# Introduction to Machine Learning
# Work 3
# Lazy learning exercise

---

## Contents

# 1 Lazy learning exercise

## 1.1 Introduction

In this exercise, you will learn about lazy learning. In particular, you will work on a comparative study in the use of different retention techniques and reduction techniques in a K-Nearest Neighbor algorithm. You will apply lazy learning to a classification task. It is assumed that you are familiar with the concept of cross-validation. If not, you can read this paper (*located at http://ai.stanford.edu/~ronnyk/accEst.pdf*):

*[1] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the International Joint Conferences on Articial Intelligence IJCAI-95. 1995.*

Briefly, an s-fold cross validation (s = 10 in your case) divides a data set into s equal-size subsets. Each subset is used in turn as a test set with the remaining (s-1) data sets used for training.

For the validation of the different algorithms, you need to use a T-Test or another statistical method. Next reference is a **mandatory reading proposal** (in Racó) on this topic:

*[2] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7 (December 2006), 1-30.*

This article details how to compare two or more learning algorithms with multiple data sets.

## 1.2 Methodology of the analysis

As in the previous work assignment, you will analyze the behavior of the different algorithms by comparing the results in a pair of well-known data sets (**medium and large size**) from the UCI repository. In that case, you will also use the class as we are testing several supervised learning algorithms. In particular, in this exercise, you will receive the data sets defined in .arff format but divided in **ten training and test sets** (they are the *10-fold cross-validation* sets you will use for this exercise).

This work is divided in several steps:

1. Read the source data for training and testing the algorithms. Improve the parser developed in previous works in order to use the class attribute, too. Now, you need to read and save the information from a training and their corresponding testing file in a `TrainMatrix` and a `TestMatrix`, respectively. Recall that you need to normalize all the numerical attributes in the range [0..1]. For representing the instance base, the most used by its simplicity and applicability is a flat structure. The instances are represented as a feature vector approach by means of a set of attribute-value pairs. The `TrainMatrix` and the `TestMatrix` contain the set of instances for training and testing, respectively.

2. Write a Python function that automatically repeats the process described in previous step for the 10-fold cross-validation files. That is, read automatically each training case and run each one of the test cases in the selected classifier.

3. Implement a `K`-Instance-Based Learning algorithm. Write a Python function for classifying, using a K-NN algorithm, each instance from the `TestMatrix` using the `TrainMatrix` to a classifier called `kIBLAlgorithm(...)`. You decide the parameters for this classifier. Note that instead of retrieving the most similar instance, the similarity function will return the `K` most similar instances. **Justify your implementation and add all the references you have considered for your decisions.**

   a. Assume that the K-IBL algorithm returns the `K` most similar instances (i.e., also known as cases) from the `TrainMatrix` to the `current_instance`. The value of `K` will be setup in your evaluation to 1, 3, 5, and 7.
   b. Implement as similarity function the Minkowski metric, analyze the r = 1, 2, and 3.
   c. To decide the solution of the `current_instance`, you may consider using **two voting policies:** Most voted solution and Modified Plurality or another one that you may decide.

      I. **Most voted solution** is the simplest method. Technically, a method for breaking ties should also be specified. You should decide the method in case of ties.
      II. **Modified Plurality** computes the most voted solution but in case of ties, it removes the last `k` nearest neighbors and computes again the most voted solution. In case of ties, the process is repeated. The process finishes when there is a winner solution or when just one nearest neighbor remains.

   d. Instance-based learning algorithms may learn during the generalization process. In this work, you will implement different retention policies.
      I. **Never retain (NR**). The algorithm never retains the `current_instance` in the instance base. The generalization process is based exclusively on the instances present in the training set.
      II. **Always retain (AR)**. The algorithm retains all new solved instances in the instance base.
      III. **Different Class retention (DF)**. The algorithm retains all the instances that have been solved incorrectly.
      IV. **Degree of Disagreement (DD)**. This strategy is based on a measure of dispersion -or disagreement- of the k most similar instances. It is inspired by the active learning policy, which, in turn, uses a variation of the Query by Committee (QbC) algorithm. The proposal is to use the k set as a virtual committee. So, each member of the committee can be seen as a case in k, and the vote it generates as the class associated to this case. The formula is as follows:

$$d = \frac{\#remaining\_cases}{(\#classes - 1) \times \#majority\_cases}$$

The equation measures d as the degree of disagreement over the `K` retrieved instances, where `#classes` is the number of different classes[1] in the instance base, where `#majority_cases` is the number of instances with the most assigned class (i.e., the majority class in `K`) and `#remaining_cases` is the number of instances with classes other than the majority one. In this manner, `|k|= #majority_cases + #remaining_cases`. Thus, disagreement value d reaches value 1 when every instance in `K` yields to a different classification and so, instances disagree the most, while it approximates to 0 when there is a clear majority. #classes -1 is introduced in order to normalize disagreement from 0 to 1. As an implementation detail, it is important that the `K` set is ordered according to the similarity. In case of ties, you will solve ties by just considering the first one that has been found. So, it is the most assigned and most similar class since `K` is ordered according to similarity.

For evaluating the performance of the K-IBL algorithm, we will use the classification **accuracy** (i.e., the average of correctly classified instances) and the **efficiency** (i.e., average problem-solving time). To this end, at least, you should store the number of cases correctly classified, the number of cases incorrectly classified, and the problem-solving time. This information will be used for the evaluation of the algorithm. You can store your results in a memory data structure or in a file. Keep in mind that you need to compute the average accuracy and the average efficiency over the 10-fold cross-validation sets.

At the end, you will have a K-IBL algorithm with different values for the `K` parameter, different values for the `r` parameter, two policies for deciding the solution of the `current_instance`, and four different retention policies. You will analyze the behavior of these parameters in the K-IBL algorithm and decide which combination results in the **best k-IBL algorithm**.

You can compare your results in terms of classification accuracy and efficiency (in time). Extract conclusions by analyzing **two large** enough data sets (the same as in step 3). **At least one of these data sets will contain numerical and nominal data**.

4. Departing from the best k-IBL algorithm. Modify the k-IBL algorithm so that it includes a preprocessing for reducing the training set, you will call this algorithm as `reductionKIblAlgorithm(...)`. In general instance-based learning algorithms retain all the training instances. However, the training set may be too large and it may also contain inconsistent and noisy instances. For this reason, it is important to decide which instances to store for use during the generalization. In this work, you will implement several preprocessing reduction techniques:

[1]This is a general definition which, in our case, could correspond to all the classes in the instance base or just the classes appearing in k. We take the former option, so all classes are considered at each iteration step.

a. Condensed Nearest Neighbour Rule (CNN) or Selective Nearest Neighbour Rule (SNN).
b. Edited Nearest Neighbour rule (ENN) or Repeteated Edited Nearest Neighbour rule (RENN) or All k-NN algorithms.
c. IB3
d. **Two algorithms** of the family of DROP algorithms (i.e., DROP1, DROP2, DROP3, DROP4, or DROP5).
e. Analyze the results of the `reductionKIblAlgorithm` in front of the previous `kIBLAlgorithm` implementation. To do it, setup both algorithms with the best combination obtained in your previous analysis. In this case, you will analyze your results in terms of classification accuracy, efficiency and storage (i.e, the average percentage of instances stored from the training to generalize the testing instances). Accordingly, the storage of the `kIBLAlgorithm` is 100%.

Note that all the reduction algorithms are detailed in:

*[3] D. Randall Wilson and Tony R. Martinez. 2000. Reduction Techniques for Instance-BasedLearning Algorithms. Mach. Learn. 38, 3 (March 2000), 257-286. DOI: https://doi.org/10.1023/A:1007626913721 https://link.springer.com/content/pdf/10.1023%2FA%3A1007626913721.pdf*

## 1.3 Work to deliver

In this work, you will implement Instance-based learning algorithms. You will analyze the behavior of different parameters in the k-IBL algorithm and the use of reduction techniques in a preprocessing process for reducing the memory requirements and to increase the execution speed. You may select two data sets (large enough to extract conclusions) for your analysis. At the end, you will find a list of the data sets available.

You will implement your own python code and use it to extract the performance of the different combinations. Performance will be measured in terms of classification accuracy, efficiency, and storage. The accuracy measure is the average of correctly classified cases. That is the number of correctly classified instances divided by the total of instances in the test file. The efficiency is the average problem-solving time. For the evaluation, you will use a T-Test or another statistical method [2].

From the accuracy and efficiency results, you will extract conclusions showing graphs of such evaluation and reasoning about the results obtained.

In your analysis, you will include several considerations.

1. You will analyze the k-IBL algorithm, with different parameter combinations. You will analyze which is the most suitable algorithm. The one with the highest accuracy will be named as the best k-IBL algorithm.

2. Once you have decided the best k-IBL algorithm, you will analyze different reduction techniques. The idea is to analyze if reduction techniques may increase accuracy,

efficiency and storage in a k-IBL algorithm and to extract conclusions of which reduction technique is the best one.

For example, some of questions that it is expected you may answer with your analysis:

- Which is the best value of K at each dataset?
- Which k-IBL algorithm is the best one at each dataset?
- Did you find useful the use of a voting scheme for deciding the solution of the current_instance?
- Which is the best similarity function for the k-IBL?
- Did you find differences in performance among the k-IBL and the reduction k-IBL?
- According to the data sets chosen, which reduction method provides you more advice for knowing the underlying information in the data set?
- Do you think it will be much better to improve the retention process rather than reducing the training set?

Apart from explaining your decisions and the results obtained, it is expected that you reason each one of these questions along your evaluation.

Additionally, **you should explain how to execute your code**. Remember to add any reference that you have used in your decisions.

You should deliver a report as well as the code in Python in a PyCharm project in Racó by **December, 22$^{th}$, 2019**.

Remember that the maximum size of the report is 20 pages, including description and graphs.

# 2 Data sets

Below, you will find a table that shows in detail the data sets that you can use in this work. All these data sets are obtained from the UCI machine learning repository. First column describes the name of the domain or data set. Next columns show #Cases = Number of cases or instances in the data set, #Num. = Number of numeric attributes, #Nom = Number of nominal attributes, #Cla. = Number of classes, Dev.Cla. = Deviation of class distribution, Maj.Cla. = Percentage of instances belonging to the majority class, Min.Cla. = Percentage of instances belonging to the minority class, MV = Percentage of values with missing values (it means the percentage of unknown values in the data set). When the columns contain a '-', it means a 0. For example, the Glass data set contains 0 nominal attributes and it is complete as it does not contain missing values.

| | Domain | #Cases | #Num. | #Nom. | #Cla. | Dev.Cla. | Maj.Cla. | Min.Cla. | MV |
|---|---|---|---|---|---|---|---|---|---|
| | *Adult* | 48,842 | 6 | 8 | 2 | 26.07% | 76.07% | 23.93% | 0.95% |
| | *Audiology* | 226 | - | 69 | 24 | 6.43% | 25.22% | 0.44% | 2.00% |
| | *Autos* | 205 | 15 | 10 | 6 | 10.25% | 32.68% | 1.46% | 1.15% |
| * | *Balance scale* | 625 | 4 | - | 3 | 18.03% | 46.08% | 7.84% | - |
| * | *Breast cancer Wisconsin* | 699 | 9 | - | 2 | 20.28% | 70.28% | 29.72% | 0.25% |
| * | *Bupa* | 345 | 6 | - | 2 | 7.97% | 57.97% | 42.03% | - |
| * | *cmc* | 1,473 | 2 | 7 | 3 | 8.26% | 42.70% | 22.61% | - |
| | *Horse-Colic* | 368 | 7 | 15 | 2 | 13.04% | 63.04% | 36.96% | 23.80% |
| * | *Connect-4* | 67,557 | - | 42 | 3 | 23.79% | 65.83% | 9.55% | - |
| | *Credit-A* | 690 | 6 | 9 | 2 | 5.51% | 55.51% | 44.49% | 0.65% |
| * | *Glass* | 214 | 9 | - | 2 | 12.69% | 35.51% | 4.21% | - |
| * | *TAO-Grid* | 1,888 | 2 | - | 2 | 0.00% | 50.00% | 50.00% | - |
| | *Heart-C* | 303 | 6 | 7 | 5 | 4.46% | 54.46% | 45.54% | 0.17% |
| | *Heart-H* | 294 | 6 | 7 | 5 | 13.95% | 63.95% | 36.05% | 20.46% |
| * | *Heart-Statlog* | 270 | 13 | - | 2 | 5.56% | 55.56% | 44.44% | - |
| | *Hepatitis* | 155 | 6 | 13 | 2 | 29.35% | 79.35% | 20.65% | 6.01% |
| | *Hypothyroid* | 3,772 | 7 | 22 | 4 | 38.89% | 92.29% | 0.05% | 5.54% |
| * | *Ionosphere* | 351 | 34 | - | 2 | 14.10% | 64.10% | 35.90% | - |
| * | *Iris* | 150 | 4 | - | 3 | - | 33.33% | 33.33% | - |
| * | *Kropt* | 28,056 | - | 6 | 18 | 5.21% | 16.23% | 0.10% | - |
| * | *Kr-vs-kp* | 3,196 | - | 36 | 2 | 2.22% | 52.22% | 47.78% | - |
| | *Labor* | 57 | 8 | 8 | 2 | 14.91% | 64.91% | 35.09% | 55.48% |
| * | *Lymph* | 148 | 3 | 15 | 4 | 23.47% | 54.73% | 1.35% | - |
| | *Mushroom* | 8,124 | - | 22 | 2 | 1.80% | 51.80% | 48.20% | 1.38% |
| * | *Mx* | 2,048 | - | 11 | 2 | 0.00% | 50.00% | 50.00% | - |
| * | *Nursery* | 12,960 | - | 8 | 5 | 15.33% | 33.33% | 0.02% | - |
| * | *Pen-based* | 10,992 | 16 | - | 10 | 0.40% | 10.41% | 9.60% | - |
| * | *Pima-Diabetes* | 768 | 8 | - | 2 | 15.10% | 65.10% | 34.90% | - |
| * | *SatImage* | 6,435 | 36 | - | 6 | 6.19% | 23.82% | 9.73% | - |
| * | *Segment* | 2,310 | 19 | - | 7 | 0.00% | 14.29% | 14.29% | - |
| | *Sick* | 3,772 | 7 | 22 | 2 | 43.88% | 93.88% | 6.12% | 5.54% |
| * | *Sonar* | 208 | 60 | - | 2 | 3.37% | 53.37% | 46.63% | - |
| | *Soybean* | 683 | - | 35 | 19 | 4.31% | 13.47% | 1.17% | 9.78% |
| * | *Splice* | 3,190 | - | 60 | 3 | 13.12% | 51.88% | 24.04% | - |
| * | *Vehicle* | 946 | 18 | - | 4 | 0.89% | 25.77% | 23.52% | - |
| | *Vote* | 435 | - | 16 | 2 | 11.38% | 61.38% | 38.62% | 5.63% |
| * | *Vowel* | 990 | 10 | 3 | 11 | 0.00% | 9.09% | 9.09% | - |
| * | *Waveform* | 5,000 | 40 | - | 3 | 0.36% | 33.84% | 33.06% | - |
| * | *Wine* | 178 | 13 | - | 3 | 5.28% | 39.89% | 26.97% | - |
| * | *Zoo* | 101 | 1 | 16 | 7 | 11.82% | 40.59% | 3.96% | - |