Technical Report

Remote Patient Monitoring System

(RPMS)

“A Formal Usability Constraints Model for Watermarking of Outsourced Datasets”

June, 2012

FAST National University of Computer & Emerging Sciences, Islamabad, Pakistan
Ownership-preserving Data Mining: A Formal Usability Constraints Model for Watermarking of Outsourced Datasets

M. Kamran and Muddassar Farooq

Abstract

The large datasets are being mined to extract hidden knowledge and patterns that assist decision makers in making effective, efficient and timely decisions in an ever increasing competitive world. This type of “knowledge-driven” data mining activity is not possible without sharing the “datasets” between their owners and data mining experts (or corporations); as a consequence, protecting ownership (by embedding watermark) on the datasets is becoming relevant. The most important challenge in watermarking (to be mined) datasets is: how to preserve knowledge in features or attributes? Usually, an owner needs to manually define “Usability constraints” for each type of dataset to preserve the contained knowledge. The major contribution of this paper is a novel formal model that facilitates a data owner to define usability constraints – to preserve the knowledge contained in the dataset – in an automated fashion. The model aims at preserving “classification potential” of each feature and other major characteristics of datasets that play an important role during the mining process of data; as a result, learning statistics and decision making rules also remain intact. We have implemented our model and integrated it with a new watermark embedding algorithm to prove that the inserted watermark not only preserves the knowledge contained in a dataset but also significantly enhances watermark security compared with existing techniques. We have tested our model on 25 different data-mining datasets to show its efficacy, effectiveness and the ability to adapt and generalize.

Index Terms

Watermarking datasets, database watermarking, data mining, right protection, data usability, knowledge-preserving watermarking.

I. INTRODUCTION

THE large datasets – generated from very large databases – are being mined to extract hidden knowledge and patterns that are proving useful for decision makers to make effective, efficient and timely decisions in a competitive world. This type of “knowledge-driven” data mining expert systems cannot be designed and developed until the owner of data is willing to share (or outsource) the dataset with data mining experts (or corporations). Recently, a startup company – Kaggle (www.kaggle.com) – has made a business case out of this need where organizations outsource their datasets and the associated business challenge to data mining experts with an objective to find novel solutions to the posted problem [1]. This validates the thesis that corporations with large databases want to get the optimized solution to a problem by leveraging the power of crowd-sourcing.

In the emerging field of “sharing datasets” with the intended recipients, protecting ownership on the datasets is becoming a challenge in itself. Recently, an article reported the illegal sale of patients data and the concerned patients have sued the original hospital for breaching their privacy [2]. An even bigger concern is that the recipient may try to take credit for contribution towards knowledge discovery and data mining (KDD) by claiming the false ownership of the shared data. To mitigate these threats, a non-disclosure agreement is usually signed with the recipient binding him that he will not sale (or further share) the dataset and will also not claim the ownership of the data. If the recipient breaches the agreement, the legitimate data owner can only sue him if he can prove in a court of Law his ownership in an unambiguous manner over the dataset. Watermarking is the commonly used mechanism to enforce and prove ownership for the digital data in different formats like audio, video, image, relational database, text and software [3], [4], [5]. The most important challenge in watermarking data mining datasets is: how to preserve knowledge in features or attributes during the embedding of watermark bits?

In order to preserve the knowledge in the dataset, one has to ensure that the predictive ability of a feature or an attribute is preserved; as a result, the classification results (and associated rules) remain preserved as well. In order to meet this requirement,
an owner is supposed to define the “usability constraints” that provide the distortion band – within which the values of a feature can change – for each feature. As a result, the classification accuracy of the dataset remains unaltered. In addition to this, the inserted watermark should be imperceptible and robust against any type of sophisticated attacks that can be launched on the watermarked dataset. To conclude, defining “usability constraints” is a challenge because a user has to strike a balance between “robustness of watermark” and “preserving knowledge contained in features”. For example, biomedical datasets may tolerate only very small amount of change – during the embedding of a watermark – in their features’ set to preserve the diagnosis rules.

At the moment, the process of defining “usability constraints” is manually repeated and is dependent on the dataset and its intended application or use. Moreover, if right protection is enforced using “fingerprinting”, the owner of data may need to define different “usability constraints” on the same dataset because in fingerprinting a different watermark for each user is added. To the best of our knowledge, no technique has been proposed to model the “usability constraints” while watermarking data mining datasets in particular, and other relational datasets in general. In this paper, we propose a novel formal model for identifying the essential “usability constraints” which must be enforced while embedding watermark in a dataset. The major contributions of our paper are:

- We propose a generic formal model to define “usability constraints” on a dataset that not only ensures the robustness of an inserted watermark but also preserves the knowledge contained in the dataset. The proposed technique is independent of the type of a dataset i.e numeric, non-numeric or strings.
- We have integrated our model in a new knowledge-preserving watermarking scheme to validate its efficacy and effectiveness.
- We show that the new knowledge-preserving watermarking scheme has significantly enhanced the security – (in terms of) deleting or changing the watermark – compared with existing techniques.
- We have conducted experiments on 25 publicly available datasets to prove that our technique can generalize to any type of dataset and achieves its objective of preserving the classification accuracy when mined with a machine learning classifier.
- We have evaluated our scheme on 5 well known machine learning classifiers – J48, SMO, Naive Bayes, IBk, and JRip – to show that their classification accuracy is preserved.

We briefly describe the related work in Section II and then introduce our approach in Section III. We present the formal model in Section IV. Subsequently, we explain that how the “usability constraints”, defined by our model, can be given as input to a watermark embedding scheme. We report the results of our experiments in Section VI. Finally, we conclude the paper with an outlook to our future work.

II. RELATED WORK

To the best of our knowledge, no technique has (so far) been proposed for modeling “usability constraints” for watermarking data mining datasets. In the work of Agrawal et al. [6], the first well known technique for watermarking numeric attributes in a database has been proposed. In this technique, message authenticated code (MAC) is calculated with the help of a secret key to identify the candidate tuples. Sion et al. [7] presented a marker tuples based watermarking technique for relational databases but these techniques are not applicable to data mining datasets because they do not aim at preserving the knowledge contained in the dataset.

Shehab et al. [8] proposed a partitioning based database watermarking technique. They modeled the process of watermark insertion as a constraint optimization problem and tested genetic algorithm (GA) and pattern search (PS) [9] optimizers. They select PS because it is able to optimize in realtime. But this technique requires defining “usability constraints” manually and does not account for preserving the knowledge contained in the data mining datasets.

Recently, in [10], we have proposed a relevant technique protecting ownership of electronic medical records (EMR) system. In this technique, information gain is used to identify the predictive ability of all features present in the EMR. The numeric feature(s) with the least predictive ability are selected to embed watermark bits to ensure information-preserving characteristic. This technique is only limited to information gain and does not generalize to other feature selection schemes. Moreover, it does not take into account certain characteristics of dataset that play a vital role in classification of the dataset. Since the major motivation of the technique is information-preserving watermarking; therefore, it does not describe any mechanism to model the “usability constraints”. Moreover, this watermarking technique is limited to numeric features only.

In comparison, the focus of our current work is on developing a formal model to define “usability constraints” for watermarking of data mining datasets in such a way that the watermark is not only robust but the knowledge contained in the dataset is also preserved. Furthermore, we also provide a mechanism to logically group the dataset into groups such that high ranked features might also be watermarked during watermarking. This is a significant enhancement because if only low ranked features are watermarked, an attacker can launch malicious attacks on low ranked features only without compromising the data quality to a great extent. In this context, our data grouping approach enables a data owner to embed a watermark in high ranked features as well while still satisfying the “usability constraints” imposed by our formal model. Last but not the least, we have significantly enhanced our recently proposed information-preserving watermarking scheme [10] for data mining datasets in such a way that it can now watermark any type of features – numeric, non-numeric or strings.
III. Approach Overview

In this paper, we present two contributions: (1) a novel framework model which derives usability constraints for all kinds of datasets; and (2) a new watermarking technique that works for numeric, non-numeric and strings datasets. Our system takes the dataset as an input, models the “usability constraints” to be enforced during the watermark embedding in the dataset. Later it uses three different optimizers to find an optimum watermark that meets the relative constraints. The top level architecture of the proposed framework is shown in Figure 1.

![Top level architecture of the proposed framework.](image)

In the first step, the predictive ability of features, present in the dataset, are calculated and the features are ranked on the basis of computed predictive ability. Using these ranks, the next step is to generate the logical groups of features. In this step, “local usability constraints” are defined for each logical group. Similarly, the “global usability constraints” are also defined that are applicable for the whole dataset. Finally, both types of constraints are used to build a meta-constraints model that is given as an input to the watermarking scheme.

IV. A Formal Model for “Usability Constraints”

We now present our formal model to define “Usability Constraints” that preserve the knowledge during the process of inserting watermark in the dataset.

**Definition 1.** [Tuple.] A tuple $\tau$ is an ordered list of elements.

The tuple is used as a basic unit for referring different parameters of a dataset.

**Definition 2.** [Learning algorithm.] Given a dataset $D_O$ with $M$ features, $N$ instances, and a class attribute $Y$, a learning algorithm $\Gamma$, groups $N$ instances into $\alpha$ different groups. Formally,

$$ \Gamma : D_O^M \rightarrow (C_\alpha, C_S) $$

where $\alpha$ is the number of distinct items in $Y$. A learning algorithm may be a classification algorithm or a clustering algorithm.

**Definition 3.** [Learning statistics.] Learning statistics $C_S$ is a tuple containing the classification statistics (or accuracy) of a particular learning algorithm. Formally:

$$ C_S \leftarrow \Gamma : D_O $$

These statistics include $TP_{rate}$, $FP_{rate}$ and decision rule boundaries $\mathcal{R}_b$. They are defined as:

$$ TP_{rate} = \left( \frac{TP}{TP + FN} \right) \times 100 $$

$$ FP_{rate} = \left( \frac{FP}{FP + TN} \right) \times 100 $$

**Decision rule boundaries** ($\mathcal{R}_b$): $\mathcal{R}_b$ denotes the threshold values that define the boundary of a particular decision rule.

**TP (true positive):** TP denotes the number of instances of a particular class correctly detected as instances of that class.

**FP (false positive):** For a particular class, the number of instances of other class(es) incorrectly detected as instances of that particular class.
TN (true negative): For a particular class, the number of instances correctly detected as instances of other class(es).

FN (false negative): For a particular class, the number of instances of that class incorrectly detected as instances of other class(es).

We store learning statistics in $C_S$, that is:

$$C_S = (TP_rate, FP_rate, \mathbb{R}_b)$$  \hspace{1cm} (5)

Definition 4. [Decision rules.] Given a dataset $D_O$ with $M$ features, a rule $r$ is a tuple constructed by mapping of $m$ features, with $m \subseteq M$, based on $C_S$ for identifying the class label $y$, where $y \in Y$. $\mathbb{R}$ contains all such rules as:

$$\mathbb{R} : (D_O, C_S) \rightarrow Y$$  \hspace{1cm} (6)

Definition 5. [Feature selection scheme.] A feature selection scheme $S$ transforms $M$-dimensional data $D_O$, having $N$ samples, $M$ features and a class attribute $Y$, in $m$-dimensional space $R^m$ (with $m \leq M$, such that $R^m \subseteq R^M$) that can yield “optimum” learning statistics. Formally,

$$S : D^M_O \rightarrow R^m$$  \hspace{1cm} (7)

In this paper, we have used 6 most commonly used different feature selection schemes: (mutual information ($I$), information gain ($IG$), information gain ratio (IGr), correlation based feature selection ($CFS$), consistency based feature subset evaluator ($CBF$), and principal components analysis ($PCA$)). All these feature selection schemes define the classification potential $C_P$ of the features.

Axiom 1. Different feature selection schemes can yield different learning statistics $C_S$ even for the same learning algorithm $\Gamma$.

Definition 6. [Mutual information.] Let $X$ and $Y$ be two random variables, then their mutual information can be defined as:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left( \frac{p(x, y)}{p(x)p(y)} \right)$$  \hspace{1cm} (8)

where $p(x, y)$ represents the joint probability distribution function of $x$ and $y$, and $p(x)$ and $p(y)$ are marginal distributions of $X$ and $Y$. Mutual information was introduced in [11]. In general, we denote the mutual information$^2$ $I(X; Y)$ (meaning the gain in information about $X$ after observing $Y$) by $I(X)$.

Definition 7. [Watermark embedding.] A watermark embedding is a transformation of a dataset $D_O$ to $D_W$ after embedding a watermark $W$.

$$\Phi : (D_O, W) \rightarrow D_W$$  \hspace{1cm} (9)

The dataset $D_W$ is shared with an intended recipient, therefore; the information lost during the process of watermark embedding must be within allowed limits in order to preserve the knowledge contained in $D_O$.

Definitions 5, 6, and 7 provide the basis for formulating Theorem 1.

Theorem 1. If for two datasets $D_O$ and $D_W$ each having $N$ instances and $M$ features, a feature $X_O \in D_O$ has values tuple $x^N_O \in X_O$ and a feature $X_W \in D_W$ has values tuple $x^N_W \in X_W$ then the classification potential of $X_O$ and $X_W$ will be the same if and only if the corresponding class label tuple for $x^1_O$ and $x^1_W$ is the same.

Proof: Let $Y_O$ and $Y_W$ be the class label tuples for $D_O$ and $D_W$ respectively. Also suppose that the datasets $D_O$ and $D_W$ consist of only one feature $X_O$ and $X_W$ respectively. The percentage classification potential of $X_O$, can be calculated as:

$$C_{P_{X_O}} = \left( \frac{I(X_O)}{\sum_{i=1}^{M} I(X_{O_i})} \right) \ast 100$$  \hspace{1cm} (10)

Similarly, the percentage classification potential of $X_W$ can be calculated as:

$$C_{P_{X_W}} = \left( \frac{I(X_W)}{\sum_{i=1}^{M} I(X_{W_i})} \right) \ast 100$$  \hspace{1cm} (11)

But classification potential of feature $X_O$ is calculated using Definition 6 as:

$$I(X_O; Y_O) = \sum_{y_o \in Y_O} \sum_{x_o \in X_O} p(x_o, y_o) \log \left( \frac{p(x_o, y_o)}{p(x_o)p(y_o)} \right)$$  \hspace{1cm} (12)

$^2$In this paper, unless otherwise specified, we use the terms predictive ability, mutual information, and classification potential interchangeably.
And for $X_W$:

$$I(X_W; Y_W) = \sum_{y_w \in Y_W} \sum_{x_w \in X_W} p(x_W, y_W) \log \left( \frac{p(x_W, y_W)}{p(x_W)p(y_W)} \right)$$

(13)

Now, it is evident from equations (10), (11), (12), and (13) that $C_{P_{X_O}} = C_{P_{X_W}}$ if and only if for each tuple $x_O^i$ and $x_W^i$, $y_O^i = y_W^i$.

Similarly, for datasets $D_O^M$ and $D_W^M$, each consisting of $M$ features, $C_{P_{X_O}} = C_{P_{X_W}}$ if and only if for each tuple $x_O^i$ and $x_W^i$, $y_O^i = y_W^i$, where $k = 1, 2, ..., M - 1, M$.

So the watermarking process should not modify the value of a candidate feature in such a way that its class label is changed. This will ensure that the learning statistics of a learning algorithm are preserved.

For example, assume we have a dataset of pregnant women and we want to identify hypertensive patients. As a rule, a normal pregnant woman in between the age of 20 to 30 has a systolic BP less than 120; therefore, after watermarking, the BP of a normal pregnant woman must remain less than 120 to avoid misdiagnosis.

**Definition 8. [Classification potential threshold.]** Given a dataset $D_O$ with $M$ features, the classification potential threshold ($C_{P_T}$) is a parameter for grouping of features – using $\gamma$ as the secret grouping parameter that has a value between 0 and 1 – based on their classification potentials. It is calculated as:

$$C_{P_T} = \gamma \times \sum_{i=1}^{M} \frac{I_i}{M}$$

(14)

This threshold is used to make different groups of a dataset so that the features with higher classification potentials are least modified during the process of watermarking. A data group $g_i$ contains features possessing similar classification potential under the threshold $C_{P_T}$.

**Definition 9. [Local usability Constraints.]** Local usability constraints $L_i$ is a tuple constituting mutual information $I(X)$ of the feature $X$ in a particular data group $g_i$.

$$L_i = I(X)$$

(15)

The local usability constraints are used to watermark features in a group $g_i$ and they are enforced at a group level only.

**Definition 10. [Global usability Constraints.]** Given a dataset, global usability constraints $G$ is a tuple that consists of features’ set produced by different feature selection schemes on that dataset.

In our case we are using 5 feature selection schemes – $IG$, $(IG_t)$, $(CFS)$, $(CBF)$, and $(PCA)$.

$$G = (R^m_{IG}, R^m_{IG_t}, R^m_{CFS}, R^m_{CBF}, R^m_{PCA})$$

(16)

Global usability constraints are enforced both at a group level and at the global dataset level. The features’ set, produced by applying a feature selection scheme to a group or a dataset, should remain unaltered.

The local usability constraints, global usability constraints and learning statistics lay the foundation for Lemma 1.

**Lemma 1.** If for two datasets $D_O$ and $D_W$, a feature selection schemes $S$ yields tuples $R^m_O$ and $R^m_W$ such that $R^m_O \neq R^m_W$, then learning statistics $C_{S_O}$ and $C_{S_W}$ obtained by applying learning algorithms on $R^m_O$ and $R^m_W$ will not be same.

**Proof:** Let $C_{S_O}$ and $C_{S_W}$ be the learning statistics of a learning algorithm $\Gamma_1$ after working on $R^m_O$ and $R^m_W$ respectively.

Now, as the results of the two feature selection schemes yielded $R^m_O$ and $R^m_W$ such that $R^m_O \neq R^m_W$, so according to our Definition 5 the “optimum” results of classification would be different for $R^m_O$ and $R^m_W$; hence we can safely conclude $C_{S_O} \neq C_{S_W}$. Therefore,

$$R^m_O \neq R^m_W \Rightarrow C_{S_O} \neq C_{S_W}$$

(17)

So, we conclude from Lemma 1 that for two datasets $D_O$ and $D_W$, a feature selection scheme $S$ should yield the same features’ set so that the learning statistics of an algorithm are preserved. The usability constraints on feature selection schemes are defined using Lemma 1.

**Definition 11. [Data separability.]** Given a dataset $D_O$, data separability is division of dataset $D_O$ into $\alpha$ sub-datasets such that:

$$D_{O_1} \cap D_{O_2} \cap ... \cap D_{O_\alpha} = \emptyset$$

(18)

and

$$D_{O_1} \cup D_{O_2} \cup ... \cup D_{O_\alpha} = D_O$$

(19)
Axiom 2. A dataset can be classified into $\alpha$ classes if and only if it is $(\alpha - 1)$ dimension separable.

The data separability helps to define the rule boundary of a decision.

Definition 12. [Data distribution.] Given a feature $X \in D_O$, a data distribution $\Omega$ is the number of occurrences of certain data values $x_i \in X$.

$$\Omega \leftarrow (\forall i | H(x_i))$$

(20)

where $H$ represents a data distribution function.

An ideal watermarking algorithm should preserve the data distribution of original dataset.

Lemma 2. If for two datasets $D_O$ and $D_W$, the data distributions $\Omega_O$ and $\Omega_W$ are such that $\Omega_O \neq \Omega_W$, then the learning statistics $C_{S_O}$ and $C_{S_W}$ of datasets $D_O$ and $D_W$ respectively, will not be the same.

Proof: Suppose a dataset $D_O$ has a class attribute $Y$ with $\alpha$ possible values, then $D_O$ can be classified correctly if it is $(\alpha - 1)$ dimension separable. Now, consider $\alpha = 2$, that is, $D_O$ has two class labels $c_1$ and $c_2$. According to Axiom 2 learning algorithm $\Gamma$ can be applied on $D_O$ to yield learning statistics $C_{S_O}$, if and only if $D_O$ is 1 dimension separable into sub-datasets $D_{O_1}$ and $D_{O_2}$ (Definition 11) such that:

$$D_{O_1} \cap D_{O_2} = ()$$

and

$$D_{O_1} \cup D_{O_2} = D_O$$

But this is possible if data distribution $\Omega_O$ is such that a boundary point separates $D_{O_1}$ from $D_{O_2}$.

Similarly for dataset $D_W$,

$$D_{W_1} \cap D_{W_2} = ()$$

and

$$D_{W_1} \cup D_{W_2} = D_W$$

Again, this is possible if data distribution $\Omega_W$ is such that a boundary point separates $D_{W_1}$ from $D_{W_2}$. So, if position of this boundary point is different from the position of boundary point of $D_{O_1}$ and $D_{O_2}$, then elements of $D_{W_1}$ will be different from $D_{O_1}$, and similarly elements of $D_{W_2}$ will be different from $D_{O_2}$; and as a consequence, learning statistics $C_{S_W} \neq C_{S_O}$.

Now, if $\alpha > 2$, then learning algorithm $\Gamma$ can be applied on $D_O$ to yield learning statistics $C_{S_O}$, if and only if $D_O$ is $(\alpha - 1)$ dimension separable into sub-datasets $D_{O_1}, D_{O_2}, ..., D_{O_\alpha}$ such that:

$$D_{O_1} \cap D_{O_2} \cap ... \cap D_{O_\alpha} = () \text{ and } D_{O_1} \cup D_{O_2} \cup ... \cup D_{O_\alpha} = D_O$$

Similarly for $D_W$, the learning algorithm $\Gamma$ can be applied to yield learning statistics $C_{S_W}$, if and only if $D_W$ is $(\alpha - 1)$ dimension separable into sub-datasets $D_{W_1}, D_{W_2}, ..., D_{W_\alpha}$ such that:

$$D_{W_1} \cap D_{W_2} \cap ... \cap D_{W_\alpha} = () \text{ and } D_{W_1} \cup D_{W_2} \cup ... \cup D_{W_\alpha} = D_W$$

But this is possible if data distribution $\Omega$ is such that $(\alpha - 1)$ boundary points separate $D_{O_1}, D_{O_2}, ..., D_{O_\alpha}$ from each other. So, if position of these boundary points is different for $D_{O_1}, D_{O_2}, ..., D_{O_\alpha}$ and $D_{W_1}, D_{W_2}, ..., D_{W_\alpha}$; then elements of $D_{O_1}, D_{O_2}, ..., D_{O_\alpha}$ and $D_{W_1}, D_{W_2}, ..., D_{W_\alpha}$ will not be same; and as a consequence, learning statistics $C_{S_O} \neq C_{S_W}$.

Hence, for two datasets $D_O$ and $D_W$, learning statistics $C_{S_O}$ and $C_{S_W}$ produced by learning algorithm $\Gamma$ will be different if the data distributions $\Omega_O$ and $\Omega_W$ of datasets $D_O$ and $D_W$ are not the same. ■

Definition 13. [Learning information loss.] If $C_{S_O}$ represent the learning statistics tuple obtained by applying a learning algorithm $\Gamma$ on dataset $D_O$, and $C_{S_W}$ represent the learning statistics` tuple obtained by applying a learning algorithm $\Gamma$ on dataset $D_W$, then learning information loss $C_{S_{L_{loss}}}$ is defined as:

$$C_{S_{L_{loss}}} = \frac{|C_{S_O} - C_{S_W}|}{C_{S_O}} \times 100$$

(21)

We use $C_{S_{L_{loss}}}$ to quantify the amount of information lost during the process of inserting watermark in a dataset.

Definition 14. [Knowledge-preserving watermarking.] Given two datasets $D_O$ and $D_W$, the watermarking scheme will be knowledge-preserving if and only if learning statistics $C_{S_O} = C_{S_W}$, that is:

$$C_{S_{L_{loss}}} = 0$$

(22)
The knowledge-preserving and lossless usability constraints model is derived from Theorem 1, Lemma 1, and Lemma 2 as follows.

**Definition 15. [Knowledge-preserving and lossless usability constraints model.]** The usability constraints model $\mathcal{h}$ for knowledge-preserving and lossless watermarking of data mining datasets is a tuple meeting following constraints:

$$Y_{D_O} = Y_{D_W}; \ C_{P_{DO}} = C_{P_{DW}}; \ and \ \Omega_{D_O} = \Omega_{D_W} \tag{23}$$

The second constraint in (23) needs to be satisfied by using local constraints (Definition 9) and global constraints (Definition 10); as a result, the output of feature selection schemes must be same for both datasets $D_O$ and $D_W$.

**Theorem 2.** If a watermarking scheme meets the constraints in $\mathcal{h}$ only then it is defined as knowledge-preserving and lossless.

**Proof:** If $C_{SO}$ and $C_{SW}$ are the learning statistics obtained by applying a learning algorithm $\Gamma$ on original dataset $D_O$, and watermarked dataset $D_W$ respectively.

Since, class labels, classification potentials and data distributions are preserved during watermarking; so according to Theorem 1, Lemma 1, and Lemma 2 the statistics $C_{SO}$ and $C_{SW}$ would be same, that is:

$$C_{SD_O} = C_{SD_W} \tag{24}$$

But the learning statistics tuple $C_{SD_O}$ is formulated as:

$$C_{SD_O} = (TP_{rate_O}, FP_{rate_O}, \mathcal{R}_{b_O}) \tag{25}$$

and learning statistics tuple $C_{SD_W}$ as:

$$C_{SD_W} = (TP_{rate_W}, FP_{rate_W}, \mathcal{R}_{b_W}) \tag{26}$$

Now, according to definition of learning information loss:

$$C_{SL_{Loss}} = \frac{|C_S - C_{SW}|}{C_S} \times 100 \tag{27}$$

where

$$C_{SL_{Loss1}} = TP_{rate_O} - TP_{rate_W}; \ C_{SL_{Loss2}} = FP_{rate_O} - FP_{rate_W}; \ C_{SL_{Loss3}} = \mathcal{R}_{b_O} - \mathcal{R}_{b_W} \tag{28}$$

By substituting equations (24), (25), (26), and (28) in equation (27), we get:

$$C_{SL_{Loss}} = \frac{(0)(0)(0)}{C_S} \times 100 \quad C_{SL_{Loss}} = 0$$

To conclude, the usability constraints, defined for the process of watermarking, must hold this theorem to ensure knowledge-preserving and lossless watermarking. We now discuss watermark embedding scheme in the following.

**V. Watermarking Scheme**

We now describe our watermarking scheme – with its foundation in the above-mentioned formal model – that not only preserves the classification potential of features but also results in (approximately) zero information loss. There are two main phases in our watermarking scheme: watermark encoding and watermark decoding.
A. Watermarking Encoding

The steps involved in the watermark encoding phase are:

Step 1: The classification potential of each feature is calculated using mutual information $I$ (Definition 6) and it is stored in a vector $R_{nk}$. The threshold $C_{PT}$ (Definition 8) is computed using a vector of classification potentials. The classification potential of features (vector $R_{nk}$) and $C_{PT}$ are then used to logically group features of the dataset into $n$ non-overlapping groups $g_0, g_1, ..., g_{n-1}$.

Step 2: The watermark is optimized and embedded in this stage while enforcing the usability constraints modeled in Section IV.

The different steps of watermark encoding phase are shown in Figure 2.

![Fig. 2. Different steps of watermark encoding phase.](image)

1) Feature Ranking: In this step, the features are ranked to: (1) logically group the data into $n$ non-overlapping partitions; and (2) to define “usability constraints” in such a manner that the information loss is zero. The ranking is done, using a well known information measure, mutual information $I$, to understand the correlation of a feature on predicting a class label. The rank of all features, present in a dataset, are stored in a vector $R_{nk}$.

2) Classification potential threshold computation: Intuitively speaking, a feature that has a large classification potential is expected to tolerate only a small change (during embedding of watermark) in its values to ensure that the decision rules remain unaltered. In Figure 3, the tolerable alteration $\Delta$ in the value of a feature is plotted against its classification potential if the learning statistics were to be preserved as proved in [10]. The figure also concurs to our expectations: (1) the features with high classification potential can tolerate only small changes, if the information is to be preserved during watermarking; and (2) the top ranked features show approximately zero tolerance towards any change. Therefore, it is very important to compute the amount of change that a feature can tolerate during the watermarking process. The data groups are constructed using $C_{PT}$ and “tolerable alteration” is computed for each group.

![Fig. 3. Plot for $C_P$ versus $\Delta$.](image)

3) Data Grouping: As mentioned before, $C_{PT}$ is used to group the features into $n$ logical non-overlapping groups. The data grouping function is given in equation (29).
A sequence of binary bits is used as a watermark. For watermarking a non-numeric feature, because our watermark embedding algorithm does not bring any change in the values of such features. To embed a watermarking a group unaltered. As a result, we have to model the challenge as an optimization problem to ensure that under given constraints, the constraint number 2 into two types: (1) global constraints \( G \) for the whole dataset; and (2) local \( L_i \) for a particular logical group \( g_i \).

**Local Usability Constraints.** The local usability constraints (Definition 9) are defined by mutual information (see equation (15)). In order to enforce them, the constraint 2 of “knowledge-preserving and lossless usability constraints model” must be met. In our formal model, we preserve the information by trying to keep the data distribution (as much as possible) unaltered. As a result, we have to model the challenge as an optimization problem to ensure that under given constraints, the “tolerable alteration” is maximized for all features in general and minimized for high ranking features within each group while watermarking a group \( g_i \).

**Global Usability Constraints.** In global usability constraints \( G \) (Definition 10), we have to ensure that the predictive ability of each feature – calculated by five well known feature selection schemes (\( IG, IG_s, CFS, CBF \), and \( PC \)) remains preserved, that is \( (R_{IG}^m = R_{IG}^m, R_{IG_s}^m = R_{IG_s}^m, R_{CFS}^m = R_{CFS}^m, R_{CBF}^m = R_{CBF}^m) \), and \( R_{PC}^m = R_{PC}^m \).

5) **Selecting Data for Watermarking:** An important step in watermarking of a dataset is to select relevant rows in which the watermark will be inserted. In this paper, we use a parameter \( \zeta \) to store information about such rows. Its main purpose is to insert the watermark in the rows of the original data \( D_O \) in such a way that local and global usability constraints are not violated.

6) **Watermark Embedding:** Watermarking non-numeric features. Data grouping step is not performed for non-numeric features because our watermark embedding algorithm does not bring any change in the values of such features. To embed a watermark in the dataset, a sequence of binary bits is used as a watermark. For watermarking a non-numeric feature \( f \), secret hash value for each row is calculated by seeding a pseudo random sequence generator \( \varnothing \) with concatenation of a secret key \( K_s \), class label of the row, and row value (ascii) using equation (31)

\[
row.hash = \varnothing(K_s||y||row.val)
\]

where, \( row.val \) denotes a particular row value and \( y \) is the class label of that row.

The secret ordering of rows is computed using the watermark bits as depicted in Algorithm 1. This secret order does not bring any change in the underlying dataset. If a row value is repeated with the same class label then the same hash value would always be generated; consequently such rows are logically placed in the same cluster. The secret order of hash values after embedding the final bit is stored to use it during the watermark decoding stage.

**Watermarking numeric features.** We model the maximizing “tolerable alteration” as a constrained optimization problem subject to certain constraints on “tolerable alterations”. We have significantly enhanced the information preserving watermarking technique of [10], using our formal model, to make it a generalized usability constraints modeling and knowledge-preserving watermarking scheme.

The lower and upper bound (denoted by \( \Delta^f_{\text{min}} \) and \( \Delta^f_{\text{max}} \)) for a feature \( f \) in the \( i^{th} \) group, are calculated using equations (32) and (33) respectively to define “tolerable alterations” for a data element.

\[
\Delta^f_{\text{min}} = \frac{\left(\frac{1}{1+\text{Rnk}_f}\right) \times \left(\frac{1}{1+\text{Pc}_f}\right)}{2^{2^i}} \tag{32}
\]

\[
\Delta^f_{\text{max}} = \frac{\left(\frac{1}{1+\text{Rnk}_f}\right) \times \left(\frac{1}{1+\text{Pc}_f}\right)}{2^{2^i}} \tag{33}
\]
where, $Rnk^f$ is the classification potential of the feature $f$ and $\gamma$ is the secret grouping parameter. Note that the relations for computing $\Delta^f_{\min}$ and $\Delta^f_{\max}$ are significantly different from the relations used in [10]. We incorporate the secret parameter $\gamma$ with the objective of having more watermark security. Also note that in the equations (32) and (33), 1 is added in denominator to avoid division by zero if a feature has zero classification potential.

In our approach, the usability constraints given in equation (23) are enforced while maximizing the allowable alterations $\Delta_i$ in a group $g_i$. This ensures that both local (within a group) as well as global (across the whole dataset) constraints are satisfied.

To conclude, we use equation (34) to define a new objective function: to maximize $\Delta_i$ under user defined “tolerable alterations” ($\Delta_{\min} \leq \Delta_i \leq \Delta_{\max}$).

The process of optimizing $\Delta_i$ follows two simultaneous steps: (1) In step 1, the constraints are verified locally for each logical group in an iterative manner; and (2) In step 2, the global usability constraints are also verified for the complete dataset. The objective function used for each type of optimization scheme is the same.

$$\max (g_i + \Delta_i)$$
$$\text{subject to}$$
$$h \text{ with } L_i \text{ and } G$$

Note that in our objective function $g_i$ is the part of data $D_O$; therefore, the only variable to be optimized is $\Delta_i$. As mentioned before, for the features having high classification potential, a very small “tolerable alteration” is allowed; therefore, the value of $\Delta_i$ – for the groups containing such features – should be zero subject to meeting the constraints mentioned in the objective function of equation (34). Therefore, if $\Delta_{\min}$ for such features violates any usability constraints then $\Delta_{\min}$ is set to zero for those features. This situation is also handled during the optimization process. In the following, we evaluate the suitability of three optimization schemes to optimize the objective function mentioned in equation (34).

Our next target is to select an optimizer$^3$ that has the ability to locate the global optimum in the optimization search space in realtime. We have used two biological heuristic based optimizers – Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) – and one classical optimizer – Mixed Integer Nonlinear programming.

**Optimization with Particle Swarm Optimization Algorithm.** Particle swarm optimization (PSO) [12] is a stochastic technique developed for continuous optimization using social behavior of flocking birds. The solutions to the problem are modeled as particles. PSO has been quite efficiently and effectively implemented for function optimization, artificial neural network training, fuzzy system control, and other areas. Moreover, it is proven to be better in many respects from evolutionary algorithms like genetic algorithms, memetic algorithms, ant-colony systems, and shuffled frog leaping [13].

In our implementation of PSO, we map the statistics contained in $\Delta_i$ using bit string having length $l$ (the length of the watermark) as depicted in Figure 4. As our particle consists of 0 and 1 bits; therefore, our technique performs the different data alterations for each distinct bit. The used fitness function is given in equation (34).

**Optimization with Genetic Algorithm.** Genetic algorithm (GA) is inspired from the principles of biological evolution: survival of the fittest. The populations of candidate solutions – called chromosomes – compete to evolve to become better solutions. GA is heuristic based method and was introduced by John Holland [14].

We have used GA to create the watermark (chromosome) for embedding it into the partitioned dataset. In our implementation of GA, the chromosome consists of mapping of feasible $\Delta_i$s into a bit string consisting of 0s and 1s having length $l$. It uses

$^3$We use the watermark optimization step for numeric features only because $\Delta$ is not required for non-numeric features in the proposed technique.
the same fitness function as that of PSO given in equation (34). The representation of the chromosome (watermark) is given in Figure 5.

When a chromosome violates the usability constraints, it is penalized by decreasing its fitness value. The penalty function is given as:

$$
\partial_{\text{new}} = \partial_{\text{old}} \cdot \frac{1}{1 + \vartheta}
$$

where $\partial_{\text{new}}$ and $\partial_{\text{old}}$ are the new and old fitness values of the chromosome respectively, and $\vartheta$ is the number of violations of the usability constraints. If no violations occur for a chromosome, the fitness value (according to equation (35)) of the chromosome remains unchanged. Moreover, as the value of $\vartheta$ increases the fitness of chromosome decreases. The feasible solution set $\rho$ contains all the chromosomes which satisfy the conditions specified in the objective function. Our chromosome representation consists of 0 and 1 bits so here again, our technique performs the different data alterations for each distinct bit. The algorithm stops once the termination criteria is met: (1) when a chromosome is found with the maximum allowable value of $\Delta_i$; or (2) $\Delta_i$ does not further increase in a predefined number of iterations; or (3) when the predefined number of generations have been evaluated.

**Optimization with Mixed Integer Nonlinear Programming.** Mixed Integer Nonlinear Programming (MINLP) is a mathematical programming technique that uses continuous and discrete variables, and nonlinear objective function and constraints. MINLPs have been applied for optimizing constrained problems with applications in engineering, finance, and other scientific problems. There are several different approaches for solving MINLPs including but not limited to: Outer Approximation (OA) methods, Branch-and-Bound (B&B), Extended Cutting Plane methods, and Generalized Benders Decomposition (GBD) [15]. The motivation to use MINLP in our scheme came from its use in designing algorithms for constrained combinatorial non-linear mathematical problems in various disciplines. In this paper, we work with the OA method proposed by Fletcher et al. in [16]. OA divides MINLP into NLP subproblem and a master Mixed Integer Program (MIP). An interested reader is referred to [17] for a survey of different MINLP techniques and their possible applications.

We have used MINLP implemented in LINDO [18] software for optimizing our non linear objective function given in equation (34). We have incorporated our “usability constraint model” in LINDO.

The optimum value of $\Delta_i$ for a particular data group $g_i$ is embedded in each row of the group subject to usability constraints as specified in the objective function given above. The numeric features in a group $g_i$ are marked with positive $\Delta_i$, if a watermark bit $b$ is 1; and with negative $\Delta_i$ if a watermark bit $b$ is 0. Algorithm 2 lists the steps of watermark embedding in numeric features.

7) **Watermark Security:** As discussed before, the current scheme further improves on the robustness level of our earlier technique [10] by using the data-grouping strategy. For brevity, we compute the probability of successfully attacking one watermark bit only. Let $Prob$ be the probability of successfully attacking one row of the data having $n$ logical groups. Since, the attacker is not aware of the secret parameters that were used during the data-grouping stage; therefore, he cannot intentionally target a particular group for launching different types of attacks. As a result, he will have to select a random feature to launch his attacks. In this case, the probability of successfully corrupting (or deleting) a watermark bit is $(0.5)^n$ because different watermark is inserted in different groups. Since, we are watermarking all rows in the data and also using the majority voting as an error correction measure; therefore, the attacker has to target at least half the total number of rows to achieve his objective. If there are $N$ rows in the dataset, the probability of successfully corrupting a watermark bit is $((0.5)^n)^{\frac{N}{2}}$. For large datasets and large values of $n$, this probability becomes significantly small. Consider, for example, a data has 100 rows and $n = 4$. The probability of successful launching an attack on this data is:
**Algorithm 2 Watermark numeric features**

<table>
<thead>
<tr>
<th>Input:</th>
<th>$D_O, W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>$D_W, \Delta$</td>
</tr>
</tbody>
</table>

construct data groups $g_n$ using $D_O$

for each bit $b$ in $W$ do

for each group $g_i$ in $g_n$ do

for each row $r_i$ in $g_i$ do

if $b == 1$ then

$D_W(r_i) \leftarrow g_i(r_i) + \Delta_i(r_i)$ subject to $L_i$ and $G$

else

$D_W(r_i) \leftarrow g_i(r_i) + (-\Delta_i(r_i))$ subject to $L_i$ and $G$

end if

end for

end for

end for

return $D_W, \Delta$

---

$Prob = ((0.5)^4)^{50} = (0.0625)^{50} = 6.22 \times 10^{-61}$

This probability in our earlier technique \cite{10} is $8.88 \times 10^{-16}$; therefore, the new technique has significantly reduced the probability of deleting a watermark from the dataset.

**B. Watermark Decoding**

The architecture of the watermark decoding phase is shown in Figure 6.

![Diagram of Watermark Decoding](image)

**Fig. 6.** Steps of watermark decoding phase.

1) **Watermark decoding from non-numeric features:** In the watermark detection phase, the hash value of a feature for each row is calculated using the same steps of watermark embedding. The secret ordering, on the basis of this hash value is calculated by comparing it with the ordering based on the hash values in $temp$ and the embedded bit is decoded from this ordering. It is important to mention here that the sole purpose of using $temp$ for ordering the rows during watermark decoding stage is to combat attacks which may change the order of rows in the watermarked data. Since, the $temp$ contains the ordering after embedding the last watermark bit $b$; therefore, the last embedded bit is decoded first. The steps of this phase are listed in Algorithm 3 that are repeated for each bit of watermark $W$.

2) **Watermark decoding from numeric features:** To decode the watermark from numeric features, a decoding threshold $T^*$ value is calculated for each group by using the same method as in \cite{10}. For decoding a watermark from a numeric feature in a group $g_{iw}$, a parameter $val$ is computed from the watermarked dataset $D_W$ as:

$$val = \Delta_i * g_{iw}$$  \hspace{1cm} (36)$$

The value of the parameter $val$ is compared with the decoding threshold $T^*$ and if $val$ is found to be greater than $T^*$, the watermark bit $b$ is decoded as 1; otherwise it is decoded as 0. Here, as an error correction mechanism, a majority voting step is performed after decoding a watermark bit $b$ from all the rows. The steps of this phase are shown in Algorithm 4.

**VI. EXPERIMENTS AND RESULTS**

We have performed our experiments on 25 different datasets\textsuperscript{4}. These biomedical and biomedicine datasets are carefully chosen from different domains so that we test our technique for two class datasets, multi-class datasets, high dimensional datasets, datasets with missing values, datasets with various type of features, imbalanced datasets, and datasets with large

\textsuperscript{4}These datasets are available online at [www.ics.uci.edu/mllearn/MLRepository.html](http://www.ics.uci.edu/mllearn/MLRepository.html) and [http://home.ccr.cancer.gov/ncifdaproteomics/patterns.asp](http://home.ccr.cancer.gov/ncifdaproteomics/patterns.asp)
Algorithm 3 Watermark decoding from non-numeric features

<table>
<thead>
<tr>
<th>Input: $D_W, K_s, temp$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $W'$</td>
</tr>
</tbody>
</table>


temp' ← $D_W$

for each row in $D_W$ do
  row.hash = $\overline{\bar{\otimes}}(K_s||y||row.val)$
  temp'(row) = $D_W(row.hash)$
end for

compute secret order using row.hash
upate the order of rows in temp' is according to their order in temp

for $j = \text{length of } W$ to 1 do
  if the rows are sorted in the descending order of hash values then
    $b = 1$
  else
    $b = 0$
  end if
  if $b == 1$ then
    sort the rows in the ascending order of hash values
  else
    sort the rows in the descending order of hash values
  end if
  $W' ← b$
end for

return $W'$

Algorithm 4 Watermark decoding from numeric features

<table>
<thead>
<tr>
<th>Input: $D_W, T^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Decoded watermark $W'$</td>
</tr>
</tbody>
</table>

construct data groups $g_n$ using $D_W$

for $j = \text{length of } W$ to 1 do
  for each group $g_i$ in $g_n$ do
    compute $val$ using equation (36)
    if $val > T^*$ then
      $b = 1$
    else
      $b = 0$
    end if
  end for
  perform majority voting to finally decode $b$
  $W' ← b$
end for

return $W'$

number of instances. We do not report the results of our study on the robustness of proposed scheme in this paper because this is done elsewhere in [10] but our current approach further improves the robustness level using the data grouping strategy discussed in Section V.

The experiments were carried out on a computer with a 1.73 Core 2 processor and a 1 GB of RAM. We set the value of $\zeta = 100\%$, which means that all rows were selected for watermarking. A data owner can choose any watermark length but we set its length to $l = 16$ bits. Five well known machine learning schemes are used to analyze their learning statistics on both original and altered datasets to show the relevance of Theorem 2 in the proposed scheme.

A. Working with non-numeric features

The example run of watermark encoding and decoding for non-numeric features is reported in [19].

For brevity, we report the test run of watermark embedding and decoding algorithms on non-numeric features in only six rows of the Protein dataset in Figure 7.

Since the proposed watermark embedding algorithm does not bring any change in the data; therefore, the results of feature selection schemes and all the learning statistics will definitely be the same for original and watermarked data. Hence, our model of Definition 15 is validated and the proposed watermarking scheme for non-numeric datasets is knowledge-preserving & lossless (Theorem 2).

B. Working with numeric features

For numeric features, we have used all 3 optimization schemes on $24^5$ datasets for optimizing the value of $\Delta$. We have used two metrics to benchmark optimized techniques: (1) the time taken to find the optimum $\Delta$; and (2) the optimized value of $\Delta$ that models the quality of the solution. The output of the optimization techniques – utilizing our formal model – is given as an input to the above-mentioned watermarking scheme. It is important to emphasize here that the proposed usability constraints

\footnote{We omitted Protein dataset from this experiment because it does not have any numeric feature.}
Fig. 7. Watermark encoding and decoding process for non-numeric features. Note that in the watermark embedding process, if a watermark bit $b$ is 1 then the rows of the dataset are sorted in the descending order of hash values and if $b$ is 0 then the rows are sorted in the ascending order. In comparison, during watermark decoding process, if rows are sorted in the descending order then the bit $b$ is decoded as 1 and the order of rows is updated in the ascending order of hash values; otherwise, $b$ is decoded as 0 and the rows are sorted in the descending order of hash values.

The model and watermarking scheme has enhanced watermark security of [10] by using the secret parameter $\gamma$ for computing $C_{P\gamma}$ and upper and lower bounds of $\Delta$s. Furthermore, embedding the watermark in high ranking features further increases the watermark security because now an attacker cannot target only low ranked features to remove the watermark from the dataset. Moreover, an attacker cannot delete the high ranking features because doing so will make the data useless for the purpose of extracting knowledge by mining datasets.

The time taken (in seconds) by each optimization technique is tabulated in Table I. All three techniques took relatively
We have conducted experiments on all the 24 datasets listed in Table I. For brevity, we have chosen a relatively small dataset – PimaIndiansDiabetes – with 8 features and a class label to build insights about the proposed technique. We report the results of data grouping on only one dataset pima – diabetes with an average mutual information, $Avg = 0.1338$, grouping parameter $=0.3$, and threshold $C_{pg} = 0.04014$ in Table II.

The optimum value of $\Delta$ for each feature in the selected dataset is depicted in Figure 8. It is evident from the figure that the optimum value of $\Delta$ for features that have relatively high classification potentials is smaller and vice versa. Moreover, it is clear from the Figure 8 that MINLP stuck at the local optima in most of the cases, whereas PSO and GA have been relatively more successful in moving closer to the global optimum for almost every feature. We have chosen PSO because it gives nearly optimum solution but in less time.

For brevity, we report a test run of watermark embedding on numeric features in only four rows of the dataset shown in Table III.

The steps of watermark embedding (with $W = 101$) process for numeric features are tabulated in Table IV. In this example, $\Delta_f1 = 1.3\%$, $\Delta_f2=0.0000054\%$, $\Delta_f3 = 1.5\%$, $\Delta_f4 = 0.06\%$, $\Delta_f5 = 0.000001\%$, $\Delta_f6 = 0.1\%$, $\Delta_f7 = 0.00049\%$, $\Delta_f8=0.001\%$ as determined by our implementation of PSO algorithm.
TABLE IV

Watermark embedding process for numeric features with a watermark $W = 101$. The numeric features in a group $g_i$ are added with positive $\Delta_i$, if a watermark bit $b$ is 1; and with negative $\Delta_i$, if a watermark bit $b$ is 0.

<table>
<thead>
<tr>
<th>$W$</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>$T_9$</th>
<th>$T_{10}$</th>
<th>$T_{11}$</th>
<th>$T_{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b=1$</td>
<td>6.078</td>
<td>148</td>
<td>73.08</td>
<td>35.021</td>
<td>0</td>
<td>33.634</td>
<td>0.627</td>
<td>50.001</td>
<td>1</td>
<td>0.078</td>
<td>0.00000799</td>
<td>1.08</td>
</tr>
<tr>
<td>1.013</td>
<td>85</td>
<td>66.99</td>
<td>29.017</td>
<td>0</td>
<td>26.627</td>
<td>0.351</td>
<td>31</td>
<td>0</td>
<td>0.013</td>
<td>0.00000459</td>
<td>0.99</td>
<td>0.0174</td>
</tr>
<tr>
<td>8.105</td>
<td>183</td>
<td>64.96</td>
<td>0</td>
<td>2</td>
<td>23.325</td>
<td>0.672</td>
<td>32</td>
<td>1</td>
<td>0.104</td>
<td>0.00000988</td>
<td>0.96</td>
<td>0</td>
</tr>
<tr>
<td>1.013</td>
<td>89</td>
<td>66.99</td>
<td>23.014</td>
<td>94</td>
<td>28.126</td>
<td>0.167</td>
<td>21</td>
<td>0</td>
<td>0.013</td>
<td>0.00000481</td>
<td>0.99</td>
<td>0.018</td>
</tr>
<tr>
<td>$b=0$</td>
<td>5.999</td>
<td>148</td>
<td>71.984</td>
<td>35</td>
<td>0</td>
<td>33.6</td>
<td>0.627</td>
<td>50</td>
<td>1</td>
<td>0.079014</td>
<td>0.00000799</td>
<td>1.0962</td>
</tr>
<tr>
<td>1</td>
<td>85</td>
<td>65.985</td>
<td>29</td>
<td>0</td>
<td>26.6</td>
<td>0.351</td>
<td>31</td>
<td>0</td>
<td>0.013169</td>
<td>0.00000459</td>
<td>1.0085</td>
<td>0.01741</td>
</tr>
<tr>
<td>7.999</td>
<td>183</td>
<td>63.966</td>
<td>0</td>
<td>0</td>
<td>23.35</td>
<td>0.672</td>
<td>32</td>
<td>1</td>
<td>0.105552</td>
<td>0.00000988</td>
<td>0.94</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>89</td>
<td>63.985</td>
<td>23</td>
<td>94</td>
<td>28.1</td>
<td>0.167</td>
<td>21</td>
<td>0</td>
<td>0.013169</td>
<td>0.00000481</td>
<td>1.0085</td>
<td>0.01808</td>
</tr>
<tr>
<td>$b=1$</td>
<td>6.077</td>
<td>148</td>
<td>73.064</td>
<td>35.021</td>
<td>0</td>
<td>33.634</td>
<td>0.627</td>
<td>50.001</td>
<td>1</td>
<td>0.077987</td>
<td>0.00000799</td>
<td>1.07976</td>
</tr>
<tr>
<td>1.013</td>
<td>85</td>
<td>66.975</td>
<td>29.017</td>
<td>0</td>
<td>26.627</td>
<td>0.351</td>
<td>31</td>
<td>0</td>
<td>0.013</td>
<td>0.00000459</td>
<td>0.988753</td>
<td>0.0174</td>
</tr>
<tr>
<td>8.105</td>
<td>183</td>
<td>64.96</td>
<td>0</td>
<td>2</td>
<td>23.325</td>
<td>0.672</td>
<td>32</td>
<td>1</td>
<td>0.103987</td>
<td>0.00000988</td>
<td>0.95979</td>
<td>0</td>
</tr>
<tr>
<td>1.013</td>
<td>89</td>
<td>66.99</td>
<td>23.014</td>
<td>94</td>
<td>28.126</td>
<td>0.167</td>
<td>21</td>
<td>0</td>
<td>0.013</td>
<td>0.00000461</td>
<td>0.988753</td>
<td>0.018</td>
</tr>
</tbody>
</table>

| $b=0$ | 5.999 | 148 | 71.984 | 35 | 0 | 33.6 | 0.627 | 50 | 1 | 0.079014 | 0.00000799 | 1.0962 | 0.021013 | 0 |
| 1 | 85 | 65.985 | 29 | 0 | 26.6 | 0.351 | 31 | 0 | 0.013169 | 0.00000459 | 1.0085 | 0.01741 | 0 |
| 7.999 | 183 | 63.966 | 0 | 0 | 23.35 | 0.672 | 32 | 1 | 0.105552 | 0.00000988 | 0.94 | 0 | 0 |
| 1 | 89 | 63.985 | 23 | 94 | 28.1 | 0.167 | 21 | 0 | 0.013169 | 0.00000481 | 1.0085 | 0.01808 | 0.000994 |
| $b=1$ | 6.077 | 148 | 73.064 | 35.021 | 0 | 33.634 | 0.627 | 50.001 | 1 | 0.077987 | 0.00000799 | 1.07976 | 0.021 | 0 |
| 1.013 | 85 | 66.975 | 29.017 | 0 | 26.627 | 0.351 | 31 | 0 | 0.013 | 0.00000459 | 0.988753 | 0.0174 | 0 |
| 8.105 | 183 | 64.96 | 0 | 2 | 23.325 | 0.672 | 32 | 1 | 0.103987 | 0.00000988 | 0.95979 | 0 | 0 |
| 1.013 | 89 | 66.99 | 23.014 | 94 | 28.126 | 0.167 | 21 | 0 | 0.013 | 0.00000461 | 0.988753 | 0.018 | 0.000994 |
It is important to mention here that the any constraints on the data type are strictly enforced during the watermark embedding process. For instance, if a feature in the original dataset does not contain a floating point number then the watermarked data should also possess the same properties of that feature. We have reported our results in Tables IV, V, and VI for an illustrative purpose only. Moreover, the rows having zero values are also left unmarked. After embedding all the watermark bits, the final data takes the form as shown in Table V.

**TABLE V**

**W**AT**E**R**M**ARK**E**D** D**ATA.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.077</td>
<td>138.000</td>
<td>73.064</td>
<td>35.021</td>
<td>0.000</td>
<td>33.634</td>
<td>0.627</td>
<td>50.001</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1.013</td>
<td>85.000</td>
<td>66.975</td>
<td>29.017</td>
<td>0.000</td>
<td>26.627</td>
<td>0.351</td>
<td>31.000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8.103</td>
<td>183.000</td>
<td>64.946</td>
<td>0.000</td>
<td>0.000</td>
<td>23.323</td>
<td>0.672</td>
<td>32.000</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1.013</td>
<td>89.000</td>
<td>66.975</td>
<td>23.014</td>
<td>94.000</td>
<td>28.128</td>
<td>0.167</td>
<td>21.000</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

The steps of watermark decoding from the watermarked data are reported in Table VI. Note that the watermark is not decoded from the features f2, f5, f7, and f8 because watermark embedding did not modify their values. The reason is that either have very high classification potential or the value of $\Delta$ for them was approximately zero.

**TABLE VI**

**W**AT**E**R**M**ARK**E**D** D**EC**O**D**I**N**G** P**RO**C**E**S**S** F**O**R** N**U**M**E**R**I**C** F**E**AT**U**R**E**S. **T**HE **V**A**L**U**E** **O**F **T**HE **P**A**R**A**M**E**T**E**R val **F**O**R** A **F**E**AT**U**R**E **I**S **C**O**M**PA**R**E**D **W**I**T**H **T**HE **D**E**C**O**D**I**N**G **T**H**R**E**S**H**O**L**D **T**∗ (S**E**E **T**A**B**LE **IV) **F**O**R** T**H**AT **F**E**AT**U**R**E **A**N**D **I**F **val **I**S **F**O**U**N**D **T**O **B**E **G**R**E**AT**E**R **T**H**A**N **T**∗, **T**HE W**A**T**E**R**M**ARK **B**I**T** b **I**S **D**E**C**O**D**E**D **A**S **1; **OT**H**E**R**W**I**S**E, **I**T **I**S **D**E**C**O**D**E**D **A**S **0. **T**H**E **W**A**T**E**R**M**ARK **B**I**T**S **S**H**O**W**N **I**N **B**O**L**D **F**A**C**E **A**R**E** T**H**E **R**E**S**U**L**T **O**F **M**A**J**O**R**I**T**Y **V**O**T**I**N**G **T**H**A**T **I**S **U**S**E**D **A**S **A**N **E**R**R**O**R **C**O**R**R**E**C**T**I**O**N **M**E**C**H**A**N**I**S**M **A**F**T**E**R **D**E**C**O**D**I**N**G **A **W**A**T**E**R**M**ARK **B**I**T** b **F**R**O**M **A**L**L **T**H**E **R**O**W**S **O**F **T**H**E **D**A**T**A **S**E**T. **T**H**E **F**I**N**A**L **R**O**W **I**N **T**H**I**S **T**A**B**L**E **S**H**O**W**S **T**H**E **D**E**C**O**D**E**D **W**A**T**E**R**M**ARK **F**R**O**M **E**A**C**H **O**F **T**H**E **M**A**R**K**E**D **F**E**AT**U**R**E**S.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Decoded bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.077</td>
<td>33.064</td>
<td>1</td>
<td>0.079</td>
<td>1.09634</td>
<td>0.02101</td>
<td>0.0363</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.013</td>
<td>66.998</td>
<td>29.017</td>
<td>26.627</td>
<td>0</td>
<td>0.01317</td>
<td>1.00497</td>
<td>0.01741</td>
<td>0.02663</td>
</tr>
<tr>
<td>8.103</td>
<td>64.968</td>
<td>0.000</td>
<td>23.323</td>
<td>1</td>
<td>0.10534</td>
<td>0.97452</td>
<td>0</td>
<td>0.02332</td>
</tr>
<tr>
<td>1.013</td>
<td>66.998</td>
<td>23.014</td>
<td>28.128</td>
<td>0</td>
<td>0.01317</td>
<td>1.00497</td>
<td>0.01381</td>
<td>0.02813</td>
</tr>
<tr>
<td>Majority voting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.077</td>
<td>33.064</td>
<td>1</td>
<td>0.079</td>
<td>1.09596</td>
<td>0.02101</td>
<td>0.0363</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.013</td>
<td>66.975</td>
<td>29.017</td>
<td>26.627</td>
<td>0</td>
<td>0.01317</td>
<td>1.00463</td>
<td>0.01741</td>
<td>0.02663</td>
</tr>
<tr>
<td>8.103</td>
<td>64.946</td>
<td>0.000</td>
<td>23.323</td>
<td>1</td>
<td>0.10537</td>
<td>0.97419</td>
<td>0</td>
<td>0.02332</td>
</tr>
<tr>
<td>1.013</td>
<td>66.975</td>
<td>23.014</td>
<td>28.128</td>
<td>0</td>
<td>0.01317</td>
<td>1.00463</td>
<td>0.01381</td>
<td>0.02813</td>
</tr>
<tr>
<td>Majority voting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.077</td>
<td>33.064</td>
<td>1</td>
<td>0.079</td>
<td>1.09634</td>
<td>0.02101</td>
<td>0.0363</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.013</td>
<td>66.998</td>
<td>29.017</td>
<td>26.627</td>
<td>0</td>
<td>0.01317</td>
<td>1.00497</td>
<td>0.01741</td>
<td>0.02663</td>
</tr>
<tr>
<td>8.103</td>
<td>64.924</td>
<td>0.000</td>
<td>23.323</td>
<td>1</td>
<td>0.10539</td>
<td>0.97386</td>
<td>0</td>
<td>0.02332</td>
</tr>
<tr>
<td>1.013</td>
<td>66.998</td>
<td>23.014</td>
<td>28.128</td>
<td>0</td>
<td>0.01317</td>
<td>1.00497</td>
<td>0.01381</td>
<td>0.02813</td>
</tr>
<tr>
<td>Majority voting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decoded Watermarks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>101</td>
</tr>
</tbody>
</table>

After applying the majority voting step on all the decoded bits, we get the bit sequence “101” which is same as the embedded watermark.

**C. Knowledge-preserving characteristic of proposed model**

We give the optimized “usability constraints”, derived from our formal model, to the enhanced watermark embedding algorithm. We report the effect of watermarking, with the proposed usability constraints model, on various feature selection schemes in Table VII (for brevity we report our result on only one dataset *pima – diabetes*). Recall that in Theorem 1, it is
proven that mutual information between a feature and the class attribute can only remain same for original and watermarked datasets, if the features belong to the same class in both datasets. To see the impact of this theorem, we report mutual information of original and watermarked features in Table VII to prove that the features’ values of the watermarked dataset have the same relation with the class labels as they had before watermark embedding. It is evident from Table VII that learning statistics of all feature selection schemes have also been preserved by enforcing “usability constraints” of our formal model.

TABLE VII

**Effect of watermarking, with proposed usability constraints model, on various feature selection schemes. In this table F1 to F8 are the dataset features and S denotes feature selection schemes. In the columns for CFS and CBF the entry Yes indicates that feature was selected by S and No means that the feature was not selected by S.**

<table>
<thead>
<tr>
<th>S</th>
<th>Original data</th>
<th>Watermarked data</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>TG</td>
<td>0.0392</td>
</tr>
<tr>
<td>0.0302</td>
<td>0.0302</td>
<td></td>
</tr>
<tr>
<td>0.0593</td>
<td>0.0593</td>
<td></td>
</tr>
<tr>
<td>0.0817</td>
<td>0.0817</td>
<td></td>
</tr>
<tr>
<td>0.2771</td>
<td>0.2771</td>
<td></td>
</tr>
<tr>
<td>0.1257</td>
<td>0.1257</td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>0.1409</td>
<td>0.1409</td>
<td></td>
</tr>
<tr>
<td>TGw.</td>
<td>TGw.</td>
<td>0.0315</td>
</tr>
<tr>
<td>0.0306</td>
<td>0.0306</td>
<td></td>
</tr>
<tr>
<td>0.0596</td>
<td>0.0596</td>
<td></td>
</tr>
<tr>
<td>0.0814</td>
<td>0.0814</td>
<td></td>
</tr>
<tr>
<td>0.0244</td>
<td>0.0244</td>
<td></td>
</tr>
<tr>
<td>0.0384</td>
<td>0.0384</td>
<td></td>
</tr>
<tr>
<td>0.0863</td>
<td>0.0863</td>
<td></td>
</tr>
<tr>
<td>0.0266</td>
<td>0.0266</td>
<td></td>
</tr>
<tr>
<td>0.0726</td>
<td>0.0726</td>
<td></td>
</tr>
<tr>
<td>CFS</td>
<td>CFS</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>PC</td>
<td>0.7382</td>
</tr>
<tr>
<td>0.5218</td>
<td>0.5218</td>
<td></td>
</tr>
<tr>
<td>0.3931</td>
<td>0.3931</td>
<td></td>
</tr>
<tr>
<td>0.2837</td>
<td>0.2837</td>
<td></td>
</tr>
<tr>
<td>0.1884</td>
<td>0.1884</td>
<td></td>
</tr>
<tr>
<td>0.103</td>
<td>0.103</td>
<td></td>
</tr>
<tr>
<td>0.0506</td>
<td>0.0506</td>
<td></td>
</tr>
</tbody>
</table>

Once the watermark is inserted in a dataset, we classify – using five well known machine learning algorithms – all 25 original datasets and their corresponding 24 watermarked datasets. We compare the change in two learning statistics: \(TP_{Rate}\) and \(FP_{Rate}\). The learning statistics \((TP_{Rate} \text{ and } FP_{Rate})\) are preserved for all learning algorithms with the exception of JRip – its \(TP_{Rate}\) and \(FP_{Rate}\) for some watermarked datasets vary in between \(\pm 0.001\) which is negligible. But we can safely generalize that the overall learning statistics – \(TP_{Rate}\) and \(FP_{Rate}\) – are preserved by enforcing the usability constraints of our formal model. Table VIII shows the learning statistics of original and watermarked pima – diabetes dataset (the results of other datasets are skipped for brevity but they also show the same pattern.).

**TABLE VIII**

**Effect of watermarking, with proposed usability constraints model, on \(TP_{Rate}\) and \(FP_{Rate}\) obtained by running the learning algorithm for classifying original dataset and watermarked dataset. In this table, \(\Delta TP\) and \(\Delta FP\) are the difference in \(TP_{Rate}\) and \(FP_{Rate}\) for \(DO\) (original data) and \(DW\) (watermarked data) respectively.**

<table>
<thead>
<tr>
<th>S</th>
<th>J48</th>
<th>CBF</th>
<th>SM</th>
<th>IBk</th>
<th>JRip</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TP_{Rate})</td>
<td>(DO)</td>
<td>(DW)</td>
<td>(\Delta TP)</td>
<td>(DO)</td>
<td>(DW)</td>
</tr>
<tr>
<td>I</td>
<td>0.738</td>
<td>0.738</td>
<td>0</td>
<td>0.737</td>
<td>0.737</td>
</tr>
<tr>
<td>TG</td>
<td>0.735</td>
<td>0.735</td>
<td>0</td>
<td>0.734</td>
<td>0.734</td>
</tr>
<tr>
<td>TGw.</td>
<td>0.738</td>
<td>0.738</td>
<td>0</td>
<td>0.737</td>
<td>0.737</td>
</tr>
<tr>
<td>CFS</td>
<td>0.749</td>
<td>0.749</td>
<td>0</td>
<td>0.748</td>
<td>0.748</td>
</tr>
<tr>
<td>CBF</td>
<td>0.738</td>
<td>0.738</td>
<td>0</td>
<td>0.737</td>
<td>0.737</td>
</tr>
<tr>
<td>PC</td>
<td>0.719</td>
<td>0.719</td>
<td>0</td>
<td>0.718</td>
<td>0.718</td>
</tr>
</tbody>
</table>

We show in Figure 9 that the rule boundaries \(R_0\) – the third element of learning statistic – are also preserved. For brevity, we show in Figure 9 the classification rules, extracted by J48 only, for the original and watermarked pima – diabetes datasets. It is evident that the rules to predict the class labels have remained unchanged.

To conclude, we have empirically proven that by enforcing our model of Definition 15 the watermarking scheme is knowledge-preserving & lossless (Theorem 2).

**VII. CONCLUSION**

In this paper, a novel knowledge-preserving and lossless usability constraints model has been proposed for watermarking data mining datasets. The proposed model has been given as an input to a new generic watermarking scheme for data mining
datasets. The benefits of this technique are: (1) identifying the vital characteristics of a dataset which need to be preserved during watermarking; (2) ranking the features on the basis of their classification potentials; (3) logically grouping the data into different groups (clusters) based on this ranking for defining local usability constraints for each group; (4) defining global usability constraints for the complete dataset; (5) modeling the local and global usability constraints in such a manner so that the learning statistics of a classifiers are preserved; (6) optimizing the watermark embedding such that all usability constraints remain intact. To the best of our knowledge, no technique in the literature exists that automatically computes “usability constraints” for a dataset that once enforced would preserve the knowledge contained in it. Moreover, the enhanced watermarking scheme can work with any type of data: numeric, non-numeric and strings. The proposed technique can be easily employed by the customers of companies like Kaggle to share datasets with data-mining experts by safeguarding and protecting their ownership. The technique, in a future work, could be extended to images that contain sensitive information (medical and surveillance) images.

REFERENCES


