Second assignment - Artificial Intelligence

Departament d'Informàtica
Universitat de Lleida

2025-2026

Description

The goal of this assignment is to evaluate the student's knowledge of the supervised and unsupervised algorithms. Those algorithms are described in the course slides.

The assignment must be developed using the provided material, which is attached along with the assignment description.

WARNING: We will conduct in-person individual validations. As specified in the course guide, failing the validation will imply failing the assignment.

Supervised Learning - Decision Trees (2.5 points)

For this part, you must modify the `decision_tree.py` file. Additionally, we will use a collection of datasets to evaluate and test our implementation, which you will find in the `datasets/supervised/` folder.

Tasks:

- (0.5 points) Complete the main method for the `decision_tree.py` file. Note that in the file `utils.py` there is a method `read_csv`. Add a boolean parameter `ignore_first` to consider the first row as the heading of the dataset, so it works with the provided datasets.
- (0.5 points) Implement the `_iterative_build_tree` method from the `DecisionTreeClassifier` class. This method receives a list of observations and a list of their corresponding labels, and will train a decision tree. The decision tree is saved at the attribute `self.tree_`.
- (0.5 points) Improve your classifier to consider all the subsets of categorical values (i.e., consider queries of $\in$ instead of $=$).
- (0.25 points) Incorporate the `beta` criterion into your tree-building logic.
- (0.75 points) Implement the `_prune_tree` method from the `DecisionTreeClassifier` class. This method will modify the attribute `self.tree_`.

Supervised Learning - Naive Bayes (3 points)

For this part, you need to modify the `bayesian.py` file. We will use a collection of datasets for training and testing the model, which you will find in the `datasets/supervised/` folder.

Tasks:

- (0.75 points) Complete the `main` method in the `bayesian.py` file. The loading of the SMS file format is provided in the `utils.py` file, but you must implement a tokenize method to split each message into tokens.
- (1 point) Implement the `fit` method for the `MultinomialNaiveBayesClassifier`.
- (1 point) Implement the `predict` method for the `MultinomialNaiveBayesClassifier`.
- (0.25 points) Improve your classifier to consider an `assumed_probability`.

Unsupervised Learning - KMeans (3.5 points)

In this part, we will focus on the *k-means* algorithm. In particular, you must modify the `clustering.py` file. You can also find in the `datasets/unsupervised/` folder the collection of datasets that we will use.

Tasks:

- (0.5 points) Complete the `main` method in the `clustering.py` file. You need to modify `read_csv` in `utils.py` and add a new boolean parameter to consider the first column as the identifier (we will ignore those if it is set to true).
- (2 points) Implement the fit method of the `KMeans` class. This method must initialize the centroids' positions (`self.centroids_`) randomly and apply the *k-means* algorithm that we described in the slides. You must consider the k and distance parameters, where distance can be `"euclidean"` or `"squared-euclidean"`. Finally, you must store the index of the centroid assigned to each prototype in the `self.X_assignments_` attribute, as well as the sum of the distance of these points to their centroid in the self.distances_ attribute.
- (1 point) 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 assignment of clusters, the one that minimizes this sum of distances. Improve the `KMeans` class such that it takes into account this number of executions and keeps it as the result of the best configuration. You must add an additional parameter (e.g., `n_restarts`) to the `KMeans` constructor. This parameter should also be modifiable through the CLI.

Report (1 point)

Analyze and present the following experiments using your implementations:

- Using a decision tree in the iris dataset, compare different values for the pruning parameters as well as beta. Compare also what happens when one is enabled but not the other.
- Using a decision tree in the iris dataset, try different values for the score function, the pruning parameters, and beta. Present the best parameters you could find for the dataset.
- Using the *k-means* algorithm in the blogdata dataset, show the total distance as a function of different values of *k*. **Note:** discuss why you decided on the metric you used.
- Investigate about the *elbow method* to automatically select a value for *k*. Using this method, which *k* is the best one for the blogdata dataset? Why?

The report must be a PDF, with a maximum of 5 pages. The report must include the description of the design decisions during the implementation, problems found, and finally the experimental results and theoretical comments required in the assignment.

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

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]

Content to deliver

The content that must be delivered is:

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

The content of the assignment must be delivered in a compressed package named `ia-prac2.[tgz|tar.gz|zip]`

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