SUPERVISED AND EXPERIENTIAL LEARNING

MASTER IN ARTIFICIAL INTELLIGENCE

Universitat Politècnica de Catalunya

PRACTICAL WORK 1:

IMPLEMENTATION AND EVALUATION OF THE RISE RULE-BASED CLASSIFIER

Albert Espín

March 20th, 2019

Table Of Contents

1. Introduction

The Rule Induction from a Set of Exemplars classifier (RISE) was introduced by Pedro Domingos from University of California in 1996 [1]. The most innovative feature of this algorithm is that it combines rule induction with instance-based learning, in the attempt of building a more accurate classifier than previous rule-based approaches.

The objective of this work is the implementation of the RISE classifier in Python 3, and the evaluation of the algorithm in the classification task, applying it to 4 data sets of different sizes and features. The accuracy of each trained model is computed, as well as the coverage and local accuracy of each rule.

2. The RISE algorithm

The RISE classifier algorithm has been implemented following Domingos' original description [1], with some minor changes. The pseudo-code is depicted in Algorithm 1.

The RISE classifier receives as input a set of training instances, represented as a vector of features, along with a class value. The objective of the algorithm is to obtain a set of rules based on the instances that will hopefully generalize good enough to classify new instances with the correct class label. As a first step, and in a way that resembles instance-based learning techniques, the RISE classifier builds an initial set of rules based on each instance. Specifically, each attribute of an instance becomes a condition in the rule, where the name of the feature is stated to be equal to the instance's value for that feature. These features can be categorical or numerical. In the latter case, later in the algorithm, the conditions can be represented as a value being within a range, lesser or equal than an upper bound and not lower than a lower bound. In the initial rules though, both bounds are equal, a circumstance that makes the condition degenerate, according to Domingos.

The accuracy of the first rule set is then computed, as the proportion of correctly labeled instances. To classify each instance, it is compared with all rules to find the nearest one (excluding the rule created from the instance, so analogous to it): if the class value of the instances matches the rule's conclusion, the classification is counted as correct, otherwise it is considered incorrect. For each instance, its nearest rule distance is computed along with whether its classification has been correct or not, which will later allow to update the accuracy in a very efficient way if only a single rule is modified, without redoing all the calculations.

The distance between a rule and a distance is computed as the sum of distances between each condition of the rule and the associated attribute of the instance. For numeric attributes, the distance is 0 if the value of the instance lies within the condition's range, otherwise it is the distance to the exceeded bound, normalized dividing by the range length (the difference between the upper and the lower bound). For categorical attributes, a simplified value difference metric is used, computing the correlation of each category (the one in the instance's attribute and in the rule's condition) to produce every possible class. To make this calculation faster, a dictionary of three levels is calculated once and reused later. The dictionary maps each categorical attribute to a sub-dictionary with all the categories mapped to an inner dictionary where each class (given the category) is assigned a probability, based on the times that the class appears by the total times that the category appears in the training instances.

The main loop of the RISE algorithm consists in updating the rules as long as the accuracy of the rule set improves. In particular, each rule is analyzed in an inner loop. For a given rule, the nearest instance that is not yet covered by the rule's conditions is found. The coverage test is very simple: a rule covers an instance if all the conditions of the rule are respected by the instance, i.e. the instance has numeric attributes within the ranges of the associated conditions and the same category as stated in categorical conditions. A modified version of the rule is then obtained, expanding the numeric ranges to an extent that allows the instance to respect them, and removing non-matching categorical conditions. This adapted rule is a generalization of the rule to meet the specific coverage requirements of the instance. An alternative rule set that would replace the original rule with its adapted version is only accepted if it produces an accuracy that is not below the previous one. This alternative accuracy can be efficiently computed by considering the previously stored list of nearest distances and classification results for each instance, to see if the new rule has a lower distance to the instance and therefore should be used to compute its classification, in which case a copy of the former accuracy is increased (by a proportion of one by the number of instances) if a previously bad classified example is now correctly labeled, or decremented in the same proportion if a former correct classification is replaced by a wrong one. If the alternative accuracy is not below the previous accuracy, the rule set adds the adapted version of the rule and removes its original version. At the end of each iteration of the main loop, the accuracy updated during the iteration is compared with the previous one. If it has improved, the algorithm continues to update the rules, otherwise the rule set is considered final.

After the rule set has been determined, the coverage of each rule is computed, measured as the proportion of instances that respect the rule's conditions, as previously explained. The accuracy of each rule is also computed, considered as the proportion of covered instances for which the rule is correct, i.e. the instance's class is the same as the one mentioned in the rule's conclusion. The set of rules annotated with the coverage and accuracy of each rule is returned as the classifier's output.

Algorithm 1: RISE classifier

```
Algorithm 1: RISE classifier
   Input  : instances /*List of training instances, with features and class*/
   Output: rule_set /*Classifier rules*/
1  /*Create a rule based on each instance, where there is a condition per instance feature with the
    form feature_name == feature_value and the consequent is class_name == class_value*/
2  $rule\_set \leftarrow \{create\_rule(instance)$ for $instance$ in $instances\}$;
3  /*The accuracy is the proportion of correctly labeled instances, assigning to each one its nearest
    rule, ignoring the instance's rule (nearest-rule distances are stored for faster later updates)*/
4  $initial\_accuracy \leftarrow get\_accuracy(rule\_set, instances, ignore\_instance\_rule = True)$;
5  $accuracy \leftarrow initial\_accuracy$;
6  $has\_accuracy\_increased \leftarrow True$;
7  /*Update the rules as long as the accuracy increases*/
8  while $has\_accuracy\_increased$ do
9      /*Try to generalize each rule*/
10     for $rule$ in $rule\_set$ do
11         /*Choose the most similar instance that does not respect all the conditions of the rule*/
12         $nearest\_instance \leftarrow get\_nearest\_non\_covered\_instance(rule, instances)$
13         /*Generalize the rule to cover the instance: remove categorical mismatching conditions
            and expand the lower/upper bounds of numeric ones, so that it is true that
            lower_bound $\leq value \leq upper\_bound * /$
14         $adapted\_rule \leftarrow get\_most\_specific\_generalization(rule, instance)$;
15         /*Check if any instance has the modified rule closer than any other, to compute its
            classification correctness and obtain an alternative accuracy*/
16         $alternative\_accuracy \leftarrow get\_updated\_accuracy(accuracy, instances, adapted\_rule)$;
17         /*Accept the generalized rule if the accuracy does not decrease*/
18         if $alternative\_accuracy \geq accuracy$ then
19             $accuracy \leftarrow alternative\_accuracy$;
20             $rule\_set \leftarrow rule\_set + adapted\_rule - rule$;
21     /*Check if the accuracy has improved during the current iteration*/
        $has\_accuracy\_improved \leftarrow accuracy > initial\_accuracy$;
22     $initial\_accuracy \leftarrow accuracy$;
23 /*For each rule, calculate the coverage (proportion of instances that respect its conditions) and
    accuracy (proportion of the covered instances whose class matches the rule's conclusion)*/
    $rule\_set \leftarrow \{(rule,\ get\_coverage\_and\_accuracy(rule,\ instances))$ for $rule$ in $rule\_set\}$;
```

The rules obtained from the classifier algorithm can be applied to new instances, e.g. a test set. The classification procedure is the same as used before to obtain the accuracy of the set of rules during the model training, with the only difference that no rule is ignored in this case, since the rules to be examined have been obtained from a separate set of instances. In fact, the implemented function to get the accuracy allows to retrieve the class label assigned to each instance, which is done for the test ones, along with the accuracy of the whole model when applied to the test set.

3. Experimental methodology

The RISE algorithm has been tested with 4 data sets to evaluate its performance. To enrich the experimental variability, these data sets include different proportions of numeric and categorical features, balanced and unbalanced class distributions, different number of classes, and some of the data sets contain missing values.

The contact lenses data set is a tiny, purely categorical data set where the requirement of lenses (none, soft or hard) should be predicted from vision-related features of the individuals. The labor data set is a small data set combining numeric and categorical features that describe contract elements and the output is binary: whether the contract should be considered as acceptable or not. Despite of its small size, it is challenging due to the high percentage of missing values (35.7%). The hepatits data sets is larger, and also combines categorical and numeric features related with hepatitis patients to determine whether they will live or die. Finally, the breast cancer data set is also two-class and with both numeric and categorical classes, but is larger than the previous ones and has the special characteristic of having inconsistent instances, i.e. examples with same features but different class.

As a pre-processing task, the missing values of the data sets have been replaced with the mean of the feature in the case of numeric attributes, and the mode for categorical ones. To ensure that all numeric attributes have the same relevance when computing rule-to-instance distances, the ranges of numeric features have been linearly normalized between 0 and 1: the minimum value has become 0, the maximum 1, and the rest lie in between. For distance calculation, there are two parameters for the RISE algorithm, that have been given the parameters that Domingos found to be empirically most suitable. The first one was named "q", with a value of 1: it is the exponent to apply each class probability difference in categorical simplified value difference calculation. The second one is "s", with a value of 2: it is the exponent to apply to each attribute-based distance in the calculation of the distance between a rule and an instance.

For each data set, K-fold cross-validation was used to obtain different training and validation splits, to build the classifier and assess its accuracy, respectively. The average of the accuracy obtained by the different model was computed, along with the standard deviation. It is important to note that the splits were imposed to be stratified: both training and validation sets maintain the proportion of classes of the original data set, to the possible extent. Since the contact lenses data set was very small, 5-fold cross-validation was used, since more folds would imply losing the stratification characteristic. For the other data sets, larger options were possible, and 10-fold cross-validation was used, as in Domingos' work, to be able to compare the results with the original paper in a more reliable way.

4. Results and discussion

4.1. Rule set interpretation

The program outputs the rules built by each classifier using the training set, along with their coverage and accuracy. The test instances are displayed with the real and predicted class label.

4.1.1. Contact lenses

Table 1 shows the rules produced with one of the training folds of Contact lenses (the third one), that obtained 100% correctness when the test instances were examined, represented in Table 2. As it would be expected, shorter, less restrictive rules have more coverage, while less examples respect rules with greater number of conditions. For example, the first rule states that a patient that has astigmatism and normal tear production rate should not need contact lenses. However, in some other cases lenses are predicted as necessary, such as with the third rule, in which the combination of astigmatism and normal tear production rate makes the classifier recommend hard lenses, or soft ones, for the same tear rate but no astigmatism, according to the fifth rule. One can see that the proportion of instances that require no contact lenses is considerably higher than those than need one, which is reflected in the rules: there are more attribute combinations leading to no contact lenses (4) than those that lead to soft (1) or hard (2) ones. In each fold, the stratification of the rules allows to have 2-3 test instances of class none and generally 1 of soft and hard, to respect the proportion of the entire data set. In the third fold, however, no instance of the hard class is chosen, due to the very small number of test instances.

Table 1: Rule set for Contact lenses (fold 3), with coverage and accuracy

Rule	**Cov.**	**Acc.**
(astigmatism == yes) && (tear-prod-rate == normal) => (contact-lenses == none)	0.3	0.333
(spectacle-prescrip == hypermetrope) && (astigmatism == yes) && (tear-prod-rate == normal) => (contact-lenses == hard)	0.15	0.333
(astigmatism == yes) && (tear-prod-rate == normal) => (contact-lenses == hard)	0.3	0.667
(astigmatism == no) && (tear-prod-rate == normal) => (contact-lenses == none)	0.25	0.2
(astigmatism == no) && (tear-prod-rate == normal) => (contact-lenses == soft)	0.25	0.8
(astigmatism == yes) && (tear-prod-rate == reduced) => (contact-lenses == none)	0.3	1.0
(spectacle-prescrip == hypermetrope) && (tear-prod-rate == reduced) => (contact-lenses == none)	0.3	1.0

Table 2: Test examples for Contact lenses (fold 3), with real and predicted classes

Age	**Spectacle prescript.**	**Astigmatism**	**Tear prod. rate**	**Class (real)**	**Class (predicted)**
Young	Myope	No	Reduced	None	None
Pre-presbyop.	Myope	No	Reduced	None	None
Pre-presbyop.	Hypermetrope	No	Normal	Soft	Soft
Pre-presbyop.	Hypermetrope	No	Reduced	None	None

4.1.2. Labor

All the folds of the Labor data set produce more than 30 rules. Some of the rules of the seventh fold are shown in Table 3. For a complete list of the 38 rules, refer to the Labor files in the "Results" folder of this project. Due to the high number of attributes of the data set (16), the test instances with the predicted labels are not shown in a table, but they can be checked in the mentioned results files. Due to the lower quotient between instance number and attribute number, examples have unique attribute combinations. Therefore, it can be observed that most of the numerous rules only apply to a single training instance (0.059 of coverage, and full accuracy for it), having all or most of the attributes of the instance from which it was created. Nevertheless, the model works well with new instances, and for fold 7 all the 6 considered examples are correctly labeled. This is thanks to the fact that, even if no rule covers a test instance, it is assigned the conclusion of the rule with lowest distance.

Table 3: Rule set for Labor (fold 7), with coverage and accuracy

Rule	Cov.	Acc.
(dur == 0.0) && (0.0 <= wage1 <= 0.4) && (wage2 == 0.394) && (wage3 == 0.617) && (cola == none) && (0.846 <= hours <= 0.849) && (0.0 <= stby_pay <= 0.454) && (0.12 <= shift_diff <= 0.195) && (0.0 <= holidays <= 0.333) && (empl_hplan == none) => (Class == bad)	0.059	1.0
(0.0 <= dur <= 1.0) && (0.4 <= wage1 <= 0.6) && (0.3 <= wage2 <= 0.4) && (wage3 == 0.617) && (cola == none) && (0.769 <= hours <= 1.0) && (pension == empl_contr) && (stby_pay == 0.454) && (0.08 <= shift_diff <= 0.24) && (educ_allw == no) && (holidays == 0.333) && (vacation == average) && (lngtrm_disabil == yes) && (bereavement == yes) && (empl_hplan == full) => (Class == good)	0.059	1.0
(0.0 <= dur <= 0.5) && (0.0 <= wage1 <= 0.4) && (0.394 <= wage2 <= 0.4) && (wage3 == 0.617) && (cola == none) && (0.846 <= hours <= 1.0) && (pension == none) && (stby_pay == 0.454) && (0.12 <= shift_diff <= 0.195) && (0.167 <= holidays <= 0.333) && (lngtrm_disabil == no) && (dntl_ins == none) && (empl_hplan == none) => (Class == bad)	0.059	1.0
(dur == 1.0) && (0.5 <= wage1 <= 0.98) && (0.5 <= wage2 <= 0.56) && (0.097 <= wage3 <= 0.968) && (cola == none) && (hours == 1.0) && (pension == empl_contr) && (stby_pay == 0.454) && (0.12 <= shift_diff <= 0.195) && (educ_allw == no) && (0.333 <= holidays <= 0.5) && (lngtrm_disabil == yes) && (dntl_ins == half) && (bereavement == yes) => (Class == good)	0.059	1.0
(0.0 <= dur <= 0.58) && (0.16 <= wage1 <= 0.361) && (wage2 == 0.394) && (wage3 == 0.617) && (cola == none) && (0.615 <= hours <= 0.846) && (pension == empl_contr) && (stby_pay == 0.454) && (0.08 <= shift_diff <= 0.4) && (educ_allw == no) && (0.333 <= holidays <= 0.5) && (lngtrm_disabil == yes) && (dntl_ins == half) && (bereavement == yes) => (Class == good)	0.059	1.0
(dur == 0.0) && (0.0 <= wage1 <= 0.4) && (wage2 == 0.394) && (wage3 == 0.617) && (0.846 <= hours <= 1.0) && (0.167 <= stby_pay <= 0.454) && (0.0 <= shift_diff <= 0.195) && (holidays == 0.333) && (lngtrm_disabil == no) && (dntl_ins == none) && (bereavement == no) && (empl_hplan == none) => (Class == bad)	0.059	1.0
...	...	...

4.1.3. Hepatitis

The Hepatitis data set also has a considerable number of attributes (19), but a larger number of instances (155), which allows the rules to generalize a bit more. Nevertheless, each fold generates around 100 instances. The first 6 of the 116 rules of fold 1 can be seen in Table 4. Most of the rules cover 4 examples (0.029 coverage, considering that approximately 90% of the instances are used for training), but their accuracy largely vary: while some of them match all the covered training instances (such as rule 1), others are only successful in labeling the instance they were created from (for instance rule 4). This leads to lower the accuracy when the closest rule to a test instance represents an intricate training example, not very representative of a general reality for the problem. In this fold, the classifier achieves 64.7% of accuracy on the 17 tested examples. The whole set of rules and test instances with their real and predicted class values can be seen in the Hepatitis files present in the "Results" folder of this project.

Table 4: Rule set for Hepatitis (fold 1), with coverage and accuracy

Rule	**Cov.**	**Acc.**
(0.521 <= Age <= 0.761) && (Sex == 0.0) && (0.0 <= Steroid <= 1.0) && (Antivirals == 1.0) && (Fatigue == 0.0) && (0.0 <= Malaise <= 1.0) && (Anorexia == 1.0) && (Liver big == 0.0) && (Liver firm == 0.0) && (0.0 <= Spleen palpable <= 1.0) && (0.0 <= SPiders <= 1.0) && (Ascites == 1.0) && (Varices == 1.0) && (0.065 <= Bilirubin <= 0.377) && (0.182 <= Alk phosphate <= 0.327) && (0.009 <= Sgot <= 0.08) && (0.326 <= Alubimin <= 0.465) && (Protime == 0.619) && (Histology == 1.0) => (Class == Live)	0.029	1.0
(0.718 <= Age <= 0.761) && (Sex == 0.0) && (0.0 <= Steroid <= 1.0) && (Antivirals == 1.0) && (Fatigue == 0.0) && (0.0 <= Malaise <= 1.0) && (Anorexia == 1.0) && (0.0 <= Liver big <= 1.0) && (0.0 <= Liver firm <= 0.583) && (0.0 <= Spleen palpable <= 1.0) && (SPiders == 0.0) && (Ascites == 1.0) && (Varices == 1.0) && (0.146 <= Bilirubin <= 0.221) && (0.295 <= Alk phosphate <= 0.524) && (0.041 <= Sgot <= 0.36) && (0.279 <= Alubimin <= 0.399) && (0.38 <= Protime <= 0.619) && (0.0 <= Histology <= 1.0) => (Class == Die)	0.029	0.75
(0.437 <= Age <= 0.606) && (Sex == 0.0) && (Steroid == 1.0) && (Antivirals == 1.0) && (Fatigue == 1.0) && (Malaise == 1.0) && (Anorexia == 1.0) && (Liver big == 1.0) && (0.0 <= Liver firm <= 1.0) && (0.0 <= Spleen palpable <= 1.0) && (0.0 <= (SPiders <= 1.0) && (Ascites == 1.0) && (0.0 <= Varices <= 1.0) && (0.091 <= Bilirubin <= 0.221) && (0.216 <= Alk phosphate <= 0.42) && (0.014 <= Sgot <= 0.199) && (0.326 <= Alubimin <= 0.488) && (0.56 <= Protime <= 0.66) && (Histology == 1.0) => (Class == Die)	0.029	0.25
(0.324 <= Age <= 0.62) && (Sex == 0.0) && (0.0 <= Steroid <= 1.0) && (Antivirals == 1.0) && (Fatigue == 0.0) && (Malaise == 0.0) && (Anorexia == 0.0) && (Liver big == 1.0) && (Liver firm == 0.0) && (0.0 <= Spleen palpable <= 1.0) && (SPiders == 0.0) && (0.0 <= Ascites <= 1.0) && (0.0 <= Varices <= 1.0) && (0.117 <= Bilirubin <= 0.558) && (0.342 <= Alk phosphate <= 0.703) && (0.003 <= Sgot <= 0.416) && (0.163 <= Alubimin <= 0.419) && (0.51 <= Protime <= 0.619) && (Histology == 1.0) => (Class == Live)	0.029	0.5
(0.451 <= Age <= 0.493) && (Sex == 0.0) && (0.0 <= Steroid <= 1.0) && (Antivirals == 1.0) && (Fatigue == 1.0) && (Malaise == 1.0) && (Anorexia == 1.0) && (Liver big == 1.0) && (Liver firm == 1.0) && (Spleen palpable == 1.0) && (SPiders == 1.0) && (Ascites == 1.0) && (Varices == 1.0) && (0.078 <= Bilirubin <= 0.091) && (0.126 <= Alk phosphate <= 0.219) && (0.0 <= Sgot <= 0.077) && (0.442 <= Alubimin <= 0.605) && (0.47 <= Protime <= 1.0) && (Histology == 0.0) => (Class == Live)	0.029	1.0
(0.296 <= Age <= 1.0) && (Sex == 0.0) && (Steroid == 1.0) && (Antivirals == 1.0) && (Fatigue == 0.0) && (0.0 <= Malaise <= 1.0) && (Anorexia == 1.0) && (Liver big == 1.0) && (0.0 <= Liver firm <= 1.0) && (Spleen palpable == 1.0) && (SPiders == 1.0) && (Ascites == 1.0) && (Varices == 1.0) && (Bilirubin == 0.052) && (0.089 <= Alk phosphate <= 0.295) && (0.028 <= Sgot <= 0.151) && (0.442 <= Alubimin <= 0.535) && (0.619 <= Protime <= 0.74) && (Histology == 0.0) => (Class == Live)	0.029	1.0
...	...	...

4.1.4. Breast cancer

The Breast cancer data set has 9 meaningful attributes (the identifier one is ignored) and 699 instances. The folds produce different models, and the first 6 of the 352 rules generated by the first fold's classifier are represented in Table 5. Due to the lower number of attributes and greater number of instances, more generic rules can be obtained than in previous data sets, such as the Labor one, so that each rule covers multiple training instances and can be well generalized to different test cases. Among the remarkable cases we find rule 2, that covers a third of the total training instances, many more than other ones, such as the first one, covering only the 0.5% of them. This means that the ranges of values expressed by the second rule are general enough to cover a variety of instances, but what is more important is that all of them are successfully recognized as a benignant case: it has extracted some patterns of biological measurements that discard the presence of malignant cancer.

Table 5: Rule set for Breast cancer (fold 1), with coverage and accuracy

Rule	**Cov.**	**Acc.**
(0.222 <= Clump Thickness <= 0.778) && (Uniformity of Cell Size == 0.556) && (Uniformity of Cell Shape == 0.333) && (0.222 <= Marginal Adhesion <= 1.0) && (0.222 <= Single Epithelial Cell Size <= 1.0) && (0.0 <= Bare Nuclei <= 0.889) && (Bland Chromatin == 0.222) && (0.0 <= Normal Nucleoli <= 0.444) && (Mitoses == 0.0) => (Class == Malignant)	0.005	1.0
(0.0 <= Clump Thickness <= 0.556) && (Uniformity of Cell Size == 0.0) && (Uniformity of Cell Shape == 0.0) && (Marginal Adhesion == 0.0) && (0.0 <= Single Epithelial Cell Size <= 0.111) && (Bare Nuclei == 0.0) && (0.0 <= Bland Chromatin <= 0.222) && (Normal Nucleoli == 0.0) && (Mitoses == 0.0) => (Class == Benign)	0.326	1.0
(Clump Thickness == 0.444) && (0.111 <= Uniformity of Cell Size <= 0.222) && (Uniformity of Cell Shape == 0.222) && (0.0 <= Marginal Adhesion <= 0.222) && (0.111 <= Single Epithelial Cell Size <= 0.556) && (0.0 <= Bare Nuclei <= 1.0) && (0.111 <= Bland Chromatin <= 0.444) && (Normal Nucleoli == 0.0) && (Mitoses == 0.0) => (Class == Malignant)	0.006	0.5
(Clump Thickness == 0.222) && (0.0 <= Uniformity of Cell Size <= 0.111) && (Uniformity of Cell Shape == 0.0) && (Marginal Adhesion == 0.0) && (Single Epithelial Cell Size == 0.111) && (0.111 <= Bare Nuclei <= 0.283) && (Bland Chromatin == 0.222) && (Normal Nucleoli == 0.0) && (Mitoses == 0.0) => (Class == Benign)	0.008	1.0
(Clump Thickness == 0.444) && (0.111 <= Uniformity of Cell Size <= 0.222) && (0.222 <= Uniformity of Cell Shape <= 0.333) && (0.222 <= Marginal Adhesion <= 0.333) && (0.111 <= Single Epithelial Cell Size <= 0.333) && (0.222 <= Bare Nuclei <= 0.667) && (0.222 <= Bland Chromatin <= 0.333) && (0.333 <= Normal Nucleoli <= 0.667) && (Mitoses == 0.0) => (Class == Malignant)	0.006	0.75
(Clump Thickness == 0.333) && (0.0 <= Uniformity of Cell Size <= 0.333) && (0.0 <= Uniformity of Cell Shape <= 0.333) && (0.0 <= Marginal Adhesion <= 0.111) && (Single Epithelial Cell Size == 0.111) && (0.0 <= Bare Nuclei <= 0.222) && (Bland Chromatin == 0.111) && (Normal Nucleoli == 0.0) && (Mitoses == 0.0) => (Class == Benign)	0.03	1.0
...	...	...

4.2. Global results

The accuracy of all folds have been stored in tabular files in the "Results" folder, and are represented in Table 6, along with the mean accuracy and the standard deviation. The results are compared in Table 7 with those reported by Domingos' original implementation of the RISE algorithm. It can be seen that the current implemented has obtained slightly better results for the 3 of the 4 tested data sets. However, both implementations should be considered as producing similar results, due to the standard deviation being greater than the difference between means. The standard deviation is higher in two data sets and lower in the other one, but similar on average (8.33 compared with 8.77). As for the Breast cancer data set, it was not tested in the original work of Domingos, so it cannot be compared. In fact, another data set also called Breast cancer was used there, but due to the great difference in the attributes it makes no sense to compare the two cases.

Table 6: Accuracy for each data set (for each fold, mean and standard deviation)

Accuracy / Data set	**Breast cancer**	**Contact lenses**	**Hepatitis**	**Labor**
Cross-validation fold 1	0.944	0.8	0.84	1
Cross-validation fold 2	0.843	0.8	0.08	0.833
Cross-validation fold 3	0.9	1	0.813	0.833
Cross-validation fold 4	0.929	0.8	0.867	1
Cross-validation fold 5	0.971	0.8	0.733	1
Cross-validation fold 6	0.957	-	0.8	0.833
Cross-validation fold 7	0.971	-	0.8	1
Cross-validation fold 8	0.914	-	0.867	0.8
Cross-validation fold 9	0.913	-	0.867	0.8
Cross-validation fold 10	0.986	-	0.8	1
Mean	0.933	0.84	0.784	0.91
Standard deviation	0.04	0.08	0.079	0.091

Table 7: Comparison of mean accuracy with standard deviation between the original implementation and the current one.

Implementation / Data set	**Breast cancer**	**Contact lenses**	**Hepatitis**	**Labor**
Original	-	77.2±12.8	78.3±5.7	87.2±7.8
Current	93.3±4.0	84.0±8.0	78.4±7.9	91.0±9.1

5. Code execution instructions

The code files of this work is found in the "Source" folder. In particular, the "main.py" is the one to be executed, by running the following shell command: "python3 -m main". It is required to have Python 3 installed, e.g using "sudo apt-get install python3.6" in a Linux machine. An alternative to the terminal is to open "main.py" in a Python IDE (e.g. PyCharm) and press the run button.

6. Conclusions and future work

The RISE algorithm is an original algorithm that combines instance-based learning with rule-based induction, that has proved to be capable of producing considerably accurate results in the 4 tested data sets. For a greater understanding of how good it is in nowadays' machine learning context, it would be interesting to compare these accuracy results with those of state-of-the-art classification algorithms, such as Support Vector Machines, using the same training and testing splits as used for the tested data sets, and possibly extend the research to a wider collection of data sets.

7. References

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