ELEN0062-1 Introduction to machine learning

Project 1 - Classification algorithms

Authors:
Elodie Jamotton
Joachim Paquay

Teachers:
P. Geurts
L. Wehenkel
Assistant:
J-M. Begon

Academic year 2019-2020

1 Decision tree (`dt.py`)

1.1 For both datasets, observe how the decision boundary is affected by tree depths

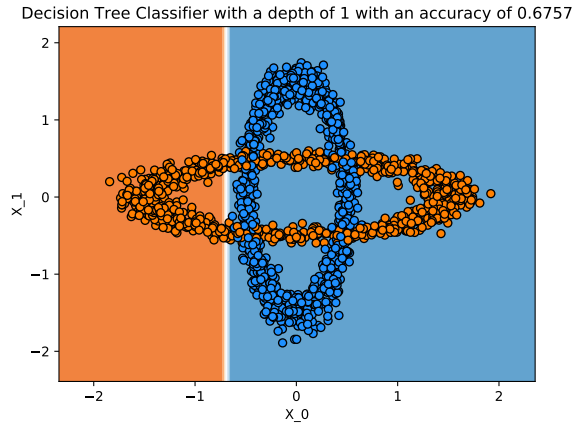
The first dataset is created based on two perpendicular ellipses which major axes have rotational angles of 0 and π radians compared to the x_0 axis of the graph. These ellipses are clearly visible in the figures representing the data and their classification, they are gathered in figure 1. There are different graphs corresponding to the different values of the decision tree depth.

The second dataset is also created based on two perpendicular ellipses but now, their major axes have rotational angles of $\frac{\pi}{2}$ and $\frac{3\pi}{4}$ radians compared to the x_0 axis of the graph. This second dataset and its different classifications are shown in figure 2.

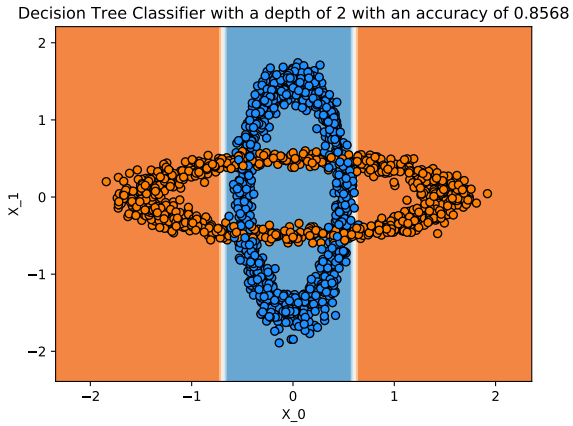
For both datasets, one can see that the classification with a decision tree of only one depth is not sufficient at all. Indeed, the points' domain is divided, vertically or horizontally, in two parts and the classification is made with only one feature. It is then clear that data are underfitted.
When the decision tree has a depth of two, a second vertical line separates a subclasse in two parts for the first dataset. For the second one, an additional horizontal line can be seen but also a vertical one. However, the data with a value of x_1 in the rough interval $]-1, 1]$ seem to have a random class.
By increasing more and more the decision tree depth, the number of lines separating the two classes (orange and blue ones) increases till reaching a maximum number of separations that can be observed in figures 1e and 2e for the first and second datasets respectively. Moreover, increasing the depth also improves the data fitting which decreases the global error but going too deep in the decision tree may lead to overfitting. Overfitting means that the error due to the unseen points begins to increase and misclassifications appear.

In the case of the first dataset, classification boundary is not clearly distinguishable because of the data plot, this is why the latter are not represented in figures 1d and 1e. It is clear that decision trees of depth equals to one, two and four are underfitting the data as a large part of the orange points are in the blue class. For the two last depths, only a small region in the data domain differs from each other. This region corresponds to a junction of the two classes of points which means that classification is harder to predict there. For depth equals to 8, the class prediction can't be clearly seen in the graph while the boundary is well defined for an unlimited depth. As in this region data are not well separated, the figure which represents a decision tree without imposed depth overfits the data while the one which represents a decision tree with a depth of 8 is the best case represented in this exercise.

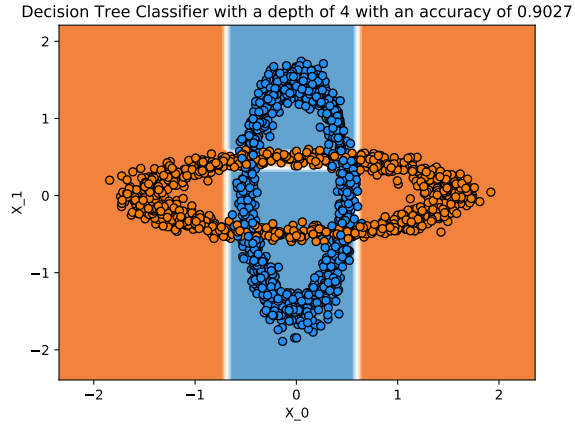
In the case of the second dataset, the decision boundary is more distinguishable so the data points are always plotted. For a depth equals to 8, data have been already overfitted as a thin orange part taking a few orange points but also blue points is in the figure 2d. Underfitting cases are clearly represented for depths equal to 1 and 2, and the best represented case is the decision tree with depth equals to 4.



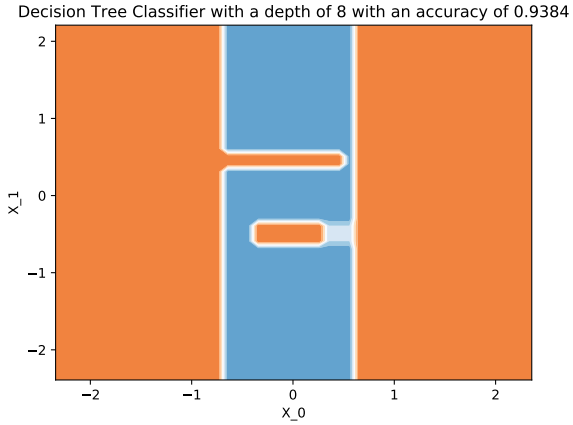
(a) Depth = 1



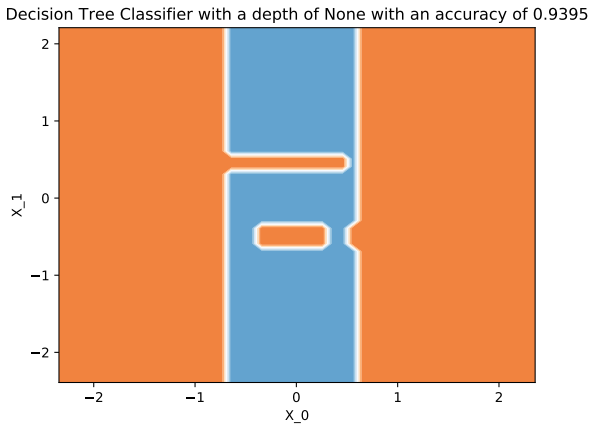
(b) Depth = 2



(c) Depth = 4

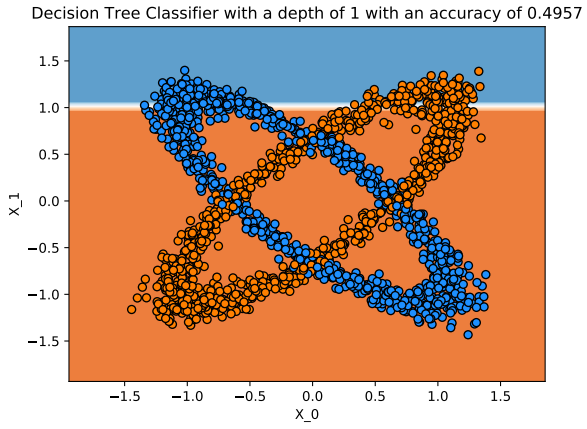


(d) Depth = 8

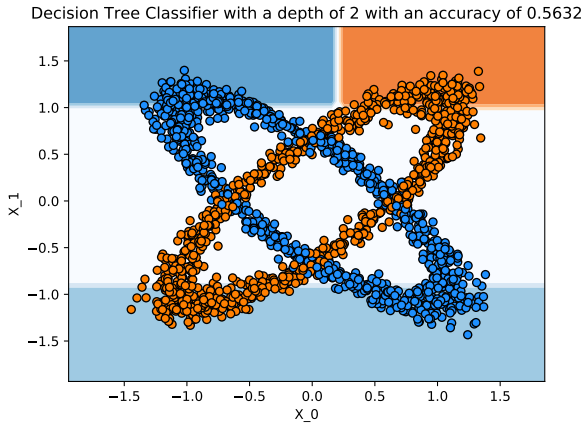


(e) No depth

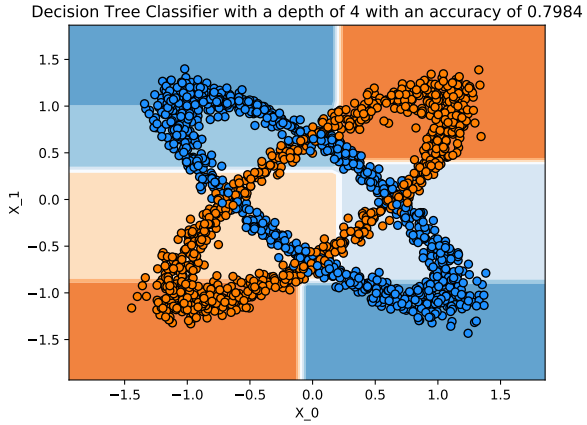
Figure 1: Dataset 1 and its classification for different values of decision tree depth



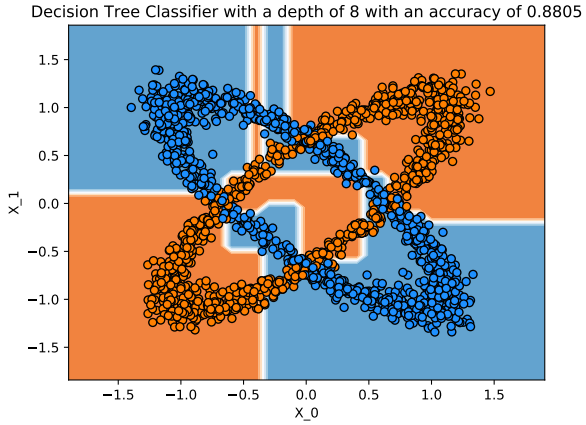
(a) Depth = 1



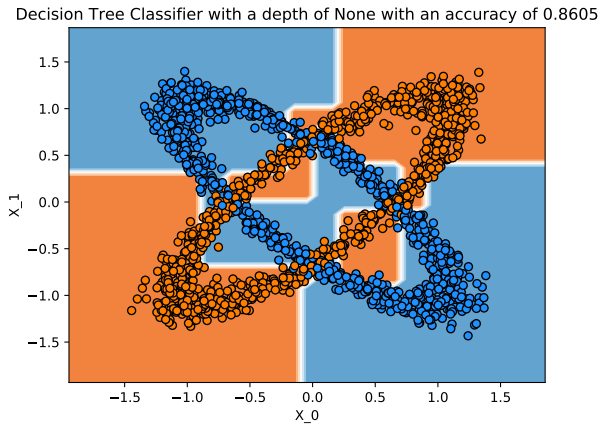
(b) Depth = 2



(c) Depth = 4



(d) Depth = 8



(e) No depth

Figure 2: Dataset 2 and its classification for different values of decision tree depth

1.2 Report the average test set accuracies (over five generations of the dataset) along with the standard deviations for each depth. Briefly comment on them

For both datasets, the average accuracy increases with the decision tree depth increasing. Regarding the standard deviations, they do not follow a regular variation but the number of generations may not be sufficient to make conclusions. Accuracies and standard deviations asked are in table 1.

Depths		1	2	4	8	None
Accuracies	Dataset 1	0.6795	0.8663	0.8844	0.9237	0.9239
	Dataset 2	0.4952	0.6715	0.7499	0.8503	0.8551
Standard deviations	Dataset 1	0.005	0	0.0145	0.0086	0.009
	Dataset 2	0.0049	0.1114	0.1101	0.0376	0.0331

Table 1: Accuracies and standard deviations for each decision tree depth and for each dataset

1.3 Base on both the decision boundaries and the test accuracies, discuss the differences between the two datasets

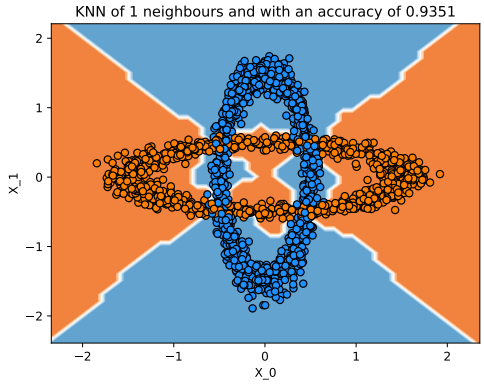
The datasets creation is explained in the section 1.1 and it seems that the simple rotation of the ellipses changes the classification and the prediction of data. Indeed, one can see that the decision boundaries of the first dataset are easier to draw than the one of the second dataset as there exist more "white" zones in the figure 2 that shows the difficulty of prediction in these zones where blue and orange points are mixed together. It is easy to understand that, as the classification is made thanks to the values of x_0 and x_1, ellipses with axis having a unique value of one of these two variables are easier to divide in different parts.

In terms of accuracies, the same observation can be done as values for the first dataset are higher than for the second one.

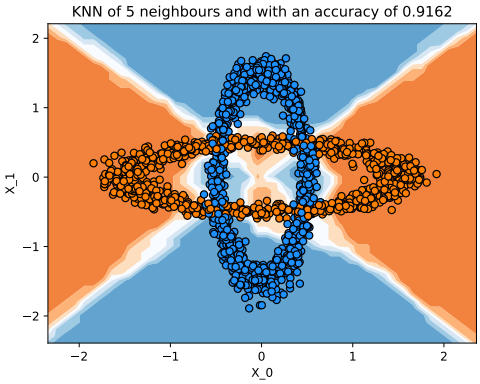
2 K-nearest neighbours (`knn.py`)

2.1 For both datasets, observe how the decision boundary is affected by the number of neighbours

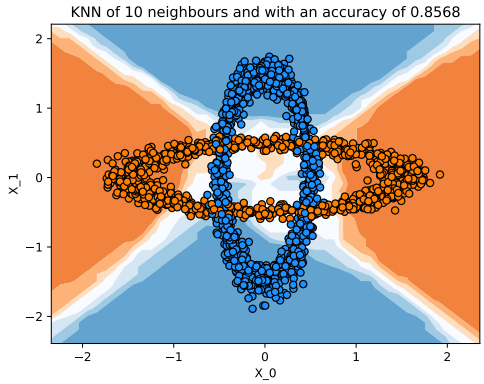
The implementation of the K-nearest neighbours is in the file `knn.py` and the graphs obtained with it are represented in figures 3 and 4.



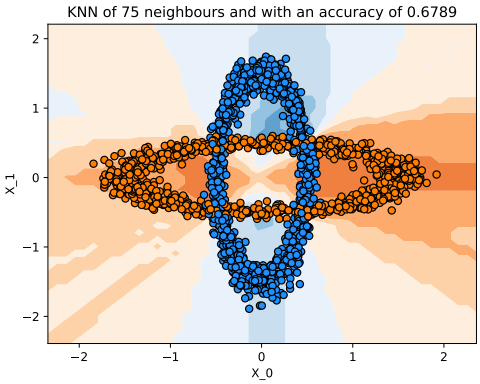
(a) Number of neighbours = 1



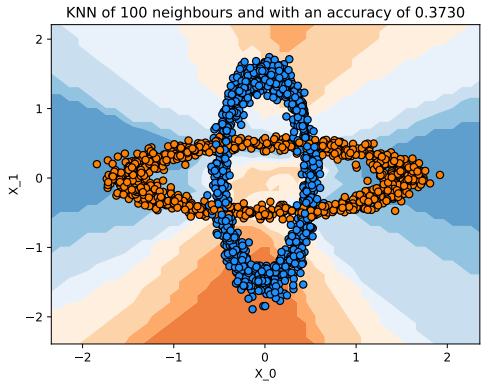
(b) Number of neighbours = 5



(c) Number of neighbours = 10

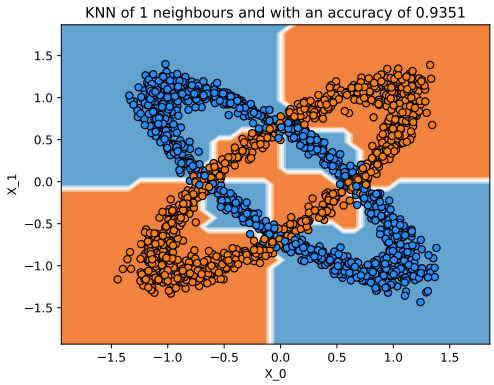


(d) Number of neighbours = 75

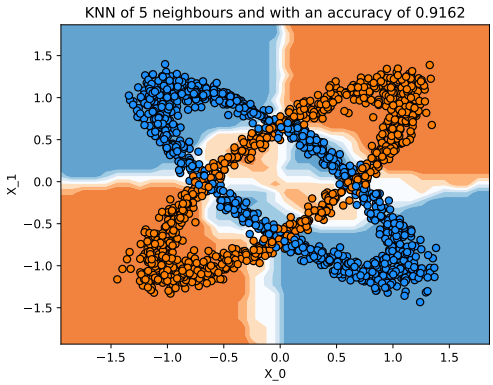


(e) Number of neighbours = 100

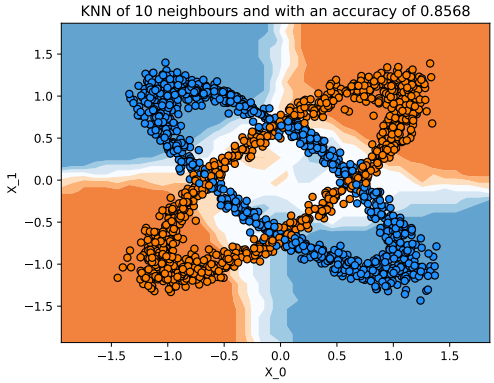
Figure 3: Dataset 1 and its decision boundary for different number of neighbours



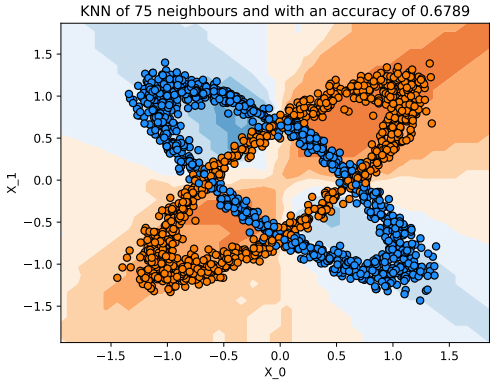
(a) Number of neighbours = 1



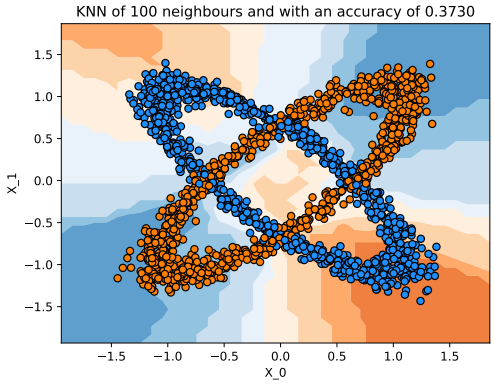
(b) Number of neighbours = 5



(c) Number of neighbours = 10



(d) Number of neighbours = 75



(e) Number of neighbours = 100

Figure 4: Dataset 2 and its decision boundary for different number of neighbours

The figures 3 and 4 show the different decision boundaries with respect to the number of neighbours.
As one can see both datasets have the same general behaviour. Indeed, it is clear that for 1 and 5 neighbours, this is a situation of overfitting because colours are following too much the ellipse shape.

One can observe that for 10 neighbours this is almost good because, in the middle of the graph, there are more withe zones. However, a small overfitting is still present because some blue and orange zones are following the ellipses in the centre of the graph.

Now, 75 and 100 neighbours are clearly underfitting cases because there are orange points in blue zone and blue points in orange zone, which translate the characterisation of underfitting.

So, one can deduce that for the present case, i.e. a dataset of 2000 samples and a training set of 150 samples, the optimal number of neighbours will be between 10 and 75 but closer to 10 than to 75.

2.2 Use a ten-fold cross validation strategy to optimise the value of the `n_neighbors` parameter and obtain an unbiased estimate of the test accuracy for the second dataset

2.2.1 Explain your methodology

In order to use a ten-fold cross validation strategy and find the optimal value of $n\_neighbors$, the strategy was simply to loop from 1 to $n\_neighbors$ and for each iteration of the loop to call the class $KNeighborsClassifier$ with a certain number of neighbours (according to the loop). Then, the function named $cross\_val\_score$ is used to compute the accuracies of the ten-fold cross validation and the mean of these accuracies is computed for each different number of neighbours. By determining the best average accuracy, the optimal number of neighbours is found.

The $cross\_val\_score$ function is perfect to do a K-fold cross validation strategy because this strategy consists in separating the dataset into K sections then use one of the K sections as training subset and the remaining K-1 section as testing subset in order to compute a score (here the accuracy). And in order to have a mean accuracy, this operation is restarted for all the K sections (i.e each K section will be used as a training set at some point) and then the mean accuracy over the K accuracies can compute.
The $cross\_val\_score$ function will generate an array containing all the accuracies for the K sections in the same way as said above.

2.2.2 Report the score you obtain and the optimal value of `n_neighbors`. Do they corroborate your decision boundary-based intuition? Justify your answer

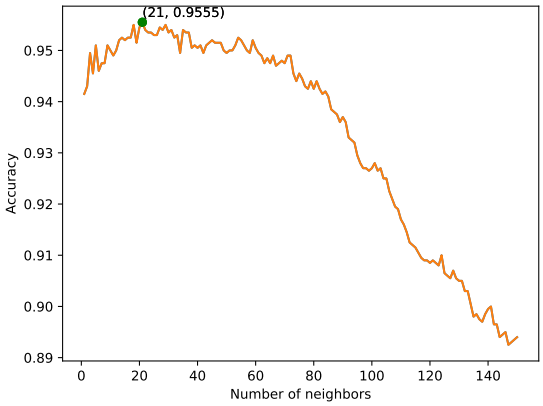


Figure 5: Evolution of the accuracy depending on the number of neighbours

One can see in the graph 6 that the best number of neighbours is 21 and it comes with an accuracy of 95.55%. As expected, the optimal number of neighbours is between 10 and 75 and is closer to 10 than to 75. Indeed, the discussion of the graph 4c is the same as in section 2.1. To summarise, there is a little overfitting for 10 neighbours (because there are some orange and blue zones in the middle of the graph) but it is really close to a good solution so there is no surprise that the optimal number of neighbours is 21.

2.3 How do you think would fare this optimal value on the first dataset?

In order to know the optimal value for the first dataset, the ten-fold cross validation strategy was used over the first dataset. However, as the graphs are fairly the same (figures 3 and 4), it is logical to have fairly the same results as the second dataset when doing a ten-fold cross validation.

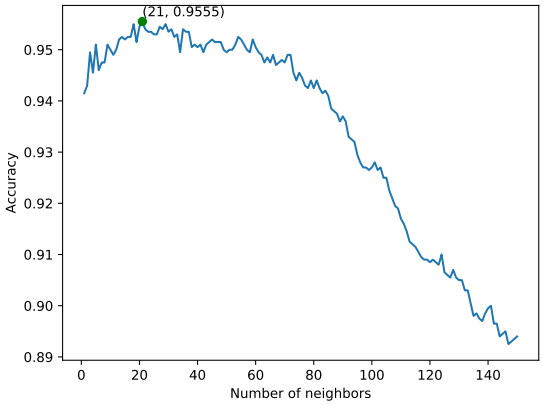


Figure 6: Evolution of the accuracy depending on the number of neighbours

3 Naive Bayes classifier (`naive_bayes.py`)

3.1 Demonstration of equivalence

First, it can be proven that

$$\hat{f}(x) = argmax_y Pr(y|x_1, ..., x_p) \tag{1}$$

is equivalent to

$$\hat{f}(x) = argmax_y Pr(y) \prod_{i=1}^{p} Pr(x_i|y) \tag{2}$$

under the assumption of

$$P(\mathcal{X}_i|\mathcal{Y}, \mathcal{Z}) = P(\mathcal{X}_i|\mathcal{Y}) \qquad \forall \mathcal{Z} \in \mathcal{P}(X_1, ..., X_{i-1}, X_{i+1}, ..., X_p) \tag{3}$$

If one takes the part after the *argmax* from the equation (1) and if Bayes is used, one has:

$$Pr(y|x_1, ..., x_p) = \frac{Pr(y)Pr(x_1, ..., x_p|y)}{Pr(x_1, ..., x_p)} \tag{4}$$

Now if only the numerator is taken from (4) and if it is developed thanks to (3), one has:

$$\begin{aligned} Pr(y)Pr(x_1, ..., x_p|y) &= Pr(y)Pr(x_1|y)Pr(x_2|y, x_1)...Pr(x_p|y, x_1, ..., x_{p-1}) \\ &= Pr(y)Pr(x_1|y)Pr(x_2|y)....Pr(x_p|y) \end{aligned} \tag{5}$$

Now if (5) is injected into (4), one obtains:

$$Pr(y|x_1, ..., x_p) = \frac{Pr(y)Pr(x_1|y)Pr(x_2|y)....Pr(x_p|y)}{Pr(x_1, ..., x_p)} \tag{6}$$

Finally one can say that

$$\begin{aligned} (1) &= argmax_y \frac{Pr(y)Pr(x_1|y)Pr(x_2|y)....Pr(x_p|y)}{Pr(x_1, ..., x_p)} \\ &= argmax_y Pr(y)Pr(x_1|y)Pr(x_2|y)....Pr(x_p|y) = (2) \end{aligned} \tag{7}$$

and the equivalence is demonstrated.

3.2 Implement your own naive Bayes estimator according to the above description and following the scikit-learn convention

The implementation of a naive Bayes estimator is in the file `naive_bayes.py` and the graphs obtained with it are represented in figures 7 and 8.

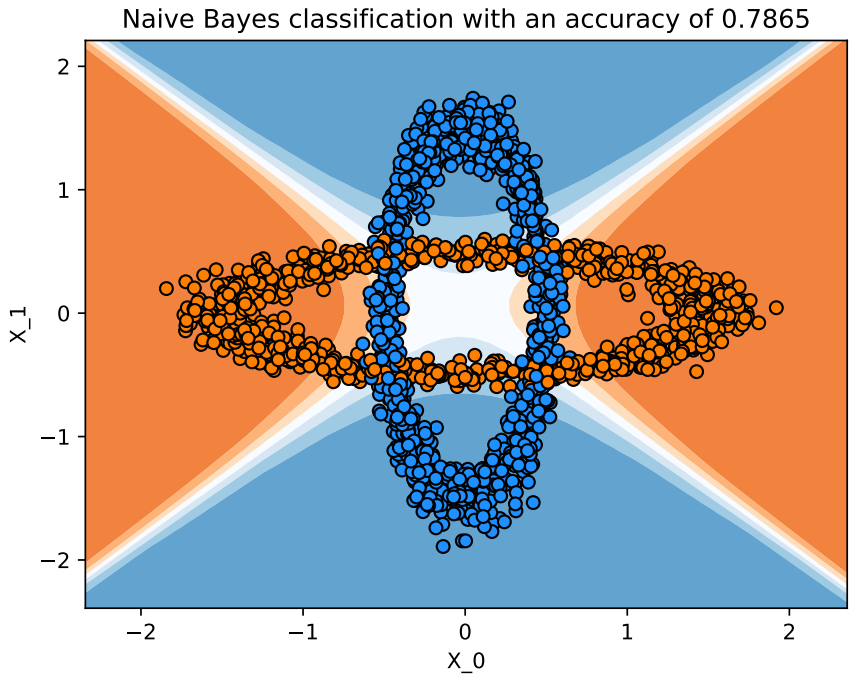


Figure 7: Dataset 1 and its classification made by a naive Bayes estimator

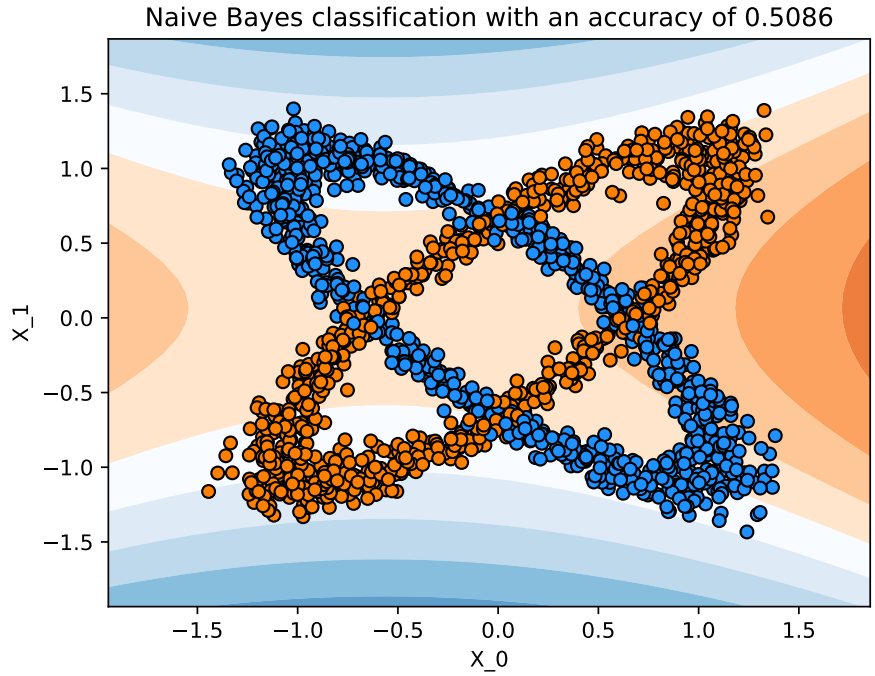


Figure 8: Dataset 2 and its classification made by a naive Bayes estimator

3.3 Compute the testing set accuracy on both datasets and interpret the results

The same conclusion as with the decision tree classifier can be made here. Indeed, the accuracy for the first dataset is higher than for the second one for the same reasons as in section 1.3. The accuracies for each dataset are in table 2.

	Dataset 1	Dataset 2
Accuracies	0.7865	0.5086

Table 2: Accuracies for each dataset