Constrained Dynamic Rule Induction Learning

Fadi Thabtah[a], Issa Qabajeh[b], Francisco Chiclana[c]
a. Applied Business and Computing, NMIT, Auckland, New Zealand
b. School of Computer Sciences and Informatics, De Montfort University, Leicester, UK
c Centre for Computational Intelligence, Faculty of Technology, De Montfort University, Leicester, UK

Abstract

One of the known classification approaches in data mining is rule induction (RI). RI algorithms such as PRISM usually produce If-Then classifiers, which have a comparable predictive performance to other traditional classification approaches such as decision trees and associative classification. Hence, these classifiers are favourable for carrying out decisions by users and hence they can be utilised as decision making tools. Nevertheless, RI methods, including PRISM and its successors, suffer from a number of drawbacks primarily the large number of rules derived. This can be a burden especially when the input data is largely dimensional. Therefore, pruning unnecessary rules becomes essential for the success of this type of classifiers. This article proposes a new RI algorithm that reduces the search space for candidate rules by early pruning any irrelevant items during the process of building the classifier. Whenever a rule is generated, our algorithm updates the candidate items frequency to reflect the discarded data examples associated with the rules derived. This makes items frequency dynamic rather static and ensures that irrelevant rules are deleted in preliminary stages when they don't hold enough data representation. The major benefit will be a concise set of decision making rules that are easy to understand and controlled by the decision maker. The proposed algorithm has been implemented in WEKA (Waikato Environment for Knowledge Analysis) environment and hence it can now be utilised by different types of users such as managers, researchers, students and others. Experimental results using real data from the security domain as well as sixteen classification datasets from University of California Irvine (UCI) repository reveal that the proposed algorithm is competitive in regards to classification accuracy when compared to known RI algorithms. Moreover, the classifiers produced by our algorithm are smaller in size which increase their possible use in practical applications.

Keywords: Classification, Data Mfining, Prediction, PRISM, Rule Induction, Online Security

1. Introduction

Data mining, which is based on computing and mathematical sciences, is a common intelligent tool currently used by managers to perform key business decisions. Traditionally, data analysts used to spend a long time gathering data from multiple sources and little time was spent on analysis due to the limited computing resources. Though since the rapid development of computer networks and the hardware industry, analysts nowadays are spending more time on examining data, seeking useful concealed information. In fact, after the recent development of cloud computing, data collection, noise removal, data size and data location are no longer obstacles facing analysts. Data analysis or data mining is concerned about finding patterns from datasets that are useful for users, particularly managers, to perform planning (Thabtah and Hamoud, 2014).

One of the known data mining tasks that involve forecasting class labels in previously unseen data based on classifiers learnt from training dataset is classification. Normally, classification is performed in two steps: Constructing a model often named the classifier from a training dataset, and then utilising the classifier to guess the value of the class of test data accurately. This type of learning is called supervised learning since while building the classifier the learning is guided toward the class label. Common applications for classification are medical diagnoses (Rameshkumar et al., 2013), phishing detection (Abdelhamid et al., 2014), etc. There have been many different classification approaches including decision trees (Witten and Frank, 2005), Neural Network (NN) (Mohammad et al., 2013), Support Vector Machine (SVM) (Cortes and Vapnik, 1995), Associative Classification (AC) (Thabtah, et al., 2004), rule induction (RI) (Holt, 1993) and others. The latter two approaches, i.e. AC and RI, extract classifiers which contain "If-Then" rules so this explains their wide spread applicability. However, there are differences between AC and RI especially in the way rules are induced as well as pruned. This article falls under the umbrella of RI research.

PRISM is one of the RI techniques which was developed in (Cendrowska, 1987) and slightly enhanced by others, i.e. (Stahl and Bramer, 2008) (Elgibreen and Aksoy, 2013) (Stahl and Bramer, 2014). This algorithm employs separate-and-conquer strategy in knowledge discovery in which PRISM generates rules according to the class labels in the training dataset. Normally for a class, PRISM starts with an empty rule and keeps appending items to the rule's body until this rule reaches zero error (Definition 8- Section 2.2). When this occurs, the rule gets induced and the training data samples connected with the rule are discarded. The algorithm continues building other rules in the same way until no more data connected with the current class can be found. At this point, the same steps are repeated for the next-in-line class until the training dataset becomes empty.

One of the obvious problems associated with PRISM is the massive numbers of rules induced, which normally results in large size classifiers. This problem is attributed to the way PRISM induces the rules where it keeps adding items to the rule's body until the rule becomes 100% accurate despite the low data coverage. In other words, PRISM does not mind inducing many specific rules, each covering a single data sample, rather producing a rule, say, with 90% accuracy covering 10 data samples. This excessive learning limits the applicability of PRISM as a decision making tool for managers in application domains and definitely overfits the training dataset. This is since managers normally prefer a summarised set of rules that they are able to control and comprehend rather a larger high maintenance set of rules. In fact, there should be a trade-off between the number of rules offered and the predictive accuracy performance of these rules.

This paper investigates shortcomings associated with PRISM algorithm. Specifically, we look into three main issues:

1) The search space reduction: When constructing a rule for a particular class, PRISM has to evaluate the accuracy of all available items linked with that class in order to select the best one that can be added to the rule's body. This necessitates large computations when the training data has many attribute values and can be a burden especially when several unnecessary computations are made for items that have low data representation (weak items). A frequency threshold that we call (*freq*) can be employed as a pre-pruning of items with low frequency. It prevents these items from being part of rules, and therefore the search space gets minimised.
2) PRISM only generates a rule when its error is zero, which may cause overfitting the training dataset. We want to derive good quality rules, not necessarily with 100% accuracy, to reduce overfitting and increase data coverage. We utilise rule's strength parameter (Rule_Strength) to separate between acceptable and non-acceptable rules in our classifier.
3) When removing training data linked with a rule, we ensure that other items which have appeared in the removed data are updated. In particular, we amend the frequency of the impacted items. This indeed maintains the true weight of the items rather the computed frequency from the initial input dataset.

In response to the above raised issues, we are developing in this article, a new dynamic learning method based on RI that we name enhanced Dynamic Rule Induction (eDRI). Our algorithm discovers the rule one by one per class and primarily uses a freq threshold to limit the search space for rules by discarding any items with insufficient data representation. For each rule, eDRI updates items frequency that appeared within the deleted training instances of the generated rule. This indeed gives a more realistic classifier with lower numbers of rules leading to a natural pruning of items during the rule discovery phase. Lastly, the proposed algorithm limits the use of the default class rule by generating rules with accuracy <100%. Often these rules are ignored by PRISM algorithm since they don't hold zero error. These rules are only used during class prediction phase instead of the default class rule and when no 100% accuracy rule is able to classify a test data.

This paper is structured as follows: Section 2 illustrates the classification problem and its main related definitions. Section 3 critically analyses PRISM and its successors, and Section 4 discusses the proposed algorithm and its related phases besides a comprehensive example that reveals eDRI's insight. Section 5 is devoted to the data and the experimental results analysis, and finally, conclusions are provided in Section 6.

2. The Classification Problem and Definitions

Given a training dataset *T*, which has *x* distinct columns (attributes) Att_1, Att_2, … ,Att_n, one of which is the class, i.e. *cl*. The cardinality of *T* is |*T*|. An attribute may be nominal which means it takes a value from a predefined set of values or continuous. Nominal attributes values are mapped to a set of positive integers whereas continuous attributes are preprocessed by discretising their values using any discretisation method. The aim is to make a classifier from *T,* e.g. $Classfiier : Att \rightarrow cl$, which forecasts the class of previously unseen dataset.

Our classification method employs a user threshold called ***freq***. This threshold serves as a fine line to distinguish strong items ruleitems<item, class> from weak ones based on their computed occurrences in *T*. Any ruleitem that its frequency passes the *freq* is called as a strong ruleitem, otherwise it is called weak ruleitem. Below are the main terms used and their definitions.

Definition 1: An *item* is an attribute plus its values name denoted (A_i, a_i).

Definition 2: A *training example* in *T* is a row consisting of attribute values (A_{j1}, a_{j1}), …, (A_{jv}, a_{jv}), plus a class denoted by c_j.

Definition 3: A *ruleitem r* has the format<*body*, *c*>, where *body* is a set of disjoint items and *c*is a class value.

Definition 4: The frequency threshold (*freq*) is a predefined threshold given by the end user.

Definition 5: The body frequency (*body_Freq*) of a *ruleitem r in T* is the number of data examples in *T* that match *r's body*.

Definition 6: The frequency of a *ruleitem r in T* (*ruleitem_freq*) is the number of data examples in *T* that match *r*.

Definition 7: A *ruleitem r* passes the *freq* threshold if, *r*'s |*body_Freq*|/ |*D*| ≥ *freq*. Such a *ruleitem* is said to be a strong *ruleitem*.

Definition 8: A rule *r* expected accuracy is defined as |*ruleitem_freq*|/ |*body_Freq*|.

Definition 9: A rule in our classifier is represented as: $body \rightarrow cl$, where body is a set of disjoint attribute values and the consequent is a class value. The format of the rule is: $a_1 \wedge a_2 \wedge ... \wedge a_n \rightarrow cl_1$

3. Literature Review

PRISM is a key algorithm for building classification models that contain simple yet effective easy to understand rules. This algorithm was developed in 1987 based on the concept of separate and conquer where data examples are separated using the available class labels. For each class (w_i) data samples, an empty rule (If nothing then w_1) is formed. The algorithm computes the frequency of each attribute value linked with that class and appends the attribute value with the largest frequency to the current rule's body. PRISM terminates the process of a building a rule when that rule has an accuracy 100% according to definition (8). When the rule is generated, the algorithm continues producing the remaining possible rules from w_1' s data subset until the data subset becomes empty or no more attribute value can be found with an acceptable accuracy. At that point, PRISM moves to the second class data subset and repeats the same steps. The algorithm terminates when the training dataset is evaluated and when this happens the rules formed make the classifier. Figure 1 below depicts PRISM algorithm major steps.

```
Input: Training dataset T
Output: A classifier that consists of If-Then rules
Step 1: For each subset Ti in T that belong to $w_i$ Do
Step 1.1 Make an empty rule, $r_j$: If Empty then $w_i$
Step 1.2: Calculate Attx in Ti $p(w_i = i| Attx)$;
Step 1.3: Append the Attx with the largest $p(w_i = i| Attx)$ to the body of $r_j$
Step 2: Repeat steps 1.1-1.3 until $r_j$ has 100% accuracy or no longer can be improved
Step 2.1: Generate $r_j$ and insert it into the classifier
Step 3: Discard all  data examples from Ti that match $r_j$'s body
Step 4: Continue producing rules until Ti is empty or no rule with accepted error can be found
Step 5: Repeat steps 1-4 until no more data subsets of can be $w_i$ found
Step 6: IF(Ti does not contain any data examples  of class $w_i$ )
Generate the classifier.
```

Fig. 1 PRISM Pseudocode

Below are the main pros and cons associated with PRISM algorithm based on the comprehensive review that we have done,

PRISM Pros

1) The simplicity of rules generation in which only the rule's accuracy parameter is computed to decide on the rule significance.
2) The classifier contains easy to understand rules which can empower decision makers particularly in domain applications that necessitate interpretations.
3) Easy implementation especially with the available computing resources. Actually, PRISM can be easily implemented in different versions: local, online, distributed and in environments that require parallelism.
4) The predictive power of its classifier can be seen as acceptable when contrasted to other classic data mining approaches such as search methods, decision trees, neural networks, associative classification and many others.

PRISM Cons

1) In the original PRISM algorithm, there is no clear search space reduction mechanism for candidate items and therefore for large dimensional datasets, the expected numbers of candidate items can be huge. This may limits its use for certain applications.
2) Noisy datasets that contain incomplete attributes and missing values. This can been as a major challenge for PRISM since no clear mechanisms for handling noise is presented. Currently, adopted approaches from information theory are used to handle noise (Bramer, 2000) (Stahl and Bramer, 2014).
3) Handling numeric attributes. No build-in strategy is available in PRISM to discretise continuous attributes.
4) No clear rule pruning methodology is present in the original PRISM. This may lead to the discovery of large numbers of rules and lead to combinatorial explosion (Abdelhamid and Thabtah, 2014). There is a high demand on pruning methods to cut down the number of rules without hindering the overall performance of the classifiers.
5) Conflicting rule: There is no clear mechanism on how to resolve conflicting rules in PRISM. Currently the choice is random and favours class labels with the largest frequency linked with the item rather than keep multiple class labels per item.
6) Breaking tie among items frequency while building a rule: When two or more items have the same frequency, PRISM looks at their denominator in the expected accuracy formula. Yet, sometimes these items have similar accuracy and denominators which makes the choice random. Arbitrary selection without scientific justification can be seen as a biased decision and may not be useful for overall algorithm's performance.

One of the major challenges faced by PRISM algorithm is noisy training datasets. These are datasets that contain missing values, data redundancy, and incomplete data examples. A modified version has been developed called N- RISM to handle noisy datasets besides focusing on maximising the prediction accuracy (Bramer, 2000). Experimental tests using eight datasets from the UCI (Lichman, 2013) have showed consistent performance of N-PRISM when compared with classic PRISM particularly on classification accuracy obtained from noisy and noise-free datasets. It should be noted that the differences between the original PRISM and N-PRISM are very minimal.

Modified PRISM methods were developed based on a previously design pre-pruning method called J-Pruning by (Bramer, 2002) (Stahl and Bramer, 2012). The purpose was to reduce overfitting the training dataset. J-Pruning is based on the information theory test performed typically in decision tree by measuring the significance of removing an attribute from the rule's body. The algorithm of (Stahl and Bramer, 2012) was proposed to address rule pruning issue during the process of building rules for each class label.

Experimental results using sixteen UCI datasets showed decrement on the number of items per rule. In 2015, (Othman and Bryant, 2015) have investigated instance reduction methods as a paradigm for rule pruning in RI. They claimed that minimising the training dataset to the subset needed to learn the rules is one way to shrink the search space. The authors have applied three common instance reduction methods namely DROPS, ENN and ALLKNN on limited number of datasets to evaluate their impact on the resulting classifiers. The limited results obtained can be seen as a promising direction for using instance reduction methods as pre-pruning phase in RI.

The database coverage pruning (Liu et al., 1998) and its successors (Abdelhamid, et al., 2014) (Thabtah, et al., 2011) were one of the major breakthroughs in associative classification that can be employed successfully in RI as late or post pruning methods. These pruning methods necessitate a positive data coverage per rule with a certain accuracy so the rule can be part of the classifier. A new version of the database coverage pruning was developed by (Ayyat, et al., 2014) as a rule prediction measure. The authors have proposed a method that considers the rule rank as a measure of goodness besides the number of similar rules sharing items in their body. These two parameters play a critical role in differentiating among available rules especially in predicting the test data class labels.

Recently, (Elgibreen and Aksoy, 2013) investigated a common problem in PRISM and RULES (Pham and Soroka, 2007) family of algorithms which is the tradeoff between training time and classification accuracy during constructing the rule based classifiers. Moreover, the same article also highlighted performance deterioration of RI methods when applied to incomplete datasets. The result is a new covering algorithm that utilises "Transfer Knowledge" approach to fill in missing attribute values specifically the target attribute in the training dataset before mining kicks in. The "Transfer Knowledge" is basically building up a knowledge base based on learning curves via agents from different environments and from previous learning experience and then using this knowledge base to fill in incomplete data examples (Ramon et al., 2007). Experimental results against eight datasets revealed that the improved covering algorithm consistently produced competitive classifiers when compared with other RI algorithms such as RULES-IS and PRISM.

One of the major obstacles facing the data mining algorithm is the massive amount of data that are stored and scattered in different geographical location. Decision makers are striving to have an "on the fly" mining approach that processes very huge database simultaneously hoping to improve planning decisions. Most of the research works on parallelisation in classification are focused on decision trees. However, RI may present simple classifiers to decision makers. The big data problem have been investigated by amending PRISM algorithm to handle parallelisation (Stahl and Bramer, 2012). The authors developed a strategy for parallel RI called Parallel Modular Classification Rule Induction (PMCRI). This strategy is a continuation of an early work by the same authors in 2008 which resulted in parallel PRISM (P-PRISM) (Stahl and Bramer, 2008). P-PRISM algorithm was disseminated to overcome PRISM's excessive computational process of testing the entire population of data attribute inside the training dataset. The parallelism involves sorting the attribute values based on their frequency in the input dataset and the class attribute. This means the algorithm will need only this information for processing the rules and hence holding these data rather than the entire input data reduces computing resources such as processing time and memory. The attribute values and their frequency are then distributed to clusters and rules are generated from each cluster. All rules are finally merged to form the classifier. Experiments using a replication of the Diabetes and Yeast datasets from UCI repository have been conducted. The results show that P-PRISM as well as the parallel RI strategy scales well. However, a better approach to evaluate the parallelisation is to utilise unstructured real data such as Bioinformatics or text mining where dimensionality is huge and number of instances vary rather structured datasets.

(Stahl and Bramer, 2014) developed a PRISM based method for ensemble learning in classification. Usually ensemble learning is an approach in classification used to improve the classifier's predictive accuracy by deriving multiple classifiers using any learning approach such as Neural Network, decision trees, etc, and then merging them to form a global classifier. Since PRISM often suffers from overfitting problem to decrease this risk, the authors have built multiple classifiers based on PRISM using the ensemble learning approach. The results derived from fifteen UCI datasets revealed that the ensemble learning model based on PRISM was able to generate results comparable with classic PRISM algorithm. Moreover, a parallel version of the new model has also been implemented by the authors and tested with respect to training time. Results on run time showed that the parallel version scales well during the process of constructing the classifier.

PRISM successors have focused mainly on improving the algorithm scalability or reducing overfitting. We have seen approaches such as P-PRISM that showed promising research direction toward parallel data mining using RI. Other approaches such as Ensemble Learning based PRISM was able to minimise overfitting the training data by generating ensemble classifiers that later on are integrated together to make a final classifier. Lastly, early pruning has been introduced to further minimise the search space by the introduction of J-pruning. There are needs to further cut down the irrelevant candidate items during forming the rules in PRISM. This indeed will have a positive impact on the algorithm's efficiency in mining the rules and building the classifier. We think that post pruning is essential in PRISM and should increase its chance of being used as a data mining solution and decision making tool in practical applications. Yet since the classifiers produced by this family of algorithms is large in size this limits its usage.

One promising solution to shrink the size of the classifier is by eliminating rules overlapping in the training example and having rules to cover larger data samples. This solution can be accomplished by having live items frequency during the mining phase since PRISM relies primarily on item's frequency to build rules. In particular, PRISM algorithm often employs the rule's accuracy as a measure to generate the rule especially when the rule's accuracy reaches 100%. This normally results in low data coverage rules. We want each candidate item that can be part of a rule body to be associated with its true frequency in the training dataset that usually changes when a rule is generated. Recall that when a rule is outputted by PRISM, all training data linked with it are discarded and this may affect items appearing in those discarded rows. When each item has its true data representation while building the classifier this indeed decreases the search space and all insufficient candidate items will be deleted rather stored as in PRISM. The overall benefit will be that of having fewer number of rules classifiers. These classifiers can be seen as a real decision support system since they hold concise knowledge that are maintainable and usable by decision makers.

4. Enhanced Dynamic Rule Induction Algorithm

The proposed algorithm (Figure 2) has two primary phases: rule production and class assignment of test data. In phase (1), eDRI produces rules from the training dataset that have accuracy>=Rule_Strength. This parameter is similar to the confidence threshold used in associative classification (Thabtah, et al., 2004) that aims to generate near perfect rules besides rules with 100% accuracy as classic PRISM. The proposed learning method ensures the production of rules that have zero error as well as rules that survive the Rule_Strength parameter. The algorithm terminates building up the classifier when no more rules have an acceptable accuracy or the training dataset becomes empty. When this occurs, all rules get merged together to form the classifier. There is another parameter utilised by eDRI algorithm in phase one to minimise the search space of items. This parameter is called freq and it is similar to the minimal support threshold used in the association rule. The freq parameter is primarily utilised to differentiate between items that are frequent (have high number of occurrences in the training dataset) from those which are not frequent. This surely eliminates items which have low frequency, i.e. <freq threshold, as early as possible and thus saving computing resources as well as ensuring only significant items (frequent items) can be part of any rule's body. The infrequent items are kept in PRISM in the hope of producing 100% accuracy rules, which indeed can be seen a major deficiency.

All rules are induced in phase (1). Whenever a rule is built, training data examples linked with it are deleted, and the frequency of the waiting candidate items to be added which appeared in the discarded data, are instantly updated. The update involves decrementing their frequencies. This can be seen as a quality assurance measure of not relying on the original frequency of items computed initially from the training dataset. Rather we have a dynamic frequency per item that is continuously changing whenever a rule is generated. Having said this, eDRI is a RI algorithm that does not allow items inside rules to share training data examples, hence ensuring live classifiers that are not dependant on a static training datasets, instead a dynamic dataset that lessens every time a rule is formed. In phase (2), rules derived are used to guess the type of the class for test cases. Our algorithm assumes that the attributes inside the training dataset are nominal and any continuous attribute must be discretised before the inducing the rules starts.

4.1 Rule Production and Classifier Building

The proposed algorithm scans the training dataset to record items plus their frequencies. Then it creates the first rule by appending the item that when added to the current rule achieves the best accuracy. Any item with frequency less than the freq threshold is ignored. The algorithm continues adding items to the current rule's body until the rule becomes with 100% accuracy. For any rule that cannot reaches 100%, our algorithm checks whether its accuracy is greater than the Rule_Strength threshold. If the rule passes the Rule_Strength it will be created otherwise it will be deleted. Two noticeable differences between our algorithm and PRISM successors in building rules:

a) In eDRI, no item is added to the rule's body unless it has the minimum frequency requirement. Otherwise the item gets ignored and will not be part of any rule
b) In eDRI, there is a possibility of creating rules that has accuracy less than 100% whereas PRISM only allows the generation of perfect rules.

In eDRI, whenever a rule is generated, the following applies

1. The rule's data samples in the training dataset are discarded
2. Before building the next in-line rule, our algorithm updates items frequencies which have appeared in the removed data samples.

The proposed algorithm continues creating rules for the current class until no more items with sufficient frequency can be found. At this point, eDRI moves to the next class and repeats the same process until the training dataset becomes empty.

The above rule discovery procedure keeps items rank dynamic specifically since an item frequency with the class is continuously amended whenever a rule is derived. The dynamism provides a distinguishing advantage for the algorithm in determining items which become weak during constructing the classifier. This minimises the search space for candidate items and should provide smaller in size classifiers. As matter fact, the proposed algorithm develops a pruning procedure that reduces overfitting and results in rules with larger data coverage than PRISM. To clarify, PRISM keeps adding items to the rule's body regardless the number of data examples that are covered by the rule. The focus of PRISM is maximising rule's accuracy even when the discovered rule covers one data example. This obviously may overfit the training data and results in large numbers of specific rules. A simple run of PRISM algorithm over the "Weather.nominal" (14 data samples) dataset from the UCI repository revealed two rules covering a single data example each. In other words, 33.33% of the PRISM classifier (2 rules out of six) covers just two data examples. This result if limited show how PRISM continues training without providing a red flag when to stop rule learning. For eDRI, these two rules are basically discarded since they don't hold enough data representation.

Since PRISM algorithm creates only rules with 100% accuracy, there is no rule preference procedure. On the other hand, eDRI classifier may contain rules with good accuracy not necessarily 100% and hence our algorithm utilises a rule sorting procedure to differentiate among rules. The rule's strength and frequency are the main criteria employed to sort rules. When two or more rules have the same strength then eDRI uses the rule's frequency as a tie breaker. Finally, whenever two or more rules have identical strength and frequency then the algorithm prefers rules with less number of items in their body.

```
Input: Training dataset T, Minimum Rule_Strength, Minimum Frequency (freq) thresholds
Output: A classifier that consists of If-Then rules
Step 1: For each Attx in T Do
Step 1.1: Calculate Attx in Ti $p(w_i = i| Attx)$;
Step 1.2: Append the Attx with the largest accuracy $(w_i = i| Attx)$ to the body of $r_j$
Step 2: Repeat steps 1.1-1.2 until $r_j$ has either
    a)  100% accuracy
    b)  or no longer can be improved and it has strength >= Rule_strength
Step 2.1: Generate $r_j$
Step 3: Discard all  data examples from T that contained $r_j$'s body
Step 3.1: Update the frequency of all impacted candidate items to reflect step 3
Step 3.2: Continue producing rules from Ti until all remaining unclassified items have frequency <freq threshold
          or no more data in Ti can be found
Step 4: Generate the classifier.
```

Fig. 2 eDRI Pseudocode

4.2Test Data Prediction

Our classifier consists of two types of rules

a) Rules with 100% accuracy: Primary rules (higher rank)

b) Rules with good accuracy, i.e. < 100%: Secondary rules (lower rank)

Whenever a test data is about to be classified, eDRI goes over the rules in the classifier in top down fashion starting with the primary rules. The first rule that has items identical to the test data classifies it. In other words, if the rule's body is contained with the test data then this rule's class is assigned to the test data. If no primary rules match the test data then eDRI moves into the lower rank rules to forecast the test data class. This procedure limits the use of the default class rule which may increase the numbers of misclassifications. It should be noted that when no rules are in the classifier match the test data then the default class rule is fired.

4.4 Example on the Proposed Algorithm and PRISM

This section provides a thorough example to distinguish the learning process of PRISM and our proposed algorithm, In particular, we show how rules are induced. The dataset displayed in Table 1 is used for the purpose of comparison. Assume that the freq threshold is set to 3 and the Rule_Strength to 80%. eDRI picks the item that has the largest accuracy when linked with a class after calculating the frequency of all items in Table 2.The largest accuracy item, i.e. (4/4), is associated with "Outlook=overcast" and it is linked with class "YES". Our algorithm generates the below rule since it achieves 100% accuracy and removes its training data (Red text of Table 1) besides updating the frequency of all items appeared in the removed data as shown in Table 2. All items highlighted in red in Table 2 failed to pass the freq threshold therefore they are ignored. Our algorithm has substantially minimised the search space by keep only strong items (the one not highlighted red in Table 2). Whereas, PRISM keeps these items aiming to seek for specific rules.

RULE (1) If "Outlook=overcast" then YES (4/4)

After R1 is created, three items linked with class "YES" will be discarded since they become weak and these are "Temperature=cool", "Humidity=high" and "Windy=true". Their frequency is highlighted in red within the third column of Table 2. eDRI starts building a new rule from the remaining data examples excluding the removed data of RULE (1). The largest accuracy (item+class), i.e. 80%, is linked with "Humidity=high, NO" according to the updated frequency and accuracy of Table 2 (Columns 4 and 5). So the algorithm starts making the second rule below

Table 1: Sample training dataset from (Witten and Frank, 2005)

outlook	**temperature**	**humidity**	**windy**	**play**	**covering rule**
sunny	hot	high	FALSE	no	Rule 2
sunny	hot	high	TRUE	no	Rule 2
overcast	hot	high	FALSE	yes	Rule 1
rainy	mild	high	FALSE	yes	Rule 3
rainy	cool	normal	FALSE	yes	Rule 3
rainy	cool	normal	TRUE	no	Default Rule: NO
overcast	cool	normal	TRUE	yes	Rule 1
sunny	mild	high	FALSE	no	Rule 2
sunny	cool	normal	FALSE	yes	Rule 3
rainy	mild	normal	FALSE	yes	Rule 3
sunny	mild	normal	TRUE	yes	Default Rule: NO
overcast	mild	high	TRUE	yes	Rule 1
overcast	hot	normal	FALSE	yes	Rule 1
rainy	mild	high	TRUE	no	Default Rule: NO

RULE (2) If “Humidity=high” then NO (4/5)

Since RULE (2) is not yet 100% accurate, all instances associated with it are separated in Table 3 to seek another possible item that can maximise the accuracy. eDRI computes the frequency of items of Table 3 as shown in Table 4. Based on Table 4, the highest rule’s accuracy, i.e. 3/3, is linked with “Outlook=sunny” so this item is added to the current rule’s body, and the rule’s accuracy becomes 100%. Hence, we generate Rule (2) below, remove all training data connected with it (the yellow text of Table 1), and update the frequency and accuracy for the remaining impacted items.

RULE (2) If “Humidity=high and Outlook=sunny” then NO (3/3)
After creating RULE (2), four items have been ignored since they become weak (their frequencies are highlighted in red in Table 1- Column 6). Actually, there are no longer items linked with class “NO” simply because none of the remaining ones survive the freq threshold. The first two rules cover successfully seven examples in Table 1 and only seven data examples are left. eDRI calculates again the accuracy of possible remaining items. Item “Windy=False, YES” is the largest accurate item with accuracy 100% (4/4) hence a the below rule is formed.

Table 2: Candidate items frequency and accuracy during rule generation process

	During Rule 1 Generation		After generating Rule (1)		After generating Rule (2)		After generating Rule (3)	
Candidate Item+Class	Original Frequency in the training data	Rules Accuracy	Freq. status	Rules Accuracy	Freq. status	Rules Accuracy	Freq. status	Rules Accuracy
Outlook=sunny, NO	3	60%	3	60%	0			
Outlook= rainy, NO	2							
Temperature= hot, NO	2							
Temperature= mild, NO	2							
Temperature= cool, NO	1							
Humidity= high, NO	4	57.15%	4	80%	1			
Humidity= normal, NO	1							
Windy= true, NO	3	50.00%	3	75%	2			
Windy= false, NO	2							
Outlook=overcast, YES	4	100%	0					
Outlook=sunny, YES	2	40.00%						
Outlook= rainy, YES	3	60.00%	3	60%	3	60%	0	
Temperature= hot, YES	2		0					
Temperature= mild, YES	4	66.67%	4	66.67	4	80%	2	
Temperature= cool, YES	3	75.00%	2					
Humidity= high, YES	3	42.85%	1					
Humidity= normal, YES	6	85.71	4	80%	4	80%	1	
Windy= true, YES	3	50.00%	1		2			
Windy= false, YES	6	75.00%	4	66.67	4	100%	0	

Table 3: Data samples associated with item "Humidity=high"

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
rainy	mild	high	FALSE	yes
sunny	mild	high	FALSE	no
rainy	mild	high	TRUE	no

Table 4: Updated accuracy of items computed from Table 3

Item+ NO	Frequency	Rule's Accuracy
Outlook=sunny	3	100%
Outlook=rainy	2	50%
Temperature= hot	2	100%
Temperature= mild	3	67%
Windy= true	2	100.00%
Windy= false	3	67%
Windy= false	3	67%

RULE (3) If "Windy=false" then YES (4/4)

This rule covers four data examples so they will be deleted (highlighted in green in Table 1) and three examples are left unclassified since non of the remaining items pass the freq threshold hence a default class rule is generated based on the largest frequency class connected with the unclassified instances (Default class rule is "NO" 2/3).

The classifier of eDRI contains 3 rules and one default class rule as follows

RULE (1) If "Outlook=overcast" then YES (4/4)
RULE (2) If "Humidity=high and Outlook=sunny" then NO (3/3)
RULE (3) If "Windy=false" then YES Otherwise "YES"
Otherwise "NO" (2/3)

In the above example, eDRI was able to induce three possible rules and a default class whereas PRISM produced six possible rules and a default class as shown below. Actually, most of PRISM rules cover very limited data examples (1 or 2 rows) whereas our algorithm rules cover at least three data examples. This makes a difference especially in large datasets or datasets with high dimensionality as we will see in the experimental section. The fact that our algorithm derived less rules by 42.85% than PRISM from a dataset of just 14 examples is an evidence if limited on the power of the new rule induction procedure proposed.

Prism rules
----

1) If outlook = overcast then yes
2) If humidity = normal and windy = FALSE then yes
3) If temperature = mild and humidity = normal then yes
4) If outlook = rainy and windy = FALSE then yes
5) If outlook = sunny and humidity = high then no
6) If outlook = rainy and windy = TRUE then no
   Otherwise "YES".

5. Data and Experimental Results

5.1 Settings

For the rule discovery phase, all algorithms have utilised 10-fold cross validation in the experiments. This procedure has been adopted to compute the error rate during constructing the classifiers since it reduces overfitting and it is commonly used in classification. All experiments have been conducted on a computing machine with 1.7 GHz processor. We have implemented our algorithm in WEKA (Witten and Frank, 2005) for fair testing so WEKA environment was used for conducting all experimental results. WEKA is an open source Java platform that was developed at the University of Waikato, New Zealand. It contains different implementations of data mining and machine learning methods for tasks including classification, clustering, regression, association rules and feature selection. We tested the applicability of eDRI algorithm and in general RI on two sets of data: real data related to website security (Mohammad, et al, 2015) and sixteen UCI datasets (Lichman, 2013). The aim of the experiments is to show the pros and cons of our algorithm when compared to other RI algorithms. We have chosen three known RI algorithms (PRISM, OneRule (Holt, 1993), Conjunctive Rule (Witten and Frank, 2005)) and one hybrid decision tree algorithm called PART (Frank and Witten, 1998) for fair comparison. The choice of these algorithms are based on two facts:

1) They produce If-Then classifiers
2) They utilise different learning strategies to induce the rules

It should be noted that the majority of PRISM successors such as (P-PRISM, N-PRISM) focus on treating noisy datasets and parallelism as described earlier in Section 3. Thus, their rule learning strategy is the same as the one in PRISM so we use PRISM WEKA's version for comparison. For eDRI, the frequency threshold has been set to 1% and the Rule_Strength to 50%. These values have been obtained after running a number of warming up experiments where the results show a balance between the classifier's size and the classification accuracy. Below are the main evaluation measures used to analyse the experimental results

1) Classifiers error rate (%)
2) Classifier size measured in the available numbers of rules
3) The number of instances covered by the rule and the available number of items in the rule's body (rule's length). We have introduced two measures named Average Rule Length and Weighted Average Rule Length described later in this section along with their mathematical notations.
4) The number of data examples scanned to generate the classifier. We want to measure the increase and decrease in the search space by eDRI when compared to PRISM.
5) Time taken to build the classifiers.

5.2 Security Data Results

Phishing is one of social engineering techniques used to exploit unawareness of people (Abdelhamid, et al., 2014). It allows hackers to get an advantage of the weaknesses in the web by demanding confidential information from users, such as usernames, passwords, financial account credentials and credit card details. This is often performed by inserting links and objects within emails that direct users to fake websites, which look like the original website. Often, victims of phishing may lose their bank details and other private information to the phishy email senders (Phishers). As a matter of fact, phishing costs financial institutions as well as online users hundreds of millions in monetary damages annually.

We consider a real dataset related to website phishing in the first sets of experiments. The data has been collected using a PHP script of (Mohammad, et al., 2012) and contains thirty features plus the class. The dataset features have been considered in previous research articles which supports our selection, i.e. (Abdelhamid, et al., 2014). The phishing dataset is of type binary classification since there is two class labels (Phishy, Legitimate). We have utilised over 11000 website examples from different sources including Yahoo directory, Millersmiles and Phishtank archives. A sample of ten data examples associated with sixteen features are depicted in Table 5. The first row of the table shows the sample features and the last column in the table is the class attribute (Phishy (-1) / Legitimate (1)). Some features are given two possible values (-1 for phishing and 1 for legitimate) and others ternary values (-1 for phishing, 1 for legitimate and 0 for suspicious). Features used are not limited to "URL of Anchor", "URL Length", "Having_IP_Address", "Prefix_Suffix", "Iframe", "Right"Click", etc. More details on the complete set of features can be found in (Mohammad, et al., 2015.

Table 5: Sample of ten websites data related to sixteen phishing features

having_IP_ Address	URL_ Length	Shortining Service	at- Symbol	Double Slash	Prefix_ Suffix	Sub Domain	SS Lfinal	Domain _registr ation	Favicon	Port	…	Class
-1	1	1	1	-1	-1	-1	-1	-1	1	1	…	-1
1	1	1	1	1	-1	0	1	-1	1	1	…	-1
1	0	1	1	1	-1	-1	-1	-1	1	1	…	-1
1	0	1	1	1	-1	-1	-1	1	1	1	…	-1
1	0	-1	1	1	-1	1	1	-1	1	1	…	1
-1	0	-1	1	-1	-1	1	1	-1	1	1	…	1
1	0	-1	1	1	-1	-1	-1	1	1	1	…	-1
1	0	1	1	1	-1	-1	-1	1	1	1	…	-1
1	0	-1	1	1	-1	1	1	-1	1	1	…	1
1	1	-1	1	1	-1	-1	1	-1	1	1	…	-1

having_IP_Address	URL_Length	Shortining_Service	at-Symbol	Double_Slash	Prefix_Suffix	Sub_Domain	SSLfinal	Domain_registration	Favicon	Port	…	Class
-1	1	1	1	-1	-1	-1	-1	-1	1	1	…	-1
1	1	1	1	1	-1	0	1	-1	1	1	…	-1
1	0	1	1	1	-1	-1	-1	-1	1	1	…	-1
1	0	1	1	1	-1	-1	-1	1	1	1	…	-1
1	0	-1	1	1	-1	1	1	-1	1	1	…	1
-1	0	-1	1	-1	-1	1	1	-1	1	1	…	1
1	0	-1	1	1	-1	-1	-1	1	1	1	…	-1
1	0	1	1	1	-1	-1	-1	1	1	1	…	-1
1	0	-1	1	1	-1	1	1	-1	1	1	…	1
1	1	-1	1	1	-1	-1	1	-1	1	1	…	-1

Table 6: Best nine features detected by information gain filtering method

Feature name	IG Rank	IG Score
SSLfinal_State	1	0.4994
URL_of_Anchor	2	0.4770
Prefix_Suffix	3	0.1230
web_traffic	4	0.1145
having_Sub_Domain	5	0.1090
Links_in_tags	6	0.0470
Request_URL	7	0.0460
SFH	8	0.0370
Domain_registeration_length	9	0.0360

Figure 3 shows the error rate (%) results of all considered algorithms against complete 31-features and the top 10-features after preprocessing the original dataset using information gain (IG) feature selection method (Table 6). The reason for selecting these features is since they are the only ones that pass preprocessing phase by scoring above the minimum score of IG. Figure 3 demonstrated a good and consistent performance in regards to error rate by the RI algorithms considered. Precisely, eDRI achieved competitive error rate to PRISM on the complete phishing data and outperformed all RI algorithms on the top ten features set. Surprisingly PRISM outperformed eDRI on the 31-features dataset by 2.47% yet derived 626 more rules. On the other hand and for the selected ten features by IG, eDRI outperformed PRISM and the rest of the RI algorithms. To be more specific, eDRI achieved 7.94%, 3.92% and 3.92% lower error rate than PRISM, Conjunctive Rule and OneRule algorithms. These figures give a clear evidence that the proposed algorithm's learning strategy has improved the classifier predictive power at least of the reduced features set of the security data. Actually, eDRI ability to limit the use of the default class has a positive effect on its accuracy. Moreover, pruning useless and redundant rules enhanced the performance by ensuring only effective rules are utilised in prediction. These rules unlike those of PRISM and its successors cover larger portions of the training dataset. One notable result on the phishing dataset is that PART decision tree algorithm produced the least error rate. PART has achieved less error than eDRI since it adopts information theory pruning (Entropy) to further discards rules. In addition this algorithm employs Reduced Error Pruning (REP) to prune partial decision trees which normally degrade error rate. We believe that if eDRI utilises Entropy for post pruning we could end up with further improved classifiers.

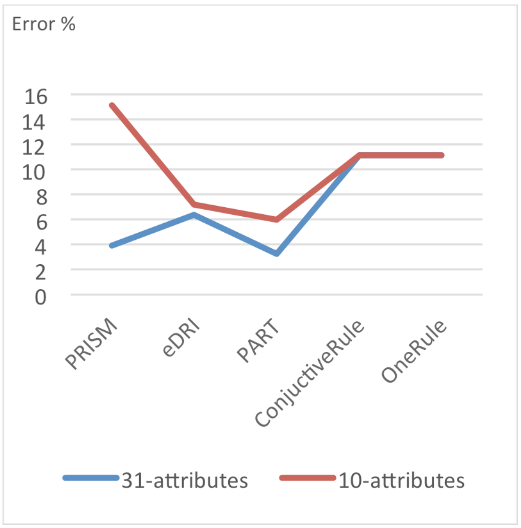

Fig. 3 The One error rate(%) of the considered algorithms on the phishing dataset

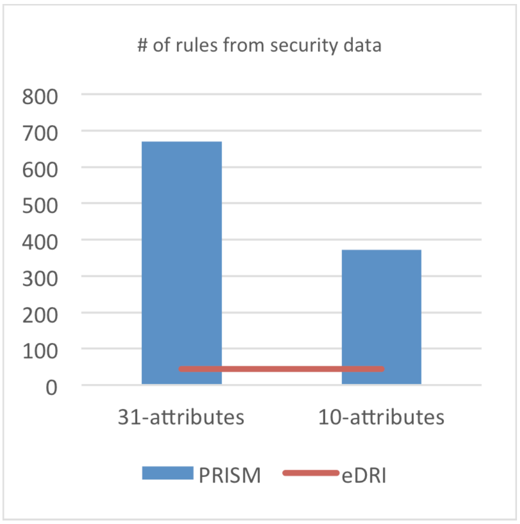

Fig. 4 Number of rules generated by PRISM and eDRI algorithms from the security data

We looked at the number of rules resulted by our algorithm and PRISM, which is shown in Figure 4. This figure illustrates that eDRI substantially minimised the classifier size. In fact, PRISM has derived 626 and 326 more rules than eDRI from the 31-features dataset and 10-features dataset where many of its rules are covering very limited data samples hence overfitting the training dataset. We looked further into the classifier produced by eDRI from the complete phishing dataset and observed that out of the 44 rules generated, there are 22 rules have error greater than zero, making them represent 50% of eDRI's classifier. Despite that these rules don't hold an accuracy of 100%, they are new useful knowledge that PRISM classifier don't hold. This is a good indication that generating rules not necessarily perfect (100% accuracy) not only minimises the classifier size but also covers larger numbers of training data examples per rule. Hence the overall performance of the classifier gets enhanced besides improving human controllability. In other words, having a smaller classifiers is a definite advantage for managers and decision makers since they are able to govern less amount knowledge during the decision making process. These classifiers work fine for applications such as medical diagnoses where general practitioners can enjoy a concise set of rules for daily diagnoses of their patients.

We have investigated the relationship between the rule length in the classifier and the number of instances each rule has covered. We created new simple measures called the average rule length (ARL) and weighted average rule length (WARL) that are showing in the below equations:

$$\text{ARL} = \frac{(\text{\# of rules of length } n * \text{rule length n})}{Total\ \#\ of\ rules}$$

$$\text{WARL} = \sum_{i=0}^{n} ri's\ length * \frac{\#\ of\ data\ example\ covere\ by\ ri}{size\ of\ the\ training\ dataset}$$

For instance, assume that we have the following two rules:
Rule 1 (if a=windy b = mild then yes) (length 2), and this rule covers 3 data instances
Rule 2 (if a=sunny then no) (length 1), and this rule covers 97 data instances
ARL = (1+2)/2 = 1.5
WARL = 2 * 3/100 + 1 * 97/100 = 1.03

For the dataset we consider and based on PRISM's classifier, the ARL and WARL computed from 10-features and 31-features datasets are (7.19, 4.23) and (10.09, 3.99) respectively. Whereas and based on eDRI's classifier, the ARL and WARL computed from 10-features and 31-features datasets are (4.67, 3.18) and (6.48, 3.96) respectively. The WARL figures demonstrate how the training attributes have been utilised to build the classifier. For instance, PRISM needed more features to make the rules than eDRI especially from the 10-features dataset. As a matter of fact, PRISM was excessively searching for more items to add per rule during rule building process. This explains the many specific rules (rules associated with large number of items) derived by PRISM which each classifies few training data examples. We assume that the above rationale is behind a larger WARL of PRISM. On the other hand and for the ARL computed, it seems that also eDRI has less number of items linked with each rule and this explains that our algorithm prefers general rules with lower length than PRISM. These general rules may contain several specific rules and this can be another reason behind the lower numbers of rules in eDRI's classifiers.

Since eDRI has been implemented in WEKA, we have investigated the number of times our algorithm pass over the training data examples when compared to PRISM. Hence, we recorded the number of rows that both algorithms have to scan while building the rules. It turns out that PRISM has passed over 34,465,178 training examples during the process of creating 670 rules. Whereas eDRI scanned 9,388,208 data examples resulting in 44 rules. These results evidently show that the proposed algorithm has substantially reduced the search space for rules. The fact that PRISM has to pass over many more millions of data examples explains the poor rule discovery mechanism employed by this algorithm, which overfits the training data, aiming to induce perfect yet low data coverage rules. This mechanism requires overtraining by repetitively splitting data examples whenever an item is appended to a rule in order to increase the rule's accuracy. The outcome guarantees the production of rules having zero error but it is definitely time consuming and computing resource demanding. The frequency threshold employed in eDRI was able to successfully discard many items that have low data representation. This obviously decreased the search space of items and therefore the classifier size is reduced. Moreover, allowing rule's which are not 100% accurate to be generated decreases the number of data examples that eDRI passes over. This is since eDRI is not always eager to generate rules that are perfect rather it prefers rule's with large data coverage but with acceptable accuracy, and this was obvious in its classifier that contains 22 non-perfect rules.

5.3 UCI Data Results

Different datasets from the UCI data repository have been used to generalise the performance of the proposed algorithm. The datasets have been selected based on different characteristics including the number of features (attributes), the size, the number of classes, and the type of the attributes as shown in Table 7. All second sets experiments have been conducted in WEKA. Table 7 also displays the error rate of the classifiers generated by PRISM, OneRule, Conjunctive Rule, PART and eDRI algorithms. We also show the average error rate of all considered algorithms in Figure 5. Based on the average error rate results of the classifiers, eDRI has derived on average the highest predictive RI classifiers but PART which is a decision tree algorithm maintained the best classifiers. In the figure, it is clear that eDRI algorithm performed on average well when compared to OneRule, PRISM and Conjunctive Rule algorithms. More specific, on average eDRI gained lower error rate by 9.70%, 14.05%, 13.56% than PRISM, Conjunctive Rule and OneRule algorithms respectively. This gain has been resulted from the dynamic rules production by this algorithm which surely keeps the accurate rules which lead to improvement of the predictive power. On the other hand, decision tree algorithm has higher classification accuracy on overage than our algorithm. To be more precise, PART has on average 2.54% slightly higher accuracy than eDRI on the sixteen UCI datasets used in the experiments. This can be attributed to the Reduced Error Pruning (REP), which are used by this algorithm during the process of constructing the classifier. The fact that eDRI is competitive to PART and produced on average higher predictive classifiers than its own kind is an achievement.

We further analysed Table 7 to seek detailed performance per dataset by the considered algorithms. Based on the table, the won-tie-lost records of eDRI against PRISM, Conjunctive Rule, OneRule and PART are 14-0-2, 12-1-3, 10-0-6 and 3-0-13. In other words, the proposed algorithm consistently outperformed PRISM and the other considered RI algorithms on the UCI datasets. This is evidently show good predictive performance by the proposed algorithm on small, medium and large datasets which makes it a favourable tool for decision making. PRISM performance on the UCI datasets has improved if compared to the real data from the security domain. This could be since the security data features are highlight correlated with the class attribute and this forces PRISM in searching deeply within the dataset for several specific rules. This explains the large rule's length vs. instances covered for PRISM.

Table 7: UCI datasets characteristics besides the considered algorithms error rate generated from the UCI datasets

Dataset	**# of classes**	**# of attributes**	**# of instances**	**PRISM**	**eDRI**	**PART**	**Conjunctive Rule**	**OneRule**
Contact lenses	3	5	24	45.84	33.33	16.66	37.50	29.16
Vote	2	17	435	6.67	6.43	4.96	4.96	4.37
Weather	2	5	14	64.28	35.71	42.85	35.71	57.14
Labor	2	17	57	25.35	22.80	17.54	24.56	31.57
Glass	7	10	214	51.40	47.66	39.25	53.73	48.59
Iris	3	5	150	12.04	10.00	4.66	39.33	4.00
Diabetes	2	9	768	39.01	27.08	26.56	34.37	26.43
Segment Challenge	7	20	1500	14.66	13.08	11.77	70.93	47.60
Zoo	7	11	101	57.42	6.93	7.92	40.59	93.06
Tic-Tac	2	10	958	4.32	8.45	5.84	31.00	30.06
Pima	2	7	768	43.74	23.56	23.30	25.65	25.26
Breast cancer	2	10	286	41.50	32.51	30.41	33.56	34.26
German_credit	2	16	1000	36.20	28.80	30.70	30.60	28.90
Autos	2	15	690	21.30	17.86	14.20	14.49	14.49
Soybean	19	36	683	14.63	10.10	8.49	73.79	66.47
Unbalanced	2	33	856	4.20	3.03	1.63	1.40	1.40

The classifier size of both PRISM and eDRI are shown in Figure 5. The figure clearly demonstrates that our algorithm's pruning method has significantly minimised the number of rules and no deterioration on the classifier's accuracy has been observed. The average number of rules derived by PRISM and eDRI from the sixteen UCI datasets are 117.18 and 33.31 respectively. This drastic decrease in the classifier size of eDRI is attributed to a) the discarding of weak items during building the rule and b) the production of rules not necessarily having 100%. These two distinctive advantages have resulted in concise classifiers of eDRI if compared to PRISM. We think that the dynamic update of candidate items cuts down the search space of the items and therefore less numbers of candidate items are presented. So removing the overlapping of the training instances among rules has a good effect on the classifiers size. In particular, eDRI ensures that all candidate items frequencies are amended on the fly whenever a rule gets produced, this minimises the available numbers of candidate items for the next rules, and hence rules generated quicker and usually classifies more data.

The time taken to construct the classifiers by PRISM, PART and our algorithm has been recorded to evaluate the efficiency factor. Figure 7 depicts the results in ms for the considered algorithms and against the UCI datasets we consider. The figure obviously illustrates that eDRI utilises less time to build the classifier than PART and PRISM due to there is no need to keep splitting training data samples in order to maximise each rule's accuracy. This has definite advantage at least over PRISM algorithm, which repetitively adds items to the rule aiming to reach 100% accuracy. This repetitive addition of items results in three major disadvantages:

- The resulting rules covering very small data samples. This may lead to the production of several redundant rules.
- More data samples need to scanned to identify the highest frequent items to be added to the current rule and eventually overfitting the training dataset. There is no clear stopping condition for rule learning in PRISM
- More training time is wasted

Figure 7 also demonstrated that PART is slower than eDRI in building the classifier due to the multiple pruning procedures invoked by PART.

We investigated the search space used by PRISM and eDRI to determine the major differences between both algorithms in constructing the classifier. Hence, we recorded the number of data samples (rows) scanned by both algorithms in determining items frequency and accuracy while building the rules. Table 8 illustrates our finding. For most of the datasets considered, PRISM had to scan more data samples to come up with the final classifier except for the “Labor” and “Vote” datasets. For instance and for the “Segment Challenge” dataset which consists of 20 attributes and 1500 instances, PRISM has to pass over 7,147,542 rows whereas eDRI scanned 938,263 rows. This is a clear sign that the proposed algorithm had substantially reduced the search space for rules in its classifier's building process. The fact that eDRI on average reduced the search space by around four times less than PRISM is a definite bonus. This is since it stops building rules early when any rule has an acceptable error rate.

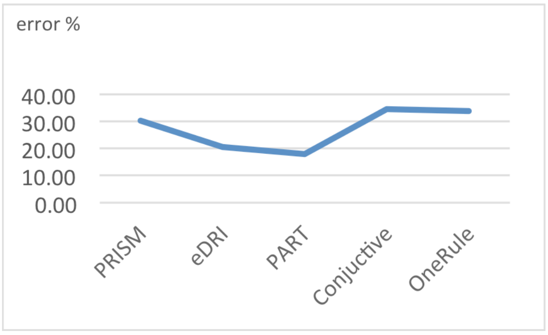

Fig. 5 Average error rate (%) for the considered algorithms on the UCI datasets

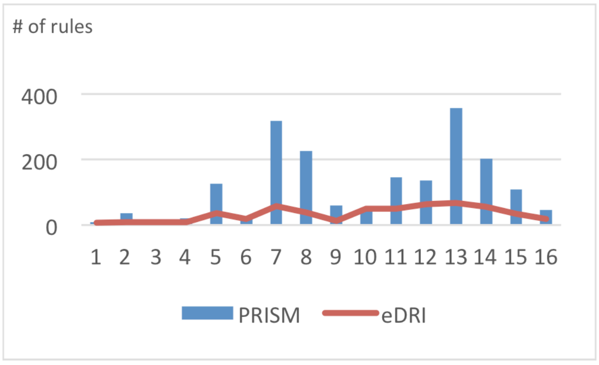

Fig. 6 The classifier size of PRISM and eDRI algorithms on the datasets

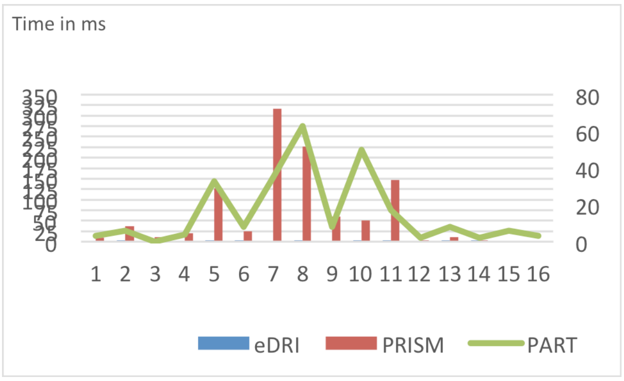

Fig. 7 The classifier building time in ms for PRISM and eDRI algorithms on the datasets

Table 8:# of scanned data samples during building the classifier by PRISM and eDRI on the UCI datasets

Datasset	**PRISM**	**eDRI**
Contact lenses	1503	168
Vote	212805	223797
Weather	1160	196
Labor	13796	19507
Glass	325951	58613
Iris	20613	15265
Diabetes	1979939	418680
Segment Challenge	7,147,542	938,263
Zoo	72,852	21,011
Tic-Tac	509,652	120,526
Pima	682,505	358,451

6. Conclusions

Rule Induction (RI) is a promising classification approach in data mining that attracted researchers due to its simplicity. This article deals with major shortcomings associated with PRISM, which is considered one of the known RI algorithms. Specifically, we investigated the problem of generating rules with limited data coverage by PRISM and discarding good quality rules covering larger training data portions, hence outputting massive classifiers. A new modified PRISM based algorithm that we call Enhanced Dynamic Rule Induction (eDRI) has been proposed and built into the WEKA environment to deal with the abovementioned deficiencies. Our algorithm minimises the search space for rules by cutting down items with low data representation and stopping learning when any rule meets the rule's strength threshold. These improvements guarantee smaller sized classifiers that don't overfit the training dataset and maintain their predictive performance.

The classifiers derived by the proposed algorithm are of high benefit to managers particularly when taking key business decisions since they can serve as a rich knowledge base inside a decision making tool. This is due to the classifiers being easy to understand, having high predictive accuracy, and they are controllable as well as robust (flexible to be amended without drastic change). Moreover, eDRI can be employed as a prediction module embedded inside a decision support system in various domain applications that necessitate both simplicity of the outcome and good accuracy, such as medical diagnoses systems. Since general practitioners usually have busy clinics with tight allocated time per patient they favour decision support systems with a concise set of knowledge similar to our classifiers.

Results using a real dataset related to security as well as general classification datasets from the UCI repository have been conducted utilising a number of RI and decision tree algorithms. The results revealed that the proposed algorithm is highly competitive with respect to error rate, search space, classifier size and processing time when compared to PRISM, OneRule, PART and Conjunctive Rule algorithms. Moreover, eDRI consistently produced fewer number of rules than PRISM on both the UCI and the security datasets, which increases its applicability. Nevertheless, PART decision tree algorithm outperformed eDRI in accuracy and on average has generated fewer numbers of rules for the majority of the datasets used. Hence, pruning using Entropy can be seen as advantageous for eDRI as a post pruning procedure. In addition, the current version of eDRI has no embedded discretisation method for continuous attributes.

In the near future, we would like to build eDRI as part of a decision support system for medical diagnoses to further evaluate its effectiveness and suitability in practical applications. In addition, eDRI will be embedded soon by researchers in Huddersfield University as an anti-phishing tool. This will raise awareness of public employees about phishing because of eDRI's outcome simplicity that can be understood by different users. Furthermore, eDRI can be utilised in web and mobile accessibility to evaluate websites and mobile applications for visually impaired users based on discovering correlations among features related to these user categories. We also intend in the near future to employ information theory pruning in particular Entropy to increase the predictive power of eDRI besides further decreasing its classifier size.

References

[1] Abdelhamid N., and Thabtah F. (2014) Associative Classification Approaches: Review and Comparison. *Journal of Information and Knowledge Management (JIKM),* 13, 1450027*. Sept 2014. Worldscinet.*

[2] Abdelhamid N., Ayesh A., Thabtah F. (2014) Phishing detection based associative classification data mining. Expert Systems with Applications 41 (13) Pages 5948–5959, Oct 2014.

[3] Ayyat Susan, Lu J., Thabtah F. (2014) Class Strength Prediction Method for Associative Classification. Proceedings of the IMMM 2014, The Fourth International Conference on Advances in Information Mining and Management, pp. 5-10. Paris, France, July 2014. Best Paper Award.

[4] Bramer, M.A. (2000) Automatic induction of classification rules from examples using N-PRISM", *Research and Development in Intelligent Systems XVI*, Springer-Verlag, pp. 99–121, 2000.

[5] Bramer A. (2002) An information-theoretic approach to the pre-pruning of classification rules, in: B. N. M Musen, R. Studer (Eds.), Intelligent Information Processing, Kluwer, 2002, pp. 201{212}.

[6] Cendrowska, J. (1987) PRISM: An algorithm for inducing modular rule*s. International Journal of Man-Machine Studies*, Vol.27, No.4, 349-370.

[7] Cortes, C., and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning, 20* (3), 273 – 297.

[8] Elgibreen H. A., Aksoy M. S. (2013) Rules – Tl: A Simple And Improved Rules Algorithm For Incomplete And Large Data. Journal of Theoretical and Applied Information Technology, pp. 28-40.

[9] Frank, E., and, Witten, I. (1998) Generating accurate rule sets without global optimisation. *Proceedings of the Fifteenth International Conference on Machine Learning,* (p. . 144–151). Madison, Wisconsin.

[10] Holte, R.C. (1993). Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. Machine Learning, 11, pp 63-90.

[11] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[12] Liu, B., Hsu, W., and Ma, Y. (1998) Integrating classification and association rule mining. *Proceedings of the Knowledge Discovery and Data Mining Conference- KDD*, 80-86. New York.

[13] Mohammad RM, Thabtah F, McCluskey L (2013) Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 1-16 (2013).

[14] Mohammad R. Thabtah F. McCluskey L., (2015) Phishing websites dataset. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites Accessed January 2016.

[15] Othman O. M. and Bryant C. H. (2015) Pruning Classification Rules with Instance Reduction Methods. International Journal of Machine Learning and Computing, Vol. 5, No. 3, June 2015.

[16] Pham D. T. and Soroka A. J. (2007) An Immune-network inspired rule generation algorithm (RULES-IS). In Third Virtual International Conference on Innovative Production Machines and Systems, Whittles Dunbeath, 2007.

[17] Rameshkumar, K., Sambath, M. and Ravi, S. (2013) Relevant association rule mining from medical dataset using new irrelevant rule elimination technique. *Proceedings of ICICES ,* 300-304.

[18] Ramon J., EDRIessens K., and Croonenborghs T. (2007) Transfer Learning in Reinforcement Learning Problems Through Partial Policy Recycling: Machine Learning. ECML 2007. vol. 4701, J. Kok, J. Koronacki, R. Mantaras, S. Matwin, D. Mladenic, and A. Skowron, Eds., ed: Springer Berlin / Heidelberg, 2007, pp. 699-707.

[19] Stahl, F., Bramer, M.A. (2014) Random Prism: an alternative to Random Forests. Research and Development in Intelligent Systems XXVIII, 2011, pp 5-18, Springer.

[20] Stahl F., Bramer M. (2012) Computationally efficient induction of classification ruleswith the PMCRI and J-PMCRI frameworks, Knowledge-Based Systems 35, (2012) 49–63.

[21] Stahl F., and Bramer M. (2008) P-Prism: A Computationally Efficient Approach to Scaling up Classification Rule Induction. Artificial Intelligence in Theory and Practice II, IFIP – The International Federation for Information Processing Volume 276, 2008, pp 77-86.

[22] Thabtah F., Hammoud S (2014) Parallel Associative Classification Data Mining Frameworks Based Mapreduce. To Appear in *Journal of Parallel Processing Letter*. March 2015. World Scientific.

[23] Thabtah, F., Cowling, P., and Peng, Y. (2004) MMAC: A new multi-class, multi-label associative classification approach. *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04*), 217-224.

[24] Thabtah F., Hadi W., Abdelhamid N., Issa A. (2011) Prediction Phase in Associative Classification. Journal of Knowledge Engineering and Software Engineering. Volume: 21, Issue: 6(2011) pp. 855-876. WorldScinet.

[25] Witten I. H. and Frank E. (2005). Data Mining: Practical Machine Learning Tools and Techniques.