Third assignment - Artificial Intelligence 2022-2023

Departament d'Informàtica i Enginyeria Industrial
Universitat de Lleida
carlos@diei.udl.cat
eduard.torres@udl.cat
josep.alos@udl.cat

1. Description

The goal of this assignment is to evaluate the knowledge of the student about the supervised and unsupervised algorithms. Those algorithms are described in the slides of the subject, along with the tasks that compose this assignment.

The assignment must be developed on top of the provided material, attached along with the description of the assignment.

1.1. Decision trees - Implementation

1.1.1. T9 - Recursive construction of the decision tree. 1.5 points

Implement the recursive `buildtree(part, scoref, beta)` method, which depends on the impurity measure `scoref` and the `beta` criterion to restrict the construction.

1.1.2. T10 - Iterative construction of the decision tree. 2 points

Implement the `iterative_buildtree(part, scoref, beta)` method. The resulting tree must be the same as the one built by the recursive version (`buildtree(...)` method).

1.1.3. T12 - Classify method. 0.5 points

Once we have the tree we will be able to classify new data using it. Implement the function `classify(tree, row)` that returns the label predicted for `row`. The criterion to assign the label in leaves that have multiple labels must be justified in the report.

IMPORTANT In the lectures we have developed the code assuming that the last element of each row is the label. In future data, we will not have this value. You must ensure that the method will work correctly with a `row` that does not have this last value "label".

1.1.4. T16 - Tree pruning. 1 point

Implement the `prune(tree: DecisionNode, threshold: float)` method. This method must execute the pruning algorithm explained in the lectures:

- *Build a complete tree* - In our particular case, we already receive this tree as a parameter.
- For each pair of leaves with the same parent, test if joining them increments the impurity below the `threshold`. If that is the case, join the leaves and convert the parent to a leaf.
- Repeat for each pair of leaves until we cannot prune more of them.

1.2. Decision trees - Evaluation.

1.2.1. T13 - Test performance. 0.25 point

Define the `get_accuracy(tree: DecisionNode, dataset)` method that, given a tree and a dataset (either training or test), it returns the ratio of correctly classified rows in the dataset.

1.2.2. Cross-validation. 1 point

Implement the `cross_validation(dataset, k, agg, seed, scoref, beta, threshold)` method. This method is the responsible of executing the *cross-validation* algorithm, described below. The parameters are:

- dataset: The data to be used
- k: The number of partitions (*folds*) that will be generated
- agg: The aggregation function of the different accuracies
- seed: The seed used to separate the dataset
- scoref, beta, threshold: Parameters that will be used to build the tree.

As an aggregation function you can use the mean:

```
def mean(values: List[float]):
    return sum(values) / len(values)
```

1.2.3. Find the best threshold parameter. 0.25 points

As we know, modifying the parameters will result in trees with more or less generalization capabilities. Explore at least 4 distinct values for the `threshold` parameter. Choose the best one based on the score obtained by *cross-validation.*

For this, split first the `iris.csv` datset into train and test. Obtain the best configuration using cross-validation over the train dataset. Once you have choosen the best parameters, train a new tree using the entire train dataset and then report the accuracy of this tree over the test dataset.

Explain the results in the report.

1.3. Clustering - Implementation.

For all the exercises, consider the squared euclidean distance function.

1.3.1. T9 - Total distance. 0.5 points

Modify the `kcluster` method to return a tuple composed by (1) the found centroids, and (2) the sum of the distances of all the points to their centroid.

1.3.2. T11 - Restarting policies. 2 points

We want k-means to assign all the k clusters such that it minimizes the sum of the distances from all the items to their centroid. As the result of k-means depends on the initialization of the centroids, we should execute it multiple times to find the best assignation of clusters, the one that minimizes this sum of distances.

Improve the `kcluster` method such that it takes into account this number of executions (parameterizable) and it keeps as the result of the best configuration.

1.4. Clustering - Evaluation.

1.4.1. T10 - Distance in function of k. 0.5 points

Show the total distance in function of different values of k.

1.4.2. Choose a value of k. 0.5 points

Until this moment, we used a fixed k value to do our experiments. With a real dataset, it is difficult to estimate the value we should choose.

Search for the "elbow method". This method allows us to find a value for k that is good enough without overfiting our model. Determine this value k for the dataset `blogdata_full.txt`. Add to the report the plot obtained from the results in the previous exercise, and analyze the result.

2. Implementation

The quality and efficiency of the algorithms will be graded. You must take into account the following points for the implementation:

- The programming language is Python.
- The usage of descendant and Object-Oriented Programming design.
- The simplicity and readability of the code.
- It is recommended to follow the style guide [1].

3. Report

Report in pdf (maximum ≈ 4 pages long) that includes the description of the design decisions during the implementation, along with the experimental results and theoretical comments. The report must contain **at least** the items 1.1.3, 1.2.3, and 1.4.2.

The first page must contain the name of the group members. The assignment can be done in pairs or individually.

4. Content to deliver

The content that must be delivered is:

[1] `https://www.python.org/dev/peps/pep-0008/`

- All the source code files, either added or modified.
- PDF report.

The content of the assignment must be delivered in a compressed package named `ia-prac3.[tgz|tar.gz|zip]`