SUPERVISED AND EXPERIENTIAL LEARNING

RULES: A Simple Rule Extraction System

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Ramon Mateo Navarro

Master in Artificial Intelligence, UPC-URV-UB

Spring semester, Course 2021/2022

Index

Datasets

Mushroom Data Set

Objective and attributes

The purpose of this dataset is to predict if mushrooms are edible or poisonous.
This dataset is formed by 22 different categorical attributes and 8124 instances.

Preprocess

The preprocess step in this dataset is made for simply the analysis of the data when the algorithm generates rules to make it easy to debug the code and see what is happening inside code.

The CSV file contains the information using only letters. The first letter indicates if the mushroom is edible or poisonous; that is the attribute that we want to try to predict. : **p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u** . The rest of the letters are the different attributes that mushroom has.

The preprocess step changes letters by their respective names. The names of what represent each letter for each attribute can be found in agaricus-lepiota.names file inside the Data folder.

To make this step. I generate 23 dictionaries that contain for each letter the word that represents. For each attribute the preprocess function replaces the letter to their respective word. In this step we have one exception and it's that the attributes edible and poisonous are placed by 0 and 1 to make the prediction easier.

This is an example of the final result:

1,convex,smooth,brown,bruises,pungent,free,close,narrow,black,enlarging,equal,smooth,smooth,white,white,partial,white,one,pendant,black,scattered,urban

The dataset has similar instances for edible and poisonous. The following plot shows the distribution.

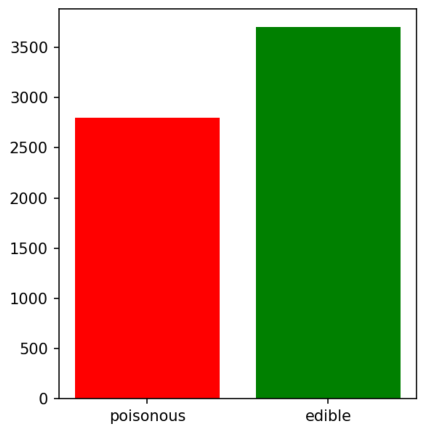


Figure 1 Distribution of poisonous and edible

Finally the data is split by hand in two different files. One for training and another for tests. The train set contains 6500 instances and the test contains 1624 instances.

Algerian Forest Fires Dataset Data Set

Objective and attributes

The dataset includes 244 instances that regroup data of two regions of Algeria,namely the Bejaia region located in the northeast of Algeria and the Sidi Bel-abbes region located in the northwest of Algeria.

Note: For simplicity these two regions are unified only in one region.

The objective of this dataset is to try to predict if fire or not fire according to the different attributes in the region.

Preprocess

In this dataset the preprocess only changes “***fire***” and “***not fire***” for 0 and 1 respectively.

The data distribution between **fire** and **not fire** can see in the following figure

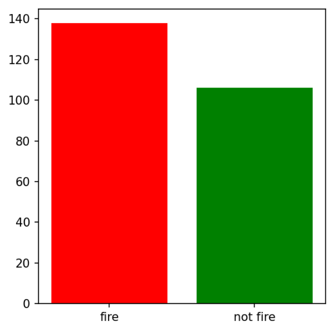


Figure 2: Distribution of fire and not fire

Balance Scale Data Set

Objective and attributes

The dataset includes 625 instances. This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance.

Preprocess

In this dataset the preprocess only changes “***L***”, “**B**” and “**R”** for 0 ,1 and 2 respectively.

The data distribution can be seen on the following figure.

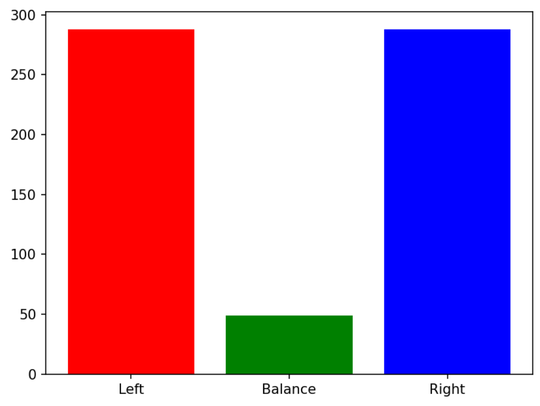


Figure 3: Distribution of Left, Balance and Right

In this case the data is not evenly distributed. It’s possible to see how the Balance class has a lot of fewer instances than the other two classes.

Algorithm

Pseudocode

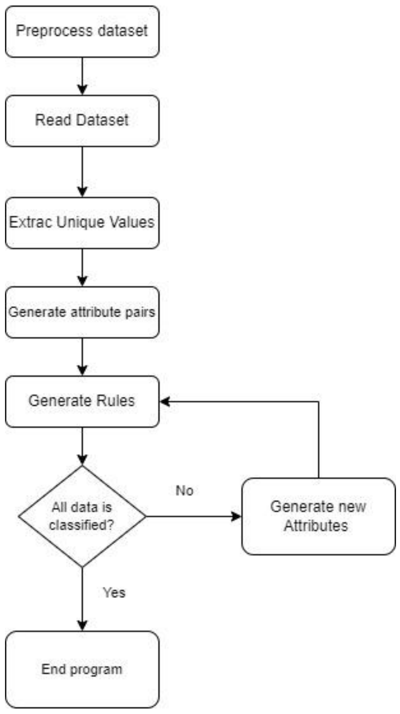
The pseudo code algorithm is this:

```
get ∀ columns unique Attributes
generate pair of attributes of different columns
while data_pending_to_classify or rule to evaluate:
      get rule
      apply rule to all rows pending to classify.
      if rule applied to rows belong a single class
            delete rows classified
            add rule
if not all data is classified
      generate new pair of rules with four attributes
      go to while.
```

As long as there is a rule to evaluate and data, the loop continues. Inside the loop, the code takes a rule that at this point has not been evaluated. With this rule, the code evaluates for all remaining rows if it can be applied to any row.

The only exception is that the rule can only be classified in one class. If the rule covers two or more different classes (poisonous and edible), the rule is discarded.

Flow diagram



This is the flow of my code. The first step when you choose the dataset is the preprocessing step. The last instruccion of the preprocess saves the data for avoid to re execute in future executions this step.

The following steps are to read the dataset preprocessed, extract all unique values for each column and generate a pair of attributes. With these pairs of attributes the code generate rules.

Once this steps are complete the algorithm starts and tries to classify all instances using rules.

If all rules are visited and its remaining data for classifying the code generates new rules but this time with four attributes at each rule. At the end the program plots the results.

Results

Rules are saved in three json files following this example of format:

```
"1": [[0, 1, ["flat", "grooves"]], [0, 1, ["bell", "grooves"]], [0, 1, ["conical ", "scaly"]]...
```

The first attribute in each rule indicates the columns ***"i"*** and ***"j"*** where code needs to find flat and grooves. If attributes are found in column 0 and 1 then the prediction is 1.

Note: *In the following results to avoid large prints only appear the first five rules. The value is in percentage. The length of data indicates the large test dataset.*

Note 2: *In the discussion the comparison between results are explained*

Mushroom Dataset

The results for Mushroom dataset are the following.

```
====== Start classification ======
====== All data classified ======
====== End classification ======
Rule: 0 Coverage: 0.015386982612709647
Rule: 1 Coverage: 0.015386982612709647
Rule: 2 Coverage: 0.40006154793045084
Rule: 3 Coverage: 0.030773965225419295
Rule: 4 Coverage: 0.015386982612709647
Max coverage: 12.586551777196492
For class: 1 the total rules are: 34
For class: 0 the total rules are: 51
Len of data is 1625
Total correct predictions: 1201
Accuracy: 0.7390769230769231
```

The results show how the best rule only covers 12% of total data. The number of total rules that we need is 85 and the total instances are 6499. So the mean of coverage is 1.18% for each rule.

The following figure show how data is classified during the algorithm

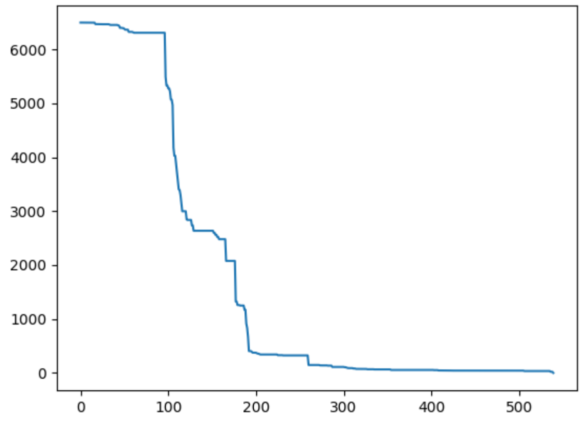


Figure 4 . Evolution of instances remaining to classify

In figure 4 it is possible to see how the data is classified. Each descent in the frame represents that a visited rule can classify the data into a single class.

In this case it's possible to classify all data using this algorithm with good accuracy. According to the results with this algorithm the results show 73% of accuracy.

Fire Dataset

The results for the Fire dataset are the following.

```
====== Start classification ======
====== All data classified ======
====== End classification ======
Rule: 0 Coverage: 0.5128205128205128
Rule: 1 Coverage: 1.0256410256410255
Rule: 2 Coverage: 1.0256410256410255
Rule: 3 Coverage: 0.5128205128205128
Rule: 4 Coverage: 1.0256410256410255
```

```
Max coverage: 1.0256410256410255
For class: 0 the total rules are: 84
For class: 1 the total rules are: 66
Len of data is 49
Total correct predictions: 30
Accuracy: 0.6122448979591837
```

The results show how the best rule only covers 1% of total data. So we need 150 rules to classify 195 instances. That means that mean of coverage of rules is 0.66%

The following figure show how data is classified during the algorithm

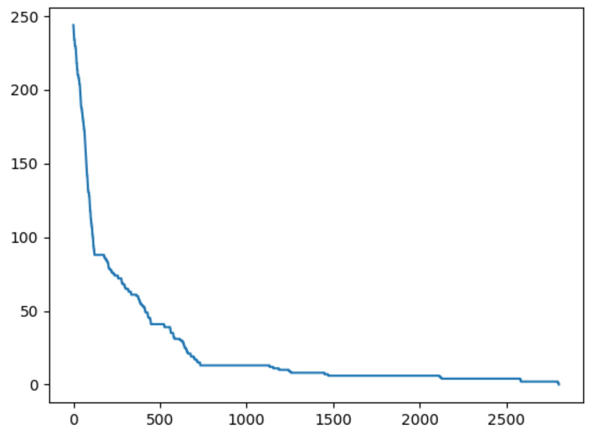


Figure 5 . Evolution of instances remaining to classify

In this plot it's possible to see how the first rules can classify a lot of instances. The algorithm requires a lot of iterations to find new rules that could classify new rows after the first rules.

Balance dataset

The results for the Balance dataset are the following.

```
====== Start classification ======
====== End classification ======
====== Classifying the data remaining ======
====== All data classified ======
====== End data remaining classification ======
Rule: 0 Coverage: 0.2
Rule: 1 Coverage: 0.2
Rule: 2 Coverage: 0.2
Rule: 3 Coverage: 0.2
Rule: 4 Coverage: 0.2
Max coverage: 0.2
For class: 2 the total rules are: 217
For class: 0 the total rules are: 242
For class: 1 the total rules are: 41
Len of data is 125
Total correct predictions: 0
Accuracy: 0.0
```

In this case the results show that all rules cover the same percentage of data. The length of the training dataset is 500 instances and the classifier generates 500 rules. That means for each instance we need a new rule so on testing it's impossible to predict anything because the code doesn't have the rule to predict. In the following plot we can see that.

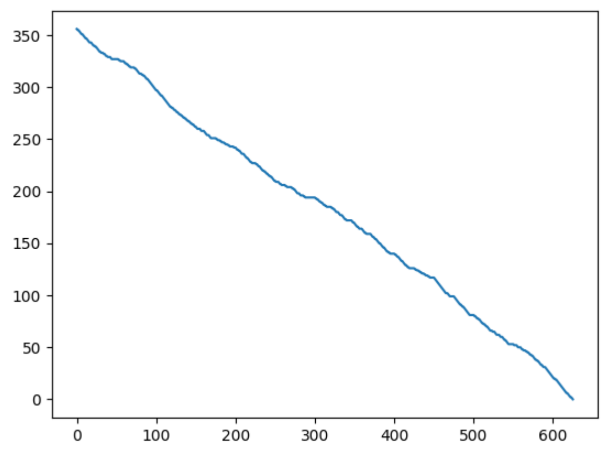


Figure 6. Evolution of instances remaining to classify

The plot shows a brief continuous line. And we can’t see that some rule could classify one or more instances.

Discussion and conclusion

According to the results we can see how the best dataset where this algorithm can be applied is where attributes are all categorical variables or have a lot of categorical attributes. This is because the algorithm generates rules using a pair of attributes so if the dataset contains numerical values it can generate more combinations than categorical variables and it's more difficult that for each rule group a lot of rows. This can be checked in the Balance dataset that only has four columns of attributes (and the column of class) but these four are numeric. The results show how it can't classify more than one row for each rule. And on the other hand we have a mushroom dataset that is fully categorical. In this case the rules can correctly classify a lot of rows.

One form to fix this issue that could be studied in the future work it's in the preprocessing step, adding one step more that consists in detecting what is the range of the values for each column. Once we have it we can group and reduce the number of unique attributes. Similar to group by bins.

Evaluating the precision in this algorithm it's not necessary because the precision is always 1. The reason is that rules are only added when rules can only classify one class.

Bibliography

“Algerian Forest Fires Dataset Data Set.” *UCI Machine Learning Repository*, 22 October 2019, https://archive.ics.uci.edu/ml/datasets/Algerian+Forest+Fires+Dataset++. Accessed 21 April 2022.

“Balance Scale Data Set.” *UCI Machine Learning Repository*, https://archive.ics.uci.edu/ml/datasets/balance+scale. Accessed 21 April 2022.

“Mushroom Data Set.” *UCI Machine Learning Repository*, 27 April 1987, https://archive.ics.uci.edu/ml/datasets/mushroom. Accessed 21 April 2022.