Supervised and experiential Learning

Rule-Based Classifiers

Gerard Caravaca Ibáñez
gerard.caravaca.ibanez@estudiantat.upc.edu

30/3/2023

Contents

1 Introduction

A rule-based classifier is a type of machine learning model that makes predictions based on a set of predefined rules. These rules are created by domain experts or generated automatically from the training data. Rule-based classifiers are widely used in various domains, including natural language processing, computer vision, and predictive maintenance. The main advantage of rule-based classifiers is their transparency, as they provide clear explanations for their predictions, making them suitable for applications where interpretability is critical. However, rule-based classifiers have some limitations, such as their inability to handle complex relationships between features and their susceptibility to overfitting. Despite these limitations, rule-based classifiers remain a popular choice in many real-world applications due to their simplicity, transparency, and effectiveness.

In this work, I will analyze the PRISM algorithm, which is a rule-based classifier that uses probabilistic logic programming to learn rules from data. The main goal of this is to explore the applicability of this type of classifiers in several domains to see how the algorithm handles different dataset sizes. For this, I will first describe the PRISM algorithm. Subsequently, I will define the various datasets used and their particularities. Finally, I will show how the algorithm works on each of the datasets.

2 PRISM

The PRISM [4] algorithm was introduced by Jadzia Cendrowska in 1987 as an approach for the induction of modular rules from data. The goal of PRISM is to find rules that explain relationships between object attributes and classify objects into categories. The basic approach of this algorithm is based on the fact that from a training set the algorithm generates a series of rules, formed by the conjunction of attributes, which are able to classify the examples of the set in their corresponding class. The idea is that if the set is sufficiently representative of the domain, then these rules can be used to classify the rest of the objects in the same domain.

2.1 Rule generation algorithm

The most important step of the algorithm is the rule generation step. For this, the algorithm follows the steps represented in algorithm 1. In order to understand the algorithm well, it is important to understand that rules are evaluated by their Precision and that a rule is perfect if its precision is equal to 1. Following this concept, the best rule of a set of rules is the one with the highest precision and that in case of a tie, the rule with the highest coverage will be chosen.

The algorithm initializes the data for each class to the entire dataset and progressively constructs rules by selecting the most accurate rule in each iteration of the while loop. Once a rule is found, it is added to the ruleset for instances *inst*, which is then updated to remove instances covered by the rule. This process continues until all classes are covered.

To select the best rule, a new rule is initialized and the set E is updated containing the remaining instances of *inst*. The algorithm exhaustively searches for unused attributes in the rule and selects the one with the highest precision. In case of a tie, the rule with the greatest coverage is chosen, as explained previously.

Once the best (attribute, value) pair is found, the rule is extended by assigning the value to the attribute. E is then updated to contain only the examples covered by the rule, and the rule is evaluated in E to determine the exit condition. The loop continues while the rule coverage is greater than one item, if the rule has 100% accuracy, or if the rule is complete, meaning that all attributes in the dataset have been used and it cannot be extended.

Algorithm 1 PRISM rule generation

Require: *instances*: matrix of instances; *targets*: array of instance labels
Ensure: *rules*: set of rules

$rules \leftarrow \emptyset$
for each class c **in** *classes* **do**
  $E \leftarrow i \mid i \in [0, instances.shape[0]] \wedge targets[i] = c$ ▷ Indexes of instances belonging to c
  while remaining instances of class c in E **do**
    $rule \leftarrow emptyRule$
    $inst \leftarrow E.copy()$
    while not $rule.perfect()$ **and** $rule.coverage() > 1$ **and** attributes available **do**
      $rules \leftarrow \emptyset$
      for each pair attribute-value (A, V) available **do**
        $Rav \leftarrow Rule(c)$
        $Rav.insertValue(attr, val)$
        $Rav.evaluate(instances[inst], targets[inst])$ ▷ Update accuracy and coverage
        $rules.append(Rav)$
      end for
      $best \leftarrow rules[getBestRuleId(rules)]$
      $rule.extend(best)$
      $inst \leftarrow rule.inference(inst, instances)$ ▷ Get instances covered by rule
      $rule.evaluate(instances[inst], targets[inst])$
    end while
    $rules.append(rule)$
    $E \leftarrow removeInstances(E, rule)$ ▷ Remove instances covered by rule
  end while
end for
return *rules*

2.2 Advantages and limitations

Having described how the algorithm works, the main advantages and disadvantages of the algorithm will be discussed. This point will be used later in the experimentation to be able to analyse the results with the appropriate criteria.

On the one hand, PRISM is a powerful and flexible classification tool that can handle complex datasets with multiple classes and attributes. Moreover, It is rule-based, which means that is easier to understand and interpret the resulting model than using other techniques. This fact makes it possible for the incorporation of expert knowledge into the model, which can lead to more accurate and reliable results.

On the other hand, PRISM is based on a greedy search algorithm, which means that it may

not always find the optimal set of rules. In addition, it can be sensitive to noisy, which can lead to overfitting or underfitting of the model. Furthermore, it can be computationally expensive, especially when dealing with large datasets or complex models. A final problem is that it is intended to be used with categorical attributes and it is not able to handle missing values. Therefore in order to be applied to datasets with continuous variables a preprocessing is required.

In summary, PRISM is a powerful and flexible classification tool that can be useful for a wide range of applications. However, it has its limitations, and its effectiveness depends on the quality of the data and the expertise of the user.

3 Domain

In order to be able to analyse the algorithm in different domains, it has been decided to use three different datasets of different sizes. One of them will be of small size, less than 500 instances, the second one will be of medium size, between 500 and 200 instances, and the third one will be the largest with more than 2000 instances. Retrieved from the UCI dataset repository [5], the datasets utilized for the comparison are all categorical in nature and have no missing values. These two aspects are problematic for prism, thus they were excluded. The datasets that were ultimately selected for comparison are the hayes-roth, balance-scale and kr-vs-kp datasets.

3.1 Hayes-Roth

The Hayes-Roth dataset [2] is a small, relatively simple dataset in the field of machine learning. It was originally collected by Hayes-Roth, a researcher who was studying expert systems in the early days of artificial intelligence. The dataset was donated to the UCI machine learning repository in 1989 and has since been widely used as a benchmark for classification algorithms.

The dataset consists of 160 instances and 5 attributes (see Table 1). Each instance represents a person, and the attributes describe various characteristics of that person. The attributes are:

- Name: the name of the person (a nominal attribute)
- Hobby: the person's hobby (a nominal attribute)
- Age: the person's age (a nominal attribute)
- Educational level: the person's level of education (a nominal attribute)
- Class: the person's class (a categorical attribute with 3 possible values: 1, 2, or 3)

Data Set Type:	Multivariate	**Instances:**	160	**Area:**	Social
Attribute Type:	Categorical	**Attributes:**	5	**Date Donated**	1989-03-01
Tasks:	Classification	**Missing Values?**	No	**Web Hits:**	127201

Tab. 1: Hayes Roth dataset features.

The dataset contains no missing values and all attributes are categorical. The class variable is the target variable for classification tasks, and the goal is to predict the class of a new instance

based on its other attributes.

To summarize, this is a relatively small domain, but it has been used as a benchmark for a variety of classification algorithms, including decision trees, nearest neighbor classifiers, and support vector machines. The simplicity of the dataset makes it a good choice for exploring the behavior of different classification algorithms, and its small size makes it easy to work with and analyze.

3.2 Balance-Scale

The balance-scale dataset [1] is a commonly used benchmark dataset in machine learning for classification tasks. On this case, we are talking about a medium size dataset. It was originally created to study balance scales, and is often used to evaluate the performance of classification algorithms in predicting the correct balance scale outcome.

The dataset consists of 625 instances (see Table 2), each of which describes a balance scale problem in terms of its attributes. The four attributes of the dataset are:

- Left-Weight: the weight of the objects on the left side of the balance scale.
- Left-Distance: the distance of the objects on the left side of the balance scale from the fulcrum.
- Right-Weight: the weight of the objects on the right side of the balance scale.
- Right-Distance: the distance of the objects on the right side of the balance scale from the fulcrum.

The target variable is the balance scale outcome, which can take on one of three possible values:

- B: the balance scale is balanced
- L: the balance scale tips to the left
- R: the balance scale tips to the right

Data Set Type:	Multivariate	**Instances:**	625	**Area:**	Social
Attribute Type:	Categorical	**Attributes:**	4	**Date Donated**	1994-04-22
Tasks:	Classification	**Missing Values?**	No	**Web Hits:**	333949

Tab. 2: Balance Scale dataset features.

The balance-scale dataset is well-suited for classification tasks because it provides a range of different balance scale scenarios, each of which can be used to train and test classification algorithms. The dataset is balanced, with an equal number of instances for each of the three possible outcomes. This makes it a good choice for evaluating the performance of classification algorithms on comparison to imbalanced datasets.

3.3 King-Rook vs. King-Pawn

The Chess (King-Rook vs. King-Pawn) dataset [3] is a multivariate dataset containing positions of the white king, white rook, and black king on a chessboard, along with the label indicating whether the game ends in a win or a draw for white. There are a total of 3196 instances in this dataset, and it is exclusively categorical with no missing values.

It consists of 36 attributes, including the file (a unique identifier for each instance), the rank and file of the white king, the rank and file of the white rook, and the rank and file of the black king. Each attribute is categorical and takes on values corresponding to the ranks and files on the chessboard.

Data Set Type:	Multivariate	**Instances:**	3196	**Area:**	Game
Attribute Type:	Categorical	**Attributes:**	36	**Date Donated**	1989-08-01
Associated Tasks:	Classification	**Missing Values?**	No	**Web Hits:**	148315

Tab. 3: King-Rook vs. King-Pawn dataset features.

The dataset was created by generating random legal positions of the white king, white rook, and black king, and then using an endgame tablebase to determine the outcome of each game. The resulting positions were stored in a database, along with the corresponding labels indicating whether the game ends in a win or a draw for white.

This domain is often used for classification tasks, where the goal is to predict whether a given chess position will result in a win or a draw for white. It has been used in machine learning research to evaluate the performance of various classification algorithms, and has also been used as a benchmark dataset for testing chess-playing programs.

4 Experimentation

This section will explain the method that has been used to evaluate the algorithm in the various domains. In addition, the results obtained in each dataset will be presented and the performance of the classifier will be analysed. The goal of this experiments is to assess the performance of PRISM in terms of accuracy, coverage, execution time and number of rules generated.

To evaluate the algorithm, three experiments will be run, one using each of the datasets explained above. In each of the experiments, the dataset will be divided into a training set and a test set, the split will be done randomly and with 20% of the data for testing. The model will generate the rules using the training set. Subsequently, accuracy and coverage metrics will be extracted by inferring the classes of the instances in the test set using the previously generated rules. This process will be performed 10 times for each experiment and the average of the metrics will be obtained, with the intention of obtaining meaningful results.

4.1 Experiment 1 - Hayes-Roth

In this particular experiment, in addition to presenting the results obtained in this dataset, the rules generated by the algorithm will also be analysed. This dataset has been chosen to analyse the rules because, as we will see in the following sections for the other datasets, the number of

rules generated made manual analysis unfeasible. However, an example rule set for each dataset is provided in the appendix section.

As can be seen in the *Rules*, 32 rules were derived from the Hayes-Roth dataset. Out of these, it can be easily seen that for each class the algorithm chooses the rules generated with the highest accuracy, in particular 100% precision for the first ones, and that subsequently the precision decreases. The same behaviour can be seen with the recall. This can be understood from the functioning of the algorithm explained above. On analyzing the dataset, it was observed that the antecedent of the rules with precision less than 1.0 corresponded to conflicting items, where multiple instances had the same antecedent but different class labels. In contrast, there are attributes that can define a class on its own. This can be seen in rules 30, 31 and 32 which define the last class with only one attribute.

Rules

```
1  marital_status = 1 AND age = 1 AND educational_level = 2 -> 1   P=1.0 R=0.21
2  educational_level = 1 AND age = 1 -> 1   P=1.0 R=0.24
3  marital_status = 1 AND educational_level = 1 AND age = 2 -> 1   P=1.0 R=0.19
4  age = 3 AND marital_status = 1 AND educational_level = 1 -> 1   P=1.0 R=0.02
5  age = 3 AND marital_status = 1 AND educational_level = 3 -> 1   P=1.0 R=0.02
6  age = 3 AND marital_status = 1 AND hobby = 1 AND educational_level = 2 -> 1   P=0.67 R=0.07
7  educational_level = 3 AND marital_status = 3 AND age = 1 -> 1   P=1.0 R=0.02
8  educational_level = 3 AND age = 1 AND hobby = 3 AND marital_status = 2 -> 1   P=0.67 R=0.07
9  age = 3 AND marital_status = 1 AND educational_level = 2 AND hobby = 2 -> 1   P=0.67 R=0.07
10 marital_status = 3 AND hobby = 2 AND educational_level = 1 -> 1   P=1.0 R=0.05
11 educational_level = 3 AND age = 1 AND marital_status = 2 AND hobby = 2 -> 1   P=0.33 R=0.07
12 age = 3 AND marital_status = 1 AND educational_level = 2 AND hobby = 3 -> 1   P=0.5 R=0.05
13 educational_level = 3 AND age = 1 AND marital_status = 2 AND hobby = 1 -> 1   P=0.33 R=0.07
14 marital_status = 3 AND hobby = 3 AND age = 2 AND educational_level = 1 -> 1   P=0.33 R=0.07
15 age = 2 AND educational_level = 3 -> 2   P=1.0 R=0.05
16 age = 2 AND marital_status = 2 AND educational_level = 1 -> 2   P=1.0 R=0.23
17 educational_level = 2 AND marital_status = 3 -> 2   P=1.0 R=0.05
18 educational_level = 2 AND marital_status = 2 AND age = 1 -> 2   P=1.0 R=0.2
19 age = 2 AND educational_level = 2 AND marital_status = 1 -> 2   P=1.0 R=0.16
20 educational_level = 3 AND marital_status = 2 AND age = 3 -> 2   P=1.0 R=0.02
21 educational_level = 3 AND marital_status = 2 AND hobby = 2 AND age = 1 -> 2   P=0.67 R=0.07
22 marital_status = 3 AND age = 2 AND hobby = 1 -> 2   P=1.0 R=0.09
23 educational_level = 3 AND marital_status = 2 AND hobby = 1 AND age = 1 -> 2   P=0.67 R=0.07
24 age = 3 AND marital_status = 2 AND educational_level = 2 -> 2   P=1.0 R=0.02
25 age = 3 AND educational_level = 2 AND hobby = 3 AND marital_status = 1 -> 2   P=0.5 R=0.05
26 marital_status = 3 AND hobby = 3 AND age = 2 AND educational_level = 1 -> 2   P=0.67 R=0.07
27 age = 3 AND educational_level = 2 AND marital_status = 1 AND hobby = 2 -> 2   P=0.33 R=0.07
28 educational_level = 3 AND marital_status = 2 AND age = 1 AND hobby = 3 -> 2   P=0.33 R=0.07
29 age = 3 AND educational_level = 2 AND marital_status = 1 AND hobby = 1 -> 2   P=0.33 R=0.07
30 educational_level = 4 -> 3   P=1.0 R=0.46
31 age = 4 -> 3   P=1.0 R=0.38
32 marital_status = 4 -> 3   P=1.0 R=0.38
```

For more information we can look at the plot of Figure 1. This plot shows the recall of each rule, which provides an indication of the proportion of instances belonging to a specific class that are encompassed by a single rule. The plot clearly reinforces the comments explained in the previous paragraph. As expected, the least specific rules are the ones that get the most recall.

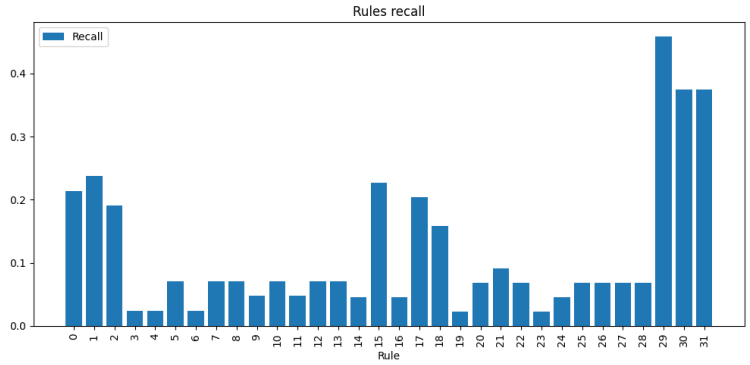

Fig. 1: Recall distribution in Hayes-Roth rules.

Using this set of rules generated to predict the class of the samples in the test set, the metrics shown in Table 4 were obtained. It should be noted that the variation in the results is due to the randomness of the split of the data. On average, an accuracy of 68% was obtained using about 32 rules and with an execution time of 90 milliseconds. Another thing to note is that with the 32 rules generated on average, all the cases represented in the test set can be covered in all cases.

It	NumRules	Accuracy	Coverage	Runtime (s)
0	30	0.66	1	0.09
1	34	0.65	1	0.10
2	35	0.68	1	0.10
3	33	0.76	1	0.09
4	31	0.68	1	0.09
5	30	0.63	1	0.08
6	31	0.67	1	0.08
7	34	0.71	1	0.09
8	32	0.69	1	0.09
9	36	0.71	1	0.10
AVG	**32.6**	**0.68**	**1**	**0.09**

Tab. 4: Hayes Roth results.

4.2 Experiment 2 - Balance-Scale

This experiment will show how the algorithm performs on a medium sized dataset. In order to maintain fairness between experiments, exactly the same process has been followed as in the previous experiment. Since the number of rules makes it impossible to analyse the set rule by rule, in this case, the recall distribution will be analysed. This accounts for much of the information in the rules since in most cases the precision is one. In this case, the distribution of recall among the rules is much more equal, with most rules having a much lower recall than in the previous experiment. This explains why the number of rules generated is much higher (see Figure 2).

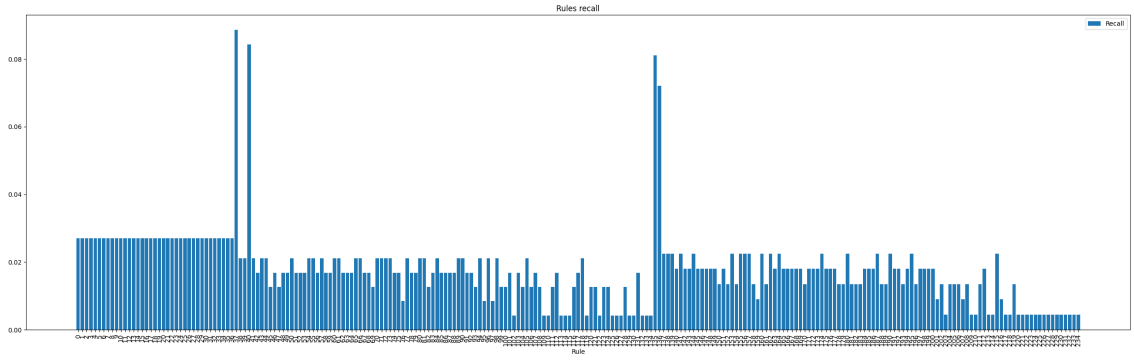

Fig. 2: Recall distribution in Balance-Scale rules.

The results obtained in this case are shown in Table 5. As can be seen from the results, the increase in the number of instances has significantly increased both the number of rules generated and the execution time. In addition, the average accuracy obtained has decreased by 63%. This is not only because the dataset is larger, but also because the dataset is harder to generalise, which means that more precise rules are needed. In short, in this experiment the number of instances has been multiplied by 4, resulting in a 7-fold increase in the number of rules generated and a 40-fold increase in the execution time.

	NumRules	**Accuracy**	**Coverage**	**Runtime (s)**
0	228	0.61	1	3.53
1	252	0.70	1	4.13
2	232	0.67	1	3.48
3	220	0.67	1	3.18
4	249	0.63	1	3.85
5	216	0.55	1	3.15
6	222	0.63	1	3.38
7	229	0.58	1	3.41
8	246	0.66	1	3.82
9	231	0.62	1	3.53
AVG	**232.5**	**0.63**	**1**	**3.54**

Tab. 5: Balance Scale results.

4.3 Experiment 3 - King-Rook vs. King-Pawn

Finally, the last experiment assesses the application of this method on large datasets of more than 2000 instances. The same protocol was strictly followed for this experiment as for the previous ones. As in the previous experiment, for this one it has been decided to analyse the recall distribution, as shown in Figure 3. For this dataset it can be seen that most instances can be classified using few rules, these are the rules with the highest recall. On the other hand, there are some instances that trigger the generation of more specific rules, these are the rules with lower recall. Compared to the previous experiment, we have a dataset with more instances but easier to classify by generating rules with higher recall.

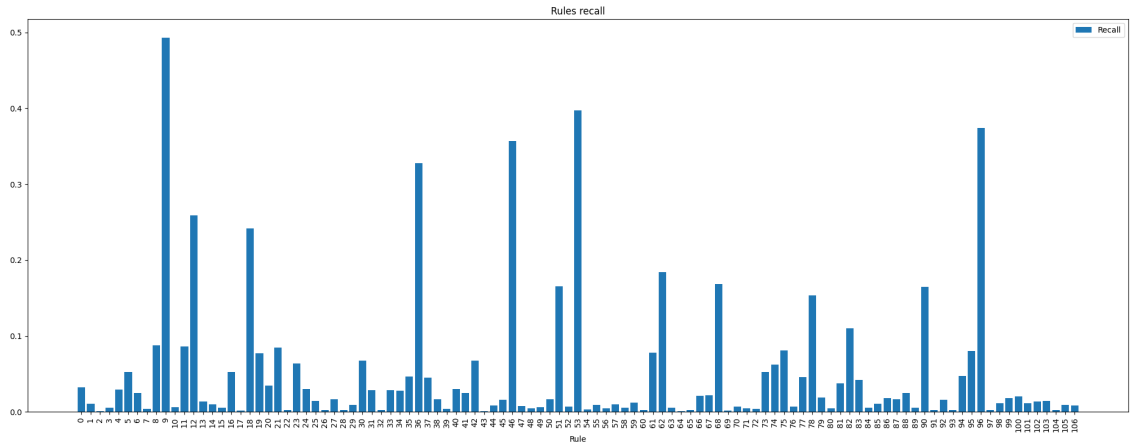

Fig. 3: Recall distribution in King-Rook vs. King-PawnKing-Rook vs. King-Pawn rules.

The results are summarized in Table 6. Although following the previous experiment one might expect a reduction in performance and an increase in the number of rules generated by the algorithm, it can be clearly seen that this is not the case. Surprisingly, for this dataset it was possible to obtain an accuracy of 99% using on average only 105 rules. To understand these results, the properties of the dataset used must be reviewed once again. In this case the dataset had only 2 classes to classify. Therefore it is understandable that the accuracy is higher, as it is easier to differentiate two classes than a larger number of classes. The same reasoning can be followed for the number of rules generated. It is reasonable that the number of rules is smaller as it is less hard to generalise a dataset with only 2 classes. In contrast, the execution time continues to grow considerably. In this case, for a 5-fold increase in the number of instances, the execution time has increased by a factor of 7. This is explained by the fact that although fewer rules have been generated, many more have had to be explored. This is because the number of attributes to be taken into account in this data set is much larger (36 attributes), so the possible combinations between attribute and value grow rapidly.

	NumRules	**Accuracy**	**Coverage**	**Runtime (s)**
0	100	0.99	1	36.1
1	112	0.99	1	33.2
2	99	0.98	1	21.8
3	110	0.98	1	24.0
4	101	0.99	1	22.5
5	104	0.98	1	22.5
6	111	0.99	1	24.9
7	101	0.99	1	22.1
8	107	0.98	1	22.9
9	109	0.98	1	24.1
AVG	**105.4**	**0.99**	**1**	**25.4**

Tab. 6: King-Rook vs. King-Pawn results.

5 Conclusions

To summarize, in this work I have managed to implement a rule-based classifier system based on the PRISM algorithm. The algorithm has been analysed, giving an explanation of its main characteristics, the steps to follow to apply it and its advantages and disadvantages. Then, I have analysed how the system helps to solve classification problems in domains with different characteristics and with three clearly differentiated sizes. From these different domains, three different experiments, one for each dataset, have been carried out. The first experiment allowed me to manually analyse each of the rules generated with their corresponding accuracy and coverage. At this point we were able to see more clearly the criteria followed by the algorithm. On the other hand, the second and third experiments allowed us to see the results of the same algorithm on two completely different datasets.

The facts highlighted in the previous paragraph lead me to draw some conclusions about what has been experienced in this work. Firstly, the results of the last experiment show that the algorithm is very suitable for datasets with few classes to be classified. This makes it clear that the key factor for the performance of the algorithm is not the number of instances but rather the difficulty of the dataset. On the downside, it is worth noting that as the dimensionality of the dataset grows and the number of possible values for each attribute increases, the number of possible classes grows and therefore the algorithm will take much longer to converge. This means that PRISM cannot be applied to datasets with very high dimensionality.

Finally, I would like to highlight the greatest virtue of this kind of classifiers, namely explainability. The fact of representing the knowledge of the model by rules makes the solution easily understandable, something that does not happen in most machine learning algorithms. Following this concept, a domain expert could be introduced into the rule creation process. This would allow the algorithm to perform better and could avoid the problem of overfitting. This is a clear branch of research to continue this work in the future.

6 Implementation details

This last section will explain in detail how the algorithm has been implemented and the steps to follow in order to run the experiments.

6.1 Directory structure

The implementation directory follows the structure shown in Figure 4. The files shown in the figure contain the following information:

- Data: contains the files for each of the datasets used.
- Results: this directory was created to store the rules generated by the algorithm and the results obtained in each iteration.
- Source: this directory contains all the classes that implement the algorithm and each of the experiments.
  - Utils: contains extra functionalities that help to read datasets and write results.
  - Prism: contains the class which implements the main PRISM algorithm.

  - Rule: contains the class which implements each of the rules generated by the model.

- Main: is the main file to be run to start the experiments.
- Venv: contains the files that make up the execution environment.
- Requirements: This file was created to facilitate the installation of necessary libraries. Basically the libraries used are pandas and numpy.

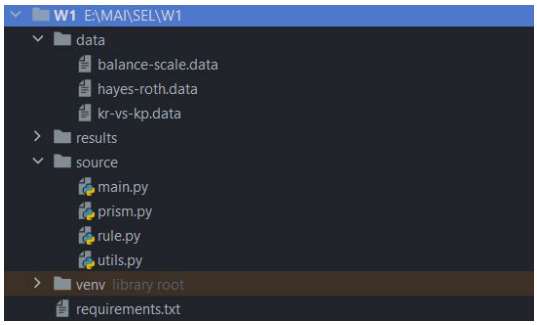

Fig. 4: Directory structure.

6.2 Python implementation

The algorithm was implemented using Python3.10 [6] and the packages pandas and numpy for loading and formatting the dataset before processing. In addition, 2 different classes have been created to implement the algorithm and another class has added to facilitate the reading of data and writing of results:

- **Rule**: python class containing the attributes and functions needed to represent a rule in the PRISM algorithm. The implemented attributes and functions are the following:
  - **Attributes**:
    * available-attributes: set of possible attributes that can be used by the rule.
    * class: target class.
    * p: number of correct items covered by the rule.
    * t: number of items covered by the rule.
    * antecedent: dictionary attribute-value storing the antecedent of the rule.
  - **Functions**:
    * accuracy(): return p/t current value of the rule.
    * coverage(): return t attribute.
    * is-perfect(): evaluate if accuracy is equal to 1.
    * precision-recall(X,Y): evaluate precision and recall of a single rule over the training dataset.
    * not-used-attributes(): return the set of available attributes that are not used by the rule.
    * insert-value(attr, value): add a pair attr-value in the antecedent.
    * evaluate(): update p and t attributes.

    - extend(rule): extend the current rule with the antecedent of a new rule.
    - inference(ids, instances): return the instances covered by the rule in the ids set.
    - covered(instance): evaluate if the rule covers an specific instance.
    - print-rule(): a friendly representation of the rule.

- **PRISM**: python class containing the attributes and functions needed to compute the PRISM algorithm. The implemented attributes and functions are the following:
  - **Attributes:**
    - attributes: set of attribute names in the input dataset.
    - instances: set of input instances.
    - targets: set of input classes.
    - rules: set of rules generated.
    - classes: set of class names in the input dataset.
  - **Functions**:
    - fit(): main method to generate rules from data. This function corresponds to the pseudocode shown in algorithm 1.
    - predict(): method that infer labels from generated rules. Return an array of predictions where if an instance is not covered, return None for that item.

- **Utils**: python file containing extra functionalities for reading data and extracting results. The implemented tools are:
  - **Functions**:
    - read(filename): read a dataset file from the data directory.
    - write-rules(path, rules, X, Y): write rules in a .rules file with recall and precision information.
    - PR-plot(path, rules, X, Y): generate a plot of the recall of each rule.

6.3 Execution instructions

In this section, we will go over the steps for running the code. Although the project was built using Python 3.10, it should also work with any python3 version. You can find all the necessary dependencies in the requirements.txt file located in the root directory of the project. The following execution steps has been tested both in Windows and Linux terminal:

1. Open the project folder in the terminal.
2. Create virtual environment:

```
$ python3 -m venv venv/
$ source venv/bin/activate
$ pip3 install -r requirements.txt
```

- If you do not have the virtual environment package installed, you can install it with:

```
$ sudo apt-get install python3.8-venv
```

3. Execute an experiment:

```
$ python3 main.py -d DATASET_NAME -n NUM_ITERATIONS -s RESULT_FOLDER
```

   (a) The parameters of this command are the following:

      i. DATASET-NAME: refers to the name of the dataset. In this work the available dataset names are: *hayes-roth*, *balance-scale* and *kr-vs-kp*.

      ii. NUM-ITERATIONS: refers to the number of iterations to get the average performance. In the experiments carried out, this parameter is always *10*.

      iii. RESULT-FOLDER: refers to the path where the results will be saved. In the experiments carried out, this parameter is always *results/*.

   (b) An example of this is:

```
$ python main.py -d balance-scale -n 10 -s results/ > result.txt
```

4. After executing this command the rules of each iteration of the model will be saved in the indicated directory, in a file named *DatasetNameIterationNum.rules*. In addition, the metrics extracted from each run will be saved in an .xlsx file format. Finally, the terminal will display a summary of the complete experiment run, which you can save in a .txt file by adding *> results.txt* to the end of the above command.

7 Appendix

7.1 Example rules - Experiment 2

Rules

```
Right-Weight = 2 AND Left-Distance = 2 AND Left-Weight = 3 AND Right-Distance = 3 -> B   P=1.0 R=0.03
Left-Weight = 1 AND Right-Distance = 1 AND Left-Distance = 5 AND Right-Weight = 5 -> B   P=1.0 R=0.03
Left-Weight = 2 AND Right-Distance = 2 AND Left-Distance = 5 AND Right-Weight = 5 -> B   P=1.0 R=0.03
Right-Distance = 4 AND Left-Weight = 4 AND Left-Distance = 1 AND Right-Weight = 1 -> B   P=1.0 R=0.03
Right-Weight = 2 AND Left-Weight = 2 AND Right-Distance = 3 AND Left-Distance = 3 -> B   P=1.0 R=0.03
Right-Weight = 3 AND Left-Weight = 3 AND Right-Distance = 4 -> B   P=1.0 R=0.03
Left-Weight = 1 AND Right-Weight = 1 AND Left-Distance = 2 AND Right-Distance = 2 -> B   P=1.0 R=0.03
Right-Weight = 2 AND Left-Distance = 2 AND Left-Weight = 1 AND Right-Distance = 1 -> B   P=1.0 R=0.03
Right-Weight = 3 AND Left-Distance = 3 AND Right-Distance = 4 AND Left-Weight = 4 -> B   P=1.0 R=0.03
Left-Weight = 2 AND Right-Distance = 2 AND Right-Weight = 1 AND Left-Distance = 1 -> B   P=1.0 R=0.03
Right-Distance = 5 AND Left-Distance = 5 AND Left-Weight = 5 AND Right-Weight = 5 -> B   P=1.0 R=0.03
Right-Weight = 2 AND Left-Weight = 2 AND Left-Distance = 4 AND Right-Distance = 4 -> B   P=1.0 R=0.03
Right-Weight = 3 AND Left-Distance = 3 AND Left-Weight = 5 AND Right-Distance = 5 -> B   P=1.0 R=0.03
Left-Weight = 1 AND Right-Weight = 1 AND Left-Distance = 5 AND Right-Distance = 5 -> B   P=1.0 R=0.03
Right-Weight = 2 AND Left-Distance = 2 AND Right-Distance = 4 AND Left-Weight = 4 -> B   P=1.0 R=0.03
Right-Distance = 2 AND Left-Weight = 2 AND Left-Distance = 3 AND Right-Weight = 3 -> B   P=1.0 R=0.03
Left-Weight = 1 AND Right-Distance = 1 AND Left-Distance = 1 AND Right-Weight = 1 -> B   P=1.0 R=0.03
Right-Weight = 4 AND Left-Distance = 4 AND Left-Weight = 3 AND Right-Distance = 3 -> B   P=1.0 R=0.03
Right-Weight = 2 AND Left-Distance = 1 AND Left-Weight = 2 AND Right-Distance = 1 -> B   P=1.0 R=0.03
Right-Distance = 2 AND Left-Distance = 2 AND Left-Weight = 4 AND Right-Weight = 4 -> B   P=1.0 R=0.03
Right-Weight = 3 AND Left-Weight = 3 AND Left-Distance = 1 AND Right-Distance = 1 -> B   P=1.0 R=0.03
Left-Weight = 1 AND Left-Distance = 4 AND Right-Distance = 2 AND Right-Weight = 2 -> B   P=1.0 R=0.03
Left-Weight = 5 AND Right-Distance = 5 AND Right-Weight = 2 AND Left-Distance = 2 -> B   P=1.0 R=0.03
Left-Distance = 3 AND Right-Distance = 3 AND Right-Weight = 1 AND Left-Weight = 1 -> B   P=1.0 R=0.03
Left-Distance = 5 AND Right-Distance = 5 AND Right-Weight = 3 AND Left-Weight = 3 -> B   P=1.0 R=0.03
Right-Distance = 4 AND Left-Weight = 4 AND Left-Distance = 4 AND Right-Weight = 4 -> B   P=1.0 R=0.03
Left-Weight = 2 AND Right-Distance = 2 AND Right-Weight = 4 AND Left-Distance = 4 -> B   P=1.0 R=0.03
Left-Weight = 5 AND Right-Weight = 5 AND Left-Distance = 2 AND Right-Distance = 2 -> B   P=1.0 R=0.03
Left-Distance = 3 AND Right-Weight = 3 AND Left-Weight = 3 AND Right-Distance = 3 -> B   P=1.0 R=0.03
Left-Distance = 5 AND Right-Distance = 5 AND Left-Weight = 2 AND Right-Weight = 2 -> B   P=1.0 R=0.03
Right-Distance = 4 AND Left-Distance = 2 AND Left-Weight = 2 AND Right-Weight = 1 -> B   P=1.0 R=0.03
Left-Weight = 1 AND Right-Distance = 1 AND Right-Weight = 3 AND Left-Distance = 3 -> B   P=1.0 R=0.03
Left-Distance = 1 AND Right-Weight = 2 AND Left-Weight = 4 AND Right-Distance = 2 -> B   P=1.0 R=0.03
Left-Weight = 5 AND Right-Distance = 5 AND Right-Weight = 1 AND Left-Distance = 1 -> B   P=1.0 R=0.03
Right-Weight = 5 AND Left-Distance = 5 AND Left-Weight = 4 AND Right-Distance = 4 -> B   P=1.0 R=0.03
Left-Weight = 1 AND Right-Weight = 4 AND Left-Distance = 4 AND Right-Distance = 1 -> B   P=1.0 R=0.03
Right-Distance = 3 AND Left-Weight = 5 AND Right-Weight = 5 AND Left-Distance = 3 -> B   P=1.0 R=0.03
Right-Distance = 1 AND Left-Weight = 5 -> L   P=1.0 R=0.09
Right-Weight = 1 AND Left-Distance = 4 AND Left-Weight = 4 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 5 AND Left-Weight = 2 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Left-Weight = 4 -> L   P=1.0 R=0.08
Right-Weight = 1 AND Left-Weight = 5 AND Right-Distance = 2 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 4 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Weight = 5 AND Right-Distance = 4 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Right-Distance = 1 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 5 AND Left-Weight = 3 -> L   P=1.0 R=0.01
Right-Distance = 1 AND Left-Distance = 5 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 4 AND Left-Weight = 2 -> L   P=1.0 R=0.01
Right-Distance = 1 AND Left-Weight = 3 AND Left-Distance = 2 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Weight = 5 AND Right-Distance = 3 -> L   P=1.0 R=0.02
Left-Distance = 5 AND Left-Weight = 5 AND Right-Weight = 4 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Left-Distance = 5 AND Left-Weight = 2 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 5 AND Left-Weight = 4 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Right-Weight = 1 AND Left-Distance = 2 -> L   P=1.0 R=0.02
Left-Weight = 5 AND Left-Distance = 4 AND Right-Distance = 3 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Left-Distance = 4 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 3 AND Left-Weight = 4 -> L   P=1.0 R=0.02
Left-Distance = 5 AND Left-Weight = 4 AND Right-Distance = 3 -> L   P=1.0 R=0.02
Left-Weight = 5 AND Left-Distance = 4 AND Right-Distance = 2 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Weight = 3 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Right-Weight = 1 AND Left-Distance = 5 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Distance = 5 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Left-Distance = 4 AND Left-Weight = 2 -> L   P=1.0 R=0.02
Left-Weight = 5 AND Left-Distance = 4 AND Right-Distance = 4 -> L   P=1.0 R=0.02
Right-Distance = 2 AND Left-Distance = 5 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 4 AND Left-Weight = 5 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Weight = 5 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 4 AND Right-Distance = 1 -> L   P=1.0 R=0.02
Right-Distance = 2 AND Left-Distance = 5 AND Left-Weight = 5 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Weight = 3 AND Left-Distance = 2 -> L   P=1.0 R=0.01
Right-Weight = 2 AND Left-Weight = 4 AND Left-Distance = 4 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Left-Distance = 3 AND Left-Weight = 2 -> L   P=1.0 R=0.02
Right-Distance = 2 AND Left-Weight = 4 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Right-Distance = 2 AND Left-Weight = 4 -> L   P=1.0 R=0.02
Left-Distance = 5 AND Left-Weight = 4 AND Right-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Distance = 5 AND Left-Weight = 4 -> L   P=1.0 R=0.02
Left-Weight = 5 AND Left-Distance = 5 AND Right-Distance = 3 -> L   P=1.0 R=0.01
Right-Weight = 1 AND Left-Distance = 4 AND Right-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Right-Distance = 1 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Right-Distance = 2 AND Left-Distance = 4 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Left-Weight = 5 AND Right-Weight = 1 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Left-Weight = 5 AND Right-Distance = 2 AND Left-Distance = 3 -> L   P=1.0 R=0.02
```

```
Right-Weight = 2 AND Left-Distance = 5 AND Right-Distance = 2 -> L   P=1.0 R=0.01
Right-Weight = 1 AND Left-Distance = 5 AND Right-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Weight = 5 AND Right-Distance = 2 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Left-Weight = 3 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Right-Distance = 1 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Right-Distance = 1 AND Left-Distance = 5 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Weight = 3 AND Left-Distance = 4 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Right-Distance = 2 AND Left-Distance = 4 -> L   P=1.0 R=0.02
Left-Weight = 4 AND Left-Distance = 4 AND Right-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Weight = 4 AND Left-Distance = 3 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Right-Distance = 2 AND Left-Distance = 5 -> L   P=1.0 R=0.02
Left-Weight = 5 AND Right-Distance = 3 AND Right-Weight = 3 -> L   P=1.0 R=0.01
Right-Distance = 2 AND Left-Distance = 4 AND Left-Weight = 4 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Distance = 3 AND Left-Weight = 2 -> L   P=1.0 R=0.01
Right-Weight = 2 AND Right-Distance = 1 AND Left-Distance = 4 -> L   P=1.0 R=0.02
Left-Distance = 5 AND Right-Weight = 4 AND Left-Weight = 4 -> L   P=1.0 R=0.01
Right-Weight = 2 AND Left-Weight = 3 AND Right-Distance = 1 -> L   P=1.0 R=0.02
Right-Weight = 1 AND Left-Weight = 4 AND Left-Distance = 2 -> L   P=1.0 R=0.01
Right-Distance = 2 AND Left-Weight = 3 AND Right-Weight = 3 -> L   P=1.0 R=0.01
Left-Weight = 5 AND Left-Distance = 4 AND Right-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Right-Distance = 2 AND Left-Distance = 3 AND Left-Weight = 3 -> L   P=1.0 R=0.0
Left-Distance = 5 AND Right-Weight = 4 AND Right-Distance = 1 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Distance = 5 AND Right-Distance = 4 -> L   P=1.0 R=0.01
Right-Weight = 1 AND Left-Weight = 5 AND Left-Distance = 2 -> L   P=1.0 R=0.02
Right-Distance = 2 AND Left-Distance = 5 AND Left-Weight = 4 -> L   P=1.0 R=0.01
Right-Weight = 2 AND Left-Weight = 5 AND Right-Distance = 3 -> L   P=1.0 R=0.02
Right-Distance = 2 AND Right-Weight = 1 AND Left-Distance = 3 -> L   P=1.0 R=0.01
Right-Weight = 2 AND Right-Distance = 2 AND Left-Distance = 2 AND Left-Weight = 3 -> L   P=1.0 R=0.0
Right-Weight = 2 AND Right-Distance = 2 AND Left-Weight = 2 AND Left-Distance = 3 -> L   P=1.0 R=0.0
Left-Weight = 5 AND Left-Distance = 5 AND Right-Distance = 4 -> L   P=1.0 R=0.01
Right-Distance = 1 AND Right-Weight = 1 AND Left-Weight = 2 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Distance = 2 AND Left-Weight = 4 AND Right-Distance = 3 -> L   P=1.0 R=0.0
Right-Distance = 2 AND Left-Weight = 5 AND Left-Distance = 2 AND Right-Weight = 3 -> L   P=1.0 R=0.0
Right-Weight = 2 AND Left-Weight = 3 AND Left-Distance = 3 AND Right-Distance = 3 -> L   P=1.0 R=0.0
Right-Distance = 2 AND Left-Distance = 4 AND Right-Weight = 3 -> L   P=1.0 R=0.01
Left-Distance = 5 AND Right-Distance = 3 AND Left-Weight = 3 -> L   P=1.0 R=0.02
Right-Distance = 1 AND Left-Distance = 4 AND Right-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Distance = 5 AND Left-Weight = 2 AND Right-Distance = 3 -> L   P=1.0 R=0.0
Right-Weight = 1 AND Right-Distance = 3 AND Left-Distance = 2 -> L   P=1.0 R=0.01
Right-Distance = 2 AND Left-Weight = 3 AND Right-Weight = 1 -> L   P=1.0 R=0.01
Right-Weight = 2 AND Left-Weight = 5 AND Left-Distance = 2 AND Right-Distance = 4 -> L   P=1.0 R=0.0
Right-Distance = 2 AND Left-Distance = 5 AND Right-Weight = 4 -> L   P=1.0 R=0.01
Left-Weight = 4 AND Right-Weight = 1 AND Right-Distance = 3 -> L   P=1.0 R=0.01
Right-Distance = 1 AND Left-Distance = 2 AND Left-Weight = 2 AND Right-Weight = 3 -> L   P=1.0 R=0.0
Right-Weight = 2 AND Right-Distance = 1 AND Left-Distance = 2 AND Left-Weight = 2 -> L   P=1.0 R=0.0
Right-Distance = 2 AND Right-Weight = 2 AND Left-Distance = 2 AND Left-Weight = 4 -> L   P=1.0 R=0.0
Left-Weight = 5 AND Right-Weight = 4 AND Right-Distance = 3 -> L   P=1.0 R=0.01
Right-Distance = 2 AND Left-Weight = 5 AND Left-Distance = 2 AND Right-Weight = 4 -> L   P=1.0 R=0.0
Left-Distance = 3 AND Left-Weight = 3 AND Right-Distance = 2 AND Right-Weight = 4 -> L   P=1.0 R=0.0
Left-Distance = 4 AND Left-Weight = 4 AND Right-Weight = 3 -> L   P=1.0 R=0.02
Right-Weight = 2 AND Left-Distance = 4 AND Right-Distance = 2 AND Left-Weight = 2 -> L   P=1.0 R=0.0
Left-Distance = 3 AND Left-Weight = 4 AND Right-Distance = 3 AND Right-Weight = 3 -> L   P=1.0 R=0.0
Left-Weight = 3 AND Right-Weight = 2 AND Left-Distance = 3 AND Right-Distance = 4 -> L   P=1.0 R=0.0
Left-Distance = 1 AND Right-Weight = 5 -> R   P=1.0 R=0.08
Left-Weight = 1 AND Right-Distance = 4 -> R   P=1.0 R=0.07
Right-Distance = 5 AND Left-Distance = 1 AND Right-Weight = 3 -> R   P=1.0 R=0.02
Right-Distance = 5 AND Right-Weight = 4 AND Left-Distance = 2 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Distance = 5 AND Left-Distance = 3 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Weight = 4 AND Left-Weight = 2 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Weight = 5 AND Right-Distance = 2 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Distance = 5 AND Right-Weight = 2 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Left-Distance = 1 AND Right-Distance = 3 -> R   P=1.0 R=0.02
Right-Distance = 5 AND Right-Weight = 5 AND Left-Distance = 4 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Weight = 4 AND Left-Weight = 3 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Distance = 5 AND Left-Distance = 4 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Left-Distance = 1 AND Right-Distance = 2 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Right-Distance = 5 AND Left-Distance = 3 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Weight = 5 AND Right-Distance = 3 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Weight = 4 AND Left-Weight = 4 -> R   P=1.0 R=0.01
Right-Distance = 5 AND Right-Weight = 4 AND Left-Distance = 3 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Distance = 5 AND Left-Distance = 2 -> R   P=1.0 R=0.01
Left-Distance = 1 AND Right-Distance = 5 AND Left-Weight = 2 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Right-Distance = 5 AND Left-Distance = 2 -> R   P=1.0 R=0.01
Left-Distance = 1 AND Right-Distance = 4 AND Left-Weight = 2 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Weight = 3 AND Right-Distance = 3 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Right-Distance = 4 AND Left-Distance = 3 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Left-Weight = 1 AND Right-Distance = 5 -> R   P=1.0 R=0.01
Left-Distance = 1 AND Right-Weight = 4 AND Left-Weight = 1 -> R   P=1.0 R=0.01
Left-Weight = 2 AND Right-Weight = 5 AND Right-Distance = 3 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Distance = 4 AND Left-Weight = 3 -> R   P=1.0 R=0.01
Left-Weight = 1 AND Right-Weight = 5 AND Left-Distance = 3 -> R   P=1.0 R=0.02
Right-Distance = 5 AND Right-Weight = 4 AND Left-Distance = 4 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Weight = 4 AND Left-Distance = 3 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Distance = 5 AND Left-Weight = 3 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Distance = 4 AND Left-Distance = 3 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Weight = 4 AND Right-Distance = 5 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Right-Distance = 5 AND Left-Weight = 2 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Weight = 3 AND Right-Distance = 2 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Right-Distance = 4 AND Left-Distance = 2 -> R   P=1.0 R=0.01
Left-Weight = 1 AND Right-Distance = 5 AND Left-Distance = 2 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Weight = 5 AND Left-Distance = 2 -> R   P=1.0 R=0.02
```

```
Left-Weight = 1 AND Right-Weight = 3 AND Right-Distance = 2 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Weight = 3 AND Right-Distance = 5 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Distance = 4 AND Right-Weight = 3 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Right-Distance = 5 AND Left-Weight = 1 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Distance = 4 AND Right-Weight = 4 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Distance = 3 AND Left-Weight = 3 -> R   P=1.0 R=0.01
Right-Weight = 5 AND Right-Distance = 4 AND Left-Distance = 4 -> R   P=1.0 R=0.01
Left-Weight = 1 AND Left-Distance = 1 AND Right-Weight = 3 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Distance = 3 AND Right-Weight = 3 -> R   P=1.0 R=0.01
Right-Distance = 5 AND Right-Weight = 5 AND Left-Weight = 3 -> R   P=1.0 R=0.01
Left-Weight = 1 AND Right-Weight = 4 AND Left-Distance = 2 -> R   P=1.0 R=0.01
Left-Distance = 2 AND Right-Distance = 5 AND Left-Weight = 4 -> R   P=1.0 R=0.02
Right-Weight = 4 AND Right-Distance = 4 AND Left-Weight = 3 -> R   P=1.0 R=0.02
Left-Weight = 1 AND Right-Distance = 3 AND Right-Weight = 2 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Right-Distance = 3 AND Left-Distance = 2 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Left-Distance = 2 AND Right-Distance = 2 -> R   P=1.0 R=0.01
Left-Distance = 1 AND Right-Weight = 4 AND Right-Distance = 2 -> R   P=1.0 R=0.01
Right-Distance = 5 AND Left-Distance = 1 AND Left-Weight = 4 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Distance = 4 AND Right-Weight = 3 -> R   P=1.0 R=0.02
Right-Distance = 5 AND Left-Weight = 1 AND Right-Weight = 2 -> R   P=1.0 R=0.02
Left-Distance = 1 AND Right-Weight = 2 AND Left-Weight = 1 -> R   P=1.0 R=0.01
Right-Weight = 5 AND Left-Weight = 2 AND Right-Distance = 4 -> R   P=1.0 R=0.02
Right-Distance = 5 AND Right-Weight = 5 AND Left-Weight = 4 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Left-Distance = 1 AND Right-Distance = 3 -> R   P=1.0 R=0.01
Right-Weight = 4 AND Right-Distance = 4 AND Left-Distance = 2 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Distance = 5 AND Left-Distance = 2 -> R   P=1.0 R=0.02
Right-Weight = 5 AND Left-Weight = 1 AND Left-Distance = 4 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Left-Distance = 1 AND Right-Weight = 3 -> R   P=1.0 R=0.02
Right-Weight = 4 AND Left-Weight = 1 AND Right-Distance = 3 -> R   P=1.0 R=0.01
Right-Distance = 5 AND Right-Weight = 3 AND Left-Distance = 2 -> R   P=1.0 R=0.01
Left-Weight = 2 AND Right-Weight = 5 AND Right-Distance = 2 AND Left-Distance = 3 -> R   P=1.0 R=0.0
Right-Weight = 4 AND Left-Distance = 1 AND Right-Distance = 4 -> R   P=1.0 R=0.01
Left-Weight = 2 AND Right-Distance = 3 AND Right-Weight = 4 -> R   P=1.0 R=0.01
Right-Distance = 5 AND Left-Weight = 3 AND Left-Distance = 4 -> R   P=1.0 R=0.01
Right-Weight = 5 AND Left-Weight = 3 AND Right-Distance = 3 -> R   P=1.0 R=0.01
Left-Distance = 2 AND Right-Weight = 3 AND Left-Weight = 1 -> R   P=1.0 R=0.01
Right-Distance = 5 AND Left-Distance = 3 AND Left-Weight = 3 AND Right-Weight = 3 -> R   P=1.0 R=0.0
Left-Distance = 2 AND Right-Weight = 3 AND Right-Distance = 4 -> R   P=1.0 R=0.0
Left-Distance = 1 AND Right-Weight = 2 AND Right-Distance = 3 -> R   P=1.0 R=0.01
Right-Weight = 5 AND Right-Distance = 4 AND Left-Weight = 3 -> R   P=1.0 R=0.02
Left-Weight = 2 AND Right-Distance = 5 AND Right-Weight = 2 AND Left-Distance = 3 -> R   P=1.0 R=0.0
Left-Distance = 2 AND Right-Distance = 3 AND Right-Weight = 3 AND Left-Weight = 3 -> R   P=1.0 R=0.0
Right-Weight = 4 AND Left-Distance = 3 AND Right-Distance = 4 -> R   P=1.0 R=0.02
Right-Distance = 2 AND Left-Weight = 1 AND Right-Weight = 4 -> R   P=1.0 R=0.01
Left-Weight = 2 AND Left-Distance = 2 AND Right-Weight = 2 AND Right-Distance = 3 -> R   P=1.0 R=0.0
Right-Weight = 5 AND Right-Distance = 2 AND Left-Distance = 2 AND Left-Weight = 4 -> R   P=1.0 R=0.0
Left-Distance = 1 AND Right-Weight = 2 AND Right-Distance = 4 -> R   P=1.0 R=0.01
Left-Weight = 2 AND Left-Distance = 1 AND Right-Distance = 2 AND Right-Weight = 2 -> R   P=1.0 R=0.0
Left-Distance = 3 AND Right-Distance = 5 AND Left-Weight = 3 AND Right-Weight = 2 -> R   P=1.0 R=0.0
Right-Weight = 5 AND Left-Weight = 2 AND Left-Distance = 4 AND Right-Distance = 2 -> R   P=1.0 R=0.0
Left-Distance = 3 AND Right-Weight = 4 AND Left-Weight = 3 AND Right-Distance = 3 -> R   P=1.0 R=0.0
Left-Distance = 2 AND Left-Weight = 1 AND Right-Distance = 2 AND Right-Weight = 2 -> R   P=1.0 R=0.0
Left-Distance = 3 AND Right-Weight = 5 AND Left-Weight = 3 AND Right-Distance = 2 -> R   P=1.0 R=0.0
Left-Weight = 2 AND Right-Distance = 4 AND Left-Distance = 2 AND Right-Weight = 2 -> R   P=1.0 R=0.0
Right-Distance = 5 AND Left-Weight = 2 AND Left-Distance = 4 AND Right-Weight = 2 -> R   P=1.0 R=0.0
Left-Distance = 3 AND Left-Weight = 1 AND Right-Distance = 2 AND Right-Weight = 2 -> R   P=1.0 R=0.0
Right-Weight = 4 AND Left-Distance = 2 AND Left-Weight = 3 AND Right-Distance = 2 -> R   P=1.0 R=0.0
Left-Distance = 3 AND Right-Distance = 5 AND Right-Weight = 3 AND Left-Weight = 4 -> R   P=1.0 R=0.0
Left-Distance = 1 AND Left-Weight = 3 AND Right-Distance = 2 AND Right-Weight = 2 -> R   P=1.0 R=0.0
Right-Distance = 3 AND Right-Weight = 5 AND Left-Distance = 3 AND Left-Weight = 4 -> R   P=1.0 R=0.0
Right-Weight = 4 AND Left-Weight = 2 AND Left-Distance = 3 AND Right-Distance = 2 -> R   P=1.0 R=0.0
Left-Distance = 2 AND Right-Weight = 3 AND Left-Weight = 4 AND Right-Distance = 3 -> R   P=1.0 R=0.0
```

7.2 Example rules - Experiment 3

Rules

```
stlmt = t -> nowin   P=1.0 R=0.03
hdchk = t -> nowin   P=1.0 R=0.01
spcop = t -> nowin   P=1.0 R=0.0
skach = t AND bkblk = f -> nowin   P=1.0 R=0.01
wkna8 = t AND dwipd = g -> nowin   P=1.0 R=0.03
mulch = t AND bknwy = f -> nowin   P=1.0 R=0.05
wkna8 = t AND rxmsq = t -> nowin   P=1.0 R=0.02
wkna8 = t AND thrsk = t -> nowin   P=1.0 R=0.0
wkna8 = t AND rimmx = f AND wkovl = t -> nowin   P=1.0 R=0.09
bxqsq = t AND rimmx = f -> nowin   P=1.0 R=0.49
skrxp = t AND katri = b -> nowin   P=1.0 R=0.01
skrxp = t AND rimmx = f AND wknck = t -> nowin   P=1.0 R=0.09
wknck = t AND rimmx = f AND bkxcr = t -> nowin   P=1.0 R=0.26
rxmsq = t AND bkxwp = t -> nowin   P=1.0 R=0.01
rxmsq = t AND wknck = t AND bxqsq = f -> nowin   P=1.0 R=0.01
mulch = t AND dsopp = t -> nowin   P=1.0 R=0.01
katri = b AND bkblk = f AND bkxbq = f -> nowin   P=1.0 R=0.05
```

```
mulch = t AND reskd = t -> nowin   P=1.0 R=0.0
wknck = t AND rimmx = f AND blxwp = t -> nowin   P=1.0 R=0.24
mulch = t AND rimmx = f AND katri = n -> nowin   P=1.0 R=0.08
rxmsq = t AND wkpos = f AND bkxbq = f AND r2ar8 = t -> nowin   P=1.0 R=0.03
wkna8 = t AND rimmx = f AND r2ar8 = t -> nowin   P=1.0 R=0.08
rxmsq = t AND qxmsq = f AND wtoeg = t -> nowin   P=1.0 R=0.0
wknck = t AND rimmx = f AND bkona = t -> nowin   P=1.0 R=0.06
rxmsq = t AND qxmsq = f AND bkxbq = f AND dsopp = f -> nowin   P=1.0 R=0.03
wkpos = f AND bkon8 = t AND bkblk = t -> nowin   P=1.0 R=0.01
wkpos = f AND qxmsq = t AND r2ar8 = t -> nowin   P=1.0 R=0.0
wkpos = f AND rxmsq = t AND bkxbq = f AND reskr = t -> nowin   P=1.0 R=0.02
wkpos = f AND katri = n AND bkxbq = f AND reskd = t -> nowin   P=1.0 R=0.0
wkpos = f AND katri = n AND bkona = t AND bkblk = f AND bknwy = f -> nowin   P=1.0 R=0.01
wknck = t AND rimmx = f AND reskr = t -> nowin   P=1.0 R=0.07
wknck = t AND rimmx = f AND thrsk = t -> nowin   P=1.0 R=0.03
wkpos = f AND dwipd = g AND thrsk = t -> nowin   P=1.0 R=0.0
wkpos = f AND katri = n AND bkxbq = f AND bkxwp = t AND bkblk = f -> nowin   P=1.0 R=0.03
wknck = t AND rimmx = f AND bkon8 = t -> nowin   P=1.0 R=0.03
wknck = t AND rimmx = f AND bkblk = t -> nowin   P=1.0 R=0.05
wknck = t AND rimmx = f AND wkovl = t AND bknwy = f -> nowin   P=1.0 R=0.33
wkpos = f AND dwipd = g AND bkblk = f AND bkxbq = f -> nowin   P=1.0 R=0.04
wkna8 = t AND bkblk = t AND simpl = t -> nowin   P=1.0 R=0.02
wkpos = f AND rxmsq = t AND bkxbq = f AND simpl = t -> nowin   P=1.0 R=0.0
wkpos = f AND katri = n AND bkxbq = f AND bkxcr = t AND bkspr = t -> nowin   P=1.0 R=0.03
wkpos = f AND katri = n AND bkxbq = f AND wkcti = f AND r2ar8 = f -> nowin   P=1.0 R=0.02
wkpos = f AND katri = n AND bkxbq = f AND wknck = f AND bkblk = f AND r2ar8 = t -> nowin   P=1.0 R=0.07
thrsk = t AND dsopp = t AND dwipd = g -> nowin   P=1.0 R=0.0
wkpos = f AND bkspr = t AND blxwp = f AND wkcti = t AND rimmx = f AND bkblk = f -> nowin   P=1.0 R=0.01
reskr = t AND katri = n AND wkovl = f AND blxwp = f AND bkxbq = f AND rimmx = f -> nowin   P=1.0 R=0.02
rimmx = t -> won   P=1.0 R=0.36
reskd = t AND katri = w -> won   P=1.0 R=0.01
katri = w AND qxmsq = t -> won   P=1.0 R=0.0
katri = w AND thrsk = t -> won   P=1.0 R=0.01
katri = w AND bxqsq = f AND bknwy = t -> won   P=1.0 R=0.02
katri = w AND bxqsq = f AND cntxt = t AND wknck = f -> won   P=1.0 R=0.17
wknck = f AND skrxp = t -> won   P=1.0 R=0.01
wknck = f AND bxqsq = f AND bkxbq = t -> won   P=1.0 R=0.4
qxmsq = t AND wkovl = f -> won   P=1.0 R=0.0
qxmsq = t AND rkxwp = t AND wknck = f -> won   P=1.0 R=0.01
thrsk = t AND wknck = f AND skewr = f -> won   P=1.0 R=0.0
qxmsq = t AND wkcti = t AND rxmsq = t -> won   P=1.0 R=0.01
thrsk = t AND bkblk = t AND wkpos = t -> won   P=1.0 R=0.01
bxqsq = f AND qxmsq = t AND dsopp = t -> won   P=1.0 R=0.01
bxqsq = f AND qxmsq = t AND wtoeg = t -> won   P=1.0 R=0.0
bxqsq = f AND wknck = f AND wtoeg = t AND dwipd = l -> won   P=1.0 R=0.08
bxqsq = f AND wknck = f AND skewr = f AND rxmsq = f -> won   P=1.0 R=0.18
reskd = t AND wkpos = t -> won   P=1.0 R=0.01
bkblk = t AND skach = t -> won   P=1.0 R=0.0
bkblk = t AND thrsk = t AND reskr = t -> won   P=1.0 R=0.0
bkblk = t AND wkovl = t AND wkpos = t AND bkxcr = t -> won   P=1.0 R=0.02
bxqsq = f AND qxmsq = t AND wknck = f AND katri = n -> won   P=1.0 R=0.02
bxqsq = f AND wknck = f AND wkpos = t AND r2ar8 = f -> won   P=1.0 R=0.17
bkblk = t AND bkxcr = f AND dsopp = t -> won   P=1.0 R=0.0
bkblk = t AND bkxcr = f AND reskr = t AND bknwy = f -> won   P=1.0 R=0.01
bkblk = t AND thrsk = t AND wkcti = t -> won   P=1.0 R=0.0
bkblk = t AND bknwy = t AND r2ar8 = f AND cntxt = t -> won   P=1.0 R=0.0
bxqsq = f AND wknck = f AND wkpos = t AND reskr = t -> won   P=1.0 R=0.05
bxqsq = f AND wknck = f AND wkpos = t AND bkona = t -> won   P=1.0 R=0.06
bxqsq = f AND wknck = f AND wkpos = t AND wkcti = t AND rxmsq = f -> won   P=1.0 R=0.08
reskd = t AND bknwy = f AND bkona = f AND bkxbq = t -> won   P=1.0 R=0.01
bxqsq = f AND wknck = f AND wtoeg = t AND bkspr = t -> won   P=1.0 R=0.05
bxqsq = f AND wknck = f AND wkpos = t AND katri = n AND bkxcr = t -> won   P=1.0 R=0.15
bkblk = t AND bkxcr = f AND bkspr = t AND cntxt = t -> won   P=1.0 R=0.02
bxqsq = f AND bknwy = t AND mulch = f AND wkovl = t -> won   P=1.0 R=0.0
bxqsq = f AND wknck = f AND wkpos = t AND dsopp = t AND simpl = f -> won   P=1.0 R=0.04
bxqsq = f AND wknck = f AND wtoeg = t AND rxmsq = f AND dsopp = f -> won   P=1.0 R=0.11
bxqsq = f AND bkblk = t AND wkovl = t AND wkpos = t AND rxmsq = f -> won   P=1.0 R=0.04
bxqsq = f AND bkxcr = f AND wkovl = f AND dsopp = t AND bkxbq = t -> won   P=1.0 R=0.01
bxqsq = f AND bkxcr = f AND wkovl = f AND dsopp = t AND reskr = f AND skrxp = f AND simpl = f -> won   P=1.0 R=0.01
bxqsq = f AND bkxcr = f AND wkovl = f AND bkona = f AND bkxbq = t AND bkspr = t -> won   P=1.0 R=0.02
thrsk = t AND wknck = f AND wkpos = t AND dsopp = f -> won   P=1.0 R=0.02
bxqsq = f AND bkblk = t AND wkovl = t AND cntxt = t AND dwipd = g -> won   P=1.0 R=0.02
bxqsq = f AND bknwy = t AND mulch = f AND dwipd = g -> won   P=1.0 R=0.01
bxqsq = f AND wknck = f AND wkpos = t AND katri = n AND bkspr = t -> won   P=1.0 R=0.16
r2ar8 = f AND wkcti = t AND reskr = f AND simpl = f AND bknwy = t -> won   P=1.0 R=0.0
r2ar8 = f AND wkcti = t AND reskr = f AND simpl = f AND wknck = f -> won   P=1.0 R=0.02
bxqsq = f AND r2ar8 = f AND bkspr = f AND katri = b AND simpl = f -> won   P=1.0 R=0.0
bxqsq = f AND r2ar8 = f AND bkspr = f AND wkpos = t AND skrxp = f AND simpl = f AND wtoeg = n -> won   P=1.0 R=0.05
bxqsq = f AND bkxcr = f AND wkovl = f AND bkona = f AND bkxbq = t AND mulch = f AND bkon8 = f AND blxwp = f -> won   P=1.0 R=0.08
bxqsq = f AND wknck = f AND wkpos = t AND katri = n AND rxmsq = f AND dsopp = f -> won   P=1.0 R=0.37
wkcti = t AND reskr = f AND dsopp = t AND r2ar8 = f AND skach = f -> won   P=1.0 R=0.0
bkblk = t AND thrsk = t AND cntxt = t AND spcop = f -> won   P=1.0 R=0.01
wkcti = t AND reskr = f AND r2ar8 = f AND simpl = f AND bkspr = f AND bkxcr = f -> won   P=1.0 R=0.02
bkblk = t AND wkovl = t AND cntxt = t AND bkxbq = f AND katri = n -> won   P=1.0 R=0.02
r2ar8 = f AND bkxcr = f AND bkxbq = f AND wknck = t AND bknwy = f AND skrxp = f AND skewr = f AND simpl = f -> won   P=1.0 R=0.01
r2ar8 = f AND wkcti = t AND reskr = f AND bkxbq = f AND bknwy = f AND bkspr = f AND skach = f AND bkxcr = f AND reskd = f -> won   P=1.0 R=0.01
r2ar8 = f AND wtoeg = t AND bkspr = f AND skewr = t AND bkxbq = f AND thrsk = f -> won   P=1.0 R=0.01
katri = b AND bkblk = t AND wkovl = t AND cntxt = t AND wkpos = f AND simpl = f -> won   P=1.0 R=0.0
r2ar8 = f AND skewr = f AND simpl = t AND bkxbq = f AND wtoeg = n AND bkxcr = f AND mulch = f -> won   P=1.0 R=0.01
```

```
107  r2ar8 = f AND wtoeg = t AND bkspr = f AND simpl = t AND bkxbq = f AND blxwp = f AND thrsk = f AND bknwy = f AND skrxp = f -> won   P=1.0 R=0.01
```

References

[1] R. S. Siegler. *Balance Scale dataset*. 1976. URL: https://archive.ics.uci.edu/ml/datasets/Balance+Scale.

[2] Barbara Hayes-Roth. *The Hayes-Roth Database*. 1977. URL: http://archive.ics.uci.edu/ml/datasets/Hayes-Roth.

[3] Alen Shapiro. *Chess (King-Rook vs. King-Pawn) Data Set*. 1989. URL: https://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King-Pawn%29.

[4] Jadzia Cendrowska. "PRISM: An algorithm for inducing modular rules". In: *International Journal of Man-Machine Studies* 33.6 (1990), pp. 699–727.

[5] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2019. URL: http://archive.ics.uci.edu/ml.

[6] Python Software Foundation. *Python Release Python 3.10.0*. 2021. URL: https://www.python.org/downloads/release/python-3100/.