University of Liège

Classification algorithms : Report

Introduction to machine learning

Maxime Meurisse (20161278)
Valentin Vermeylen (20162864)

Master in Civil Engineering, specialization in *Intelligent Systems*
Academic year 2019-2020

1 Decision tree

1. Effect of the tree depth on the decision boundary

(a) *Illustrate and explain the decision boundary for each depth*

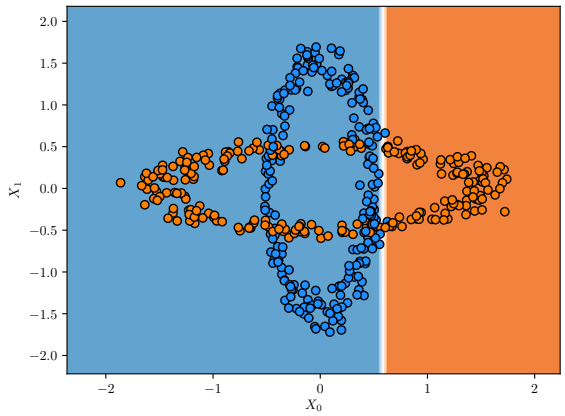
We built several decision tree models with different depth values, for both datasets. To illustrate our results, we present the decision boundary of each model at figures 1 (for dataset 1) and 2 (for dataset 2).

In order to make the visualization clearer, we only displayed 25% of the points of each testing set and did not include the depth of 8, as the decision boundary is almost identical to the unconstrained depth one.

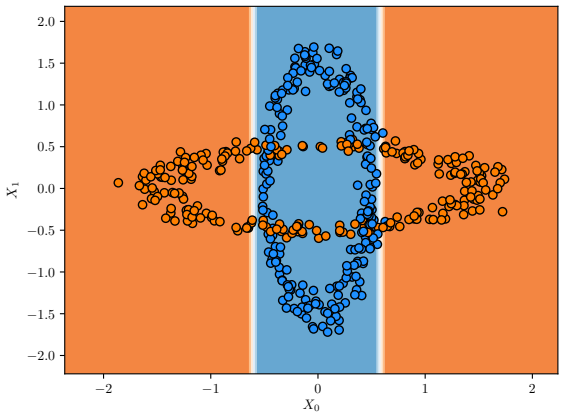
On every plot (not just for this section, but for the whole report), an accuracy is associated with each decision boundary. It is computed on the whole testing set (1850 points).

Firstly, we can see that, in all cases, the model partitions the features space along the axes. This result is logical considering the operation of the algorithm : it splits the dataset by thresholding the value of one of the features.

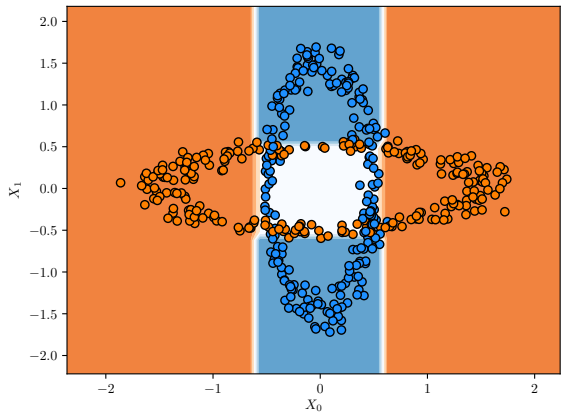
Secondly, we find that, as the depth of the tree increases, the number of partitions of the space increases exponentially. This result is also logical: incrementing the depth of a tree by 1 increases exponentially the total number of nodes since each node at depth $d-1$ is sub-divided into usually two (in this algorithm) nodes.



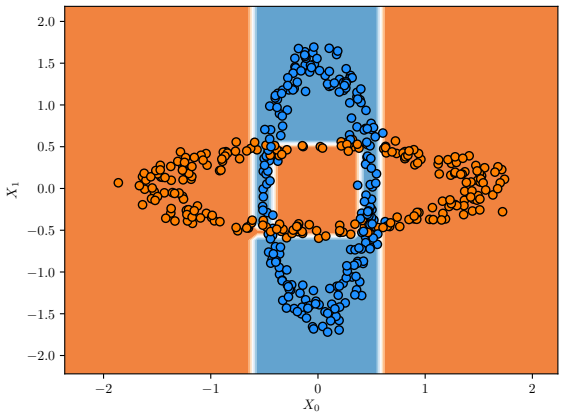
(a) Decision tree : depth of 1 (acc = 0.68)



(b) Decision tree : depth of 2 (acc = 0.87)



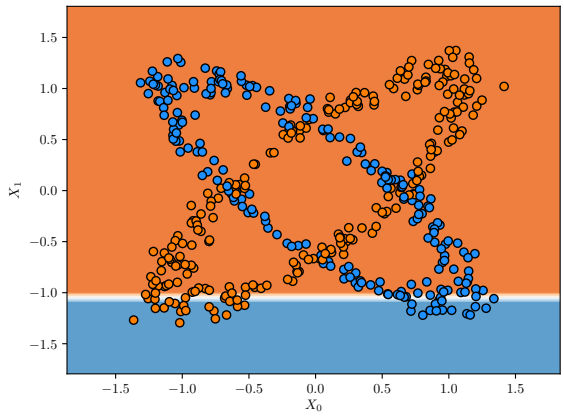
(c) Decision tree : depth of 4 (acc = 0.86)



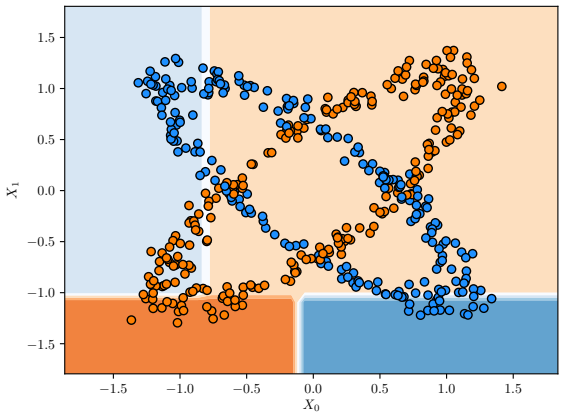
(d) Decision tree : unconstrained (acc = 0.93)

Figure 1 – Decision tree boundary for several depths using `make_data1`

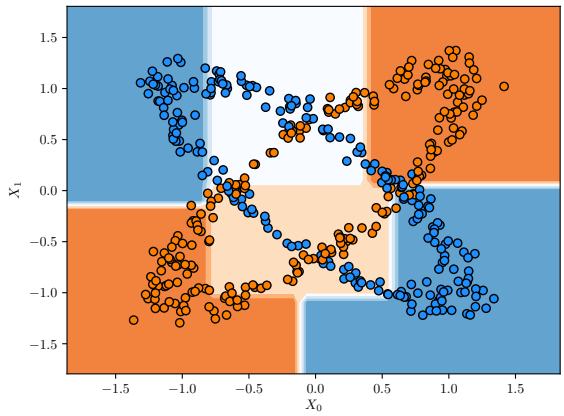
Regarding the results of dataset 1 presented in figure 1, we observe that, for a depth of 1, we have a very poor classification of the data (only one splitting is performed). For a depth of 2, we have a better decision boundary that is composed of 3 zones (the maximum for that depth would be 4). As the depth increases, the results improve. The areas are refined and become more precise but, as will be explained later, we begin to see overfitting. It is worth noting that the maximal possible number of zones for the decision boundary is 2^{depth}, which is not observed for any depth above 1 for this dataset.



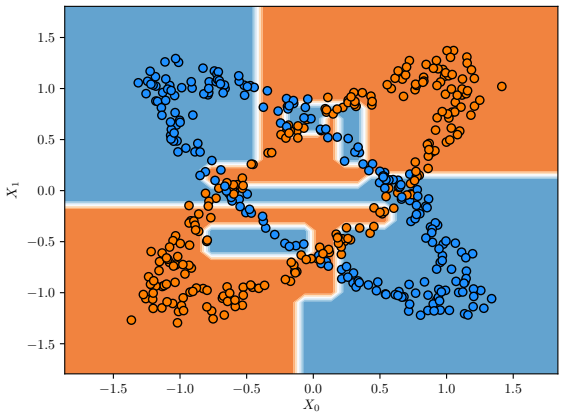
(a) Decision tree : depth of 1 (acc = 0.49)



(b) Decision tree : depth of 2 (acc = 0.55)



(c) Decision tree : depth of 4 (acc = 0.79)



(d) Decision tree : unconstrained (acc = 0.83)

Figure 2 – Decision tree boundary for several depths using `make_data2`

Concerning dataset 2, the results are very similar to those of dataset 1. For shallow depths (1 and 2), the decisions are not very good : the borders are too simple to classify the data correctly, and for higher depths (more than 4), one begins to observe the overfitting.

One difference with the first dataset is that the decision boundary is more complex in this case. This is certainly due to the fact that the ellipses of dataset 1 are parallel to the axes and those of dataset 2 are not, they cross the main axes at roughly 45 angle. Given the operation of the algorithm described previously (the splitting of the features space is performed parallel to the axes), the boundary must be more complex and is less satisfactory than for the first dataset.

For both datasets, a tree of depth 4 seems to be the best classification model. It is certainly the model that lies between the limits of underfitting and overfitting.

(b) *Underfitting and overfitting*

For both datasets, underfitting is clearly observed for a depth of 1 : a single two-zone delimitation is not sufficient to classify an acceptable number of data correctly. The very low accuracy compared to other depths is also evidence for the underfitting. For a depth of 2, the models

improve. They still are not accurate enough, but we begin to have a good accuracy for the first dataset, although too many orange points lie in the blue zone (the decision for dataset 1 is only based on the value of one feature for this depth, which is not a good thing as the other feature is also correlated to the output). The boundary is still too simplistic for that depth to be used in practice.

For dataset 1, greater depths show better accuracies but, for depths of 8 and above, we clearly see some overfitting : the model has many small specific regions to try to have as good an accuracy on the training set as possible.

For the second dataset, we also begin to see overfitting for depths above 4, with many small regions appearing and feeling "forced", non natural, as for the first dataset. As said before, a depth of 4 seems to lie between over- and underfitting.

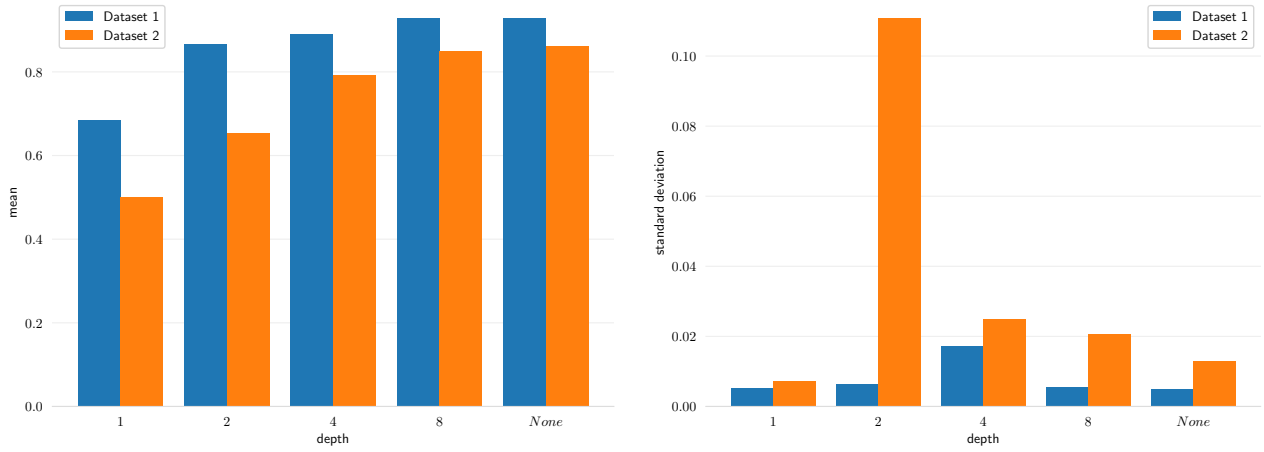
NB : We plotted the errors for the training and testing sets as the depth increases, as shown during the lectures, to see where we have under- and overfitting. However, the testing set errors did not grew after a given point, they rather plateaued after a depth of 5 for the first dataset and a depth of 7 for the second, so we did not include these plots in our analysis.

(c) *Why the model seems more confident when the depth is unconstrained ?*

Because it perfectly classifies the training set (indeed, it can grow to the point where it only has one element per leaf). As the decision tree predicts for each zone the proportion of training objects that belong to that zone and as the zones are perfectly pure (only one class in them), the model is perfectly confident, although it usually overfits a lot.

2. Average and standard deviation of test set accuracies

The accuracies of the model for different depths were computed over 5 generations. The averages and variances of these accuracies are presented in figure 3.



(a) Average

(b) Standard deviation

Figure 3 – Statistics about test set accuracies over 5 generations

For the mean, the conclusion is the same for both dataset, although the overall accuracy is better for dataset 1, with the difference between the two datasets being greater for smaller

depth (explained by the fact that a simple "axis-following" classification yields better results for dataset 1 than dataset 2 as a consequence of the positions of the ellipses). The mean accuracy increases greatly as the depth increases, reaching high accuracy levels for the training set, but leading to overfitting, as was discussed above. We also see no increase in accuracy going from a depth of 8 to unconstrained depth, in either cases. That depth is thus almost optimal (although it leads to some overfitting).

For the standard deviation, the most notable thing is the huge spike for a depth of 2 for the second dataset. As, for that case, no classification comes close to a good one, it is not surprising that the positions of the training set points impact greatly the decision boundary and thus the overall accuracy. The other values show that, in this case, the algorithm is not really biased by the training set. The accuracy will not variate much depending on the 150 samples chosen as training set.

3. Differences between the two datasets

The first dataset is composed of points that belong to two canonical ellipses (each has their major axis following one of the features space axis).

The second dataset is composed of points that belong to the same two ellipses rotated by a 45° angle, which makes it much more difficult to be well classified by an algorithm such as a decision tree, which splits the features space in vertical and horizontal lines (the lines follow the two feature axes) in order to make a classification.

The score (accuracy) of the algorithm on such a dataset is therefore worse than the score on the first dataset, which is composed of points belonging to unrotated ellipses, which makes it easier to find vertical and horizontal lines to split the features space in an effective way.

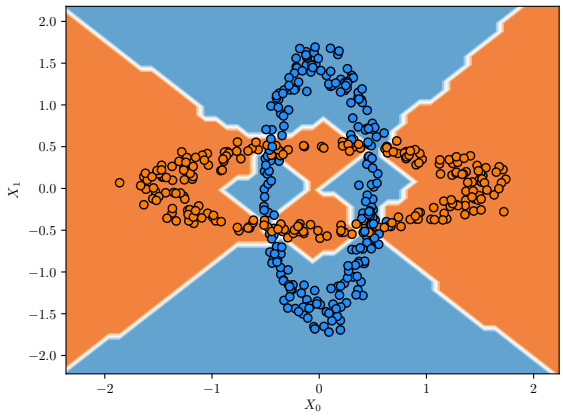
2 K-nearest neighbors

1. Effect of the number of neighbors on the decision boundary

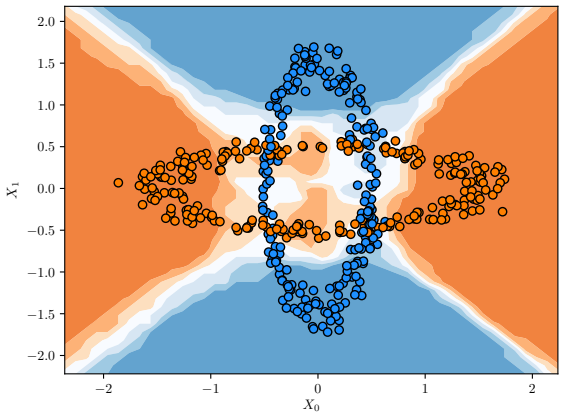
(a) *Illustrate the decision boundary for each number of neighbors*

We built several k-NN models with different numbers of neighbors, for both datasets. To illustrate our results, we present the decision boundary of each model at figures 4 (for dataset 1) and 5 (for dataset 2).

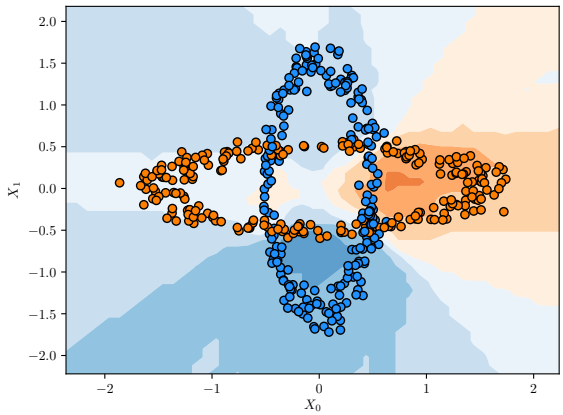
In order to make the visualization clearer, we only displayed 25% of the points of each testing set and only decided to show the most interesting plots, the other being easily interpolated from the graphs shown.



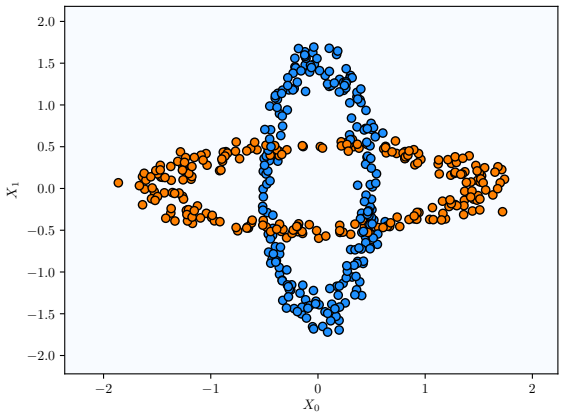
(a) k-NN : 1 neighbor (acc = 0.94)



(b) k-NN : 10 neighbors (acc = 0.86)

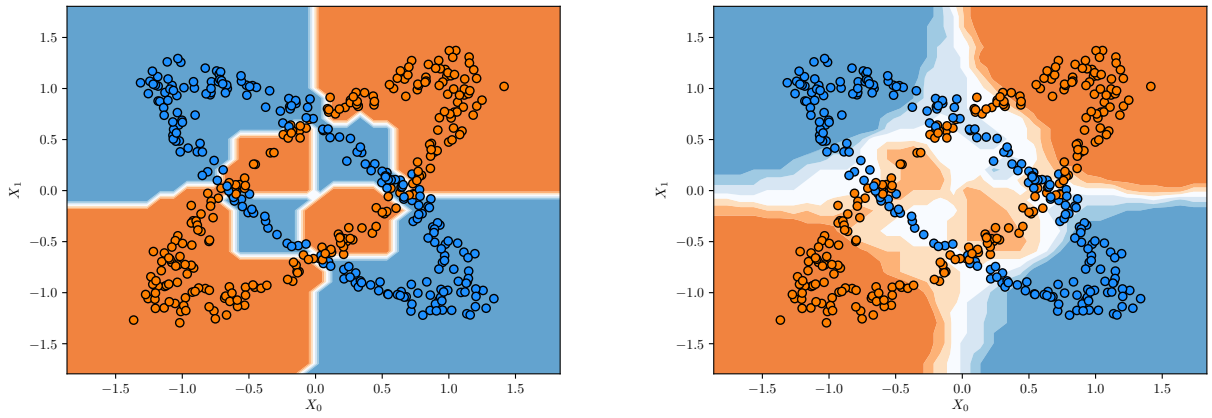


(c) k-NN : 75 neighbors (acc = 0.75)



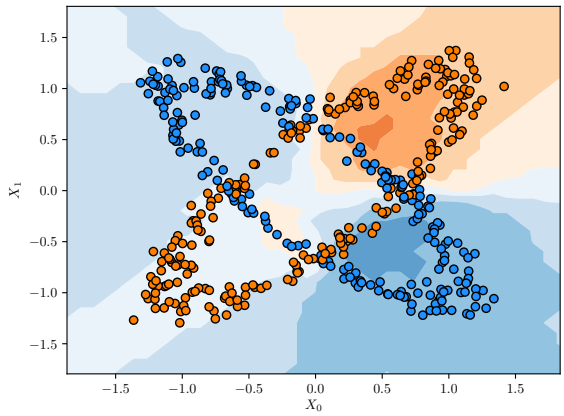
(d) k-NN : 150 neighbors (acc = 0.5)

Figure 4 – k-NN boundary for several number of neighbors using `make_data1`

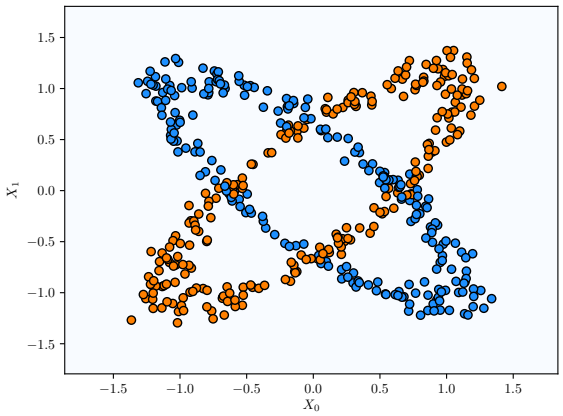


(a) k-NN : 1 neighbor (acc = 0.94)

(b) k-NN : 10 neighbors (acc = 0.86)



(c) k-NN : 75 neighbors (acc = 0.75)



(d) k-NN : 150 neighbors (acc = 0.5)

Figure 5 – k-NN boundary for several number of neighbors using `make_data2`

(b) *Evolution of the decision boundary with respect to the number of neighbors*

The more neighbors, the less confident the algorithm seems to be about what to classify the sample. It is easily understood as, for one neighbor, the certainty of being "right" is of 100%, whereas, for a greater number of neighbors, 100 for example, the probability can easily reduce as low as 50% depending on the position of the sample. That is represented by a decision boundary of a pale color. The more neighbors, the more uncorrelated data you take into account in your decision (they are far away from the sample position and should not have a great influence on the decision).

For 150 neighbors, all the space is whitish as there are almost as many points from the dataset 1 as from the dataset 2 in the learning sample used to establish the probabilities (there are a bit more blue points, which makes the boundary slightly bluish). Every testing sample will therefore have the same neighbors (the training sample is composed of 150 points), and so the whole space has a uniform decision boundary.

The paler the color, the more uncertain the model is about its prediction. As the two datasets show two "symmetric" axes (some sort of diagonals), these are white from the start, and expand as the number of neighbors increases.

In both cases, for a small number of neighbors (1 and 10), the classifications seem correct but quite too specific (1 neighbor is a case of overfitting, 10 neighbors is smoother). For a larger number of neighbors (75 or more), the classification is degraded (case of underfitting) : the high number of neighbors allows the algorithm to use a larger series of points irrelevant for the zone, thus distorting the probabilities.

2. Ten-fold cross validation

(a) *Explain your methodology*

This technique consists in sampling our whole dataset in 10 samples. One sample is chosen as the testing set and the other 9 are used at learning sets.

The score (accuracy) is then computed and the operation is repeated by taking another testing set among the samples not yet used as such.

The operation is therefore repeated 9 times so that each sample is used once as testing set. The 10 computed scores are averaged to obtain the score for that number of neighbors.

As part of our project, we used the function `cross_val_score` of the `sklearn` library : it returns a scoreboard that contains the score of each of the 10 subsets. We computed the mean of this scoreboard in order to have the score of this strategy. That score was then used to find the optimal number of neighbors, as described below. The estimate is unbiased because it does not depend anymore on 150 specific samples, but takes into account the whole dataset.

(b) *Score and optimal number of neighbors to use*

To determine the optimal number of neighbors to use, we tested the ten-fold cross validation strategy on several number of neighbors (from 5 to 150 with a step of 1) and computed the score for each. These scores are shown in figure 6.

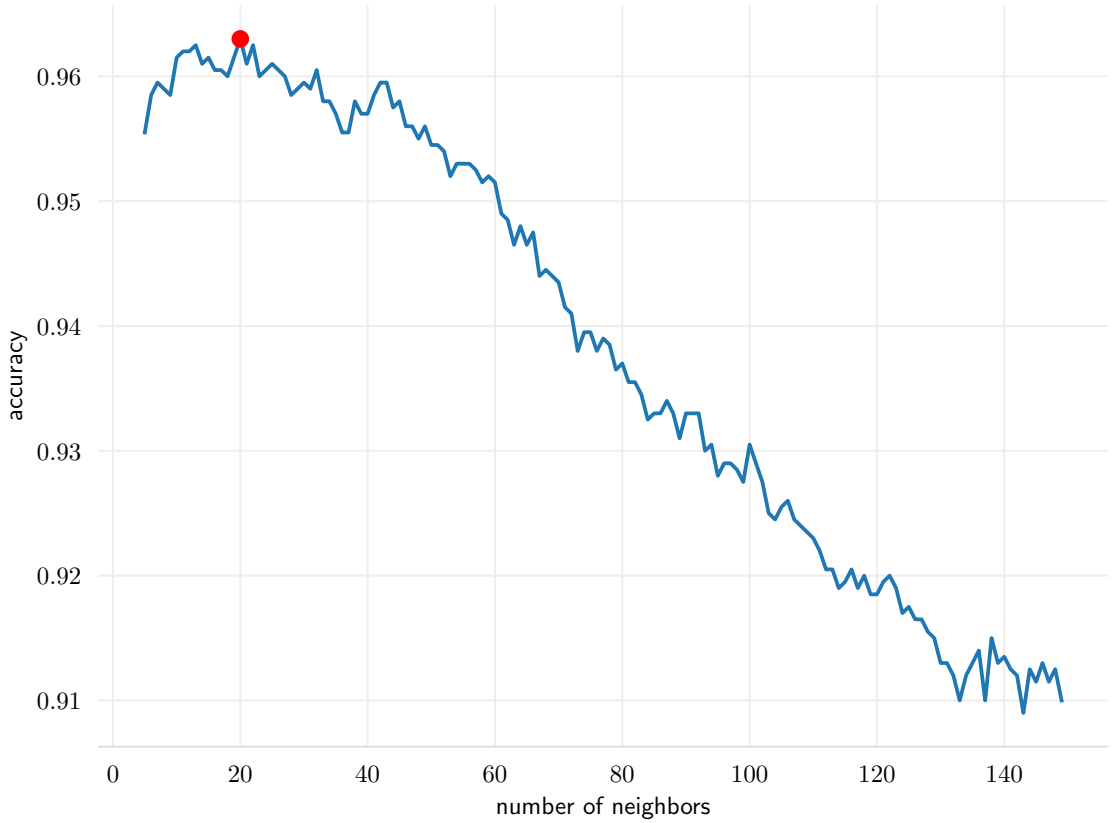


Figure 6 – Scores obtained for multiple value of number of neighbors with k-fold strategy on dataset 2

The scores obtained show us that the optimal number of neighbors to take is 20. This number provides an accuracy of 0.963.

This result is in agreement with our intuition : an optimal number of neighbors is not very high so as not to take into account too many specific or isolated points but is not too small so that the decision is not dictated by a very small group of points that could be noise or make the strategy local, blind to the bigger picture.

The decision boundaries also confirm this intuition : as seen on figure 5, the optimal value was between 1 (underfitting) and 75 (overfitting).

NB : It is worth noting that the k-cross test validation was performed on the whole dataset (2000 samples), and not on the 150 training samples alone. This explains why the accuracies given in this section are much higher than the ones given in the plots of the first section.

3. Optimal value on the first dataset

We believe it would fare almost in the same way as it did for the second dataset, as the ellipse dispositions are almost the same in each dataset, and as k-NN is rotational invariant. Indeed, the distance between 2 objects is rotational invariant.

To confirm our hypothesis, we made the test for the first dataset and found out that, indeed, 20 neighbors is optimal.

3 Naive Bayes classifier

1. The Naive Bayes classifier prediction

Let y be the class variables and $\boldsymbol{x} = (x_1, x_2, ...)$ the dependent input variables.

Bayes' theorem states that

$$P\left(y|x_1, \ldots, x_n\right) = \frac{P(y)P\left(x_1, \ldots x_n|y\right)}{P\left(x_1, \ldots, x_n\right)}$$

Knowing that

$$P\left(x_1, \ldots, x_n|\, y\right) = \prod_{i=1}^{n} P\left(x_i|y, x_1, \ldots, x_{i-1}\right)$$

and that the x_i are conditionally independent given the output (the *naive Bayes independance assumption*), *i.e.*

$$P\left(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\right) = P\left(x_i|y\right)$$

we can write

$$P\left(y|x_1, \ldots, x_n\right) = \frac{P(y)\prod_{i=1}^{n} P\left(x_i|y\right)}{P\left(x_1, \ldots, x_n\right)}$$

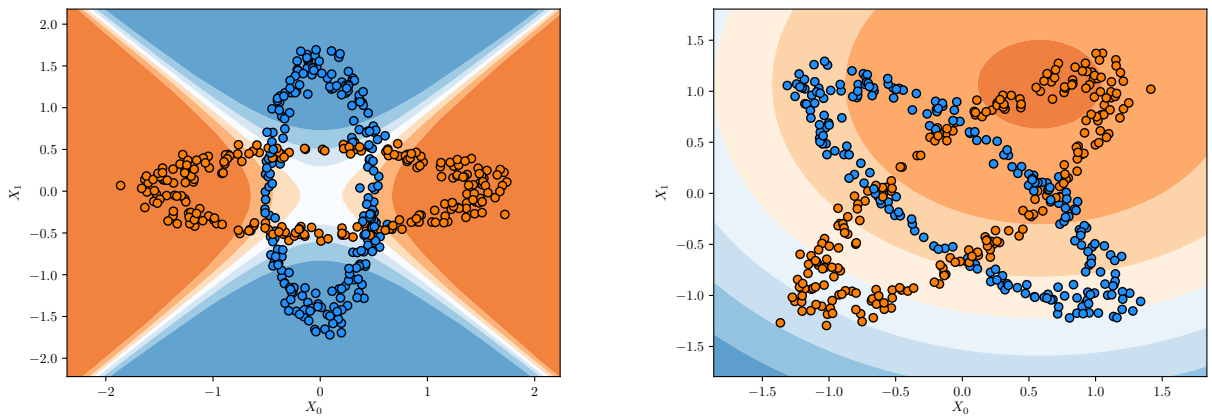
The denominator being constant given y, we can finally write

$$\begin{aligned}\hat{f}(\mathbf{x}) &= \operatorname{argmax}_y \mathrm{P}\left(y|x_1, \ldots, x_n\right) \\ &= \operatorname{argmax}_y \mathrm{P}(y) \prod_{i=1}^{n} \mathrm{P}\left(x_i|y\right)\end{aligned}$$

3. Testing set accuracy

We used our own naive Bayes estimator on both datasets. To illustrate our results, we present the decision boundary of each dataset at figure 7.

In order to make the visualization clearer, we only displayed 25% of the points of each testing set.



(a) Dataset 1 (`make_data1`)

(b) Dataset 2 (`make_data2`)

Figure 7 – Decision boundary for Naive Bayes classifier algorithm for both datasets

The accuracies computed for each testing set are presented in table 1.

Dataset 1	**Dataset 2**
0.7978	0.5535

Table 1 – Testing set accuracy for both datasets.

Based on figure 7, we see that the estimator gives satisfactory results for dataset 1 but poor results for dataset 2. This finding is reinforced by the computed scores : not far from 80% accuracy for dataset 1 against just above 50% for dataset 2.

One reason for that might be that the "mixing zone" (*i.e.* the central rhombus in which the two classes of points are present) is bigger for dataset 2, and so the probability densities of the two classes are not as clearly separable as for dataset 1. If we look closely at the statisical values of the two datasets, we see that the two classes of the second dataset have almost the same means and almost the same variances along the 2 axes, whereas the variances for the first dataset are much more different, and so the naive Bayes classification works better on the first dataset. Indeed, the choice to presume Gaussianity for the likelihood makes it so that the values of the means and variances for the two classes impact greatly the accuracy of the algorithm. Having them too close leads to poor performances.