Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

SEL: RULE-BASED CLASSIFIER: RULES

Author:

Marc Pascual Roig

March 2024

Contents

1 Introduction

The aim of this work is to analyse and evaluate the rule-based classifier RULES[1] in different scenarios in order to get an insight of the results and performance of the algorithm through diverse datasets in terms of data size, number of features and type of features.

We will also test different configurations of the RULES algorithm with the objective to compare them.

2 Datasets

The datasets chosen for this project are the Lenses dataset, the Energy Efficient (ENB) dataset, the Dry Bean dataset and the Mushroom dataset. General information about them will be provided when discussing the applicability of the algorithm.

3 Pseudo-code

2 versions of the algorithm RULES will be used. The first one only adds rules if the rule is relevant (it classifies at least one instance that is not already classified). The second one adds all rules that classify at least one instance of the whole dataset regardless whether it is already classified. Both algorithms differ from the original algorithm in [1], since we do not generate rules for the instances that have not been classified. Observe that the only instances that are not classified after all conditions have been checked are the ones that share the same values but have different classes assigned, so there is no sense in adding these rules for prediction purposes.

Algorithm 1 Rule Generation Algorithm

Require: NumberAttributes, Instances
Ensure: Rules

NumberCombinations $\leftarrow$ 1
Rules $\leftarrow \varnothing$
while (NumberCombinations $\leq$ NumberAttributes) **and** not All instances are classified **do**
    Find all selectors (pairs Attribute-Value) from NON-classified instances
    Form conditions as a combination of NumberCombinations selectors
    for each Condition $_i$ **do**
        if all instances satisfying the condition belong to the same class C_i **then**
            Create the rule R: Condition $i \rightarrow C_i$
            Check irrelevant conditions against all previously obtained rules
            Rules $\leftarrow$ Rules + R
        end if
    end for
    NumberCombinations $\leftarrow$ NumberCombinations +1
end while
return Rules

Algorithm 2 Rule Generation Algorithm with extra rules

```
Require: NumberAttributes, Instances
Ensure: Rules
NumberCombinations ← 1
Rules ← ∅
while (NumberCombinations ≤ NumberAttributes) and not All instances are classified do
    Find all selectors (pairs Attribute-Value) from NON-classified instances
    Form conditions as a combination of NumberCombinations selectors
    for each Condition_i do
        if all instances satisfying the condition belong to the same class C_i then
            Create the rule R: Condition i → C_i
            Rules ← Rules + R
        end if
    end for
    NumberCombinations ← NumberCombinations +1
end while
return Rules
```

4 Methodology

The general procedure will consist of different configurations of the fit and predict method of the algorithm, which will be performed on the above commented datasets.

The data is shuffled with a fixed seed for reproducibility and fairness in the comparison. We use a 70% - 30% split for the training and test sets, respectively. Additionally, we note that datasets containing continuous features will be processed accordingly. Any preprocessing steps will be indicated in the corresponding section, as RULES only works with discrete values.

For each dataset, the following will be considered as possible configurations of the algorithm. Regarding the *fit* method and the definition of the rules:

1. Only meaningful rules in the training dataset are considered (eliminate irrelevant rules) - Algorithm 1.
2. Skip the step that checks if the rule is irrelevant - Algorithm 2.

The second configuration makes the algorithm order-independent and will potentially cover more instances in the test set without a cost in the training time since all rules are checked anyways. It obviously comes with the drawback of generating much more rules, which will make the understanding of the predictions harder.

Regarding the *predict* method and the prediction of the test instances:

1. Rules are applied sequentially and only to not yet classified instances.
2. If several rules apply to one single instance all predictions are saved and the final prediction consists of the most voted class by the rules. In case of a tie, the instance is left as "Unlabeled".

Notice that the second approach is not necessarily a safer one although it seems to be more cautious with instances that get contradictory labels from different rules. However, this is not granted since RULES generates rules with only not yet classified instances and therefore the rules that are generated first should be more important. Hence, the second approach can lead to mistakes that otherwise would not happen.

That yields a total of 4 configurations:

1. Only relevant rules and rules are applied sequentially.
2. Only relevant rules and all predictions are taken into account.
3. All rules and rules are applied sequentially.
4. All rules and all predictions are taken into account.

5 Metrics

The complexity of the different approaches will be evaluated according to:

- Number of rules generated.
- Number of rules seen.
- Training time.
- Prediction time.

Notice that the number of rules that RULES evaluates as candidate rules is an indicator of the complexity of the dataset. It mainly depends on the number of instances, the number of features, the number of unique values inside each feature, and the dependencies between features in order to determine the target class - since the number of rules explored increases exponentially with the number of conditions in the rules. Actually, this is the main limitation of RULES. In general, the number of rules that are evaluated becomes unfeasible when 4-5 conditions are required in the antecedent - it depends on the number of features and unique values per feature, too.

For the performance, the following will be taken into account:

- **Accuracy**. The number of instances correctly classified over the number of instances classified in the test set. RULES generates rules that are 100% precise in the training set.
- Percentage of **unclassified instances**. Instances might be left as unclassified either because the instance does not satisfy any antecedent or because we take the approach that discards instances when contradictory rules apply to the same instance.
- **Precision, coverage and recall** for each of the rules. In the context of the RULES algorithm and the different configurations we will indicate the metrics in 2 different formats, depending on how we consider whether an instance needs to be taken into account. In the first format we consider only instances that apply to the rule in the sequential approach - all instances that have been already classified will be discarded. In the second format we consider that all instances will be taken into account independently whether they have been already classified by a previous rule. These metrics intend to assess the sequential and the voting approach, respectively.

6 Results

6.1 Lenses dataset

Lenses dataset is a simple and short dataset with 24 instances and 4 categorical features. It will be useful to view how the different approaches perform and to see how the complexity of the algorithms will scale with larger datasets, in particular the number of rules.

Model	# Rules	Train Time (s)	Predict Time (s)	Accuracy (%)	Unclassified (%)
1	7	0.0	0.0	62.5	0.0
2	7	0.0	0.0	66.7	62.5
3	45	0.0	0.0	75.0	0.0
4	45	0.0	0.0	83.3	25.0

Table 1: Comparison of Models

The total number of rules that have been evaluated is 83.

The resulting 7 rules for the "only relevant rules" approach are the following. In Table 2 and 3 the metrics for each of the rules in the test set are shown.

- RULE 1: IF TearProdRate=reduced THEN ContactLens=none
- RULE 2: IF Age=pre-presbyopic AND Astigmatism=no THEN ContactLens=soft
- RULE 3: IF Age=presbyopic AND TearProdRate=normal THEN ContactLens=hard
- RULE 4: IF Astigmatism=no AND TearProdRate=normal THEN ContactLens=soft
- RULE 5: IF Age=pre-presbyopic AND SpectaclePrescrip=hypermetrope AND Astigmatism=yes THEN ContactLens=none
- RULE 6: IF Age=young AND SpectaclePrescrip=hypermetrope AND TearProdRate=normal THEN ContactLens=hard
- RULE 7: IF SpectaclePrescrip=myope AND Astigmatism=yes AND TearProdRate=normal THEN ContactLens=hard

Rule	Precision (%)	Coverage (%)	Recall (%)
1	100	37.5	60.0
2	/	0.0	0.0
3	0.0	37.5	0.0
4	100	12.5	50.0
5	/	0.0	0.0
6	/	0.0	0.0
7	100	12.5	100

Table 2: Metrics of the rules in the sequential approach.

From the analysis of the metrics we see that all the error for the first approach comes from rule 3, which completely missclassifies all instances in the test set. The second approach does not improve the results much and fails to classify most of the instances since the number of rules is very reduced and there is a high number of ties (1 rule is voting for class A and another rule is voting for class B). For instance, now 6 of the instances are covered both by rule 1 and 2 with

Rule	Precision (%)	Coverage (%)	Recall (%)
1	100	37.5	60.0
2	0.0	25.0	0.0
3	0.0	37.5	0.0
4	66.7	37.5	100.0
5	/	0.0	0.0
6	0.0	12.5	100
7	100	12.5	100

Table 3: Metrics of the rules in the voting approach (for the same 7 rules).

different conclusions. Observe that rule 2 is not used in the prediction stage in the sequential approach, but it is used in the voting approach (and all predictions are bad).

The third and fourth approach manage to improve the accuracy because more rules are considered, which mitigate the impact of rule 3 in the results. Furthermore, we have much less ties because more rules apply to each instance and the percentage of unclassified instances is reduced significantly.

Notice that the main problem is that the training set is not enough representative and that led to generating a bad rule. Therefore, the approach that considers a voting scheme is arguably more useful if a bad rule is generated at some point because a rule on its own is less meaningful.

6.2 Energy efficient dataset

The Energy efficient dataset consists of information about buildings related to energy purposes. It contains 768 instances and 8 categorical - actually they are numerical but already discrete - features, and 2 continuous numerical target variables. So for this case, we need to pre-process the target variable. We will use only one target variable - "Heating load" - for the classification problem, which has been divided into 15 equally spaced buckets: (5.973, 8.483], ..., (40.627, 43.1]. The metrics are shown in Table 4.

Model	# Rules	Train Time (s)	Predict Time (s)	Accuracy (%)	Unclassified (%)
1	160	421.94	1.2	89.3	6.9
2	160	421.94	1.4	89.4	13.9
3	1994	396.66	18.0	82.5	1.3
4	1994	396.66	19.5	86.0	6.9

Table 4: Comparison of Models

The number of rules evaluated is 102087.

We observe that for this dataset the approaches that consider less rules perform slightly better and are able to cover slightly less instances in the test set. It is hard to understand the cause of the decrease in the performance when all the rules are taken into consideration - since 1994 rules have been generated -, but it is arguably due to later rules affecting the first rules results. As expected, when more rules are considered the coverage in the test set increases. Finally, we also observe that the approach that considers the voting scheme performs better in both cases at the cost of leaving out more instances.

The following rules are the ones that achieve a greater coverage and their respective metrics in the train set.

Rule	Conditions	Y1_buckets	Precision	Coverage	Recall
1	X1=0.62 AND X7=0.1	(10.955, 13.428]	100.00%	2.98%	17.02%
2	X1=0.62 AND X7=0.4	(15.901, 18.373]	100.00%	2.79%	44.12%
3	X1=0.62 AND X7=0.25	(13.428, 15.901]	100.00%	2.61%	17.07%
7	X1=0.66 AND X7=0.1	(10.955, 13.428]	100.00%	2.05%	11.70%
8	X1=0.66 AND X7=0.4	(13.428, 15.901]	100.00%	2.79%	18.29%
9	X1=0.66 AND X7=0.25	(10.955, 13.428]	100.00%	3.17%	18.09%
14	X1=0.86 AND X7=0.4	(30.737, 33.209]	100.00%	2.98%	30.77%
15	X1=0.86 AND X7=0.25	(28.264, 30.737]	100.00%	2.79%	29.41%
27	X1=0.79 AND X7=0.25	(38.155, 40.627]	100.00%	2.98%	55.17%
26	X1=0.79 AND X7=0.4	(40.627, 43.1]	100.00%	2.42%	76.47%
29	X1=0.74 AND X7=0.1	(8.483, 10.955]	100.00%	2.42%	40.62%
30	X1=0.74 AND X7=0.4	(13.428, 15.901]	100.00%	2.05%	13.41%

Table 5: Rules with greater coverage in the train set. The metrics are computed according to the sequential approach.

Rules that use 3 or more conditions in the antecedent have a coverage below 1%.

The precision and coverage of these rules in the test set oscillate between 80%-100% and 1%-3%, respectively, see Table 6.

Rule	Conditions	Y1_buckets	Precision	Coverage	Recall
1	X1=0.62 AND X7=0.1	(10.955, 13.428]	100.00%	1.73%	8.00%
2	X1=0.62 AND X7=0.4	(15.901, 18.373]	80.00%	2.16%	28.57%
3	X1=0.62 AND X7=0.25	(13.428, 15.901]	100.00%	2.60%	15.00%
7	X1=0.66 AND X7=0.1	(10.955, 13.428]	100.00%	3.90%	18.00%
8	X1=0.66 AND X7=0.4	(13.428, 15.901]	100.00%	2.16%	12.50%
9	X1=0.66 AND X7=0.25	(10.955, 13.428]	100.00%	1.30%	6.00%
14	X1=0.86 AND X7=0.4	(30.737, 33.209]	100.00%	1.73%	19.05%
15	X1=0.86 AND X7=0.25	(28.264, 30.737]	100.00%	2.16%	25.00%
27	X1=0.79 AND X7=0.25	(38.155, 40.627]	100.00%	1.73%	57.14%
26	X1=0.79 AND X7=0.4	(40.627, 43.1]	100.00%	3.03%	100.00%
29	X1=0.74 AND X7=0.1	(8.483, 10.955]	100.00%	3.03%	50.00%
30	X1=0.74 AND X7=0.4	(13.428, 15.901]	100.00%	3.90%	22.50%

Table 6: Metrics of the same rules in the test set.

Observe that we have the same results for the metrics in both approaches since the rules have clear disjoint antecedents and hence all these rules apply to different instances.

Rule	Conditions	Y1_buckets	Precision	Coverage	Recall
87	IF X1=0.9 AND X6=3 AND X7=0.1	(28.264, 30.737]	0.00%	0.43%	0.00%
88	IF X1=0.9 AND X6=3 AND X8=4	(35.682, 38.155]	0.00%	0.43%	0.00%

Table 7: Example of rules and their respective metrics in the test set (according to the sequential approach).

Rule	Conditions	Y1_buckets	Precision	Coverage	Recall
87	IF X1=0.9 AND X6=3 AND X7=0.1	(28.264, 30.737]	0.00%	0.43%	0.00%
88	IF X1=0.9 AND X6=3 AND X8=4	(35.682, 38.155]	0.00%	0.87%	0.00%

Table 8: Example of rules and their respective metrics in the test set (according to the voting approach).

The main cause of errors is due to rules that only classify one instance (see Table 7 and 8). Notice that 160 rules are generated from the 538 instances of the train set - which is a lot - because there are few instances that share the same values for each feature. Therefore, many of the rules generated have been validated with only one or few instances.

From the rules we also infer that the features that are actually more relevant - X1 and X7 - appear much more frequently in the antecedents of the rules. Actually, the first 32 rules generated only use these 2 variables. For the "all rules" approach we find rules that use other variables, nevertheless the presence of X1 and X7 is still remarkable.

6.3 Dry bean dataset

Next we evaluate the Dry bean dataset. We will use a reduced version of 4000 instances randomly chosen. It contains 16 continuous numerical features and 6 possible classes as target: Seker, Barbunya, Bombay, Cali, Dermosan, Horoz and Sira.

In order to make the dataset suitable for the RULES algorithm we first discretize the values of each feature into 6 equally spaced buckets that range from the minimum value to the maximum value of the feature.

To deal with time issues related to the number of rules seen (view Graphic 1) we only generated rules up to 3 combinations and stopped, so the algorithm did not terminate. We managed to classify 1639 out of the 2800 instances in the training set.

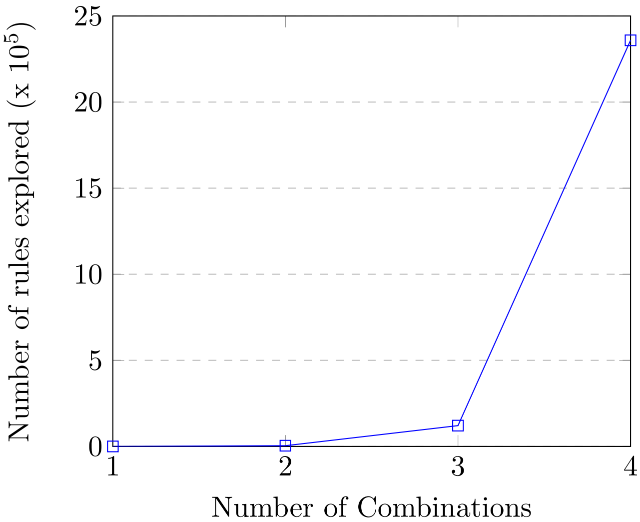

Figure 1: Relationship between the number of combinations and the number of rules seen by the algorithm.

Model	# Rules	Train Time (s)	Predict Time (s)	Accuracy (%)	Unclassified (%)
1	211	287.1	0.6	94.0	40.8
2	211	287.1	0.6	95.1	41.8
3	19083	277.01	52.7	93.1	39.7
4	19083	277.01	51.8	93.9	40.1

Table 9: Comparison of Models

The number of explored rules is 120960. The following rules are the ones that achieve a greater coverage in the train set and therefore are arguably more relevant in the decision process:

Rule	**Condition**
1	Area_buckets=(123617.5, 157974.0]
2	Area_buckets=(157974.0, 192330.5]
9	Compactness_buckets=(0.64, 0.697]
24	Area_buckets=(20341.861, 54904.5] AND Eccentricity_buckets=(0.479, 0.587]

Rule	**Class**	**Precision**	**Coverage**	**Recall**
1	BOMBAY	100.00%	1.21%	29.31%
2	BOMBAY	100.00%	2.11%	50.86%
9	HOROZ	100.00%	4.11%	27.91%
24	SEKER	100.00%	5.21%	35.87%

Table 10: Rules and their respective metrics in the train set (sequential approach).

Rule	**Class**	**Precision**	**Coverage**	**Recall**
1	BOMBAY	100.00%	1.25%	34.09%
2	BOMBAY	100.00%	1.75%	47.73%
9	HOROZ	100.00%	5.17%	37.80%
24	SEKER	96.97%	5.50%	33.16%

Table 11: Rules and their respective metrics in the test set (sequential approach).

Rule	**Class**	**Precision**	**Coverage**	**Recall**
1	BOMBAY	100.00%	1.25%	34.09%
2	BOMBAY	100.00%	1.75%	47.73%
9	HOROZ	100.00%	7.67%	56.10%
24	SEKER	97.50%	6.67%	40.41%

Table 12: Rules and their respective metrics in the test set (voting approach).

The results obtained reassemble the ones of the *ENB* dataset. On the one hand, the voting scheme performs slightly better than the sequential approach - and the model with few rules performs better, too. Notice that the number of rules generated is surprisingly high (19083) and it is impossible to analyse the cause of the decrease in the accuracy with this approach. Again, we conclude that the "all rules" configuration manages to classify more instances, but they are badly classified and hence the lower accuracy.

From the rules we also infer that the "Area" feature is more decisive tha other features. However, there are lots of rules and it does not exist a simple way to classify the instances, but many conditions and values have to be considered in the prediction.

6.4 Mushrooms dataset

This data set includes descriptions of hypothetical samples (8124 instances) corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended.

No pre processing of the data has been done. Notice that the dataset contains missing values, however, it does not suppose an inconvenient since no rules with missing values are generated.

Model	# Rules	Train Time (s)	Predict Time (s)	Accuracy (%)	Unclassified (%)
1	71	6.6	0.2	100.0	0.0
2	71	6.6	0.2	100.0	0.0
3	1994	7.1	5.6	100.0	0.0
4	1994	7.1	5.6	100.0	0.0

Table 13: Comparison of Models

We observe in Table 13 that all configurations of RULES lead to a flawless performance managing to classify all instances. We see in Table 14 that the rules generated are simple (rules contain either 1 or 2 conditions in the antecedents) and achieve great values of coverage and recall. We expect that rules with high coverage and recall are more consistent than rules with lower values since more instances support the rule.

The following rules are the ones that achieve a greater coverage in the train set and therefore are arguably more relevant in the decision process:

Number	**Rule**	**Precision**	**Coverage**	**Recall**
6	IF odor=f THEN class=p	100.00%	27.19%	56.30%
7	IF odor=c THEN class=p	100.00%	2.25%	4.66%
8	IF odor=s THEN class=p	100.00%	7.32%	15.15%
9	IF odor=y THEN class=p	100.00%	7.12%	14.75%
10	IF odor=l THEN class=e	100.00%	5.03%	9.73%
11	IF odor=a THEN class=e	100.00%	4.92%	9.52%
12	IF odor=p THEN class=p	100.00%	2.80%	5.79%
17	IF stalk-color-above-ring=g THEN class=e	100.00%	6.77%	13.10%
21	IF stalk-color-below-ring=g THEN class=e	100.00%	4.70%	9.08%
27	IF population=a THEN class=e	100.00%	4.50%	8.71%
48	IF cap-surface=f AND bruises=t THEN class=e	100.00%	2.69%	5.20%

Table 14: Rules and their respective metrics in the train set.

These rules cover more than 80% of the instances and therefore we are able explain pretty well the decision process with them. In particular, there is one specific attribute that above half of the "poisonous" mushrooms have (odor = f). For the "edible" mushrooms, it exists a limited and simple range of attributes that precisely classify a mushroom as "edible", too. Therefore, there is no much to say apart from remarking that the classes are easily determined with only using rules of at most 2 conditions. In such cases - when the rules are super precise and the trains set covers well all the space -, it is much preferable the "Only relevant rules" approach since we achieve the same results with a significantly lower number of rules, but in terms of performance there is no difference.

7 Conclusions

Overall, we have seen that all configurations of the RULES algorithm perform quite consistent through diverse datasets - the performance depending on the complexity of the features and the diversity of values. They also perform quite similar in terms of accuracy.

Nevertheless, one should consider the following situations with special emphasis. First, rules that do not achieve great coverage or rules that are validated with few instances have more chances to fail in the prediction. In such cases, it might be interesting to study the usage of the voting approach and even consider all the possible rules. This is a possible approach to deal with lack of samples in order to generalise better the results as well. Secondly, the search space increases exponentially with the number of conditions. Therefore, dimensionality reduction techniques are also a possibility for complex datasets, or even finishing the algorithm at some certain degree of coverage. Other things to take into account are defining the buckets appropriately for continuous variables.

Finally, our results indicate that the voting approach performs slightly better than the sequential approach - but on the other hand it has more unclassified instances. Besides, the "only relevant rules" approach seems to perform better when the training set is enough representative and covers all the space.

References

[1] D.T. Pham and M.S. Aksoy. Rules: A simple rule extraction system. *Expert Systems with Applications*, 8(1):59–65, 1995.