Universitat Politècnica de Catalunya

Supervised and Experiential Learning

Practical Work 1

A Rule-based Classifier PRISM

Author:

Clara Rivadulla Duró

Spring Semester

2022-23

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Contents

1 Introduction

For the first practical work of the course, the main purpose is to implement a rule-based classifier. As my $NumStud = 17$, the $NumCla$ to implement is 1 ($NumCla = NumStud \bmod 4$), which corresponds to the PRISM algorithm.

The PRISM Algorithm was introduced by Cendrowska in 1987 in his paper *PRISM: An Algorithm for Inducing Modular Rules* [1] as an alternative to Quinlan's ID3 algorithm [2] that, although based on ID3, uses a different induction strategy to induce rules which are modular, thus avoiding many of the problems associated with decision trees.

PRISM is a rule induction algorithm that learns a set of modular rules from a given dataset. The algorithm works by recursively partitioning the data into subsets based on the values of the attributes, and then inducing a set of rules for each partition. These rules are then combined to form a global set of rules that cover the entire dataset.

2 Methodology

2.1 Pseudo-code

The pseudo-code used for the implementation of the PRISM algorithm is the one available in the slides of the course [3], based on Cendrowska's paper.

Algorithm 1
PRISM

```
Prism ← $\emptyset$
for each class $C_i$ do do
    E ← set of instances of class $C_i$
    while E $\neq \emptyset$ do
        Create rule R: $\emptyset \rightarrow C_i$
        while not R is perfect and attributes are available do
            for each pair attribute-value (A-V) not appearing in R do
                Form $R'_{\mathrm{a-v}}$ extending the rule R adding the condition $A = V$ in the antecedent
                Compute the precision $p/t$ (positive/total) of $R'_{\mathrm{a-v}}$
            end for
            Select $R'_{\mathrm{op}}$ as the $R'_{\mathrm{a-v}}$ maximizing the precision $p/t$ in all instances ▷ if there is a tie, select the one with higher $p$
            R ← $R'_{\mathrm{op}}$
        end while
        E ← E - instances covered by the rule R
        Prism ← Prism + R
    end while
end for
return (Prism)
```

As a summary, the algorithm begins with an empty set of rules, *Prism*. Then, for every class C_i of a dataset, we instantiate E as the set of instances of such class. While this set is not empty, we keep defining rules until all instances of E are covered by them.

A rule is defined by checking all the available attributes (initially, all of them) and its possible values. For an attribute A, we try to find the value V that maximizes the precision p of the rule. If there's a tie, we select the one with the higher p. We repeat this process for every attribute, until there are no available attributes or the rule is perfect (the precision is maximum).

2.2 Selection of the Data Sets

The data sets have been chosen according to the restrictions given: choose 3 different data sets, one of small size ($\#instances \leq 500$), one of medium size ($500 < \#instances \leq 2000$) and one of large size ($\#instances > 2000$).

Besides, as the PRISM algorithm can only deal with qualitative attributes, we've also avoided data sets that contain numeric or non categorical values.

So, taking all of this into consideration, the 3 selected data sets for testing the algorithm are, for each mandatory size:

Congressional Voting Records Data Set [4] (*Small*) It contains the voting records of members of the US House of Representatives from the 98th to the 101st Congress (1983-1990). The data set includes *435 instances*, one for each member of the House, and *16 attributes*, including 15 binary attributes representing the members' votes on various bills and a class label indicating the political party of each member: 'democrat' or 'republican'.

Tic-Tac-Toe Endgame Data Set [5] (*Medium*) It contains *958 instances* representing all possible endgame scenarios for the classic game of Tic-Tac-Toe. The data set has *9 attributes* representing each square on the board, with three possible values: x (representing the player who put an "X" on the square), o (representing the player who put an "O" on the square), and b (representing a blank square). The goal of the data set is to predict the class label for each instance, which represents the outcome of the game: 'positive' or 'negative'.

Mushroom Data Set [6] (*Large*) It contains information about various characteristics of mushrooms, including their physical attributes, habitat, and edibility. The data set has *8124 instances* and *23 attributes*, including a class label indicating whether each mushroom is 'edible' or 'poisonous'.

Additionally, another medium-sized dataset has been tested to counter the mediocre results obtained with the Tic-Tac-Toe Endgame Data Set (explained in subsection 3.2).

Car Evaluation Data Set [7] (*Medium*) It contains information about various attributes of cars and their acceptability. The data set has *1728 instances* and *6 attributes*, including a class label indicating the acceptability of the car, that can take one of four values: 'unacc', 'acc', 'good', and 'vgood'.

The datasets have been previously preprocessed in order to delete rows with missing values, as well as to change the name of the class column.

2.3 How to execute the code

Before running the code for the first time, you must change the `path` written in the 8th line of the `main.py` file to the one where you've stored the folder containing the unzipped project. Once the path is changed, you must follow the next steps:

1. Open the folder containing the code of the project (*source*) in the terminal
   ```
   cd <root_folder_of_project>/source
   ```

2. Create a virtual environment using Python
   ```
   python3 -m venv venv/
   ```
3. Open the virtual environment
   ```
   source venv/bin/activate
   ```
4. Install the required dependencies
   ```
   pip install -r requirements.txt
   ```
5. Run the main file of the project
   ```
   python main.py
   ```

3 Results

In this section, we discuss the results obtained for every data set. To evaluate them, we define 3 different metrics for every given rule R:

- *Precision (R)*: ratio between the #instances satisfying the antecedent of R and the consequent of R, and the #instances satisfying the antecedent of R.
- *Coverage (R)*: ratio between the #instances satisfying the antecedent of R, and the #Total instances in the training dataset.
- *Recall (R)*: ratio between the #instances satisfying the antecedent of R and the consequent of R, and the #Total instances in the training dataset belonging to the same class label than R is classifying.

The Precision metric has been computed both for the train and the test datasets. As expected, the Precision of all the rules is always the maximum (a 100%) when it comes to the train set, which shows that the algorithm is indeed working properly, but can vary a lot or even be null (0%) in the test set, due to the datasets being randomly split. So, when discussing the Precision, we're going to be referring to the Precision obtained in the test set. Moreover, the model is evaluated with an overall accuracy measure, defined as:

- *Accuracy*: percentage of test instances correctly classified. $\frac{\#instances\ correctly\ classified}{Total\ \#instances}$

Next, we show a general comparison of the characteristics of the datasets and the accuracies obtained for each one of them (Table 1).

Dataset	# Classes	# Attributes	Train size	Test size	# Rules	Accuracy
Congressional Voting Records	2	16	185	47	14	97.87%
Tic Tac Toe Endgame	2	9	766	192	148	71.88%
Car Evaluation	4	6	1382	346	231	84.39%
Mushroom	2	22	4515	1129	45	99.82%

Table 1: Comparison between the characteristics, rules and accuracies of every dataset.

Just by looking at this table, we can get some interesting insights. First, we can see that the factors that seem to play a crucial role when the accuracy is high are: a larger number of attributes, and a smaller number of rules. This is the case of the *Congressional Voting Records* and the *Mushroom* datasets, with which we obtain the higher accuracies (97.87% and 99.82%, respectively). Second, we cannot say that the size of the datasets is specially relevant, at least with our selected datasets, for both the one with more instances and the one with fewer instances give outstanding results. Third, we're not able to confirm that the number of classes is influential in any way.

Now, given the insanely large number of rules obtained for some datasets, we've found it impossible to delve into all of them, so we're going to focus only on the most relevant ones. Table 2 gives us an overview of the best three rules of every dataset (the complete tables of results can be found in Appendix A).

Congressional Voting Records Data Set				
Rule	Class	Precision	Recall	Coverage
(1) if physician-fee-freeze == y and synfuels-corporation-cutback == n then	republican	100.0	84.09	40.0
(2) if synfuels-corporation-cutback == y and physician-fee-freeze == y and adoption-of-the-budget-resolution == y and anti-satellite-test-ban == n then	democrat	100.0	1.03	0.54
(3) if physician-fee-freeze == n then	democrat	96.15	95.88	50.27
Tic-Tac-Toe Endgame Data Set				
Rule	Class	Precision	Recall	Coverage
(1) if V5 == o and V1 == o and V9 == o then	negative	100.0	15.33	5.22
(2) if V5 == o and V3 == o and V7 == o then	negative	100.0	14.56	4.96
(3) if V5 == x and V9 == x and V1 == x then	positive	100.0	13.66	9.01
Car Evaluation Data Set				
Rule	Class	Precision	Recall	Coverage
(1) if safety == low then	unacc	100.0	48.8	33.94
(2) if persons == 2 and safety == high then	unacc	100.0	15.71	10.93
(3) if safety == med and persons == 2 then	unacc	100.0	14.98	10.42
Mushroom Data Set				
Rule	Class	Precision	Recall	Coverage
(1) if odor == f then	p	100.0	73.65	28.48
(2) if stalk-color-above-ring == g then	e	100.0	17.12	10.5
(3) if stalk-color-above-ring == p and bruises == t then	e	100.0	16.58	10.17

Table 2: Best 3 rules for every data set, sorted by descending Precision and Recall.

If we have a look at the best rule of every data set, we can see that the most complete rule is the one from the *Congressional Voting Records Data Set*, with an 84.09% of Recall and a 40.00% of Coverage, followed by the one from the *Mushroom Data Set*, with a 73.65% of Recall and a 28.48% of Coverage. The worst best rule belongs to the *Tic-Tac-Toe Endgame Data Set*, with a 15.33% of Recall and a 5.22% of Coverage. Luckily, almost all of them are a 100% precise in the test set, and if not, very close to a 100%, so no instances are being misclassified with the best rules.

These observations are no surprise, since they seem to have a strong correlation with the overall accuracies shown previously in Table 1. Having a few strong rules could be decisive for the outcome of our rule-based model.

Now, let's dig a bit deeper into the individual results for each data set.

3.1 Congressional Voting Records

As we can see in Table 1, 2 out of the 3 best rules correspond to the 'democrat' class. Rule (2), tho, has a very small Recall (1.03%) and Coverage (0.54%), despite the 100% of Precision. This means that the rule represents a very small fraction of the dataset, being still strong. Rules (1) and (3), have a high Precision, Recall, and Coverage, so they precisely represent a considerable amount of instances in the dataset: Rule (1) represents an 84.09% of the 'republican' class, and Rule (2) a 95.88% of the 'democrat' class, which leads us to the conclusion that the 'physician-fee-freeze' attribute (whether the member voted to pass the bill that would freeze the fees paid to doctors) is decisive to determine the political nature of a member in the US house.

Regarding the rest of the results (A.1), there are no more relevant rules. Except for the fourth rule, with a 60.0% of Precision, a 7.95% of Recall and a 3.78% of Coverage, the rest of the rules (from (5) to (14), a 28.60%) are useless in the test dataset, as they are zero precise. This would be a problem in case that the Coverage or Recall were large, but as they're small they don't seem to be causing many misclassifications.

3.2 Tic-Tac-Toe Endgame

For the Tic-Tac-Toe Endgame dataset, the three best rules are as precise as possible, but don't cover a considerable fraction of the dataset. Rule (1), that being the best rule, has only a 15.33% of Recall (Table 2). This could explain why the accuracy achieved with this dataset is the lowest (71.88%, as seen in Table 1). Not having solid and general rules makes it difficult to correctly classify instances of unseen data.

Now, if we look at the full results (A.2), both classes are represented almost equally, since there are rules with similar Precision and Recall for both the 'positive' and 'negative' classes, so in this respect the classification is not unbalanced.

What is a bit worrying is that only 63 out of 148 (a 42.60%) rules are working well when classifying the test dataset. The other 85 are zero precise, but still have an insignificant Recall, and although they don't seem to bother, they are still there, and they're irrelevant.

Nevertheless, we cannot draw great conclusions or meaning from this data set, since its attributes and the rules obtained are not easily understandable from a semantically logical point of view.

3.3 Car Evaluation

With the Car Evaluation dataset, the best rules are clear, strong, and could have been intuitively guessed. When it comes to the evaluation of a car (whether it is acceptable, unacceptable, good or very good) we all would agree that a low safety or a small capacity are unacceptable characteristics. The best 3 rules refer to this (see Table 2).

Rule (1) is a 100% precise, and has a 48.8% of Recall and a 33.94% of Coverage. So, if the safety of the car is low, the car will be automatically be classified as 'unnacc'. Rules (2) and (3), with a Recall of the 15.71% and a 14.98% respectively, show that a capacity of 2 people is also unsuitable for a car.

So, in this case, we're classifying the 'unacc' cars very well, but what about the rest of the classes? There are several rules that cover them with a 100% of Precision, but with a very small Recall (the maximum is of a 6%, for 'good') (A.3). Not to mention that 161 rules (a 69.70%) of the total (231) are not precise at all in the test dataset. Thus, the overall model is far away from perfect but still has an acceptable accuracy (84.39%, as seen in Table 1).

3.4 Mushroom

As seen in Table 1, the accuracy achieved with the mushroom dataset is the highest, 99, 29%, being almost perfect. This is no surprise due to the fact that 39 of the 45 rules (a 86.70%) of the model (subsection A.4) have a Precision of the 100%, and the other 6 are not precise, but they're still very few.

Rule (1) is the one with the highest Recall (73.65%) and Coverage (28.48%), and the premise is very simple: if the odor is fishy, then the mushroom is poisonous. Rules (2) and (3) are simple as well, involving only one and two conditions (the color of the mushroom stalk above the ring, and whether the mushroom has bruises), but still give quite good results. If we look at the full results (subsection A.4), we can see that at least the 10 most relevant rules are simple as well, and that both the 'poisonous' and 'edible' mushrooms are being correctly classified.

4 Conclusions

After implementing the PRISM algorithm and putting it to test with the four selected datasets, we can conclude that:

- The size of the dataset does not seem to affect the accuracy of the model. Besides, a large dataset doesn't necessarily imply that the number of rules obtained must be large as well, or vice versa.
- The higher the number of attributes, the easiest it seems to get a smaller rule set, but the number of classes don't seem to have any correlation neither with the size of the rule set or the accuracy of the model.
- Having a few simple, precise and strong rules (with high Precision, Recall and Coverage results) is translated into a better overall accuracy.
- The rules of the models obtained from the PRISM algorithm implementation are always a 100% precise in the train set, which means that the model can perfectly classify its instances.
- Some models can have a great accuracy but still contain many irrelevant rules in the test set, but that mainly depends on the train/test split.
- The best results are achieved with the *Mushroom* dataset; the worst, with *Tic-Tac-Toe Endgame.*
- The larger the data set, the bigger proportion of rules are still a 100% precise (from a 28.60% with the *Congressional Voting Records* dataset to an 86.70% with *Mushroom*).

Appendices

A Full Results

A.1 Congressional Voting Records Data Set

Rule	Class	Precision	Recall	Coverage
(1) if physician-fee-freeze == y and synfuels-corporation-cutback == n then	republican	100.0	84.09	40.0
(2) if synfuels-corporation-cutback == y and physician-fee-freeze == y and adoption-of-the-budget-resolution == y and anti-satellite-test-ban == n then	democrat	100.0	1.03	0.54
(3) if physician-fee-freeze == n then	democrat	96.15	95.88	50.27
(4) if synfuels-corporation-cutback == y and physician-fee-freeze == y and immigration == y then	republican	60.0	7.95	3.78
(5) if synfuels-corporation-cutback == y and physician-fee-freeze == y and immigration == n and duty-free-exports == y then	republican	0.0	1.14	0.54
(6) if synfuels-corporation-cutback == y and physician-fee-freeze == y and immigration == n and aid-to-nicaraguan-contras == n and anti-satellite-test-ban == y then	republican	0.0	1.14	0.54
(7) if synfuels-corporation-cutback == y and physician-fee-freeze == y and immigration == n and water-project-cost-sharing == n and handicapped-infants == y then	republican	0.0	1.14	0.54
(8) if synfuels-corporation-cutback == y and physician-fee-freeze == y and immigration == n and anti-satellite-test-ban == n and education-spending == n then	republican	0.0	1.14	0.54

(9) if synfuels-corporation-cutback == y and physician-fee-freeze == y and immigration == n and anti-satellite-test-ban == n and handicapped-infants == n and export-administration-act-south-africa == y then	republican	0.0	1.14	0.54
(10) if synfuels-corporation-cutback == y and physician-fee-freeze == y and immigration == n and water-project-cost-sharing == y and export-administration-act-south-africa == n and superfund-right-to-sue == y then	republican	0.0	1.14	0.54
(11) if synfuels-corporation-cutback == y and physician-fee-freeze == y and handicapped-infants == y and adoption-of-the-budget-resolution == n and education-spending == y and water-project-cost-sharing == y then	republican	0.0	1.14	0.54
(12) if synfuels-corporation-cutback == y and physician-fee-freeze == y and el-salvador-aid == n and adoption-of-the-budget-resolution == n then	democrat	0.0	1.03	0.54
(13) if export-administration-act-south-africa == n and superfund-right-to-sue == n and religious-groups-in-schools == y then	democrat	0.0	1.03	0.54
(14) if export-administration-act-south-africa == n and synfuels-corporation-cutback == y and water-project-cost-sharing == n and handicapped-infants == n and physician-fee-freeze == y then	democrat	0.0	1.03	0.54

A.2 Tic-Tac-Toe Endgame Data Set

Rule	Class	Precision	Recall	Coverage
(1) if V5 == o and V1 == o and V9 == o then	negative	100.0	15.33	5.22
(2) if V5 == o and V3 == o and V7 == o then	negative	100.0	14.56	4.96
(3) if V5 == x and V9 == x and V1 == x then	positive	100.0	13.66	9.01
(4) if V9 == o and V7 == o and V8 == o then	negative	100.0	11.49	3.92
(5) if V6 == o and V5 == o and V4 == o then	negative	100.0	11.11	3.79
(6) if V3 == o and V9 == o and V6 == o then	negative	100.0	10.73	3.66
(7) if V2 == o and V5 == o and V8 == o then	negative	100.0	10.34	3.52
(8) if V1 == o and V3 == o and V2 == o and V5 == x then	negative	100.0	7.66	2.61
(9) if V7 == x and V5 == x and V3 == x and V1 == o then	positive	100.0	7.33	4.83
(10) if V1 == o and V7 == o and V4 == o and V5 == x then	negative	100.0	5.36	1.83
(11) if V8 == x and V9 == x and V7 == x and V1 == o then	positive	100.0	5.35	3.52
(12) if V5 == x and V1 == b and V9 == b then	positive	100.0	4.95	3.26
(13) if V3 == x and V8 == b and V1 == x and V2 == x then	positive	100.0	4.16	2.74
(14) if V5 == b and V9 == x and V1 == b then	positive	100.0	3.76	2.48
(15) if V1 == o and V5 == x and V9 == o and V3 == b then	positive	100.0	2.57	1.7
(16) if V3 == x and V5 == x and V7 == b then	positive	100.0	2.57	1.7
(17) if V6 == x and V3 == x and V9 == x and V5 == o and V7 == b then	positive	100.0	2.38	1.57
(18) if V3 == x and V1 == x and V8 == x and V2 == x then	positive	100.0	2.18	1.44

(19) if V7 == x and V1 == x and V4 == x and V2 == x then	positive	100.0	1.98	1.31
(20) if V9 == b and V1 == x and V5 == x and V8 == o then	positive	100.0	1.98	1.31
(21) if V5 == b and V1 == x and V2 == o and V6 == o then	positive	100.0	1.58	1.04
(22) if V9 == b and V4 == x and V1 == o and V7 == o and V5 == x then	positive	100.0	1.58	1.04
(23) if V7 == x and V6 == b and V1 == x and V8 == x then	positive	100.0	1.39	0.91
(24) if V2 == x and V7 == b and V3 == o and V1 == o and V5 == x then	positive	100.0	1.39	0.91
(25) if V7 == x and V2 == b and V8 == o and V5 == b then	positive	100.0	1.19	0.78
(26) if V5 == b and V8 == o and V7 == b and V1 == x then	positive	100.0	1.19	0.78
(27) if V3 == x and V9 == x and V6 == x and V1 == x and V7 == o then	positive	100.0	1.19	0.78
(28) if V1 == b and V9 == o and V2 == o and V8 == b then	positive	100.0	0.99	0.65
(29) if V1 == b and V9 == o and V6 == b and V4 == o then	positive	100.0	0.99	0.65
(30) if V7 == x and V1 == x and V3 == x and V5 == x and V9 == o then	positive	100.0	0.99	0.65
(31) if V4 == x and V7 == x and V1 == x and V5 == x and V9 == o then	positive	100.0	0.99	0.65
(32) if V6 == x and V7 == b and V2 == b and V3 == o then	positive	100.0	0.79	0.52
(33) if V4 == b and V3 == x and V5 == b and V1 == o and V6 == x then	positive	100.0	0.79	0.52
(34) if V1 == o and V5 == b and V8 == b and V3 == o and V9 == x then	negative	100.0	0.77	0.26

(35) if V1 == b and V9 == x and V5 == x and V7 == x and V3 == x then	positive	100.0	0.59	0.39
(36) if V6 == x and V8 == b and V7 == x and V9 == x and V3 == x then	positive	100.0	0.59	0.39
(37) if V4 == x and V2 == o and V1 == x and V8 == b and V9 == b and V3 == o then	positive	100.0	0.59	0.39
(38) if V1 == b and V9 == o and V3 == o and V7 == o and V5 == x then	positive	100.0	0.59	0.39
(39) if V5 == o and V2 == o and V6 == o and V3 == b then	positive	100.0	0.59	0.39
(40) if V2 == x and V6 == b and V9 == b and V3 == b then	positive	100.0	0.4	0.26
(41) if V1 == x and V7 == x and V6 == x and V5 == b then	positive	100.0	0.4	0.26
(42) if V1 == b and V8 == x and V6 == x and V3 == b then	positive	100.0	0.4	0.26
(43) if V2 == x and V4 == b and V9 == b and V1 == o and V3 == o then	positive	100.0	0.4	0.26
(44) if V7 == x and V1 == x and V4 == x and V8 == x and V2 == b then	positive	100.0	0.4	0.26
(45) if V7 == x and V1 == x and V3 == x and V4 == x and V6 == b then	positive	100.0	0.4	0.26
(46) if V4 == x and V3 == b and V9 == b and V6 == b then	positive	100.0	0.4	0.26
(47) if V7 == x and V1 == x and V9 == x and V4 == x and V6 == b then	positive	100.0	0.4	0.26
(48) if V1 == o and V5 == b and V8 == b and V6 == b then	negative	100.0	0.38	0.13
(49) if V1 == o and V5 == o and V9 == b and V4 == o then	negative	100.0	0.38	0.13
(50) if V2 == b and V4 == x and V9 == b and V7 == b then	positive	100.0	0.2	0.13

(51) if V1 == b and V4 == b and V9 == o and V2 == o then	positive	100.0	0.2	0.13
(52) if V1 == b and V4 == b and V5 == o and V6 == o then	positive	100.0	0.2	0.13
(53) if V6 == x and V7 == x and V2 == b and V9 == b then	positive	100.0	0.2	0.13
(54) if V2 == x and V6 == b and V4 == b and V3 == o and V5 == x and V1 == o then	positive	100.0	0.2	0.13
(55) if V3 == x and V7 == o and V4 == b and V1 == o and V2 == o then	positive	100.0	0.2	0.13
(56) if V1 == b and V6 == b and V2 == b and V4 == b then	positive	100.0	0.2	0.13
(57) if V9 == b and V8 == b and V7 == b and V1 == o then	positive	100.0	0.2	0.13
(58) if V5 == o and V2 == o and V6 == o and V8 == b and V4 == x then	positive	100.0	0.2	0.13
(59) if V3 == x and V7 == o and V5 == o and V2 == b and V9 == x then	positive	83.33	0.4	0.26
(60) if V7 == o and V8 == x and V5 == b and V9 == x and V1 == o then	negative	80.0	0.77	0.26
(61) if V1 == b and V4 == b and V9 == o and V8 == b then	positive	50.0	0.2	0.13
(62) if V1 == b and V9 == o and V3 == o and V8 == o then	positive	50.0	0.2	0.13
(63) if V1 == b and V7 == x and V3 == o and V5 == o and V2 == o then	positive	50.0	0.2	0.13
(64) if V5 == b and V1 == o and V9 == b then	negative	0.0	3.07	1.04
(65) if V2 == b and V4 == x and V7 == x and V8 == b and V1 == x then	positive	0.0	1.78	1.17
(66) if V1 == b and V9 == x and V5 == x and V7 == o then	positive	0.0	1.78	1.17

(67) if V8 == o and V5 == o and V2 == x and V7 == b and V1 == x then	positive	0.0	1.39	0.91
(68) if V8 == x and V6 == x and V1 == x and V7 == x and V4 == o then	negative	0.0	1.15	0.39
(69) if V6 == x and V4 == x and V5 == x and V1 == o and V9 == x and V7 == o then	positive	0.0	0.99	0.65
(70) if V1 == x and V9 == b and V5 == x and V8 == x then	positive	0.0	0.79	0.52
(71) if V5 == o and V1 == o and V7 == o and V4 == o and V9 == x then	negative	0.0	0.77	0.26
(72) if V1 == b and V7 == x and V3 == b and V5 == x then	positive	0.0	0.59	0.39
(73) if V3 == x and V1 == x and V7 == x and V2 == x and V9 == b then	positive	0.0	0.59	0.39
(74) if V1 == b and V5 == o and V6 == b and V4 == o then	positive	0.0	0.59	0.39
(75) if V3 == x and V9 == x and V4 == x and V5 == b and V1 == o then	positive	0.0	0.59	0.39
(76) if V2 == x and V5 == x and V8 == x and V1 == x and V3 == b and V9 == o then	positive	0.0	0.59	0.39
(77) if V1 == b and V9 == o and V4 == b and V7 == b then	positive	0.0	0.4	0.26
(78) if V1 == b and V9 == o and V3 == b and V2 == b then	positive	0.0	0.4	0.26
(79) if V1 == b and V8 == x and V2 == b and V5 == x and V4 == o then	positive	0.0	0.4	0.26
(80) if V1 == b and V4 == b and V6 == b and V3 == b then	positive	0.0	0.4	0.26
(81) if V1 == b and V4 == b and V3 == x and V2 == x and V6 == o then	positive	0.0	0.4	0.26
(82) if V7 == o and V4 == b and V1 == o and V6 == o then	positive	0.0	0.4	0.26

(83) if V3 == x and V7 == o and V5 == o and V1 == b and V2 == o then	positive	0.0	0.4	0.26
(84) if V9 == o and V2 == o and V7 == b and V1 == x and V6 == x then	positive	0.0	0.4	0.26
(85) if V8 == x and V3 == o and V4 == o and V7 == b and V9 == o then	positive	0.0	0.4	0.26
(86) if V2 == o and V4 == x and V8 == x and V6 == x and V1 == x then	positive	0.0	0.4	0.26
(87) if V9 == x and V8 == x and V7 == x and V1 == x and V2 == b and V6 == o then	positive	0.0	0.4	0.26
(88) if V3 == b and V7 == b and V1 == o and V9 == x then	positive	0.0	0.4	0.26
(89) if V5 == b and V1 == o and V9 == o and V2 == o then	negative	0.0	0.38	0.13
(90) if V5 == b and V1 == o and V7 == b and V6 == b then	negative	0.0	0.38	0.13
(91) if V1 == o and V5 == b and V3 == o and V8 == o then	negative	0.0	0.38	0.13
(92) if V1 == o and V5 == b and V7 == b and V3 == o and V4 == b then	negative	0.0	0.38	0.13
(93) if V1 == o and V5 == b and V9 == o and V4 == o then	negative	0.0	0.38	0.13
(94) if V1 == o and V7 == o and V6 == o and V2 == o then	negative	0.0	0.38	0.13
(95) if V1 == o and V5 == o and V9 == b and V2 == o and V3 == o then	negative	0.0	0.38	0.13
(96) if V9 == x and V2 == x and V4 == x and V3 == x and V5 == x then	negative	0.0	0.38	0.13
(97) if V9 == x and V2 == x and V4 == x and V7 == x and V3 == x then	negative	0.0	0.38	0.13
(98) if V5 == o and V1 == o and V7 == o and V6 == o then	negative	0.0	0.38	0.13

(99) if V5 == o and V3 == o and V7 == b and V1 == o and V2 == o then	negative	0.0	0.38	0.13
(100) if V1 == x and V8 == x and V3 == x and V4 == x and V5 == x then	negative	0.0	0.38	0.13
(101) if V5 == o and V8 == o and V3 == o and V1 == o then	negative	0.0	0.38	0.13
(102) if V1 == x and V6 == x and V8 == x and V3 == x and V5 == x then	negative	0.0	0.38	0.13
(103) if V5 == o and V9 == x and V1 == x and V6 == x and V7 == x and V2 == x then	negative	0.0	0.38	0.13
(104) if V6 == o and V2 == o and V7 == o and V5 == o then	negative	0.0	0.38	0.13
(105) if V6 == x and V4 == x and V5 == x and V2 == x and V7 == x then	positive	0.0	0.2	0.13
(106) if V6 == x and V4 == x and V5 == x and V2 == x and V8 == x then	positive	0.0	0.2	0.13
(107) if V2 == x and V5 == x and V8 == x and V1 == x and V6 == x then	positive	0.0	0.2	0.13
(108) if V2 == x and V8 == o and V5 == b and V9 == x and V6 == b then	positive	0.0	0.2	0.13
(109) if V2 == x and V4 == b and V6 == b and V9 == x and V3 == x then	positive	0.0	0.2	0.13
(110) if V2 == x and V8 == o and V5 == b and V4 == b and V7 == x then	positive	0.0	0.2	0.13
(111) if V2 == x and V9 == x and V3 == x and V6 == x and V4 == x then	positive	0.0	0.2	0.13
(112) if V2 == x and V9 == x and V3 == x and V6 == x and V7 == x then	positive	0.0	0.2	0.13
(113) if V7 == x and V1 == x and V3 == x and V4 == x and V8 == x then	positive	0.0	0.2	0.13
(114) if V2 == x and V3 == x and V8 == o and V5 == b and V6 == b and V7 == x then	positive	0.0	0.2	0.13

(115) if V2 == x and V5 == x and V8 == x and V7 == x and V6 == x then	positive	0.0	0.2	0.13
(116) if V2 == x and V3 == x and V9 == x and V4 == b and V1 == b then	positive	0.0	0.2	0.13
(117) if V7 == x and V1 == x and V6 == x and V2 == b and V3 == b then	positive	0.0	0.2	0.13
(118) if V2 == x and V5 == x and V8 == x and V4 == x and V3 == x then	positive	0.0	0.2	0.13
(119) if V2 == x and V5 == x and V8 == x and V9 == x and V4 == x then	positive	0.0	0.2	0.13
(120) if V3 == x and V8 == o and V5 == o and V2 == b and V6 == o then	positive	0.0	0.2	0.13
(121) if V2 == x and V6 == b and V4 == b and V5 == b and V9 == b then	positive	0.0	0.2	0.13
(122) if V2 == x and V1 == b and V8 == b and V3 == x and V5 == x then	positive	0.0	0.2	0.13
(123) if V8 == x and V5 == x and V2 == x and V7 == x and V9 == b and V3 == b then	positive	0.0	0.2	0.13
(124) if V6 == x and V7 == x and V2 == b and V5 == x and V3 == o and V8 == b then	positive	0.0	0.2	0.13
(125) if V3 == x and V8 == o and V5 == o and V7 == o and V6 == o then	positive	0.0	0.2	0.13
(126) if V3 == x and V4 == b and V2 == b and V8 == o and V5 == o and V1 == o then	positive	0.0	0.2	0.13
(127) if V3 == x and V7 == o and V4 == b and V1 == o and V6 == b then	positive	0.0	0.2	0.13
(128) if V3 == x and V6 == x and V2 == x and V1 == x and V7 == x then	positive	0.0	0.2	0.13
(129) if V9 == x and V7 == b and V2 == x and V5 == b and V1 == o then	positive	0.0	0.2	0.13
(130) if V9 == x and V8 == x and V7 == b and V2 == x and V5 == x and V1 == b then	positive	0.0	0.2	0.13

(131) if V2 == o and V5 == o and V6 == o and V3 == o then	positive	0.0	0.2	0.13
(132) if V3 == x and V4 == x and V9 == b and V2 == x and V6 == b then	positive	0.0	0.2	0.13
(133) if V4 == b and V1 == x and V2 == x and V6 == b and V7 == x then	positive	0.0	0.2	0.13
(134) if V2 == o and V9 == b and V6 == o and V8 == b and V4 == o then	positive	0.0	0.2	0.13
(135) if V7 == o and V4 == b and V8 == o and V1 == o and V2 == b then	positive	0.0	0.2	0.13
(136) if V7 == o and V4 == b and V5 == b and V6 == o and V8 == o then	positive	0.0	0.2	0.13
(137) if V1 == b and V4 == x and V6 == o and V3 == b and V2 == x then	positive	0.0	0.2	0.13
(138) if V9 == b and V7 == x and V3 == b and V1 == o and V2 == o then	positive	0.0	0.2	0.13
(139) if V3 == x and V4 == x and V8 == x and V6 == x and V5 == x then	positive	0.0	0.2	0.13
(140) if V8 == b and V5 == b and V3 == b and V6 == b then	positive	0.0	0.2	0.13
(141) if V1 == b and V6 == b and V2 == o and V3 == o and V4 == o then	positive	0.0	0.2	0.13
(142) if V3 == x and V4 == x and V1 == b and V2 == b and V6 == o then	positive	0.0	0.2	0.13
(143) if V9 == b and V5 == o and V4 == b and V7 == o and V8 == o then	positive	0.0	0.2	0.13
(144) if V8 == b and V5 == b and V7 == b and V2 == b then	positive	0.0	0.2	0.13
(145) if V9 == b and V7 == x and V2 == x and V6 == b and V8 == x then	positive	0.0	0.2	0.13
(146) if V5 == o and V8 == b and V1 == b and V7 == o and V4 == o then	positive	0.0	0.2	0.13

(147) if V7 == o and V8 == x and V2 == o and V5 == o and V1 == o then	positive	0.0	0.2	0.13
(148) if V4 == b and V6 == b and V3 == o and V9 == o and V1 == x then	positive	0.0	0.2	0.13

A.3 Car Evaluation Data Set

Rule	Class	Precision	Recall	Coverage
(1) if safety == low then	unacc	100.0	48.8	33.94
(2) if persons == 2 and safety == high then	unacc	100.0	15.71	10.93
(3) if safety == med and persons == 2 then	unacc	100.0	14.98	10.42
(4) if maint == low and buying == med and safety == med and lug_boot == big and persons == 4 then	good	100.0	6.0	0.22
(5) if buying == low and maint == med and persons == 4 and safety == high and lug_boot == small then	good	100.0	6.0	0.22
(6) if maint == low and buying == med and safety == high and persons == 4 and lug_boot == small then	good	100.0	6.0	0.22
(7) if buying == low and maint == low and safety == med and lug_boot == big and persons == 4 then	good	100.0	6.0	0.22
(8) if buying == low and maint == med and safety == med and persons == 4 and lug_boot == big then	good	100.0	6.0	0.22
(9) if safety == high and buying == low and lug_boot == big and persons == 4 and maint == low then	vgood	100.0	5.88	0.22
(10) if safety == high and buying == low and maint == med and lug_boot == big and persons == more then	vgood	100.0	5.88	0.22
(11) if safety == high and buying == med and maint == med and lug_boot == big and persons == 4 then	vgood	100.0	5.88	0.22
(12) if safety == high and buying == low and persons == 4 and maint == high and lug_boot == big then	vgood	100.0	5.88	0.22

(13) if safety == high and buying == med and maint == low and persons == 4 and lug_boot == big then	vgood	100.0	5.88	0.22
(14) if safety == high and buying == med and maint == med and persons == more and lug_boot == big then	vgood	100.0	5.88	0.22
(15) if safety == high and buying == low and maint == low and persons == more and lug_boot == big then	vgood	100.0	5.88	0.22
(16) if maint == low and buying == low and persons == more and safety == med and lug_boot == big then	good	100.0	4.0	0.14
(17) if safety == high and persons == 4 and maint == high and buying == med then	acc	100.0	3.44	0.8
(18) if safety == med and persons == 4 and buying == med and maint == med then	acc	100.0	3.12	0.72
(19) if safety == high and persons == 4 and maint == vhigh and buying == low then	acc	100.0	3.12	0.72
(20) if safety == high and persons == 4 and buying == med and maint == vhigh then	acc	100.0	3.12	0.72
(21) if safety == high and persons == 4 and buying == high and maint == low then	acc	100.0	2.81	0.65
(22) if persons == 4 and safety == high and buying == high and maint == high then	acc	100.0	2.81	0.65
(23) if safety == high and persons == 4 and maint == med and buying == high then	acc	100.0	2.81	0.65
(24) if persons == 4 and maint == low and buying == vhigh and safety == high then	acc	100.0	2.81	0.65
(25) if safety == med and persons == 4 and buying == low and maint == high then	acc	100.0	2.5	0.58
(26) if safety == med and buying == vhigh and persons == 4 and lug_boot == small then	unacc	100.0	1.35	0.94
(27) if safety == med and persons == 4 and buying == high and lug_boot == small then	unacc	100.0	1.35	0.94

(28) if buying == vhigh and maint == high and persons == more and safety == high then	unacc	100.0	1.14	0.8
(29) if maint == vhigh and safety == high and buying == high and persons == 4 then	unacc	100.0	1.14	0.8
(30) if maint == vhigh and buying == high and persons == more and safety == med then	unacc	100.0	0.94	0.65
(31) if safety == med and buying == vhigh and maint == high and persons == more then	unacc	100.0	0.94	0.65
(32) if maint == vhigh and safety == high and buying == high and persons == more then	unacc	100.0	0.94	0.65
(33) if buying == vhigh and persons == 4 and safety == high and maint == vhigh then	unacc	100.0	0.94	0.65
(34) if safety == med and persons == more and maint == low and buying == vhigh and lug_boot == big then	acc	100.0	0.94	0.22
(35) if safety == med and persons == more and buying == med and maint == high and lug_boot == big then	acc	100.0	0.94	0.22
(36) if safety == med and persons == 4 and buying == med and maint == vhigh and lug_boot == big then	acc	100.0	0.94	0.22
(37) if safety == med and persons == 4 and lug_boot == big and maint == low and buying == vhigh then	acc	100.0	0.94	0.22
(38) if safety == med and persons == more and buying == high and maint == high and lug_boot == big then	acc	100.0	0.94	0.22
(39) if buying == low and lug_boot == small and safety == high and maint == high and persons == 4 then	acc	100.0	0.94	0.22
(40) if maint == high and buying == vhigh and persons == 4 and safety == high then	unacc	100.0	0.83	0.58

(41) if safety == med and maint == med and buying == high and lug_boot == big and persons == 4 then	acc	100.0	0.62	0.14
(42) if persons == more and maint == med and buying == vhigh and safety == high and doors == 4 then	acc	100.0	0.62	0.14
(43) if safety == med and persons == more and maint == med and buying == high and lug_boot == big then	acc	100.0	0.62	0.14
(44) if maint == med and safety == high and buying == med and persons == 4 and lug_boot == small then	acc	100.0	0.62	0.14
(45) if safety == med and persons == 4 and maint == low and buying == high and lug_boot == big then	acc	100.0	0.62	0.14
(46) if safety == med and persons == 4 and maint == vhigh and doors == 3 and lug_boot == med then	unacc	100.0	0.31	0.22
(47) if safety == med and persons == 4 and maint == vhigh and doors == 2 and lug_boot == med then	unacc	100.0	0.31	0.22
(48) if safety == med and persons == 4 and buying == vhigh and lug_boot == med and maint == high then	unacc	100.0	0.31	0.22
(49) if safety == med and persons == 4 and maint == vhigh and buying == vhigh and lug_boot == big then	unacc	100.0	0.31	0.22
(50) if safety == med and persons == 4 and maint == vhigh and lug_boot == small and buying == low then	unacc	100.0	0.31	0.22
(51) if safety == med and lug_boot == small and persons == more and buying == high and maint == high then	unacc	100.0	0.31	0.22

(52) if safety == med and persons == 4 and maint == high and buying == med and lug_boot == small then	unacc	100.0	0.31	0.22
(53) if safety == med and persons == more and lug_boot == small and maint == vhigh and buying == low then	unacc	100.0	0.31	0.22
(54) if safety == med and persons == more and lug_boot == small and maint == low and buying == high then	unacc	100.0	0.31	0.22
(55) if safety == med and persons == 4 and buying == high and maint == vhigh and lug_boot == big then	unacc	100.0	0.31	0.22
(56) if safety == med and buying == vhigh and persons == 4 and maint == high and lug_boot == big then	unacc	100.0	0.31	0.22
(57) if safety == med and persons == 4 and buying == vhigh and lug_boot == med and maint == vhigh then	unacc	100.0	0.21	0.14
(58) if safety == med and buying == med and maint == vhigh and lug_boot == small and doors == 3 then	unacc	100.0	0.21	0.14
(59) if safety == med and buying == vhigh and maint == med and persons == more and lug_boot == small then	unacc	100.0	0.21	0.14
(60) if doors == 2 and persons == more and lug_boot == small and safety == high and maint == low then	unacc	100.0	0.21	0.14
(61) if safety == med and doors == 2 and persons == more and buying == low and lug_boot == small then	unacc	100.0	0.1	0.07
(62) if safety == med and buying == med and lug_boot == small and maint == vhigh and doors == 4 then	unacc	100.0	0.1	0.07
(63) if persons == more and safety == high and maint == vhigh and buying == med then	acc	75.0	2.5	0.58

(64) if safety == high and persons == more and buying == med and maint == high then	acc	66.67	2.81	0.65
(65) if safety == high and buying == med and lug_boot == med and maint == med and persons == more then	vgood	50.0	3.92	0.14
(66) if persons == more and safety == high and buying == vhigh and maint == low then	acc	50.0	3.12	0.72
(67) if buying == low and doors == 2 and safety == high and lug_boot == med and maint == low then	good	50.0	2.0	0.07
(68) if safety == high and buying == low and maint == med and doors == 4 and lug_boot == big then	vgood	50.0	1.96	0.07
(69) if safety == med and persons == 4 and lug_boot == med and doors == 4 and buying == high then	acc	50.0	0.62	0.14
(70) if safety == med and persons == more and maint == med and doors == 4 and buying == vhigh then	acc	50.0	0.31	0.07
(71) if buying == low and maint == med and persons == more and safety == med and lug_boot == big then	good	0.0	8.0	0.29
(72) if maint == low and buying == low and persons == 4 and safety == high and lug_boot == small then	good	0.0	8.0	0.29
(73) if safety == high and buying == low and persons == more and lug_boot == big and maint == high then	vgood	0.0	7.84	0.29
(74) if buying == low and safety == high and maint == low and persons == more and lug_boot == small then	good	0.0	6.0	0.22
(75) if safety == high and buying == low and lug_boot == med and persons == more and maint == low then	vgood	0.0	5.88	0.22

(76) if maint == low and buying == med and persons == more and safety == med and doors == 3 then	good	0.0	4.0	0.14
(77) if safety == high and buying == med and maint == low and persons == more and doors == 5more then	vgood	0.0	3.92	0.14
(78) if safety == high and maint == low and buying == med and doors == 4 and persons == more then	vgood	0.0	3.92	0.14
(79) if safety == high and lug_boot == med and persons == 4 and doors == 5more and buying == med then	vgood	0.0	3.92	0.14
(80) if safety == high and persons == 4 and maint == med and buying == vhigh then	acc	0.0	3.75	0.87
(81) if safety == med and persons == more and buying == low and maint == high then	acc	0.0	3.44	0.8
(82) if buying == low and lug_boot == med and maint == med and persons == more and safety == med and doors == 3 then	good	0.0	2.0	0.07
(83) if buying == low and maint == low and lug_boot == med and safety == med and doors == 4 and persons == 4 then	good	0.0	2.0	0.07
(84) if buying == low and maint == med and lug_boot == med and persons == 4 and safety == med and doors == 4 then	good	0.0	2.0	0.07
(85) if maint == low and buying == med and lug_boot == med and persons == 4 and safety == high and doors == 3 then	good	0.0	2.0	0.07
(86) if buying == low and maint == med and persons == more and safety == high and lug_boot == small and doors == 4 then	good	0.0	2.0	0.07
(87) if maint == low and buying == med and persons == more and safety == med and doors == 4 and lug_boot == med then	good	0.0	2.0	0.07

(88) if buying == low and lug_boot == med and maint == low and persons == 4 and safety == high and doors == 3 then	good	0.0	2.0	0.07
(89) if buying == low and lug_boot == med and maint == med and doors == 5more and safety == med and persons == 4 then	good	0.0	2.0	0.07
(90) if maint == low and buying == med and persons == more and safety == high and doors == 3 and lug_boot == small then	good	0.0	2.0	0.07
(91) if buying == low and lug_boot == med and maint == low and safety == med and doors == 5more and persons == 4 then	good	0.0	2.0	0.07
(92) if buying == low and maint == med and persons == more and doors == 5more and lug_boot == med and safety == med then	good	0.0	2.0	0.07
(93) if maint == low and buying == med and doors == 2 and safety == high and lug_boot == med and persons == 4 then	good	0.0	2.0	0.07
(94) if buying == low and maint == med and safety == high and doors == 2 and lug_boot == med and persons == 4 then	good	0.0	2.0	0.07
(95) if maint == low and buying == med and safety == med and doors == 5more and persons == more and lug_boot == big then	good	0.0	2.0	0.07
(96) if buying == low and persons == more and maint == med and safety == high and doors == 5more and lug_boot == small then	good	0.0	2.0	0.07
(97) if maint == low and lug_boot == med and doors == 4 and safety == med and persons == 4 and buying == med then	good	0.0	2.0	0.07
(98) if persons == more and lug_boot == med and buying == low and maint == low and safety == med and doors == 4 then	good	0.0	2.0	0.07

(99) if doors == 2 and persons == more and safety == high and buying == med and maint == low and lug_boot == med then	good	0.0	2.0	0.07
(100) if doors == 2 and buying == low and maint == med and safety == high and persons == more and lug_boot == med then	good	0.0	2.0	0.07
(101) if safety == high and buying == low and lug_boot == med and doors == 4 and persons == 4 and maint == low then	vgood	0.0	1.96	0.07
(102) if safety == high and buying == low and lug_boot == med and maint == high and doors == 4 and persons == more then	vgood	0.0	1.96	0.07
(103) if safety == high and buying == low and lug_boot == med and maint == med and persons == more and doors == 4 then	vgood	0.0	1.96	0.07
(104) if safety == high and buying == low and lug_boot == med and doors == 5more and maint == high and persons == more then	vgood	0.0	1.96	0.07
(105) if safety == high and buying == low and lug_boot == med and persons == 4 and maint == high and doors == 4 then	vgood	0.0	1.96	0.07
(106) if safety == high and buying == low and maint == med and lug_boot == med and persons == more and doors == 5more then	vgood	0.0	1.96	0.07
(107) if safety == high and buying == low and maint == med and doors == 3 and persons == 4 and lug_boot == big then	vgood	0.0	1.96	0.07
(108) if safety == high and lug_boot == med and buying == low and persons == 4 and doors == 5more and maint == low then	vgood	0.0	1.96	0.07
(109) if safety == high and lug_boot == med and buying == low and maint == med and doors == 3 and persons == more then	vgood	0.0	1.96	0.07

(110) if safety == high and lug_boot == med and buying == med and doors == 4 and persons == 4 and maint == low then	vgood	0.0	1.96	0.07
(111) if safety == high and lug_boot == med and persons == 4 and buying == low and maint == med and doors == 4 then	vgood	0.0	1.96	0.07
(112) if safety == high and lug_boot == med and buying == med and maint == med and doors == 4 and persons == 4 then	vgood	0.0	1.96	0.07
(113) if safety == high and lug_boot == med and buying == low and maint == high and doors == 5more and persons == 4 then	vgood	0.0	1.96	0.07
(114) if buying == med and maint == low and doors == 3 and persons == more and safety == high and lug_boot == med then	vgood	0.0	1.96	0.07
(115) if maint == vhigh and safety == med and persons == more and buying == vhigh then	unacc	0.0	1.25	0.87
(116) if buying == vhigh and maint == vhigh and safety == high and persons == more then	unacc	0.0	1.25	0.87
(117) if safety == high and persons == more and buying == high and maint == low and lug_boot == med then	acc	0.0	1.25	0.29
(118) if persons == more and safety == high and buying == high and maint == high and lug_boot == med then	acc	0.0	1.25	0.29
(119) if persons == more and safety == med and buying == med and maint == med and lug_boot == med then	acc	0.0	1.25	0.29
(120) if persons == more and safety == high and buying == high and maint == med and lug_boot == med then	acc	0.0	1.25	0.29
(121) if persons == more and safety == med and maint == low and buying == high and lug_boot == big then	acc	0.0	1.25	0.29

(122) if safety == med and persons == more and buying == med and lug_boot == big and maint == vhigh then	acc	0.0	1.25	0.29
(123) if safety == med and persons == 4 and maint == low and buying == med and lug_boot == small then	acc	0.0	1.25	0.29
(124) if safety == med and persons == more and maint == med and lug_boot == big and buying == med then	acc	0.0	1.25	0.29
(125) if safety == med and persons == 4 and lug_boot == big and buying == high and maint == high then	acc	0.0	1.25	0.29
(126) if persons == more and safety == high and buying == high and lug_boot == big and maint == low then	acc	0.0	1.25	0.29
(127) if safety == med and persons == 4 and lug_boot == big and maint == vhigh and buying == low then	acc	0.0	1.25	0.29
(128) if safety == med and persons == more and buying == low and maint == vhigh and lug_boot == big then	acc	0.0	1.25	0.29
(129) if safety == med and persons == 4 and maint == low and buying == low and lug_boot == small then	acc	0.0	1.25	0.29
(130) if persons == more and safety == high and buying == high and maint == high and lug_boot == big then	acc	0.0	1.25	0.29
(131) if persons == more and buying == low and maint == vhigh and safety == high and lug_boot == big then	acc	0.0	1.25	0.29
(132) if persons == more and maint == med and buying == high and lug_boot == big and safety == high then	acc	0.0	1.25	0.29

(133) if persons == more and buying == low and safety == high and maint == vhigh and lug_boot == med then	acc	0.0	1.25	0.29
(134) if safety == med and persons == 4 and buying == med and maint == high and lug_boot == big then	acc	0.0	1.25	0.29
(135) if persons == more and safety == high and lug_boot == small and doors == 3 and maint == med then	acc	0.0	0.94	0.22
(136) if safety == med and persons == more and maint == low and lug_boot == med and buying == high then	acc	0.0	0.94	0.22
(137) if persons == more and maint == med and safety == high and buying == vhigh and doors == 5more then	acc	0.0	0.94	0.22
(138) if safety == med and persons == 4 and lug_boot == med and doors == 5more and buying == high then	acc	0.0	0.94	0.22
(139) if persons == more and lug_boot == small and buying == low and safety == med and maint == low then	acc	0.0	0.94	0.22
(140) if safety == med and persons == 4 and buying == low and doors == 2 and maint == med then	acc	0.0	0.62	0.14
(141) if safety == med and persons == more and doors == 5more and buying == med and lug_boot == small then	acc	0.0	0.62	0.14
(142) if safety == med and maint == med and buying == low and lug_boot == small and doors == 4 then	acc	0.0	0.62	0.14
(143) if safety == med and doors == 2 and persons == more and lug_boot == small and buying == med then	unacc	0.0	0.42	0.29

(144) if safety == med and lug_boot == small and persons == more and buying == high and maint == med then	unacc	0.0	0.42	0.29
(145) if doors == 2 and persons == more and lug_boot == small and safety == high and maint == med then	unacc	0.0	0.42	0.29
(146) if safety == med and persons == more and lug_boot == med and doors == 3 and buying == high and maint == med then	acc	0.0	0.31	0.07
(147) if safety == med and persons == more and lug_boot == med and doors == 3 and maint == vhigh and buying == med then	acc	0.0	0.31	0.07
(148) if safety == med and persons == more and lug_boot == med and buying == low and maint == vhigh and doors == 3 then	acc	0.0	0.31	0.07
(149) if persons == more and maint == med and doors == 4 and lug_boot == small and safety == high and buying == med then	acc	0.0	0.31	0.07
(150) if safety == med and persons == more and maint == med and lug_boot == small and buying == med and doors == 3 then	acc	0.0	0.31	0.07
(151) if safety == med and persons == more and lug_boot == med and maint == low and buying == vhigh and doors == 3 then	acc	0.0	0.31	0.07
(152) if safety == med and persons == more and maint == med and doors == 4 and lug_boot == small and buying == med then	acc	0.0	0.31	0.07
(153) if safety == med and persons == 4 and lug_boot == med and buying == med and maint == low and doors == 3 then	acc	0.0	0.31	0.07
(154) if safety == high and lug_boot == small and persons == more and buying == high and maint == low and doors == 4 then	acc	0.0	0.31	0.07

(155) if safety == med and lug_boot == med and persons == more and doors == 5more and maint == vhigh and buying == med then	acc	0.0	0.31	0.07
(156) if safety == med and lug_boot == med and persons == 4 and doors == 4 and buying == med and maint == high then	acc	0.0	0.31	0.07
(157) if safety == med and persons == more and lug_boot == med and buying == low and maint == vhigh and doors == 4 then	acc	0.0	0.31	0.07
(158) if safety == med and persons == 4 and maint == med and doors == 4 and buying == vhigh and lug_boot == big then	acc	0.0	0.31	0.07
(159) if safety == med and lug_boot == med and persons == 4 and buying == low and doors == 3 and maint == low then	acc	0.0	0.31	0.07
(160) if safety == med and lug_boot == med and persons == more and buying == high and maint == high and doors == 4 then	acc	0.0	0.31	0.07
(161) if safety == med and lug_boot == med and persons == 4 and doors == 5more and buying == med and maint == high then	acc	0.0	0.31	0.07
(162) if safety == med and lug_boot == med and persons == more and maint == low and doors == 2 and buying == med then	acc	0.0	0.31	0.07
(163) if safety == med and persons == 4 and lug_boot == med and maint == vhigh and doors == 4 and buying == med then	acc	0.0	0.31	0.07
(164) if safety == med and lug_boot == med and persons == more and doors == 5more and maint == low and buying == vhigh then	acc	0.0	0.31	0.07
(165) if safety == med and lug_boot == med and buying == low and maint == vhigh and doors == 5more and persons == 4 then	acc	0.0	0.31	0.07

(166) if safety == high and lug_boot == small and persons == more and buying == high and maint == high and doors == 4 then	acc	0.0	0.31	0.07
(167) if safety == med and lug_boot == med and persons == more and buying == low and doors == 2 and maint == low then	acc	0.0	0.31	0.07
(168) if safety == med and persons == 4 and lug_boot == med and buying == vhigh and maint == med and doors == 5more then	acc	0.0	0.31	0.07
(169) if persons == more and lug_boot == small and buying == low and safety == high and maint == vhigh and doors == 5more then	acc	0.0	0.31	0.07
(170) if persons == more and lug_boot == small and safety == high and buying == high and doors == 5more and maint == low then	acc	0.0	0.31	0.07
(171) if maint == med and persons == more and buying == vhigh and doors == 3 and lug_boot == big and safety == med then	acc	0.0	0.31	0.07
(172) if persons == more and maint == med and safety == high and buying == vhigh and doors == 3 and lug_boot == big then	acc	0.0	0.31	0.07
(173) if persons == more and lug_boot == small and safety == high and buying == high and maint == high and doors == 5more then	acc	0.0	0.31	0.07
(174) if safety == med and lug_boot == med and persons == more and buying == high and maint == high and doors == 5more then	acc	0.0	0.31	0.07
(175) if maint == med and safety == med and buying == low and lug_boot == small and doors == 5more and persons == 4 then	acc	0.0	0.31	0.07
(176) if maint == med and persons == more and lug_boot == med and safety == med and buying == vhigh and doors == 3 then	acc	0.0	0.31	0.07

(177) if persons == more and lug_boot == small and safety == high and buying == low and maint == high and doors == 5more then	acc	0.0	0.31	0.07
(178) if safety == med and lug_boot == med and persons == 4 and maint == low and doors == 2 and buying == low then	acc	0.0	0.31	0.07
(179) if persons == more and lug_boot == small and safety == high and buying == high and maint == med and doors == 5more then	acc	0.0	0.31	0.07
(180) if safety == med and lug_boot == med and persons == 4 and doors == 4 and buying == vhigh and maint == med then	acc	0.0	0.31	0.07
(181) if persons == more and buying == low and lug_boot == small and safety == high and maint == vhigh and doors == 3 then	acc	0.0	0.31	0.07
(182) if safety == med and lug_boot == med and persons == more and buying == high and maint == med and doors == 4 then	acc	0.0	0.31	0.07
(183) if safety == med and lug_boot == med and buying == med and maint == vhigh and doors == 4 and persons == more then	acc	0.0	0.31	0.07
(184) if safety == med and lug_boot == med and buying == low and maint == vhigh and doors == 5more and persons == more then	acc	0.0	0.31	0.07
(185) if persons == more and lug_boot == small and safety == high and buying == high and doors == 3 and maint == low then	acc	0.0	0.31	0.07
(186) if lug_boot == med and safety == med and persons == 4 and maint == vhigh and doors == 5more and buying == med then	acc	0.0	0.31	0.07
(187) if buying == low and maint == med and safety == med and doors == 3 and persons == 4 and lug_boot == med then	acc	0.0	0.31	0.07

(188) if persons == more and lug_boot == small and doors == 4 and safety == high and buying == low and maint == high then	acc	0.0	0.31	0.07
(189) if persons == more and maint == med and buying == low and safety == med and doors == 5more and lug_boot == small then	acc	0.0	0.31	0.07
(190) if lug_boot == med and doors == 2 and maint == med and persons == more and buying == low and safety == med then	acc	0.0	0.31	0.07
(191) if lug_boot == med and persons == 4 and buying == med and maint == med and safety == high and doors == 3 then	acc	0.0	0.31	0.07
(192) if persons == more and safety == high and lug_boot == small and buying == high and maint == high and doors == 3 then	acc	0.0	0.31	0.07
(193) if lug_boot == med and maint == high and doors == 3 and buying == low and safety == high then	acc	0.0	0.31	0.07
(194) if persons == more and safety == high and maint == med and lug_boot == small and buying == high and doors == 4 then	acc	0.0	0.31	0.07
(195) if lug_boot == med and doors == 2 and safety == high and maint == med and buying == med and persons == 4 then	acc	0.0	0.31	0.07
(196) if safety == med and lug_boot == med and persons == 4 and maint == low and buying == vhigh and doors == 5more then	acc	0.0	0.31	0.07
(197) if persons == more and buying == med and safety == med and maint == high and lug_boot == med and doors == 3 then	acc	0.0	0.31	0.07
(198) if persons == more and safety == high and doors == 2 and lug_boot == med and buying == low and maint == high then	acc	0.0	0.31	0.07

(199) if safety == med and persons == 4 and buying == low and doors == 4 and maint == vhigh and lug_boot == med then	acc	0.0	0.31	0.07
(200) if persons == more and lug_boot == small and buying == med and maint == low and doors == 4 and safety == med then	acc	0.0	0.31	0.07
(201) if maint == med and persons == more and safety == high and buying == med and doors == 5more and lug_boot == small then	acc	0.0	0.31	0.07
(202) if safety == med and doors == 3 and buying == high and maint == high and persons == more and lug_boot == med then	acc	0.0	0.31	0.07
(203) if doors == 2 and lug_boot == med and buying == vhigh and maint == med and persons == more and safety == high then	acc	0.0	0.31	0.07
(204) if buying == low and lug_boot == small and doors == 3 and maint == med and safety == med and persons == 4 then	acc	0.0	0.31	0.07
(205) if doors == 2 and maint == low and buying == med and safety == med and persons == 4 and lug_boot == med then	acc	0.0	0.31	0.07
(206) if buying == low and maint == vhigh and doors == 4 and lug_boot == small and persons == more and safety == high then	acc	0.0	0.31	0.07
(207) if safety == med and doors == 2 and persons == more and maint == low and buying == vhigh then	unacc	0.0	0.21	0.14
(208) if safety == med and doors == 2 and lug_boot == med and persons == more and buying == high and maint == high then	unacc	0.0	0.1	0.07
(209) if safety == med and doors == 2 and lug_boot == med and persons == 4 and buying == high and maint == high then	unacc	0.0	0.1	0.07

(210) if safety == med and doors == 2 and lug_boot == med and persons == 4 and buying == vhigh and maint == med then	unacc	0.0	0.1	0.07
(211) if safety == med and doors == 2 and lug_boot == med and persons == more and buying == med and maint == high then	unacc	0.0	0.1	0.07
(212) if safety == med and doors == 2 and persons == 4 and buying == high and lug_boot == med and maint == med then	unacc	0.0	0.1	0.07
(213) if safety == med and doors == 2 and lug_boot == med and persons == more and maint == med and buying == high then	unacc	0.0	0.1	0.07
(214) if safety == med and persons == 4 and doors == 3 and lug_boot == med and buying == high and maint == high then	unacc	0.0	0.1	0.07
(215) if safety == med and doors == 2 and persons == 4 and buying == vhigh and maint == low and lug_boot == med then	unacc	0.0	0.1	0.07
(216) if safety == med and persons == 4 and doors == 3 and lug_boot == med and buying == vhigh and maint == med then	unacc	0.0	0.1	0.07
(217) if safety == med and doors == 2 and lug_boot == med and persons == more and maint == vhigh and buying == med then	unacc	0.0	0.1	0.07
(218) if safety == med and doors == 2 and persons == 4 and buying == med and maint == high and lug_boot == med then	unacc	0.0	0.1	0.07
(219) if safety == med and persons == more and lug_boot == small and buying == med and maint == high and doors == 4 then	unacc	0.0	0.1	0.07
(220) if safety == med and lug_boot == med and maint == low and doors == 3 and persons == 4 and buying == vhigh then	unacc	0.0	0.1	0.07

(221) if doors == 2 and persons == more and buying == low and maint == vhigh and lug_boot == small and safety == high then	unacc	0.0	0.1	0.07
(222) if safety == med and doors == 2 and lug_boot == med and persons == more and buying == vhigh and maint == med then	unacc	0.0	0.1	0.07
(223) if safety == med and maint == low and buying == high and persons == 4 and lug_boot == med and doors == 2 then	unacc	0.0	0.1	0.07
(224) if safety == med and buying == med and maint == high and doors == 3 and persons == more and lug_boot == small then	unacc	0.0	0.1	0.07
(225) if doors == 2 and persons == more and buying == low and maint == high and lug_boot == small and safety == high then	unacc	0.0	0.1	0.07
(226) if safety == med and persons == 4 and doors == 3 and lug_boot == med and buying == high and maint == low then	unacc	0.0	0.1	0.07
(227) if safety == med and doors == 2 and maint == vhigh and buying == low and persons == more and lug_boot == med then	unacc	0.0	0.1	0.07
(228) if safety == med and buying == med and persons == 4 and maint == vhigh and doors == 2 and lug_boot == small then	unacc	0.0	0.1	0.07
(229) if maint == high and doors == 3 and buying == med and persons == 4 and lug_boot == med and safety == med then	unacc	0.0	0.1	0.07
(230) if lug_boot == small and persons == more and buying == vhigh and maint == low and doors == 5more and safety == med then	unacc	0.0	0.1	0.07
(231) if doors == 2 and maint == high and buying == high and lug_boot == small and persons == more and safety == high then	unacc	0.0	0.1	0.07

A.4 Mushroom Data Set

Rule	Class	Precision	Recall	Coverage
(1) if odor == f then	p	100.0	73.65	28.48
(2) if stalk-color-above-ring == g then	e	100.0	17.12	10.5
(3) if stalk-color-above-ring == p and bruises == t then	e	100.0	16.58	10.17
(4) if odor == p then	p	100.0	11.4	4.41
(5) if odor == a then	e	100.0	11.34	6.95
(6) if odor == l then	e	100.0	11.27	6.91
(7) if ring-type == e and population == s then	e	100.0	10.98	6.73
(8) if population == a then	e	100.0	10.91	6.69
(9) if odor == c then	p	100.0	9.28	3.59
(10) if stalk-color-below-ring == g and stalk-color-above-ring == w then	e	100.0	5.78	3.54
(11) if spore-print-color == r then	p	100.0	3.26	1.26
(12) if population == c then	p	100.0	2.41	0.93
(13) if cap-color == e and stalk-color-above-ring == w and stalk-color-below-ring == p then	e	100.0	2.06	1.26
(14) if stalk-color-below-ring == p and stalk-color-above-ring == w and cap-color == g then	e	100.0	1.84	1.13
(15) if cap-color == n and stalk-color-below-ring == p and stalk-color-above-ring == w then	e	100.0	1.81	1.11
(16) if cap-color == e and stalk-color-below-ring == w and stalk-color-above-ring == w then	e	100.0	1.66	1.02
(17) if habitat == l and stalk-color-below-ring == n then	e	100.0	1.34	0.82
(18) if cap-shape == s then	e	100.0	0.76	0.47
(19) if cap-color == g and stalk-color-above-ring == w and cap-surface == y and stalk-color-below-ring == w and stalk-shape == t then	e	100.0	0.76	0.47

(20) if cap-color == n and habitat == d and stalk-color-above-ring == w and stalk-color-below-ring == w and gill-color == n then	e	100.0	0.54	0.33
(21) if cap-color == n and gill-color == w and stalk-color-below-ring == w and stalk-shape == t and stalk-color-above-ring == w then	e	100.0	0.54	0.33
(22) if stalk-surface-above-ring == y and cap-color == n then	e	100.0	0.51	0.31
(23) if gill-color == u and stalk-color-above-ring == w and cap-color == n and stalk-color-below-ring == w then	e	100.0	0.47	0.29
(24) if gill-color == p and cap-color == n and stalk-root == b and stalk-color-below-ring == w and stalk-color-above-ring == w then	e	100.0	0.43	0.27
(25) if habitat == u and cap-surface == f and cap-shape == x and spore-print-color == n then	e	100.0	0.36	0.22
(26) if habitat == u and cap-surface == f and cap-shape == f and spore-print-color == n then	e	100.0	0.33	0.2
(27) if habitat == u and gill-color == g and spore-print-color == k then	e	100.0	0.25	0.16
(28) if cap-color == g and gill-color == n and cap-surface == f and stalk-color-below-ring == w and bruises == t and stalk-color-above-ring == w then	e	100.0	0.25	0.16
(29) if ring-number == t and cap-color == c then	e	100.0	0.22	0.13
(30) if ring-number == t and habitat == p and cap-color == p then	e	100.0	0.22	0.13
(31) if ring-number == t and cap-color == n and bruises == t then	e	100.0	0.22	0.13
(32) if cap-color == g and gill-color == w and cap-surface == f and stalk-color-above-ring == w and stalk-color-below-ring == w then	e	100.0	0.22	0.13

(33) if ring-number == t and cap-color == g then	e	100.0	0.18	0.11
(34) if habitat == u and cap-surface == f and spore-print-color == k and cap-shape == f and gill-color == n then	e	100.0	0.11	0.07
(35) if cap-color == g and stalk-color-above-ring == w and population == y and stalk-color-below-ring == w and gill-color == u and cap-surface == f then	e	100.0	0.11	0.07
(36) if gill-color == p and cap-color == g and stalk-color-below-ring == w and population == v and cap-surface == f and bruises == t and stalk-color-above-ring == w then	e	100.0	0.11	0.07
(37) if habitat == u and cap-surface == f and cap-shape == f and gill-color == k and spore-print-color == k then	e	100.0	0.07	0.04
(38) if gill-color == u and cap-color == g and stalk-color-above-ring == w and stalk-color-below-ring == w and bruises == t and spore-print-color == k and cap-shape == f then	e	100.0	0.04	0.02
(39) if cap-color == g and gill-color == u and stalk-color-above-ring == w and stalk-color-below-ring == w and bruises == t and cap-shape == x and population == v and cap-surface == f then	e	100.0	0.04	0.02
(40) if habitat == u and cap-surface == f and spore-print-color == k and gill-color == n and cap-shape == x then	e	0.0	0.14	0.09
(41) if habitat == u and cap-surface == f and gill-color == p and cap-shape == x and spore-print-color == k then	e	0.0	0.14	0.09
(42) if cap-color == g and stalk-color-above-ring == w and population == y and stalk-color-below-ring == w and stalk-shape == t and cap-surface == f and gill-color == p then	e	0.0	0.14	0.09

(43) if cap-color == g and habitat == u and gill-color == k and cap-shape == x and spore-print-color == k then	e	0.0	0.07	0.04
(44) if habitat == u and population == y and cap-shape == f and gill-color == p and spore-print-color == k then	e	0.0	0.07	0.04
(45) if cap-color == g and habitat == u and spore-print-color == k and gill-color == p and cap-shape == f and population == v then	e	0.0	0.04	0.02

References

[1] J. Cendrowska, *PRISM: An Algorithm for Inducing Modular Rules*, vol. 27, pp. 349–370. 1987.

[2] J. R. Quinlan, *Induction of decision trees.* 1986.

[3] M. Sànchez-Marrè, *Supervised and Experiential Learning. Rule-Based Classifiers*, pp. 18–28. 2022-23.

[4] *UCI Machine Learning Repository: Congressional Voting Records Data Set.*

[5] *UCI Machine Learning Repository: Tic-tac-toe Endgame Data Set.*

[6] *UCI Machine Learning Repository: Mushroom Data Set.*

[7] *UCI Machine Learning Repository: Car Evaluation Data Set.*