# Using The Set of Attributes of Frequent Itemsets for Better Rough Set based Rules

HYONTAI SUG
Division of Computer Engineering
Dongseo University
47 Jurye-ro, Sasang-gu, Busan 47011
KOREA
shtt@gdsu.dongseo.ac.kr

*Abstract:* - Inductive learning algorithms want to find a function that reflects a given sequence of input and output pairs where the input and output pairs consist of value vectors. Rough set systems can extract minimal set of rules that act as the function of inductive learning, and are well known for their strong mathematical background. The found set of rules is solely based on data so that no prejudiced views can be inserted in the found rule set. But even though the good property, rough set based rule systems have the tendency of being unstable in the sense that their performance is very dependent on given training data sets due to their sole reliance on given data. In order to avoid such property of the rough set based rule systems this paper suggests using the attributes of frequent items only in the input vector to find the rules. The attributes can be found by applying association rule algorithms. Experiments with several real world data sets show that better rough set based rules could be found in accuracy by using the attributes only, especially when the attributes of input have key-like characteristics.

*Key-Words:* - Inductive learning, rough set theory, association rules, frequent attributes, data mining

## 1 Introduction

In inductive learning when we are given a sequence of input and output pairs of $<x_i, y_i>$, where $x_i$ is a possible input, and $y_i$ is the associated output of $x_i$, we want to find a function f such that $f(x_i) = y_i$ for all i's. Rough set-based machine learning algorithms are one of widely accepted machine learning technologies because of their good property after Z. Pawlak introduced rough set theory as a mathematical tool for data analysis [1]. Good property of rough set theory is that it can describe uncertain facts solely based on data, while many other machine learning algorithms use training methods that reflect prior knowledge, such as heuristics [2]. As a result, there is no room for prejudiced views to be inserted in the discovered knowledge [3].

Let T = (U, A, C, D) be a decision system, where U is a nonempty finite set called the universe, A is a nonempty finite set of attributes, C and D are subsets of A representing conditional and decision attributes respectively. Then a: U $\rightarrow$ Va for a $\in$ A, Va is called the value set of a. So, the elements of U are data instances. Rough set based rule systems find minimal rules for T.

There are also some other data mining or machine learning techniques solely based on data. Association rules [4, 5, 6] are one of representative techniques of such kind. Association rule systems find rules of association, where the association resides between sets of items in database. Minimum support that represents how many times an itemset occurs in a transaction database is given to find frequent itemsets. Because association rule algorithms find itemsets that occur more than given minimum support, this fact allows the algorithms to find the itemsets practically even for very large databases by supplying the minimum support appropriately.

When we find association rules in a decision table where the shape of the table is the same as decision system T in rough set systems, the found rules are called class association rules. The main difference between class association rules and rough set based rules is the minimization process on the rules. While association rule algorithms treat the minimization of conditional part of their found rules as a separate or additional process [7, 8], rough set based rule systems consider the minimization process as a necessity.

In section two we review related work, and in section three our method and the result of experiment will be given, and in section four conclusions will be discussed. This paper is the

extended version of previous work in CSCC 2017 [9].

## 2 Related Work

Originally, rough set theory based rule finding systems can deal with nominal attributes only [1], but, because many real world data sets contain continuous values so that MODLEM algorithm was invented to deal with the both kinds of data [10]. But, because it is an unstable algorithm due to overfitting [11], bagging was applied to compensate such property of the algorithm, and shows better performance in several data sets [12, 13]. Bagging stands for Bootstrap AGGregatING. Bootstrapping is a random sampling method based on replacement. Aggregating means that the final decision or classification is based on voting of generated machine learning models that come from the random samples. Because several or many training data sets are generated from the bootstrapping, a bagging method may avoid such unstable property of target machine learning algorithms [14]. Because MODLEM is prone to find the most accurate classifier with respect to overall accuracy, minority classes are often neglected. This property is also well studied for another representative unstable machine earning algorithms, decision trees [15, 16]. In [17] an oversampling method is suggested for minority classes to improve sensitivity while preserving overall accuracy. In addition to the efforts to improve the performance of rough set baes rule systems, several research activities to combine rough set and association rules have been performed. In [18] class association rule mining method is integrated with rough set approach where each transaction record has a class value, and shows good performance. In [19] a rough set based association rule approach was proposed for customer preference analysis, where the analysis generates rough set attribute functions with association rules. In [20] rough set and genetic algorithm based approach was used to find, so called, weighted association rules.

## 3 Method and Experiment

Because rough set theory-based algorithms find rules very thoroughly, it may confront with the overfitting problem. Overfitting training data set is a very well-known problem [21, 22]. Overfitting in machine learning algorithms occurs, because a training data set usually does not cover data space fully so that it makes the algorithms unstable. Therefore, if we apply the rough set-based rule discovery method directly to real world data sets, we may not get such good results as we expected, especially if the size of data set is small compared to the domain of the data set, and moreover, if the data set values are very specific for some attribute values. In other words, the data values are subdivided very much. In order to prove our assertion we'll perform several experiments with real world data sets. In the followings we will provide the results of experiments on three real world data sets called, zoo, sonar, and postoperative, all from UCI machine learning repository [23]. MODLEM algorithm [24] will used to generate rough set based rules, because MODLEM is a standard method that can find rules based on rough set theory. MODLEM can deal with both of continuous and nominal values. All experiments are based on 10-fold cross validation.

### 3.1 Data set 'zoo'

For our first experiment, a data set called 'zoo' from UCI machine learning depository is used. Zoo data has 17 conditional attributes and one decision attribute. The decision attribute has 7 different class values which classify animals in a zoo. The total number of instances is 101. Table 1 shows the meaning of each attribute.

**Table 1.** The attributes of data set 'zoo'

| domain | attribute |
|---|---|
| Unique for each instance except 'frog' | Name |
| Boolean | Hair, Feathers, Eggs, Milk, Airborne, Aquatic, Predator, Toothed, Backbone, Breathes, Venomous, Fins, Tail, Domestic, Catsize |
| Numeric (0, 2, 4, 5, 6, 8) | Legs |

The class attribute has seven different values from 1 to 7. There are 41, 20, 5, 13, 4, 8, and 10 instances from class 1 to 7 respectively. There are two instances of animal name 'frog', each has slight different attribute values. So, there are 100 different animal names in the data set.

### 3.1.1 Rules from the original data

MODLEM generated 7 rules with the accuracy of 42.6% from the original data set in 10-fold cross validation as follows:

Rule 1. If milk in {1} Then class = 1 (41/41, 100%)

Rule 2. If name in {chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren} Then class = 2 (20/20, 100%)

Rule 3. If name in {pitviper, seasnake, slowworm, tortoise, tuatara} Then class = 3 (5/5, 100%)

Rule 4. If name in {bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna} Then class = 4 (13/13, 100%)

Rule 5. If name in {frog, newt, toad} Then class = 5 (4/4, 100%)

Rule 6. If name in {flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp} Then class = 6 (8/8, 100%)

Rule 7. If name in {clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm} Then class = 7 (10/10, 100%)

The percentage in parentheses shows the confidence of the rule, and the fraction represents the number of classified instances over the number instances having the same condition part. As we see in the rules, MODLEM found rules precisely by using the name attribute in all the rules except in rule 1, and rules are very accurate. But, the test result is not good. In the data set each instance has a unique animal name except two instances, ‘frog’, and the above six rules use name attribute only to classify them. But, name attribute has key-like characteristics. So, we can see the reason why the test accuracy is somewhat low, because in 10-fold cross-validation, one tenth of data set is used for testing, while the others are used for training alternately.

### 3.1.2 Rules from frequent attributes only after discretization

Even though MODLEM can handle continuous values, because association rule algorithms cannot deal with continuous values, discretization method which is based on entropy by Fayyad et al. [25] was applied for application of the association rule algorithm. When MODEL was applied to this discretized data, the same seven rules with the same accuracy were generated for the discretized data also like the original data set.

Class association rule algorithm was used to find frequent attributes, because the data set has several conditional attributes and a decision attribute. Table 2 shows the frequent itemsets when minimum support is 10.

**Table 2.** Frequent itemsets

| itemset | support |
|---|---|
| hair=1 feather=0 eggs=0 milk=1 airborne=0 aquatic=0 predator=1 toothed=1 backbone=1 breathes=1 venomous=0 fins=0 legs='(3-4.5]' tail=1 domestic=0 | 12 |
| hair=1 feather=0 eggs=0 milk=1 airborne=0 aquatic=0 predator=1 toothed=1 backbone=1 breathes=1 venomous=0 fins=0 legs='(3-4.5]' tail=1 catsize=1 | 11 |
| hair=1 feather=0 eggs=0 milk=1 airborne=0 aquatic=0 predator=1 toothed=1 backbone=1 breathes=1 venomous=0 fins=0 legs='(3-4.5]' domestic=0 catsize=1 | 12 |
| hair=1 feather=0 eggs=0 milk=1 airborne=0 aquatic=0 toothed=1 backbone=1 breathes=1 venomous=0 fins=0 legs='(3-4.5]' tail=1 domestic=0 catsize=1 | 16 |
| hair=1 feather=0 eggs=0 milk=1 airborne=0 predator=1 toothed=1 backbone=1 breathes=1 venomous=0 fins=0 legs='(3-4.5]' tail=1 domestic=0 catsize=1 | 11 |

According to the frequent itemsets in table 2, the collection of attributes in the found frequent itemsets is {hair, Feather, eggs, milk, airborne, aquatic, toothed, backbone. breathes, venomous, fins, legs, tail, catsize}, and the collection of attributes in infrequent itemsets is {name, predator}. MODLEM algorithm was applied using these attributes of frequent itemsets only to see the effect of attribute selection. Eight rules with the accuracy of 96.0% were generated as follows:

Rule 1. If milk in {1} Then class = 1 (41/41, 100%)

Rule 2. If feather in {1} Then class = 2 (20/20, 100%)

Rule 3. If toothed in {1} AND fins in {0} AND legs in {'(-inf-1]'} Then class = 3 (3/3, 60%)

Rule 4. If legs in {'(3-4.5]'} AND hair in {0} AND aquatic in {0} Then class = 3 (2/2, 40%)

Rule 5. If fins in {1} AND eggs in {1} Then class = 4 (13/13, 100%)

Rule 6. If aquatic in {1} AND legs in {'(3-4.5]'} AND hair in {0} AND toothed in {1} Then class = 5 (4/4, 100%)

Rule 7. If legs in {'(4.5-inf)'} AND aquatic in {0} AND eggs in {1} Then class = 6 (8/8, 100%)

Rule 8. If backbone in {0} AND airborne in {0} Then class = 7 (10/12, 100%)

### 3.1.3 Rules after eliminating attribute 'name' only for discretized data

Comparing the previous two results from the original data set and the modified data set by selecting the 14 attributes only, we doubt that the attribute 'name' might have negative effect for accuracy, because the attribute occurs very often in the rule set from the original data set. After omitting 'name' attribute only, we ran MODLEM again, resulting in 10 rules with accuracy of 94.1% as follows:

Rule 1.If milk in {1} Then class = 1 (41/41, 100%)

Rule 2. If feather in {1} Then class = 2 (20/20, 100%)

Rule 3. If toothed in {1} AND fins in {0} AND legs in {'(-inf-1]'} Then class = 3 (3/3, 60%)

Rule 4. If legs in {'(3-4.5]'} AND hair in {0} AND aquatic in {0} Then class = 3 (2/2, 40%)

Rule 5. If fins in {1} AND eggs in {1} Then class = 4 (13/13, 100%)

Rule 6. If aquatic in {1} AND legs in {'(3-4.5]'} AND hair in {0} AND toothed in {1} Then class = 5 (4/4, 100%)

Rule 7. If legs in {'(4.5-inf)'} AND predator in {0} Then class = 6 (7/7, 87.5%)

Rule 8. If legs in {'(4.5-inf)'} AND airborne in {1} Then class = 6 (6/6, 75%)

Rule 9. If backbone in {0} AND airborne in {0} AND predator in {1} Then class = 7 (8/8, 80%)

Rule 10. If backbone in {0} AND legs in {'(-inf-1]'} Then class = 7 (4/4, 40%)

When we removed attribute 'predator' only that is the other attribute not having been selected above, MODLEM generated the same result of accuracy of 42.6% with the original data, discretized and un-discretized as well. The above experiments prove that our assertion is true and show the property of overfitting.

### 3.1.4 Rules after eliminating attribute 'name' and 'predator' attributes from the original data

One more experiment after eliminating the two attributes, 'name' and 'predator' in the original data set without discretization was performed, and we found similar result with accuracy of 96.0% by MODLEM, and the found 8 rules have slight different shape in conditional part with the one after discretization in section 3.1.2 due to the discretization as follows:

Rule 1. If milk in {1} Then class = 1 (41/41, 100%)

Rule 2. If feather in {1} Then class = 2 (20/20, 100%)

Rule 3. If toothed in {1} AND fins in {0} AND legs < 1Then class = 3 (3/3, 60%)

Rule 4. If legs >= 3 AND tail in {1} AND eggs in {1} AND aquatic in {0} Then class = 3 (2/2, 40%)

Rule 5. If fins in {1} AND eggs in {1} Then class = 4 (13/13, 100%)

Rule 6. If aquatic in {1} AND legs >= 3 AND toothed in {1} AND hair in {0} Then class = 5 (4/4, 100%)

Rule 7. If legs >= 5.5 AND aquatic in {0} AND eggs in {1} Then class = 6 (8/8, 100%)

Rule 8. If backbone in {0} AND airborne in {0} Then class = 7 (10/12, 100%)

Table 3 summarizes all the previous experiments using MODLEM for the data sets.

**Table 3.** The summary of the experiments on zoo data set

| | Data set | Accuracy (%) | Number of rules |
|---|---|---|---|
| No discretization | The original | 42.6 | 7 |
| | Data set whose 'predator' attribute has been removed | 42.6 | 7 |
| | Data set whose 'name' and 'predator' attribute have been removed | 96.0 | 8 |
| After discretization | The original | 42.6 | 7 |
| | Data set whose 'predator' attribute has been removed | 42.6 | 7 |
| | Data set whose 'name' attribute has been removed | 94.1 | 10 |
| | Data set whose 'name' and 'predator' attribute have been removed | 96.0 | 8 |

## 3.2 Data set 'sonar'

For our second experiment, a data set called 'sonar' from UCI machine learning depository is used. Sonar data were obtained from bouncing sonar signals from a metal cylinder and rocks. The data set has 60 conditional attributes and one decision attribute. The 60 conditional attributes have a real number between 0 and 1 representing the energy within a particular frequency band. The decision attribute has two different class values, rock(R) and metal cylinder(M). The total number of instances is 208. There are 97 instances in the rock class, and 111 instances in the metal cylinder class.

### 3.2.1 Rules from the original data

When MODLEM is applied to the original data set, the accuracy is 70.6731% with 59 rules. The followings are the generated rules:

Rule 1. (A11 < 0.09) AND (A9 < 0.13) => (class = R) (22/22, 22.68%)
Rule 2. (A30 < 0.16) => (class = R) (5/5, 5.15%)
Rule 3. (A13 < 0.06) => (class = R) (5/5, 5.15%)
Rule 4. (A17 < 0.08) => (class = R) (6/6, 6.19%)
Rule 5. (A38 < 0.06) => (class = R) (4/4, 4.12%)
Rule 6. (A42 < 0.03) => (class = R) (3/3, 3.09%)
Rule 7. (A47 < 0.02) => (class = R) (5/5, 5.15%)
Rule 8. (A52 < 0) AND (A21 < 0.81) => (class = R) (9/9, 9.28%)
Rule 9. (A6 >= 0.29) => (class = R) (2/2, 2.06%)
Rule 10. (A9 < 0.03) => (class = R) (6/6, 6.19%)
Rule 11. (A12 >= 0.62) => (class = R) (2/2, 2.06%)
Rule 12. (A3 < 0.01) AND (A16 < 0.27) => (class = R) (7/7, 7.22%)
Rule 13. (A5 < 0.01) AND (A1 >= 0.01) => (class = R) (8/8, 8.25%)
Rule 14. (A32 < 0.08) => (class = R) (4/4, 4.12%)
Rule 15. (A36 >= 0.99) => (class = R) (3/3, 3.09%)
Rule 16. (A38 >= 0.97) => (class = R) (1/1, 1.03%)
Rule 17. (A56 < 0) => (class = R) (2/2, 2.06%)
Rule 18. (A6 < 0.02) AND (A1 < 0.02) => (class = R) (5/5, 5.15%)
Rule 19. (A15 < 0.06) AND (A1 >= 0) => (class = R) (7/7, 7.22%)
Rule 20. (A51 < 0) AND (A1 < 0.02) => (class = R) (5/5, 5.15%)
Rule 21. (A17 >= 0.91) AND (A1 < 0.04) => (class = R) (6/6, 6.19%)
Rule 22. (A13 < 0.1) AND (A6 < 0.07) => (class = R) (8/8, 8.25%)
Rule 23. (A37 < 0.05) AND (A1 < 0.02) => (class = R) (3/3, 3.09%)
Rule 24. (A38 < 0.07) AND (A1 < 0.01) => (class = R) (2/2, 2.06%)
Rule 25. (A39 < 0.06) AND (A1 < 0.02) => (class = R) (3/3, 3.09%)
Rule 26. (A41 >= 0.71) AND (A1 >= 0.02) => (class = R) (3/3, 3.09%)
Rule 27. (A42 < 0.04) AND (A1 < 0.02) => (class = R) (3/3, 3.09%)
Rule 28. (A53 < 0) AND (A1 < 0.03) => (class = R) (1/1, 1.03%)
Rule 29. (A8 < 0.03) AND (A2 < 0.02) => (class = R) (4/4, 4.12%)
Rule 30. (A24 >= 1) AND (A2 < 0.03) => (class = R) (2/2, 2.06%)
Rule 31. (A10 < 0.07) AND (A2 < 0.02) => (class = R) (9/9, 9.28%)

Rule 32. (A44 >= 0.43) AND (A5 < 0.13) => (class = M) (22/22, 19.82%)
Rule 33. (A54 >= 0.02) AND (A2 >= 0.02) => (class = M) (20/20, 18.02%)
Rule 34. (A28 >= 0.99) AND (A1 < 0.07) => (class = M) (16/16, 14.41%)
Rule 35. (A55 >= 0.02) => (class = M) (9/9, 8.11%)
Rule 36. (A60 >= 0.02) => (class = M) (6/6, 5.41%)
Rule 37. (A31 < 0.12) => (class = M) (3/3, 2.7%)
Rule 38. (A36 < 0.03) => (class = M) (4/4, 3.6%)
Rule 39. (A10 < 0.02) => (class = M) (2/2, 1.8%)
Rule 40. (A14 < 0.04) => (class = M) (2/2, 1.8%)
Rule 41. (A23 < 0.06) => (class = M) (2/2, 1.8%)
Rule 42. (A29 < 0.06) => (class = M) (2/2, 1.8%)
Rule 43. (A41 < 0.05) => (class = M) (3/3, 2.7%)
Rule 44. (A25 >= 0.95) AND (A9 >= 0.13) => (class = M) (12/12, 10.81%)
Rule 45. (A22 >= 1) AND (A1 >= 0.01) => (class = M) (6/6, 5.41%)
Rule 46. (A3 < 0) => (class = M) (1/1, 0.9%)
Rule 47. (A10 >= 0.64) => (class = M) (2/2, 1.8%)
Rule 48. (A26 < 0.16) => (class = M) (3/3, 2.7%)
Rule 49. (A33 >= 0.99) => (class = M) (1/1, 0.9%)
Rule 50. (A54 < 0) => (class = M) (2/2, 1.8%)
Rule 51. (A28 >= 0.96) AND (A5 < 0.05) => (class = M) (14/14, 12.61%)
Rule 52. (A1 in [0.07, 0.07]) => (class = M) (5/5, 4.5%)
Rule 53. (A23 >= 0.99) AND (A1 >= 0.02) => (class = M) (3/3, 2.7%)
Rule 54. (A30 >= 0.98) AND (A1 >= 0.02) => (class = M) (3/3, 2.7%)
Rule 55. (A9 >= 0.51) AND (A1 < 0.02) => (class = M) (2/2, 1.8%)
Rule 56. (A57 < 0) AND (A1 >= 0.02) => (class = M) (1/1, 0.9%)
Rule 57. (A22 >= 0.94) AND (A13 >= 0.22) => (class = M) (9/9, 8.11%)
Rule 58. (A3 in [0, 0]) => (class = M) (1/1, 0.9%)
Rule 59. (A5 >= 0.16) AND (A6 < 0.09) => (class = M) (3/3, 2.7%)

### 3.2.2 Rules from frequent attributes only after discretization

Discretization method was applied for the application of the association rule algorithm. 39 attributes were discretized as ‘all’ meaning only one value for the attributes. These attributes were eliminated before applying the association rule algorithm, because their combinations to make itemsets are meaningless.

When minimum support ratio of 0.3 was applied, 14 frequent itemsets were found. The attribute set in found frequent itemsets is {A4, A5, A9, A10, A11, A12, A13, A28, A35, A44, A45, A46, A49, A54}. Each attribute has two discretized values. After selecting the attributes, MODLEM was applied. MODLEM generated 41 rules with the accuracy of 82.25%. The followings are found rules:
Rule 1. (A9 in {'(-inf-0.1164]'}) AND (A5 in {'(-inf-0.0392]'}) AND (A45 in {'(-inf-0.38545]'}) => (class = R) (28/28, 32.18%)
Rule 2. (A9 in {'(-inf-0.1164]'}) AND (A4 in {'(-inf-0.052]'}) AND (A11 in {'(-inf-0.19795]'}) AND (A28 in {'(-inf-0.9233]'}) AND (A54 in {'(-inf-0.0225]'}) => (class = R) (42/42, 48.28%)
Rule 3. (A12 in {'(-inf-0.22505]'}) AND (A46 in {'(-inf-0.07315]'}) AND (A35 in {'(0.19475-inf)'}) => (class = R) (29/29, 33.33%)
Rule 4. (A5 in {'(-inf-0.0392]'}) AND (A10 in {'(-inf-0.16315]'}) AND (A49 in {'(-inf-0.04525]'}) => (class = R) (25/25, 28.74%)
Rule 5. (A9 in {'(-inf-0.1164]'}) AND (A10 in {'(0.16315-inf)'}) AND (A28 in {'(-inf-0.9233]'}) AND (A49 in {'(0.04525-inf)'}) => (class = R) (3/3, 3.45%)
Rule 6. (A12 in {'(-inf-0.22505]'}) AND (A28 in {'(0.9233-inf)'}) AND (A35 in {'(-inf-0.19475]'}) => (class = R) (2/2, 2.3%)
Rule 7. (A5 in {'(-inf-0.0392]'}) AND (A10 in {'(-inf-0.16315]'}) AND (A4 in {'(0.052-inf)'}) AND (A47 in {'(0.06235-inf)'}) => (class = R) (3/3, 3.45%)
Rule 8. (A5 in {'(-inf-0.0392]'}) AND (A12 in {'(-inf-0.22505]'}) AND (A13 in {'(0.16265-inf)'}) => (class = R) (11/11, 12.64%)
Rule 9. (A47 in {'(-inf-0.06235]'}) AND (A4 in {'(-inf-0.052]'}) AND (A9 in {'(0.1164-inf)'}) AND (A35 in {'(0.19475-inf)'}) => (class = R) (13/13, 14.94%)
Rule 10. (A9 in {'(-inf-0.1164]'}) AND (A46 in {'(-inf-0.07315]'}) AND (A4 in {'(-inf-0.052]'}) AND (A47 in {'(0.06235-inf)'}) => (class = R) (8/8, 9.2%)
Rule 11. (A47 in {'(-inf-0.06235]'}) AND (A46 in {'(0.07315-inf)'}) AND (A10 in {'(0.16315-inf)'}) AND (A35 in {'(0.19475-inf)'}) => (class = R) (4/4, 4.6%)
Rule 12. (A9 in {'(-inf-0.1164]'}) AND (A12 in {'(-inf-0.22505]'}) AND (A49 in {'(0.04525-inf)'}) AND (A13 in {'(0.16265-inf)'}) AND (A47 in {'(0.06235-inf)'}) => (class = R) (6/6, 6.9%)
Rule 13. (A13 in {'(-inf-0.16265]'}) AND (A12 in {'(0.22505-inf)'}) AND (A5 in {'(0.0392-inf)'}) AND (A44 in {'(-inf-0.4271]'}) => (class = R) (2/2, 2.3%)
Rule 14. (A5 in {'(-inf-0.0392]'}) AND (A47 in {'(-inf-0.06235]'}) AND (A4 in {'(-inf-0.052]'}) AND

(A44 in {'(-inf-0.4271]'}) => (class = R) (12/12, 13.79%)
Rule 15. (A9 in {'(-inf-0.1164]'}) AND (A11 in {'(0.19795-inf)'}) AND (A46 in {'(0.07315-inf)'}) AND (A4 in {'(0.052-inf)'}) AND (A28 in {'(-inf-0.9233]'}) => (class = R) (1/1, 1.15%)
Rule 16. (A12 in {'(-inf-0.22505]'}) AND (A49 in {'(-inf-0.04525]'}) AND (A10 in {'(0.16315-inf)'}) AND (A35 in {'(0.19475-inf)'}) => (class = R) (7/7, 8.05%)
Rule 17. (A5 in {'(-inf-0.0392]'}) AND (A4 in {'(-inf-0.052]'}) AND (A28 in {'(-inf-0.9233]'}) AND (A12 in {'(0.22505-inf)'}) AND (A35 in {'(0.19475-inf)'}) => (class = R) (4/4, 4.6%)
Rule 18. (A12 in {'(-inf-0.22505]'}) AND (A11 in {'(0.19795-inf)'}) AND (A4 in {'(0.052-inf)'}) AND (A45 in {'(-inf-0.38545]'}) AND (A35 in {'(0.19475-inf)'}) AND (A54 in {'(-inf-0.0225]'}) => (class = R) (4/4, 4.6%)
Rule 19. (A10 in {'(-inf-0.16315]'}) AND (A49 in {'(-inf-0.04525]'}) AND (A4 in {'(0.052-inf)'}) AND (A9 in {'(0.1164-inf)'}) AND (A28 in {'(-inf-0.9233]'}) => (class = R) (4/4, 4.6%)
Rule 20. (A54 in {'(0.0225-inf)'}) AND (A5 in {'(0.0392-inf)'}) => (class = M) (17/17, 17.17%)
Rule 21. (A28 in {'(0.9233-inf)'}) AND (A10 in {'(0.16315-inf)'}) AND (A47 in {'(0.06235-inf)'}) => (class = M) (27/27, 27.27%)
Rule 22. (A45 in {'(0.38545-inf)'}) AND (A9 in {'(0.1164-inf)'}) => (class = M) (28/28, 28.28%)
Rule 23. (A54 in {'(0.0225-inf)'}) AND (A4 in {'(0.052-inf)'}) => (class = M) (14/14, 14.14%)
Rule 24. (A44 in {'(0.4271-inf)'}) AND (A11 in {'(0.19795-inf)'}) => (class = M) (20/20, 20.2%)
Rule 25. (A28 in {'(0.9233-inf)'}) AND (A45 in {'(0.38545-inf)'}) => (class = M) (8/8, 8.08%)
Rule 26. (A28 in {'(0.9233-inf)'}) AND (A11 in {'(0.19795-inf)'}) AND (A5 in {'(0.0392-inf)'}) AND (A10 in {'(-inf-0.16315]'}) => (class = M) (5/5, 5.05%)
Rule 27. (A4 in {'(0.052-inf)'}) AND (A9 in {'(-inf-0.1164]'}) AND (A49 in {'(-inf-0.04525]'}) AND (A10 in {'(0.16315-inf)'}) => (class = M) (1/1, 1.01%)
Rule 28. (A4 in {'(0.052-inf)'}) AND (A46 in {'(0.07315-inf)'}) AND (A35 in {'(-inf-0.19475]'}) => (class = M) (16/16, 16.16%)
Rule 29. (A4 in {'(0.052-inf)'}) AND (A5 in {'(0.0392-inf)'}) AND (A11 in {'(-inf-0.19795]'}) AND (A46 in {'(0.07315-inf)'}) AND (A49 in {'(-inf-0.04525]'}) => (class = M) (3/3, 3.03%)
Rule 30. (A28 in {'(0.9233-inf)'}) AND (A12 in {'(0.22505-inf)'}) AND (A11 in {'(-inf-0.19795]'}) => (class = M) (2/2, 2.02%)
Rule 31. (A4 in {'(0.052-inf)'}) AND (A49 in {'(0.04525-inf)'}) AND (A13 in {'(-inf-0.16265]'}) AND (A10 in {'(-inf-0.16315]'}) => (class = M) (2/2, 2.02%)
Rule 32. (A4 in {'(0.052-inf)'}) AND (A49 in {'(0.04525-inf)'}) AND (A12 in {'(0.22505-inf)'}) AND (A5 in {'(0.0392-inf)'}) AND (A10 in {'(-inf-0.16315]'}) => (class = M) (6/6, 6.06%)
Rule 33. (A35 in {'(-inf-0.19475]'}) AND (A10 in {'(0.16315-inf)'}) AND (A12 in {'(-inf-0.22505]'}) => (class = M) (4/4, 4.04%)
Rule 34. (A4 in {'(0.052-inf)'}) AND (A49 in {'(0.04525-inf)'}) AND (A9 in {'(0.1164-inf)'}) AND (A12 in {'(0.22505-inf)'}) AND (A10 in {'(0.16315-inf)'}) => (class = M) (22/22, 22.22%)
Rule 35. (A35 in {'(-inf-0.19475]'}) AND (A10 in {'(0.16315-inf)'}) AND (A5 in {'(-inf-0.0392]'}) AND (A47 in {'(0.06235-inf)'}) => (class = M) (5/5, 5.05%)
Rule 36. (A9 in {'(-inf-0.1164]'}) AND (A47 in {'(-inf-0.06235]'}) AND (A11 in {'(0.19795-inf)'}) => (class = M) (2/2, 2.02%)
Rule 37. (A49 in {'(0.04525-inf)'}) AND (A35 in {'(-inf-0.19475]'}) AND (A46 in {'(-inf-0.07315]'}) AND (A5 in {'(0.0392-inf)'}) => (class = M) (5/5, 5.05%)
Rule 38. (A4 in {'(0.052-inf)'}) AND (A46 in {'(0.07315-inf)'}) AND (A10 in {'(0.16315-inf)'}) AND (A11 in {'(-inf-0.19795]'}) => (class = M) (4/4, 4.04%)
Rule 39. (A4 in {'(0.052-inf)'}) AND (A46 in {'(0.07315-inf)'}) AND (A10 in {'(0.16315-inf)'}) AND (A12 in {'(0.22505-inf)'}) AND (A13 in {'(0.16265-inf)'}) AND (A47 in {'(0.06235-inf)'}) => (class = M) (22/22, 22.22%)
Rule 40. (A49 in {'(0.04525-inf)'}) AND (A11 in {'(0.19795-inf)'}) AND (A12 in {'(-inf-0.22505]'}) AND (A5 in {'(0.0392-inf)'}) AND (A4 in {'(-inf-0.052]'}) => (class = M) (3/3, 3.03%)
Rule 41. (A13 in {'(-inf-0.16265]'}) AND (A9 in {'(0.1164-inf)'}) AND (A46 in {'(0.07315-inf)'}) AND (A5 in {'(0.0392-inf)'}) AND (A4 in {'(-inf-0.052]'}) AND (A47 in {'(0.06235-inf)'}) => (class = M) (1/1, 1.01%)

In addition, MODLEM was applied for the discretized data without eliminating any attributes. 31 rules were generated with the accuracy of 80.7692%. Each rule in the rule set is longer length than the above.

### 3.2.3 Rules from the original data with frequent attributes only

After selecting the found 14 frequent attributes only, MODLEM was applied for the original data set. MODLEM generated 51 rules with the accuracy of

75% for the original data set. The followings are found rules:
Rule 1. (A11 < 0.09) AND (A9 < 0.13) => (class = R) (22/22, 22.68%)
Rule 2. (A9 < 0.03) => (class = R) (6/6, 6.19%)
Rule 3. (A12 >= 0.62) => (class = R) (2/2, 2.06%)
Rule 4. (A12 < 0.07) AND (A5 < 0.07) => (class = R) (10/10, 10.31%)
Rule 5. (A5 < 0.01) AND (A4 >= 0.02) => (class = R) (6/6, 6.19%)
Rule 6. (A12 < 0.17) AND (A47 < 0.05) => (class = R) (21/21, 21.65%)
Rule 7. (A9 < 0.09) AND (A46 < 0.03) => (class = R) (5/5, 5.15%)
Rule 8. (A9 < 0.11) AND (A5 < 0.04) AND (A4 >= 0.01) => (class = R) (22/22, 22.68%)
Rule 9. (A12 in [0.16, 0.17]) => (class = R) (4/4, 4.12%)
Rule 10. (A12 < 0.1) AND (A5 < 0.04) => (class = R) (14/14, 14.43%)
Rule 11. (A5 >= 0.15) AND (A4 < 0.04) => (class = R) (2/2, 2.06%)
Rule 12. (A35 >= 0.72) AND (A45 < 0.24) => (class = R) (16/16, 16.49%)
Rule 13. (A5 < 0.02) AND (A4 < 0.02) => (class = R) (3/3, 3.09%)
Rule 14. (A9 in [0.05, 0.05]) => (class = R) (2/2, 2.06%)
Rule 15. (A28 < 0.18) AND (A4 < 0.04) => (class = R) (2/2, 2.06%)
Rule 16. (A10 in [0.04, 0.07]) => (class = R) (10/10, 10.31%)
Rule 17. (A11 < 0.11) AND (A4 < 0.02) => (class = R) (16/16, 16.49%)
Rule 18. (A12 < 0.16) AND (A35 < 0.14) => (class = R) (5/5, 5.15%)
Rule 19. (A47 < 0.06) AND (A9 >= 0.34) => (class = R) (5/5, 5.15%)
Rule 20. (A49 >= 0.13) AND (A5 < 0.03) => (class = R) (1/1, 1.03%)
Rule 21. (A54 < 0) AND (A12 < 0.18) => (class = R) (3/3, 3.09%)
Rule 22. (A12 < 0.1) AND (A5 >= 0.11) => (class = R) (4/4, 4.12%)
Rule 23. (A28 < 0.3) AND (A4 < 0.02) => (class = R) (3/3, 3.09%)
Rule 24. (A47 in [0.06, 0.06]) => (class = R) (2/2, 2.06%)
Rule 25. (A49 in [0.03, 0.03]) => (class = R) (2/2, 2.06%)
Rule 26. (A44 in [0.11, 0.11]) => (class = R) (1/1, 1.03%)
Rule 27. (A54 >= 0.02) AND (A4 >= 0.02) => (class = M) (20/20, 18.02%)
Rule 28. (A28 >= 0.99) AND (A9 < 0.33) => (class = M) (16/16, 14.41%)
Rule 29. (A28 < 0.07) => (class = M) (2/2, 1.8%)
Rule 30. (A4 >= 0.15) AND (A9 < 0.2) => (class = M) (7/7, 6.31%)
Rule 31. (A28 in [0.96, 0.98]) => (class = M) (6/6, 5.41%)
Rule 32. (A11 >= 0.34) AND (A49 >= 0.05) => (class = M) (19/19, 17.12%)
Rule 33. (A13 >= 0.7) => (class = M) (2/2, 1.8%)
Rule 34. (A4 >= 0.06) AND (A9 < 0.08) => (class = M) (6/6, 5.41%)
Rule 35. (A35 < 0.13) AND (A44 >= 0.16) => (class = M) (17/17, 15.32%)
Rule 36. (A54 < 0) AND (A10 >= 0.16) => (class = M) (7/7, 6.31%)
Rule 37. (A11 >= 0.5) AND (A12 < 0.54) => (class = M) (6/6, 5.41%)
Rule 38. (A5 < 0.01) AND (A4 < 0.02) => (class = M) (1/1, 0.9%)
Rule 39. (A35 < 0.1) AND (A46 >= 0.08) => (class = M) (14/14, 12.61%)
Rule 40. (A10 < 0.03) AND (A4 >= 0.02) => (class = M) (3/3, 2.7%)
Rule 41. (A46 >= 0.37) AND (A4 >= 0.03) => (class = M) (12/12, 10.81%)
Rule 42. (A4 in [0.13, 0.13]) => (class = M) (1/1, 0.9%)
Rule 43. (A12 < 0.04) AND (A4 >= 0.04) => (class = M) (2/2, 1.8%)
Rule 44. (A47 >= 0.25) AND (A4 >= 0.03) => (class = M) (10/10, 9.01%)
Rule 45. (A28 >= 0.92) AND (A4 >= 0.04) => (class = M) (16/16, 14.41%)
Rule 46. (A35 < 0.19) AND (A44 >= 0.18) => (class = M) (22/22, 19.82%)
Rule 47. (A11 in [0.29, 0.3]) => (class = M) (8/8, 7.21%)
Rule 48. (A11 in [0.3, 0.31]) => (class = M) (5/5, 4.5%)
Rule 49. (A28 in [0.3, 0.36]) => (class = M) (7/7, 6.31%)
Rule 50. (A44 >= 0.31) AND (A28 >= 0.81) => (class = M) (19/19, 17.12%)
Rule 51. (A54 >= 0.02) AND (A28 < 0.45) => (class = M) (9/9, 8.11%)

We see some better results after selecting 14 frequent attributes only than those of using the original data untouched as summarized in table 4.

**Table 4.** The summary of the experiments on sonar data set

| | **Data set** | **Accuracy (%)** | **Number of** |
|---|---|---|---|

| | | | rules |
|---|---|---|---|
| No discretization | The original | 70.6731 | 59 |
| | Data set with frequent 14 attributes | 7.05 | 51 |
| After discretization | The original | 80.7692 | 31 |
| | Data set with frequent 14 attributes | 82.25 | 41 |

### 3.3 Data set 'postoperative'

For our third experiment, a data set called 'postoperative' from UCI machine learning depository is used. The data set is used to determine where patients in a postoperative recovery area should be sent to. There are three classes I(Intensive care unit), S(can go home), A(sent to general hospital), each having 2, 24, and 64 instances respectively. There are 8 conditional attributes. Seven conditional attributes have three nominal values, and one conditional attribute named 'comport' has an integer value between 0 and 20. Table 5 shows the meaning of the attributes.

**Table 5**. The attributes of data set 'postoperative'

| attribute | meaning |
|---|---|
| Lcore | Patient's internal temperature |
| Lsurf | Patient's surface temperature |
| Lo2 | Oxygen saturation in % |
| Lbp | Blood pressure |
| SURFstbl | Stability of patient's surface temperature |
| COREstbl | Stability of patient's core temperature |
| BPstbl | Stability of patient's blood pressure |
| comport | Patient's perceived comport at discharge |

#### 3.3.1 Rules from the original data

MODLEM generated 27 rules with the accuracy of 63.3333% from the original data set as follows:
Rule 1. (Lcore in {low}) AND (Lbp in {high}) AND (BPstbl in {stable}) AND (Lsurf in {mid}) => (class = I) (1/1, 50%)
Rule 2. (BPstbl in {unstable}) AND (Lsurf in {low}) AND (Lcore in {mid}) AND (Lo2 in {good}) => (class = I) (1/1, 50%)
Rule 3. (comport < 8.5) AND (Lsurf in {low}) => (class = S) (2/2, 11.11%)
Rule 4. (COREstbl in {unstable}) AND (BPstbl in {stable}) => (class = S) (3/3, 16.67%)
Rule 5. (comport >= 12.98) AND (Lcore in {high}) AND (Lo2 in {good}) => (class = S) (2/3, 11.11%)
Rule 6. (Lcore in {low}) AND (Lsurf in {mid}) AND (comport >= 10.48) => (class = S) (1/1, 5.56%)
Rule 7. (Lo2 in {excellent}) AND (Lcore in {low}) AND (BPstbl in {unstable}) => (class = S) (1/1, 5.56%)
Rule 8. (Lo2 in {excellent}) AND (comport >= 12.98) AND (SURFstbl in {unstable}) AND (Lcore in {mid}) => (class = S) (1/1, 5.56%)
Rule 9. (Lsurf in {high}) AND (SURFstbl in {stable}) AND (BPstbl in {stable}) => (class = S) (2/2, 11.11%)
Rule 10. (Lbp in {mid}) AND (BPstbl in {modStable}) AND (Lsurf in {low}) => (class = S) (1/1, 5.56%)
Rule 11. (BPstbl in {unstable}) AND (Lsurf in {low}) AND (SURFstbl in {stable}) AND (Lo2 in {excellent}) => (class = S) (1/1, 5.56%)
Rule 12. (BPstbl in {modStable}) AND (Lsurf in {mid}) AND (Lo2 in {good}) AND (Lbp in {mid}) => (class = S) (1/1, 5.56%)
Rule 13. (BPstbl in {unstable}) AND (Lsurf in {mid}) AND (SURFstbl in {unstable}) AND (Lcore in {mid}) AND (comport < 10.48) => (class = S) (1/1, 5.56%)
Rule 14. (Lo2 in {excellent}) AND (BPstbl in {stable}) AND (Lsurf in {mid}) AND (Lbp in {mid}) AND (SURFstbl in {stable}) AND (comport < 10.48) => (class = S) (2/2, 11.11%)
Rule 15. (Lsurf in {high}) AND (BPstbl in {modStable, unstable}) => (class = A) (11/11, 19.64%)
Rule 16. (Lbp in {low, high}) AND (BPstbl in {modStable}) AND (comport >= 6) => (class = A) (11/11, 19.64%)
Rule 17. (Lbp in {low}) => (class = A) (3/3, 5.36%)
Rule 18. (BPstbl in {unstable}) AND (Lsurf in {mid}) AND (SURFstbl in {stable}) => (class = A) (6/6, 10.71%)
Rule 19. (Lo2 in {excellent}) AND (SURFstbl in {unstable}) AND (comport < 12.98) AND (COREstbl in {stable}) AND (BPstbl in {stable}) => (class = A) (6/6, 10.71%)
Rule 20. (Lbp in {high}) AND (comport < 10.48) AND (Lsurf in {low}) AND (BPstbl in {stable, unstable}) => (class = A) (4/4, 7.14%)
Rule 21. (Lbp in {high}) AND (Lsurf in {mid}) AND (Lo2 in {excellent}) AND (Lcore in {mid})

=> (class = A) (4/4, 7.14%)
Rule 22. (Lcore in {low}) AND (Lsurf in {high}) => (class = A) (3/3, 5.36%)
Rule 23. (Lcore in {high}) AND (Lsurf in {mid}) AND (BPstbl in {unstable}) => (class = A) (3/3, 5.36%)
Rule 24. (comport >= 10.48) AND (BPstbl in {unstable}) AND (Lcore in {mid}) => (class = A) (4/4, 7.14%)
Rule 25. (Lsurf in {low}) AND (BPstbl in {stable}) AND (Lbp in {mid}) AND (comport >= 8.5) => (class = A) (6/6, 10.71%)
Rule 26. (comport >= 10.48) AND (SURFstbl in {stable}) AND (Lcore in {mid}) => (class = A) (5/5, 8.93%)
Rule 27. (SURFstbl in {unstable}) AND (Lcore in {mid}) AND (comport < 10.48) AND (BPstbl in {stable}) AND (Lo2 in {good}) => (class = A) (4/4, 7.14%)

### 3.3.2 Rules from frequent attributes only from the original data set

When minimum support ratio of 0.11 was supplied, all the six attributes except attribute Lo2 and SURFstbl were appeared. Discretization by Fayyad et al.'s method made the all the values of attribute 'comport' one nominal value, 'all'. Even though 'comport' attribute has one value, the attribute was not eliminated, because it's the only one having one value so that the combination with the other attribute values is meaningful. After dropping the two attributes from the original data set, MODLEM was applied, resulting in 19 rules with the accuracy of 66.6667%. The followings are found rules:
Rule 1. (Lcore in {low}) AND (Lbp in {high}) AND (BPstbl in {stable}) AND (Lsurf in {mid}) => (class = I) (1/1, 100%)
Rule 2. (comport < 8.5) AND (Lsurf in {low}) => (class = S) (2/2, 16.67%)
Rule 3. (COREstbl in {unstable}) AND (BPstbl in {stable}) => (class = S) (3/3, 25%)
Rule 4. (Lsurf in {high}) AND (BPstbl in {stable}) AND (Lbp in {high}) => (class = S) (2/2, 16.67%)
Rule 5. (comport >= 12.98) AND (Lcore in {high}) AND (Lsurf in {low}) => (class = S) (1/1, 8.33%)
Rule 6. (Lcore in {low}) AND (comport >= 10.48) AND (Lsurf in {mid}) => (class = S) (1/1, 8.33%)
Rule 7. (Lsurf in {high}) AND (BPstbl in {stable}) AND (Lcore in {mid}) AND (Lbp in {mid}) => (class = S) (1/1, 8.33%)
Rule 8. (Lcore in {low}) AND (BPstbl in {unstable}) AND (Lbp in {high}) => (class = S) (1/1, 8.33%)
Rule 9. (BPstbl in {modStable}) AND (Lsurf in {low}) AND (Lbp in {mid}) => (class = S) (1/1, 8.33%)
Rule 10. (Lsurf in {high}) AND (BPstbl in {modStable, unstable}) => (class = A) (11/11, 26.19%)
Rule 11. (Lbp in {low}) => (class = A) (3/3, 7.14%)
Rule 12. (Lsurf in {low}) AND (Lbp in {high}) AND (comport >= 6) AND (COREstbl in {stable}) => (class = A) (8/8, 19.05%)
Rule 13. (Lsurf in {low}) AND (BPstbl in {stable}) AND (Lbp in {mid}) AND (comport >= 8.5) => (class = A) (6/6, 14.29%)
Rule 14. (Lcore in {high}) AND (Lsurf in {mid}) AND (BPstbl in {unstable}) => (class = A) (3/3, 7.14%)
Rule 15. (Lsurf in {high}) AND (Lcore in {low}) => (class = A) (3/3, 7.14%)
Rule 16. (COREstbl in {unstable}) AND (BPstbl in {modStable, unstable}) => (class = A) (3/3, 7.14%)
Rule 17. (BPstbl in {modStable}) AND (Lbp in {high}) AND (Lsurf in {mid}) => (class = A) (4/4, 9.52%)
Rule 18. (BPstbl in {unstable}) AND (comport >= 10.48) AND (Lcore in {mid}) => (class = A) (4/4, 9.52%)
Rule 19. (Lcore in {low}) AND (BPstbl in {unstable}) AND (Lsurf in {mid}) AND (Lbp in {mid}) => (class = A) (1/1, 2.38%)

Applying MODLEM to the discretized data set with all the attributes generates 24 rules with the accuracy of 63.3333% that is three less number of rules with identical accuracy compared to the original data. Applying MODLEM to the discretized data set by eliminating the two infrequent attributes generates 18 rules with the accuracy of 65.5556%. Table 6 summarizes the results.

**Table 6.** The summary of the experiments on postoperative data set

| | **Data set** | **Accuracy (%)** | **Number of rules** |
|---|---|---|---|
| No discretization | The original | 63.3333 | 27 |
| | Data set with frequent 6 attributes | 66.6667 | 19 |
| After discretization | The original | 63.3333 | 24 |
| | Data set with frequent 6 attributes | 65.5556 | 18 |

# 4 Conclusion

Rough sets are one of widely accepted machine learning technologies because of their strong mathematical background and capability of finding minimal rules based on given data sets. Good property of rough set theory is that it can describe uncertain facts solely based on data. As a result, there is no room for prejudiced views to be inserted in the found knowledge from the data. But, because of this fact, it may become unstable algorithms. A machine learning algorithm becomes unstable when the performance of the algorithm is varying on given training data sets. The problem is also known as overfitting problem. Overfitting in rough set based algorithms may occur because they find rules very precisely, but, on the other hand, it may not be easy to find a training data set that covers its data space perfectly.

On the other hand, association rule algorithms find rules of association, where the association resides between sets of items in database. Association rule algorithms find itemsets that occur more than given minimum support. In other words, they can be used to fine attribute of frequent itemsets in the database.

In order to overcome the problem of overfitting in rough set based rule extraction algorithms, we find frequent itemsets using class association rule algorithm, then we select attributes that cover the frequent itemsets. By using the selected attributes only, we may find better set of rough set based rules in accuracy. Various experiments have been performed using public data sets in UCI machine learning repository to support our suggested method, and obtained good results.

*References:*


[1] Z. Pawlak, Rough Sets, *International Journal of Parallel Programming*, Vol. 15, No. 5, 1982, pp. 341-356.

[2] F. Fleuret, P. Abbet, C. Dubout, L. Lefakis, The MASH Project, *Lectures Notes in Computer Science*, Vol. 6913, 2011, pp. 626-629.

[3] M. Bal, Rough Sets Theory as Symbolic Data Mining method: An Application on Complete Decision Table, *Information Science Letters*, Vol. 2, No. 1, 2013, pp. 35-47.

[4] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*, 1993, pp. 207-216.

[5] J. Hipp, U. Guntzer, G. Nakhaeizadeh, Algorithms for association rule mining – a general survey and comparison, *ACM SIGKDD Explorations Newsletter*, Vol. 2, No. 58, 2000, pp. 58-64

[6] J. Li, H. Shen, R. Toper, Mining the optimal class association rule set, *Knowledge-Based Systems*, Vol. 15, 2002, pp. 399-405.

[7] P. Deekumpa, P. Sooraksa, Associate rule minimization using Boolean algebra set function, *Proceedings of 4th International Joint Conference on Information and Communication Technology, Electronic and Electrical Engineering* (JICTEE2014), 2014, DOI: 10.1109/JICTEE.2014.6804099.

[8] E.F. Ashmouni, R.A. Ramadan, A.A. Rashed, Espresso for Rule Mining, *Procedia Computer Science*, Vol. 32, 2014, pp. 596-603.

[9] H. Sug, Using Machine Learning Methods Jointly to Find Better Set of Rules in Data Mining, *Proceedings of 21st International Conference on Circuits, Systems, Communications, and Computers - CSCC 2017*, July 2017.

[10] J. Stefanowski, The rough set based rule induction technique for classification problems. *Proceedings of the 6th European Conference on Intelligent Techniques and Soft Computing*, EUFIT-98, 1998, pp. 109-113.

[11] L. Breiman, Bagging predictors, *Machine Learning*, Vol. 24, No. 2, 1996, pp. 123-140.

[12] J. Stefanowski, The bagging and n2-classifiers based on rules induced by MODLEM, *Lecture Notes in Artificial Intelligence*, Vol. 3066, Springer-Verlag, 2004, pp. 488-497.

[13] J. Stefanowski, On Combined Classifiers, Rule Induction and Rough Sets, *Lecture Notes of Computer Science*, Vol. 4374, Splinger-Verlag, 2007, pp. 329-350.

[14] T. G. Dietterich, An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, *Machine Learning*, Vol. 40, Issue 2, 2000, pp. 139-157.

[15] Quinlan, J.R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc., 1993.

[16] L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and Regression Trees*, Wadsworth International Group, 1984.

[17] J. Stefanowski, S. Wilk, Improving Rule-Based Classifiers Induced by MODLEM by Selective Pre-processing of Imbalanced Data,

*Proceedings of the International Workshop on Rough Sets in Knowledge Discovery: Foundations and Applications*, ECML 2007 – RSKD'07, 2007, pp. 54-65.

[18] T. Slimani, Class Association Rules Mining based Rough Set Method, *International Journal of Engineering and Technology*, Vol. 6, No. 6, 2015, pp. 2786-2794.

[19] S. Liao, H. Chang, A rough set-based association rule approach for a recommendation system for online customers, *Information Processing & Management*, Vol. 52, Issue 6, pp. 1142-1160, 2016.

[20] S.B. Sagar, A. Tiwari, Rough set and genetic based approach for maximization of weighted association rules, *International Journal of Education and Computer Science*, Vol. 8, No. 3, pp. 54-63, 2016.

[21] D.J. Leinweber, Stupid Data Miner tricks, *The Journal of Investing*, Vol. 16, No. 1, pp. 15-22. 2007.

[22] J. Brownlee, Overfitting and Underfitting with Machine Learning Algorithms, http://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/, 2016.

[23] A. Frank, A. Suncion, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Sciences, 2010.

[24] J. Stefanowski, The rough set based rule induction technique for classification problems, *Proceedings of 6th European Congress on Intelligent Techniques and Soft Computing*, pp. 109-113 1998.

[25] U. M. Fayyad, K.B. Irani, Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, *Proceedings of Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022-1027, 1993.