

ELEN0062 - Introduction to machine learning

Project 1 - Classification algorithms

October 2nd, 2026

In this first project, you will get accustomed to some classical machine learning algorithms and concepts, such as underfitting and overfitting. For each algorithm, we ask you to deliver a separate Python script. Make sure that your experiments are reproducible (e.g., by manually fixing random seeds). Add a *brief* report (PDF format) giving your observations and conclusions. The report must use the provided template (`report_template.tex`) and must not exceed **4 pages, including figures and tables**. The AI-use declaration described below does not count toward this limit. Do not modify the font size, margins, or section structure of the provided template. Concise answers are expected: do not repeat the project statement or include unnecessary implementation details.

Each project must be carried out by groups of *two to three students* and submitted to Gradescope¹ before *October 23rd, 23:59 GMT+2*. You will make two submissions on Gradescope: one for your Python scripts and one for your report. Note that attention will be paid to how you present your results. Careful thought, in particular—but not limited to—should be given when it comes to plots.

Use of AI. The use of generative AI tools is permitted, but it is neither encouraged nor required. Regardless of whether such tools are used, you are expected to fully understand and be able to explain every aspect of your submission. Your report must include a short AI-use declaration stating which tools were used, for which tasks, to what extent, and how their outputs were verified. If you did not use any generative AI tool, state this explicitly. Students may be asked to explain any part of their code, methodology, or report. Failure to disclose the use of generative AI tools, or any other external sources, may be considered academic fraud.

Files

You are given several files, among which are `data.py` and `plot.py`. The first one generates binary classification datasets with two real input variables. More precisely, you will mainly work with (a modified version of) the `make_moons` dataset from scikit-learn (see Figure 1). An additional XOR dataset will be used in the last question.

Generate datasets of 1500 samples. The first 1200 will be used as the learning set and the remaining ones as the test set.

The second file contains a function which depicts the decision boundary of a trained classifier. Note that you should use a dataset independent of the learning set to visualize the boundary.

Only modify the other files, which you must archive before submitting.

¹<https://www.gradescope.com>. The course entry code is JN44ND. Please register with your

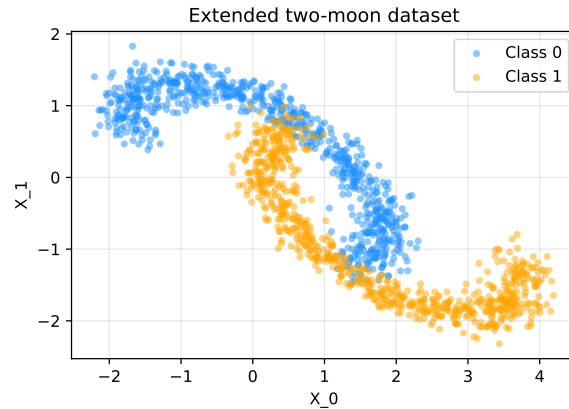


Figure 1: Modified two-moon dataset inspired by `make_moons`: the arcs are extended beyond semicircles, then horizontally elongated and rotated.

1 Decision tree (`dt.py`)

In this section, we will study decision tree models (see `DecisionTreeClassifier` from `sklearn.tree`). More specifically, we will observe how the model's complexity affects the classification boundary. To do so, we will build several decision tree models with `max_depth` values of 1, 3, 6, 12, and `None`. Answer the following questions in your report.

1. Observe how the decision boundary is affected by tree complexity.
 - (a) Illustrate and explain the decision boundary for each hyper-parameter value.
 - (b) Discuss when the model is clearly underfitting/overfitting and detail your evidence for each claim.
 - (c) Explain why the model seems more confident when `max_depth` is the largest.
2. Report the average test set accuracies over five generations of the dataset, along with the standard deviations, for each value of `max_depth`. Briefly comment on them.

2 K-nearest neighbors (`knn.py`)

In this section, we will study nearest neighbors models (see the `KNeighborsClassifier` class from `sklearn.neighbors`). More specifically, we will observe

official ULiège email address.

how model complexity impacts the classification boundary. To do so, we will build several nearest neighbor models with `n_neighbors` values of 1, 9, 45, 225, and 901. Answer the following questions in your report.

1. Observe how the decision boundary is affected by the number of neighbors.
 - (a) Illustrate the decision boundary for each value of `n_neighbors`.
 - (b) Comment on the evolution of the decision boundary with respect to the number of neighbors.
2. Report the average test set accuracies over five generations of the dataset, along with the standard deviations, for each value of `n_neighbors`. Briefly comment on them.

3 Naive Bayes classifier (`nb.py`)

In this section, we study the Naive Bayes (NB) classifier model. NB is a probability-based method which computes its predictions according to the posterior probability:

$$\hat{f}(\mathbf{x}) = \operatorname{argmax}_y p(y|x_1, \dots, x_p), \quad (1)$$

where $\mathbf{x} = (x_1, \dots, x_p)$ denotes the p input variables. To compute this posterior, the NB classifier postulates that the input variables are conditionally independent given the output. More formally,

$$p(\mathcal{X}_i|\mathcal{Y}, \mathcal{Z}) = p(\mathcal{X}_i|\mathcal{Y}) \quad \forall \mathcal{Z} \in \mathcal{P}(\{\mathcal{X}_1, \dots, \mathcal{X}_{i-1}, \mathcal{X}_{i+1}, \dots, \mathcal{X}_p\}) \quad (2)$$

where $\mathcal{P}(S)$ designates the power set of S . Given this hypothesis, the predictions of the NB classifier can be obtained as

$$\hat{f}(\mathbf{x}) = \operatorname{argmax}_y p(y) \prod_{i=1}^p p(x_i|y). \quad (3)$$

To compute (3), the prior $p(y)$ and the likelihoods $p(x_i|y)$ must be estimated. For discrete variables, which is the case of $\mathcal{Y}$, you can use the corresponding ratio over the learning set.

For the continuous input variables, we will estimate each class-conditional density using kernel density estimation (KDE²), a non-parametric density estimator. Given the n_y learning examples belonging to class y , the density of input variable i is estimated as

$$\hat{p}(x_i|y) = \frac{1}{n_y h} \sum_{j:y_j=y} K\left(\frac{x_i - x_{ji}}{h}\right), \quad (4)$$

²https://en.wikipedia.org/wiki/Kernel_density_estimation

where $h > 0$ is the bandwidth and K is a kernel function. In this project, use a Gaussian kernel:

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right). \quad (5)$$

The bandwidth h controls how much the estimated density is smoothed.

1. Show that (3) is equivalent to (1) under the NB independence assumption.
2. Implement your own NB estimator according to the above description and following the scikit-learn convention³. Use log-probabilities when combining likelihoods to avoid numerical underflow. *Suggestion: Fill in the class whose template is given in `nb.py`.*
3. Still using the same sizes for the learning and test sets as in the previous experiments (i.e., respectively 1200 and 300), illustrate the decision boundaries obtained for bandwidth values $h = 0.005, 0.03, 0.2, 0.8$, and 3.2. Comment on the effect of the bandwidth on the estimated classifier and discuss underfitting/overfitting when relevant.
4. Report the average test set accuracies over five generations of the dataset, along with the standard deviations, for each value of h . Briefly comment on the results.

4 Method comparison (`mc.py`)

We now compare the three methods. Since decision trees, k-nearest neighbors, and KDE Naive Bayes depend on a hyper-parameter, we need to find a way to tune it. You can re-use the set of candidate hyper-parameter values suggested in the previous questions, paying attention to the maximum number of neighbors that can be used by your protocol.

Answer the following questions in your report:

1. Explain a way to tune the value of `max_depth`, `n_neighbors`, and the KDE bandwidth h on the learning set.
2. Implement this method and use it to compute the average test set accuracies over five generations of the two-moon dataset, along with the standard deviations, for the tuned decision tree, k-nearest neighbors, and KDE Naive Bayes methods.
3. Compare these results. Discuss the ranking of the methods on the two-moon dataset and relate the observed results to their decision boundaries and modeling assumptions.

³<http://scikit-learn.org/dev/developers/>

4. We now consider the XOR dataset provided in `data.py`. Apply the three methods to this dataset, tuning their hyper-parameters using only the learning set, and report their mean test set accuracies and standard deviations over five generations. Illustrate the corresponding decision boundaries.
5. Compare the behavior of the three methods on the XOR dataset. In particular, explain why the KDE Naive Bayes classifier can fail on this problem even though KDE is able to model complex one-dimensional densities.