Supervised and experiential learning

Practical Work 1

Valeriu Vicol

Barcelona, April 14, 2022

Contents

1 Introduction **2**

2 Rise classifier **2**

2.1 Structure 2

2.2 Implementation 3

2.3 Improvements 4

3 Datasets **5**

3.1 Data processing 5

3.2 Wine 5

3.3 Breast cancer 6

3.4 Obesity 7

4 Conclusion **9**

1 Introduction

The purpose of this practical work is to gain practical knowledge rule induction algorithms, in case of this work is the RISE algorithm [1], which represent a way of unifying the instance-based and rule-based induction algorithms. Bellow we can find in depth descriptions of how current implementation is structure and why it is done in this way.

2 Rise classifier

Main concept of RISE is to create a set of rule from the instance and then try to extend the boundaries/constraints of the rules in order to increase the precision of rules-set, the process is continuous until the precision is not increasing anymore.

2.1 Structure

In Figure 1 we can see the class structure of the classifier, algorithm was implemented mainly using object-oriented programming [3] style instead of keeping everything in array and there are few reasons for that. First reason is that is easier to read for humans, second is that if we would keep everything in the array it would be hard to maintain the code and to extend it and also with object-oriented programming we can improve the performance of the code of the algorithm:

- `RiseClassifier`: represents the core class of algorithms which contains the classifier implementation with some data processing functions
- `RiseUtils`: represents the utils class of algorithms which is computing distance between instance and rules, this logic was moved to another class instead of `RiseClassifier` as we can experiment with improvements and distances without touching the core logic of classifier.
- `Instance`: represent the instance from the dataset in OOP format
- `GenericInstanceAttribute`: represent an abstract class which define a property on the instance, we look at this class like an contract definition which will be used across the program.
  - `NumericalInstanceAttribute`: represent the concrete implementation, with some logic related to numerical attributes of the instance.
  - `CategoricalInstanceAttribute`: represent the concrete implementation, with some logic related to categorical attributes of the instance, which are ordinal encoded beforehand.
- `InstanceRule`: represent the rule inducted from instance class is similar to `Instance` except it has more utils method which are computing the distance and extending rule
- `GenericAttributeRule`: represent abstract class which define a concrete rule for concrete attribute
  - `NumericalInstanceAttribute`: represent the concrete implementation, with some logic related to numerical attributes of the instance.

– `CategoricalInstanceAttribute`: represent the concrete implementation, with some logic related to categorical attributes of the instance, which are ordinal encoded beforehand.

There are few more classes which are not included in schema as they are not related to the core algorithm but to data processing.

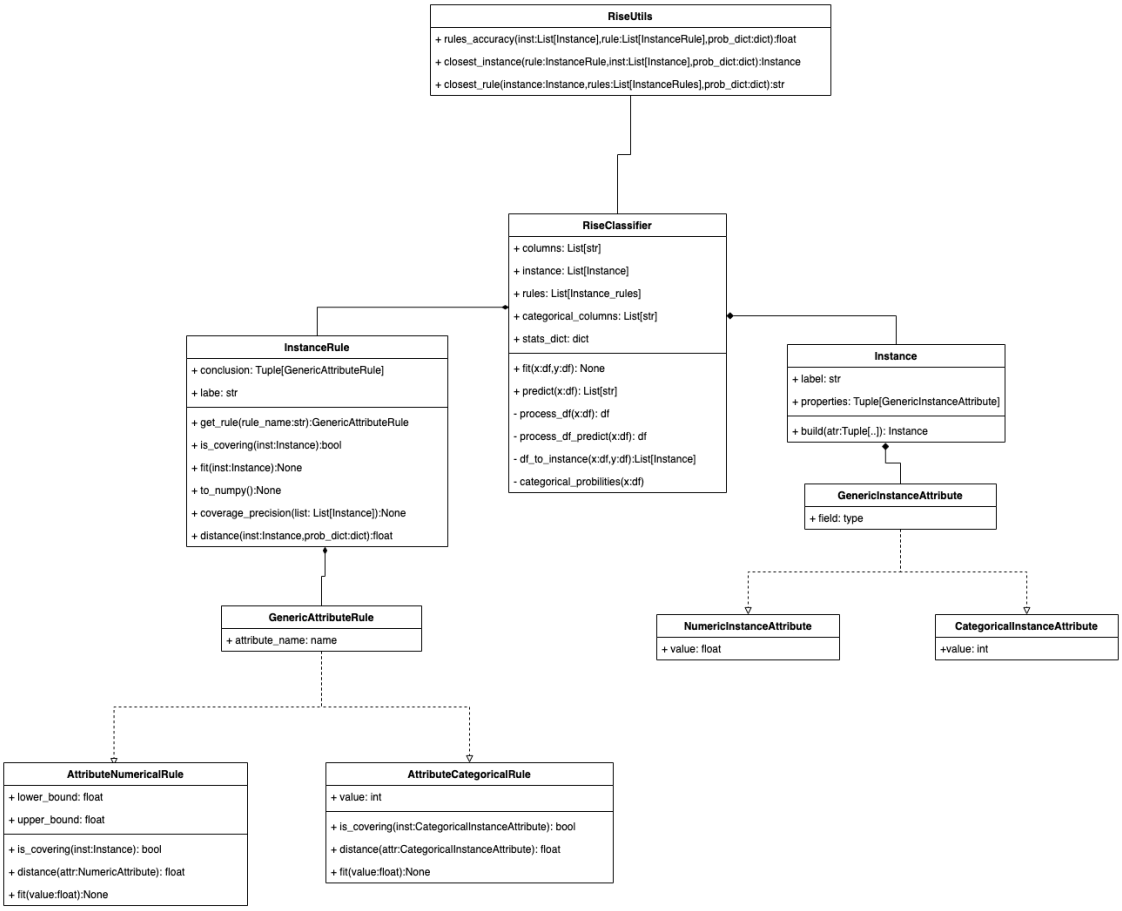

Figure 1: Rise classifier class structure

2.2 Implementation

In current section we will discuss implementation of the algorithm, in Algorithm 1 we can see the pseudocode implementation of the algorithm, there are few minor changes to original implementation, but the idea is similar, in this implementation we use more explicit description of the steps. On extra step added is index validation when we remove the rule from the rule set, as we perform removing of the rule by index we have to take into account this details. Also were implemented some minor improvements to the original implementation which are discussed in the next section. Main motivation for changing the algorithm is speed, one of the problems is that we compute many times same distance which is taking time.

input : dataframe with the instances and dataframe with the labels
output: a set of rules inferred from the dataframe

normalization with min-max scaller, and ordinal encoding for categorical values;
$instances \leftarrow$ **df_to_instances**$(dataframe, labels)$;
$stat\_dict \leftarrow$ **categorical_probabilites**$(dataframe, labels)$;
$rules \leftarrow set([\text{InstanceRule}(it) \text{ for } it \text{ in } instances])$;
$rules' \leftarrow rules$;
$final\_precision \leftarrow$ **rules_precision**$(rule, instances, stat\_dict)$;
while *True* **:**
    $intial\_precision \leftarrow final\_precision$;
    foreach $index, rule \in$ ***enum****(rules)* **do**
        if $len(rules) < index$ **then**
            break # because during this loop we update *rules* by removing some rules we have to add this validation
        end
        $closest\_instance \leftarrow$ **closest_instance**$(rule, instances, stat\_dict)$;
        $rule' \leftarrow rule$;
        $rule'$.fit$(closest_instance)$ # most specific generalization;
        $rules'[index] \leftarrow rule'$;
        if ***rules_precision****$(instances, rules', ..) \geq$* ***rules_precision****$(instances, rules, ..)$* **then**
            if $rule' \in rules$ **then**
                $rules' \leftarrow rules' - rule'$;
            end
            $rule \leftarrow rules'$;
        end
        else
            $rules'[index] \leftarrow rule$;
        end
    end
    $final\_preicision \leftarrow$**rules_precision**$(instances, rules, stats\_dict)$;
    if $intial\_precision \geq final\_precion$ **then**
        break;
    end
end

Algorithm 1: Rise classifier

2.3 Improvements

One major difference between original algorithm and current implementation is caching the distance between the rule and the instance, as we perform many times calculations for finding the closest instance to the rule. In current implementation each rule have a dict with the hash of the instance and distance to it, like this when we have a rule have to calculate the distance between the instance it tries tot find it in the dict with hashes if is not present there it performs the calculations and then save distance and hash of instance in the dict. This small extra step decrease the time of execution significantly.

3 Datasets

For testing the algorithm were picked 3 different datasets with different size, algorithms have categorical and numerical attributes as for first dataset was picked wine because it is relatively small, and it has only numerical attributes, similar thing is with second choice as it also mainly contains the numerical attributes and one categorical. And the third dataset is obesity which contains plenty of numerical attributes and categorical.

3.1 Data processing

As the preprocessing step were used only 2 steps, normalization of numerical attributes using MinMaxScaller and encoding categorical ones using OrdinalEncoder. We don't have any missing attributes in datasets, so we don't have to add additional step of filling in missing attributes. Then using StratifiedKFold from sklearn library [2] dataset was split in 3 folds, so we can train algorithm more than one time on the dataset so see how it is behaves.

3.2 Wine

In this section we analyze the results of the algorithm on wine dataset in the Table 1 we can see the results of algorithm on the train set, we can clearly see that algorithm has very good results as all the metrics are more than 99%, yes algorithm is overfitting, but we think that is inevitable for all rule induction algorithms to overfit the train instances. Next step is to evaluate algorithm on the test set in Table 2 we can observe the results, and again we can clearly say that algorithm has good results, even though in fold 2 and 3, results decreased, but all metrics are around 90% which is a good result.

metric	**value**
balanced accuracy	99.29 %
accuracy	99.15 %
f1 score	99.15 %
precision	99.17 %
recall	99.15 %

(a) fold 1

metric	**value**
balanced accuracy	100%
accuracy	100%
f1 score	100%
precision	100%
recall	100%

(b) fold 2

metric	**value**
balanced accuracy	100%
accuracy	100%
f1 score	100%
precision	100%
recall	100%

(c) fold 3

Table 1: Metrics of the rise classifier on the train set

metric	**value**
balanced accuracy	94.44 %
accuracy	93.33 %
f1 score	93.21 %
precision	94.01 %
recall	93.33 %

(a) fold 1

metric	**value**
balanced accuracy	89.8%
accuracy	89.83%
f1 score	89.86%
precision	91.09%
recall	89.83%

(b) fold 2

metric	**value**
balanced accuracy	88.52%
accuracy	89.83%
f1 score	89.74%
precision	91.04%
recall	89.83%

(c) fold 3

Table 2: Metrics of the rise classifier on the test set

Bellow are first 2 and last 2 rules inducted by the rise classifier from the dataset. As we can see each rule is covering small amount of instances around 1.6% which is 2-3 instance which, but the precision of are 100% this results can be interpreted as each rule knows, few instances and is able to classify them. We can observe this in inferred boundaries of the rule bellow. One more thing is to mention here the last rules are covering only one instance as we can see this rule didn't even extend the attributes, the reason why this happens is topic of investigation.

- class[3] - coverage[1.681%] - precision[100.0%] – ['(Alcohol in [0.404,0.602])', '(Malic.acid in [0.504,0.549])', '(Ash in [0.32,0.392])', '(Acl in [0.48,0.48])', '(Mg in [0.196,0.196])', '(Phenols in [0.172,0.221])', '(Flavanoids in [0.03,0.068])', '(Nonflavanoid.phenols in [0.5,0.846])', '(Proanth in [0.148,0.177])', '(Color.int in [0.422,0.858])', '(Hue in [0.247,0.34])', '(OD in [0.17,0.196])', '(Proline in [0.215,0.29])']
- class[2] - coverage[1.681%] - precision[100.0%] – ['(Alcohol in [0.16,0.16])', '(Malic.acid in [0.004,0.516])', '(Ash in [0.196,0.196])', '(Acl in [0.451,0.451])', '(Mg in [0.174,0.185])', '(Phenols in [0.352,0.497])', '(Flavanoids in [0.274,0.405])', '(Nonflavanoid.phenols in [0.308,0.442])', '(Proanth in [0.322,0.461])', '(Color.int in [0.0,0.117])', '(Hue in [0.464,0.928])', '(OD in [0.649,0.675])', '(Proline in [0.0,0.204])']
- ... ...
- class[3] - coverage[0.847%] - precision[100.0%] – ['(Alcohol in [0.923,0.923])', '(Malic.acid in [0.755,0.755])', '(Ash in [0.885,0.885])', '(Acl in [0.716,0.716])', '(Mg in [0.321,0.321])', '(Phenols in [0.33,0.33])', '(Flavanoids in [0.088,0.088])', '(Nonflavanoid.phenols in [0.811,0.811])', '(Proanth in [0.369,0.369])', '(Color.int in [0.663,0.663])', '(Hue in [0.106,0.106])', '(OD in [0.125,0.125])', '(Proline in [0.215,0.215])']
- class[1] - coverage[0.847%] - precision[100.0%] – ['(Alcohol in [0.902,0.902])', '(Malic.acid in [0.2,0.2])', '(Ash in [0.59,0.59])', '(Acl in [0.278,0.278])', '(Mg in [0.691,0.691])', '(Phenols in [0.678,0.678])', '(Flavanoids in [0.823,0.823])', '(Nonflavanoid.phenols in [0.208,0.208])', '(Proanth in [0.663,0.663])', '(Color.int in [0.347,0.347])', '(Hue in [0.496,0.496])', '(OD in [0.921,0.921])', '(Proline in [0.39,0.39])']

3.3 Breast cancer

For the breast cancer dataset we can see similar picture as we had in for the wine dataset, but however in the case of breast cancer algorithm is able to generalize better as we can see in Table 3 and in Table 4. Even though the metrics for train are not 100% as in wine dataset for the train but in the test we can clearly see the difference for fold 2 & 3 we can higher results comparing to the wine dataset. In file `/out/cancer.txt` we have classification report we can see that we have recall for class Malignant around 96%, which is a good result, but there is space for improvement.

metric	value
balanced accuracy	99.51 %
accuracy	99.36 %
f1 score	99.36 %
precision	99.37 %
recall	99.36 %

(a) fold 1

metric	value
balanced accuracy	99.53%
accuracy	99.57%
f1 score	99.57%
precision	99.57%
recall	99.57%

(b) fold 2

metric	value
balanced accuracy	99.67%
accuracy	99.57%
f1 score	99.57%
precision	99.58%
recall	99.57%

(c) fold 3

Table 3: Metrics of the rise classifier on the train set

metric	value
balanced accuracy	94.29%
accuracy	94.85%
f1 score	94.85%
precision	94.85%
recall	94.85 %

(a) fold 1

metric	value
balanced accuracy	96.46%
accuracy	96.14%
f1 score	96.16%
precision	96.29%
recall	96.14 %

(b) fold 2

metric	value
balanced accuracy	95.64%
accuracy	96.57%
f1 score	96.55%
precision	96.59%
ine recall	96.57%

(c) fold 3

Table 4: Metrics of the rise classifier on the test set

Bellow we can see the inferred rule from the dataset one major difference is coverage comparing to wine dataset, but this difference is more related to size of the dataset. Also, we can see that second rule coverage is 6.87% and has 100% which is impressive if we compare with the last rules which have about 0.2% coverage.

- class[Malignant] - coverage[0.429%] - precision[100.0%] – ['(Clump Thickness in [1.0,1.0])', '(Uniformity of Cell Size in [0.444,0.778])', '(Uniformity of Cell Shape in [0.667,0.778])', '(Marginal Adhesion in [0.333,0.333])', '(Single Epithelial Cell Size in [0.333,1.0])', '(Bare Nuclei == 1.0)', '(Bland Chromatin in [0.778,0.778])', '(Normal Nucleoli in [0.0,0.889])', '(Mitoses in [0.0,0.0])']
- class[Benign] - coverage[6.867%] - precision[100.0%] – ['(Clump Thickness in [0.111,0.444])', '(Uniformity of Cell Size in [0.0,0.0])', '(Uniformity of Cell Shape in [0.0,0.0])', '(Marginal Adhesion in [0.0,0.0])', '(Single Epithelial Cell Size in [0.111,0.111])', '(Bare Nuclei == 0.0)', '(Bland Chromatin in [0.111,0.111])', '(Normal Nucleoli in [0.0,0.0])', '(Mitoses in [0.0,0.0])']
- ... ...
- class[Malignant] - coverage[0.215%] - precision[100.0%] – ['(Clump Thickness in [1.0,1.0])', '(Uniformity of Cell Size in [0.222,0.222])', '(Uniformity of Cell Shape in [0.556,0.556])', '(Marginal Adhesion in [0.111,0.111])', '(Single Epithelial Cell Size in [0.222,0.222])', '(Bare Nuclei == 5.0)', '(Bland Chromatin in [0.333,0.333])', '(Normal Nucleoli in [1.0,1.0])', '(Mitoses in [0.111,0.111])']
- class[Malignant] - coverage[0.215%] - precision[100.0%] – ['(Clump Thickness in [0.778,0.778])', '(Uniformity of Cell Size in [1.0,1.0])', '(Uniformity of Cell Shape in [1.0,1.0])', '(Marginal Adhesion in [1.0,1.0])', '(Single Epithelial Cell Size in [0.444,0.444])', '(Bare Nuclei == 1.0)', '(Bland Chromatin in [0.778,0.778])', '(Normal Nucleoli in [1.0,1.0])', '(Mitoses in [0.556,0.556])']

3.4 Obesity

And the last data set is obesity data set, as we can see in Table 5 and 6 results don't differ too much from the previous data sets, one major difference is that the metrics decreased in the test set to 76% - 81% which leave some space for improvement, but still is good results. In terms of execution time data set took high amount of time, which again lead us to previous idea that we have to improve the performance of algorithm. Also, we have to say that the data set has plenty of categorical and numerical attributes comparing with the previous 2 datasets, and also there are 7 categories for classification and in classification report `out/obesity.txt` we can see that algorithm has good precision and recall for all classes.

metric	value
balanced accuracy	99.29 %
accuracy	99.29 %
f1 score	99.29 %
precision	99.3 %
recall	99.29 %

(a) fold 1

metric	value
balanced accuracy	99.21%
accuracy	99.22%
f1 score	99.22%
precision	99.22%
recall	99.22%

(b) fold 2

metric	value
balanced accuracy	99.22%
accuracy	99.25%
f1 score	99.25%
precision	99.25%
recall	99.22%

(c) fold 3

Table 5: Metrics of the rise classifier on the train set

metric	value
balanced accuracy	81.42%
accuracy	81.68%
f1 score	82.04%
precision	84.19%
recall	81.68%

(a) fold 1

metric	value
balanced accuracy	76.59%
accuracy	76.85%
f1 score	76.88%
precision	78.06%
recall	76.85%

(b) fold 2

metric	value
balanced accuracy	78.32%
accuracy	78.51%
f1 score	79.03%
precision	80.02%
recall	79.2%

(c) fold 3

Table 6: Metrics of the rise classifier on the test set

Bellow we can see inferred rules of the classifier on the train set with the coverage and precision as we can see that coverage of the rules is very small main reason of this is the size of the data set and variety of the classes in dataset.

- class[Obesity_Type_II] - coverage[0.142%] - precision[100.0%] – ['(Gender == 1.0)', '(Age in [0.256,0.376])', '(Height in [0.648,0.742])', '(Weight in [0.56,0.62])', '(family_history_with_overweight == 1.0)', '(FAVC == 1.0)', '(FCVC in [0.5,0.597])', '(NCP in [0.667,0.667])', '(CAEC == 2.0)', '(SMOKE == 0.0)', '(CH2O in [0.492,0.545])', '(SCC == 0.0)', '(FAF in [0.264,0.454])', '(TUE in [0.0,0.429])', '(CALC == 2.0)', '(MTRANS == 3.0)']
- class[Obesity_Type_II] - coverage[0.142%] - precision[100.0%] – ['(Gender == 1.0)', '(Age in [0.248,0.376])', '(Height in [0.626,0.742])', '(Weight in [0.551,0.62])', '(family_history_with_overweight == 1.0)', '(FAVC == 1.0)', '(FCVC in [0.145,0.5])', '(NCP in [0.667,0.667])', '(CAEC == 2.0)', '(SMOKE == 0.0)', '(CH2O in [0.492,0.504])', '(SCC == 0.0)', '(FAF in [0.264,0.459])', '(TUE in [0.0,0.001])', '(CALC == 2.0)', '(MTRANS == 3.0)']
- ... ...
- class[Obesity_Type_III] - coverage[0.071%] - precision[100.0%] – ['(Gender == 0.0)', '(Age in [0.268,0.268])', '(Height in [0.36,0.36])', '(Weight in [0.491,0.491])', '(family_history_with_overweight == 1.0)', '(FAVC == 1.0)', '(FCVC in [1.0,1.0])', '(NCP in [0.667,0.667])', '(CAEC == 2.0)', '(SMOKE == 0.0)', '(CH2O in [0.778,0.778])', '(SCC == 0.0)', '(FAF in [0.006,0.006])', '(TUE in [0.291,0.291])', '(CALC == 2.0)', '(MTRANS == 3.0)']
- class[Obesity_Type_III] - coverage[0.071%] - precision[100.0%] – ['(Gender == 0.0)', '(Age in [0.268,0.268])', '(Height in [0.329,0.329])', '(Weight in [0.508,0.508])', '(family_history_with_overweight == 1.0)', '(FAVC == 1.0)', '(FCVC in [1.0,1.0])', '(NCP in [0.667,0.667])', '(CAEC == 2.0)', '(SMOKE == 0.0)', '(CH2O in [0.839,0.839])', '(SCC == 0.0)', '(FAF in [0.0,0.0])', '(TUE in [0.254,0.254])', '(CALC == 2.0)', '(MTRANS == 3.0)']

4 Conclusion

As part of this practical work we implemented the rise classification algorithm which represent a way of unifying the instance-based learning and rule induction algorithms. As we saw in previous sections we can achieve great results with rise, but major problem in my opinion is the speed of algorithm because the training time is taking great amount of time for big datasets. For further work should be investigated any ways of improving the speed of the algorithm like avoid computing more than one time distance between rule and instance, this is the main pain point in main opinion, because this is done when we want to extend a rule or get the precision of rule-set which is costly. Work around with hashing values of instance decreased the execution for wine set from 400 seconds to 12 seconds, but however the amount of time for big datasets is high. For more in-depth investigation of the results we can check the output files in the `output` directory where we can see complete image of output of the algorithm.

References

[1] Pedro Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168, 1996.

[2] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[3] Tim Rentsch. Object oriented programming. *ACM Sigplan Notices*, 17(9):51–57, 1982.