3]. A *confidence* of an association rule $X \Rightarrow Y$, denoted as $conf(X \Rightarrow Y)$, is the percentage of transactions in $\mathcal{D}$ that contain Y among those containing X.

$$conf(X \Rightarrow Y) = sup(X \Rightarrow Y)/sup(X)$$

The computational task in discovering association rules is to mine for a given set $\mathcal{D}$ of transactions all association rules with the support greater than a user-specified minimum support threshold *minimumSupport* and the confidence greater than a minimum confidence threshold *minimumConficence*. Association rules that meet these two conditions are called *strong association rules*.

To mine strong association rules, as a first step, one usually finds itemsets with a support greater then a minimum support threshold.

Definition *Frequent itemsets*, denoted as $\mathcal{F}$, are those itemsets whose support is greater than a minimum support threshold *minimumSupport*, that is:

$$\mathcal{F} = \{X \subseteq I \mid sup(X) > minimumSupport\}.$$

An enormous effort has been made to efficiently discover frequent itemsets and association rules [6, 100, 5, 28, 131, 132, 56, 23].

Usually the task of discovering association rules is decomposed into two steps [6]:

1. All combinations of items with supports greater than a given minimum support threshold, frequent itemsets, are mined.
2. The frequent itemsets are used to generate association rules that hold the minimum confidence condition. The idea is as follows: let F be a frequent itemset and $Y \subseteq F$. Any rule $F \setminus Y \Rightarrow Y$ is a strong association rule if $\frac{sup(F)}{sup(F \setminus Y)} > minimumConfidence$.

2.1.1. Apriori

One of the most popular algorithms for discovering frequent itemsets is Apriori [100]. The idea behind this algorithm is that any subset of a frequent itemset must be frequent and any superset of an infrequent itemset must be infrequent. Thus, candidate m-itemsets (itemsets having m items) can be generated by joining frequent $(m-1)$-itemsets, and removing those that contain any infrequent subset. This method generates all possible frequent candidates.

Apriori (see Algorithm 1) counts occurrences of items to find frequent 1-itemsets. Then in m-th pass, it generates the candidate itemsets $\mathcal{X}_m$ based on frequent $(m-1)$-itemsets using the $aprioriGen$ function described later in this section. Next, the database is scanned to count the supports of the candidates. Each candidate has an associated field to store its support. Only frequent itemsets from $\mathcal{X}_m$ are added to $\mathcal{F}_m$.

We will use the following notation for Apriori:

— $\mathcal{X}_m$ denotes candidate m-itemsets, which are potentially frequent.

— $\mathcal{F}_m$ are frequent m-itemsets.

— $X.c$ means the support field of the itemset X.

— $X[i]$ is the i-th item in the itemset X.

— $X[1]{\cdot}X[2]{\cdot}X[3]{\cdot}\ldots{\cdot}X[m]$ denotes m-itemset, which consists of $X[1], X[2], X[3], \ldots, X[m]$.

Algorithm 1 The Apriori algorithm

input: $\mathcal{D}$ // a transaction database
input: minimumSupport
$\mathcal{F}_1$ ={frequent 1-itemsets}
for $(m = 2; \mathcal{F}_{m-1} \neq \emptyset; m++)$ **do begin**
$\mathcal{X}_m$ = $aprioriGen(\mathcal{F}_{m-1})$ //generate new candidates
$supportCount(\mathcal{X}_m)$
$\mathcal{F}_m = \{X \in \mathcal{X}_m | X.\text{c} \geq \text{minimumSupport }\}$
end
return $\bigcup_m \mathcal{F}_m$

Algorithm 2 The candidate generation algorithm

function $aprioriGen$(**var** $\mathcal{F}_m$)
for all $Y, Z \in \mathcal{F}_m$ **do begin**
if $Y[1] = Z[1] \wedge \ldots \wedge Y[k-1] = Z[k-1] \wedge Y[k] < Z[k]$ **then begin**
$X = Y[1] \cdot Y[2] \cdot Y[3] \cdot \ldots \cdot Y[k-1] \cdot Y[k] \cdot Z[k]$
add X to $\mathcal{X}_{m+1}$
end
end
for all $X \in \mathcal{X}_{m+1}$ **do begin**
for all m-itemsets $Z \subset X$ **do begin**
if $Z \notin \mathcal{F}_m$ **then** delete X from $\mathcal{X}_{m+1}$
end
end
return $\mathcal{X}_{m+1}$
end

Algorithm 3 The support count algorithm

procedure $supportCount$(**var** $\mathcal{X}_m$)
for all transactions $T \in \mathcal{D}$ **do begin**
for all candidates $X \in \mathcal{X}_m$ **do begin**
if $X \subseteq T$ **then** $X.c$++
end
end
end

The $aprioriGen$ function in the first step merges frequent sets $\mathcal{F}_m$ and generates candidates $\mathcal{X}_{m+1}$. In the second step, the function deletes all itemsets $X \in \mathcal{X}_{m+1}$ such that at least one $(m-1)$-subset of X is not in $\mathcal{F}_m$.

The essential for time efficiency in frequents itemsets finding in Apriori fashion manner is counting of the support for candidates.

2.1.2. Generalised Association Rules with Taxonomy

The problem of generalised association rules has been introduced in [107] and further discussed in [79, 129, 78, 111]. In generalised association rules there is a taxonomy (an *is-a* hierarchy) on items and association between items on any level of taxonomy can be found. For example, given a taxonomy: milk *is-a* drink, mineral water *is-a* drink, bread *is-a* food, a rule that *people who buy food tend to buy mineral water* may be inferred. This rule may hold even if rules that *people who buy bread tend to buy mineral water* and *people who buy food tend to buy mineral water* do not hold.

2.1.3. Quantitative Association Rules

In [108], the problem of mining association rules in large relational tables containing both quantitative and nominal attributes has been introduced. To tackle this problem, quantitative attributes can be partitioned. Then nominal attributes and partitioned quantitative (continuous or integer) attributes can be mapped into binary attributes and association rules mined. An example of a quantitative rule can be: *10% of people who are at most 35 years old and drive sports car have 2 cars*. The introduced problem of quantitative rules has been widely discussed in data mining literature [53, 82, 22, 140, 58, 85, 11, 67].

2.2. Decision Tree

First we define training and test sets.

Definition A *training set* is a set of samples with known class label which are used to train a classifier.

Definition A *test set* is a set of samples with known class label which are used to assess a classifier.

Then we describe a concept of a decision tree.

A decision tree is a class discriminator [27]. It represents recursive splits of a training set into disjoint subsets until each subset, which represents a node, consists only or dominantly[1]

[1] One should avoid creating a node without a dominant class, nevertheless, a dominant class in a higher node or a randomly chosen class can be used then.

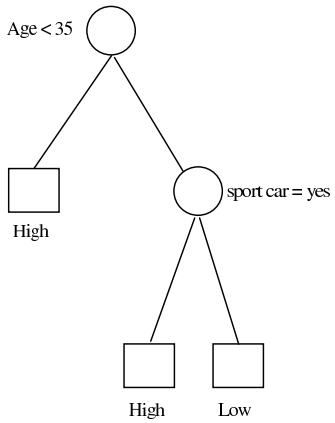

Figure 2.1. An example of a decision tree.

of the train samples from one class. Each non-leaf node, i.e., a node with at least one child, contains a test (a *split point*) on one or more attributes, which determines how to split data. In this dissertation, only tests on one attribute are considered. For continuous attributes we use tests defined as follows:

$$v_A < v_{thr},$$

where A is a continuous attribute, v_A is a value of an attribute A, and v_{thr} is a value threshold.

Let B be a nominal attribute with k possible values $\{v_1, \ldots, v_k\}$ and $V \subseteq \{v_1, \ldots, v_k\}$. For nominal attributes we use tests defined as follows:

$$v_B \in V,$$

v_B is a value of an attribute B.

For binary attributes we use also the following notation:

$$v_B = v,$$

v_B is a value of an attribute B and v is one of the possible values of an attribute B.

Figure 2.1 shows an example of a decision tree, which uses two tests. The first test (in the root of the tree) splits a training set according to the test: $Age < 35$. Training samples which

meet the test go into the left child node. The remaining samples go to the right child node. The second test is: $Sport\ car = yes$. The class attribute describes the level of risk for a car insurance company that an insured car will be damaged to some extent. The possible values of the class attribute are $\{High, Low\}$.

The concept of a decision tree has been widely developed [80, 103]. Very notable is Quinlan's contribution [88, 92, 93, 91, 94] and his algorithms for decision trees such as ID3 [87] and its later version C4.5 [89].

The process of developing a decision tree consists of two phases:

1. Growth phase,
2. Pruning phase.

Phase 1 is described by Algorithm 4, where the notation is as follows:

— P - a training set,

— T - a tree,

— t - a test,

— R_t - a set of possible results of a test t,

— $t(x)$ - a results of a test t for a sample x.

Algorithm 4 The growth phase of a decision tree

```
procedure buildRecurrent(P, T)
  if stop criterion is met then
    T.label = a dominant category in P, if present,
              a dominant category in a higher node or a random category, otherwise
    return
  t = the best test choosen for P
  T.test = t
  for all r ∈ R_t // for all possible results of a test t
    P' := {x ∈ P | t(x) = r}
    buildRecurrent(P', T.leaf(r))
end
```

The key point of Algorithm 4 is a process of finding the best split of data. To this end, one of the split selection methods can be used [77], such as *Gini index* [27], *information gain* [46] based on entropy [106], *gain ratio* [89], χ^2 splitting criterion [65].

Definition *Gini index* for a data set Z with k classes is:

$$gini(Z) = 1 - \sum_{j=1}^{k} p_j^2,$$

where p_j is the relative frequency of class j in a data set Z, $p_j = \frac{|\{z \in Z | class = j\}|}{|Z|}$.

$Gini$ *index* measures impurity of a class distribution in a node. This index shows how often a randomly chosen sample from the training samples in a node would be incorrectly classified if it were randomly classified according to the distribution of classes in the training samples. $gini(Z)$ reaches its minimal possible value of 0 when all training samples in Z fall into a single class.

$Gini_{split}$ *index* measures impurity of a partition of a set.

Definition $Gini_{split}$ *index* for a data set Z partitioned into l subsets $Z_1, Z_2, ..., Z_l$ is:

$$gini_{split}(Z) = \sum_{i=1}^{l} \frac{|Z_i|}{|Z|} gini(Z_i),$$

where $|Z_i|$ ($|Z|$) is the number of elements in the set Z_i (Z respectively).

$Gini_{split}$ *index* is a weighted average of $Gini$ *index* for all subsets which a set was partition into. A value of $Gini_{split}$ *index* is in the range of $\langle 0, 1 \rangle$. To choose the best split, a partition with the lowest obtainable value of $Gini_{split}$ *index* among considered partitions should be found.

An other splitting method is *information gain* [46], which is based on entropy.

Definition *Entropy* for a data set Z with k classes is:

$$entropy(Z) = -\sum_{j=1}^{k} p_j \log p_j,$$

where p_j is the relative frequency of class j in a data set Z.

Definition *Information gain* for a data set Z and an attribute A is:

$$gain(Z, A) = entropy(Z) - \sum_{v \in values(A)} \frac{|Z_v|}{|Z|} entropy(Z_v),$$

where $values(A)$ represents each possible value of an attribute A and Z_v is the subset of samples from the set Z for which the attribute A has the value v, where $|Z_v|$ ($|Z|$) is the number of elements in the set Z_v (Z respectively).

In order to find the best split, information gain is calculated for each attribute. The attribute with the highest value of information gain is chosen.

The next phase of the process of developing a decision tree, Phase 2, pruning, reduces *over-fitting* in a decision tree. Over-fitting occurs when a classifier describes a random error or noise instead of interesting relations. The concept of over-fitting refers to the situation in which an algorithm creates a classifier which perfectly fits the training samples but has lost its

capability of generalising to instances not present during training [99]. Instead of learning, a classifier memorises training samples.

An over-fitted classifier gives excellent results on a training set, nevertheless, results obtained on a test set are poor.

The pruning phase can be performed according to the Minimum Description Length (MDL) principle [97, 95, 90, 81, 80].

In MDL (Minimum Description Length) principle the best model for encoding data has the lowest value of the sum of the cost of describing a data set given the model and the cost of describing this model [80].

Definition The *total cost* of encoding is defined as follows:

$$cost(M, D) = cost(D|M) + cost(M),$$

where M is a model that encodes a data set D, $cost(D|M)$ is the cost of encoding a data set D in terms of a model M, $cost(M)$ is the cost of encoding a model M.

In case of a decision tree, the goal of MDL pruning is to find a subtree which best describes a training set. A subtree is obtained by pruning an initial decision tree T.

The pruning algorithm consists of two components:

1. The encoding component that calculates the cost of encoding data and a model,
2. The algorithm that compares subtrees of an initial decision tree T.

The cost of encoding a training set given a decision tree T is the sum of classification errors for training samples. A classification error for a sample s occurs if the class label produced by the decision tree T is different from an original class label of a sample s. The count of classification errors is collected during the growth phase.

The cost of encoding a model includes the cost of describing a decision tree and the cost of describing tests used in each internal node of a tree.

If a node in a decision tree is allowed to have either zero or two children, it can be described as one bit, because there are only two possibilities.

The cost of a split depends on the type of an attribute used in a split. For a continuous attribute A and a test of the form $v_A < v_{th}$, the cost C of encoding this test is the overhead of encoding v_{th}. Though the value of C should be determined for each test of this type in a decision tree, an empirically chosen constant value of 1 is assumed as proposed in [80].

For a nominal attribute B with k possible values $\{v_1, \ldots, v_k\}$ and a test of the form $v_B \in V$,

where $V \subseteq \{v_1, \ldots, v_k\}$, the cost of a test is calculated as $\ln n_B$, where n_B is the number of tests on an attribute B in a tree.

To determine whether to convert a node into a leaf, the algorithm calculates the code length $C(t)$ for each node t as follows:

$C_{leaf}(t) = L(t) + Errors(t)$, if t is a leaf,

$C_{both}(t) = L(t) + L_{test}(t) + C(t_1) + C(t_2)$, if t is has both children: t_1 and t_2,

where $L(t)$ is the number of bits required to encode a node (for a node with either zero or two children $L(t)$ is equal to one bit), $Errors(t)$ is the sum of classification errors for a node t and $L_{test}(t)$ is the cost of encoding a test in a node t.

We use the pruning strategy which was first presented in [81] and then used in [80]. According to this strategy, both children of a node t are pruned and the node t is converted into a leaf if $C_{leaf}(t) < C_{both}(t)$.

2.3. Emerging Patterns

The notion of *Emerging Patterns* (EP) was introduced in [68, 40, 39, 38]. Emerging Patterns capture significant changes and differences between data sets. They are defined as itemsets whose supports increase significantly from one data set to another.

Let us assume that there is a training data set $\mathcal{D}$ with n binary attributes. Each instance in the training data set $\mathcal{D}$ is associated with one of k labels, $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. The training data set $\mathcal{D}$ is partitioned into k disjoint sets $\mathcal{D}_i,\ i = 1, ..., k$ containing all instances of class $\mathcal{C}_i$.

$$\mathcal{D}_i = \{X \in \mathcal{D} \mid X \; is \; an \; instance \; of \; class \; \mathcal{C}_i\}$$

Let us assume that I is the set of all items (binary attributes). An itemset X is a subset of I.

Definition A *support of an itemset* X in a data set $\mathcal{D}$ is:

$$sup_{\mathcal{D}}(X) = \frac{|\{S \in \mathcal{D} | X \subseteq S\}|}{|\mathcal{D}|}.$$

Definition The *growth rate of an itemset* X from a data set $\mathcal{D}'$ to $\mathcal{D}''$ is defined as follows:

$$growthRate_{\mathcal{D}' \to \mathcal{D}''}(X) = \begin{cases} \frac{sup_{\mathcal{D}''}(X)}{sup_{\mathcal{D}'}(X)}; & sup_{\mathcal{D}'}(X) \neq 0 \\ = 0; & sup_{\mathcal{D}'}(X) = 0 \text{ and } sup_{\mathcal{D}''}(X) = 0 \\ = \infty; & sup_{\mathcal{D}'}(X) = 0 \text{ and } sup_{\mathcal{D}''}(X) \neq 0. \end{cases}$$

Definition An *ρ-Emerging Pattern* (also called an *EP*) from $\mathcal{D}'$ to $\mathcal{D}''$ is an itemset X if $growth\ Rate_{\mathcal{D}' \to \mathcal{D}''}(X) \geq \rho$, where ρ is a growth rate threshold and $\rho > 1$.

EPs with $growthRate$ equal to ∞ are called Jumping Emerging Patterns (JEP) [70]. JEPs are itemsets which are present in one set and not present in the other.

After the introduction of Emerging Patterns several eager-learning classifiers based on EPs were proposed.These algorithms discover EPs in the training phase and then classify each new sample based on discovered EPs. One of the examples of eager-learning classifiers based on EPs is CAEP [41].

In [69], a lazy classifier DeEPs was also presented. When DeEPs needs to classify a sample, it mines only EPs related to this sample. It repeats this process for each sample from a testing set.

In subsequent sections, we will present in more detail two mentioned algorithms: CAEP and DeEPs.

2.3.1. Review of CAEP

One of the first classifiers based on Emerging Patterns was CAEP (Classification by Aggregating EPs) [41]. CAEP algorithm utilises the idea that each EP can differentiate a class membership of instances which contain this EP. The discriminating power comes from a big difference between supports of this EP in classes. Unfortunately, an EP may cover only a small fraction of instances and cannot be used itself to classify all instances, because it will only yield accurate predictions for the fraction of instances which contain this EP. Thus, it is better to combine differentiate power of a set of EPs [41] and let all the EPs that a test sample contains contribute to a final decision about a class label associated with a given test sample and take the advantage of covering more instances than a single EP can cover.

Let us assume that the data set $\mathcal{D}$ has been partitioned into subsets $\mathcal{D}_i, i = 1, ..., k$ according to the class labels $\mathcal{C}_i$. $\mathcal{D}'_i$ is the opponent class and is equal $\mathcal{D}'_i = \mathcal{D} \setminus \mathcal{D}_i$. We refer to EPs mined from $\mathcal{D}'_i$ to $\mathcal{D}_i$ as the *EPs of class* $\mathcal{C}_i$.

The contribution of a single EP, E of class $\mathcal{C}_i$ is given by $\frac{growthRate(E)_{\mathcal{D}'_i \to \mathcal{D}_i}}{growthRate(E)_{\mathcal{D}'_i \to \mathcal{D}_i}+1} sup_{\mathcal{C}_i}(E)$. The first term can be seen as a conditional probability that an instance is of class $\mathcal{C}_i$ given that this instance contains the Emerging Pattern E. The second term is a fraction of the instances of class $\mathcal{C}_i$ that contain the Emerging Pattern E. The contribution of E is proportional to both $growthRate(E)_{\mathcal{D}'_i \to \mathcal{D}_i}$ and $sup_{\mathcal{C}_i}(E)$.

Table 2.1. Saturday morning activity for weather conditions [87]

Class $\mathcal{P}$				Class $\mathcal{N}$			
outlook	temperature	humidity	windy	outlook	temperature	humidity	outlook
overcast	hot	high	false	sunny	hot	high	false
rain	mild	high	false	sunny	hot	high	true
rain	cool	normal	false	rain	cool	normal	true
overcast	cool	normal	true	sunny	mild	high	false
sunny	cool	normal	false	rain	mild	high	true
rain	mild	normal	false				
sunny	mild	normal	true				
overcast	mild	high	true				
overcast	hot	normal	false				

Table 2.2. The transformed Saturday morning activity for weather conditions [87]

Class $\mathcal{P}$	Class $\mathcal{N}$
{overcast, hot, high, false}	{sunny, hot, high, false}
{rain, mild, high, false}	{sunny, hot, high, true}
{rain, cool, normal, false }	{ rain, cool, normal, true}
{overcast, cool, normal, true }	{ sunny, mild, high, false}
{sunny, cool, normal, false}	{ rain, mild, high, true}
{rain, mild, normal, false }	
{sunny, mild, normal, true }	
{overcast, mild, high, true }	
{overcast, hot, normal, false }	

The overall score of an instance for the classes is the sum of the contribution of the individual EPs.

Definition Given an instance S to be classified and a set $\mathcal{E}(\mathcal{C}_i)$ of EPs of a class $\mathcal{C}_i$ discovered from a training data set, *an aggregate score* of instance S for $\mathcal{C}_i$ is defined as:

$$score(S, \mathcal{C}_i) = \sum_{E \subseteq S, E \in \mathcal{E}(\mathcal{C}_i)} \frac{growthRate(E)_{\mathcal{D}'_i \to \mathcal{D}_i}}{growthRate(E)_{\mathcal{D}'_i \to \mathcal{D}_i} + 1} sup_{\mathcal{C}_i}(E). \tag{2.1}$$

A calculated score is normalised using a *base score*, which is a score at a fixed percentile (for instance, 75%) for training instances of each class. A normalised score of an instance S for class $\mathcal{C}_i$ is the ratio $score(S, \mathcal{C}_i)/baseScore(\mathcal{C}_i)$.

A class with the largest normalised score is chosen.

Example [41] The following example shows the process of classification with CAEP.

Table 2.1 presents the training set for predicting if there are good weather conditions for some *Saturday activity* [87]. The transformed training set for CAEP is shown in Table 2.2.

Table 2.3. The scores of training instances of Saturday morning activity for weather conditions

Class $\mathcal{P}$		Class $\mathcal{N}$	
$score(X, \mathcal{P})$	$score(X, \mathcal{N})$	$score(X, \mathcal{P})$	$score(X, \mathcal{N})$
18.44	0.31	4.89	5.51
16.65	0.39	8.37	5.47
15.76	0.05	2.8	**5.4**
15.28	0.21	9.93	4.97
14.52	0.41	10.31	4.8

An example of an Emerging Pattern of class $\mathcal{N}$, i.e., from $\mathcal{N}$ to $\mathcal{P}$, is $E1 = \{sunny, mild\}$ with $sup_{\mathcal{P}}(E1) = 1/9 = 11.11\%$, $sup_{\mathcal{N}}(E1) = 1/5 = 20\%$ and $growthRate_{\mathcal{P}\to\mathcal{N}}(E1) = 1.8$.

A Jumping Emerging Pattern of class $\mathcal{N}$ is, for instance, $E2 = \{sunny, mild, high\}$ with $sup_{\mathcal{P}}(E2) = 0$, $sup_{\mathcal{N}}(E1) = 1/5 = 20\%$ and $growthRate_{\mathcal{P}\to\mathcal{N}}(E1) = \infty$.

An example of a Jumping Emerging Pattern of class $\mathcal{P}$ is $E3 = \{sunny, mild, true\}$ with $sup_{\mathcal{P}}(E3) = 1/9 = 11.11\%$, $sup_{\mathcal{N}}(E3) = 0$ and $growthRate_{\mathcal{N}\to\mathcal{P}}(E3) = \infty$.

Let us assume (as in [41]) that an instance $S = \{sunny, mild, high, true\}$ is to be classified and the growth rate threshold $\rho = 1.1$. Among Emerging Patterns with the growth rate at least 1.1, S contains the following Emerging Patterns of class $\mathcal{P}$: $E3 = \{sunny, mild, true\}$ ($sup_{\mathcal{P}}(E3) = 1/9 = 11.11\%$ and $growthRate_{\mathcal{N}\to\mathcal{P}}(E3) = \infty$), $E4 = \{mild\}$ ($sup_{\mathcal{P}}(E4) = 44\%$ and $growthRate_{\mathcal{N}\to\mathcal{P}}(E4) = 1.11$) and S contains 10 Emerging Patterns of class $\mathcal{N}$ with growth rate at least 1.1: $E5 = \{sunny\}$, $E6 = \{high\}$, $E1 = \{sunny, mild\}$, $E7 = \{sunny, high\}$, $E8 = \{sunny, true\}$, $E9 = \{mild, high\}$, $E10 = \{high, true\}$, $E2 = \{sunny, mild, high\}$, $E11 = \{sunny, high, true\}$, $E12 = \{mild, high, true\}$.

The values of support and growth rate of mentioned EPs are as follows:

$sup_{\mathcal{N}}(E5) = 60\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E5) = 2.7$,

$sup_{\mathcal{N}}(E6) = 80\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E6) = 2.4$,

$sup_{\mathcal{N}}(E1) = 20\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E1) = 1.8$,

$sup_{\mathcal{N}}(E7) = 60\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E1) = \infty$,

$sup_{\mathcal{N}}(E8) = 20\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E1) = 1.8$,

$sup_{\mathcal{N}}(E9) = 40\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E9) = 1.8$,

$sup_{\mathcal{N}}(E10) = 40\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E10) = 3.6$,

$sup_{\mathcal{N}}(E2) = 20\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E2) = \infty$,

$sup_{\mathcal{N}}(E11) = 20\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E11) = \infty$,

$sup_{\mathcal{N}}(E12) = 20\%$, $growthRate_{\mathcal{P}\to\mathcal{N}}(E12) = 1.8$.

The aggregated score of S for $\mathcal{P}$ is calculated as follows: $score(S, \mathcal{P}) = \frac{1.11}{1.11+1} \cdot 0.44 +$

$\frac{\infty}{\infty+1} \cdot 0.11 = 0.33$. The score for $\mathcal{N}$ is equal to: $score(S, \mathcal{N}) = 0.41 + 0.56 + 0.12 + 0.60 + 0.12 + 0.24 + 0.31 + 0.20 + 0.20 + 0.12 = 2.88$.

To show the process of score normalisation, let us assume that there are five training instances for each class and their scores are presented in Table 2.3.

The (median) base scores for $\mathcal{P}$ and $\mathcal{N}$ are 15.76 and 5.4, respectively. Normalised scores for the instance S are $normalisedScore(S, \mathcal{P}) = 0.33/15.76 = 0,21$, $normalisedScore(S, \mathcal{N}) = 2.88/5.4 = 0.53$, thus S is assigned to class $\mathcal{N}$.

□

2.3.2. Review of DeEPs

The DeEPs (Decision-making by Emerging Patterns) [69] algorithm was designed to discover those Emerging Patterns which sharply contrast two classes of data in the context of a given test sample which is to be classified, i.e., the lazy approach is used. In this section, we briefly describe the phases of the classification process with the usage of the DeEPs algorithm.

Assume that there is a set $\mathcal{D}_p = \{P_1, \ldots, P_m\}$ of positive training instances, a set $\mathcal{D}_n = \{N_1, \ldots, N_n\}$ of negative training instances, and a set of test instances $\mathcal{T}$ in a classification problem. T, a test sample from $\mathcal{T}$, is to be classified.

Intersection

The first step in discovery of EPs is to perform the intersection of the training data with T, namely $T \cap P_1, \ldots, T \cap P_m$ and $T \cap N_1, \ldots, T \cap N_n$. The values that do not occur in the test sample T are removed from the training data sets, resulting in sparser training data.

For continuous attributes neighbourhood-based intersection [69] can be applied as follows: let us assume that the attribute A is continuous with the domain [0,1][2]. S is the training instance and T is the test instance. $T \cap S$, i.e., the reduced training instance, will contain the attribute A if its value for S is in the neighbourhood $[x_A - \alpha, x_A + \alpha]$, where x_A is the value of the attribute A for T. The parameter α is called the *neighbour factor* and is used to adjust the length of the neighbourhood. Applying neighbourhood-based intersection, DeEPs can perform an intersection for continuous attributes as well.

Discovery of the Patterns

In this step the interesting patterns - (Jumping) Emerging Patterns are mined in the following way:

[2] All continuous attributes with different domains can be normalised into the domain [0,1].

Maximal itemsets in $T \cap P_1, \ldots, T \cap P_m$ and separately in $T \cap N_1, \ldots, T \cap N_n$ are found. To concisely represent the patterns, the border concept, structured by the boundary elements of a pattern space, is used in [38, 72, 73]. Based on the maximal sets the patterns with the infinite frequency-changing rate are mined, i.e., those subsets of T which occur in $\mathcal{D}_p$ and do not occur in $\mathcal{D}_n$ and subsets of T which occur in $\mathcal{D}_n$ and do not occur in $\mathcal{D}_p$. The third set of subsets of T are those which occur in both sets $\mathcal{D}_p$ and $\mathcal{D}_n$. The itemsets from the third set are reduced to those whose frequency in sets $\mathcal{D}_p$ and $\mathcal{D}_n$ changes significantly. Moreover, they are optional to the classification process and if high decision speed is important, may not be mined. Detailed information about finding borders and its application to Emerging Patterns can be found in [71].

Determining Scores for Test Sample

Having selected the important Emerging Patterns, DeEPs calculates classification scores based on frequencies in classes of the discovered EPs. A collective score of a test sample T for a class $\mathcal{C}$ is determined by aggregating frequencies of EPs in a class $\mathcal{C}$ [69].

Definition The *compact score* of T for class $\mathcal{C}$ is the percentage of instances in $\mathcal{D}_\mathcal{C}$ that contain at least one EP, that is:

$$compactScore(\mathcal{C}) = \frac{count_{\mathcal{D}_\mathcal{C}(\mathcal{E}(\mathcal{C}))}}{|\mathcal{D}_\mathcal{C}|},$$

where $\mathcal{E}(\mathcal{C})$ is the collection of all EPs of class $\mathcal{C}$, $\mathcal{D}_\mathcal{C}$ is the set of training instances with class $\mathcal{C}$, and $count_{\mathcal{D}_\mathcal{C}(\mathcal{E}(\mathcal{C}))}$ is the number of instances in $\mathcal{D}_\mathcal{C}$ that contain at least one of the EPs from $\mathcal{E}(\mathcal{C})$.

The special way of the aggregation avoids duplicate contribution of training instances to the compact summation, e.g., a training instance I which contains Emerging Patterns $E1$, $E2$ and $E5$ from $\mathcal{E}(\mathcal{C})$ is counted only once, not three times.

Having calculated the compact scores for the positive and negative class, DeEPs assigns for the test instance T the class with the highest score. A majority rule is used to break a tie.

Example [69] The following example shows the process of classification with DeEPs. In this example an instance is a set of attribute-value pairs.

Table 2.1 is used as a training data set and an instance $S = \{(outlook, sunny),$ $(temperature, mild), (humidity, high), (windy, true)\}$ is to be classified.

The first step in DeEPs is intersection. The training set reduced with the instance S is shown in Table 2.4. Then interesting patterns are mined. Jumping Emerging Patterns for class $\mathcal{P}$ are: $\{(outlook, sunny),\ (temperature, mild),\ (windy, true)\}$. For class $\mathcal{N}$ Jumping Emerging

Table 2.4. Reduced training set.

Class $\mathcal{P}$				Class $\mathcal{N}$			
outlook	temperature	humidity	windy	outlook	temperature	humidity	outlook
-	-	high	-	sunny	-	high	-
-	mild	high	-	sunny	-	high	true
-	-	-	-	-	-	-	true
-	-	-	true	sunny	mild	high	-
sunny	-	-	-	-	mild	high	true
-	mild	-	-				
sunny	mild	-	true				
-	mild	high	true				
-	-	-	-				

Patterns are: $\{(outlook, sunny), (humidity, high)\}$, $\{(outlook, sunny), (temperature, mild), (humidity, high)\}$, $\{(outlook, sunny), (humidity, high), (windy, true)\}$. The last step is the calculation of compact scores: $compactScore(\mathcal{N}) = \frac{3}{5} = 0.6$ and $compactScore(\mathcal{P}) = \frac{1}{9} = 0.11$. The instance S is assigned to class $\mathcal{N}$.

□

DeEPs for Data Sets with More Than Two Classes

DeEPs can be easily extended to data sets with more than two classes. Let us assume that there is a database containing k classes of training instances $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$. The reduced training instances by the intersection with T are denoted as $\mathcal{D}'_1, \mathcal{D}'_2, \ldots, \mathcal{D}'_k$ respectively. DeEPs discovers Emerging Patterns (represented by borders) with respect to $\mathcal{D}'_1$ and $(\mathcal{D}'_2 \cup \ldots \cup \mathcal{D}'_k)$, those EPs with respect to $\mathcal{D}'_2$ and $(\mathcal{D}'_1 \cup \mathcal{D}'_3 \cup \ldots \cup \mathcal{D}'_k)$, those with respect to $\mathcal{D}'_3$ and $(\mathcal{D}'_1 \cup \mathcal{D}'_2 \cup \mathcal{D}'_4 \cup \ldots \cup \mathcal{D}'_k)$, etc. Then the compact scores for k classes are calculated. The class with the largest compact score is chosen.

2.4. Meta-learning

Meta-learning can be described as learning from information generated by a learner(s) [29]. We may also say that it is learning of meta-knowledge from information on lower level.

Meta-learning may use several classifiers trained on different subsets of the data and every sample is classified by all trained classifiers. Different classification algorithms may be used. The classifiers are trained on the training set or its subsets and then predicted classes are gathered from these classifiers. To choose a final class, *simple voting* or *weighted voting* is used [110]. In simple voting, all voters, e.g., classifiers, are equal and have the same strength of their

vote. In weighted voting, voters may have different strength of their votes, weights. To find a final decision, weights are used. Simple voting is a special case of weighted voting, where all weights are equal.

This approach is effective for "unstable" learning algorithms for which a small change in a training set gives significantly different hypothesis [29] [26]. These are, e.g., decision trees, decisions rules [27, 47].

The most popular meta-learning algorithms are *bagging* and *boosting*.

2.4.1. Bagging

Bagging [26] is a method for generating multiple classifiers from the same training set. A final class is chosen by, e.g., voting.

Let $\mathcal{T}$ be the training set with n labelled samples and C be the classification algorithm, e.g., decision tree.

We learn k base classifiers $cl_1, cl_2, .., cl_k$. Every classifier uses C algorithm and is trained on $\mathcal{T}_i$ training set.

$\mathcal{T}_i$ consists of n samples chosen uniformly at random with replacement from the original training set $\mathcal{T}$. The number of samples may be also lower than the number of records in the original training set and be usually in the range of $\frac{2}{3}n$ and n.

Every trained classifier gives his prediction for a sample and a final class is chosen according to *simple voting* (every voter has the same weight).

2.4.2. Boosting

In *boosting* method [52] (like in *bagging*) a set of k classifiers $cl_1, cl_2, .., cl_k$ is created. Classifiers use C algorithm and are trained on $\mathcal{T}_i$ training sets, which are subsets of an original training set $\mathcal{T}$.

The difference is in the process of choosing the $\mathcal{T}_i$ training sets. In *bagging* samples are drawn according to a uniform distribution. In *boosting* samples misclassified by a previous classifier have a higher probability to be drawn when a training subset is drawn for a next classifier.

An example *boosting* method is *AdaBoost* [52], which we present below.

Let $P_{il},\ i = 1..k, l = 1..n$, be a probability that a sample s_l will be drawn to $\mathcal{T}_i$ from an original training set $\mathcal{T}$, $n = |T|$ is the number of samples in the original training set $\mathcal{T}$.

The probabilities $P_{1l},\ l = 1..n$, i.e., the probabilities that a sample S_l will be drawn to $\mathcal{T}_1$[3] from an original training set $\mathcal{T}$, are equal for all samples:

$$P_{1l} = \frac{1}{n},\ l = 1..n,$$

where n, $n = |T|$, is the number of samples in the original training set $\mathcal{T}$.

For each classifier $cl_i,\ i = 2..k$, the probabilities P_{il} are calculated in the following way: First, the sum SP_i of the probabilities P_{il} for samples for which classifier cl_i gave the wrong answer is calculated:

$$SP_i = \sum_{l:S_l\ is\ missclassified\ by\ cl_i} P_{il},\ l = 1..n.$$

Then α_i fractions are computed:

$$\alpha_i = \frac{1}{2}\log\frac{1 - SP_i}{SP_i},\ i = 1..k.$$

The probabilities $P_{i+1,l},\ i = 1..k-1, l = 1..n$ are modified as follows:

$$P_{i+1,l} = \begin{cases} P_{i,l} \cdot e^{-\alpha_i}; & S_l \text{ is correctly classified by } cl_i \\ P_{i,l} \cdot e^{\alpha_i}; & S_l \text{ is missclassified by } cl_i \end{cases},\ i = 1..k-1, l = 1..n.$$

Then the probabilities $P_{i+1,l},\ i = 1..k-1, l = 1..n$ are normalised.

A training subset $\mathcal{T}_{i+1},\ i = 1..k-1$ is drawn according to probabilities $P_{i+1,l},\ i = 1..k-1, l = 1..n$ and used to train $cl_{i+1},\ i = 1..k-1$ classifier.

A final class is chosen using *weighted voting* with α_i fraction for each classifier.

2.5. Classification Accuracy Measures

In experiments presented in this thesis the accuracy, sensitivity, specificity, precision and F-measure [96] were used.

Let us assume that there are two classes: positive and negative.

True positives (denoted as tp) is the number of positive instances that are classified as positive.

[3] $\mathcal{T}_1$ is used to train a classifier cl_1.

True negatives (denoted as tn) is the number of negative instances that are classified as negative.

False positives (denoted as fp) occur when instances that should be classified as negative are classified as positive.

False negatives (denoted as fn) occur when instances that should be classified as positives are classified as negatives. [25]

Sensitivity, also called *recall rate*, measures the proportion of actual positives which are correctly identified as such:

$$Sensitivity = \frac{tp}{tp + fn},$$

where tp (fp) denotes the number of true (false) positives. tn (fn) is the number of true (false) negatives. *Specificity* measures the proportion of negatives which are correctly identified:

$$Specificity = \frac{tn}{tn + fp}.$$

Precision measures the proportion of samples pointed out by classifier which are correctly identified:

$$Precision = \frac{tp}{tp + fp}.$$

Accuracy is the percentage of correctly classified samples:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}.$$

F-measure is a weighted harmonic mean of precision and recall. The general formula of F-measure is as follows:

$$F_\alpha = \frac{(1 + \alpha^2) \cdot precision \cdot recall}{\alpha \cdot precision + recall},$$

where $\alpha > 0$.

When $\alpha = 1$, recall and precision are evenly weighted:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

For a class attribute with multiple (k) values the aforementioned measures are calculated separately for each value of the class attribute, that is, for class $\mathcal{C}_i, i = 1, \ldots, k$, positive class corresponds to class $\mathcal{C}_i$ and negative class to all remaining classes.

Then macro-average is used in order to compute collective measures:

$$Macromeasure = \frac{1}{k} \sum_{i=1}^{k} measure.$$

3. Privacy Preserving Association Rules and Classification Literature Review

This chapter presents literature relevant in the context of privacy preserving association rules mining and classification.

3.1. Levels of Privacy Preserving

In privacy preserving data mining there are two levels of incorporating privacy [121], namely,

— aggregate level,

— individual level.

3.1.1. Aggregate Level of Privacy Preserving

In the case of preserving privacy on an aggregate level, an owner of data does not want any miner to discover all or part of knowledge/relations hidden in a published data set. For instance, in the case of association rules mining, an owner does want to hide particular rules and let a miner to discover the remaining rules.

3.1.2. Individual Level of Privacy Preserving

In the case of preserving privacy on an individual level, individual values of users' (objects') characteristics (values of attributes) are preserved. A miner is able to discover hidden knowledge, e.g., to build a model, however, exact objects' characteristics (e.g., true vales of an attribute *Salary*) are not provided.

3.2. Types of Data Partitioning in Privacy Preserving Data Mining

There are four types of data partitioning in privacy preserving data mining, namely,

— horizontally partitioned,

— vertically partitioned,

— arbitrary partitioned,

— centralised.

The types strictly determine the algorithms used in a given case.

3.2.1. Horizontally Partitioned

In the scenario with horizontally partitioned data [63], there are several sites having the same characteristics (i.e., the same attributes describing objects, e.g., clients) about different objects/people. An example of the same characteristics collected over many sites are supermarkets from the same chain that have the same product assortment and gather information about transactions of their clients.

3.2.2. Vertically Partitioned

Having different information (attributes) about the same objects (clients) collected over many sites, data is partitioned (distributed) vertically [112]. For instance, a hospital may gather information about the same people from a given corporation as an insurance company, which cooperates with this corporation.

3.2.3. Arbitrary Partitioned

The two types of data presented above can be combined together. Thus, not only can information about objects (attributes) be partitioned, but both attributes and objects are distributed over many sites. This more generalised partitioning of data is called arbitrary partitioning [117].

3.2.4. Centralised

When there is only one site that collects information (all attributes for all objects), then data is not partitioned and is called centralised, i.e., stored in one database [123].

3.3. Methods of Data Modification in Privacy Preserving Data Mining

The modification methods, in general, are used to distort values of objects' characteristics and to incorporate a desired level of privacy. Only distorted values are revealed.

3.3.1. Randomisation-based Methods

Randomisation-based methods (perturbation) are one type of the distortion methods. They are used to modify original values at random. In this scheme, only distorted values are stored in a centralised database.

Binary Attributes

A basic randomisation-based method for distorting binary attributes modifies original values in the following way:

Given a binary attribute with possible values of 0 and 1, each (original) value is kept with the probability p or flipped with the probability $1-p$ [98, 9]. All attributes are distorted in the same manner, however, each attribute may have a different value of the probability p. Distorted values of binary attributes create a new database and are supplied to a miner. The only information a miner gets is a distorted database and a value of probability p for each attribute.

Definition A *randomisation factor* is a probability that an original value of an attribute will be retained during a distortion.

A result of a distortion process is a realisation of probabilistic function of an original database that constitutes a distorted database. A miner knows both a distorted database as well as a distorting procedure (a randomisation factor p in this scenario). However, a miner does not know an original database.

In the randomisation-based method presented above, the distortion process is applied to each item in a transaction independently. Moreover, the distortion process for a transaction T_i does not use any information about a transaction T_j, where $i \neq j$. This makes the process of distorting a true database independent for each object. Thus, collecting of all true data to distort all transactions is not necessary because each transaction can be distorted separately. Furthermore, additional distorted transactions can be added to a central distorted database at any time and a data mining process can be repeated over the whole collected distorted data.

Nominal Attributes

The following distortion method for nominal attributes was proposed in [7]: an original value of an attribute is kept with a probability p or changed with a probability $1-p$. Remaining nominal attributes are distorted in the same way, however, each attribute may have a different value of a probability p.

We will specify the process of changing a value in more detail. One of possible solutions is to assign the same probabilities for all values except for an original value and draw a new value, that is, for an attribute with k values, the probability for an original value is p and for other values is equal to $\frac{1-p}{k-1}$.

In general, for nominal attributes we may define $\mathbf{P}$ matrix of retaining/changing values of an attribute [12, 14].

Definition $\mathbf{P}$ is a matrix of retaining/changing values of a nominal attribute of order k x k:

$$\mathbf{P} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,k} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k,1} & a_{k,2} & a_{k,3} & \cdots & a_{k,k} \end{pmatrix},$$

where $a_{r,p} = Pr(v_p \rightarrow v_r)$ is a probability that a value v_p will be changed to a value v_r and the sum of all elements in each column is equal to 1.

Values of a nominal attribute are distorted according to the probabilities from $\mathbf{P}$ matrix. There is a special type of $\mathbf{P}$ matrix, where $a_{r,r} = p$ and the probabilities of changing a value of an attribute are equal. The matrix of this type for an attribute with k values will look as follows:

$$\mathbf{P} = \begin{pmatrix} p & \frac{1-p}{k-1} & \cdots & \frac{1-p}{k-1} \\ \frac{1-p}{k-1} & p & \cdots & \frac{1-p}{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1-p}{k-1} & \frac{1-p}{k-1} & \cdots & p \end{pmatrix}.$$

A year after $\mathbf{P}$ matrix of retaining/changing values of an attribute was proposed in [12] the solution with the same functionality called Random Substitution Perturbation in [42] and Random Replacement Perturbation in [43] was presented.

Continuous Attributes

There are three main methods of distorting continuous attributes which do not assume any knowledge about values of attributes of other objects: the additive perturbation method [7], multiplicative perturbation [66], and the retention replacement perturbation [8].

The *additive perturbation method* is also called *value distortion method* [7]. In this method a random value drawn from a given distribution, e.g., a uniform or normal distribution, is added to an original value of an attribute. Only a modified value is revealed to an external organisation. There are two main distortion distributions:

— uniform distribution - the random variable has the uniform distribution between $\langle -\alpha, \alpha \rangle$ and the mean equal to 0,

— normal distribution - the random variable has the normal distribution with mean equal to 0 and standard deviation σ.

Table 3.1. The example of the original database.

Id	Salary	Age	Sex	Previous credits	Bad credit
1	1000	35	M	none	N
2	1500	37	F	overdue	Y
3	5000	41	M	present	N
4	3000	44	M	repaid	N
5	4200	50	F	repaid	N
6	2000	28	F	none	N
7	1000	30	M	none	Y

Table 3.2. The example of the distorted database with uniform distortion distribution $\langle -500, 500 \rangle$ for Salary, $\langle -10, 10 \rangle$ for Age and $p = 0.6$ for Sex and Previous credits attributes.

Id	Salary	Age	Sex	Previous credits	Bad credit
1	1353.32	33.42	M	repaid	N
2	1611.83	40.64	M	overdue	Y
3	5428.27	51.27	M	present	N
4	2573.22	39.51	F	none	N
5	4145.89	42.67	M	repaid	N
6	2258.34	38.72	F	none	N
7	1054.03	36.65	M	overdue	Y

The additive perturbation method is considered as preserving little privacy in some cases [64], for instance, when age in range $\langle 18, 65 \rangle$ is randomised by adding a random value from the uniform distribution in range $\langle -18, 18 \rangle$ and the distorted value is equal to 0, then it is known that the true age is 18. Moreover, a random matrix-based spectral filtering technique can be used to retrieve original data from a data set distorted by adding random values [64]. However, for high level of privacy the estimation becomes too erroneous. Furthermore, if a data set has a random component in it and random noise is added to distort data, the spectral filtering method does not filter the actual data accurately.

In the *multiplicative perturbation* an original value of an attribute is multiplied by a random value drawn from a given distribution [66]. To distort values of an object, the rotation perturbation [31] where the vector of values of attributes for a given object is multiplied by a rotation matrix can also be used. Nevertheless, when one is able to estimate the rotation matrix, it might be possible to breach the privacy [76].

Considering the aforementioned properties of the additive and multiplicative perturbation methods, we also describe the retention replacement perturbation and use this perturbation in algorithms presented in the thesis.

In the *retention replacement perturbation* [8], an original value of an attribute is kept with a probability p and with a probability $1 - p$ an original value is replaced with an element selected from a replacing probability distribution function (pdf) on a domain of the attribute. A retention replacement perturbation where the replacing pdf is a uniform pdf is called a uniform perturbation.

Example distortion

Table 3.1 presents the example of the original database. It consists of the continuous attributes: *Salary* and *Age*, the nominal attribute: *Previous credits*, the binary attributes: *Sex*, *Bad credit*. Table 3.2 shows the example of the distorted database. The continuous attributes were distorted by means of the additive perturbation with the uniform distortion distribution in the range $\langle -500, 500 \rangle$ for *Salary* and $\langle -10, 10 \rangle$ for *Age* attribute. The attributes *Sex* and *Previous credits* where distorted with the probability of retaining an original value equal to 0.6. The attribute *Bad credit* was not distorted. In the scheme where randomisation-based methods are used, only the distorted database and the parameters of the distortion process are revealed.

3.3.2. Blocking

A value of an attribute can be changed by replacing it with an *unknown value*, a value that does not occur in domains of attributes (often represented as '?'). Blocking replaces an original value with an unknown value instead of placing a false value, which is sometimes more desirable, e.g., for medical applications [101]. Applying this method, privacy can be preserved at both aggregate and individual levels, however, blocking is popular in hiding association rules [101, 1].

3.3.3. Aggregation

In this method, values of an attribute are aggregated to form a broader group. In [7], the *value-class membership method* was defined where values of an attribute are partitioned into disjoint mutually-exclusive classes which become new values of a modified attribute. A special case of aggregation is discretisation in which values of a continuous attribute are discretised into intervals. Instead of an original value, an interval in which an original value lies is provided.

3.3.4. Swapping

A value of an attribute for a given sample can be interchanged with a value of the same attribute for a different sample. The drawback of this method is that original values for different samples should be known to interchange values for different samples, thus swapping cannot be performed separately for each sample without knowledge about original values for other samples and original values have to be stored.

3.3.5. Sampling

In this method, only a sample of population is revealed. To discover hidden knowledge, the sample data should reflect relations among all data, that is, probability distribution functions of attributes should be preserved. Sampling does not modify values stored in a database, thus a data mining process is performed on real data, which eliminates the problem of building a model on distorted data [101]. Nevertheless, revealing true data, even for a few samples from a true database, may be a serious drawback in some cases, for instance, medical applications, where all true data about some patients would be revealed.

In order to avoid this drawback, that is, not to reveal true data about objects, the sample can be generated according to probability distribution functions of attributes [37].

3.4. Privacy Preserving Techniques

In privacy preserving data mining, the following three main techniques are used:

— heuristic-based,

— cryptography-based,

— randomisation-based.

3.4.1. Heuristic-based Techniques

A selective modification of data is an NP-Hard[1] problem [121]. To address the complexity issue, heuristic algorithms can be used. This solution is especially popular in hiding given association rules [122, 74, 83]. Heuristic-based techniques are mostly used for centralised data.

[1] The proof in the case of privacy preserving association rules mining can be found in [21].

3.4.2. Cryptography-based Techniques

Secure Multiparty Computation, which is the *cryptography-based technique*, can be used to solve the following problem: two or more parties want to create a model based on their private input, but they are not willing to disclose their private data to anybody else. This problem is called *Secure Multiparty Computation* (SMC). Computations are performed in a distributed network on inputs from each participant. The assumption is that an output of a computation is correct and no more information is revealed to any participant than its own input and output. *Secure two party computation* was first investigated in [128]. Later it was generalised to secure multiparty computation in [57, 30, 24]. *Secure multiparty computation* is used for partitioned data.

Moreover, encryption techniques which enable one to perform computations over encrypted data without being able to decrypt can be used in privacy preserving [55, 120].

3.4.3. Reconstruction-based Techniques

The idea behind *reconstruction-based techniques* is that having only data distorted by one of the randomisation-based methods, a miner reconstructs (estimates) an original distribution of attributes and based on the reconstructed (estimated) distributions builds a model of aggregated data [7]. The reconstruction-based techniques are used for centralised data.

For the purpose of this thesis we define a reconstructed (or estimated) distribution of an attribute and a reconstructed (or estimated) support of an itemset in the following way:

Definition A *reconstructed (or estimated) distribution* of an attribute A is an original (true) distribution of the attribute A reconstructed (estimated) based on values of the attribute A distorted by means of a randomisation-based method.

Definition A *reconstructed (or estimated) support* of an itemset I is an original (true) support of the itemset I in the original set $\mathcal{T}$ of transactions reconstructed (estimated) based on transactions from the set $\mathcal{D}$ distorted by means of a randomisation-based method.

There are two main scenarios in privacy preserving data mining tasks conducted with the usage of the reconstruction-based technique on data distorted according to a randomisation-based method which distorts values of each attribute in a sample independently and does not use any information about other samples. In both scenarios, privacy is preserved on an individual level and data is stored in a centralised database.

These two scenarios have in common that an untrusted miner knows only a distortion procedure, its parameters and a distorted data set.

The first scenario can be used in surveys, especially Internet surveys. An original answer of a user is distorted according to a distortion procedure with known parameters and only distorted values of an answer are sent to a centralised database, where they are stored. An original answer is not stored at all. In this case, both a miner and a customer know a distorting procedure and its parameters.

Having collected distorted answers from users, a miner is able to build a model using reconstruction-based techniques and a data mining task can be repeated many times by a miner.

In this scenario a distortion procedure may be implemented in an application on a customer side that distorts a customer's answer and sends only distorted values to a central collector. This application may be built in a web browser in the case of Internet surveys.

Furthermore, additional distorted customers' answers can be added to a central distorted database at any time and a data mining process can be repeated over the whole collected data.

The second scenario assumes that an original data set is collected and owned by an authorised organisation. The collected data needs to be shared with a foreign untrusted organisation, for instance, when an authorised organisation wants to have a data mining task performed. An authorised organisation distorts an original data set using a randomisation-based method and passes a distorted data set, a description of a randomisation-based method and its parameters to a foreign organisation. An original data set is not shared.

A foreign organisation knows parameters of a randomisation-based method and a distorted data set, thus it can perform a data mining task on all received data or its part. The results of a data mining task, e.g., a decision tree, performed by a foreign organisation can be passed to an authorised organisation. An authorised organisation can use the results to classify its customers.

In both scenarios an untrusted miner knows only a distortion procedure, its parameters and a distorted data set. The result of a data mining task performed by an untrusted miner is a model built on an aggregate level, e.g., a classifier.

3.5. Privacy Measures

Since privacy preserving data mining was introduced, several privacy measures have been proposed in literature. We will describe the most important ones.

3.5.1. Basic Privacy

In Basic Privacy (BP) [98] [9], which was originally considered in the case of privacy preserving association rules mining, a miner does not have an access to a distorted database after a mining process is performed. The basic privacy represents the probability that an original entry of a given random customer for an item i can be accurately reconstructed from a distorted database (before a mining process). This privacy measure P in the case when an item is present in a true database can be calculated as follows (details are available in [98] and [10]):

$$P = 1 - \frac{p^2 s_0}{s_0 p + (1 - s_0)(1 - q)} - \frac{(1 - p)^2 s_0}{s_0(1 - p) + (1 - s_0)q} \ ,$$

where:

— s_0 is the average true support of individual items in a database,

— p denotes a randomisation factor (the same for all items) in the case when an item is present in a true database,

— q denotes a randomisation factor (the same for all items) in the case when an item is not present in a true database.

3.5.2. Reinterrogation Privacy

In Reinterrogated Privacy (RP) [9], a miner can use an output of a mining process and *re-interrogate* a distorted database to reduce privacy.

A miner can use knowledge of an output of a mining process, e.g., association rules or supports of frequent itemsets, to reduce privacy of an individual customer's entry. Having a frequent itemset and its support in a distorted database as well as its estimated support in a true database, a miner may infer what an original customer's entry is, i.e., whether the customer bought an item or not.

Example Let p denote for a given attribute the randomisation factor in the case when an item is present in a true database and q when an item is not present. Let $p = q = 0.95$ and the reconstructed support of the itemset $\{a, b, c\}$ is 0.005. The probability that an original transaction does not contain any item from the itemset $\{a, b, c\}$ is very low, $0.05 * 0.05 * 0.05 * 0.995 = 0.000124375$. Thus, the probability that the original transaction contains at least one item from $\{a, b, c\}$ is $1 - 0.000124375 = 0.999875625$ and suggests that at least one item from this itemset is present in the original transaction.

□

Table 3.3. Agrawal-Srikant privacy measure for different distortion distributions.

Distortion method	Confidence level		
	50%	95%	99,9%
Discretisation	0,5 * W	0.95 * W	0,999 * W
Uniform	0,5 * 2α	0.95 * 2α	0,999 * 2α
Normal	1,34 * σ	3,92 * σ	6.58 * σ

As shown in the example, a miner knowing a reconstructed support of itemsets can predict with high probability that at least one item from a given itemset is in an original transaction. The procedure of calculating the reinterrogation privacy is described in [10].

3.5.3. Agrawal-Srikant Privacy Measure

A measure proposed in [7] is based on how closely original values of a modified random variable can be estimated.

Definition The *Agrawal-Srikant privacy measure* at $c\%$ confidence level is the length of the interval $(x_2 - x_1)$ if an original value of an attribute can be estimated with $c\%$ confidence that a value x lies in the interval $\langle x_1, x_2 \rangle$.

Table 3.3 shows the calculated values of Agrawal-Srikant privacy measure for different modification methods (discretisation with equal lengths of intervals, value distortion method with uniform and normal distortion distributions) and levels of confidence. W is the length of intervals in discretisation of a domain of an attribute. The parameter of a uniform distribution α and the standard deviation σ are defined in Section 3.3.1.

In order to preserve privacy at high level, one should be interested in high privacy levels, i.e., 25%, 50%, 100%, and 200% of a domain range of an original attribute.

3.5.4. Privacy Based on Differential Entropy

The definition of privacy based on the *differential entropy* was proposed in [2] and is defined as follows:

Definition *Differential entropy of a random variable* A is:

$h(A) = -\int_{\Omega_A} f_A(a) \log_2 f_A(a) da,$

where Ω_A is a domain of a variable A and f_A is a density function of a variable A.

Definition *Privacy (level) inherent in a random variable* A is:

$\Pi(A) = 2^{h(A)},$

where $h(A)$ is the *differential entropy* of variable A.

A random variable A distributed uniformly between 0 and a has privacy (level) equal to a. For general random variable C, $\Pi(C)$ denotes the length of the interval over which a uniformly distributed random variable has the same privacy as C.

3.5.5. Range Privacy

We introduced *Range Privacy* in [17, 16]. $n\%$ *Range Privacy* for a random variable A means that we use a distorting random variable Y with privacy measured according to the definition based on differential entropy (Section 3.5.4) equal to $n\%$ of the domain range of values of the random variable A, for instance, to achieve 100% level of privacy for a random variable A with the range of its values equal to 10, a distorting random variable Y should have privacy measure equal to 10 (e.g., for the uniform distribution, a random variable distributed between -5 and 5 can be used).

3.6. Privacy Preserving Association Rules Mining

Since the notion of Privacy Preserving Data Mining was introduced, association rules mining with incorporated privacy has been widely discussed [98, 101, 21, 34, 102, 112, 61, 50].

Proposals presented in [101, 21, 34, 102] show how to prevent given rules from being discovered by a miner. The possible solutions for hiding given association rules are to falsify some tuples or replace original values with unknowns. In order to perform hiding task, a complete original database is required as a starting point.

Cryptographic techniques in privacy preserving association rules mining for individual values in distributed data were considered, e.g., in [112] [61]. In these works databases are distributed across a number of sites and each site is willing to share only mining process results, but does not want to reveal the source data. Currently available techniques for distributed database require a corresponding part of a true database at each site.

[112] deals with association rules mining over data vertically partitioned across two organisations. To compute a support of an itemset (for each transaction a part of items is possessed by one organisation and the rest by the other organisation), multi-party computation techniques are used. The base operation, claimed to be secure, is a calculation of a scalar product, which is used to compute, in a secure way, more complex statistics, e.g., a support. The authors of [112] claim that the secure computation of scalar products does not reveal exactly which transactions support a subset of an itemset.

Association rules can be mined over horizontally partitioned data, e.g., according to the algorithm presented in [61]. It incorporates cryptography techniques to provide, as authors stated, a secure union of locally frequent itemsets and testing support threshold without revealing support count. These two techniques are used to find frequent sets hidden in transactions partitioned across different organisations.

[32] presents, as authors stated, four secure multiparty computations: the secure sum, the secure set union, the secure size of a set intersection, and the scalar product to be used in distributed privacy preserving data mining. Based on these methods algorithms for association rules mining for both vertically and horizontally partitioned data were presented in the aforementioned paper.

A framework for mining association rules from a centralised distorted database was proposed in [98]. A scheme called MASK (detailed description of MASK can be found in Section 3.6.1) attempts to simultaneously provide a high degree of privacy to a user and retain a high degree of accuracy in the mining results. To address efficiency, several optimisations for MASK were originally proposed in [98]. The main optimisation, which reduces time complexity, requires randomisation factors to be constant for all items. This is the most important disadvantage of this optimisation, because it does not allow using different randomisation factors for different items. Non-uniform randomisation factors help to achieve higher accuracy [124] because people have different privacy concerns about different attributes.

Another optimisation, called EMASK, was proposed in [9]. In general, EMASK does not break the exponential complexity of reconstructing an original support with respect to the length of an intemset and does not allow different randomisation factors when an item is present in an original database and when it is not.

In [124], a general framework for privacy preserving association rule mining was proposed. It allows attributes to be randomised using different randomisation factors, based on their privacy levels. In that work, it was also theoretically proven that the use of non-uniform randomisation factors can lead to more accurate mining results than the use of one randomisation factor. The empirical experiment results also verified this statement. An efficient algorithm RE, Recursive Estimation (details about RE algorithm can be found in Section 3.6.2) for mining frequent itemsets under this framework was developed in [124] also. The RE algorithm uses different randomisation factors, but it does not break an exponential complexity in estimating a support.

In the next subsections, we will describe in more detail solutions for a centralised database, i.e., MASK framework and Recursive Estimation algorithm.

3.6.1. Mining Associations with Secrecy Konstraints (MASK) Scheme

In this section, we present basic information about the MASK (Mining Associations with Secrecy Konstraints[2]) scheme [98] for Privacy Preserving Data Mining over centralised data.

Original Distortion Procedure in MASK Scheme

For distorting a transactional data set in original MASK scheme a basic randomisation method for binary attributes described in 3.3.1 was used.

Estimating Singleton Support

Let $\mathcal{T}$ be a true data set[3] represented by a matrix $\mathbf{T}$. We denote a distorted data set, obtained accordingly to the distortion procedure for binary attributes presented in Section 3.3.1, as $\mathcal{D}$ and its matrix representation as $\mathbf{D}$.

Now we will focus on an i-th item. Let C_1^T and C_0^T be the numbers of 1's and 0's, respectively, in i-th column of $\mathbf{T}$ (1 means that an item is present in a transaction and 0 means that an item is not present). C_1^D and C_0^D denote the numbers of 1's and 0's, respectively, in i-th column of $\mathbf{D}$.

A support of an i-th item in the true matrix $\mathbf{T}$ can be estimated based on a support of this item in $\mathbf{D}$ using the following equation:

$$\mathbf{C}^T = \mathbf{M}^{-1}\mathbf{C}^D, \tag{3.1}$$

where

$$\mathbf{M} = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix}, \qquad \mathbf{C}^D = \begin{bmatrix} C_1^D \\ C_0^D \end{bmatrix}, \qquad \mathbf{C}^T = \begin{bmatrix} C_1^T \\ C_0^T \end{bmatrix}.$$

$\mathbf{M}$ is a transition matrix that represents probabilities that a given value of an attribute (or values of a combination of attributes) is changed to a different value (or values) or retained.[4] If a column in the matrix $\mathbf{T}$ has n 1's, approximately np 1's and $n(1-p)$ 0's in the matrix $\mathbf{D}$ for the same column will be obtained. And similarly, when a column in the matrix $\mathbf{T}$ has m 0's,

[2] The authors use Konstraints instead of Constraints to achieve abbreviation: MASK.

[3] In real applications a true data set is not stored. Only distorted tuples are collected.

[4] $\mathbf{M}$ is more general than $\mathbf{P}$ matrix for a binary attribute because in $\mathbf{M}$ values of combinations of attributes can be used ($\mathbf{M}$ matrix for a combination of attributes is shown Equation 3.3). We denote this matrix as $\mathbf{M}$ (even for a binary attribute) to emphasize that a combination of attributes can be used.

these will generate approximately mp 0's and $m(1-p)$ 1's for the same column in the matrix $\mathbf{D}$.

Given the number of 0's and 1's (values $C_0^{\mathbf{D}}$ and $C_1^{\mathbf{D}}$ respectively) in the distorted data set, it is possible to estimate the number of 1's (value $C_1^{\mathbf{T}}$), the support of an i-th item in the true data set.

Estimating n-itemset Support

Equation 3.1, which is applicable to singletons, can be extended to compute the support of an n-itemset. The matrices $\mathbf{C^D}$ and $\mathbf{C^T}$ are defined in a more general way:

$$\mathbf{C^D} = \begin{bmatrix} C^D_{2^n-1} \\ . \\ . \\ . \\ C^D_1 \\ C^D_0 \end{bmatrix}, \quad \mathbf{C^T} = \begin{bmatrix} C^T_{2^n-1} \\ . \\ . \\ . \\ C^T_1 \\ C^T_0 \end{bmatrix}. \tag{3.2}$$

$C_k^{\mathbf{T}}$, respectively $C_k^{\mathbf{D}}$, is the number of tuples in $\mathbf{T}$, respectively $\mathbf{D}$, matrix that have a binary form of k (in n bits) for a given itemset. For a 2-itemset C_0^T refers to the number of 00's and C_2^T to the number of 10's.

The matrix $\mathbf{M}$ is defined as follows:

$$\mathbf{M} = \begin{bmatrix} m_{0,0} & m_{0,1} & m_{0,2} & \dots & m_{0,2^n-1} \\ m_{1,0} & m_{1,1} & m_{1,2} & \dots & m_{1,2^n-1} \\ \vdots & & & \ddots & \vdots \\ m_{2^n-1,0} & m_{2^n-1,1} & m_{2^n-1,2} & \dots & m_{2^n-1,2^n-1} \end{bmatrix}, \tag{3.3}$$

where $m_{i,j}$ is a probability that a tuple of the form C_j^T in the matrix $\mathbf{T}$ goes to a tuple of the form C_i^D in $\mathbf{D}$.

For instance, $m_{1,2}$ for a 2-itemset is the probability that tuple *10* is distorted to tuple *01* during the distortion process and $m_{1,2} = (1-p)(1-p)$, if p is the same for considered items. The value of $m_{1,2}$ results from the change made for both items ($1-p$ probability was used) and the independent distortion for both items (multiplication of the probabilities was used).

Having generalised matrices $\mathbf{C}^D$ and $\mathbf{C}^T$, the same Equation 3.1 can be used to estimate the support of an n-itemset.

In general (without the assumption that the value of p is the same for all items), MASK

scheme needs an exponential number of counters (2^n counters for an n-itemset) and makes the process infeasible in practice [9].

Process of Mining Frequent Itemsets

To mine all itemsets frequent in the undistorted data set $\mathcal{T}$ given only the distorted data set $\mathcal{D}$, the Privacy Preserving Apriori-MASK (PPApriori-MASK) algorithm can be used (please, see Algorithm 5). The support counting procedure, as well as the Apriori algorithm, should be modified and use the MASK scheme for estimating supports of candidates for frequent itemsets (please, see Algorithm 6). The function $aprioriGen$ stays the same as in the original Apriori (please, refer to Algorithm 2).

We will use the following notation for MASK:

— $\mathcal{X}_m$ denotes candidate m-itemsets, which are potentially frequent.

— $\mathcal{F}_m$ are frequent m-itemsets based on estimated original support.

— $X[i]$ is the i-th item in the itemset X.

— $X[1]{\cdot}X[2]{\cdot}X[3]{\cdot}\ldots{\cdot}X[m]$ denotes m-itemset, which consists of $X[1], X[2], X[3], \ldots, X[m]$.

— $\mathcal{T}$ is the original data set.

— $\mathcal{D}$ is the data set distorted according to the MASK scheme and each item i is distorted with the matrix $\mathbf{M}_i$.

— $X.\mathbf{C}^D$ means the support vector field of the itemset X in the distorted data set $\mathcal{D}$.

— $X.\mathbf{C}^T$ means the support vector field of the itemset X in the true data set $\mathcal{T}$.

— $X.\mathbf{C}^T_j$ means the j-th element of the support vector $\mathbf{C}^T$ of the itemset X.

— $X.\mathbf{C}^D_j$ means the j-th element of the support vector $\mathbf{C}^D$ of the itemset X.

— $X.\mathbf{M}$ is $\mathbf{M}$ matrix (see Equation 3.3) for the itemset X.

Algorithm 5 The PPApriori-MASK algorithm, the Apriori algorithm modified to use MASK

input: $minimumSupport$
input: $\mathcal{D}$ // binary distorted data set
$\mathcal{F}_1$ ={1-itemsets which are frequent based on estimated original support of singletons}
for $(m = 2; \mathcal{F}_{m-1} \neq \emptyset; m++)$ **do begin**
$\mathcal{X}_m$ = $aprioriGen(\mathcal{F}_{m-1})$ //generate new candidates
$supportCount(\mathcal{X}_m)$
$\mathcal{F}_m = \{X \in \mathcal{X}_m | X.\mathbf{C}^T_{2^m-1} \geq minimumSupport\ \}$
end
return $\bigcup_m \mathcal{F}_m$

The PPApriori-MASK algorithm generates candidates for frequent sets with a given length in the Apriori-like fashion and uses the MASK scheme to estimate an original support of a candidate in the true database based on a support counted in distorted transactions. Then, the

Algorithm 6 The support count algorithm for MASK scheme

```
procedure supportCount(var X_m)
 for all transactions T ∈ D do begin
  for all candidates X ∈ X_m do begin
   X.C_j^D ++      //j is the number which has a binary form (in m bits) of X
                   // in the distorted transaction T
  end
 end
 for all candidates X ∈ X_m do begin
  X.C^T = (X.M^{-1})(X.C^D)
 end
end
```

minimum support condition is checked based on the estimated supports of candidates. As the result of the PPApriori-MASK algorithm, the itemsets with the estimated support greater than or equal to $minimumSupport$ are provided.

Distorting Data

The distorting method discussed in Section 3.6.1 is the one of the simplest randomisation methods. It assumes the same value of p for all attributes.

Different Privacy for Attributes

In a more general scenario each item has a different probability which is used to decide whether to change the value (0 or 1) or not while conducting the distortion process. $p_1, p_2, ..., p_k$ are parameters of this process, where k is the number of different items in a database.

The motivation for different probabilities for each item is that people usually have different privacy concerns about different items [124]. For instance, sensitivity of questions varies in a survey, i.e., values of different attributes are of different importance to users. Information about gender and age is usually not as sensitive as income [125][5]. People can accept lower privacy for less sensitive information. Thus, higher accuracy can be achieved because of the trade-off between accuracy and privacy. Less privacy for less sensitive attributes will incur the increase of accuracy [125].

Moreover, different people can have different privacy concerns about the same attribute. Despite this variety the method which assumes different privacy for attributes does not take into account these differences. It only allows different attributes to have different randomisation factors, but values of the same attribute cannot have different randomisation factors.

[5] It does not mean that there are no people who prefer more privacy for age than income. But the number of these people is small.

Method with Different Randomisation Factors for 0's and 1's Values for Each Attribute

Randomisation factors in range 0.7 - 0.9 make the number of 1's to increase in a database. As a result of this growth the time of performing a data mining process increases [9]. Thus, the distortion process (especially the parameters of this process, i.e., $\mathbf{M}$ matrix) influences the processing time. The less randomisation factor is, the more time takes the algorithm to mine frequent itemsets.

As the distortion is a culprit, a method with different randomisation factors for 0's and 1's can be used to avoid the growth of processing time.

High randomisation factor for 0's (for example 0.96) prevents from rapid growth of the number of 1's in a distorted database. Relatively low probability for 1's (0.3; 0.7) lets a miner maintain the desired level of privacy [9].

Let p_i denote for a given attribute A_i the randomisation factor for 1's and q_i for 0's. Having p_i and q_i as the parameters of the distortion process, M_i matrix has the following elements:

$$\mathbf{M_i} = \begin{bmatrix} p_i & 1-q_i \\ 1-p_i & q_i \end{bmatrix}.$$

Example Let us assume that a matrix $\mathbf{M_i}$ for a binary attribute A_i is as follows:

$$\mathbf{M_i} = \begin{bmatrix} 0.4 & 0.04 \\ 0.6 & 0.96 \end{bmatrix},$$

and a matrix $\mathbf{C_i^T}$ for the attribute A_i calculated based on an original matrix $\mathbf{T}$ is equal to:

$$\mathbf{C^T} = \begin{bmatrix} 100 \\ 1900 \end{bmatrix}.$$

A matrix $\mathbf{C_i^D}$, which shows the number of 1's and 0's for the distorted attribute A_i, can be estimated in the following way:

$$\mathbf{C^D} = \mathbf{M} \cdot \mathbf{C^T} = \begin{bmatrix} 0.4 & 0.04 \\ 0.6 & 0.96 \end{bmatrix} \cdot \begin{bmatrix} 100 \\ 1900 \end{bmatrix} = \begin{bmatrix} 0.4 \cdot 100 + 0.04 \cdot 1900 \\ 0.6 \cdot 100 + 0.96 \cdot 1900 \end{bmatrix} = \begin{bmatrix} 116 \\ 1884 \end{bmatrix}.$$

Due to the high probability for 0's equal to 0.96, the number of 1's in the distorted matrix $\mathbf{D}$ is slightly higher than the number of 1's in the original matrix $\mathbf{T}$ and it will not significantly increase the time of mining frequent itemsets. Moreover, low probability for 1's, that is, high

privacy level, causes that only 40% of 1's in the original matrix $\mathbf{T}$ is present in the distorted matrix $\mathbf{D}$.

□

Methods Summary

We have described the following three methods of distorting the data:

1. Method with the same randomisation factors for all attributes [98],
2. Method with different randomisation factors for attributes [9],
3. Method with different randomisation factors for 0's and 1's values for each attribute [124, 125].

The last method is the most general. The two prior methods are special cases of the last one. Note that the first method can be viewed as a special case of the third wherein $p_i = q_i = p$. The second method is a special case of the third when $p_i = q_i$.

3.6.2. Recursive Estimation

The *recursive estimation* (RE) algorithm [124] estimates a support of itemsets in a centralised database distorted with the randomisation-based method. The most generalised method with different randomisation factors for each item and the value of this item can be used in this scheme.

Let p_x is a randomisation factor for value 1 of an item x (value 1 is kept with a probability p_x and flipped with a probability $1 - p_x$) and q_x be a randomisation factor for value 0 of an item x.

Let S_I denote a true support of an itemset I in a true database $\mathcal{T}$ and S'_I represent a support of I in a distorted database $\mathcal{D}$. The RE algorithm calculates an estimate of S_I recursively as follows:

$$\begin{cases} S^{RE}_{\emptyset} = S'_{\emptyset} = |\mathcal{T}| = |\mathcal{D}| \\ S^{RE}_{I} = \frac{S'_I - \sum_{f \subset I} \{S^{RE}_f * \prod_{x \in f}(p_x - (1-q_x)) * \prod_{I \setminus f}(1-q_x)\}}{\prod_{x \in I}(p_x - (1-q_x))} \end{cases} \tag{3.4}$$

An estimate of a true support S^{RE}_I is derived based on the estimates of the true supports of all subsets of I, i.e, $\{S^{RE}_f | f \subset I\}$. Conducting the mining process in a level-wise fashion, i.e., from low level to high, at level k the estimates of the true support for each k-itemset's subset are known and the estimate of the support of k-itemset can be directly computed. Despite this advantage, the RE algorithm iterates through all the subsets of k-itemset and does not break the

exponential complexity. An estimate of a true support S_I^{RE} is the same as an estimate provided by MASK. The variances of RE and MASK unbiased estimates are the same, as well [124].

3.7. Privacy Preserving Classification

Privacy preserving classification has been extensively discussed in literature [7, 75, 2, 46, 127, 139, 126].

A cryptographic approach to privacy preserving classification was proposed in [75]. In that paper, a method for inducing a decision tree for horizontally partitioned data between two parties was presented. The method is based on the ID3 algorithm and cryptographic techniques such as the oblivious transfer protocol and oblivious polynomial evaluation.

For vertically partitioned data a decision tree construction problem was presented in [45]. The solution also assumes only two parties and uses the secure scalar product based on a semi-trusted party.

The cryptographic approach for the ID3 decision tree over data horizontally distributed over two or more parties was introduced in [127]. It allows a miner to compute frequencies of values or tuples of values in customers' data, but without revealing the part of data which is privacy-sensitive. This solution can be applied to fully distributed data (each transaction is provided by different party) contrary to the approach presented in [75]. Moreover, naïve Bayes classifier can be build based on this approach and association rules mined.

For vertically partitioned data distributed over two or more parties a privacy preserving decision tree algorithm based on ID3 was introduced in [114] and [115]. However, the solution for privacy preserving ID3 decision tree learning, which is scalable in terms of computation and communication cost and can be run even for a large number of parties without a need for third parties, was presented in [49].

In [59], as authors stated, a secure protocol to compute the Pseudo-Scalar Product was proposed. It was used to build a decision tree not only over horizontally and vertically but also over arbitrary partitioned data in a privacy preserved way. Moreover, it protects each party's privacy against up to the $n - 2$ corrupted parties.

Later the C4.5 based privacy-preserving decision tree for vertically partitioned data was proposed in [54]. The solution is based on the calculation of the union of the databases of all parties and assumes that one or more parties know the class attribute. However, the third parties are not needed.

A different approach to build a privacy preserving decision tree over vertically partitioned data was proposed in [133]. It uses homomorphic encryption and digital envelope technique.

In privacy preserving classification naïve Bayes classifier has been used also. In [63], it was shown how to use this classifier for horizontally partitioned data. In [138], a scheme based on homomorphic encryption for the same type of partitioning was proposed.

The solution for vertically partitioned data was proposed in [113] and the algorithms for both horizontally and vertically partitioned data were presented in [116]. In [127], the solution for naïve Bayes classifier for horizontally and fully distributed data was presented.

In [62], another privacy preserving classifier, kNN, for horizontally partitioned data was introduced. Later, in [104], an algorithm for computing the nearest neighbors of records, which was based on secure multiparty computation primitives over horizontally distributed data, was proposed and in [105] it was shown how this algorithm can be used in kNN classification. This type of classifier was also discussed in [126].

kNN classifier for vertically distributed data was presented in [134] and [135]. It is based on a secure protocol for multiple parties.

Another privacy preserving classifier, SVM, for vertically partitioned data was proposed in [130]. It utilises the secure matrix addition techniques and distributed SVM. In [60], the matrix factorisation theory was used instead. A different approach to the privacy preserving SVM classifier was also presented in [117]. It was designed for vertically, horizontally and even arbitrarily partitioned data.

To sum up the proposals for distributed privacy preserving data mining, different privacy preserving classification algorithms, a decision tree, naïve Bayes, kNN, and SVM, were proposed for both vertically and horizontally partitioned data. These proposals utilise cryptography techniques.

The pioneer work in privacy preserving classification for centralised data was [7], where R. Agrawal and R. Srikant proposed how to build a decision tree over centralised data distorted with the randomisation-based method (except the target/class attribute) and then classify not distorted data with this decision tree. In this solution, they also presented the algorithm (in this dissertation this algorithm will be called AS) for a probability distribution reconstruction for continuous attributes, which estimates an original probability distribution based on distorted samples (details about the algorithm AS can be found in Section 3.7.2).

Paper [2] extends the AS algorithm and presents the EM reconstruction algorithm, which does not take into account nominal attributes either (for details refer to Section 3.7.2).

Randomised Response technique for related-question model was presented in [46]. It allows creating a decision tree but only for nominal attributes. Randomised Response technique for unrelated-question model was discussed in [136, 137] and applied in building naïve Bayes classifier.

The solution we proposed in [14] differs from those above, because it enables a miner to classify centralised perturbed data containing simultaneously continuous and nominal attributes by meand of randomisation-based methods to preserve privacy on an individual level. This approach uses the EM/AS algorithm (described in details in Section 3.7.2) to reconstruct a probability distribution for nominal attributes and the ARVeSNA algorithm (please refer to Section 3.7.2) for assigning reconstructed values to samples for this type of attributes to build a decision tree simultaneously with continuous attributes.

In [15] we proposed the EQ algorithm (details can be found in Section 3.7.2) for reconstructing a probability distribution of nominal attributes. The algorithm achieves better results, especially for high level of privacy, i.e., low probability of retaining an original value of a nominal attribute.

3.7.1. Decision Tree

To build a decision tree over data containing nominal and continuous attributes distorted by means of randomisation-based methods (except a target/class attribute), the algorithms described in Section 3.7.2 can be incorporated in the standard process of building a decision tree presented in Section 2.2. Classification is performed in the standard way because it classifies undistorted samples. The reconstruction types used in a decision tree building are described in this section.

Reconstruction Types

While using a decision tree as a classifier in privacy preserving, there are four reconstruction types: *Local*, *By class*, *Global* and *Local All* [7] [14]. The reconstruction type *Global* means that a reconstruction of a probability distribution is performed only in a root of a tree. In the case of the *By class* type, a reconstruction is done separately for each class, but only in a root node. For the *Local* type, a reconstruction is performed in every node divided into classes. The *Local all* type means that a reconstruction is used in every node without dividing into classes.

3.7.2. Algorithms for Distribution Reconstruction and for Assigning Reconstructed Values to Samples

The algorithms for distribution reconstruction of both nominal and continuous attributes are described in this section. Moreover, the algorithms for assigning reconstructed values to samples for nominal and continuous attributes are presented. The definition of information loss in reconstruction is introduced, as well.

Information Loss

The lack of precision in the reconstruction of a probability distribution is called information loss. It is defined as follows [2]:

Definition Information loss $\mathcal{I}(f_X, \hat{f}_X)$ equals half of the expected value of L_1 norm between the original probability distribution f_X and its estimate $\hat{f}_X$.

$\mathcal{I}(f_X, \hat{f}_X) = \frac{1}{2}E[\int_{\Omega_X} \mid f_X - \hat{f}_X \mid]$

Information loss $\mathcal{I}(f_X, \hat{f}_X)$ lies between 0 and 1. $\mathcal{I}(f_X, \hat{f}_X) = 0$ means the perfect reconstruction, and $\mathcal{I}(f_X, \hat{f}_X) = 1$ implies that there is no overlap between the original distribution and its estimate.

AS Algorithm for Probability Distribution Reconstruction of Continuous Attributes

The algorithm AS for a probability density function reconstruction for continuous attributes distorted with the randomisation-based method was proposed in [7].

The algorithm solves the following problem:
Original values $x_1, x_2, ..., x_n$ of a one-dimensional distribution are the realisation of n independent random variables $X_1, X_2, ..., X_n$ with the same distribution as the variable X. To hide information, n independent random variables $Y_1, Y_2, ..., Y_n$ with the same distribution as the random variable Y have been used. Given $x_1 + y_1, x_2 + y_2, ..., x_n + y_n$ (y_i is the realisation of the random variable Y_i) and cumulative distribution function F_Y for the variable Y, a cumulative distribution function F_X for the random variable X is to be estimated.

The solution to the given problem is as follows:
Let w_i be the value of $X_i + Y_i$, thus $w_i = x_i + y_i$. The individual values x_i and y_i are not known, only their sums are revealed. Assuming that the probability density function f_X for variable X and f_Y for Y are known, Bayes rule [51] can be used to estimate the posterior (cumulative) distribution function F'_{X_1} for the variable X_1. The posterior distribution function F'_{X_1} can be written as follows:

$$F'_{X_1}(a) = \int_{-\infty}^{a} f_{X_1}(z|X_1 + Y_1 = w_1)dz, \tag{3.5}$$

where $F'_{X_1}(a)$ is the estimator of the posterior (cumulative) distribution function $F_{X_1}(a)$. Using Bayes rule, we obtain:

$$F'_{X_1}(a) = \int_{-\infty}^{a} \frac{f_{X_1+Y_1}(w_1|X_1 = z)f_{X_1}(z)}{f_{X_1+Y_1}(w_1)}dz. \tag{3.6}$$

Then expanding the denominator, we get:

$$F'_{X_1}(a) = \int_{-\infty}^{a} \frac{f_{X_1+Y_1}(w_1|X_1 = z)f_{X_1}(z)}{\int_{-\infty}^{\infty} f_{X_1+Y_1}(w_1|X_1 = z')f_{X_1}(z')dz'}dz. \tag{3.7}$$

Since the inner integral is independent of the outer, it can be treated as a constant and moved outside the outer integral:

$$F'_{X_1}(a) = \frac{\int_{-\infty}^{a} f_{X_1+Y_1}(w_1|X_1 = z)f_{X_1}(z)dz}{\int_{-\infty}^{\infty} f_{X_1+Y_1}(w_1|X_1 = z)f_{X_1}(z)dz}. \tag{3.8}$$

Since Y_1 is independent of X_1, thus:

$$F'_{X_1}(a) = \frac{\int_{-\infty}^{a} f_{Y_1}(w_1 - z)f_{X_1}(z)dz}{\int_{-\infty}^{\infty} f_{Y_1}(w_1 - z)f_{X_1}(z)dz}. \tag{3.9}$$

Since $f_{X_1} \equiv f_X$ and $f_{Y_1} \equiv f_Y$:

$$F'_{X_1}(a) = \frac{\int_{-\infty}^{a} f_Y(w_1 - z)f_X(z)dz}{\int_{-\infty}^{\infty} f_Y(w_1 - z)f_X(z)dz}. \tag{3.10}$$

To estimate the posterior distribution function F'_X given $x_1+y_1, x_2+y_2, ..., x_n+y_n$, the average for each X_i can be computed:

$$F'_X(a) = \frac{1}{n}\sum_{i=1}^{n} F'_{X_i} = \frac{1}{n}\sum_{i=1}^{n} \frac{\int_{-\infty}^{a} f_Y(w_i - z)f_X(z)dz}{\int_{-\infty}^{\infty} f_Y(w_i - z)f_X(z)dz}. \tag{3.11}$$

The posterior density function f'_X is obtained by differentiating F'_X:

$$f'_X(a) = \frac{1}{n}\sum_{i=1}^{n} \frac{f_Y(w_i - a)f_X(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z)f_X(z)dz}. \tag{3.12}$$

Having a large number of samples, f'_X should correspond to the original probability density function f_X.

To estimate f'_X, the knowledge of f_Y and f_X is needed. f_Y is known, because the distorting

distribution function is known for a miner. As the original probability density function f_X is unknown, a uniform distribution is assumed as an initial estimate of density function and then refined in an iterative way by applying Equation 3.12. See Algorithm 7 for details.

Algorithm 7 The AS algorithm

$f_X^0 :=$ uniform distribution
$j := 0$ // iteration number
repeat
$f_X^{j+1}(a) = \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(w_i - a) f_X^j(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z) f_X^j(z) dz}$
$j := j + 1$
until(stopping criterion met)

To reduce the calculation complexity, partitioning of the domain (of the data values) into intervals can be used. R. Agrawal and R. Srikant make two approximations in [7]. First, the distance between z and w_i (or between a and w_i) is approximated as the distance between mid-points of the corresponding intervals. Second, the density distribution function $f_X(a)$ is approximated with the average of the density distribution function over the interval in which a lies.

Applying these two approximations to the posterior density function, one obtains:

$$f'_X(a) = \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(m(w_i) - m(a)) f_X(I(a)) da}{\int_{-\infty}^{\infty} f_Y(m(w_i) - m(z)) f_X(I(z)) dz}, \tag{3.13}$$

where:

— $I(x)$ denotes the interval in which x lies,

— $m(I_p)$ denotes the mid-point of the interval I_p,

— $m(x)$ denotes the mid-point of the interval $I(x)$,

— $f_X(I_p)$ is the average value of a probability density function over the interval I_p, i.e., $f_X(I_p) = \frac{\int_{I_p} f_X(z) dz}{\int_{I_p} dz}$.

Let I_p for $p = 1, \ldots, k$ denotes p-th interval and L_p the width of the interval I_p. The integral in the denominator can be replaced with a sum, since $m(z)$ and $f_X(I_z)$ are constant within an interval.

$$f'_X(a) = \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(m(w_i) - m(a)) f_X(I(a))}{\sum_{t=1}^{k} f_Y(m(w_i) - m(I_t)) f_X(I_t) L_t} \tag{3.14}$$

The average value of a posterior density function over the interval I_p can be computed in the following way:

$$f'_X(I_p) = \frac{\int_{I_p} f'_X(z)dz}{L_p}. \tag{3.15}$$

Substituting Equation 3.14, one obtains:

$$f'_X(I_p) = \int_{I_p} \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(m(w_i) - m(z)) f_X(I(z))dz}{\sum_{t=1}^{k} f_Y(m(w_i) - m(I_t)) f_X(I_t) L_t} / L_p. \tag{3.16}$$

$I(z) = I_p$ over the interval I_p, hence:

$$f'_X(I_p) = \int_{I_p} \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(m(w_i) - m(I_p)) f_X(I_p)dz}{\sum_{t=1}^{k} f_Y(m(w_i) - m(I_t)) f_X(I_t) L_t} / L_p. \tag{3.17}$$

The numerator is constant over the interval I_p and $\int_{I_p} dz = L_p$, hence the equation can be rewritten as follows:

$$f'_X(I_p) = \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(m(w_i) - m(I_p)) f_X(I_p)}{\sum_{t=1}^{k} f_Y(m(w_i) - m(I_t)) f_X(I_t) L_t}. \tag{3.18}$$

Let $N(I_p)$ be the number of points that lie in the interval I_p, i.e., the number of elements in the set $\{w_i | w_i \in I_p\}$. Since points from the same interval have the same mid-point $m(w_i)$, the equation can be written as follows:

$$f'_X(I_p) = \frac{1}{n} \sum_{s=1}^{k} N(I_s) \frac{f_Y(m(I_s) - m(I_p)) f_X(I_p)}{\sum_{t=1}^{k} f_Y(m(I_s) - m(I_t)) f_X(I_t) L_t}. \tag{3.19}$$

Let $Pr'(X \in I_p)$ be the probability that X belongs to the interval I_p, i.e., $Pr'(X \in I_p) = f'_X(I_p)L_p$. Hence, multiplying both sides of Equation 3.19 by L_p and using $Pr(X \in I_p) = f_X(I_p)L_p$, we obtain:

$$Pr'(X \in I_p) = \frac{1}{n} \sum_{s=1}^{k} N(I_s) \frac{f_Y(m(I_s) - m(I_p)) Pr(X \in I_p)}{\sum_{t=1}^{k} f_Y(m(I_s) - m(I_t)) Pr(X \in I_t)}. \tag{3.20}$$

Equation 3.20 can be used to calculate the next approximation of the original probability density function in the Algorithm 7.

Equation 3.20 gives $O(k^3)$ computation complexity, because for each interval (there are k intervals) the value of the sum and the value of the denominator (for each element of the sum) are calculated.

It is possible to calculate the probability for each interval I_p, where $p = 1, \ldots, k$, with $O(k^2)$ calculation complexity.

The denominator is independent from I_p, hence it can be calculated once. However, the denominator depends on s, thus has to be calculated for each possible value of s separately.

To stop an iterate reconstruction, three possible stopping criteria were proposed in [7].

The first criterion is met when the reconstructed distribution is statistically the same as the original distribution. To check the similarity of distributions, for instance, χ^2 measure (details about χ^2 can be found in [109]) can be used. This criterion could be used only for testing, because the original distribution is not known in practice.

The second solution is to compare the randomised current estimate of the original distribution with the distorted distribution used for the reconstruction and stop when these two distributions are statistically the same. This criterion assumes that the current estimate which is close enough to the original distribution should be the same after the distortion as the distorted distribution used for the reconstruction. As stated in [7], the difference between two distorted distributions is not a reliable indicator.

The last approach is to compare two consecutive estimates of the original distribution. When the difference is small enough, the process is completed. 1% of the threshold of χ^2 test was used in [7].

As stated in [2], the AS algorithm may not always converge and even it converges, there is no guarantee that it gives a reasonable estimate of the original distribution. There was no proof given for that statement and this issue was not mentioned in [7].

Algorithm EM for Probability Distribution Reconstruction of Continuous Attributes

The algorithm for a probability density function reconstruction for continuous attributes distorted by means of the the randomisation-based method was proposed in [2], as well. The algorithm was called EM by the authors. The problem to be solved is the same as for the AS algorithm.

Let us assume that the domain of the attribute X is discretised into k intervals. The density function $f_X(x)$ is constant over i-th interval and is equal to θ_i. It restricts $f_X(x)$ to a class parameterised by the finite set of parameters $\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$. To emphasize the parametric dependence of the density function on Θ, the following notation is used $f_{X;\Theta}(x)$ and the density function can be written as follows:

$$f_{X;\Theta}(x) = \sum_{i=1}^{k} \theta_i I_{\Omega_i}(x), \tag{3.21}$$

where $I_{\Omega_i} = 1$, when x belongs to the interval $\Omega_i(x)$, and 0, otherwise. $f_{X;\Theta}(x)$ is a density, thus

$\sum_{i=1}^{k} \theta_i m(\Omega_i) = 1$, where $m(\Omega_i)$, according to previously used notation, denotes the mid-point of the interval Ω_i.

The above form can approximate any density function with an arbitrary precision. According to this parametrisation, an estimator of Θ should be calculated.

Let $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ be realisations of n independent and identically distributed random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, each with the density function $f_X(x)$. These realisations constitute the original data set. Assume that $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ are realisations of n independent and identically distributed random variables $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$, each with the density function $f_Y(y)$. These realisations constitute the perturbations to the original data set. Given the perturbed values $\mathbf{z} = \{z_1, z_2, \ldots, z_n\}$, where $z_i = x_i + y_i$, and the density function $f_Y(y)$, $f_X(x)$ should be estimated. The perturbed random variables will be denoted by $\mathbf{Z} = \{Z_1, Z_2, \ldots, Z_n\}$. Let $\hat{\Theta} = \{\hat{\theta_1}, \hat{\theta_2}, \hat{\theta_3}, ..., \hat{\theta_k}\}$ be the estimate of parameters produced by the EM algorithm.

Given a large enough set of observations $\mathbf{Z} = \mathbf{z}$, maximum-likelihood estimate (MLE) [48] of the parameter Θ denoted as $\hat{\Theta}_{ML}$ should be found.

$$\hat{\Theta}_{ML} = \arg\max_{\Theta} \ln f_{\mathbf{Z};\Theta}(\mathbf{z}) \tag{3.22}$$

A maximum-likelihood estimator has many desired properties, e.g., consistency, asymptotic unbiasedness, and asymptotic minimum variance among unbiased estimators [119]. It is not always possible to find a maximum likelihood estimator directly (using Equation 3.22), and $f_{\mathbf{Z};\Theta}(\mathbf{z})$ is this case.

To find $\hat{\Theta}_{ML}$, D. Agrawal i C. C. Aggarwal derived a reconstruction algorithm which fits into the broad framework of Expectation Maximisation algorithms. In the proposed algorithm a set of data $\mathbf{X} = \mathbf{x}$ is assumed to be observable and $\ln f_{\mathbf{X};\Theta}(\mathbf{x})$ is maximised over all values of Θ (M-step). However, $\mathbf{x}$ is unavailable, thus $\ln f_{\mathbf{X};\Theta}(\mathbf{x})$ is replaced by its conditional expected value given $\mathbf{Z} = \mathbf{z}$ and the current estimate of Θ (E-Step).

Let Q be a function defined as follows:

$$Q(\Theta, \hat{\Theta}) = E[\ln f_{\mathbf{X};\Theta}(\mathbf{X}) | \mathbf{Z} = \mathbf{z}, \hat{\Theta}]. \tag{3.23}$$

$Q(\Theta, \hat{\Theta})$ is the expected value of $\ln f_{\mathbf{X};\Theta}(\mathbf{X})$ computed with respect to $f_{\mathbf{X}|\mathbf{Z}=\mathbf{z};\hat{\Theta}}$ (the density of $\mathbf{X}$ given $\mathbf{Z} = \mathbf{z}$ and the parameter vector $\hat{\Theta}$).

At the beginning Θ is initialised to a nominal value, then EM algorithms iterates over E and M-steps:

— E-step: compute $Q(\Theta, \Theta^j)$,

— M-step: update $Q^{j+1} = \arg\max_\Theta Q(\Theta, \Theta^j)$.

Further derivation of the E-steps and M-steps is problem specific and is presented below.

The following theorems characterise the E-step and M-step.

Theorem 3.7.1 *The value of* $Q(\Theta, \hat{\Theta})$ *during the E-step is given by:*

$$Q(\Theta, \hat{\Theta}) = \sum_{i=1}^{k} \psi_i(\mathbf{z}; \hat{\Theta}) \ln \theta_i,$$

where

$\psi_i(\mathbf{z}; \hat{\Theta}) = \hat{\theta}_i \sum_{j=1}^{N} \frac{Pr(Y \in z_j - \Omega_i)}{f_{\mathbf{Z};\hat{\Theta}}(z_j)}$ *and* $v \in z_j - \Omega_i$*, if* $z_j - v \in \Omega_i$.

In the M-step the value of Θ which maximises $Q(\Theta, \hat{\Theta})$ is calculated.

Theorem 3.7.2 *The value of* Θ *which maximises* $Q(\Theta, \hat{\Theta})$ *in the M-step is given by:*

$$\theta_i = \frac{\psi_i(\mathbf{z}; \hat{\Theta})}{m_i N},$$

where

$\psi_i(\mathbf{z}; \hat{\Theta}) = \hat{\theta}_i \sum_{j=1}^{N} \frac{Pr(Y \in z_j - \Omega_i)}{f_{\mathbf{Z};\hat{\Theta}}(z_j)}$.

The proofs can be found in [2].

Given that $Z = X + Y$, and X and Y are independent, the probability density function for Z can be computed in the following way:

$$f_{Z;\hat{\Theta}}(z) = \int f_X(v) f_Y(z - v) dv$$

$$= \sum_{i=1}^{k} \int_{\Omega_i} \hat{\theta}_i f_Y(z - v) dv = \sum_{i=1}^{k} \hat{\theta}_i Pr(Y_i \in z - \Omega_i).$$

Having all that properties, the EM algorithm can be described in details (see Algorithm 8). The stopping criterion for the EM algorithm is the same as for the AS algorithm.

Algorithm 8 The EM reconstruction algorithm

initialise parameters $\theta_i^0 = \frac{1}{k}, i = 1, \ldots, k, j = 0$
repeat
update Θ according to the following equation $\theta_i^{j+1} = \frac{\psi_i(\mathbf{z}; \Theta^k)}{m_i N}$
$j = j + 1$
until(stopping criterion met)

The proof that the EM algorithm converges can be found in [2]. The authors of the EM algorithm stated that it is theoretically the best algorithm and having a large set of distorted samples, the EM algorithm can reconstruct the original distribution with little or without information loss [2].

As in the AS algorithm, it is possible to reduce computation complexity for the EM algorithm too. This time the reduction increases memory complexity.

For the AS algorithm, the values of the denominator should be stored for each interval, separately, for the EM algorithms for each sample. The number of samples is usually much more higher than the number of intervals. The lack of the optimisation (complexity $O(k^2N)$ instead of $O(kN)$) leads to the significant increase of computation time.

The AS and EM algorithms are based on the same principle. However, the AS algorithm assumes additional simplifications. In order to compare both algorithms, the iterative computations will be studied. To compute the value of the probability density function for a continuous attribute, Equation 3.19 can be used for the AS algorithm, for the EM algorithm the following equation:

$$\theta_p^{j+1} = \frac{\theta_p^j \sum_{s=1}^{N} \frac{Pr(Y \in z_s - \Omega_p)}{\sum_{t=1}^{k} \theta_t^j Pr(Y_t \in z_s - \Omega_t)}}{m_p N}. \tag{3.24}$$

Both algorithms assume discretisation of a continuous attribute and a constant probability density function over an interval.

In contrast with the EM algorithm, the AS algorithm assumes that the distance between two points is the distance between mid-points of the intervals in which the points lie. This assumption is used in the nominator ($m(I_s) - m(I_p)$) and the denominator ($m(I_s) - m(I_t)$) of Equation 3.19 and reduces computation complexity for the AS algorithm to $O(k^3)$ (comparing to $O(k^2N)$ for the EM algorithm)[6], because the distance for all the points which lie in the same interval is the same (the mid-point is used to compute the distance). For the AS algorithm, the nominator is calculated once per each interval and multiplied by the number of points which lie in this interval ($N(I_s)$).

Assigning Reconstructed Values to Samples for Continuous Attributes

The algorithm for assigning reconstructed values to samples for continuous attributes was presented in [7]. We describe this algorithm in this section.

[6] The number intervals (k) is usually significantly smaller than the number of samples (N).

Given the number of samples in intervals, reconstructed values can be assigned to samples, which allows a miner to choose the best test for a tree node using, e.g., *gini index*[7].

Let $I_1, ..., I_m$ denote m intervals and $N(I_k)$ be the number of samples in I_k interval. Samples should be sorted in an ascending order and assigned to consecutive intervals as follows: $N(I_1)$ first samples are assigned to the first interval I_1, the next $N(I_2)$ samples to the second interval I_2, etc.

When a split point in a test lies between the interval I_s and I_{s+1}, the samples assigned to intervals $I_1, ..., I_s$ meet the test and samples assigned to intervals $I_{s+1}, ..., I_m$ do not meet the test.

Please, observe that assigned intervals should not be treated as estimates of original values.

EM/AS Algorithm for Probability Distribution Reconstruction of Nominal Attributes

In [14], we proposed the EM/AS algorithm for reconstructing a probability distribution of a nominal attribute.

The EM/AS algorithm is based on two algorithms: AS proposed in [7] and its extension EM presented in [2]. Both algorithms reconstruct a probability distribution of continuous attributes.

To reconstruct probability distribution of a nominal attribute, both EM and AS algorithms were modified to obtain the EM/AS (Algorithm 9). The modifications of both algorithms (AS and EM) give the same result.

The algorithm solves the following problem: a nominal attribute X has the possible values $v_1, v_2, v_3, ..., v_k$ and n samples. Value for each sample is modified according to a probability $Pr(v_p \rightarrow v_r)$ (a probability that a value v_p will be changed to a value v_r). $X(s)$ means a value of an attribute X for a sample s. An original probability distribution of an attribute X should be reconstructed.

Algorithm 9 The EM/AS nominal attribute probability distribution reconstruction algorithm

$Pr(X = v_p)^0 := \frac{1}{k}, p = 1, ..., k$
$j := 0$ //iteration number
repeat
$Pr(X = v_p)^{j+1} = \frac{1}{n} \sum_{s=1}^{n} \frac{Pr(v_p \rightarrow X(s)) Pr^j(X=v_p)}{\sum_{t=1}^{k} Pr(v_t \rightarrow X(s)) Pr^j(X=v_t)}$
$j := j + 1$
until(stopping criterion met)

The algorithm starts with the uniform distribution and calculates the estimate of the probability distribution in every iteration.

[7] For details about *gini index* see Section 2.2.

Stopping criterion is the same as for the AS and EM algorithms (the algorithm is stopped when the difference between successive estimates of the original probability distribution becomes small, as little as 1% of the threshold of the χ^2 test).

EQ Algorithm for Probability Distribution Reconstruction of Nominal Attributes

In [15] we proposed the EQ algorithm, the name of the algorithm comes from the phrase *system of EQuations*, that reconstructs the probability distribution of nominal attributes and can be used instead of the EM/AS algorithm. The EQ algorithm outperforms the EM/AS, especially for high levels of privacy [15].

The problem to be solved is the same as for the EM/AS algorithm: there are a nominal attribute X with the possible values $v_1, v_2, v_3, ..., v_k$ and n samples. A value for each sample is modified according to a probability $Pr(v_p \rightarrow v_r)$ (a probability that a value v_p will be changed to a value v_r) and we want to reconstruct an original probability distribution of an attribute X.

Let us assume that there is an attribute *Colour* with 3 values: $v_1 = green$, $v_2 = blue$, and $v_3 = black$.

For the original value of the attribute, e.g., $green$, the probability $Pr(v_1 \rightarrow v_1)$ that the value will be the same after the modification is known, as well as the probability of changing the value from $green$ to $blue$ and from $green$ to $black$. Moreover, when the value of the attribute after the distortion is, e.g., $green$, the original value was one of the three possible values: $green$, $blue$, and $black$ and all the probabilities $Pr(v_1 \rightarrow v_1)$, $Pr(v_2 \rightarrow v_1)$, $Pr(v_3 \rightarrow v_1)$ how the value has become $green$ are known.

Let Z be the attribute after the modification with the possible values $v_1, v_2, v_3, ..., v_k$. In the example, the attribute Z has 3 values: $green$, $blue$, and $black$ and the following equation can be written:

$$P(Z = green) = a_{1,1}P(X = green) + a_{1,2}P(X = blue) + a_{1,3}P(X = black),$$

where $a_{s,p} = Pr(v_p \rightarrow v_s)$. For colours $blue$ and $black$ the similar equations can be written:

$$P(Z = blue) = a_{2,1}P(X = green) + a_{2,2}P(X = blue) + a_{2,3}P(X = black)$$
$$P(Z = black) = a_{3,1}P(X = green) + a_{3,2}P(X = blue) + a_{3,3}P(X = black).$$

Now there are 3 equations and 3 unknown variables ($P(X = green)$, $P(X = blue)$, $P(X = black)$), thus the system of linear equations can be solved.

In general there is the following system of k equations:

$$
\begin{aligned}
P(Z = v_1) &= a_{1,1}P(X = v_1) + a_{1,2}P(X = v_2) + \cdots + a_{1,k}P(X = v_k) \\
P(Z = v_2) &= a_{2,1}P(X = v_1) + a_{2,2}P(X = v_2) + \cdots + a_{2,k}P(X = v_k) \\
&\vdots \\
P(Z = v_k) &= a_{k,1}P(X = v_1) + a_{k,2}P(X = v_2) + \cdots + a_{k,k}P(X = v_k)
\end{aligned}
$$

with k unknown variables.

Let $\mathbf{X}$ be the column vector with elements $x_1, ..., x_k$, where $x_i = P(X = v_i)$ and $\mathbf{Z}$ be the column vector with elements $z_1, ..., z_k$, where $z_i = P(Z = v_i)$. Let $\mathbf{P}$ be the matrix of retaining/changing values of a nominal attribute. We can rewrite the system of equations in the matrix form as:

$$\mathbf{Z} = \mathbf{PX} \tag{3.25}$$

To find values of $P(X = v_i),\ i = 1, \ldots, k$, we need to solve Equation 3.25. We can solve it by left multiplying both sides by inverted $\mathbf{P}$, i.e., $\mathbf{P}^{-1}$ (only if inverted $\mathbf{P}$ exists).

Nonexistence of the inverted matrix is not troublesome because the number of values of a nominal attribute is known before collecting data starts and a non-singular matrix $\mathbf{P}$ can be chosen, which guarantee the existence of inverted $\mathbf{P}$ matrix.

ARVeSNA Algorithm for Assigning Reconstructed Values to Samples for Nominal Attributes

We proposed the algorithm for assigning reconstructed values to samples for nominal attributes in [12, 14] and describe this algorithm in this section.

Having reconstructed a probability distribution of a nominal attribute, reconstructed values can be assigned to samples in order to find the best test for a tree node using, e.g., *gini index*. Please, observe that assigned values should not be treated as estimates of original values.

The algorithm solves the following problem:
Since modified values of a nominal attribute are given, the probability distribution of a modified attribute (i.e., $P(Z = v_i),\ i = 1, \ldots, k$) and the number of all samples n are known. The reconstructed probability distribution ($P(X = v_i),\ i = 1, \ldots, k$) is estimated. The aim is to assign reconstructed values to samples taking into account the reconstructed probability distribution.

In order to solve this problem, the number of distorted samples ($n_Z(v_i)$) is counted separately for each value of an attribute and the number of original samples ($n_X(v_i) = P(X = v_i)n$) is estimated.

Then the difference, called $\delta(v_i)$, between $n_Z(v_i)$ and $n_X(v_i)$ is calculated. $\delta(v_i) > 0$ means

that there are too many samples because there are more samples with distorted value of v_i than the reconstructed number of samples for the value v_i suggests. A sample corresponding to a positive value of $\delta(v_i)$ is found and assigned with a reconstructed value v_j for which a value of $\delta(v_j)$ is negative and the reconstructed value v_j has the highest probability to be distorted to the value v_i. Values of corresponding $\delta(v_i)$ and $\delta(v_j)$ are updated and the process is continued until all values of $\delta(v_i),\ i = 1, \ldots, k$ are zero.

Having completed the process, samples with the reconstructed values are assigned according to an original (reconstructed) probability distribution.

In the case of the reconstruction of a probability distribution for each class, this process is performed for every single class separately. Moreover, having a reconstructed probability distributions divided into classes, the best test can be chosen without assigning the values.

When a test for a given node has been chosen during a decision tree building and the probability of retaining the original value is greater than 0.5, samples with v_j value which meet the test and have negative $\delta(v_j)$ are found first if a chosen sample with v_i value and positive $\delta(v_i)$ meets the test and samples with v_j value which do not meet the test and have negative $\delta(v_j)$ are found first if a chosen sample with v_i value and positive $\delta(v_i)$ does not meet the test. When the probability of retaining the original value is less than 0.5, samples with v_j value which do not meet the test and have negative $\delta(v_j)$ are found first if a chosen sample with v_i value and positive $\delta(v_i)$ meets the test and samples with v_j value which meet the test and have negative $\delta(v_j)$ are found first if a chosen sample with v_i value and positive $\delta(v_i)$ does not meet the test.

The described algorithms for nominal attributes presented in this section can be combined with those for continuous[8] attributes and allow a miner to mine databases containing both nominal and continuous attributes simultaneously.

[8] Continuous attributes are modified with the additive perturbation technique.

4. Optimisation for MASK Scheme in Privacy Preserving Association Rules Mining

In order to eliminate exponential complexity in estimating a support of an itemset with respect to its cardinality, we propose the optimisation for MASK scheme [13]. Instead of $O(2^n)$ complexity, where n is the number of all possible items in a database, we have $O(2^{rThreshold})$, where $rThreshold$, $rThreshold < n$, is a constant. We will call MASK scheme with our proposed optimisation as MMASK (Modified MASK).

4.1. Reducing Number of Items in Estimating n-itemsets Support

The reduction of a number of items in estimating the original support of an n-itemset X can be obtained by choosing for an itemset X a subset of distorted transactions for estimation of the true support.

Let $reductionTreshold$ ($rThreshold$) denote the maximal length of an itemset used in estimating the support of an n-itemset X, $rThreshold < n$.

The true support of X can be estimated as the support of a reduced itemset $R \subset X$ in transactions which support $X \setminus R$ in a true database, $|R| < rThreshold$.

Example Let $rThreshold = 3$. We would like to estimate the support of the itemset $X = \{a, b, c, d\}$ using no more than $rThreshold$ items. Thus, the support of the itemset X is estimated as the support of, e.g., the reduced itemset $R = \{d\}$ in transactions which support the itemset $X \setminus R = \{a, b, c\}$.
□

As there is no access to a true database $\mathcal{T}$, the subset of the chosen distorted transactions $\mathcal{D}_{\mathcal{R}}$ from the distorted database $\mathcal{D}$, $\mathcal{D}_{\mathcal{R}} \subset \mathcal{D}$, should support $X \setminus R$ in the true database $\mathcal{T}$ with a high probability. The CTS algorithm for choosing distorted transactions which support a given itemset $X \setminus R$ in the true database $\mathcal{T}$ with a high probability is proposed in Section 4.3. A probability that a distorted transaction supports a given itemset in the true database is estimated based on the distorted set of transactions.

4.2. Process of Mining Frequent Itemsets with MMASK

The Privacy Preserved Apriori-MMASK (PPApriori-MMASK) algorithm for mining frequent itemsets which uses MMASK scheme (Algorithms 10, 11, 12) estimates original supports of candidate m-itemsets like the Apriori algorithm modified to use MASK, PPApriori-MASK (Algorithm 5), until m is less than or equal to $rThreshold$. When m is greater than $rThreshold$, the support of the m-itemset X is determined by estimating the support of a reduced itemset R in the set $\mathcal{D}_R$ of distorted transactions which support $X \setminus R$, $|X \setminus R| = rThreshold$, in the true database with a high probability.

The true supports for itemsets with higher length are estimated based on the transaction set $\mathcal{D}_R \subset \mathcal{D}$ until the length of a reduced itemset exceeds $rThreshold$. Then a subset of transactions $\mathcal{D}_{R'}$ is chosen (by means of the CTS algorithm, for details refer to Section 4.3) from the subset $\mathcal{D}_R$. The chosen $\mathcal{D}_{R'}$ subset of transactions support the subset $X' \setminus R'$ of the itemset candidate X', $|X' \setminus R'| = rThreshold$, $X \subset X'$, with a high probability. The true support of the itemset X' is estimated as the support of an itemset R' in the subset of transactions $\mathcal{D}_{R'}$.

The true supports for longer candidate itemsets are estimated based on the transaction set $\mathcal{D}_{R'}$ until the length of a reduced candidate itemset exceeds $rThreshold$. Then the subset of transactions $\mathcal{D}_{R''}$ is chosen (by means of the CTS algorithm, for details refer to Section 4.3) and the process is continued.

We will use the following notation for MMASK (Algorithms 10, 11, 12):

— $\mathcal{X}_m$ denotes candidate m-itemsets, which are potentially frequent.

— $\mathcal{F}_m$ are frequent m-itemsets based on estimations of original supports of itemsets.

— $X[i]$ is the i-th item in the itemset X.

— $X[1]{\cdot}X[2]{\cdot}X[3]{\cdot}\ldots{\cdot}X[m]$ denotes m-itemset, which consists of $X[1], X[2], X[3], \ldots, X[m]$.

— $\mathcal{T}$ is the original data set.

— $\mathcal{D}$ is the data set distorted according to the MASK scheme and each item i is distorted according to the matrix $\mathbf{M}_i$.

— $X.R$ is the reduced itemset of X.

— $X.R.\mathbf{C}^D$ is the support vector field of the reduced itemset $X.R$ in the distorted data set $\mathcal{D}$.

— $X.R.\mathbf{C}^T$ is the support vector field of the reduced itemset $X.R$ in the true data set.

— $X.R.\mathbf{C}^T_j$ is the j-th element of the vector $X.R.\mathbf{C}^T$.

— $X.R.\mathbf{C}^D_j$ is the j-th element of the vector $X.R.\mathbf{C}^D$.

— $X.R.\mathbf{M}$ is a $\mathbf{M}$ matrix (see Equation 3.3) for the reduced itemset $X.R$.

— $X.R.\mathbf{M}^{-1}$ is an inverted $\mathbf{M}$ matrix for the reduced itemset $X.R$.

— $X.R_F$ is the lexicographically first ancestor of the reduced itemset $X.R$.

— $X.\mathcal{D}_R$ is the subset of transactions from the database $\mathcal{D}$ which support $X.R$ with a high probability.

Algorithm 10 The PPApriori-MMASK algorith, Apriori algorithm modified to use MMASK

input: $minimumSupport$
input: $\mathcal{D}$ // binary distorted data set
input: $rThreshold$, $rThreshold > 0$
$\mathcal{F}_1$ ={1-itemsets which are frequent based on estimations of original support of singletons}
for all $X \in \mathcal{F}_1$ **do begin**
$X.\mathcal{D}_R = \mathcal{D}$
$X.R = X$
end
$index = 1$
for $(m = 2; \mathcal{F}_{m-1} \neq \emptyset; m$++$)$ **do begin**
$index$++
$\mathcal{X}_m$ = $aprioriGen(\mathcal{F}_{m-1})$ //generate new candidates
if index $> rThreshold$ **then begin**
for all $X \in \mathcal{X}_m$ **do begin**
$X.\mathcal{D}_R = \{O \in X.\mathcal{D}_R | O$ supports $X.R_F$ with a high probability in the true data set$\}$
end
end
$supportCount(\mathcal{X}_m)$
$\mathcal{F}_m = \{X \in \mathcal{X}_m | X.R.\mathbf{C}^T_{2^m-1} \geq minimumSupport\ \}$
if $index > rThreshold$ **then** $index = 1$
end
return $\bigcup_m \mathcal{F}_m$

The PPApriori-MMASK algorithm generates candidates for frequent sets with a given length in the Apriori-like fashion. In every iteration of candidates generation when the reduced itemset used to estimate an original support of a candidate in the true database based on a support counted in distorted transactions exceeds $rThreshold$, the reduction of the set of transactions which is used to estimate an original support of the candidate is performed. Then, having estimated the original supports of candidates, the minimum support condition is checked and candidates which are not frequent are removed. As the result of the PPApriori-MMASK algorithm, the itemsets with the estimated support greater than or equal to $minimumSupport$ are provided.

Example Let $rThreshold = 3$. We would like to estimate the support of the candidate $X = \{a, b, c, d\}$ based on distorted database $\mathcal{D}$. This candidate was generated in the Apriori-like fashion from the frequent sets $\{a, b, c\}$ and $\{a, b, d\}$. We can use either $\{a, b, c\}$ or $\{a, b, d\}$ as a condition to reduce the transaction set. Let us assume that we choose the lexicographically

first set, $\{a, b, c\}$[1], thus, $R = \{d\}$, $X \setminus R = \{a, b, c\}$. Then the subset $\mathcal{D}_R$ of the distorted transactions which support $X \setminus R$ with a high probability is chosen (we use the CTS algorithm for choosing transactions subset described in the next section to achieve this goal). Having chosen the subset $\mathcal{D}_R$ of the transactions, we estimate the true support of the candidate $X = \{a, b, c, d\}$ as the estimated true support for the singleton $R = \{d\}$ based on the subset $\mathcal{D}_R$ of distorted transactions because all distorted transactions in $\mathcal{D}_R$ used to estimate the support of $R = \{d\}$ support the frequent set $X \setminus R = \{a, b, c\}$ in the true database with a high probability. We compute the support for supersets of $X \setminus R = \{a, b, c\}$ in the same manner until candidates with the length greater than $|\{a, b, c\}| + rThreshold$, for instance $X' = \{a, b, c, d, e, f, g\}$, appear.

Given the subset $\mathcal{D}_R$, we can choose from this subset the transactions which support $\{d, e, f\}$ set with a high probability and obtain $\mathcal{D}_{R'}$ subset, which contains distorted transactions. Those transactions support $X' \setminus R' = \{a, b, c, d, e, f\}$ itemset with a high probability in the true database, because the distorted transactions in $\mathcal{D}_R$ already support $X \setminus R = \{a, b, c\}$ in the true database with a high probability. Then, we estimate the support of $X' = \{a, b, c, d, e, f, g\}$ set like we compute the singleton $R' = \{g\}$ support based on distorted transactions in $\mathcal{D}_{R'}$.
□

The reduced subset of distorted transactions is chosen (for each candidate) only in those passes in which candidates have length $k \cdot rTreshold + 1, k = 1, 2...$ In other passes (for a given candidate) the reduced subset of distorted transactions from the latest pass is used. Thus, the number of reduced sets of distorted transactions is less than or equal to the number of candidates in the current pass.

Instead of MASK $O(2^n)$ complexity of estimating an original support of itemsets with respect to n, the number of all possible items in a database, MMASK has $O(2^{rThreshold})$ complexity, where $rThreshold$, $rThreshold < n$, is a constant.

We will illustrate the reduction of complexity of estimating an original support of itemsets with respect to their length on the following example.

Example Let $rThreshold = 3$ and we would like to estimate the support of the candidate itemsets $X = \{a, b, c, d\}$ and $X' = \{a, b, c, d, e, f, g\}$ based on distorted database $\mathcal{D}$, like in the previous example.

[1] In future work, we plan to investigate which subset should be chosen to achieve the best accuracy. See Section 4.7.

Algorithm 11 The candidate generation algorithm for MMASK

function $aprioriGen$(**var** $\mathcal{F}_m$)
 for all $Y, Z \in \mathcal{F}_m$ **do begin**
  if $Y[1] = Z[1] \wedge \ldots \wedge Y[k-1] = Z[k-1] \wedge Y[k] < Z[k]$ **then begin**
   $X = Y[1] \cdot Y[2] \cdot Y[3] \cdot \ldots \cdot Y[k-1] \cdot Y[k] \cdot Z[k]$
   $X.\mathcal{D}_R = Y.\mathcal{D}_R$
   if index $> rThreshold$ **then begin**
    $X.R_F = Y.R$
    $X.R = Z[k]$
   end else begin
    $X.R = Y.R \cup Z[k]$
   end
   add X to $\mathcal{X}_{m+1}$
  end
 end
 for all $X \in \mathcal{X}_{m+1}$ **do begin**
  for all m-itemsets $Z \subset X$ **do begin**
  if $Z \notin \mathcal{F}_m$ **then** delete X from $\mathcal{X}_{m+1}$
  end
 end
 return $\mathcal{X}_{m+1}$
end

For better visualisation of the reduction of complexity of estimating an original support of itemsets with respect to their length, we consider also the estimation of the support of some subsets of the candidate itemsets $X = \{a, b, c, d\}$ and $X' = \{a, b, c, d, e, f, g\}$, for instance, $\{a\}$, $\{a, b, c\}$, etc.

For the itemset $\{a\}$, the vector C^D and the estimated vector C^T have $2^1 = 2$ elements and these vectors are the same for MASK and MMASK:

$$\mathbf{C^D} = \begin{bmatrix} C_1^D \\ C_0^D \end{bmatrix}, \quad \mathbf{C^T} = \begin{bmatrix} C_1^T \\ C_0^T \end{bmatrix}. \tag{4.1}$$

For the itemset $\{a, b, c\}$, the vector C^D and the estimated vector C^T have $2^3 = 8$ elements and these vectors are also the same for MASK and MMASK, because $|\{a, b, c\}| = rThreshold = 3$:

Algorithm 12 The support count algorithm for MMASK

procedure $supportCount$(**var** $\mathcal{X}_m$)
 for all transactions $T \in \mathcal{D}$ **do begin**
  for all candidates $X \in \mathcal{X}_m$ **do begin**
   if $T \in X.\mathcal{D}_R$ **then** $X.\mathbf{C}_j^D$++ //j is the number which has a binary form (in m digits) // of $X.R$ in the transaction T
  end
 end
 for all candidates $X \in \mathcal{X}_m$ **do begin**
  $X.R.\mathbf{C}^T = X.R.\mathbf{M}^{-1}X.R.\mathbf{C}^D$
  end
end

$$\mathbf{C^D} = \begin{bmatrix} C^D_{2^3-1=7} \\ . \\ . \\ . \\ C^D_1 \\ C^D_0 \end{bmatrix}, \quad \mathbf{C^T} = \begin{bmatrix} C^T_{2^3-1=7} \\ . \\ . \\ . \\ C^T_1 \\ C^T_0 \end{bmatrix}. \tag{4.2}$$

For the itemset $X = \{a, b, c, d\}$, the vector C^D and the estimated vector C^T for MASK have $2^4 = 16$ elements and the vector C^D and the estimated vector C^T for MMASK have $2^1 = 2$ elements, because $|X = \{a, b, c, d\}| > rThreshold = 3$ and the support of the itemset $X = \{a, b, c, d\}$ is calculated as the estimation of the support of the reduced itemset $R = \{d\}$ based on the subset $\mathcal{D}_R$ of the distorted transactions which support $X \setminus R = \{a, b, c, d\} \setminus \{d\} = \{a, b, c\}$ with a high probability in the true database:

$$\mathbf{C^D_{MASK}} = \begin{bmatrix} C^D_{2^4-1=15} \\ . \\ . \\ . \\ C^D_1 \\ C^D_0 \end{bmatrix}, \quad \mathbf{C^T_{MASK}} = \begin{bmatrix} C^T_{2^4-1=15} \\ . \\ . \\ . \\ C^T_1 \\ C^T_0 \end{bmatrix},$$

$$\mathbf{C^D_{MMASK}} = \begin{bmatrix} C^D_1 \\ C^D_0 \end{bmatrix}, \quad \mathbf{C^T_{MMASK}} = \begin{bmatrix} C^T_1 \\ C^T_0 \end{bmatrix}. \tag{4.3}$$

For the itemset $\{a, b, c, d, e, f\}$, the vector C^D and the estimated vector C^T for MASK

have $2^6 = 64$ elements and the vector C^D and the estimated vector C^T for MMASK have $2^3 = 8$ elements, because $|\{a, b, c, d, e, f\}| > rThreshold = 3$ and the support of the itemset $\{a, b, c, d, e, f\}$ is calculated as the estimation of the support of reduced itemset $\{d, e, f\}$ based on the subset $\mathcal{D}_R$ of the distorted transactions which support $\{a, b, c, d, e, f\} \setminus \{d, e, f\} = \{a, b, c\} = X \setminus R$ with a high probability in the true database.

For the itemset $X' = \{a, b, c, d, e, f, g\}$, the vector C^D and the estimated vector C^T for MASK have $2^7 = 128$ elements. For MMASK, the reduced itemset $|\{d, e, f, g\}|$ would have the length greater than $rThreshold = 3$, hence, the reduction is performed for the second time and the support of the itemset $X' = \{a, b, c, d, e, f, g\}$ is calculated as the estimation of the support of reduced itemset $R' = \{d\}$ based on the subset $\mathcal{D}_{R'} \subset \mathcal{D}_R$ of the distorted transactions which support $X' \setminus R' = \{a, b, c, d, e, f, g\} \setminus \{g\} = \{a, b, c, d, e, f\}$ with a high probability in the true database. The distorted transactions in subset $\mathcal{D}_{R'}$ are chosen from the subset $\mathcal{D}_R$ on the condition that those chosen transactions support $\{d, e, f\}$ with a high probability in the true database. Please, observe that the distorted transactions from the subset $\mathcal{D}_R$ support $X \setminus R = \{a, b, c\}$ with a high probability in the true database, hence, by choosing the distorted transactions from $\mathcal{D}_R$ that support $\{d, e, f\}$ with a high probability in the true database, we find the distorted transactions that support $X' \setminus R' = \{a, b, c, d, e, f, g\} \setminus \{g\} = \{a, b, c, d, e, f\}$ with a high probability in the true database.

In each pass, the vector C^D and the estimated vector C^T for MMASK have at most $2^{rThreshold}$ elements, where $rThreshold$ is a constant, contrary to MASK, where the number of elements is equal to 2^k, where k is the length of an itemset.

□

4.3. CTS Algorithm for Choosing Transactions Subset

Having a vector C^D and an estimated vector C^T (for detailed information about the structure of vectors C^D and C^T refer to Section 3.6.1), we know an estimated support $C^T_{2^n-1}$ of a candidate itemset C (for instance, the candidate $\{a, b, c\}$). Thus, $C^T_{2^n-1}$ transactions should be kept in the subset $\mathcal{D}_R$[2]. Randomisation factors in the range $\langle 0.7, 0.9 \rangle$, which are usually used, let us infer that it is very probable that transactions which support a candidate itemset C in a distorted database also support this set in a true database. Thus, we want to keep these transactions in our subset. There are two possible cases:

[2] We focus only on the $C^T_{2^n-1}$ and $C^D_{2^n-1}$ values because we are interested in the support of the set $\{a, b, c\}$, but not its subsets.

— $C^T_{2^n-1} \leq C^D_{2^n-1}$ – in this case we choose the first $C^T_{2^n-1}$ transactions[3] from a distorted database which support a candidate itemset set C (for instance, $\{a, b, c\}$)[4].

— $C^T_{2^n-1} > C^D_{2^n-1}$ – all $C^D_{2^n-1}$ distorted transactions which support a candidate itemset set C (for instance, $\{a, b, c\}$) in a distorted database are kept in the subset C_A. Then we choose i $(0 \leq i \leq 2^n - 2)$ for which there is the highest probability[5] that a true tuple which supports a candidate itemset set C (for instance, $\{a, b, c\}$) was distorted to value i and C^D_i is greater than zero.

Example Let $C^T_{2^n-1} \leq C^D_{2^n-1}$, $rThreshold$ = 2, and the candidate is $\{a, b, c\}$.

$$\mathbf{C}^D = \begin{bmatrix} 172 \\ . \\ . \\ . \end{bmatrix}, \quad \mathbf{C}^T = \begin{bmatrix} 168 \\ . \\ . \\ . \end{bmatrix}$$

The first 168 distorted transactions from 172 transactions which support $\{a, b\}$ in the distorted database $\mathcal{D}$ are chosen.

□

Example Let $C^T_{2^n-1} > C^D_{2^n-1}$, $rThreshold$ = 2, the candidate is $\{a, b, c\}$.

$$\mathbf{C}^D = \begin{bmatrix} 172 \\ 20 \\ 9 \\ . \end{bmatrix}, \quad \mathbf{C}^T = \begin{bmatrix} 191 \\ 23 \\ 11 \\ . \end{bmatrix}$$

All 172 distorted transactions which support $\{a, b\}$ in the distorted database are chosen. The additional 191 - 172 = 19 transactions are needed. Then the highest probable value of i (possible values are: 0, 1 or 2) is found. The probabilities for different values of i are computed based on $\mathbf{M}$. Let us assume that $i = 1$. Then we choose 9 distorted transactions which support $\{b\}$ in the distorted database. There are still 10 transaction left to choose. Let assume that the second highest probability that a distorted transaction comes from an original transaction which support $\{a, b\}$ is for $i = 2$. There are 20 distorted transactions for $i = 2$, that is, the distorted transactions which support $\{a\}$ in the distorted database. The first 10 of them are chosen. Thus, 191 distorted transactions are chosen.

□

[3] Considering only items from the set $\{a, b, c\}$, all $C^D_{2^n-1}$ transactions have the same probability to support $\{a, b, c\}$ in the true database.

[4] We plan to determine the best way to choose $C^T_{2^n-1}$ transactions (see Section 4.7).

[5] We use $\mathbf{M}$ matrix to compute this probability (for details about $\mathbf{M}$ matrix see Section 3.6.1).

4.4. Error Measures

We use two kinds of mining errors presented in [98], Support Error and Identity Error, in our experiments:

— Support Error (ρ):

This measure reflects the average relative error in the reconstructed support values for those itemsets that are correctly identified to be frequent. Denoting the reconstructed support by *recSupport* and the actual support by *actSupport*, the support error is computed over all frequent itemsets F as follows:

$$\rho = \frac{1}{|F|}\Sigma_{x \in F}\frac{|recSupport_x - actSupport_x|}{actSupport_x} * 100\ [\%].$$

We compute this measure separately for each length of itemsets, that is, for 1-itemsets, 2-item- sets, etc.

— Identity Error (σ):

This measure reflects the percentage error in identifying frequent itemsets and has two components: σ^+ indicating the percentage of false positives, and σ^- indicating the percentage of false negatives. Denoting the reconstructed set of frequent itemsets with R and the correct set of frequent itemsets with F, these measures are computed as follows:

$$\sigma^+ = \frac{|R \setminus F|}{|F|} * 100\ [\%], \quad \sigma^- = \frac{|F \setminus R|}{|F|} * 100\ [\%].$$

An additional measure to those used in [98] is calculated in the experiments.

— Accuracy of Identity (f):

This measure reflects the accuracy of identifying frequent itemsets and shows how many sets are correctly identified to be frequent.

$$f = |F \cap R|$$

This measure is equal to the number of all frequent sets mined in a true database minus false negatives.

$$f = |F| - \sigma^- * \frac{|F|}{100}$$

4.5. Relaxation

As the false negative error causes the mining process to miss some frequent itemsets, it is possible to use an effect of marginally relaxing $minimumSupport$. The relaxation proposed in [98] can decrease the false negative error component, which decreases the number of true frequent itemsets that are missed. As a result of the relaxation the false positive error components goes up, inevitably attracts not frequent sets.

The relaxation can be also applied to the MMASK scheme.

Algorithm 13 The PPApriori-rMMASK algorithm, the apriori algorithm modified to use the rMMASK scheme (i.e., MMASK and relaxation)

---

input: $minimumSupport$
input: $\mathcal{D}$ // binary distorted data set
input: $rThreshold$, $rThreshold > 0$
input: $relax$
input: $rrelax$
$\mathcal{F}_1$ ={1-itemsets which are frequent based on estimated original support of singletons,
that is, have estimated support greater than or equal to $\frac{minimumSupport}{1+relax}$ }
for all $X \in \mathcal{F}_1$ **do begin**
$X.\mathcal{D}_R = \mathcal{D}$
$X.R = X$
end
$index = 1$
$currentRelax = relax$ // relaxation
for $(m = 2; \mathcal{F}_{m-1} \neq \emptyset; m$++$)$ **do begin**
$index$++
$\mathcal{X}_m$ = $aprioriGen(\mathcal{F}_{m-1})$ //generate new candidates
if $index > rThreshold$ **then begin**
for all $X \in \mathcal{X}_m$ **do begin**
$X.\mathcal{D}_R = \{O \in X.\mathcal{D}_R | O$ supports $X.R_F$ with a high probability in the true database$\}$
end
end
$supportCount(\mathcal{X}_m)$
$\mathcal{F}_m = \{X \in \mathcal{X}_m | X.R.\mathbf{C}^T_{2^m-1} \geq \frac{minimumSupport}{1+currentRelax}\}$
if $index > rThreshold$ **then begin**
$index = 1$
$currentRelax = currentRelax + rrelax$ // reduction relaxation
end
end
return $\bigcup_m \mathcal{F}_m$

Let $relax$ be the parameter of the relaxation, for instance, $relax = 0.02 = 2\%$. The relaxed minimum support ($minimumSupport'$) is calculated as follows:

$$minimumSupport' = \frac{minimumSupport}{1 + relax} = \frac{minimumSupport}{1 + 0.02}.$$

Moreover, the relaxation can be repeated every time the reduction of the distorted transaction set is performed. We will call it the *reduction relaxation* (with the *rrelax* parameter). The modified minimum support ($minimumSupport'$) in the reduction relaxation is calculated as follows:

$$minimumSupport' = \frac{minimumSupport}{1 + rrelax}.$$

These two relaxations can be combined at the same time in the MMASK scheme, we will call this scheme rMMASK (please, see the PPApriori-rMMASK algorithm, Algorithm 13).

At the beginning, the PPApriori-rMMASK algorithm finds frequent sets with relaxation equal to the parameter $relax$. Then, the relaxation is increased by the parameter $rrelax$ in each iteration if the length of the reduced itemset is greater than $rThreshold$. As the result of the PPApriori-rMMASK algorithm, the itemsets with the estimated support greater than or equal to relaxed $minimumSupport$ are provided.

4.6. Experimental Evaluation

In this section, we present the results of the experiments conducted to check the accuracy and the efficiency of the modified MASK scheme.

4.6.1. Data Sets

Our experiments were carried out on the following databases:

— A database, Led-24 [84] from UCI Machine Learning Repository [20], which contains information about Light Emitting Diode. There are about 3200 tuples with 25 attributes, 24 of them are binary. One attribute was not distorted because it is a nominal attribute and was treated as 0 for value of 0 and 1 for the remaining values to transform it to a binary attribute.

— A real database, Dna from UCI Machine Learning Repository [20], which contains information about DNA sequences. There are 2000 tuples with 180 binary attributes and 1 nominal attribute. One attribute was not distorted because it is a nominal attribute and was treated as 0 for value of 0 and 1 for the remaining values to transform it to a binary attribute.

— A synthetic database, T10I8D100kN100, generated from the IBM Almaden generator [6]. The data set was created with parameters T=10 (the average size of the transactions), I=8

Table 4.1. The results of mining the frequent sets in the set T10I8D100kN100 with parameters p = 0.5, q = 0.97, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	97	2.5	1.0	0.0	97	97	2.5	1.0	0.0	97
2	2522	2520	7.2	3.6	3.5	2432	2520	7.2	3.6	3.5	2432
3	10930	11553	11.6	9.3	15.0	9910	11553	11.6	9.3	15.0	9910
4	10185	12758	17.1	17.6	42.9	8389	7474	13.6	29.2	2.6	7214
5	2021	4499	26.3	26.0	148.6	1495	665	21.5	67.2	0.1	662
6	24	440	49.4	58.3	1791.7	10	2	0.7	95.8	4.2	1
7	0	4	-	-	-	0	0	-	-	-	0

Table 4.2. The results of mining the frequent sets in the set T10I8D100kN100 with parameters p = 0.5, q = 0.87, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	98	5.4	1.0	1.0	97	98	5.4	1.0	1.0	97
2	2522	2704	20.4	10.8	18.0	2250	2704	20.4	10.8	18.0	2250
3	10930	16780	37.5	25.9	79.5	8094	16780	37.5	25.9	79.5	8094
4	10185	22411	62.4	40.1	160.2	6098	7787	16.4	36.3	12.7	6490
5	2021	5810	115.9	57.9	245.4	851	1129	16.6	57.3	13.2	862
6	24	200	318.5	79.2	812.5	5	28	6.8	70.8	87.5	7

(the average size of the maximal potentially frequent itemsets), D=100k (the number of transactions), N=100 (the number of items). More information on a generator and naming convention can be found in [6]). The data set contains about 100000 tuples with each customer purchasing about ten items on average.

— A synthetic database, T20I16D50kN200, generated from the IBM Almaden generator [6]. The data set was created with parameters T=20 (the average size of the transactions), I=16 (the average size of the maximal potentially large itemsets), D=50k (the number of transactions), N=200 (the number of items). It contains about 50000 tuples with each customer purchasing about 20 items on average.

4.6.2. Accuracy vs Privacy

The first experiment was conducted on the synthetic database with the distortion parameters of $p = 0.5$ and $q = 0.97$, and no relaxation. The results of this experiment are shown in Table 4.1. The level (denoted as L. in tables), which corresponds to the consecutive iterations in Apriori-like algorithms, indicates the length of frequent itemsets, $|F_0|$ indicates the number of frequent itemsets at a given level, $|F_r|$ ($|F_{rm}|$) shows the number of mined frequent itemsets from the distorted database using MASK (MMASK). The other columns are the measures defined in Section 4.4.

Table 4.3. The results of mining the frequent sets in the set T10I8D100kN100 with parameters p = 0.5, q = 0.77, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	95	8.3	4.1	1.0	94	95	8.3	4.1	1.0	94
2	2522	2733	47.3	20.3	28.7	2010	2733	47.3	20.3	28.7	2010
3	10930	19677	127.7	42.8	122.9	6248	19677	127.7	42.8	122.9	6248
4	10185	20248	305.9	68.9	167.7	3168	8764	31.9	62.6	48.7	3809
5	2021	1515	594.0	92.5	67.5	151	1319	30.9	69.2	34.5	622
6	24	3	-	100.0	12.5	0	74	99.7	70.8	279.2	7

The results indicate that, firstly, for MASK the support error (ρ) is less than 10% for the two first levels. The support error grows up to about 50% for 6-itemsets.

The modified algorithm achieves the same results for level 1-3 because $rThreshold$ is 3. For higher levels ρ is less than 22%, which is more than 2 times less than for the original algorithm.

Secondly, the negative identity errors are high for both algorithms, especially for level 6 - about 50% for the original algorithm and 95% for the modified (see the experiment with the relaxation). The positive identity errors are very high for MASK (about 150% for level 5 and more than 10 times higher for level 6). For levels 1-3 the highest positive error is for level 3, namely 15%. For higher levels the positive identity error for MMASK does not exceed 5%.

The comparison measures f_r and f_{rm} show that for levels higher than 3 MMASK discovers less true frequent sets.

Summarising, MMASK for $p = 0.5, q = 0.97$ has lower support error and positive error and higher negative error. This makes f_r higher than f_{rm}.

Tables 4.2 and 4.3 show the results of the experiment for the set T10I8D100kN100 with lower q (higher privacy), $q = 0.87$ and $q = 0.77$, respectively.

Decreasing value of q (p is constant and equal to 0.5) results in increasing privacy. For $q = 0.97, 0.87$ and 0.77 Basic Privacy is equal to 63.8%, 81% and 86.1%, respectively. Thus, a drop of q from 0.97 to 0.87 causes Basic Privacy to increase by more than 17%.

As stated in [98], MASK performs much more worse with lower probabilities for $p = q$. Conducted experiments confirmed this property of MASK (constant p and variable q).

MMASK performs significantly better for lower probabilities. The support error for MASK is as high as 300-600% for levels 5-6, when for MMASK does not exceed 17% for levels 4-6 with $q = 0.87$ and 32% for levels 4-5 with $q = 0.77$. Level 6 is critical for $q = 0.77$, because support error is 99.7%, but it is still better than MASK, because the original algorithm has not discovered any true frequent set. There is the only one case when MMASK performs worse than MASK the positive error for level 6 and $q = 0.77$.

Table 4.4. The results of mining the frequent sets in the set Dna with parameters p = 0.5, q = 0.97, rThreshold = 3, minimumSupport = 0.05

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	181	181	4.4	0.0	0.0	181	181	4.4	0.0	0.0	181
2	13126	11715	13.7	17.2	6.5	10867	11715	13.7	17.2	6.5	10867
3	11118	23213	17.2	30.5	139.3	7723	23213	17.2	30.5	139.3	7723
4	1403	9314	25.7	36.3	600.1	894	4118	22.0	47.3	240.8	739
5	174	1245	34.9	49.4	664.9	88	91	13.0	74.1	26.4	45
6	4	69	57.8	50.0	1675.0	2	0	-	100.0	0.0	0

Table 4.5. The results of mining the frequent sets in the set Dna with parameters p = 0.5, q = 0.87, rThreshold = 3, minimumSupport = 0.05

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	181	181	7.2	0.0	0.0	181	181	7.2	0.0	0.0	181
2	13126	10467	29.1	29.4	9.1	9266	10467	29.1	29.4	9.1	9266
3	11118	89833	40.2	44.3	752.3	6197	89833	40.2	44.3	752.3	6197
4	1403	73920	58.3	57.1	5225.8	602	22635	39.5	57.8	1571.1	592
5	174	6554	60.2	82.2	3748.9	31	153	16.6	88.5	76.4	20
6	4	87	-	100.0	2175.0	0	0	-	100.0	0.0	0
7	0	1	-	-	-	0	0	-	-	-	0

To sum up, MMASK is significantly better with higher privacy (lower probability q). The accuracy error is always better for MMASK (for levels greater than 3).

Appendix A.1 presents the results of the experiments for the set T20I16D50kN200 with $q = 0.87$ and $q = 0.77$, where p is constans and equals to 0.5.

The results of the experiments with $p = q$ for synthetic data sets are quite similar. The modified algorithm accomplishes better results than MASK for $p = 0.8$ and $p = 0.7$ (for detailed results refer to Appendix A.1).

The experiments on the real database (either with $p = q$ or different p and q) lead to the same conclusions (results for the set Dna are shown in Tables 4.4, 4.5, 4.6 and Appendix A.1).

Table 4.6. The results of mining the frequent sets in the set Dna with parameters p = 0.5, q = 0.77, rThreshold = 3, minimumSupport = 0.05

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	181	181	12.1	0.0	0	181	181	12.1	0.0	0	181
2	13126	9423	67.0	38.9	11	8019	9423	67.0	38.9	11	8019
3	11118	105312	96.3	56.9	904	4789	105312	96.3	56.9	904	4789
4	1403	135880	122.3	79.5	9665	287	89646	84.1	76.0	6366	337
5	174	12075	100.0	92.0	6932	14	3807	27.9	87.4	2175	22
6	4	43	-	100.0	1075	0	6	-	100.0	150	0

Table 4.7. The results of mining the frequent sets with the relaxation in the set T10I8D100kN100 with parameters p = 0.5, q = 0.87, relax = 0.05, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	98	5.4	1.0	1.0	97	98	5.4	1.0	1.0	97
2	2522	2802	20.5	9.6	20.7	2279	2802	20.5	9.6	20.7	2279
3	10930	17991	37.3	24.6	89.2	8244	17991	37.3	24.6	89.2	8244
4	10185	24610	61.8	38.9	180.6	6220	8688	16.6	32.0	17.3	6921
5	2021	6523	114.1	56.9	279.6	872	1397	17.1	51.6	20.7	979
6	24	239	318.5	79.2	975.0	5	39	7.6	62.5	125.0	9

Table 4.8. The results of mining the frequent sets with the reduction relaxation in the set T10I8D100kN100 with parameters p = 0.5, q = 0.87, rrelax = 0.05, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	98	5.4	1.0	1.0	97	98	5.4	1.0	1.0	97
2	2522	2704	20.4	10.8	18.0	2250	2704	20.4	10.8	18.0	2250
3	10930	16780	37.5	25.9	79.5	8094	16780	37.5	25.9	79.5	8094
4	10185	22411	62.4	40.1	160.2	6098	8679	16.6	32.1	17.3	6916
5	2021	5810	115.9	57.9	245.4	851	1411	17.2	51.5	21.3	980
6	24	200	318.5	79.2	812.5	5	39	7.6	62.5	125.0	9

Table 4.9. The results of mining the frequent sets with both relaxations in the set T10I8D100kN100 with parameters p = 0.5, q = 0.87, relax = 0.01, rrelax = 0.02, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	98	5.4	1.0	1.0	97	98	5.4	1.0	1.0	97
2	2522	2718	20.4	10.7	18.4	2253	2718	20.4	10.7	18.4	2253
3	10930	17000	37.5	25.7	81.2	8123	17000	37.5	25.7	81.2	8123
4	10185	22816	62.3	39.9	163.9	6123	8340	16.5	33.7	15.6	6756
5	2021	5925	115.3	57.6	250.8	857	1304	17.0	53.6	18.2	937
6	24	208	318.5	79.2	845.8	5	35	8.0	66.7	112.5	8

Table 4.10. The results of mining the frequent sets with different randomisation factors for items in the set T10I8D100kN100 (p = 0.5 .. 0.4, q = 0.87 .. 0.88, rThreshold = 3, minimumSupport = 0.005)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	97	6.2	1.0	0.0	97	97	6.2	1.0	0.0	97
2	2522	3041	36.8	8.0	28.5	2321	3041	36.8	8.0	28.5	2321
3	10930	23321	65.9	26.3	139.7	8056	23321	65.9	26.3	139.7	8056
4	10185	31175	133.9	51.7	257.8	4916	7824	20.8	48.0	24.9	5293
5	2021	4645	307.2	83.3	213.1	338	1165	18.9	62.4	20.0	760
6	24	27	-	100.0	112.5	0	29	4.0	91.7	112.5	2

Relaxation

MASK and MMASK lead to the high false negative error. To reduce it, the relaxation can be used. Table 4.7 shows the results with 5% relaxation for MASK, i.e., rMASK and MMASK, i.e., rMMASK.

For rMASK and rMMASK compared to the case without the relaxation (compare the results presented in Table 4.2) the support error is similar. The false negative error is smaller than without the relaxation. As a negative result of the relaxation, the false positive error is higher. Lower minimum support causes the number of discovered frequent sets (true frequent sets also) to grow.

The results with 5% reduction relaxation are shown in Table 4.8. The support error and identity errors are quite similar compared to 5% relaxation (Table 4.7). f measure for levels 1-3 is lower because the reduction relaxation works for levels higher than $rThreshold$. However, for levels 4-6 f measure is almost the same for both types of the relaxation. Second difference is that the reduction relaxation does not influence the original MASK scheme.

Both relaxations can be combined. The results with 1% relaxation and 2% reduction relaxation are shown in Table 4.9. This combined relaxation is not as strong as the relaxations presented above - the number of correctly discovered frequent itemsets is lower compared with both relaxation applied separately for MMASK and levels 4-6.

By combining relaxations, we can control the number of discovered frequent itemsets of particular length. A higher relaxation results in more discovered frequent itemsets for all lengths of itemsets. A reduction relaxation applied on a particular level makes the number of discovered frequent itemsets higher for this and higher levels.

Different Randomisation Factors for Items

Tables 4.10 and 4.11 show the results of the experiments with different randomisation factors for different items for *T10I8D100kN100* and *Led-24* databases, respectively. The half of the

Table 4.11. The results of mining the frequent sets with different randomisation factors for items in the set Led-24 with parameters p = 0.5 .. 0.4, q = 0.87 .. 0.88, rThreshold = 3, minimumSupport = 0.01

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	25	25	3.3	0.0	0.0	25	25	3.3	0.0	0.0	25
2	300	300	10.7	0.0	0.0	300	300	10.7	0.0	0.0	300
3	2279	1871	24.3	18.3	0.4	1862	1871	24.3	18.3	0.4	1862
4	4713	4053	30.7	47.8	33.8	2461	4033	26.1	40.5	26.1	2805
5	2654	2070	28.1	76.8	54.7	617	2381	23.8	60.4	50.2	1050
6	413	130	17.2	96.9	28.3	13	242	16.3	88.4	47.0	48
7	20	0	-	100.0	0.0	0	0	-	100.0	0.0	0

Table 4.12. The results of mining the frequent sets with different randomisation factors for items and the relaxation in the set Led-24 (p = 0.5 .. 0.4, q = 0.87 .. 0.88, relax = 0.01, rThreshold = 3, minimumSupport = 0.01)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	25	25.0	3.3	0.0	0	25	25	3.3	0.0	0.0	25
2	300	300.0	10.7	0.0	0	300	300	10.7	0.0	0.0	300
3	2279	1882.0	24.3	17.8	0.39491	1873	1882	24.3	17.8	0.4	1873
4	4713	4134.0	30.6	47.1	34.861	2491	4134	26.0	39.7	27.4	2843
5	2654	2144.0	27.9	76.2	56.9706	632	2527	23.7	59.1	54.3	1086
6	413	141.0	17.2	96.9	30.9927	13	274	17.3	87.4	53.8	52
7	20	0.0	-	100.0	0	0	0	-	100.0	0.0	0

Table 4.13. The results of mining the frequent sets with different randomisation factors for items and the reduction relaxation in the set Led-24 (p = 0.5 .. 0.4, q = 0.87 .. 0.88, rrelax = 0.02, rThreshold = 3, minimumSupport = 0.01)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	25	25	3.3	0.0	0.0	25	25	3.3	0.0	0.0	25
2	300	300	10.7	0.0	0.0	300	300	10.7	0.0	0.0	300
3	2279	1871	24.3	18.3	0.4	1862	1871	24.3	18.3	0.4	1862
4	4713	4053	30.7	47.8	33.8	2461	4161	25.9	39.5	27.8	2851
5	2654	2070	28.1	76.8	54.7	617	2591	23.6	58.6	56.3	1098
6	413	130	17.2	96.9	28.3	13	284	17.7	87.2	55.9	53
7	20	0	-	100.0	0.0	0	0	-	100.0	0.0	0

Table 4.14. The results of mining the frequent sets with different randomisation factors for items and both relaxations in the set Led-24 (p = 0.5 .. 0.4, q = 0.87 .. 0.88, relax = 0.01, rrelax = 0.02, rThreshold = 3, minimumSupport = 0.01)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	25	25	3.3	0.0	0.0	25	25	3.3	0.0	0.0	25
2	300	300	10.7	0.0	0.0	300	300	10.7	0.0	0.0	300
3	2279	1882	24.3	17.8	0.4	1873	1882	24.3	17.8	0.4	1873
4	4713	4134	30.6	47.1	34.9	2491	4273	25.8	38.7	29.4	2888
5	2654	2144	27.9	76.2	57.0	632	2716	23.5	57.6	59.9	1125
6	413	141	17.2	96.9	31.0	13	310	18.4	85.5	60.5	60
7	20	0	-	100.0	0.0	0	1	-	100.0	5.0	0

Table 4.15. The results of mining the frequent sets with different randomisation factors for items in the set T10I8D100kN100, p=0.5 .. 0.4, q=0.87 .. 0.88, rThreshold = 2, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	97	6.2	1.0	0.0	97	97	6.2	1.0	0.0	97
2	2522	3041	36.8	8.0	28.5	2321	3041	36.8	8.0	28.5	2321
3	10930	23321	65.9	26.3	139.7	8056	10013	14.4	18.2	9.8	8937
4	10185	31175	133.9	51.7	257.8	4916	10174	19.7	34.5	34.4	6668
5	2021	4645	307.2	83.3	213.1	338	507	23.0	76.6	1.7	472
6	24	27	-	100.0	112.5	0	2	7.7	91.7	0.0	2

items was distorted with parameters $p = 0.4$, $q = 0.88$ and the remaining items with parameters $p = 0.5$, $q = 0.87$.

The results once again show that MMASK scheme is better than MASK for low probabilities. In both experiments, f measure is higher for the modified algorithm (for levels greater than $rTreshold$, namely 3 in this exeperiment). Only once the modified algorithm performs worse (the positive error for *Led-24* database on level 6).

As shown in the experiments presented in this chapter, high false negative errors can be reduced by means of the relaxation. Tables 4.12, 4.13, and 4.14 show the results of the experiments with 1% relaxation (Table 4.12), 2% reduction relaxation (Table 4.13), and those relaxations combined (Table 4.14) for the set Led24 and different randomisation factors. We can notice the fall of the false negative error and the growth of the false positive error. The higher relaxation is, the higher f measure is achieved. The combined relaxation is stronger in reducing false negative errors than those single relaxations (the individual relaxation parameters are the same) as well as in increasing f measure.

The additional results for the experiments with applied relaxation are shown in Appendix A.1.

Summarising, MMASK is better for lower probabilities when different randomisation factors are used for items. The relaxation can be used to reduce the false negative error and boost f measure.

4.6.3. Accuracy and rThreshold Parameter

Tables 4.15 and 4.10 show the results of the experiments on the synthetic database for $rThreshold = 2$ and $rThreshold = 3$, respectively. The results of the experiments for different values of $rThreshold$ were significantly worse. Analysing those two experiments, we can notice that the results for level 3 are better with $rThreshold = 2$. For higher levels some measures are better for $rThreshold = 3$, others for $rThreshold = 2$.

Table 4.16. The results of mining the frequent sets with different randomisation factors for items in the set Led-24, p=0.5 .. 0.4, q=0.87 .. 0.88, rThreshold = 2, minimumSupport = 0.01

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	25	25	3.3	0.0	0.0	25	25	3.3	0.0	0.0	25
2	300	300	10.7	0.0	0.0	300	300	10.7	0.0	0.0	300
3	2279	1871	24.3	18.3	0.4	1862	2155	13.5	6.1	0.7	2139
4	4713	4053	30.7	47.8	33.8	2461	4644	16.7	22.7	21.3	3641
5	2654	2070	28.1	76.8	54.7	617	2583	19.2	45.3	42.6	1453
6	413	130	17.2	96.9	28.3	13	225	13.2	80.9	35.4	79
7	20	0	-	100.0	0.0	0	1	-	100.0	5.0	0

The experiments for *Led-24* database (Tables 4.16 and 4.11) were conducted with the parameter $rThreshold = 2$ and $rThreshold = 3$ (no relaxation), respectively.

For *Led-24* database the results were the best for the values of $rThreshold$ mentioned above, namely 2 and 3. The results for $rThreshold = 2$ are better for level 4-6. For level 7 the false positive error is higher than for $rThreshold = 3$.

The additional experimental results can be found in Appendix A.1. Based on the performed experiments, we can say that the best value of $rThreshold$ depends on a given data set.

Choosing $rThreshold = 2$, we can expect shorter time of the mining process. But significant growth in the number of discovered frequent sets may lead to reversed proportion of processing time. On the synthetic database the process with $rThreshold = 2$ is more than 4 times faster, but on the real database the process with $rThreshold = 2$ is slower less than 20% (there are more frequent itemsets for level 3-5 for $rThreshold = 2$).

4.6.4. Efficiency

Figures 4.1 and 4.2 show the running time of the original algorithm based on Apriori and the modified algorithm, as compared to Apriori itself, for various settings of the minimum support parameter for *Led-24* and *T10I8D100kN100* database, respectively. The experiments with the original algorithm and MMASK were conducted on the distorted database and Apriori was applied on the original database. Running time of rMMASK with the relaxation equal to 1% and the reduction relaxation equal to 2% was very close to MMASK time (e.g., for the set *T10I8D100kN100* and rMMASK time was about 1-2% higher than for MMASK) and was omitted to better visualise differences in running time.

Figures 4.1 and 4.2 show that there are huge differences in running times between MASK and Apriori algorithm – specifically, mining the distorted database with the original algorithm

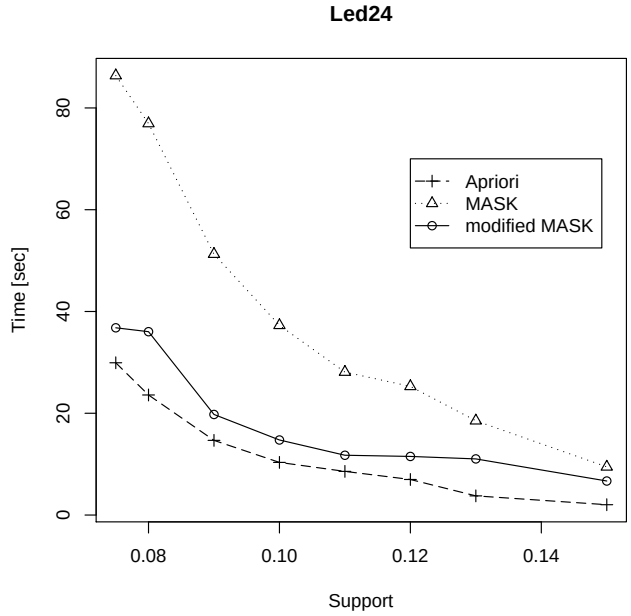

Figure 4.1. Time [s] vs support for mining the association rules in the set Led-24.

can take as much as three to four times more time than in the case when the original database is mined. Overheads between those two algorithms are larger for lower minimum support.

The presented optimisation MMASK, which generates candidates in the Apriori-like fashion, makes the time of the mining process almost as fast as Apriori. This is the advantage, which makes MMASK viable in practice (EMASK cannot use different randomisation factors for 0's and 1's values).

The reduction in time cost could be also related to lower number of frequent items discovered by MMASK than MASK.

4.7. Conclusions and Future Work

We investigated the problem of the effectiveness and efficiency in the privacy preserving MASK scheme, and proposed the new optimisation, MMASK, which is different from those presented in literature devoted to privacy preserving data mining. MMASK estimates support of itemsets based on a reduced subset of distorted transaction. The main advantage of this optimisation is that it breaks the exponential complexity of estimating support of an itemset with respect to its cardinality and makes discovering frequent itemsets and, by this, association rules with preserving privacy viable in practice. The next advantage is that the proposed optimisation can be used with different randomisation factors for 0's and 1's, that is, when an item

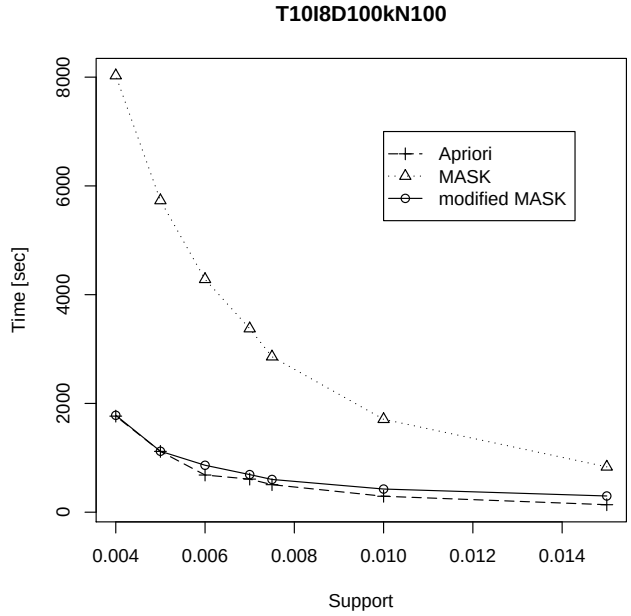

Figure 4.2. Time [s] vs support for mining the association rules in the synthetic set T10I8D100kN100.

is not present and is present in an original database. Moreover, it allows different items to have different randomisation factors. Furthermore, for high levels of privacy it achieves significantly better results than the original MASK scheme. The effectiveness and efficiency of the new solution have been tested on the synthetic and real databases.

In future work, we plan to investigate the possibility of extension of our results to quantitative [108] and generalised [107] association rules.

We also plan to investigate which subset of a candidate should be chosen as a condition to reduce a transaction set to achieve the best accuracy when a candidate length exceeds $reductionThreshold$ parameter. Another possible solution is to combine the results obtained from subsets of a candidate.

We plan to determine the best way to choose $C^T_{2^n-1}$ transactions while choosing distorted transactions which support a candidate itemset with a high probability in a true database. Now the first $C^T_{2^n-1}$ transactions are chosen.

We also plan to determine what are the best values for the $rThreshold$ and find a rule to help a miner to choose the best value of the $rThreshold$ for a given set.

5. Ordered Attributes in Privacy Preserving Classification

The solutions presented in literature devoted to privacy preserving classification with the usage of randomisation-based methods do not take into account a special case of attributes: ordinal and integer attributes. For encryption techniques the order preserving encryption for integer attributes has been proposed [4], nevertheless, for randomisation-based methods such attributes have been treated as nominal in case of ordinal attributes and continuous (real) for integer attributes. Integer attributes treated as continuous may have non-integer values after a distortion, which are not consistent with a domain of an original attribute. The difference between ordinal attributes and numeric attributes with integer values is that ordinal attributes have a finite number of values contrary to integer attributes with an infinite domain. With regard to this difference, these types of attributes should be considered separately.

In this section, we present our new approach for ordinal and integer attributes [17]. This approach takes an order and a domain of an attribute into account during a distortion and reconstruction procedure. It means that it is possible to have different probabilities of changing an original value which depend on a distance between an original value and a potential (allowed according to a domain of an original attribute) distorted value and maintain a domain of a modified attribute the same as a domain of an original attribute.

Having the same domain for an original and modified attribute, the same metadata for both original and modified values can be used. This advantage gives a higher protection against data disclosure when a distortion procedure and its parameters are only revealed to authorised users to keep data more private, that is, in the case of confident data disclosure, a potential attacker cannot figure out whether it is original or modified data based on domains of attributes.

To better protect data against unwanted disclosure by not providing any additional information for a potential attacker, for instance, whether an attribute is modified or not, distributions of modified attributes should be similar to distributions of original attributes. Also, the proposed methods enable one to maintain similar distribution of an attribute before and after modification.

Furthermore, similar distributions of an original and modified attribute prevent parameters

of a distortion procedure from being estimated by a potential attacker based on distributions of distorted attributes. For example, an attacker knowing a probable distribution of an original attribute (based on a nature of a concept described by an attribute or a distribution of a similar attribute for a population) and a distribution of a modified attribute (based on disclosed distorted data) may assume a given type of a distortion procedure and try to estimate parameters of this procedure. Having estimated parameters of a distortion procedure, an attacker can perform a data mining process on disclosed distorted data with high accuracy.

The method proposed for ordinal attributes corresponds to the additive perturbation for continuous attributes and can be used to distort discretised attributes with the same manner as continuous attributes.

Ordered attributes distorted using the presented methods can be used simultaneously with nominal and continuous attributes, hence one is able to build a classifier, for instance, a decision tree, with preserved privacy over data containing all mentioned types of attributes.

5.1. Ordinal Attributes in Privacy Preserving Data Mining

Treating ordinal attributes as nominal attributes (without an order) may lead to loss of some information. In privacy preserving data mining, using solutions which take into account an order of possible values of an attribute is even more important because we want to have the possibility of changing the values of an attribute using a randomisation-based method with different probabilities of changing an original value which depend on a distance between an original value and a potential distorted value.

Let us consider a discretised *salary* attribute with 5 possible values: *low*, *med-low*, *medium*, *med-high*, *high*, which has the order and finite number of possible values. Thus, is an ordinal attribute. The attribute is distorted according to the randomisation-based method presented in Section 3.3.1 with the probability of retaining an original value equal to p and the probabilities that an original value will be changed to a different value equal to $\frac{1-p}{k-1}$, where k is the number of possible values of the attribute. The sum of all probabilities gives 1.

$$p + (k-1)\frac{1-p}{k-1} = 1$$

In this approach, the probabilities that an original value v_i will be changed to v_j, $j = 1 \ldots k, i \neq j$, are equal. Thus, the order of possible values of the attribute *salary* is not taken into account, for instance, value *med-low* may be changed to either *low* or *high* with the same

probability (there is no difference in the probability of changing the original value *med-low* to the value *low*, that is, the value which is the nearest neighbour according to the order and changing the original value *med-low* to the value *high*, that is, the value which is the k-th nearest neighbour, where $k > 1$, for instance, in the case of the considered attribute *salary* third nearest neighbour).

It is more probable that there are more people with *low* value than *high* and changing value *med-low* to the highest value of the attribute *salary* will significantly change the distribution of the attribute after distortion, what may help a potential attacker to estimate parameters of a distortion procedure in the case when a distortion procedure and its parameters are only revealed to authorised users to keep data more private.

To solve this issue, we may assign probabilities of changing values in the following way (Algorithm 14):

Algorithm 14 PAO, the algorithm for the probability assigning for ordinal attributes

input: l // number of neighbour values (separately right and left) to be assigned
// with a given probability
input: $v_0, v_1, \ldots, v_{k-1}$ // k possible values of a given attribute
input: $r_0 = p, r_1, \ldots, r_l$ // given probabilities for neighbour values, $2l < k$
output: $a_{i,j}, 0 \leq i \leq k-1, 0 \leq j \leq k-1$ // elements of matrix $\mathbf{P}$ initialised to 0,
// $a_{i,j} = Pr(v_j \rightarrow v_i)$
for $(i = 0; i < k; i$++$)$ **do begin**
$a_{i,i} = r_0$
for $(s = 1; s < l; s$++$)$ **do begin**
$a_{(i-s) \bmod k,i} = r_s$
$a_{(i+s) \bmod k,i} = r_s$
end
end

Let us assume that we assign probabilities of retaining/changing a given original value of an attribute, for instance, the value *medium* for the attribute *salary*, to all values of this attribute, for example, *low*, *med-low*, *medium*, *med-high*, *high* for the attribute *salary*.

First we assign the probability of retaining an original value (for instance, $Pr($*medium* $\rightarrow$ *medium*$)$). Then we assign the probabilities of changing an original value of an attribute to the values that are the right and left nearest neighbours according to the order of possible values of the attribute (for example, $Pr($*medium* $\rightarrow$ *med-low*$)$ and $Pr($*medium* $\rightarrow$ *med-high*$)$). In the next step, we assign the probabilities of changing the value to the values that are the right and left second nearest neighbours (for instance, $Pr($*medium* $\rightarrow$ *low*$)$ and $Pr($*medium* $\rightarrow$ *high*$)$), etc. We continue until all values we want an original value of the attribute to be changed to are processed (for the remaining values, we assign a probability equal to 0).

This procedure is repeated for all other possible original values of an attribute, for example, the values *low*, *med-low*, *med-high*, *high* for the attribute *salary*.

To implement this solution, we propose to use $\mathbf{P}$ matrix (for details about $\mathbf{P}$ matrix please see Section 3.3.1).

Example An example matrix $\mathbf{P}$ for the presented approach, with the first and second nearest neighbours as values to be changed to, and an ordinal attribute with six possible values is the following:

$$\begin{pmatrix} p & r_1 & r_2 & 0 & r_2 & r_1 \\ r_1 & p & r_1 & r_2 & 0 & r_2 \\ r_2 & r_1 & p & r_1 & r_2 & 0 \\ 0 & r_2 & r_1 & p & r_1 & r_2 \\ r_2 & 0 & r_2 & r_1 & p & r_1 \\ r_1 & r_2 & 0 & r_2 & r_1 & p \end{pmatrix}. \tag{5.1}$$

The probability of retaining an original value is equal to p, r_1 (separately for each neighbour value) is the probability that an original value will be changed to first nearest neighbour values according to the order of possible values of the attribute and r_2 (separately for each neighbour value) is the probability that an original value will be changed to second nearest neighbour values. As the probabilities in the columns should sum to 1, $p + 2r_1 + 2r_2 = 1$.

□

Example Let us assume that the attribute $salary$ will be distorted with the following parameters: the probability of retaining an original value is equal to 0.7 and the probability that an original value will be changed to first nearest neighbour values is equal to 0.15 for each neighbour. Thus, the probabilities are normalised in each column because $0.7 + 0.15 + 0.15 = 1$.

The matrix $\mathbf{P}$ in this case will look as follows:

$$\begin{pmatrix} 0.7 & 0.15 & 0 & 0 & 0.15 \\ 0.15 & 0.7 & 0.15 & 0 & 0 \\ 0 & 0.15 & 0.7 & 0.15 & 0 \\ 0 & 0 & 0.15 & 0.7 & 0.15 \\ 0.15 & 0 & 0 & 0.15 & 0.7 \end{pmatrix}. \tag{5.2}$$

□

In the last example, the value *med-low* of the attribute *salary* is changed to the value *low*

with the probability 0.15 and to *medium* with the same probability. The value cannot be changed to *high*.

With the probability 0.15, the value *low* is changed to *high* and from *high* to *low*. We will call this the *loop effect*. It can be disadvantageous in some cases, for instance, $salary$ attribute mentioned earlier, when we do not want to change the value *low* to *hi* to maintain the distribution before and after distortion.

When we do not want to change the highest value to the lowest and the lowest to the highest, we may use $\mathbf{P}$ with the eliminated *loop effect*. To this end (see Algorithm 15), we choose the probability of retaining an original value of an ordinal attribute. Then we assign probabilities of changing a value of an attribute to right and left first, second, ... nearest neighbour values according to the order of possible values of the attribute until we reach a value with a minimal and maximal index, i.e., an index equal to 0 and $k-1$, where k is the number of possible values of the attribute. In other words, we stop when we reach the lowest or the highest possible value of an attribute.

Algorithm 15 PAOELE, the algorithm for probability assigning for ordinal attributes without loop effect

input: l // number of neighbour values (separately right and left) to be assigned
// with a given probability
input: $v_0, v_1, \ldots, v_{k-1}$ // the k possible values of a given attribute
input: $r_0 = p, r_1, \ldots, r_l$ // the given probabilities for neighbours, $2l < k$
output: $a_{i,j}, 0 \leq i \leq k-1, 0 \leq j \leq k-1$ // elements of matrix $\mathbf{P}$ initialised to 0,
// $a_{i,j} = Pr(v_j \rightarrow v_i)$
for $(i = 0; i < k; i$++$)$ **do begin**
$a_{i,i} = r_0$
for $(s = 1; s < l; s$++$)$ **do begin**
if $0 \leq i - s$ **then** $a_{i-s} = r_s$
if $i + s < k$ **then** $a_{i+s} = r_s$
end
end

Example The example matrix $\mathbf{P}$ with the eliminated *loop effect*, which corresponds to matrix 5.2, will look as follows:

$$\begin{pmatrix} 0.7 & 0.15 & 0 & 0 & 0 \\ 0.15 & 0.7 & 0.15 & 0 & 0 \\ 0 & 0.15 & 0.7 & 0.15 & 0 \\ 0 & 0 & 0.15 & 0.7 & 0.15 \\ 0 & 0 & 0 & 0.15 & 0.7 \end{pmatrix}. \tag{5.3}$$

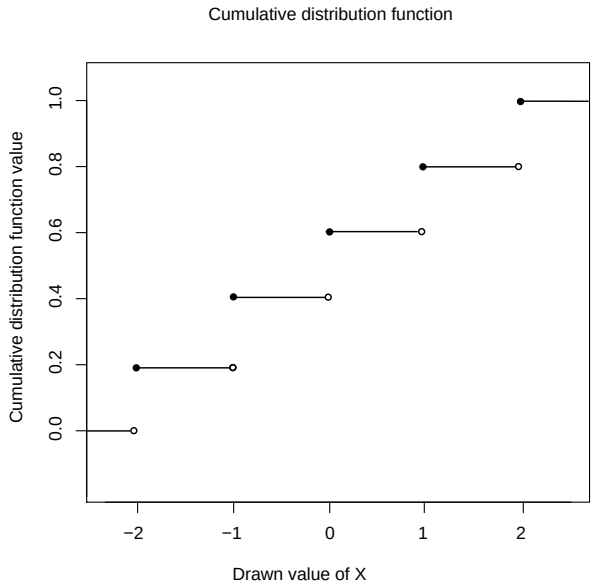

Figure 5.1. The staircased cumulative distribution function.

For each column of $\mathbf{P}$ should be normalised to 1. This process was omitted for better visualisation of the *loop effect* removal.

The following matrix represents proper values of probabilities, normalised in columns:

$$\begin{pmatrix} 0.824 & 0.15 & 0 & 0 & 0 \\ 0.176 & 0.7 & 0.15 & 0 & 0 \\ 0 & 0.15 & 0.7 & 0.15 & 0 \\ 0 & 0 & 0.15 & 0.7 & 0.176 \\ 0 & 0 & 0 & 0.15 & 0.824 \end{pmatrix}. \tag{5.4}$$

□

To reconstruct an original distribution of an attribute distorted according to the proposed method, we may use, for instance, the EM/AS [14] or EQ [15] algorithms (for details refer to Section 3.7.2), which we proposed in [14] and [15], respectively.

5.2. Integer Attributes in Privacy Preserving Data Mining

In the case of an integer attribute and the additive perturbation, we would like to distort its values by adding some noise, for example, by adding random values drawn from a known distribution, for instance, uniform, normal, etc. Adding noise values, which are continuous,

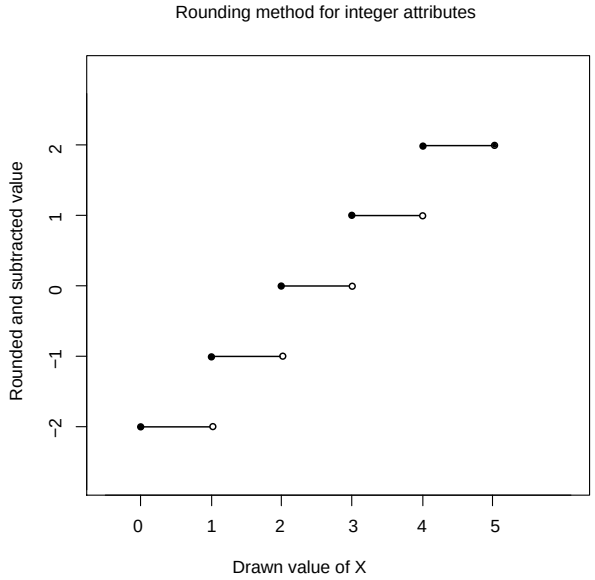

Figure 5.2. The rounding method for integer attributes.

would change an integer attribute to a continuous one. As an example we will consider the attribute *age*, which has nonnegative integer values.

To address this issue, we propose to use a special cumulative distribution function. We will call it the *staircased* cumulative distribution function [17]. This function increases in integer values, hence by adding random values drawn by means of this function, we obtain only integer values (in the considered case of the attribute *age*, only non-negative integer values).

Example Figure 5.1 shows an example of a staircased cumulative distribution function. The drawn values are $-2, -1, 0, 1, 2$. They have the same probability, namely $\frac{1}{5}$. This cumulative distribution function corresponds to a uniform distribution.

□

A second approach we propose is to draw the noise value from a known distribution (for instance, uniform) and round the drawn value. One should be careful about a method of rounding, for example, compare intendent values of probabilities of drawing each rounded value with obtained probabilities, for instance, using the uniform distortion distribution over the range $\langle -5, 5\rangle$ may give different values of probabilities for rounded values when values $\langle 4.5, 5\rangle$ are rounded to 5 and values $\langle 3.5, 4.5)$ are rounded to 4, the probability for the value 4 is two times higher then for the value 5. Furthermore, the method of rounding should be the same during a distortion and reconstruction procedure.

Example To achieve the same distortion cumulative distribution function with rounding ap-

Table 5.1. The original, distorted, and reconstructed probability distribution without and with the *loop effect* for modified attribute employment from set Credit-g. Probability: 0.6 0.2, the EM/AS algorithm.

Value index	Value	Original	Distorted	Recon-structed	Distorted with loop	Reconstructed with loop
0	unemployed	62	84	61	80	49
1	< 1	172	176	165	195	225
2	$1 \leq X < 4$	339	281	343	261	277
3	$4 \leq X < 7$	174	234	197	218	209
4	$7 \leq X < 10$	253	177	225	189	232
5	$10 \leq X < 15$	0	48	4	49	4
6	$15 \leq X < 20$	0	0	1	0	0
7	≥ 20	0	0	0	8	0

proach as for the staircased cumulative distribution function from the previous example, we can use, for instance, the uniform distribution from 0 to 5, round the drawn value to the nearest lower or the same integer value (however, the value equal to 5 should be rounded to 4) and subtract 2. As depicted in Figure 5.2, the rounding method is important, for example, rounding values to a nearest integer value (0.4 to 0, and 0.6 to 1) would results in a different cumulative distribution function of noise.

□

To reconstruct an original distribution of an attribute distributed in accordance with the described method, one may use, for instance, the AS [7] or EM [2] algorithms (for details about the AS and EM algorithms please refer to Section 3.7.2).

The presented solutions, that is, the *staircased* cumulative distribution function and the rounding method, can be also applied in a straightforward manner to the retention replacement perturbation.

5.3. Experimental Evaluation

This section presents the results of the experiments conducted with ordinal and integer attributes.

In the experiments, the accuracy, sensitivity, specificity, precision and F-measure (for definitions please refer to Section 2.5) were used. All statistics were computed as the mean of 100 multiple runs.

5.3.1. Probability Distribution Reconstruction for Ordered Attributes

Table 5.1 shows the original, distorted and reconstructed[1] probability distribution of the modified *employment* attribute which describes *the number of years for present employment* (set *Credit-g*[2]). We increased the number of possible values of the attribute to better show the *loop effect*. The original number of possible values of the attribute was 5, we changed the value ≥ 7 with index 4 to the value $7 \leq X < 10$ and added to the domain of the attribute 3 following possible values $10 \leq X < 15$, $15 \leq X < 20$, ≥ 20. However, we did not add any original observations for these 3 additional values, hence the values of the original probability distribution for these values are 0. Clearly, the possible values of the attribute have the order.

According to Table 5.1, there are no values ≥ 20 after distortion performed without the *loop effect* (in Table 5.1, the distorted probability distribution obtained without the *loop effect* is denoted as *Distorted*), so we distorted the original values of the attribute according to the order of possible values of this attribute and we did not change the lowest value to the highest, which might be important when, for instance, we do not want to change the number of years for present employment from *unemployed* to ≥ 20 for 19 years old person.

We used the probability of retaining an original value equal to 0.6 and the probability of changing an original value to the first nearest neighbour equal to 0.2 to distort the values of the attribute. In the case of distortion without the *loop effect*, the matrix $\mathbf{P}$ was normalised. While distorting data with the *loop effect*, we allowed the lowest value to be changed to the highest possible value (please refer to the column *Distorted-loop* in Table 5.1 for the distorted probability distribution of the modified *employment* attribute distorted with the *loop effect* for the value ≥ 20).

Information loss and privacy based on differential entropy after a distortion[3] in the case of distortion without the *loop effect* were equal to 0.0603 and 5.30, respectively, and in the case of distortion with the *loop effect* 0.0643 and 5.50, respectively. When we distorted values of the attribute *employment* without taking the order into account, that is, we used the probability of retaining an original value equal to 0.6 and assumed equal probabilities of changing an original value, information loss and privacy based on differential entropy after a distortion were equal to 0.0623 and 6.95, respectively.

Distorting values of the attribute without the *loop effect* we had lower information loss and

[1] EM/AS algorithm (Section 3.7.2) was used to reconstruct an original probability distribution function.

[2] All sets used in tests can be downloaded from UCI Machine Learning Repository [20].

[3] Details about privacy based on differential entropy can be found in Section 3.5.4. Information loss is defined in Section 3.7.2.

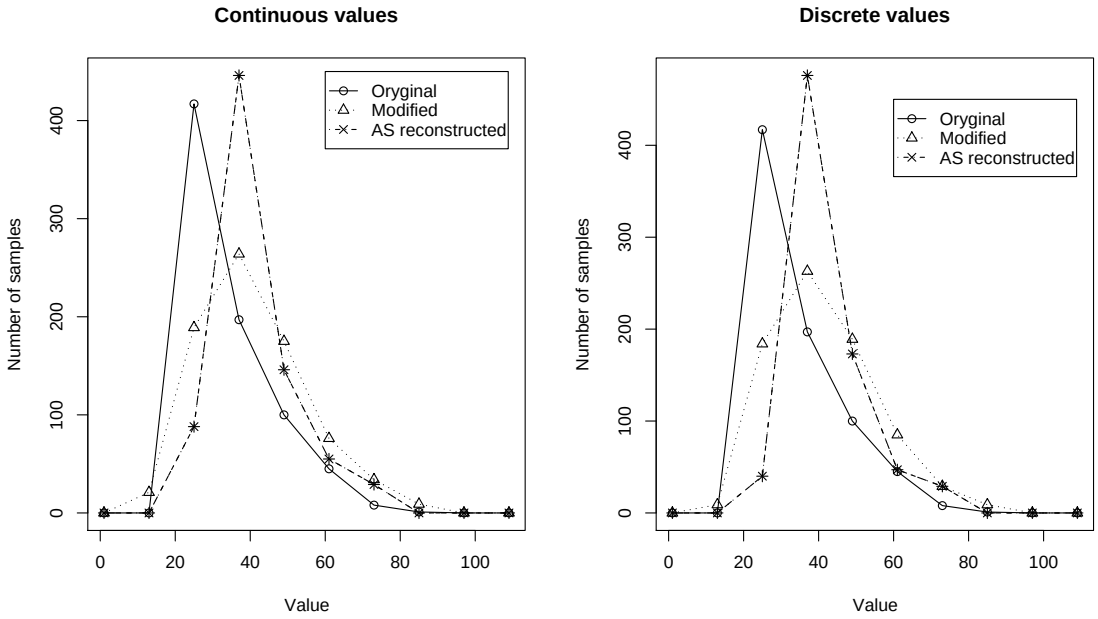

Figure 5.3. The probability distribution reconstruction for the continuous (on the left) and discrete (on the right) domain of age attribute from set Diabetes.

privacy compared to the case while values were distorted without taking the order into account. To obtain the same level of privacy while distorting values of the attribute according to the order for the case without the *loop effect* and while distorting values of the attribute without taking the order into account, we should lower the probability of retaining an original value.

Figure 5.3 shows the original, distorted and reconstructed probability distribution of *age* attribute (set *Diabetes*) for the continuous and discrete domain (the additive perturbation with the *staircased* cumulative distribution function and the uniform distribution were used).

Information loss for the continuous case was equal to 0.4529 and for discrete 0.4585. Both values were high in general because the reconstructed peak was moved to the right compared to the original distribution. Information loss for the discrete case was slightly higher, but the distorted attribute has higher privacy, namely 56.33 (55.85 for continuous case), because, in general, the higher privacy, the higher information loss, that is, preserving greater privacy causes greater information loss.

5.3.2. Classification with Ordered Attributes

In the next experiments, we performed classification with a decision tree over data distorted according to the order and without taking the order into account. We showed only two best types of reconstruction: *By class* and *Local* (for details about reconstruction types refer to Section

Table 5.2. Accuracy, sensitivity, specificity, precision, and F measures of classification with the usage of ordered attributes for Local (LO) and By class (BC) reconstruction types, the AS.EM/AS algorithms and 100% of privacy.

Set	Ord.	Acc.		Sens.		Spec.	
		LO	BC	LO	BC	LO	BC
Credit-g	yes	0.6796	0.6952	0.3682	0.3654	0.8152	0.8380
	no	0.6777	0.6963	0.3674	0.3758	0.8129	0.8352
Breast	yes	0.9173	0.9244	0.8161	0.8361	0.9736	0.9745
	no	0.9217	0.9287	0.8297	0.8514	0.9728	0.9731
Diabetes	yes	0.6620	0.6796	0.8679	0.9153	0.2807	0.2403
	no	0.6621	0.6798	0.8683	0.9158	0.2806	0.2398

Set	Ord.	Prec.		F	
		LO	BC	LO	BC
Credit-g	yes	0.4622	0.4921	0.4009	0.4100
	no	0.4568	0.4951	0.3980	0.4183
Breast	yes	0.9551	0.9578	0.8752	0.8887
	no	0.9549	0.9577	0.8833	0.8975
Diabetes	yes	0.6930	0.6927	0.7653	0.7842
	no	0.6930	0.6927	0.7655	0.7844

3.7.1). As a decision tree we used SPRINT [103] modified to incorporate privacy according to Sections 3.7.1 and 3.7.2 (for details please refer to Appendix B). The algorithms for building the decision tree over distorted data described in Sections 3.7.1 and 3.7.2 (AS algorithm for continuous attributes and EM/AS for nominal attributes) were used. We applied the additive random perturbation.

Table 5.2 presents the results of classification of *Credit-g*, *Brest*, and *Diabetes* sets for two cases: when we do not take the order into account and when the attributes with the order are distorted as shown in Sections 5.1 and 5.2.

Comparing these two cases, we can say that the accuracy, sensitivity, specificity, precision and F-measure are really close. The difference is less than 0.02.

The experiments have proved that we can use integer attributes, distort them and store as integers. Moreover, we can distort ordinal attributes according to their order. These methods allow a miner to obtain the results with compared quality preserving the same domain of attributes and distorting attributes in accordance with their order.

5.4. Conclusions and Future Work

We presented our new approach to ordered attributes in privacy preserving classification in the case when the randomisation-based method and a centralised database are used.

We proposed how to proceed with ordinal and integer attributes. A domain of an integer attribute after distortion is preserved by means of the presented methods. Moreover, ordinal attributes can be distorted and reconstructed according to their order using these methods, that is, the probabilities of changing an original value can depend on a distance between an original and modified value.

Effectiveness of the new approach was tested on real data sets. The results of the experiments showed that the proposed methods achieved comparable results as the methods for nominal and continuous attributes. However, the new methods allow a miner to use the same domain for integer attributes and distort ordinal attributes according to the order of possible values of these attributes.

In future, we plan to investigate the possibility of extension of our approach to preserve privacy for target/class attribute.

6. Privacy Preserving Emerging Patterns

In this chapter, we investigate whether we may take an advantage of discriminating power of Emerging Patterns (EPs) in classification over data with preserved privacy.

The process of classification with the usage of EPs depends on a type of a classifier we use. There are two main types of classifiers: (i) eager learners and (ii) lazy learners. An example of the former ones is CAEP [41], whereas of the latter ones is DeEPs [69]. For more details about CAEP and DeEPs please see Section 2.3.

In this chapter, we propose the eager Privacy Preserving Classifier with Emerging Patterns, ePPCwEP, which utilises CAEP schema and lazy Privacy Preserving Classifier with Emerging Paterns, lPPCwEP, based on DeEPs for privacy preserving classification for centralised data modified by means of the randomisation-based methods.

6.1. Eager Approach to Classification in Privacy Preserving Data Mining with Emerging Patterns

In our eager learner schema in the ePPCwEP algorithm [18], EPs are mined once from a training data set and then are used to calculate a category for each test sample. The process of training the eager classifier based on EPs consists of two steps:

1. Discovering EPs.
2. Calculating statistics used for classification of test samples.

Having learned a classifier, a category for test samples is determined based on the statistics calculated during the training phase.

The entire process of building the eager classifier with Emerging Patterns and classifying a test set is shown in Algorithm 16.

The two steps of the training phase and the testing process are described in the following sections. For the sake of simplicity, we assume all attributes to be binary. We will show how to extend our approach to nominal and continuous attributes in Section 6.1.4.

Algorithm 16 ePPCwEP, the eager Privacy Preserving Classifier with Emerging Patterns

input: $\mathcal{D}$ // distorted training set (can contain binary, nominal and continuous attributes)
input: $\mathcal{S}$ // undistorted test set
input: $minimumSupport$
input: ρ // threshold for growth rate
// Training phase
// 1. Discovering EPs
transform all continuous and nominal attributes to binary attributes
partition $\mathcal{D}$ into $\mathcal{D}_i, i = 1, ..., k$ subsets according to the class labels $\mathcal{C}_i$
$\mathcal{D}'_i$ is the opponent subset to $\mathcal{D}_i$, $\mathcal{D}'_i = \mathcal{D} \setminus \mathcal{D}_i, i = 1, ..., k$
for $(i = 1; i < k; i++)$ **do begin**
mine frequent sets with the estimated support greater than or equal to $minimumSupport$ using MMASK from $\mathcal{D}_i$ and $\mathcal{D}'_i$
store supports of frequent sets
find EPs from $\mathcal{D}'_i$ to $\mathcal{D}_i$ with growth rate greater than or equal to the ρ threshold based on frequent sets
// 2. Calculating statistics
calculate the aggregate scores for all training instances for class $\mathcal{C}_i$
calculate the base score $baseScore(\mathcal{C}_i)$ for class $\mathcal{C}_i$
end
// Testing phase
for each test instance $S \in \mathcal{S}$ **do begin**
calculate aggregate and normalised score of S for each class $\mathcal{C}_i$
assign to S the class $\mathcal{C}_j$ for which S has the largest normalised score
end

6.1.1. Discovering EPs

The first step of the process of building an eager learning classifier is to discover EPs from a training data with distorted values. To this end, we perform the following procedure.

Partitioning Data

We partition a distorted training data set $\mathcal{D}$ into subsets $\mathcal{D}_i, i = 1, ..., k$ according to the class labels $\mathcal{C}_i$ which are not distorted. For each class we have a pair of subsets: $\mathcal{D}_i$ containing instances with class label $\mathcal{C}_i$ and the second subset $\mathcal{D}'_i$ gathering the remaining instances, the opponent class $\mathcal{D}'_i = \mathcal{D} \setminus \mathcal{D}_i$.

Mining Frequent Sets

For each class $\mathcal{C}_i$ we mine frequent sets, that is, sets with estimated original supports greater than or equal to $minimumSupport$, from $\mathcal{D}_i$ and $\mathcal{D}'_i$ separately. To this end, we use Apriori approach to generate candidates with n-items from frequent $(n-1)$-itemsets.

While preserving privacy, an original support of an itemset cannot be directly counted from a distorted training set. Hence, to check a minimal support condition, we use the MASK pro-

cedure of estimating an original n-itemset support based on distorted training set, as described in Section 3.6.1, and apply the MMASK optimisation presented in Chapter 4 to reduce time complexity.

Estimated supports of frequent sets are stored for further calculations.

Contrasting Classes with EPs

Having mined frequent sets for all subsets $\mathcal{D}_i$ and $\mathcal{D}'_i, i = 1, ..., k$, we find EPs from $\mathcal{D}'_i$ to $\mathcal{D}_i$ with a growth rate greater than or equal to the ρ threshold and call them the EPs of a class $\mathcal{C}_i$.

As we have already estimated the supports for frequent sets, any simple method can be used to find EPs. A naive method is to check for every frequent set from a subset $\mathcal{D}_i$ whether there is a corresponding frequent set in a subset $\mathcal{D}'_i$. In case of not finding a corresponding frequent set, we know that a growth rate equals ∞ and we have discovered a Jumping Emerging Pattern. For cases where both subsets contain a given itemset, we need to calculate the growth rate and choose only EPs with the growth rate greater than or equal to ρ. The calculated growth rates are stored, because we will need them in the next step of training a classifier.

In our approach we used the presented above method for finding EPs because we have estimated the supports of the frequent itemsets and we can easily find EPs and calculate growth rates. Moreover, in this method we do not need to perform other estimations of supports of EPs and we avoid an additional, high cost of estimating support in the case of privacy preserved data. In fact, we have focused more on the idea of EPs in privacy preserving classification than on the efficiency.

6.1.2. Calculating Statistics

Having discovered EPs with a growth rate greater than or equal to the ρ threshold, we need to calculate statistics which will be used during the testing phase. To this end, we adapt the CAEP [41] classifier. The process is summarised below.

First, aggregate scores $score(S, \mathcal{C})$ of all training instances for all classes are calculated (like in Equation 2.1 for original CAEP). Given an instance S and a set $E(\mathcal{C})$ of EPs of a class $\mathcal{C}$ discovered from a training data set, the aggregate score of S for $\mathcal{C}$ is defined as:

$$score(S, \mathcal{C}) = \sum_{e \subseteq s, e \in E(\mathcal{C})} \frac{egrowthRate(e)}{egrowthRate(e) + 1} esup_{\mathcal{C}}(e), \tag{6.1}$$

where $esup_{\mathcal{C}}(e)$ is the estimated suport calculated according to MMASK (see Chapter 4) and $egrowthRate(e)$ is calculated based on the estimated supports.

Then, the $baseScore(\mathcal{C}_i)$ for each class $C_i, i = 1, ..., k$, are found. The $baseScore(\mathcal{C}_i)$ is a score for the training instances of class C_i at a fixed percentile, for example, 75%.

6.1.3. Testing Phase

According to the privacy preserving schema proposed in [7], the testing set is not distorted. Thus, having base scores, all EPs, its estimated supports and growth rates, we classify test samples, which have undistorted values of attributes, like in original CAEP. Hence, as stated in [41], for each test sample S and class $\mathcal{C}_i, i = 1, ..., k$, the aggregate score, $score(S, \mathcal{C}_i)$, is calculated (see Equation 6.1). Then, the aggregate scores are normalised and sample S is assigned to the class $\mathcal{C}_j$ for which S has the largest normalised score.

6.1.4. Non-binary Attributes in Privacy Preserving Emerging Patterns Mining and Eager Approach to Classification

In order to use nominal and continuous attributes in the presented algorithm, we propose to transform attributes of these types into binary attributes and calculate randomisation probabilities for transformed attributes.

Nominal Attributes

We transform a nominal attribute A with k possible values to k binary attributes, A_1 to A_k, in the following way: an attribute A_i has the value of 1 when the attribute A is equal to the i-th value, otherwise the attribute A_i has the value of 0.

There are two natural scenarios of distorting a nominal attribute with k values in a training set. The first is to transform a nominal attribute to k binary attributes and then distort all k attributes. The second is to distort a nominal attribute and after that transform it to k binary attributes. The first scenario may result in improper values of k attributes, for example, two 1's simultaneously, and may lead to logical contradiction in data. Let us consider the discretised attribute age with values $\langle 0, 10)$; $\langle 10, 20)$; $\langle 20, 30)$; $\langle 30, 40)$, etc. While distorting binary attributes after transformation, we may obtain, for example, values of 1 for binary attributes corresponding to values $\langle 10, 20)$ and $\langle 30, 40)$, which means that the age of the person is in the range $\langle 10, 20)$ and in the range $\langle 30, 40)$ simultaneously.

In both scenarios, we need to fix randomisation parameters. For binary attributes distorted after a transformation, we set the probability of retaining the original value of 1 to p_i and

analogous probability for 0 to q_i for each attribute ($i = 1, ..., k$). In the case when binary attributes have the same privacy, please note that $p_i = p,\ i = 1, ..., k$ and $q_i = q,\ i = 1, ..., k$. In particular, q_i may be equal to p_i. Having transformed a nominal attribute into binary attributes, new attributes are distorted according to the probabilities p_i and $q_i, i = 1, ..., k$. While distorting and estimating the support, we assume that the binary attributes $A_1, ..., A_k$ are independent.

In the second scenario, we set a distortion parameters for a nominal attribute before a transformation, distort an attribute and then calculate p_i and q_i for all binary attributes, $i = 1, ..., k$. We will show the idea of calculating probabilities.

Let us consider the case when the probability of retaining an original value for each k possible values of the attribute A is equal to p. The probability of changing a value is the same for each value of the attribute and is equal to $\frac{1-p}{k-1}$. Thus the matrix $\mathbf{P}$ of retaining/changing values of the nominal attribute A looks as follows:

$$\mathbf{P} = \begin{pmatrix} p & \frac{1-p}{k-1} & \frac{1-p}{k-1} & \dots & \frac{1-p}{k-1} \\ \frac{1-p}{k-1} & p & \frac{1-p}{k-1} & \dots & \frac{1-p}{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1-p}{k-1} & \frac{1-p}{k-1} & \frac{1-p}{k-1} & \dots & p \end{pmatrix}.$$

Let p'_i and q'_i be the probabilities of retaining the value of 1 and 0 for i-th binary attribute, respectively. The probabilities of retaining an original value for all values of the nominal attribute are the same, p, as well as the probabilities of changing an original value, $\frac{1-p}{k-1}$, thus $p'_i = p'$ and $q'_i = q'$ for $i = 1, ..., k$.

Now we will calculate the value of p'. To this end, we look at the matrix $\mathbf{P}$. p is the probability that the attribute A, which has the value v_i, will have the same value after a distortion. $p' = p'_i$ is the probability that the attribute A_i which has the value 1 will have the same value after a distortion. When the attribute A before a distortion has the value v_i, the attribute A_i has the value 1. The relation between the values of the attributes A and A_i also holds after a distortion. Thus, $p' = p$.

To calculate q', we need to look closer at the matrix $\mathbf{P}$. For the sake of simplicity, we focus on q'_1, that is, the probability that 0 of the binary attribute which comes from v_1 will be kept. q'_1 can be computed as follows:

$$q'_i = q'_1 = P_A(V_{A'} \in \{v_2, v_3, ..., v_k\} | V_A \in \{v_2, v_3, ..., v_k\})$$
$$= P_A(v_2)[P_A(v_2 \to v_2) + P_A(v_2 \to v_3) + \ldots + P_A(v_2 \to v_k)]$$

$$+P_A(v_3)[P_A(v_3 \to v_2) + P_A(v_3 \to v_3) + \ldots + P_A(v_3 \to v_k)]$$
$$+ \ldots + P_A(v_k)[P_A(v_k \to v_2) + P_A(v_k \to v_3) + \ldots + P_A(v_k \to v_k)],$$

where $Pr(v_p \to v_r)$ is a probability that the value v_p will be changed to the value v_r and A' represent the attribute A after a distortion, V_A is the value of the attribute A.

Let us assume that $P_A(v_i) = \frac{1}{k}$, i.e., there is the same number of samples in a training data set for each value of the attribute. If we have a particular training set, we can estimate the original number of samples for each value of an attribute and alter the calculations shown below (for details about estimation please see Section 3.7.2). Given $P_A(v_i) = \frac{1}{k}$, we get:

$$q'_i = q'_1 = \frac{1}{k-1}(k-1)[(k-2)\frac{1-p}{k-1} + p] = \frac{k-2p}{k-1}.$$

Finally, $\mathbf{P}'$ matrix has the following elements:

$$\mathbf{P}' = \begin{bmatrix} p & 1 - \frac{k-2p}{k-1} \\ 1-p & \frac{k-2p}{k-1} \end{bmatrix}.$$

Estimating the support in the second scenario, we also assume that the attributes $A_1 ... A_k$ are independent.

The matrix $\mathbf{P}'$ can be calculated for any specific matrix $\mathbf{P}$ in a way similar to the presented above.

Continuous Attributes

The solution for continuous attributes is to discretise an attribute and calculate probabilities of changing/retaining an original value.

This solution applies to both randomisation-based methods for continuous attributes: the additive perturbation and the retention replacement perturbation (for details about these methods please refer to Section 3.3.1, [7], [8]).

The chosen perturbation method does not have an influence on the discretisation process and any discretisation method to divide the domain of the attribute into intervals can be applied. However, the discretisation method influences the way that the distortion procedure parameters are calculated.

The idea of calculating elements of the matrix $\mathbf{P}$ is the same for both perturbation methods, namely, it consists in looking for the probability $P(A = v_i | X = x)$ that the value of a nominal attribute A will be equal to v_i given that a value of the continuous attribute X is equal to x:

$$P(A = v_i | X = x) = P(Z \in I_i | X = x),\ i = 1, ..., k,$$

where Z is a continuous attribute after distortion and I_i is the i-th interval which corresponds to v_i value.

We will show how to compute these probabilities for the retention replacement perturbation. Let assume that we have k intervals after discretisation, i.e., k values of a nominal attribute. Let p be the probability that an original value of a continuous attribute is kept. In the retention replacement perturbation, the probability p does not depend on x, the given value of the attribute X, as well as the probability $P(Z \in I_i|R')$, where R' means we change an original value x of the continuous attribute X during the distortion and R we retain an original value x for the continuous attribute Z.

$$P(Z \in I_i|X = x)$$

$$= pP(Z \in I_i|R) + (1-p)P(Z \in I_i|R')$$

$$= \begin{cases} p + (1-p)P(Z \in I_i|R'); & x \in I_i \\ (1-p)P(Z \in I_i|R'); & x \notin I_i \end{cases}.$$

For the uniform perturbation and intervals with the same length $P(Z \in I_i|R') = \frac{1}{k}$.

Having calculated the above probabilities, we transform the discretised nominal attribute to k binary attributes as shown in the previous section.

6.2. Lazy Approach to Classification in Privacy Preserving Data Mining with Emerging Patterns

In contrast to the eager learner schema, where EPs are mined once from a training data set and then are used to calculate a category for each test sample, in this section we propose how to classify with a lazy approach using Emerging Patterns. A lazy learner waits until it gets a test sample, then it discovers Emerging Patterns in the context of a test sample and chooses a final category.

The modified lPPCwEP algorithm that we proposed in [19] is based on DeEPs (for details about DeEPs refer to Section 2.3.2) and contains three steps, which are repeated for each test sample:

1. Preparation of binary training data,
2. Discovery of Emerging Patterns,
3. Calculation of statistics and choice of a final category.

The entire process of classification of a test sample S is shown in Algorithm 17 and described in the subsequent sesctions.

Algorithm 17 lPPCwEP, the lazy Privacy Preserving Classifier with Emerging Patterns

input: $\mathcal{D}$ // distorted training set (can contain binary, nominal and continuous attributes)
input: S // undistorted test sample
input: $minimumSupport$
input: ρ // threshold for growth rate
input: P_{thr} // threshold for P_{match} probability
for $(i = 1; i < k; i++)$ **do begin** // k is the number of classes
// Prepare binary training data
$\mathcal{D}_i = \{t \in \mathcal{D} \mid t$ has the class label $\mathcal{C}_i\}$
$\mathcal{D}'_i = \mathcal{D} \setminus \mathcal{D}_i$ // $\mathcal{D}'_i$ is the opponent subset to $\mathcal{D}_i$
// transform the set $\mathcal{D}_i$ into the binary set $\mathcal{B}_i$ and $\mathcal{D}'_i$ into the binary set $\mathcal{B}'_i$
for each set $\mathcal{D}_i$ and $\mathcal{D}'_i$ **do begin**
for each attribute and training sample **do begin**
calculate P_{match} probability
assign 1 to a binary attribute in either $\mathcal{B}_i$ or $\mathcal{B}'_i$ if P_{match} probability is greater than P_{thr}
assign 0 otherwise
end
end
// Discover Emerging Patterns
mine frequent sets with the supports greater than or equal to $minimumSupport$ from $\mathcal{B}_i$ and $\mathcal{B}'_i$, separately
store supports of frequent sets
find EPs from $\mathcal{B}'_i$ to $\mathcal{B}_i$ with the growth rate greater than or equal to the ρ threshold based on frequent sets
calculate the value of compact summation, that is, the number of samples which support at least one EP for a given class $\mathcal{C}_i$
end
assign to the test sample S the class $\mathcal{C}_j$ for which S has the largest value of compact summation

6.2.1. Preparation of Binary Training Data

We modified the first step of classification (in DeEPs called Intersection) to meet privacy preserving conditions and described it below.

We are given a classification problem: we would like to classify a test instance S based on a training set of instances with classes $\mathcal{C}_i, i = 1, ..., k,$.

The preparation of binary training data process is performed for each class $\mathcal{C}_i, i = 1, ..., k$, separately. For a given class $\mathcal{C}_i$, the set $\mathcal{D}_i$ denotes all the instances with class label $\mathcal{C}_i$. The remaining instances constitute the set $\mathcal{D}'_i$.

For a test sample S and each attribute for each training sample L, a probability that an original value of an attribute X for the training sample L matches a value of attribute X in the test sample S given a value of a modified attribute Z for the training sample L is calculated, where the attribute Z is obtained by distorting values of the original attribute X.

Nominal Attributes

Let us assume that we are given a test sample S, a training sample L, a probability distribution of a modified attribute (i.e., $P(Z = v_i),\ i = 1, ..., k$)[1] and the parameters of the distortion procedure.

A reconstructed probability distribution ($P(X = v_i),\ i = 1, ..., k$) is estimated performing the reconstruction of a probability distribution using EM/AS algorithm (details about EM/AS algorithm are described in Section 3.7.2).

Let us assume that the distorted value of the attribute Z for the training sample L is equal to v_q ($Z = v_q$) and the original value of the attribute X for the test sample S is equal to v_i ($X = v_i$). We can calculate the probability that the value of the original attribute X for the training sample L is equal to v_i given that the value of the distorted attribute Z for the training sample L is equal to v_q:

$$P_{match}(X = v_i | Z = v_q),\ i = 1, ..., k.$$

Using Bayes' Theorem, we get:

$$P_{match}(X = v_i | Z = v_q) = \frac{P(X = v_i) \cdot P(Z = v_q | X = v_i)}{P(Z = v_q)}.$$

$P(Z = v_q | X = v_i)$ is the probability that the value v_i of the attribute X will be changed to the value v_q and is known because the parameters of a distorting method are known. $P(Z = v_q)$ can be computed based on a training data set and $P(X = v_i)$ estimated, thus $P_{match}(X = v_i | Z = v_q)$ for the nominal attribute can be calculated.

A reconstruction of a probability distribution for an original attribute ($P(X = v_i),\ i = 1, ..., k$) and the calculation of a probability distribution for a modified attribute ($P(Z = v_i),\ i = 1, ..., k$) can be performed using only samples with the same class as a training sample L to better reflect the characteristics of a given class, which can be significantly different from the characteristics of other classes separately or all classes together. In order to use only samples with the same class as a training sample L for the reconstruction, P_{match} probability is calculated as follows:

[1] The probability $P(Z = v_i),\ i = 1, ..., k$ can be calculated using a distorted training data set.

$$P_{match}(X = v_i|Z = v_q) = \frac{P(X = v_i|C(L)) \cdot P(Z = v_q|X = v_i)}{P(Z = v_q|C(L))},$$

where $C(L)$ is a class label for a given training sample.

Continuous Attributes

For the additive perturbation and a continuous attribute, we calculate P_{match} probability in the following way:

Let X be an original attribute. An attribute Y is used to modify X and obtain a modified attribute Z. Let us assume that the attribute Z is equal to the distorted value z for a given training sample L and the value of the attribute X for a test sample S is equal to t.

To estimate P_{match} probability, a neighbourhood-based match with α parameter is used. P_{match} is the probability that an original value of the attribute X for the training sample L belongs to the α-neighbourhood $N(t, \alpha)$ of t given that the value of the distorted attribute Z is equal to z for the training sample L:

$$P_{match}(X \in N(t, \alpha)|Z = z) = P(X \geq t - \alpha \wedge X \leq t + \alpha|Z = z, X + Y = Z) \tag{6.2}$$

$$P_{match}(X \in N(t, \alpha)|Z = z) = F_X(t+\alpha|Z = z, X+Y = Z) - F_X(t-\alpha|Z = z, X+Y = Z), \tag{6.3}$$

where F_X is a cumulative distribution function of X.

$$P_{match}(X \in N(t, \alpha)|Z = z) = \int_{t-\alpha}^{t+\alpha} f_X(r|Z = z, X + Y = Z)dr \tag{6.4}$$

After applying Bayes' Theorem, we obtain:

$$P_{match}(X \in N(t, \alpha)|Z = z) = \int_{t-\alpha}^{t+\alpha} \frac{f_Z(Z = z, X + Y = Z|X = r)f_X(r)}{f_Z(z)}dr. \tag{6.5}$$

Since Y is independent of X and the denominator is independent of the integral, we can write:

$$P_{match}(X \in N(t, \alpha)|Z = z) = \int_{t-\alpha}^{t+\alpha} \frac{f_Y(z - r)f_X(r)}{f_Z(z)}dr. \tag{6.6}$$

f_Y is known and $f_X(r)$ can be estimated as shown in Section 3.7.2, thus $P_{match}(X \in N(t, \alpha)|Z = z)$ can be computed.

As in the reconstruction of a probability distribution of a continuous attribute, a domain of an attribute is divided into intervals (see Section 3.7.2), thus, we replace the α-neighbourhood in the neighbourhood-based match by an interval-based neighbourhood with α parameter.

Please note that the interval-based neighbourhood with α parameter equal to 0 is an interval within which a value of a given attribute for a test sample lies. For α equal to 1, the interval-based neighbourhood covers an interval within which a value of a given attribute for a test sample lies and its right and left first neighbour intervals, etc.

Like for nominal attributes, in order to calculate the probability $P_{match}(X \in N(t,\alpha)|Z = z)$ for a continuous attribute, $f_X(r)$ can be estimated and $f_Z(z)$ calculated based only on samples with the same class as a training sample L.

$$P_{match}(X \in N(t,\alpha)|Z = z) = \int_{t-\alpha}^{t+\alpha} \frac{f_Y(z-r) f_X(r|C(L))}{f_Z(z|C(L))} dr \tag{6.7}$$

For the retention replacement perturbation and a continuous attribute, we calculate the probability P_{match} in the following way:

Let X be an original attribute and Z be a modified attribute obtained from X by means of the retention replacement perturbation with the probability p. Let the attribute Z be equal to the distorted value z for a given training sample L and the value of the attribute X for a test sample S be equal to t.

In order to estimate the probability P_{match}, we use a neighbourhood-based match with α parameter. P_{match} is the probability that an original value of the attribute X for the training sample L belongs to the α-neighbourhood $N(t,\alpha)$ of t given that the value of distorted attribute Z is equal to z.

$$P_{match}(X \in N(t,\alpha)|Z = z) = P(X \geq t-\alpha \wedge X \leq t+\alpha|Z = z) \tag{6.8}$$

Let $\mathbf{1}(condition)$ be an indicator function which takes 1 when $condition$ is met and 0 otherwise.

$$P_{match}(X \in N(t,\alpha)|Z = z) = p\,\mathbf{1}(z \in \langle t-\alpha, t+\alpha\rangle) + (1-p)\int_{t-\alpha}^{t+\alpha} g(r|X)dr, \tag{6.9}$$

where $g()$ is a density function of a distorting distribution used in the retention replacement perturbation.

Since $g()$ is independent of X, we obtain:

$$P_{match}(X \in N(t,\alpha)) = p\,\mathbf{1}(z \in \langle t-\alpha, t+\alpha\rangle) + (1-p)\int_{t-\alpha}^{t+\alpha} g(r)dr. \tag{6.10}$$

Assuming a uniform distribution as the distorting distribution for the retention replacement perturbation, we can write:

$$\begin{aligned} P_{match}(X \in N(t,\alpha)) &= p\,\mathbf{1}(z \in \langle t-\alpha, t+\alpha\rangle) + (1-p)\frac{t+\alpha-t+\alpha}{d_{max}-d_{min}} \\ &= p\,\mathbf{1}(z \in \langle t-\alpha, t+\alpha\rangle) + (1-p)\frac{2\alpha}{d_{max}-d_{min}}, \end{aligned} \tag{6.11}$$

where d_{max} and d_{min} are maximal and minimal value of a domain of the attribute X.

Transformation of Training Set to Binary Data

Having calculated the probabilities P_{match} for nominal and continuous attributes, we choose a value of a probability threshold p_{thr} for each attribute separately. For the probabilities P_{match} greater than p_{thr}, we have 1 for a new attribute corresponding to the attribute X. For values less than or equal to p_{thr}, we assign 0. In the sequel, the binary training set obtained from $\mathcal{D}_i$ as a results of this transformation will be denoted by $\mathcal{B}_i$ and the binary training set obtained from $\mathcal{D}'_i$ by $\mathcal{B}'_i$.

Details about the method of choosing a threshold probability p_{thr} for both types of attributes can be found in Section 6.3.

6.2.2. Discovery of Emerging Patterns

In order to discover EPs from the binary training sets $\mathcal{B}_i$ and $\mathcal{B}'_i$, $i = 1, \ldots, k$, the original procedure from DeEPs can be used. We perform the following procedure.

Mining Frequent Sets

Frequent sets with support greater than or equal to $minimumSupport$ are mined from $\mathcal{B}_i$ and $\mathcal{B}'_i$, $i = 1, \ldots, k$, separately.

To this end, we apply Apriori approach to generate candidates with n-items based on frequent $(n-1)$-itemsets. To check a minimal support condition, supports of itemsets in a binary training set $\mathcal{B}_i$ and a set $\mathcal{B}'_i$ are directly calculated from $\mathcal{B}_i$ and a set $\mathcal{B}'_i$. For each frequent set, its support is stored for further calculations.

Contrasting Test Sample with EPs

Having mined frequent sets from binary training sets $\mathcal{B}_i$ and $\mathcal{B}'_i$, $i = 1, \ldots, k$, EPs from $\mathcal{B}'_i$ to $\mathcal{B}_i$ with the growth rate greater than or equal to the ρ threshold are found.

Any method can be used to find EPs, for instance, the naive one presented in Section 6.1.1. Alternatively, one may find maximal frequent itemsets and borders instead and then find EPs and growth rates [38] [72].

6.2.3. Calculating Statistics

To choose a final class for the test sample S, the compact summation is used for each class $\mathcal{C}_i, i = 1, ..., k$. Thus samples which support at least one EP for a given class $\mathcal{C}_i, i = 1, ..., k$, are calculated. A class $\mathcal{C}_j$ with the highest relative count wins.

6.3. Experimental Evaluation

This section presents the results of the experiments conducted according to the both proposed (eager and leazy) approaches to privacy preserving classification with Emerging Patterns.

All sets used in our tests can be downloaded from UCI Machine Learning Repository [20]. We used the following sets: *Australian*, *Iris*, *Breast*, *Credit-g*, *Diabetes*, *Sick*, *Wine*, *Mushroom*, chosen under the following conditions: at least one set should contain only continuous attributes (*Diabetes*), at least one set should contain both continuous and nominal attributes (*Australian*, *Credit-g*), and at least one of the sets should have a class attribute with multiple values (*Wine*, *Iris*).

In the experiments, accuracy, sensitivity, specificity, precision and F-measure were calculated (see Section 2.5 for definitions). To achieve more reliable results, we applied 10-fold *cross-validation*[2] [36] and calculated the average of 100 multiple runs.

In all experiments, we distorted all attributes except for a class/target attribute for a training data set. To count supports of itemsets in the eager approach, the modified estimation with *reductionTreshold* equal to 3 (empirically chosen) was applied if not explicitly stated that the standard estimation was used. We used the additive perturbation with uniform distortion distribution for continuous attributes, unless explicitly stated that the retention replacement perturbation with a uniform distribution was used. For the additive perturbation, at the beginning of

[2] Cross-validation is a statistical method of evaluating classifiers. In k-fold cross-validation a data set is divided into k of nearly the same cardinality subsets. In k-th iteration k-th subset is used as a test set and the union of the remaining $k - 1$ subsets is used as a training set.

Table 6.1. The results of classification with the eager EP classifier and the decision tree (denoted as T) for p = 0.55, 0.7, 0.9, the minimal support equal to 0.1 and the minimal growth rate 2 (the distortion of binary attributes).

Set	p	Acc.	Sens.	Spec.	Prec	F	t[s]	T acc.	T t[s]
Australian	0.55	0.6812	0.6433	0.7134	0.6518	0.6416	1.850	0.4841	0.297
	0.7	0.8246	0.7617	0.8745	0.8322	0.7934	2.253	0.8230	0.174
	0.9	0.8232	0.8396	0.8051	0.7835	0.8054	1.453	0.8247	0.112
Iris	0.55	0.5867	0.8389	0.8456	0.4631	0.3696	0.030	0.4400	0.033
	0.7	0.9133	0.9644	0.9689	0.7889	0.7728	0.030	0.6467	0.020
	0.9	0.9200	0.9700	0.9733	0.7667	0.7505	0.019	0.8467	0.010

each experiment in the eager approach, a domain of each continuous attribute was discretised into 5 bins each of which covered equal number of samples. For the retention replacement perturbation, the same discretisation was applied to continuous attributes. To determine base scores, 75% threshold was set.

As a decision tree, we used SPRINT [103] modified to incorporate privacy according to Sections 3.7.1 and 3.7.2 (for details please refer to Appendix B). The EM/AS algorithm for nominal attributes was used to reconstruct a probability distribution. The reconstruction for continuous attributes was performed according to the AS algorithm. For all experiments *Local*, one of two the best reconstruction types, was used, i.e., a reconstruction of a probability distribution was performed in every node divided into classes (for more details about reconstruction types please refer to Section 3.7.1).

6.3.1. Eager Approach

First we show the values of accuracy measures and time of classification in the eager approach to classification using Emerging Patterns and compare them to the values of measures obtained for the decision tree.

Experiments with Distorting Binary Attributes

In this experiment, we applied the first scenario of distorting nominal attributes, i.e., having discretised continuous attributes, we binarised all nominal attributes, then assigned to the transformed attributes the probability of retaining an original value ($p \in \{0.55, 0.7, 0.9\}$) and distorted the data. The minimal growth rate for EPs was $\rho = 2$.

Table 6.1 shows the accuracy (Acc.), sensitivity (Sens.), specificity (Spec.), precision (Prec.), F-measure (F) and time (t[s]) for classification with EPs based on the eager approach and accuracy (T acc.), time (T t[s]) for the decision tree classifier.

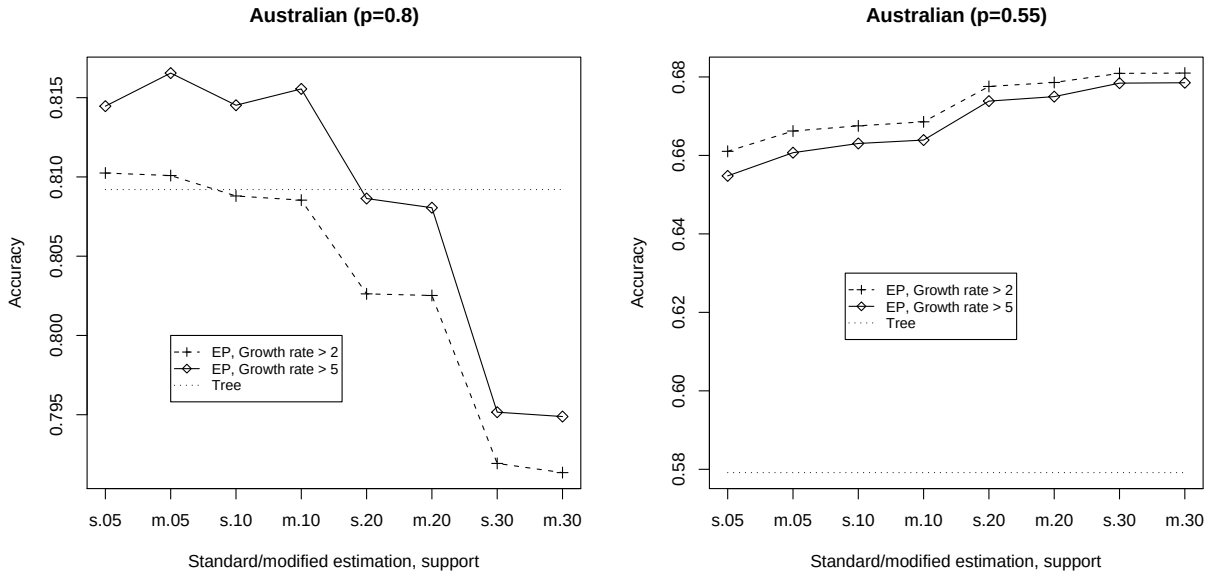

Figure 6.1. Accuracy of classification with the EP classifier and the decision tree (distortion of nominal attributes).

As we expected, lowering the p probability from 0.9 to 0.55 (i.e., increasing privacy) decreases accuracy (there are few exceptions where the measure for $p = 0.7$ is slightly higher than for 0.9). Sensitivity, specificity, precision, and F also decrease. The results for the decision tree showed the same pattern. We can observe that incorporating privacy by a randomisation-based technique for centralised data causes accuracy loss.

Comparing the results between the EP classifier and the decision tree, we may say that for low privacy (i.e., high probability p) the eager EP classifier performed at least as good as the decision tree. However, for high privacy (i.e., $p = 0.55$) the EP classifier yielded significantly better results for all tested sets.

For *Iris* set cumulative time of training and testing phases was similar, but for the set *Australian* with more instances and attributes, the EP classifier performed longer, about an order of magnitude.

Selection of Values of Support and Growth Rate Thresholds

In the next experiment, we tried to determine the influence of a minimal support and a minimal growth rate threshold on classification results.

We discretised continuous attributes and applied the second scenario of distorting nominal attributes, that is, nominal attributes were distorted and then transformed to binary attributes calculating probabilities of retaining an original value for all new attributes, because the first scenario might lead to a logical contradiction in distorted data (see Section 6.1.4).

Table 6.2. The results of classification with the eager EP classifier and the decision tree for p = 0.8, the minimal support equal to 0.05 and the minimal growth rate equal to 5 (in the case of the distortion of nominal attributes).

Set	Acc.	Sens.	Spec.	Prec.	F	Time[s]	T acc.	T time[s]
Australian	0.8166	0.7882	0.8365	0.7985	0.7901	6.46	0.8092	0.22
Iris	0.8579	0.9460	0.9489	0.6915	0.6639	0.04	0.7032	0.12
Breast	0.9643	0.9813	0.9523	0.9210	0.9492	1.00	0.9248	0.01
Diabetes	0.6707	0.6522	0.7019	0.8016	0.7162	2.15	0.6589	0.03
Sick	0.8176	0.8345	0.8163	0.2317	0.3603	4.91	0.9218	0.05
Credit-g	0.6732	0.6544	0.6814	0.4710	0.5417	355.18	0.6661	0.52
Wine	0.8619	0.9413	0.9474	0.6481	0.6196	15.48	0.6074	0.13
Mushroom	0.8893	0.5789	0.9325	0.9376	0.6701	19.48	0.7966	7.73

The probability of retaining an original value for each non-binary attributes was equal to $p \in \{0.55, 0.8\}$. We assumed that the probabilities of changing an original value were the same within each nominal attribute.

Figure 6.1 presents the accuracy of classification for different values of the minimal support (from 0.05 to 0.30) and the minimal growth rate ($\rho = 2$ and 5) in case of low ($p = 0.8$) and high ($p = 0.55$) privacy. We showed the difference between MASK (denoted by *s*) and the MMASK estimation (denoted by *m*) and the results for the decision tree.

For low privacy, we can draw the same conclusions as for the CAEP algorithm without preserving privacy [41], that is, we obtained the best results for small values of the support. We may also say that the accuracy stabilised for the minimal support equal to about 5%. The underlying reason of the similarity of these results is that increasing the probability p we are closing to the case without privacy ($p = 1$), which we have in the case of the original CAEP algorithm.

For high privacy (lower p), we obtained different results. While increasing the minimal support, we observed higher accuracy, which reached the maximum for the minimal support equal to about 30%. The reason might be that while incorporating high privacy, we insert a lot of noise into our data and for a low support threshold we find EPs fitted into that noise. Increasing the minimal support, we reduce overfitting of the classifier.

For the set *Australian*, we obtained better results for the minimal growth rate threshold equal to 5 for low privacy and for the minimal growth rate threshold equal to 2 for high privacy. Nevertheless, the results for other data sets did not confirm this regularity.

We also presented the results for the decision tree. As shown in the previous experiment, for low privacy the EP classifier is only slightly better, but for high privacy it achieves significantly higher accuracy than the decision tree.

Table 6.3. The results of classification with the eager EP classifier and the decision tree for p = 0.55, the minimal support equal to 0.2 and the minimal growth rate equal to 5 (in the case of the distortion of nominal attributes).

Set	Acc.	Sens.	Spec.	Prec.	F	Time[s]	T acc.	T time[s]
Australian	0.6750	0.6352	0.7054	0.6497	0.6287	1.74	0.5792	0.27
Iris	0.6755	0.8723	0.8804	0.5331	0.4655	0.02	0.4141	0.12
Breast	0.9436	0.9094	0.9640	0.9393	0.9224	0.19	0.8627	0.01
Diabetes	0.6058	0.5877	0.6379	0.7517	0.6555	0.31	0.5773	0.03
Sick	0.9085	0.2976	0.9482	0.3272	0.3267	1.07	0.8639	0.06
Credit-g	0.6374	0.3704	0.7521	0.4019	0.3621	5.50	0.5988	0.62
Wine	0.6626	0.8654	0.8749	0.4541	0.3863	1.03	0.3730	0.18
Mushroom	0.7393	0.4267	0.7889	0.3101	0.3302	83.45	0.5033	14.98

The accuracy for the standard and the modified estimation was similar (it is better to choose the modified estimation for low values of a minimal support, as it yielded slightly better results and needed significantly less time than the standard one, see Section 6.3.1).

Experiments with Distorting Nominal Attributes

Once again we applied the second scenario of distorting nominal attributes in the experiment. Tables 6.2 and 6.3 show the results of classification for $p \in \{0.8, 0.55\}$, respectively. According to the conclusions from the previous experiments, we chose the minimal support equal to 0.05 for low and 0.2 for high privacy.

The EP classifier for $p = 0.8$ only once performed worse than the decision tree, and for high privacy $p = 0.55$ yielded much more higher accuracy than the decision tree (even more than 40% better for the set *Wine* and 20% for the set *Iris*, relatively). Thus we conclude that the EP classifier gives better results than the decision tree, especially for the high level of privacy.

Comparing time of training and classifying, the EP classifier needed more time to perform the entire process. The difference was about one order of magnitude (except for the set *Credit-g*, where the classifier needed to discover large number of EPs).

Time of Classification

Time of training and testing the EP classifier with the standard and the modified estimation is shown in Figure 6.2. The minimal growth rate was set to 2 and nominal attributes were distorted according to the second scenario.

For the high support threshold there was not much difference between the two types of the estimation, but for the low support threshold (when itemsets with large number of items are frequent) the modified estimation reduced time to a large extent.

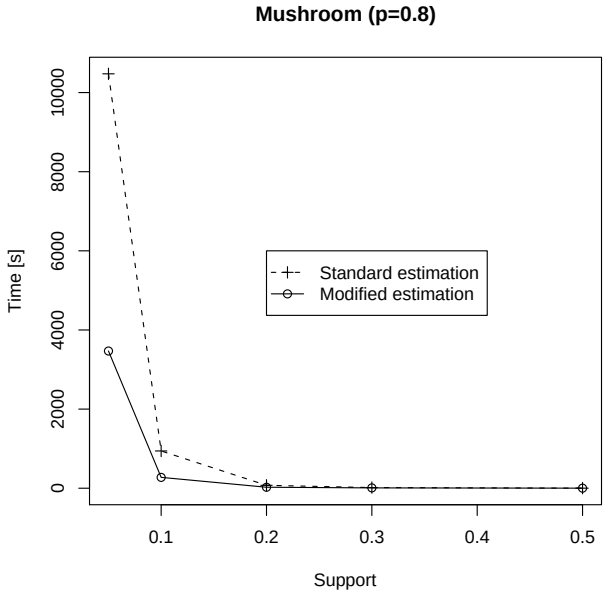

Figure 6.2. Time of classification with EP classifier.

Corresponding time for the decision tree was equal to 7.8 seconds, which was close to the runtime of the EP algorithm with the modified estimation for 0.3 minimal support.

6.3.2. Lazy Approach

This section presents the results of the experiments conducted according to the proposed lazy approach to privacy preserving classification with Emerging Patterns. The eager approach was used for a comparison (nominal and discretised attributes in the eager approach were binarised after the distortion as we proposed in [19], see also Section 6.1.4).

For nominal and continuous attributes, the reconstruction and calculation of the probability P_{match} were performed using only samples with the same class as a training sample, because the empirical results confirmed this was the better approach. Also based on empirical results, we assumed α to be 0 in the interval-based neighbourhood.

For nominal attributes, the probability threshold p_{thr} was set as equal to the probability that an original value will be retained. For continuous attributes the probability threshold p_{thr} was chosen arbitrarily.

Experiments with Different Minimal Support Threshold Values

In this experiment, we compared the accuracy of classification for different values of the support threshold, given minimal growth rate equal to 2 and the probability threshold p_{thr} for continuous attributes equal to 0.5.

Table 6.4. The results of classification with the lazy and eager EP classifiers, the minimal support equal to 0.2 and the minimal growth rate 2, p_{thr} (for continuous attributes) equal to 0.5.

Set	P[%]	Acc.	t[s]	Sens.	Spec.	Prec.	F	eEP acc.	eEP t[s]
Austral.	100	0.7403	0.89	0.6769	0.7933	0.7507	0.6776	0.8031	0.29
	150	0.7121	2.28	0.8131	0.6325	0.6771	0.7063	0.7618	0.53
	200	0.6638	1.41	0.5720	0.7392	0.6831	0.5767	0.6804	1.60
Breast	100	0.7629	0.43	0.3299	0.9915	0.9691	0.4480	0.9650	0.14
	150	0.6972	0.70	0.1292	0.9979	0.9813	0.2114	0.9535	0.15
	200	0.7436	0.56	0.2674	0.9965	0.9813	0.3969	0.9387	0.16
Diabetes	100	0.6236	0.42	0.8480	0.2137	0.6709	0.7347	0.6717	0.07
	150	0.6501	0.59	0.8618	0.2655	0.6903	0.7580	0.6095	0.13
	200	0.6388	0.54	0.8171	0.3147	0.6938	0.7401	0.6168	0.13
Iris	100	0.8633	0.01	0.8680	0.9340	0.8837	0.8410	0.8730	0.01
	150	0.7267	0.02	0.7326	0.8652	0.8004	0.7049	0.6851	0.01
	200	0.8058	0.02	0.8116	0.9034	0.8639	0.7851	0.6673	0.01
Wine	100	0.6357	0.08	0.6637	0.8193	0.6909	0.6179	0.8817	0.56
	150	0.5654	0.11	0.5699	0.7795	0.7096	0.5283	0.6924	1.03
	200	0.6662	0.11	0.6842	0.8319	0.7428	0.6489	0.7160	1.05

Table 6.5. The results of classification with the lazy and eager EP classifiers, the minimal support equal to 0.05 and the minimal growth rate 2, p_{thr} (for continuous attributes) equal to 0.5.

Set	P[%]	Acc.	t[s]	Sens.	Spec.	Prec.	F	eEP acc.	eEP t[s]
Austral.	100	0.7497	1.99	0.6400	0.8410	0.7820	0.6726	0.6805	0.95
	150	0.7286	5.38	0.8091	0.6664	0.7003	0.7114	0.6025	0.65
	200	0.6762	4.44	0.5868	0.7491	0.6880	0.5923	0.6120	0.61
Breast	100	0.8025	0.50	0.4502	0.9886	0.9665	0.5734	0.9650	1.09
	150	0.7329	0.72	0.2410	0.9957	0.9731	0.3656	0.9502	0.60
	200	0.7697	0.60	0.3474	0.9948	0.9760	0.4945	0.9389	0.65
Diabetes	100	0.6456	0.45	0.8965	0.1855	0.6755	0.7631	0.6805	0.95
	150	0.6506	0.60	0.8568	0.2766	0.6919	0.7575	0.6025	0.65
	200	0.6410	0.56	0.8165	0.3229	0.6957	0.7420	0.6120	0.61
Iris	100	0.8898	0.02	0.8924	0.9461	0.9002	0.8748	0.8735	0.04
	150	0.7283	0.02	0.7345	0.8660	0.8009	0.7088	0.6850	0.02
	200	0.8095	0.02	0.8150	0.9051	0.8676	0.7866	0.6685	0.02
Wine	100	0.6621	0.10	0.6928	0.8332	0.7126	0.6423	0.8719	24.21
	150	0.5813	0.11	0.5852	0.7874	0.7167	0.5474	0.6850	6.23
	200	0.6875	0.12	0.7083	0.8429	0.7509	0.6775	0.7179	4.50

Table 6.4 presents the results of the classification with EPs with the lazy and eager approach and (relative) minimal support equal to 0.2. For a given privacy level (denoted as P[%]), we showed for the lazy approach accuracy (Acc.), time of classification (t[s]), sensitivity (Sens.), Specificity (Spec.), precision (Prec.), and F measure (F). For the eager approach accuracy (eEP acc.) and time of classification (eEP t[s]) were shown. Table 6.5 presents the same set of measures for lower minimal support, namely 0.05.

In general, we observed that for the lazy approach the lower the minimal support was, the higher accuracy the classifier yielded. It is not a strict rule because in some cases we did not observe higher accuracy or almost the same (for the set *Diabetes* and 150% level of privacy we observed almost no improvement, namely, 0.0005 higher accuracy for the minimal support threshold equal to 0.05). F-measure showed the same pattern (only for *Australian* set and 100% level of privacy, we observed slightly lower value, which was caused by decrease in sensitivity). Considering sensitivity, specificity and precision, we observed better results in 35 out of 45 cases.

For the eager approach, we observed lower accuracy for 9 out of 15 cases, while lowering the minimal support threshold. Furthermore, we noticed significant drop of accuracy for the set *Australian*.Thus, the above rule, as shown in the previous experiments, did not apply to the eager approach.

Comparing the results between the lazy and eager EP classifiers, we noticed that for the minimal support threshold equal to 0.2, the lazy classifier was better only for *Iris* set with 100 and 150% level of privacy. On the other hand, for the minimal support threshold equal to 0.05, it was better for *Australian*, *Iris* and *Diabetes* but only for 150 and 200% level of privacy.

Based on the presented results, we may say that the lazy approach achieved better results for lower minimal support threshold, and the eager approach for higher minimal support threshold.

Experiments with Jumping Emerging Patterns

In Table 6.6, the results of the experiments for the lazy and eager approach with only jumping emerging patterns, the minimal support equal to 0.05 and p_{thr} (for continuous attributes) equal to 0.5 were shown.

Compared to Table 6.5, the lazy approach performed better for all sets except for *Diabetes*. For the eager approach, we obtained worse results in 11 cases out of 15. We checked the values of the minimal growth rate between 2 and ∞ for the lazy approach and the results seemed to show the general rule that the higher minimal growth rate is, the better results we obtain.

The eager approach did not show this regularity. The reason may be that a score for a given

Table 6.6. The results of classification with the lazy and eager JEP classifiers for the minimal support equal to 0.05 and p_{thr} (for continuous attributes) equal to 0.5.

Set	P[%]	Acc.	t[s]	Sens.	Spec.	Prec.	F	eEP acc.	eEP t[s]
Austral.	100	0.7834	2.00	0.7578	0.8050	0.7658	0.7487	0.8111	8.29
	150	0.7821	5.39	0.9166	0.6747	0.7244	0.7974	0.7422	9.41
	200	0.6846	4.49	0.6418	0.7201	0.6713	0.6282	0.6533	9.92
Breast	100	0.8635	0.50	0.6395	0.9813	0.9507	0.7502	0.9638	1.09
	150	0.7920	0.72	0.4165	0.9893	0.9561	0.5671	0.9489	0.60
	200	0.8439	0.60	0.5718	0.9869	0.9604	0.7018	0.9420	0.65
Diabetes	100	0.6237	0.46	0.7357	0.4132	0.7054	0.7091	0.6563	0.95
	150	0.6392	0.60	0.7767	0.3788	0.7044	0.7317	0.5878	0.65
	200	0.6220	0.57	0.7304	0.4176	0.7036	0.7070	0.5971	0.61
Iris	100	0.9012	0.02	0.9042	0.9519	0.9078	0.8893	0.8559	0.04
	150	0.7298	0.02	0.7366	0.8669	0.8013	0.7116	0.6708	0.02
	200	0.8173	0.02	0.8223	0.9090	0.8723	0.7962	0.6607	0.03
Wine	100	0.6717	0.10	0.7039	0.8391	0.7134	0.6586	0.8656	24.52
	150	0.5871	0.11	0.5908	0.7904	0.7169	0.5573	0.6732	6.24
	200	0.6893	0.12	0.7104	0.8438	0.7501	0.6801	0.7144	4.51

class in the eager approach depends on the growth rate of an emerging pattern. Having only Jumping Emerging Patterns, there is no difference between those JEPs (please refer to Equation 2.1).

Experiments with Different Values of Probability Threshold p_{thr}

In this experiment, we changed p_{thr} for continuous attributes from 0.5 to 0.3 (Table 6.7). Comparing the results with the previous experiments (Table 6.6), we observed that for all cases except for 2 cases we obtained better accuracy. Comparing F-measure we obtained better values for all cases except for one case (*Diabetes* with 150% level of privacy). Decreasing further p_{thr} (to the value of 0.2) caused significantly worse results, as well as increasing p_{thr} value. The results for p_{thr} equal to 0.4 were, in general, better than the results for 0.5 and worse than the results for 0.3.

Moreover, we finally obtained high sensitivity for *Breast* set (while specificity retained its high value) and high specificity for *Diabetes* set with high sensitivity.

p_{thr} was used only in the lazy approach, thus the results for the eager approach did not change while p_{thr} was decreased.

To sum up, we empirically showed that the lazy EP classifier yielded the best classification results for p_{thr} equal to 0.3 and the minimal support 0.05.

To compare the best possible to obtain accuracy of both EP classifiers, we used for the lazy approach the results from Table 6.7 with JEPs, minimal support equal to 0.05, p_{thr} equal to

Table 6.7. The results of classification with the lazy JEP classifiers for the minimal support equal to 0.05 and p_{thr} (for continuous attributes) equal to 0.3.

Set	Priv.[%]	Acc.	Time[s]	Sens.	Spec.	Prec.	F
Australian	100	0.7970	9.76	0.8336	0.7687	0.7479	0.7817
	150	0.7998	9.44	0.9060	0.7140	0.7250	0.7997
	200	0.6709	6.31	0.8004	0.5665	0.6057	0.6796
Breast	100	0.9397	0.84	0.8805	0.9706	0.9411	0.9077
	150	0.8938	0.92	0.7407	0.9741	0.9384	0.8231
	200	0.9421	0.94	0.9157	0.9557	0.9166	0.9140
Diabetes	100	0.6815	0.77	0.7163	0.6163	0.7830	0.7400
	150	0.6111	0.86	0.5381	0.7496	0.8155	0.6261
	200	0.6583	0.99	0.7198	0.5431	0.7572	0.7207
Iris	100	0.9050	0.02	0.9083	0.9541	0.9132	0.8952
	150	0.8152	0.02	0.8196	0.9093	0.8513	0.7999
	200	0.8548	0.02	0.8628	0.9305	0.8796	0.8392
Wine	100	0.7493	0.17	0.7675	0.8745	0.7646	0.7424
	150	0.7044	0.18	0.7193	0.8504	0.7516	0.6901
	200	0.7684	0.21	0.7826	0.8821	0.7952	0.7644

Table 6.8. The results of classification with the lazy JEP classifier, the minimal support equal to 0.05 and eager EP classifier with the minimal support equal to 0.2 and the minimal growth rate equal to 2.

Set	p	Acc.	t[s]	Sens.	Spec.	Prec.	F	e Acc.	e t[s]	T acc.
Aus.	0.5	0.8334	1.28	0.8531	0.8169	0.7916	0.8187	0.8006	0.22	0.8201
	0.3	0.8166	1.44	0.8450	0.7928	0.7735	0.8003	0.7920	0.21	0.7778
	0.15	0.7464	1.18	0.7103	0.7734	0.7297	0.7087	0.6642	1.03	0.6316
Bre.	0.5	0.9136	0.68	0.8197	0.9672	0.9431	0.8742	0.9593	0.18	0.9357
	0.3	0.8715	0.80	0.6959	0.9731	0.9420	0.7940	0.9637	0.13	0.9280
	0.15	0.8796	0.69	0.7200	0.9710	0.9406	0.8034	0.9555	0.13	0.9014
Dia.	0.5	0.6532	0.32	0.5681	0.8125	0.8482	0.6754	0.6844	0.06	0.6815
	0.3	0.6506	0.31	0.5972	0.7486	0.8177	0.6833	0.6680	0.07	0.6449
	0.15	0.6000	0.31	0.5558	0.6811	0.7682	0.6352	0.6208	0.11	0.5992
Iris	0.5	0.8917	0.01	0.9583	0.9613	0.7185	0.6979	0.9073	0.01	0.7907
	0.3	0.8413	0.01	0.9389	0.9426	0.6750	0.6430	0.8667	0.01	0.6921
	0.15	0.6307	0.01	0.8538	0.8649	0.5367	0.4531	0.7157	0.01	0.5096
Wine	0.5	0.8703	0.06	0.9449	0.9515	0.6758	0.6472	0.9189	0.19	0.7823
	0.3	0.7907	0.06	0.9134	0.9213	0.5848	0.5380	0.8544	0.39	0.6578
	0.15	0.5986	0.07	0.8356	0.8495	0.4857	0.3939	0.6994	0.62	0.4625

0.3, and for the eager approach the results from Table 6.4 with the minimal support equal to 0.2 and minimal growth rate 2 because these were the best parameters for each classifier. The lazy classifier won in 10 cases out of 15. Given 200% level of privacy, it lost only for one set, *Australian*, but the difference was lower than 1%. Thus, the lazy EP classifier yields better results than the eager EP classifier for the additive perturbation with a uniform distribution, especially for high level of privacy.

The results of classification for the retention replacement perturbation with the uniform distribution and $p \in \{0.5, 0.3, 0.15\}$ are shown in Table 6.8. As in the previous comparison, for the lazy approach the JEP classifier with the minimal support equal to 0.05 was used and for the eager EP classifier the following parameters were applied, the minimal support threshold equal to 0.2 and the minimal growth rate equal to 2. Continuous attributes were discretised into 5 bins with equal number of samples[3].

Comparing the results for the retention replacement perturbation, the eager EP classifier (accuracy of this classifier denoted in Table 6.8 as e Acc. and summarised time of training and classification as e t[s]) obtainded the best accuracy for all sets except for *Australian*. However, the eager classifier yielded better accuracy for the *Australian* set than the decision tree for $p \in \{0.3, 0.15\}$, that is, for high level of privacy. The best results for *Australian* set were obtained by the lazy classifier, which yielded worse results than the decision tree for *Diabetes* only for $p = 0.5$ and *Breast*.

To conclude, the accuracy obtained by the lazy and eager EP classifiers depended on the type of the perturbation method, However, both EP classifiers yielded better results than the decision tree for the additive and the retention replacement perturbations.

Time of Classification

To compare time of classification, we used the same experiments as in the previous comparison. Time of classification for the lazy approach depends on the p_{thr}, namely, the lower p_{thr}, the higher time of classification, because lower p_{thr} means more 1's in a reduced data set and more frequent sets.

The lazy classifier was slower than the eager classifier, but not more than an order of magnitude. Only for *Wine* the lazy algorithm was faster. As we focused on the idea of EP in privacy preserving classification, both algorithms were not optimised for time complexity.

[3] The results for discretisation into 10 bins can be found in Appendix A.2.

6.4. Conclusions and Future Work

We presented the new approach to privacy preserving classification for centralised data distorted with randomisation-based methods. It is based on Emerging Patterns and yields better results than the decision tree based on the SPRINT algorithm (for details please refer to Appendix B), especially for high privacy.

We presented both the eager and lazy approach to classifcation with the usage of Emerging Patterns. The eager classifier, ePPCwEP, discovers Emerging Patterns once and based on these patterns chooses a final category for each test sample. The lazy instance-based classifier, lPPCwEP, which is a good solution when a training data set changes often, waits until a test sample comes. Then it mines Emerging Patterns in the context of this sample and chooses a final category, that is, it discovers EPs for each test sample separately.

For the eager approach, we proposed also how to transform continuous and nominal attributes to be used in this approach, hence we are able to use both types of attributes simultaneously with the eager learner. The lazy approach does not need a transformation of these types of attributes.

For the additive perturbation, the new lazy approach gives, in general, better results than the eager EP classifier (especially for high levels of privacy). For the retention replacement the eager EP classifier yeilds better results than the lazy EP classifier. Both algorithms outperformed the decision tree classifier in terms of accuracy measures of classification for the additive and retention replacement perturbations.

As we focused on the idea of Emerging Patterns in privacy preserving classification, the presented ePPCwEP and lPPCwEP classifiers based on EPs are slower than the decision tree despite the MMASK optimisation used for estimating itemset supports in the eager approach. Moreover, the presented lPPCwEP classifier based on EPs and lazy approach to classification (Emerging Patterns are mined for each test sample) is slower than an eager ePPCwEP classifier.

In the future, we plan to focus on the efficiency of this solution. We would like to discover maximal frequent sets instead of frequent sets and operate on borders to improve efficiency of the presented solution, what may be quite hard for the eager learner, because estimating a support of an itemset with maximal number of items at the beginning of the process would be really time consuming. However, this improvement is straightforward for the lazy learner. We also would like to increase the accuracy of results.

We also plan to propose an approach enabling classification of a distorted test set. For the

eager learner, we would like to estimate the support for combinations of nominal attributes without their transformation to binary attributes.

7. Privacy Preserving Classification with Meta-learning

In the case of privacy preserving classification with centralised data distorted by means of a randomisation-based method we may use at least two algorithms for a reconstruction of a probability distribution for continuous attributes: AS [7] and EM [2]. For nominal attributes we also have at least two possible algorithms EM/AS [14] and EQ [15] at our disposal. Hence, we have at least four combinations for sets containing continuous and nominal attributes simultaneously. When we use a decision tree as a classifier in privacy preserving, there are four reconstruction types offered in literature, namely, *Local*, *By class*, *Global* and *Local All* [7] [14].

Combining the currently available in literature algorithms for the reconstruction of a probability distribution and the reconstruction types gives at least 16 possibilities. There is no one combination of algorithms which performs best and we can only choose the best combination for a specific case, that is, for a given data set and parameters of a distortion procedure.

Not only cannot we point out the best combination of algorithms, but it is hard to choose the best reconstruction type. We may say that there are two best reconstruction types: *Local* and *By class* [7] [14], but we still cannot choose the best reconstruction type. The experiments conducted in [7] showed that these two reconstruction types are the best while building decision trees over distorted data containing only continuous attributes. [14] confirmed this statement for continuous and nominal attributes used simultaneously. Both papers did not point out the best reconstruction type because for some data sets *Local* gives better results, for others *By class*.

Taking into account the high number of combinations of algorithms and reconstruction types, we propose to use meta-learning [16] to eliminate these drawbacks. In privacy preserving data mining we may use meta-learning (without hierarchical structures) in two scenarios. The first scenario is to apply *bagging* or *boosting* for a chosen combination of algorithms and a reconstruction type. In the second scenario, *bagging* or *boosting* methods are applied separately to each different combination of algorithms and reconstruction types. Thus, different base classifiers are used, contrary to the first approach where only one base classifier is considered. In both scenarios we use all classifiers together and calculate a final class by *voting*.

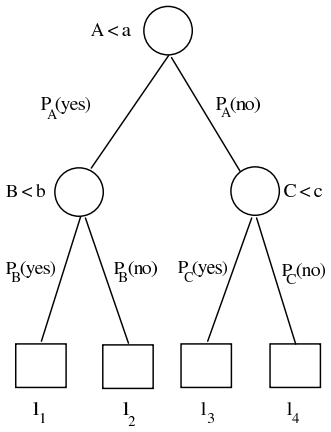

Figure 7.1. The decision tree and the probabilities for tests.

7.1. Boosting for Distorted Data

According to [7, 2, 14, 15] it is assumed that a classifier is trained over distorted data, but test over undistorted data. *Boosting* needs to classify train (distorted) data to compute probabilities P_{il} and α_i fractions (see Section 2.4.2). Classifying distorted data in the same manner as we classify undistorted data leads to inaccuracy. In this section, we propose how to classify train (distorted) data to be able to use *boosting* for distorted data.

In a standard test node in a (binary) decision tree, a given value of an attribute is checked whether it meets a test condition or not. Let us assume that the left branch is chosen when it meets a test and the right branch when it does not.

Having a distorted value x_A of an attribute A, we may calculate the probability $P_A(yes)$ that a given value x_A meets a test condition and the probability $P_A(no)$ that it does not meet a test condition, $P_A(no) = 1 - P_A(yes)$.

For a sample S that we want to classify we calculate the probability Pl_i for i-th leaf that the sample S could fall into this leaf. This probability is equal to the multiplication of probabilities $P_X(yes)$ when we choose a left branch and $P_X(no)$ when we choose a right branch for each test in the path which leads to i-th leaf.

Let us assume there is a decision tree with 3 tests, that is, internal nodes (see Figure 7.1): the first test on an attribute A in the root, second on an attribute B on the left, and third on an attribute C on the right. The probability of the leaf l_1 is $Pl_1 = P_A(yes) \cdot P_B(yes)$.

Having estimated the probabilities for l leaves, we can calculate the probability that the sample S belongs to the category C_j.

$$P_S(C_j) = \sum_{i \in \{a | LeafCategory_a = C_j\}} Pl_i$$

We choose the category with the highest probability and assign it to the sample S.

To calculate $P_S(C_j)$, we need to estimate $P_X(yes)$ and $P_X(no)$ for each test. We propose the solution for nominal and continuous attributes in the subsequent sections.

7.1.1. Calculating Probabilities for Nominal Tests

For nominal attributes we calculate the probability $P_X(yes)$, the probability $P_X(no)$ can be calculated as $P_X(no) = 1 - P_X(yes)$, in the following way:

Let us assume that X is an original nominal attribute and Z is a modified attribute. For an internal node m we know modified values (all train samples which go into this node) of a nominal attribute, so we can determine a probability distribution (i.e., $P(Z = v_i),\ i = 1, ..., k$) of the modified attribute Z. Having modified values of the attribute Z, we can perform the reconstruction using either the EM/AS or EQ algorithm and obtain the reconstructed probability distribution ($P(X = v_i),\ i = 1, ..., k$).

Let us assume that the distorted value of the attribute Z is equal to v_q ($Z = v_q$). We can calculate probabilities that $P(X = v_i | Z = v_q),\ i = 1, ..., k$. Using Bayes' Theorem, we obtain:

$$P(X = v_i | Z = v_q) = \frac{P(X = v_i) \cdot P(Z = v_q | X = v_i)}{P(Z = v_q)}.$$

$P(Z = v_q | X = v_i)$ is the probability that the value v_i of the attribute X will be changed to the value v_q and that probability is known because parameters of the distorting method are given. In order to calculate $P_X(yes)$, we sum probabilities $P(X = v_i | Z = v_q)$ for all values v_i which meet a test condition.

$$P_X(yes) = \sum_{i \in \{a | v_a \in test\}} P(X = v_i | Z = v_q)$$

7.1.2. Calculating Probabilities for Continuous Tests

For continuous attributes and the additive perturbation we calculate the probability $P_X(yes)$ in the following way:

Let X be an original attribute. Y is used to modify X by means of the additive perturbation and obtain an attribute Z. Let Z be equal to z. Let us assume that a continuous test is met if the value of the attribute X is less than or equal to t. Then, we may write:

$$P_X(yes) = P(X \leq t | Z = z, X + Y = Z) = F(t | Z = z, X + Y = Z). \tag{7.1}$$

$$P_X(yes) = \int_{-\infty}^{t} f_X(r | Z = z, X + Y = Z) dr. \tag{7.2}$$

Using Bayes' Theorem, we obtain:

$$P_X(yes) = \int_{-\infty}^{t} \frac{f_Z(Z = z, X + Y = Z | X = r) f_X(r)}{f_Z(z)} dr. \tag{7.3}$$

Since Y is independent of X and the denominator is independent of the integral, then:

$$P_X(yes) = \int_{-\infty}^{t} \frac{f_Y(z - r) f_X(r)}{f_Z(z)} dr. \tag{7.4}$$

f_Y is known, hence we can compute $P_X(yes)$ and $P_X(no)$.

For the retention replacement perturbation we calculate $P_X(yes)$ probability in the following way:

Let X be an original attribute and Z be a modified attribute obtained from X by means of the retention replacement perturbation. Let Z be equal to z. Let us assume that a continuous test is met if the value of the attribute X is less than or equal to t. Then, we may write:

$$P_X(yes) = P(X \leq t | Z = z). \tag{7.5}$$

Let $\mathbf{1}(condition)$ be an indicator function which takes 1 when $condition$ is met and 0 otherwise.

$$P_X(yes) = p\, \mathbf{1}(z < t) + (1 - p) \int_{d_{min}}^{t} g(r | X) dr, \tag{7.6}$$

where $g()$ is a density function of a distorting distribution used in the retention replacement perturbation.

Since $g()$ is independent of X, we obtain:

$$P_X(yes) = p\, \mathbf{1}(z < t) + (1 - p) \int_{d_{min}}^{t} g(r) dr. \tag{7.7}$$

Assuming a uniform distribution as the distorting distribution for the retention replacement perturbation, we can write:

$$P_X(yes) = p\,\mathbf{1}(z < t) + (1-p)\frac{t - d_{min}}{d_{max} - d_{min}}, \tag{7.8}$$

where d_{max} and d_{min} are maximal and minimal value of a domain of the attribute X.

7.2. Bagging and Boosting for Chosen Combination of Algorithms and Reconstruction Type

Having chosen a combination of algorithms for a reconstruction of a probability distribution, one algorithm for continuous attributes and second algorithm for nominal attributes, and a reconstruction type to be used in a decision tree, we may use *bagging* or *boosting*. A final class is determined according to a simple or weighted *voting* method.

We may also join votes from *bagging* and *boosting* at one time and calculate the final class. In our approach votes are combined at the same level, that is, hierarchical classifiers are not used. In this case, we choose a number of classifiers for each meta-learning method separately. For *bagging* weights are equal to 1 and for *boosting* we use α_i fraction as weights.

For example, let us assume that a decision class is a binary (0-1) attribute, the number of classifiers for *bagging* is 3 and for *boosting* is 4. For a sample S we sum weights for all classifiers which answered 0 and for all classifiers which answered 1.

$$W_0 = \sum_{j=1..7, cl_j(S)=0} w_j, \;\; W_1 = \sum_{j=1..7, cl_j(S)=1} w_j$$

Then we choose a class with the highest sum (cumulative weight).

7.3. Combining Different Algorithms and Reconstruction Types with Usage of Bagging and Boosting

There are three possible cases of using different algorithms and reconstruction types with meta-learning. We may use different combinations of algorithms or different reconstruction types.

For each combination of reconstruction algorithms and a chosen reconstruction type we separately use either *bagging* or *boosting*, like in Section 7.2. Then we determine a final class

using all created classifiers by (*weighted*) *voting* (all classifiers are on the same level and we sum all weights).

For different reconstruction types and a chosen combination of algorithms we process in the same way as for the case with different algorithms, that is, for each reconstruction types and a chosen combination of algorithms we use either *bagging* or *boosting* and then we determine a final class using all created classifiers by *voting*.

We may also combine these situations, that is, we use *bagging* or *boosting* for each possible combination of algorithms and reconstruction types. The method of calculating weights remains the same.

In all cases we may include not all possible combinations to achieve better accuracy of classification, e.g., only *Local* and *By class* for different combinations of reconstruction types.

7.4. Experimental Evaluation

This section presents the results of the experiments conducted with the usage of meta-learning in privacy preserving classification.

All sets used in tests can be downloaded from UCI Machine Learning Repository [20]. We used the following sets: *Australian*, *Credit-g*, *Diabetes*, *Segment*, chosen under the following conditions: two sets should contain only continuous attributes (*Diabetes*, *Segment*), two other sets both continuous and nominal attributes (*Australian*, *Credit-g*), one of the sets should have a class attribute with multiple values (*Segment*).

In the experiments the accuracy, sensitivity, specificity, precision and F-measure were used. We used also the definition of privacy based on the *differential entropy* [2] (see also Section 3.5.4). To achieve more reliable results, we used 10-fold *cross-validation* [36] and calculated the average of 50 multiple runs. In all experiments we distorted all attributes except for class/target attribute. All continuous attributes were distorted by means of the additive perturbation with a uniform distribution, unless explicitly stated that either a normal distribution or the retention replacement perturbation with a uniform distribution was used.

We used the SPRINT [103] decision tree modified to incorporate privacy according to Sections 3.7.1 and 3.7.2 (for details refer to Appendix B).

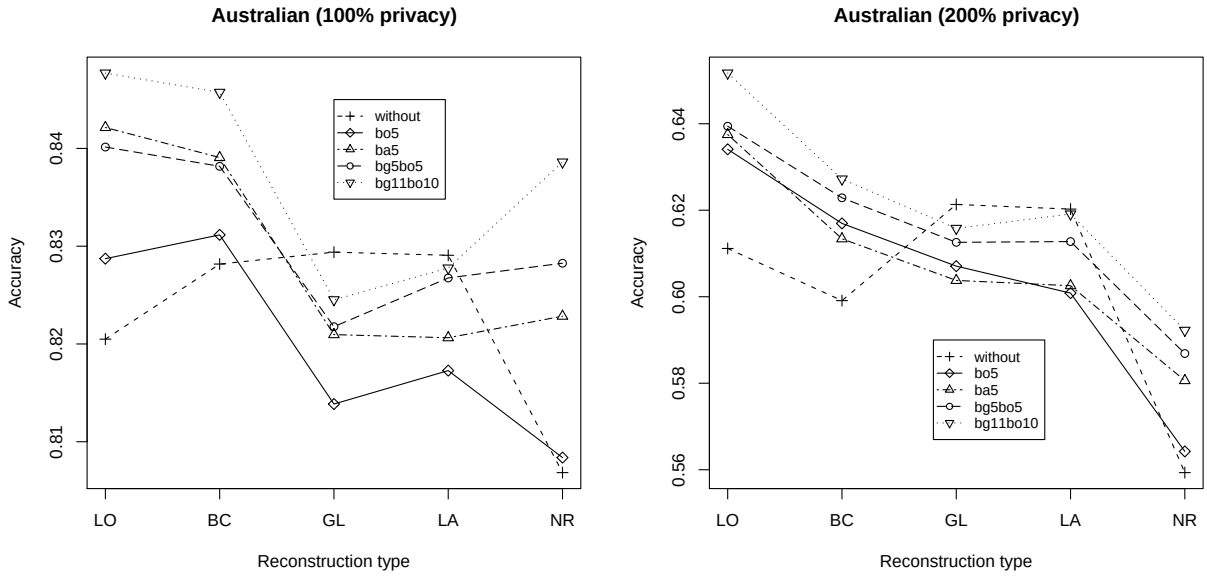

Figure 7.2. The accuracy of classification with the usage of meta-learning (bagging and boosting methods) for the set Australian with 100% and 200% privacy level for the chosen combination of algorithms - AS.EA.

7.4.1. Experiments with Chosen Combination of Algorithms and Reconstruction Type with Usage of Bagging and Boosting

Figure 7.2 shows the accuracy of classification for *Australian* set. We used the following combination of algorithms: AS.EA, i.e., AS for continuous attributes and EM/AS (called EA) for nominal attributes and conducted experiments for all possible types of reconstruction: LO, *Local*; BC, *By class*; GL, *Global*; LA, *Local all*. NR means that we did not use any reconstruction.

We used *bagging* and *boosting* separately and combined, as described in Section 7.2. We also experimented with the number of classifiers for a given meta-learning method.

Figure 7.2 presents the results for the separate usage of *bagging* - $bg5$ (5 classifiers were used), *boosting* $bo5$ (with 5 classifiers) and the combination of meta-learning methods, e.g., $bg5bo4$ - *bagging* used 5 classifiers and *boosting* 4 decision trees (the number after short name of a meta-learning method denotes how many classifiers were used for a particular meta-learning method). *Without* means that no meta-learning method was used, i.e., a single classifier was built.

For *Local* and *By class* reconstruction types and both presented levels of privacy (100%, 200%) we obtained in every case higher accuracy. For 100% level of privacy accuracy was about 84-85%. 200% level of privacy reduces accuracy to the level of 62-64%.

Table 7.1. The sensitivity, specificity, precision and F-measure with the usage of meta-learning (bagging and boosting methods) for the set Australian with 100% privacy level for the chosen combination of algorithms - AS.EA.

Measure, method	LO	BC	GL	LA	NR
Sensitivity without	0.7958	0.7958	0.8021	0.7985	0.7693
bg5bo5	0.8056	0.7836	0.7664	0.7708	0.7720
Specificity without	0.8403	0.8541	0.8508	0.8531	0.8369
bg5bo5	0.8679	0.8818	0.8654	0.8708	0.8730
Precision without	0.8031	0.8191	0.8155	0.8171	0.7982
bg5bo5	0.8343	0.8463	0.8229	0.8303	0.8321
F without	0.7956	0.8031	0.8053	0.8041	0.7774
bg5bo5	0.8162	0.8096	0.7906	0.7961	0.7972

Table 7.2. The sensitivity, specificity, precision and F-measure with the usage of meta-learning (bagging and boosting methods) for the set Australian with 200% privacy level for the chosen combination of algorithms - AS.EA.

Measure, method	LO	BC	GL	LA	NR
Sensitivity without	0.4813	0.4068	0.4404	0.4427	0.4251
bg5bo5	0.3964	0.3357	0.4301	0.4050	0.3309
Specificity without	0.7157	0.7527	0.7669	0.7632	0.6688
bg5bo5	0.8339	0.8522	0.7598	0.7801	0.7928
Precision without	0.5839	0.5801	0.5945	0.5956	0.5180
bg5bo5	0.6665	0.6581	0.5838	0.5905	0.5727
F without	0.5168	0.4669	0.4936	0.4937	0.4465
bg5bo5	0.4862	0.4343	0.4872	0.4696	0.3998

For 100% level of privacy *bagging* with 5 classifiers achieved better results than *bagging* and *boosting* together both with 5 classifiers. However, for 200% level of privacy *bagging* and *boosting* together with 5 classifiers per each method performed better.

For *Global* and *Local all* meta-learning decreased the accuracy of classification. The reason may be that in these two reconstruction types we do not divide samples into classes during the reconstruction and changes made for each training set have low influence on a decision of classifiers.

For 100% level of privacy without reconstruction meta-learning increased accuracy. We may say that it was as high as for *Global* and *Local all*. For 200% level of privacy and without the reconstruction meta-learning still yielded better results, but the overall accuracy was the lowest compared to the accuracy of all reconstruction types.

Tables 7.1 and 7.2 show the sensitivity, specificity, precision and F-measure for the set *Australian* in this experiment. For 100% level of privacy with *Local* and *By class* reconstruction types only the sensitivity for *By class* has lower value for meta-learning with the simultaneous

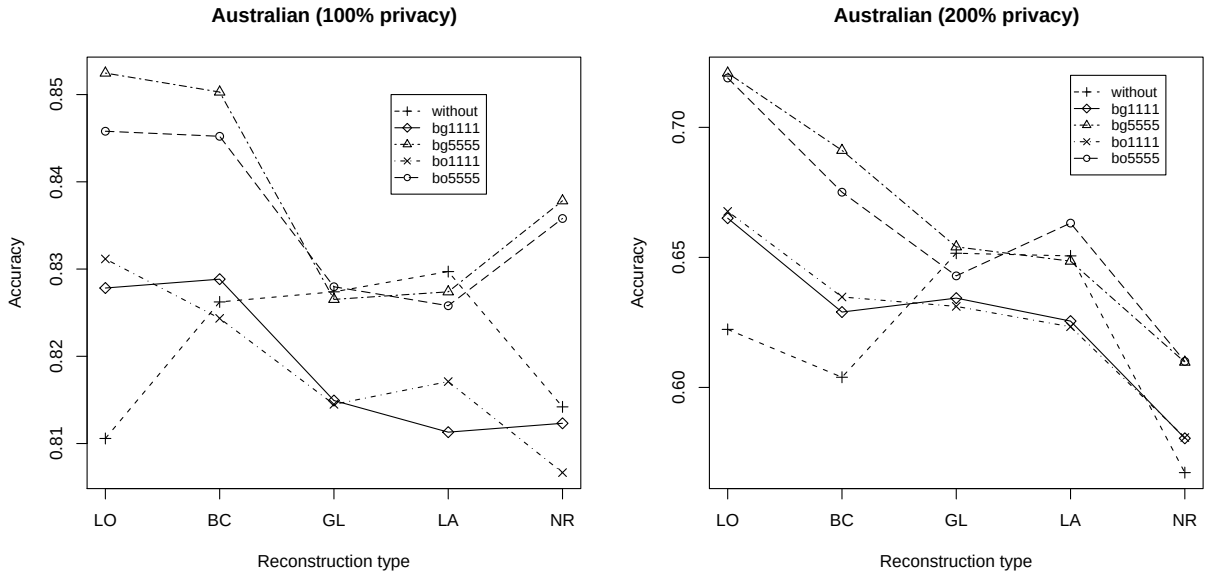

Figure 7.3. The accuracy of classification with the usage of meta-learning (bagging and boosting methods) for the set Australian with 100% and 200% privacy level for the different combinations of algorithms and the chosen reconstruction type.

usage of *bagging* and *boosting* with 5 classifiers for each method, compared to the case without meta-learning. For 200% level of privacy with *Local* and *By class* we obtained lower values for the sensitivity and F-measure (the precision increased). Thus, meta-learning caused slightly lower proportion of actual positives which were correctly identified as such. For 200% level of privacy with *Global* and *Local all* reconstruction types meta-learning yielded better results only in two cases: the sensitivity for *Global* and the specificity for *Local all*. For 100% level of privacy all measures were between about 77% and 86%, for 200% the sensitivity decreased to the level of 33%.

To sum up, we can say that in general the higher number of classifiers is used, the better accuracy meta-learning yields. The simultaneous usage of two meta-learning methods with high number of classifiers yeields results with higher accuracy than only one meta-learning method.

7.4.2. Accuracy of Classification for Different Combinations of Algorithms with Usage of Bagging and Boosting

We performed the experiments with the usage of all combinations of algorithms: AS.EA, EM.EA, AS.EQ, and EM.EQ (Figure 7.3). We used separately *bagging* and *boosting* with 1 and 5 classifiers per each combination of reconstruction algorithms and a chosen reconstruction type ($bg1111$ denotes *bagging* with 1 classifier per each combination of reconstruction algorithms, $bo5555$ means *boosting* with 5 classifiers per each combination of algorithms, etc.).

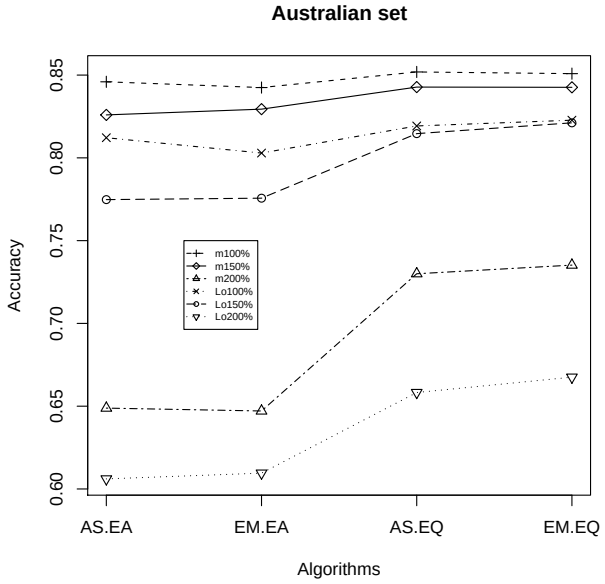

Figure 7.4. The accuracy of classification with the usage of meta-learning (simultaneously bagging and boosting with 5 classifiers per each method) for the set Australian with 100%, 150%, and 200% privacy level for only Local and By class reconstruction types compared to Local reconstruction type.

The obtained results were similar to those from the experiment presented in Section 7.4.1. Meta-learning yielded better results for *Local*, *By class* and no reconstruction, but almost no improvement for *Global* and *Local all*. For 5 classifiers per combination of algorithms (for *Local* and *By class*) we obtained high accuracy, about 85% for 100% level of privacy and about 72% for 200% level of privacy (for 200% level of privacy even higher than for the experiment in Section 7.4.1). For *bagging* and *boosting* with 1 classifier per each combination of algorithms we observed lower accuracy, but still higher than without meta-learning (except for one case).

To conclude, by using different combinations of algorithms, we obtained high accuracy. The higher number of classifiers was used, the better results meta-learning yielded. *Global* and *Local all* reconstruction types yielded poor results for meta-learning.

7.4.3. Accuracy of Classification for Different Reconstruction Types with Usage of Meta-learning

For the set *Australian* with the usage of meta-learning for all combinations of reconstruction types we obtained worse results than without meta-learning because we obtained really low accuracy for *Global* and *Local all* reconstruction types for the set *Australian*. [7, 14] and the results of the previous experiments in this chapter confirmed that these two reconstruction types seem to be the worst and for some sets they yield very poor results.

Table 7.3. The accuracy of classification with the usage of meta-learning (simultaneously bagging and boosting with 5 classifiers per each method) for only Local and By class reconstruction types and different combinations of algorithms compared to Local reconstruction type and AS.EA algorithms.

Privacy	Set	Acc.	Sens.	Spec.	Prec.	F
100%	mCredit-g	0.725	0.2426	0.9342	0.6213	0.3349
	LoCredit-g	0.6813	0.3749	0.8134	0.4636	0.4074
	mCredit-g (n)	0.73	0.328	0.9038	0.591	0.4113
	LoCredit-g (n)	0.6716	0.4212	0.7795	0.4455	0.4267
100%	mAustralian	0.8567	0.8633	0.8517	0.8288	0.8425
	LoAustralian	0.8199	0.804	0.8329	0.7995	0.7971
	mAustralian (n)	0.8548	0.8557	0.8541	0.8301	0.8396
	LoAustralian (n)	0.8261	0.8004	0.8462	0.8095	0.8021
100%	mDiabetes	0.7392	0.8687	0.4936	0.7594	0.809
	LoDiabetes	0.6908	0.8216	0.4467	0.7326	0.7718
	mDiabetes (n)	0.7409	0.8856	0.471	0.7541	0.8127
	LoDiabetes (n)	0.7039	0.8399	0.4515	0.7375	0.7827
100%	mSegment	0.8354	0.8354	0.9726	0.8412	0.8339
	LoSegment	0.7974	0.7972	0.9663	0.8022	0.7935
	mSegment (n)	0.811	0.8115	0.9685	0.8239	0.8109
	LoSegment (n)	0.7877	0.7884	0.9646	0.7994	0.7849
200%	mCredit-g	0.6889	0.1396	0.9261	0.4803	0.2301
	LoCredit-g	0.6033	0.321	0.7283	0.3439	0.3149
	mCredit-g (n)	0.6839	0.1363	0.919	0.5091	0.2502
	LoCredit-g (n)	0.6061	0.3372	0.7239	0.3495	0.325
200%	mAustralian	0.6962	0.5011	0.8526	0.7461	0.5858
	LoAustralian	0.6165	0.4814	0.7235	0.5968	0.5225
	mAustralian (n)	0.6822	0.4719	0.85	0.7267	0.5404
	LoAustralian (n)	0.5696	0.4637	0.6541	0.5566	0.4784
200%	mDiabetes	0.7158	0.8412	0.4802	0.7492	0.7901
	LoDiabetes	0.6699	0.7785	0.4666	0.7307	0.7493
	mDiabetes (n)	0.7025	0.9266	0.2859	0.7068	0.7993
	LoDiabetes (n)	0.6762	0.8484	0.3559	0.7095	0.768
200%	mSegment	0.8216	0.8231	0.9703	0.8184	0.8157
	LoSegment	0.7882	0.789	0.9647	0.7921	0.7829
	mSegment (n)	0.7802	0.7823	0.9634	0.7814	0.7712
	LoSegment (n)	0.7272	0.729	0.9546	0.7247	0.708

Table 7.4. The comparison of the meta-learning accuracy gain for undistorted and distorted data (the simultaneous usage of bagging and boosting with 5 classifiers per each method).

Set	Acc.		Priv.				Priv.(n)		
	without	meta	0%	100%	150%	200%	100%	150%	200%
Credit-g	0.7210	0.7508	4.1%	6.4%	12.6%	14.2%	8.7%	9.1%	12.8%
Australian	0.8261	0.8552	3.5%	4.5%	9.8%	12.9%	3.5%	6.6%	19.8%
Diabetes	0.7368	0.7449	1.1%	6.6%	7.7%	6.4%	5.3%	3.7%	3.9%
Segment	0.9355	0.9550	2.1%	4.8%	4.6%	4.2%	3.0%	6.8%	7.3%

To eliminate the undesirable impact of *Global* and *Local all*, we used only *Local* and *By class* reconstruction types. The results of the experiments for the set *Australian* are shown in Figure 7.4. Only for two best reconstruction types meta-learning performed better than a single classifier. For 100% level of privacy the accuracy was about 85%, for 150% slightly lower - 82-83%. For 200% level of privacy we obtained 65% of the accuracy for AS.EA and EM.EA, algorithms AS.EQ and EM.EQ yielded accuracy about 72%.

7.4.4. Accuracy of Classification for Different Combination of Algorithms and Reconstruction Types with Usage of Bagging and Boosting

The last possibility is to combine different algorithms and reconstruction types. According to the results from the previous section we use only *Local* and *By class* types of reconstruction.

The results of the experiments are shown in Table 7.3. The sets used in these experiments were distorted by means of the additive perturbation with either a uniform or normal distribution (*mCredit-g* means that the set *Credit-g* was distorted with a uniform distribution and meta-learning was used, *LoCredit-g* informs that Local reconstruction type and AS.EA algorithms were used, *mCredit-g (n)* means that the set was distorted by means of a normal distribution, etc.). Only for *Credit-g* meta-learning significantly decreased the sensitivity and F-measure. In the remaining cases, meta-learning yielded higher measures (there was only one case with the significantly worse result, the specificity for *Diabetes* set, 200% level of privacy and a nominal distortion distribution).

Table 7.4 shows the accuracy (denoted as *Acc.*) without (denoted as *without*) and with (denoted as *meta*) meta-learning without preserved privacy and the relative gain caused by meta-learning for undistorted data (*Priv. 0%*) and privacy preserved data for a uniform distorting distribution (*Priv. 100%-200%*) and a normal (*Priv.(n) 100%-200%*) distorting distributions. In all cases meta-learning gain for level of privacy 100%, 150%, and 200% was higher than for undistorted data.

Table 7.5 presents the results of the experiment with combining different reconstruction algorithms, *Local* and *By class* reconstruction types where sets were distorted by means of the retention replacement perturbation with a uniform distribution and $p \in \{0.5, 0.3, 0.15\}$ (*mCredit-g* means that meta-learning was used for the set *Credit-g*, *LoCredit-g* informs that Local reconstruction type and AS.EA algorithms were used, etc.). After distortion, continuous attributes were discretised into 5 bins each of which covered equal number of samples[1]. Similarly

[1] The results for discretisation into 10 bins can be found in Appendix A.3.

Table 7.5. The accuracy of classification with the usage of meta-learning (simultaneously bagging and boosting with 5 classifiers per each method) for only Local and By class reconstruction types and different combinations of algorithms compared to Local reconstruction type and AS.EA algorithms and the retention replacement perturbation.

p	Set	Acc.	Sens.	Spec.	Prec.	F	Time [s]
0.5	mAustralian	0.8586	0.8444	0.8698	0.8427	0.8406	7.9231
	LoAustralian	0.8203	0.8012	0.8358	0.8004	0.7975	0.1108
0.3	mAustralian	0.8443	0.8033	0.8766	0.8433	0.8184	8.1545
	LoAustralian	0.7748	0.7430	0.8003	0.7531	0.7436	0.1212
0.15	mAustralian	0.7449	0.5885	0.8676	0.7861	0.6653	10.8198
	LoAustralian	0.6300	0.5275	0.7099	0.6001	0.5539	0.2533
0.5	mCredit-g	0.7232	0.2890	0.9105	0.5793	0.3771	7.3214
	LoCredit-g	0.6688	0.4080	0.7813	0.4443	0.4192	0.1411
0.3	mCredit-g	0.7030	0.1779	0.9295	0.5240	0.2716	7.0203
	LoCredit-g	0.6360	0.3844	0.7441	0.3934	0.3788	0.1688
0.15	mCredit-g	0.6834	0.1354	0.9188	0.4431	0.2601	8.1628
	LoCredit-g	0.5799	0.3783	0.6669	0.3285	0.3403	0.2246
0.5	mDiabetes	0.7302	0.8366	0.5311	0.7665	0.7982	2.1204
	LoDiabetes	0.6837	0.7718	0.5212	0.7488	0.7570	0.0221
0.3	mDiabetes	0.7115	0.8429	0.4654	0.7448	0.7886	2.1381
	LoDiabetes	0.6436	0.7345	0.4729	0.7209	0.7243	0.0286
0.15	mDiabetes	0.6710	0.8641	0.3164	0.7011	0.7708	2.2153
	LoDiabetes	0.5974	0.7058	0.4001	0.6859	0.6909	0.0380
0.5	mSegment	0.8730	0.8728	0.9788	0.8760	0.8695	78.1030
	LoSegment	0.8291	0.8293	0.9715	0.8318	0.8239	1.0958
0.3	mSegment	0.8183	0.8182	0.9697	0.8261	0.8120	85.9497
	LoSegment	0.7382	0.7387	0.9564	0.7418	0.7249	1.3034
0.15	mSegment	0.7252	0.7252	0.9542	0.7310	0.7058	101.6288
	LoSegment	0.5885	0.5888	0.9315	0.5679	0.5667	1.6946

to the previous experiment with the additive perturbation, meta-learning significantly decreased the sensitivity and F-measure only for *Credit-g*. In the remaining cases, meta-learning yielded higher measures (there were only two cases with the significantly worse results, the specificity for *Diabetes* set and $p \in \{0.3, 0.15\}$).

In the presented experiments, application of meta-learning increased accuracy once again proving its usefulness in privacy preserving data mining.

7.4.5. Time of Training Classifiers with Meta-learning

Unfortunately, meta-learning increases time of training because several classifiers, e.g., decision trees, need to be built, which takes time.

Considering time of training for different reconstruction types for a decision tree, *Local* is the most expensive because it reconstructs a probability distribution for each class in every node of a decision tree. *By class* takes only slightly more time (for high number of classifiers) than the case without the reconstruction, because it performs the reconstruction for each class, but only in a root of a tree. For about 20 classifiers there is a difference in time of training of one order of magnitude compared to the case without meta-learning. Comparing also the results presented in Table 7.5, where 80 classifiers were used, the difference in summarised time of training and classification is between one and two orders of magnitude compared to the case with one classifier. Meta-learning increases time of training, but the time of classification is still small and almost the same.

Meta-learning is a perfect approach to use distributed computations and train classifiers on different machines. This would reduce time needed to train classifiers.

7.5. Conclusions and Future Work

In privacy preserving data mining for classification meta-learning can be used to achieve the higher accuracy and combine information from different algorithms.

The conducted experiments showed that it is better to combine only *Local* and *By class* reconstruction types than all the reconstruction types because *Global* and *Local All* may yield poor results (probably due to the reconstruction performed for data not divided into classes).

Meta-learning gives higher gain in accuracy for data with preserved privacy than for undistorted data because combines additional information from different probability reconstruction algorithms and types of reconstruction compared to the case without privacy, where probability reconstruction algorithms are not used. Moreover, meta-learning improves results when

”unstable” learning algorithms are used and classifiers with different probability reconstruction algorithms and reconstruction types may be seen as such because they yield significantly different results for different reconstruction algorithms and reconstruction types.

Unfortunately, meta-learning increases time of training classifiers. Time of classification is still significantly smaller than time of training classifiers. Furthermore, meta-learning makes harder an interpretation of a created classifier. One has to look at all decision trees to know rules of classification.

In the future, we plan to investigate the possibility of extension of our results to the usage of various classification algorithms as meta-learners (not only simple or weighted voting). We will check results for the case where every single implementation of *bagging* and *boosting* yields separately its own answer to a classifier of the higher level (contrary to the case presented in this thesis). It is also possible to pass to a classifier of the higher level not only answers of classifiers, but the training set or its subset. We also plan to use hierarchical classifiers (with 3 and more levels). We would like to try to group classifiers with, e.g., different combinations of algorithms and the same reconstruction type, and then train a classifier on the highest level on their outputs. We would like to use the presented approach to classify a distorted test set. To reduce time of training classifiers in meta-learning, we plan to use distributed computations.

8. Conclusions and Future Work

In this work, we have focused on privacy preserving association rules mining and classification. We have considered a centralised database distorted by means of a randomisation-based method, which well corresponds to Internet surveys scenario.

We have proposed the optimisation MMASK, which eliminates exponential complexity in estimating an original support of an itemset with respect to its cardinality and makes privacy preserving discovery of frequent itemsets and, by this, association rules feasible. Moreover, it enables each value of each binary attribute to have different distortion parameters with eliminated exponential complexity. We showed experimentally that the proposed optimisation increased the accuracy of the results for high level of privacy.

We have also presented how to use randomisation for both ordinal and integer attributes to modify their values according to the order of possible values of these attributes to both maintain their original domain and obtain similar distribution of values of an attribute after distortion. The presented method gives comparable results to the method which treats attributes as if they have no order, but allows a miner to use the same domain for integer attributes and distort values of integer and ordinal attributes according to the order of possible values of these attributes. Having the same domain for an original and modified attribute, a miner can use the same metadata for both original and modified values. This advantage, as well as similar distributions of values of attributes before and after distortion, give a higher protection against data disclosure when a distortion procedure and its parameters are only revealed to authorised users to keep data more private, that is, in the case of confident data disclosure, a potential attacker cannot figure out whether it is original or modified data based on domains and distributions of attributes.

In this work, the privacy preserved ePPCwEP and lPPCwEP classifiers based on Emerging Patterns have been proposed. We have used both the eager approach to classification and shown how to classify with Emerging Patterns, and the lazy approach, which is a good solution when a pre-classified training set grows often. The proposed privacy preserving ePPCwEP and lPPCwEP classifiers are modifications of eager CAEP and lazy DeEPs classifiers. Both new classifiers have outperformed the described in literature and used in this thesis decision tree classifier based on the SPRINT algorithm in terms of accuracy measure. Moreover, the privacy

preserved lPPCwEP classifier based on the lazy approach and Emerging Patterns has achieved, in general, better results than the eager ePPCwEP classifier using Emerging Patterns for the additive perturbation. This tendency is better seen for high level of privacy. For the retention replacement, the eager ePPCwEP classifier has yielded better results than the lazy lPPCwEP classifier. Both algorithms outperformed the privacy preserving decision tree classifier in terms of accuracy measures of classification for the additive and retention replacement perturbations.

We have applied meta-learning to privacy preserving classification. Not only have we used bagging and boosting, but we have combined different probability distribution of values of attributes reconstruction algorithms and reconstruction types for a decision tree. The results have shown that meta-learning can be used to achieve the higher accuracy and combine information from different algorithms. Meta-learning gives higher accuracy gain for privacy preserving scenario than for undistorted data. Unfortunately, meta-learning is charecterised by higher time complexity of learning than a single classifier. Training time can be reduced using parallel computations, but this cannot be applied in a straightforward manner in cases when boosting is used. Another drawback of meta-learning is harder interpretation of the results, e.g., one needs to check all decision trees to find out why a classifier has pointed out a final category.

We have shown that reducing the accuracy loss in privacy preserving association rules and classification compared to the previous solutions presented in literature is possible. We have also presented the optimisation for association rules mining which makes this process viable in practice.

In future work, we plan to use hierarchical classifiers and various classifiers as meta-learners. Moreover, we plan to test different base classifiers. Not only will we pass the answer of the classifier to the higher level classifier, but also we will use the distorted training set or its subset. In order to reduce time of training of hierarchical classifiers, we plan to use distributed computations. We would like to focus on efficiency of classifiers based on Emerging Patterns. We plan to extend the proposed optimisation MMASK to quantitative and generalised association rules. Furthermore, we plan to investigate the possibility of classification of a distorted set of objects. We would like to focus also on preserving privacy for a target attribute by applying a randomisation-based method to a target attribute, that is, distorting it, in a similar way we did for non-target attributes.

Bibliography

[1] Charu C. Aggarwal and Philip S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer Publishing Company, Incorporated, 2008.

[2] Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 247–255, 2001.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *SIGMOD Conference*, pages 207–216. ACM Press, 1993.

[4] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-preserving encryption for numeric data. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *SIGMOD Conference*, pages 563–574. ACM, 2004.

[5] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[6] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *SIGMOD Conference*, pages 439–450. ACM, 2000.

[8] Rakesh Agrawal, Ramakrishnan Srikant, and Dilys Thomas. Privacy preserving olap. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 251–262, New York, NY, USA, 2005. ACM.

[9] Shipra Agrawal, Vijay Krishnan, and Jayant R. Haritsa. On addressing efficiency concerns in privacy preserving data mining. *CoRR*, cs.DB/0310038, 2003.

[10] Shipra Agrawal, Vijay Krishnan, and Jayant R. Haritsa. On addressing efficiency concerns in privacy-preserving mining. In Yoon-Joon Lee, Jianzhong Li, Kyu-Young Whang, and Doheon Lee, editors, *DASFAA*, volume 2973 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2004.

[11] Leila N. Alachaher and Sylvie Guillaume. Variables interaction for mining negative and positive quantitative association rules. In *ICTAI*, pages 82–85. IEEE Computer Society, 2006.

[12] Piotr Andruszkiewicz. Privacy preserving data mining on the example of classification (in Polish). Master's thesis, Warsaw University of Technology, 2005.

[13] Piotr Andruszkiewicz. Optimization for mask scheme in privacy preserving data mining for association rules. In Marzena Kryszkiewicz, James F. Peters, Henryk Rybiński, and Andrzej Skowron, editors, *RSEISP*, volume 4585 of *Lecture Notes in Computer Science*, pages 465–474. Springer, 2007.

[14] Piotr Andruszkiewicz. Privacy preserving classification for continuous and nominal attributes. In *Proceedings of the 16th International Conference on Intelligent Information Systems*, 2008.

[15] Piotr Andruszkiewicz. Probability distribution reconstruction for nominal attributes in privacy preserving classification. In *ICHIT '08: Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology*, pages 494–500, Washington, DC, USA, 2008. IEEE Computer Society.

[16] Piotr Andruszkiewicz. Classification with meta-learning in privacy preserving data mining. In Lei Chen, Chengfei Liu, Qing Liu, and Ke Deng, editors, *DASFAA Workshops*, volume 5667 of *Lecture Notes in Computer Science*, pages 261–275. Springer, 2009.

[17] Piotr Andruszkiewicz. Privacy preserving classification for ordered attributes. In James F. Peters Urszula Stańczyk Krzysztof A. Cyran, Stanisław Kozielski and Alicja Wakulicz-Deja, editors, *Man-Machine Interactions*, volume 59/2009 of *Advances in Soft Computing*, pages 353–360. Springer, 2009.

[18] Piotr Andruszkiewicz. Privacy preserving classification with emerging patterns. In Yücel Saygin, Jeffrey Xu Yu, Hillol Kargupta, Wei Wang, Sanjay Ranka, Philip S. Yu, and Xindong Wu, editors, *ICDM Workshops*, pages 100–105. IEEE Computer Society, 2009.

[19] Piotr Andruszkiewicz. Lazy approach to privacy preserving classification with emerging patterns. In Dominik Ryżko, Piotr Gawrysiak, Henryk Rybiński, and Marzena Kryszkiewicz, editors, *Emerging Intelligent Technologies in Industry*, volume 369 of

Studies in Computational Intelligence. Springer, 2011.

[20] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007.

[21] Mikhail J. Atallah, Elisa Bertino, Ahmed K. Elmagarmid, M. Ibrahim, and Vassilios S. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the IEEE Knowledge and Data Engineering Workshop (1999)*, pages 45–52, 1999.

[22] Yonatan Aumann and Yehuda Lindell. A statistical theory for quantitative association rules. In *KDD*, pages 261–270, 1999.

[23] Roberto J. Bayardo Jr., Bart Goethals, and Mohammed J. Zaki, editors. *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[24] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.

[25] Max Bramer. *Principles of Data Mining*. Springer, 2007.

[26] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[27] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[28] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In Peckham [86], pages 255–264.

[29] Philip K. Chan and Salvatore J. Stolfo. Experiments on multi-strategy learning by meta-learning. In Bharat K. Bhargava, Timothy W. Finin, and Yelena Yesha, editors, *CIKM*, pages 314–323. ACM, 1993.

[30] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract). In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, page 462. Springer, 1987.

[31] Keke Chen and Ling Liu. Privacy preserving data classification with rotation perturbation. In *ICDM*, pages 589–592. IEEE Computer Society, 2005.

[32] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.

[33] Chris Clifton and Don Marks. Security and privacy implications of data mining. In *Workshop on Data Mining and Knowledge Discovery*, number 96-08, pages 15–19, Montreal, Canada, 2 1996. University of British Columbia Department of Computer Science.

[34] Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. In Ira S. Moskowitz, editor, *Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2001.

[35] Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors. *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.* Morgan Kaufmann, 1995.

[36] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, (10):1895–1924, 1998.

[37] Josep Domingo-ferrer and Josep M. Mateo-sanz. On resampling for statistical confidentiality in contingency tables. In *Computers and Mathematics with Applications*, pages 13–32, 1999.

[38] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: discovering trends and differences. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52, New York, NY, USA, 1999. ACM.

[39] Guozhu Dong, Jinyan Li, and Xiuzhen Zhang. Discovering jumping emerging patterns and experiments on real datasets. In *Proceedings of 9th International Database Conference on Heterogeneous and Internet Databases*, pages 155–168, July 1999.

[40] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by Aggregating Emerging Patterns. Technical report, March 1999.

[41] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by Aggregating Emerging Patterns. In *DS '99: Proceedings of the Second International Conference on Discovery Science*, pages 30–42. Springer-Verlag, London, UK, 1999.

[42] Jim Dowd, Shouhuai Xu, and Weining Zhang. Privacy-preserving decision tree mining based on random substitutions. In Günter Müller, editor, *ETRICS*, volume 3995 of *Lecture Notes in Computer Science*, pages 145–159. Springer, 2006.

[43] Jim Dowd, Shouhuai Xu, and Weining Zhang. Privacy-preserving decision tree mining using a random replacement perturbation. Technical report, The University of Texas, 2006.

[44] Wenliang Du, Yunghsiang S. Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *SDM*, 2004.

[45] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In Chris

Clifton and Vladimir Estivill-Castro, editors, *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14 of *Conferences in Research and Practice in Information Technology*, pages 1–8, Maebashi City, Japan, 2002. ACS.

[46] Wenliang Du and Zhijun Zhan. Using randomized response techniques for privacy-preserving data mining. In *KDD*, pages 505–510, 2003.

[47] Saso Dzeroski, Pance Panov, and Bernard Zenko. Machine learning, ensemble methods in. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 5317–5325. Springer, 2009.

[48] Scott R. Eliason. *Maximum likelihood estimation: Logic and practice*. Number 07-096 in Quantitative Applications in the Social Sciences. Sage University Papers, Newbury Park, 1993.

[49] Fatih Emekçi, Ozgur D. Sahin, Divyakant Agrawal, and Amr El Abbadi. Privacy preserving decision tree learning over multiple parties. *Data Knowl. Eng.*, 63(2):348–361, 2007.

[50] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–228, New York, NY, USA, 2002. ACM Press.

[51] Marek Fisz. *Probability Theory and Mathematical Statistics*. John Wiley and Sons, New York, 1963.

[52] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[53] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Mining optimized association rules for numeric attributes. In *PODS*, pages 182–191. ACM Press, 1996.

[54] Alka Gangrade and Ravindra Patel. Building privacy-preserving c4.5 decision tree classifier on multiparties. *International Journal on Computer Science and Engineering*, 1(2):199–205, 2009.

[55] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.

[56] Bart Goethals and Mohammed J. Zaki, editors. *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining*

Implementations, 19 December 2003, Melbourne, Florida, USA, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[57] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[58] Attila Gyenesei. Mining weighted association rules for fuzzy quantitative items. In Zighed et al. [141], pages 416–423.

[59] Shuguo Han and Wee Keong Ng. Multi-party privacy-preserving decision trees for arbitrarily partitioned data. *International Journal of Intelligent Control and Systems*, 12(4):351–358, 2007.

[60] Yunhong Hu, Liang Fang, and Guoping He. Privacy-preserving svm classification on vertically partitioned data without secure multi-party computation. In Haiying Wang, Kay Soon Low, Kexin Wei, and Junqing Sun, editors, *ICNC (1)*, pages 543–546. IEEE Computer Society, 2009.

[61] Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *DMKD*, 2002.

[62] Murat Kantarcioglu and Chris Clifton. Privately computing a distributed k-nn classifier. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *PKDD*, volume 3202 of *Lecture Notes in Computer Science*, pages 279–290. Springer, 2004.

[63] Murat Kantarcoglu and Jaideep Vaidya. Privacy preserving naive Bayes classifier for horizontally partitioned data. In *IEEE ICDM Workshop on Privacy Preserving Data Mining*, pages 3–9, Melbourne, FL, November 2003.

[64] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM*, pages 99–106. IEEE Computer Society, 2003.

[65] Gordon V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.

[66] Jay J. Kim, Jay J. Kim, William E. Winkler, and William E. Winkler. Multiplicative noise for masking continuous data. Technical report, Statistical Research Division, US Bureau of the Census, Washington D.C, 2003.

[67] Cheng-jun Li and Tian-qi Yang. Effective mining of fuzzy quantitative weighted association rules. In *ICEE*, pages 1418–1421. IEEE, 2010.

[68] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. JEP-classifier: Classification by aggregating jumping emerging patterns. Technical report, February 1999.

[69] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In Zighed et al. [141], pages 191–200.

[70] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowl. Inf. Syst.*, 3(2):131–145, 2001.

[71] Jinyan Li, Guozhu Dong, Kotagiri Ramamohanaro, and Limsoon Wong. DeEPs: A new instance-based discovery and classification system, 2001.

[72] Jinyan Li, Kotagiri Ramamohanarao, and Guozhu Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. In Pat Langley, editor, *ICML*, pages 551–558. Morgan Kaufmann, 2000.

[73] Jinyan Li, Kotagiri Ramamohanarao, and Guozhu Dong. Combining the strength of pattern frequency and distance for classification. In David Wai-Lok Cheung, Graham J. Williams, and Qing Li, editors, *PAKDD*, volume 2035 of *Lecture Notes in Computer Science*, pages 455–466. Springer, 2001.

[74] Xueming Li, Zhijun Liu, and Chuan Zuo. Hiding association rules based on relative-non-sensitive frequent itemsets. In George Baciu, Yingxu Wang, Yiyu Yao, Witold Kinsner, Keith Chan, and Lotfi A. Zadeh, editors, *IEEE ICCI*, pages 384–389. IEEE Computer Society, 2009.

[75] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2000.

[76] Kun Liu, Chris Giannella, and Hillol Kargupta. An attacker's view of distance preserving maps for privacy preserving data mining. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *PKDD*, volume 4213 of *Lecture Notes in Computer Science*, pages 297–308. Springer, 2006.

[77] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica Sinica*, 7(4):815–840, 1997.

[78] Chung-Leung Lui and Korris Fu-Lai Chung. Discovery of generalized association rules with multiple minimum supports. In Zighed et al. [141], pages 510–515.

[79] Kazunori Matsumoto, Takeo Hayase, and Nobuyuki Ikeda. Data mining of generalized association rules using a method of partial-match retrieval. In Setsuo Arikawa and Koichi

Furukawa, editors, *Discovery Science*, volume 1721 of *Lecture Notes in Computer Science*, pages 160–171. Springer, 1999.

[80] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. Sliq: A fast scalable classifier for data mining. In *EDBT*, pages 18–32, 1996.

[81] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. Mdl-based decision tree pruning. In *KDD*, pages 216–221, 1995.

[82] Renée J. Miller and Yuping Yang. Association rules over interval data. In Peckham [86], pages 452–461.

[83] Chirag N. Modi, Udai Pratap Rao, and Dhiren R. Patel. An efficient solution for privacy preserving association rule mining. *International Journal of Computer and Network Security*, 2(5):79–85, May 2010.

[84] Tim Oates and David Jensen. Large datasets lead to overly complex models: An explanation and a solution. In *KDD*, pages 294–298, 1998.

[85] Michal Okoniewski. Mining numerical databases with generalized quantitative association rules. In *EDBT PhD Workshop*, 2000.

[86] Joan Peckham, editor. *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. ACM Press, 1997.

[87] John R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[88] John R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[89] John R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[90] John R. Quinlan. The minimum description length principle and categorical theories. In *ICML*, pages 233–241, 1994.

[91] John R. Quinlan. Bagging, boosting, and c4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[92] John R. Quinlan. Improved use of continuous attributes in c4.5. *J. Artif. Intell. Res. (JAIR)*, 4:77–90, 1996.

[93] John R. Quinlan. Improved use of continuous attributes in c4.5. *CoRR*, cs.AI/9603103, 1996.

[94] John R. Quinlan. Simplifying decision trees. *Int. J. Hum.-Comput. Stud.*, 51(2):497–510, 1999.

[95] John R. Quinlan and Ronald L. Rivest. Inferring decision trees using the minimum

description length principle. *Inf. Comput.*, 80(3):227–248, 1989.

[96] Cornelis J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.

[97] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989.

[98] Shariq J. Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 682–693. VLDB Endowment, 2002.

[99] Lior Rokach and Oded Z. Maimon. *Data mining with decision trees: theory and applications*, volume 69 of *Series in machine perception and artificial intelligence*. World Scientific Publishing Co., Singapore; Philadelphia, PA, USA; River Edge, NJ, USA, 2008.

[100] Ashok Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In Dayal et al. [35], pages 432–444.

[101] Yücel Saygin, Vassilios S. Verykios, and Chris Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.

[102] Yücel Saygin, Vassilios S. Verykios, and Ahmed K. Elmagarmid. Privacy preserving association rule mining. In *RIDE*, pages 151–158, 2002.

[103] John C. Shafer, Rakesh Agrawal, and Manish Mehta. Sprint: A scalable parallel classifier for data mining. In C. Mohan Nandlal L. Sarda T. M. Vijayaraman, Alejandro P. Buchmann, editor, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 544–555. Morgan Kaufmann, 1996.

[104] Mark Shaneck, Yongdae Kim, and Vipin Kumar. Privacy preserving nearest neighbor search. In *ICDM Workshops*, pages 541–545. IEEE Computer Society, 2006.

[105] Mark Shaneck, Yongdae Kim, and Vipin Kumar. Privacy preserving nearest neighbor search. Technical Report TR 06-014, University of Minnesota, 2006.

[106] Claude E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[107] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In Dayal et al. [35], pages 407–419.

[108] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In H. V. Jagadish and Inderpal Singh Mumick, editors, *SIGMOD*

Conference, pages 1–12. ACM Press, 1996.

[109] Lambert M. Surhone, Miriam T. Timpledon, and Susan F. Marseken. *Pearson's Chi-Square Test*. Betascript Publishers, Beau Bassin, 2010.

[110] Alan Taylor and William Zwicker. A characterization of weighted voting. In *Proc. of the AMS*, pages 1089–1094, 1992.

[111] Li-Min Tsai, Shu-Jing Lin, and Don-Lin Yang. Efficient mining of generalized negative association rules. In Xiaohua Hu, Tsau Young Lin, Vijay V. Raghavan, Jerzy W. Grzymala-Busse, Qing Liu, and Andrei Z. Broder, editors, *GrC*, pages 471–476. IEEE Computer Society, 2010.

[112] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644. ACM, 2002.

[113] Jaideep Vaidya and Chris Clifton. Privacy preserving naïve Bayes classifier for vertically partitioned data. In *SDM*, 2004.

[114] Jaideep Vaidya and Chris Clifton. Privacy-preserving decision trees over vertically partitioned data. In *DBSec*, pages 139–152, 2005.

[115] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. Privacy-preserving decision trees over vertically partitioned data. *TKDD*, 2(3), 2008.

[116] Jaideep Vaidya, Murat Kantarcioglu, and Chris Clifton. Privacy-preserving naïve Bayes classification. *VLDB J.*, 17(4):879–898, 2008.

[117] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving svm classification. *Knowl. Inf. Syst.*, 14(2):161–178, 2008.

[118] Jaideep Vaidya, Yu Michael Zhu, and Christopher W. Clifton. *Privacy Preserving Data Mining*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[119] Aad W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 1998.

[120] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[121] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yücel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[122] Chih-Chia Weng, Shan-Tai Chen, and Hung-Che Lo. A novel algorithm for completely hiding sensitive association rules. In Jeng-Shyang Pan, Ajith Abraham, and Chin-Chen

Chang, editors, *ISDA (3)*, pages 202–208. IEEE Computer Society, 2008.

[123] Shaofei Wu and Hui Wang. Research on the privacy preserving algorithm of association rule mining in centralized database. In Fei Yu and Qi Luo, editors, *ISIP*, pages 131–134. IEEE Computer Society, 2008.

[124] Yi Xia, Yirong Yang, and Yun Chi. Mining association rules with non-uniform privacy concerns. In Gautam Das, Bing Liu, and Philip S. Yu, editors, *DMKD*, pages 27–34. ACM, 2004.

[125] Yi Xia, Yirong Yang, Yun Chi, and Richard R. Muntz. Mining association rules with non-uniform privacy concerns. Technical Report CSD-TR No. 040015, ftp://ftp.cs.ucla.edu/tech-report/2004-reports/040015.pdf, UCLA, 2004.

[126] Li Xiong, Subramanyam Chitti, and Ling Liu. Mining multiple private databases using a knn classifier. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 435–440, 2007.

[127] Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SDM*, 2005.

[128] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167. IEEE, 1986.

[129] Show-Jane Yen. Mining generalized multiple-level association rules. In Zighed et al. [141], pages 679–684.

[130] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving svm classification on vertically partitioned data. In Wee Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang, editors, *PAKDD*, volume 3918 of *Lecture Notes in Computer Science*, pages 647–656. Springer, 2006.

[131] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12(3):372–390, 2000.

[132] Mohammed J. Zaki and Karam Gouda. Fast vertical mining using diffsets. In *KDD*, pages 326–335, 2003.

[133] Justin Z. Zhan. Using homomorphic encryption and digital envelope techniques for privacy preserving collaborative sequential pattern mining. In *ISI*, pages 331–334, 2007.

[134] Justin Z. Zhan, LiWu Chang, and Stan Matwin. Building k-nearest neighbor classifiers on vertically partitioned private data. In Xiaohua Hu, Qing Liu, Andrzej Skowron, Tsau Young Lin, Ronald R. Yager, and Bo Zhang, editors, *GrC*, pages 708–711. IEEE, 2005.

[135] Justin Z. Zhan, LiWu Chang, and Stan Matwin. Privacy preserving k-nearest neighbor classification. *I. J. Network Security*, 1(1):46–51, 2005.

[136] Justin Z. Zhan and Stan Matwin. Privacy-preserving data mining in electronic surveys. In Jian Chen, editor, *ICEB*, pages 1179–1185. Academic Publishers/World Publishing Corporation, 2004.

[137] Justin Z. Zhan and Stan Matwin. Privacy-preserving data mining in electronic surveys. *I. J. Network Security*, 4(3):318–327, 2007.

[138] Justin Z. Zhan, Stan Matwin, and LiWu Chang. Privacy-preserving naive bayesian classification over horizontally partitioned data. In Tsau Young Lin, Ying Xie, Anita Wasilewska, and Churn-Jung Liau, editors, *Data Mining: Foundations and Practice*, volume 118 of *Studies in Computational Intelligence*, pages 529–538. Springer, 2008.

[139] Nan Zhang, Shengquan Wang, and Wei Zhao. A new scheme on privacy-preserving data classification. In Robert Grossman, Roberto Bayardo, and Kristin P. Bennett, editors, *KDD*, pages 374–383. ACM, 2005.

[140] Weining Zhang. Mining fuzzy quantitative association rules. In *ICTAI*, pages 99–102, 1999.

[141] Djamel A. Zighed, Henryk J. Komorowski, and Jan M. Zytkow, editors. *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings*, volume 1910 of *Lecture Notes in Computer Science*. Springer, 2000.

A. Additional Experimental Results

This section presents the additional results of the experiment we conducted (e.g., the results of the experiments for different data sets or different parameters of tested algorithms). The results were not presented in the main part of the thesis but one may be interested in these results and wanted to compare all of them.

A.1. Experimental Results for Optimisation for MASK Scheme

This section shows the additional results of the experiments presented in Section 4 for different data sets or parameters of the algorithms. The structure of the presented results is the same as in Section 4 (the results for MASK and MMASK are presented). The sets are characterised in Section 4, as well.

Table A.1. The results of mining the frequent sets in the set T20I16D50kN200 with parameters p = 0.5, q = 0.87, rThreshold = 3, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	161	5.0	3.7	1.8	158	161	5.0	3.7	1.8	158
2	3588	3752	11.6	9.3	13.9	3255	3752	11.6	9.3	13.9	3255
3	7582	12386	20.9	23.6	86.9	5796	12386	20.9	23.6	86.9	5796
4	976	8196	37.1	36.6	776.3	619	518	11.0	59.3	12.4	397
5	1	547	60.0	0.0	54600.0	1	0	-	100.0	0.0	0
6	0	2	-	-	-	0	0	-	-	-	0

Table A.2. The results of mining the frequent sets in the set T20I16D50kN200 with parameters p = 0.5, q = 0.77, rThreshold = 3, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	7.3	3.0	3.0	159	164	7.3	3.0	3.0	159
2	3588	4735	25.4	17.6	49.5	2958	4735	25.4	17.6	49.5	2958
3	7582	27872	62.0	40.3	307.9	4530	27872	62.0	40.3	307.9	4530
4	976	17643	142.8	63.4	1771.1	357	1751	20.4	65.2	144.6	340
5	1	521	-	100.0	52100.0	0	4	-	100.0	400.0	0

Table A.3. The results of mining the frequent sets in the set T10I8D100kN100 with parameters p = 0.6, q = 0.6, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	95	10.7	4.1	1.0	94	95	10.7	4.1	1.0	94
2	2522	2665	107.6	28.6	34.3	1800	2665	107.6	28.6	34.3	1800
3	10930	17618	407.6	57.5	118.7	4641	17618	407.6	57.5	118.7	4641
4	10185	12151	959.2	89.4	108.7	1075	10995	114.0	85.0	92.9	1531
5	2021	188	986.8	99.7	9.0	7	591	89.6	96.3	25.6	74
6	24	0	-	100.0	0.0	0	3	-	100.0	12.5	0

Table A.4. The results of mining the frequent sets in the set T10I8D100kN100 with parameters p = 0.7, q = 0.7, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	97	6.3	1.0	0.0	97	97	6.3	1.0	0.0	97
2	2522	2784	26.4	14.0	24.4	2168	2784	26.4	14.0	24.4	2168
3	10930	19563	49.9	28.6	107.5	7808	19563	49.9	28.6	107.5	7808
4	10185	25781	81.8	43.4	196.5	5766	8978	16.4	34.3	22.5	6687
5	2021	5109	153.4	68.3	221.1	640	1563	16.3	51.0	28.3	991
6	24	74	12.8	91.7	300.0	2	59	21.2	79.2	225.0	5

Table A.5. The results of mining the frequent sets in the set T10I8D100kN100 with parameters p = 0.8, q = 0.8, rThreshold = 3, minimumSupport = 0.005

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	98	4.6	0.0	0.0	98	98	4.6	0.0	0.0	98
2	2522	2592	11.2	5.8	8.6	2375	2592	11.2	5.8	8.6	2375
3	10930	12512	14.9	12.9	27.3	9525	12512	14.9	12.9	27.3	9525
4	10185	14169	17.3	18.9	58.1	8255	8485	10.1	22.0	5.3	7943
5	2021	4076	20.1	27.6	129.2	1464	1282	11.4	40.6	4.1	1200
6	24	270	24.4	29.2	1054.2	17	7	9.5	87.5	16.7	3
7	0	1	-	-	-	0	0	-	-	-	0

Table A.6. The results of mining the frequent sets in the set T20I16D50kN200 with parameters p = 0.7, q = 0.7, rThreshold = 3, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	5.5	3.0	3.0	159	164	5.5	3.0	3.0	159
2	3588	3962	14.0	12.2	22.6	3152	3962	14.0	12.2	22.6	3152
3	7582	15454	24.5	26.5	130.3	5571	15454	24.5	26.5	130.3	5571
4	976	9714	40.5	38.9	934.2	596	919	9.5	44.3	38.4	544
5	1	574	41.0	0.0	57300.0	1	0	-	100.0	0.0	0
6	0	3	-	-	-	0	0	-	-	-	0

Table A.7. The results of mining the frequent sets in the set T20I16D50kN200 with parameters p = 0.8, q = 0.8, rThreshold = 3, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	3.6	1.2	1.2	162	164	3.6	1.2	1.2	162
2	3588	3668	6.2	4.6	6.9	3422	3668	6.2	4.6	6.9	3422
3	7582	8279	8.0	11.1	20.3	6738	8279	8.0	11.1	20.3	6738
4	976	1361	9.0	23.1	62.5	751	648	7.0	37.7	4.1	608
5	1	7	10.1	0.0	600.0	1	0	-	100.0	0.0	0

Table A.8. The results of mining the frequent sets in the set Dna with parameters p = 0.7, q = 0.7, rThreshold = 3, minimumSupport = 0.05

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	181	181	8.0	0.0	0.0	181	181	8.0	0.0	0.0	181
2	13126	10131	33.8	32.4	9.6	8877	10131	33.8	32.4	9.6	8877
3	11118	94253	44.5	49.5	797.3	5611	94253	44.5	49.5	797.3	5611
4	1403	70498	55.2	66.4	4991.2	471	25836	32.7	63.9	1805.3	507
5	174	2966	43.5	93.7	1698.3	11	286	10.8	85.6	150.0	25
6	4	2	-	100.0	50.0	0	0	-	100.0	0.0	0

Table A.9. The results of mining the frequent sets in the set Dna with parameters p = 0.8, q = 0.8, rThreshold = 3, minimumSupport = 0.05

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	181	181	4.8	0.0	0.0	181	181	4.8	0.0	0.0	181
2	13126	11809	14.9	17.6	7.6	10815	11809	14.9	17.6	7.6	10815
3	11118	23650	16.2	33.2	145.9	7425	23650	16.2	33.2	145.9	7425
4	1403	6092	17.1	36.3	370.5	894	3012	12.6	35.0	149.7	912
5	174	397	16.8	50.0	178.2	87	84	12.5	62.1	10.3	66
6	4	7	2.0	75.0	150.0	1	0	-	100.0	0.0	0

Table A.10. The results of mining the frequent sets with the relaxation in the set T20I16D50kN200 with parameters p = 0.5, q = 0.87, relax = 0.05, rThreshold = 3, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	5.1	2.4	2.4	160	164	5.1	2.4	2.4	160
2	3588	3947	11.7	7.4	17.4	3322	3947	11.7	7.4	17.4	3322
3	7582	14119	20.7	20.5	106.7	6029	14119	20.7	20.5	106.7	6029
4	976	10136	36.5	34.7	973.3	637	717	11.5	49.9	23.4	489
5	1	734	60.0	0.0	-	1	0	-	100.0	0.0	0
6	0	3	-	-	-	0	0	-	-	-	0

Table A.11. The results of mining the frequent sets with the reduction relaxation in the set T20I16D50kN200 with parameters p = 0.5, q = 0.87, rrelax = 0.05, rThreshold = 3, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	161	5.0	3.7	1.8	158	161	5.0	3.7	1.8	158
2	3588	3752	11.6	9.3	13.9	3255	3752	11.6	9.3	13.9	3255
3	7582	12386	20.9	23.6	86.9	5796	12386	20.9	23.6	86.9	5796
4	976	8196	37.1	36.6	776.3	619	712	11.5	50.0	23.0	488
5	1	547	60.0	0.0	54600.0	1	0	-	100.0	0.0	0
6	0	2	-	-	-	0	0	-	-	-	0

Table A.12. The results of mining the frequent sets with both relaxations in the set T20I16D50kN200 with parameters p = 0.5, q = 0.87, relax = 0.05, rrelax = 0.05, rThreshold = 3, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	5.1	2.4	2.4	160	164	5.1	2.4	2.4	160
2	3588	3947	11.7	7.4	17.4	3322	3947	11.7	7.4	17.4	3322
3	7582	14119	20.7	20.5	106.7	6029	14119	20.7	20.5	106.7	6029
4	976	10136	36.5	34.7	973.3	637	979	12.3	41.3	41.6	573
5	1	734	60.0	0.0	73300.0	1	0	-	100.0	0.0	0
6	0	3	-	-	-	0	0	-	-	-	0

Table A.13. The results of mining the frequent sets with different randomisation factors for items in the set T20I16D50kN200 (p = 0.5 .. 0.4, q = 0.97 .. 0.98, rThreshold = 3, minimumSupport = 0.02)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	2.6	1.2	1.2	162	164	2.6	1.2	1.2	162
2	3588	4187	8.8	2.5	19.2	3499	4187	8.8	2.5	19.2	3499
3	7582	8904	10.9	13.1	30.5	6589	8904	10.9	13.1	30.5	6589
4	976	2471	17.4	28.3	181.5	700	393	11.5	61.6	1.8	375
5	1	148	-	100.0	14800.0	0	0	-	100.0	0.0	0

Table A.14. The results of mining the frequent sets with different randomisation factors for items and the relaxation in the set T20I16D50kN200 (p = 0.5 .. 0.4, q = 0.97 .. 0.98, relax = 0.01, rThreshold = 3, minimumSupport = 0.02)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	2.6	1.2	1.2	162	164	2.6	1.2	1.2	162
2	3588	4234	8.8	2.2	20.2	3508	4234	8.8	2.2	20.2	3508
3	7582	9182	10.9	12.1	33.2	6668	9182	10.9	12.1	33.2	6668
4	976	2623	17.3	27.2	195.9	711	429	11.7	58.4	2.4	406
5	1	167	-	100.0	16700.0	0	0	-	100.0	0.0	0

Table A.15. The results of mining the frequent sets with different randomisation factors for items and the reduction relaxation in the set T20I16D50kN200 (p = 0.5 .. 0.4, q = 0.97 .. 0.98, rrelax = 0.02, rThreshold = 3, minimumSupport = 0.02)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	2.6	1.2	1.2	162	164	2.6	1.2	1.2	162
2	3588	4187	8.8	2.5	19.2	3499	4187	8.8	2.5	19.2	3499
3	7582	8904	10.9	13.1	30.5	6589	8904	10.9	13.1	30.5	6589
4	976	2471	17.4	28.3	181.5	700	451	11.7	56.9	3.1	421
5	1	148	-	100.0	14800.0	0	0	-	100.0	0.0	0

Table A.16. The results of mining the frequent sets with different randomisation factors for items and both relaxations in the set T20I16D50kN200 (p = 0.5 .. 0.4, q = 0.97 .. 0.98, relax = 0.01, rrelax = 0.02, rThreshold = 3, minimumSupport = 0.02)

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	2.6	1.2	1.2	162	164	2.6	1.2	1.2	162
2	3588	4234	8.8	2.2	20.2	3508	4234	8.8	2.2	20.2	3508
3	7582	9182	10.9	12.1	33.2	6668	9182	10.9	12.1	33.2	6668
4	976	2623	17.3	27.2	195.9	711	480	11.6	54.4	3.6	445
5	1	167	-	100.0	16700.0	0	0	-	100.0	0.0	0

Table A.17. The results of mining the frequent sets with different randomisation factors for items in the set Dna, p=0.5 .. 0.4, q=0.87 .. 0.88, rThreshold = 2, minimumSupport = 0.05

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	181	181	8.2	0.0	0.0	181	181	8.2	0.0	0.0	181
2	13126	10635	37.6	29.2	10.2	9291	10635	37.6	29.2	10.2	9291
3	11118	115387	51.5	44.1	982.0	6212	24212	30.9	42.0	159.8	6446
4	1403	137205	82.6	70.8	9750.2	409	3935	20.7	69.5	250.0	428
5	174	16018	84.1	89.7	9195.4	18	78	11.8	93.7	38.5	11
6	4	168	-	100.0	4200.0	0	0	-	100.0	0.0	0

Table A.18. The results of mining the frequent sets with different randomisation factors for items in the set T20I16D50kN200, p=0.5 .. 0.4, q=0.87 .. 0.88, rThreshold = 2, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	168	5.5	1.8	4.3	161	168	5.5	1.8	4.3	161
2	3588	4638	19.1	8.1	37.3	3299	4638	19.1	8.1	37.3	3299
3	7582	21029	37.1	30.1	207.4	5303	6055	10.7	28.0	7.9	5458
4	976	15130	83.7	45.3	1495.5	534	793	10.8	54.5	35.8	444
5	1	1114	-	100.0	111400.0	0	0	-	100.0	0.0	0
6	0	1	-	-	-	0	0	-	-	-	0

Table A.19. The results of mining the frequent sets with different randomisation factors for items in the set T20I16D50kN200, p=0.5 .. 0.4, q=0.97 .. 0.98, rThreshold = 2, minimumSupport = 0.02

L.	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	164	164	2.6	1.2	1.2	162	164	2.6	1.2	1.2	162
2	3588	4187	8.8	2.5	19.2	3499	4187	8.8	2.5	19.2	3499
3	7582	8904	10.9	13.1	30.5	6589	5436	10.9	29.4	1.1	5354
4	976	2471	17.4	28.3	181.5	700	286	14.8	71.2	0.5	281
5	1	148	-	100.0	14800.0	0	0	-	100.0	0.0	0

A.2. Experimental Results for Privacy Preserving Classification with Emerging Patterns

This section shows the additional results of the experiments presented in Section 6 for different parameters of discretisation. The structure of the presented results is the same as in Section 6. The sets are characterised in Section 6, as well.

Table A.20. The results of classification with the lazy JEP classifier, the minimal support equal to 0.05 and eager EP classifier with the minimal support equal to 0.2 and the minimal growth rate equal to 2 algorithms and the retention replacement perturbation (continuous attributes are discretised into 10 bins with equal number of samples).

Set	p	Acc.	t[s]	Sens.	Spec.	Prec.	F	e Acc.	e t[s]	T acc.
Aus.	0.5	0.8270	1.03	0.8386	0.8168	0.7892	0.8100	0.7561	0.11	0.8171
	0.3	0.8143	1.18	0.8421	0.7914	0.7709	0.7980	0.7623	0.11	0.7612
	0.15	0.7530	1.02	0.6848	0.8049	0.7461	0.7047	0.6624	0.89	0.6301
Bre.	0.5	0.8984	0.73	0.7747	0.9720	0.9408	0.8441	0.9603	0.23	0.9344
	0.3	0.8964	0.79	0.7679	0.9706	0.9396	0.8398	0.9554	0.18	0.9236
	0.15	0.8902	0.84	0.7484	0.9716	0.9430	0.8283	0.9368	0.21	0.8939
Dia.	0.5	0.6687	0.38	0.6121	0.7758	0.8334	0.7018	0.5529	0.02	0.6682
	0.3	0.6396	0.38	0.5921	0.7292	0.8032	0.6750	0.5779	0.03	0.6352
	0.15	0.5847	0.38	0.5575	0.6349	0.7421	0.6270	0.5862	0.07	0.5929
Iris	0.5	0.8928	0.01	0.9597	0.9626	0.7471	0.7262	0.8959	0.01	0.8027
	0.3	0.8009	0.01	0.9223	0.9286	0.6073	0.5656	0.8136	0.01	0.6148
	0.15	0.5937	0.01	0.8409	0.8528	0.4638	0.3747	0.6517	0.02	0.4411
Wine	0.5	0.8444	0.05	0.9356	0.9418	0.6239	0.5918	0.8518	0.06	0.7134
	0.3	0.7632	0.05	0.9026	0.9101	0.5203	0.4728	0.7295	0.37	0.5403
	0.15	0.5336	0.05	0.8102	0.8260	0.3881	0.2928	0.5939	3.03	0.3954

A.3. Experimental Results for Privacy Preserving Classification with Meta-learning

This section shows the additional results of the experiments presented in Section 7 for different parameters of discretisation (the structure of the presented results is the same as in Section 7 and the sets are characterised in Section 7, as well).

Table A.21. The accuracy of classification with the usage of meta-learning (simultaneously bagging and boosting with 5 classifiers per each method) for only Local and By class reconstruction types and different combinations of algorithms compared to Local reconstruction type and AS.EA algorithms and the retention replacement perturbation (continuous attributes are discretised into 10 bins with equal number of samples).

p	Set	Acc.	Sens.	Spec.	Prec.	F	Time[s]
0.5	mAustralian	0.8577	0.8541	0.8608	0.8345	0.8413	14.4016
	LoAustralian	0.8179	0.8031	0.8301	0.7945	0.7955	0.2275
0.3	mAustralian	0.8450	0.8201	0.8648	0.8337	0.8227	13.9423
	LoAustralian	0.7702	0.7454	0.7902	0.7453	0.7401	0.2268
0.15	mAustralian	0.7249	0.5498	0.8631	0.7673	0.6321	16.0003
	LoAustralian	0.6285	0.5124	0.7201	0.6007	0.5466	0.2925
0.5	mCredit-g	0.7168	0.2398	0.9232	0.5730	0.3267	20.8032
	LoCredit-g	0.6704	0.3940	0.7900	0.4470	0.4122	0.4001
0.3	mCredit-g	0.7022	0.1379	0.9455	0.5172	0.2497	17.8199
	LoCredit-g	0.6291	0.3609	0.7450	0.3799	0.3605	0.3891
0.15	mCredit-g	0.6888	0.0766	0.9520	0.4561	0.2552	17.1306
	LoCredit-g	0.5926	0.3311	0.7067	0.3255	0.3104	0.3991
0.5	mDiabetes	0.7219	0.8375	0.5104	0.7575	0.7934	7.5846
	LoDiabetes	0.6689	0.7523	0.5142	0.7406	0.7437	0.1101
0.3	mDiabetes	0.6950	0.8445	0.4222	0.7293	0.7799	7.2308
	LoDiabetes	0.6352	0.7233	0.4740	0.7177	0.7167	0.1110
0.15	mDiabetes	0.6549	0.8611	0.2773	0.6884	0.7618	7.2865
	LoDiabetes	0.5929	0.6991	0.3981	0.6835	0.6862	0.1175
0.5	mSegment	0.9036	0.9036	0.9839	0.9027	0.9017	241.2220
	LoSegment	0.8487	0.8493	0.9748	0.8504	0.8461	3.8142
0.3	mSegment	0.8631	0.8631	0.9772	0.8637	0.8584	282.8256
	LoSegment	0.7629	0.7638	0.9605	0.7662	0.7557	4.5752
0.15	mSegment	0.7724	0.7730	0.9621	0.7725	0.7627	341.5651
	LoSegment	0.5893	0.5898	0.9315	0.5941	0.5798	5.8598

B. Test bed

The experimental system was created to perform all experiments presented in this thesis. The system implements all proposed in the thesis methods and chosen methods presented in literature, which were used to evaluate the new solutions by compering them with the existing ones.

The system was implemented in C++ with STL library. All experiments were performed on Intel Dual Core 2.2 GHz machine with 2 GB of RAM memory and load of the system kept at a low level.

The privacy preserved decision tree implemented for the purpose of the experiments used the SPRINT [103] algorithm modified to incorporate privacy according to Sections 3.7.1 and 3.7.2.

The SPRINT algorithm creates lists for each attribute. A list for a given attribute consists of an attribute value, a class label, and an index for each record. Lists for continuous attributes are sorted by an attribute value. Initial lists are associated with a root of a tree. As a node is split, attribute lists are partitioned and associated with children. An order is preserved in partitioned attribute lists, hence they do not need to be sorted once again.

To find the best split point, SPRINT uses gini index (for details about ginin index refer to Equation 2.2). For continuous attributes, a histogram of a class attribute is calculated. The histogram shows the number of records with a given value of a class attribute divided into groups with values of a split attribute greater than a split point and values less than a split point. Possible split points are midpoints between consecutive values of a split attribute. A split point with the lowest calculated value of gini index is chosen for each continuous attribute. For nominal attributes, the frequency distribution of a split and class attributes are calculated. Based on this distribution gini index is calculated for all combination of values of a split attribute. A test (a combination of values of a split attribute) with the lowest calculated value of gini index is chosen for each nominal attribute. A final split attribute is the one with the lowest value of gini index among all types of attributes. Having chosen a test, lists are partitioned according to this test.

In our implementation we assumed that an attribute can be used as a test once in a path. The

stopping criterion was met when at least one of the following conditions was satisfied: 1) all samples in a node have the same class label, 2) a sample set is empty, 3) all attributes were used as tests in a path.

Before each experiment missing values were imputed with the mean for continuous attributes and the mode for nominal attributes.

The heuristic of finding the best test for nominal attributes with high number of possible values presented in [103] was not used. However, we restricted subsets of possible values of nominal attributes to subsets which the support is less than or equal to integer value of the half of the maximal possible support. Furthermore, for continuous attributes, we omitted the process of checking the values that lie between samples with the same class label.

Pruning was implemented according to the Minimum Description Length (MDL) principle (for details please refer to Section 2.2).

To incorporate privacy algorithms presented in Section 3.7.2 were implemented. A decision tree used either AS or EM algorithm for a probability reconstruction of continuous attributes and either EM/AS or EQ algorithm for a probability reconstruction of nominal attributes. The ARVeSNA algorithm for assigning reconstructed values of nominal attributes to samples was implemented. For continuous attributes, the algorithm for assigning reconstructed values to samples presented in Section 3.7.2 was used.