LIÈGE université

Université de Liège

ELEN0062-1
Introduction to machine learning

Project :
Classical algorithms

DEBOR Antoine
HELD Jan

Academic year 2020-2021

1 Decision Tree

1.1

a. The decision boundary of the decision tree for the first data set and for maximal depth values of 1,2,4,8 and `None` is shown in the following figures.

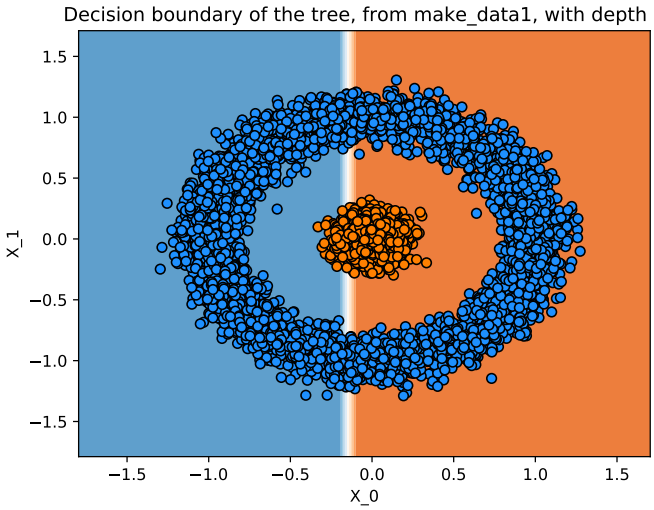

FIGURE 1

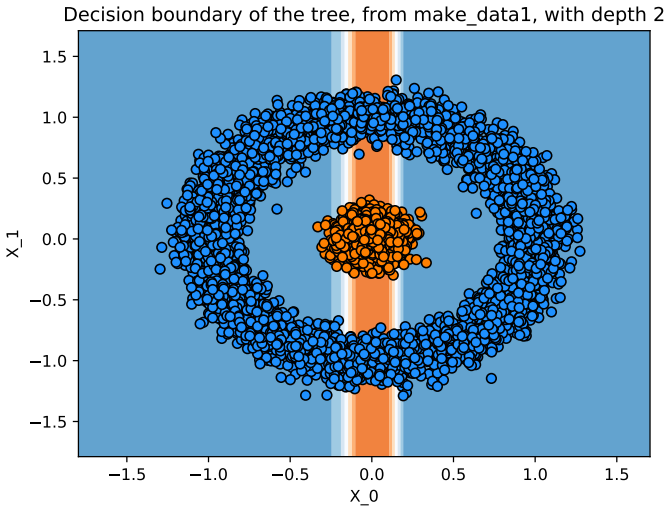

FIGURE 2

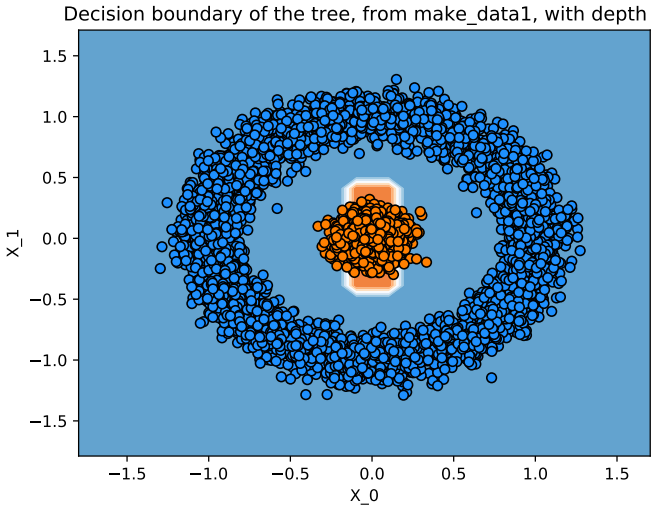

FIGURE 3

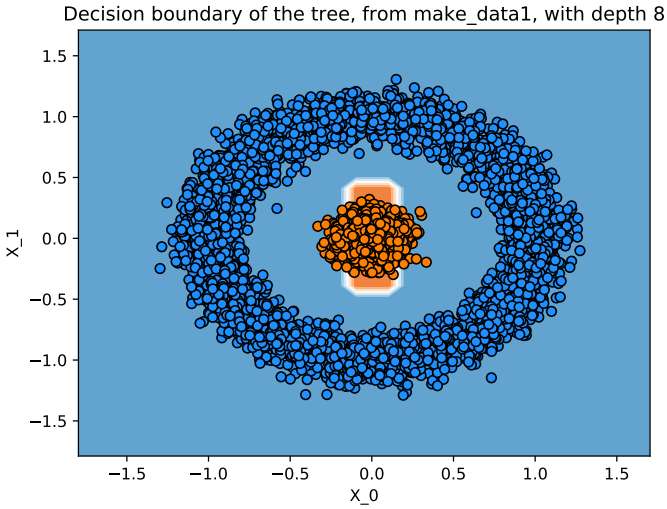

FIGURE 4

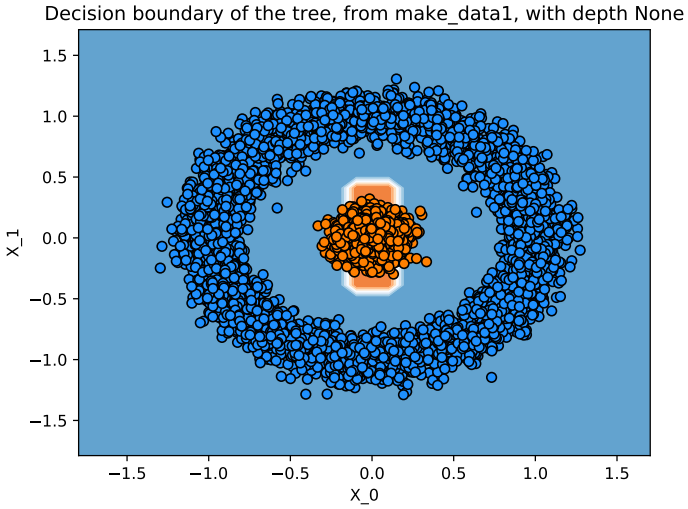

FIGURE 5

The decision boundary of the decision tree for the second data set and for maximal depth values of 1,2,4,8 and `None` is shown in the following figures.

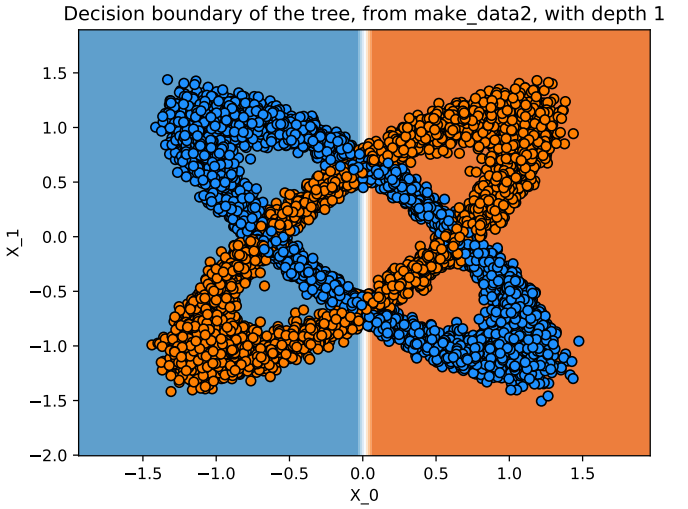

FIGURE 6

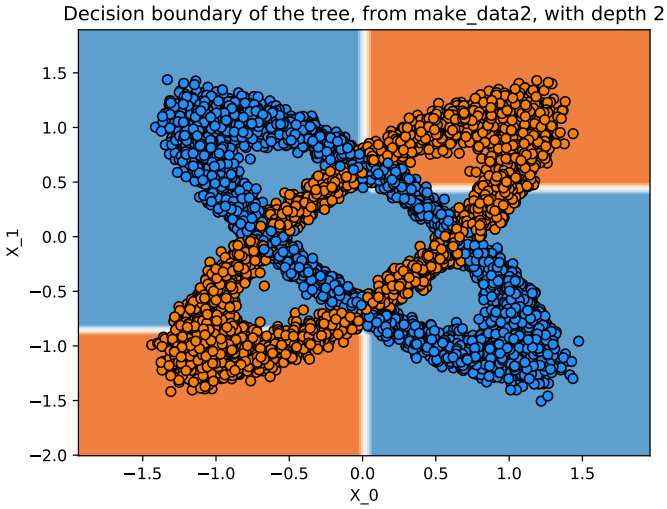

FIGURE 7

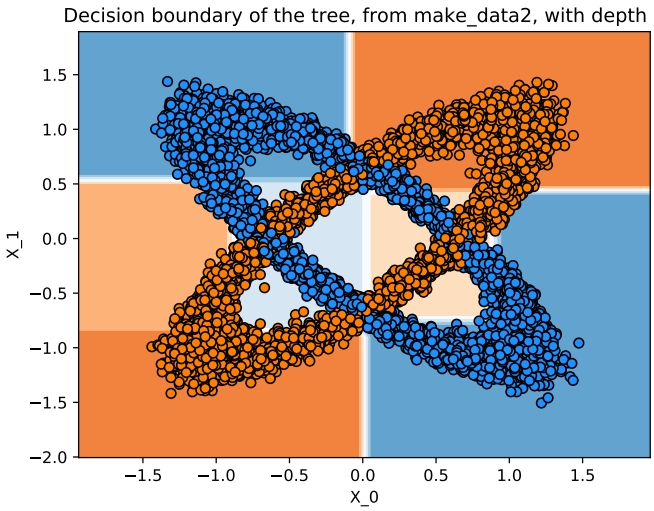

FIGURE 8

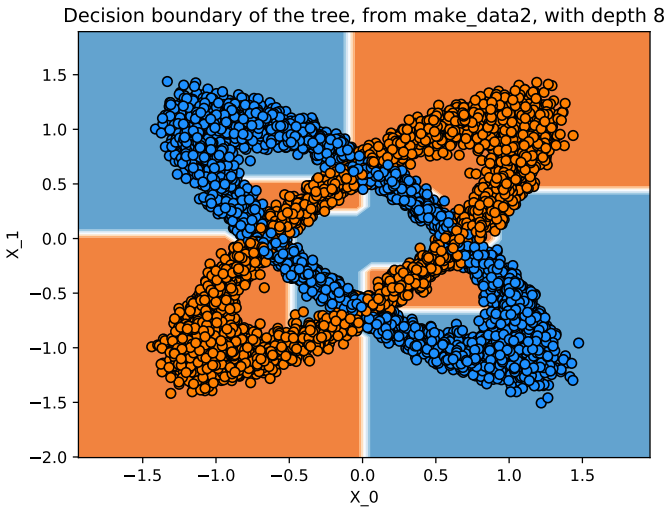

FIGURE 9

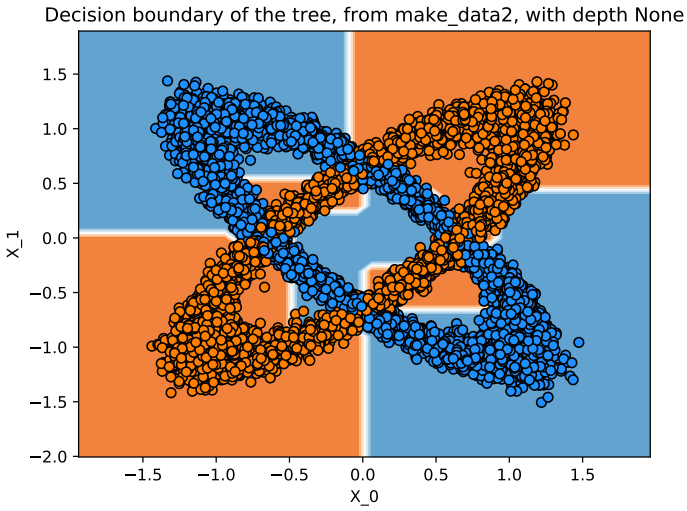

FIGURE 10

- **Depth = 1**

Both data sets :
For a depth of 1, the decision boundary is a vertical line who divides the graph in two halves. The decision tree indeed implements only one decision node and, therefore, only one of the attributes can be discriminating. As it can be seen for both data sets, the attribute X_0 is this discriminating attribute.This model is far from being optimal regarding the TS generalization because a big part of the blue objects are located in the orangeregion and, at least for the second dataset, conversely.

- **Depth = 2**

Both data sets :
By increasing the depth to a value of 2, the decision boundary seems to become more precise, although still crude. Indeed, the decision tree now implements a total of 3 decision

nodes, allowing it to devide the attribute space more precisely by permitting 4 final nodes to be reached.

- **Depth = 4**

First data set :
The model now generalizes pretty good to the TS. Most of the TS objects are located in the right region, which is not really surprising because the TS is now much more split (16 final nodes) which makes the model precise enough for the first data set's "binary" shape.

Second data set :
For the second data set with a depth of 4, one can observe that the decision boundary no longer consists in two distinct regions (blue and red), but also lighter regions. One can construe these regions as regions where the model has "doubts", meaning that the probability of belonging to a particular class in these regions is not of 100%. In the figure 8, one can see that there are a lot of both blue and orange objects in the lighter regions , which justifies the model's lack of assurance considering the not so high depth value. Indeed, at a depth of 4, since the sample can not be more split, there are therefore regions where a prediction will not be accurate.

- **Depth = 8**

First data set :
The decision boundary does not seem to change from the one relative to a depth of 4. Indeed, the LS must be already well fitted with a depth of 4 and the expected refining of the model brought by a depth of 8 (2^8 final nodes) has no impact.

Second data set :
One can see that the lighter regions of the depth-8 model boundary has nearly completely disappeared. Indeed, thanks to the increase in depth, the model is now able to better deal with the central region of the attribute space, which contains both orange and blue objects. However, it is important to notice that this apparent refining concerns the fitting of the LS, and is not relevant for every object when considering the TS.

- **Depth = Unspecified**

If the depth is unspecified, it means that the tree is expanded as much as possible until it perfectly fits the training set.

First data set :
As for the depth value of 8, there is no visible difference in the decision boundary in comparison with the depth-4 case. Again, there is no real refining.

Second data set :
The only difference with the depth-8 case is the replacement of a lighter area by a blue one [1]. Indeed, an unconstrained increase in depth eventually leads to a very self-confident model. Again, it is important to notice that such a behaviour of the model might appear not relevant when facing TS predictions, and in this case the model is going to predict

1. This can only be seen if the LS is plotted in place of the TS.

blue objects in a region (intersection point of the ellipses) where both classes coexist.

b. **Underfitting :**

For the *first data set*, one can clearly see that for a depth of 1 and 2, our model is underfitting. For a depth of 1, a half of the blue objects are predicted wrongly and by increasing the depth to 2, one can see a slight improvement, but, still, there are too many objects which are placed in the wrong region.
These observations remains for the *second data set*.
Therefore, for *both data sets*, if a tree depth of 1 or 2 is used, the performance of the model is expected to be really bad as the predictions won't be very accurate.

NB : For the first data set, one could consider that the model always underfits the data, even for higher depth values than 2. Indeed, a non negligible part of the orange objects are outside of the orange prediction region. However, since this is not as obvious as for the first two depths, those cases are not considered as *clear* underfitting.

Overfitting :

For the *first data set*, the model is never overfitting , no matter the depth of the decision tree. This can be explained by the fact that all the orange objects are densely located in one particular region clearly separated from the blue objects, i.e. without mixing the two classes (there is not a single outlier), and therefore the model can not overfit.
However, for the *second dataset*, one can see that the objects are more scattered. There exist regions in which both classes of objects coexist, in particular intersections of the two ellipses in the attribute space. One can observe that the model is overfitting for a depth of 8 and for an unconstrained depth (figure 9 and 10) . It draws boundaries around the examples it should, but the patterns it finds make no sense (especially in the intersections regions abovementioned).This model fits too good the LS and, therefore, it also takes into account noise and outliers. However, it is important to notice that the main areas where overfitting occurs are predicted to contain orange objects and only a few objects of the TS belonging to the blue class are located in those areas ; one may thus expect the accuracy to be not that much impacted by overfitting.

c. The model seems more confident when the depth is unconstrained because nearly every object of the LS is in the zone corresponding to the class to which it belongs and the probability predictions are not nuanced. Indeed, the model tends to perfectly fit the LS, meaning that it may overfit and appear more confident regarding this very LS since every object tends to be classified with a 100% probability. Again, this behaviour is in general detrimental and eventually leads to bad TS prediction results. In this case, as previously discussed, the overfitting is however expected to have no high impact on accuracy.

1.2

The tables 1 and 2 give the average test set accuracies and standard deviations from the first and the second data sets over 5 generations, for each depth.

For both data sets, one can observe that if the depth of our decision tree increases, the average

accuracy increases too. Despite the overfitting problem depict earlier, this is not really surprising as one may expect the tree to become more precise as the depth increases. One can conclude that the overfitting relative to the second data set sees its harmful effect counter-balanced by an increase in prediction accuracy. As expected, overfitting has here no fatal impact. An unexpected observation however stands, as one could have expected no enhancement in accuracy for the first data set since the decision boundary seemed to remain the same for depth values of 4, 8 and `None`. Still, this enhancement is of only a few percents.

Concerning the standard deviations, one can observe that, except for the depth of 1, they are quite constant over each depth and for both data sets. Still, they are around $10-11\%$ for the first one and 13% for the second one, which is not negligible and shows that the performance of the model varies from a generation of the test set to another.

Depth	Average accuracy	Standard deviation
1	0.7138	0.0
2	0.8229	0.1091
4	0.8763	0.1169
8	0.9031	0.1113
Unspecified	0.9192	0.1046

TABLE 1 – Average accuracies and standard deviations from the first data set, over 5 generations

Depth	Average accuracy	Standard deviation
1	0.5026	0.0
2	0.6300	0.1274
4	0.6875	0.1320
8	0.7350	0.1409
Unspecified	0.7633	0.1381

TABLE 2 – Average accuracies and standard deviations from the second data set, over 5 generations

1.3

The main difference between these two problems is that for the first data set the orange and blue objects are clearly apart in the attribute space. There is an ellipse with the blue objects and all the orange objects lie in the center of it. However, for the second data set, there are 2 ellipses, with their major axes forming a "x". In this case and as it has already been said, the intersection points of the ellipses are regions where there are a lot of blue *and* orange objects. At those intersections, it is much harder to find a decision boundary because a given point can be blue and orange at once.
This observation also appears when analysing the test accuracies. Even for a depth of 1 for the first data set, one observes a much higher accuracy than for a depth of 8 for the second data set.

One can conclude that in practice it will be much harder to find an optimal model for the second problem than for the first one.

2 K-nearest neighbors

2.1

a. The decision boundary of the k-NN algorithm for the first data set and for a number of neighbors equal to 1, 5, 10, 75, 100 and 150 is shown in the following figures.

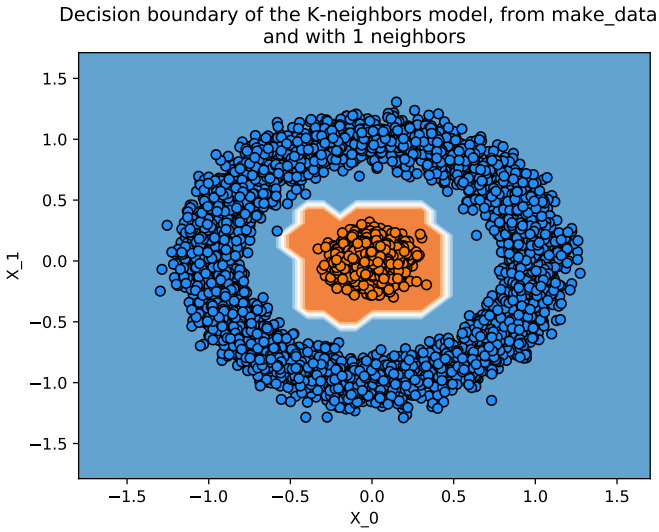

FIGURE 11

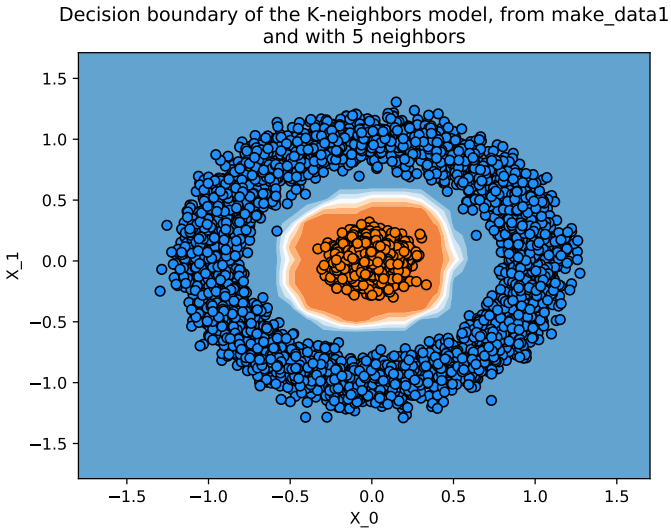

FIGURE 12

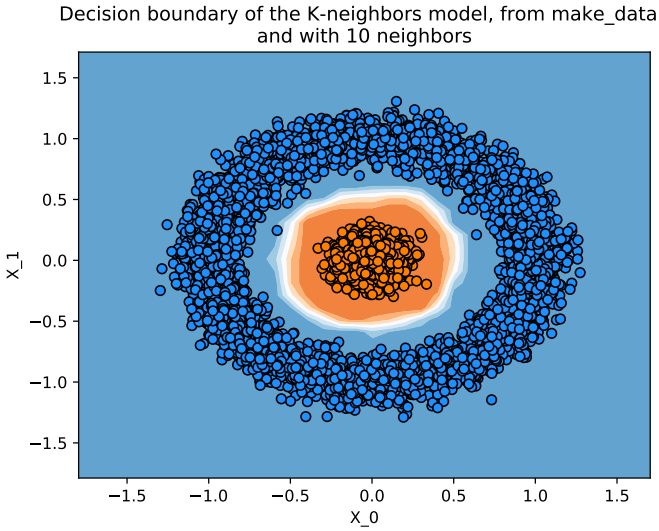

FIGURE 13

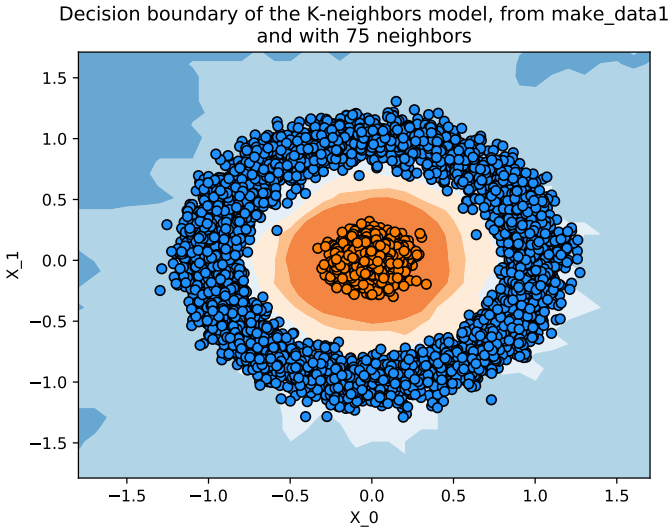

FIGURE 14

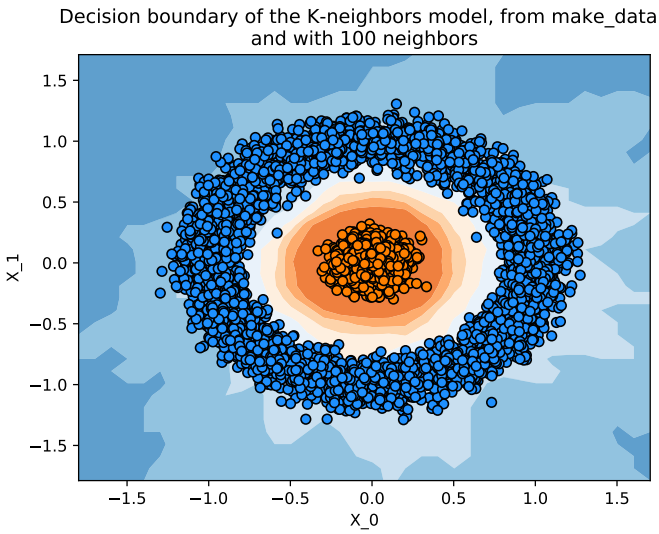

FIGURE 15

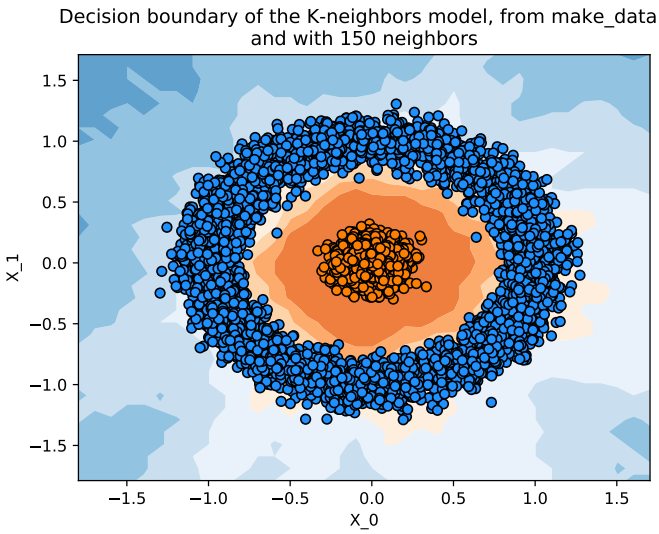

FIGURE 16

The decision boundary of the k-NN algorithm for the second data set and for a number of neighbors equal to 1, 5, 10, 75, 100 and 150 is shown in the following figures.

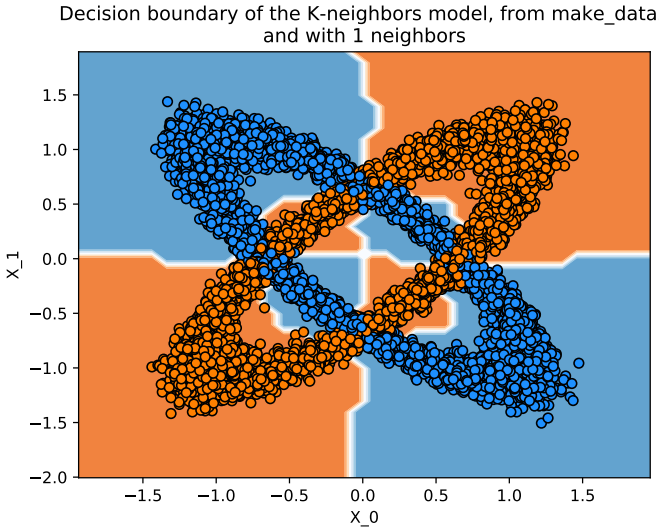

FIGURE 17

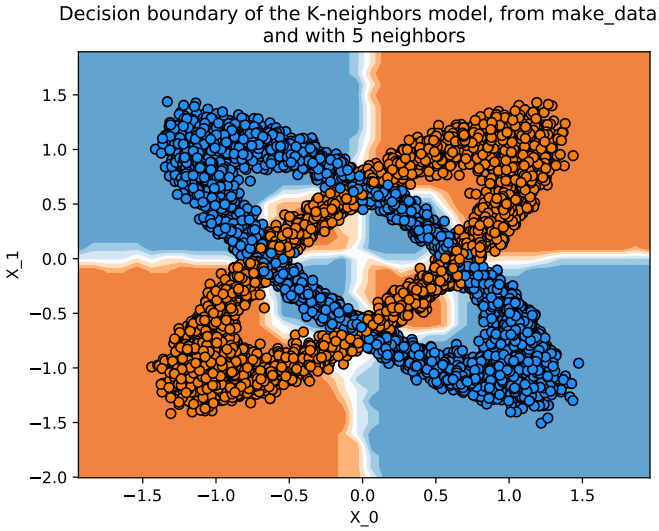

FIGURE 18

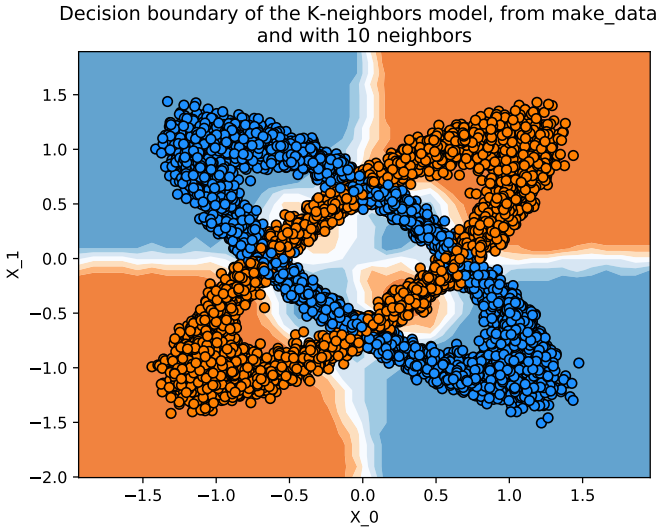

FIGURE 19

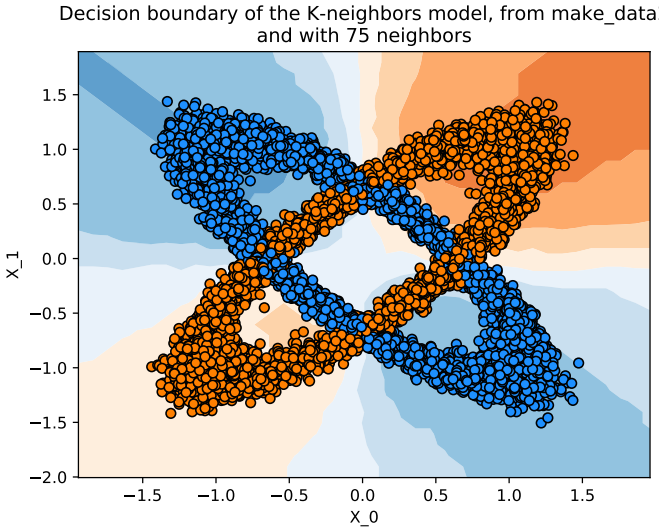

FIGURE 20

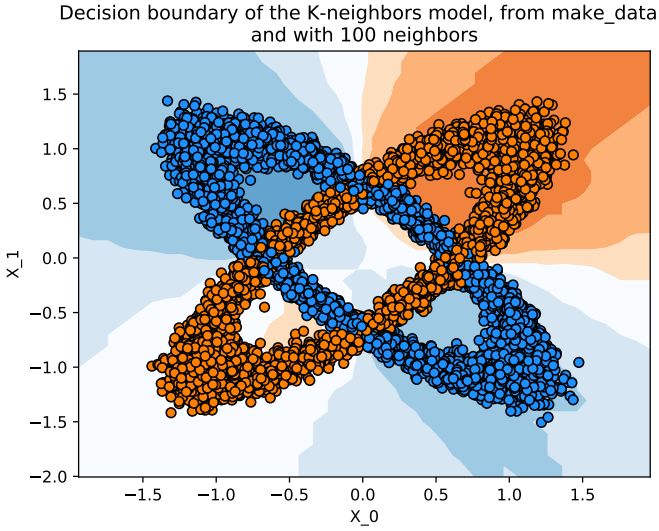

FIGURE 21

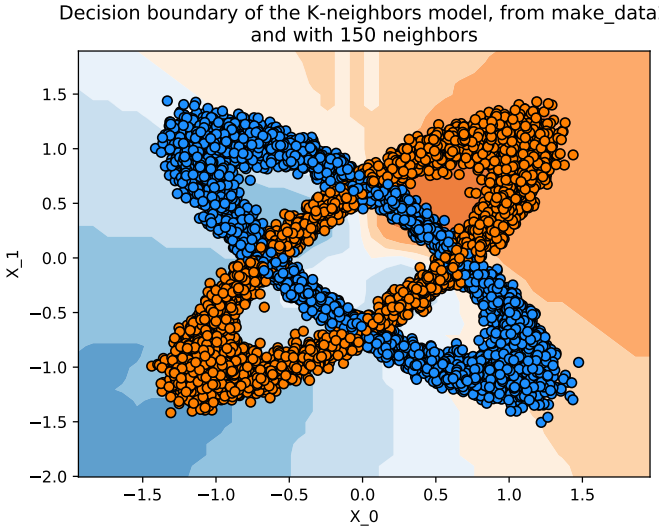

FIGURE 22

b. One can observe that, for sufficiently low values of the number of neighbors, the nearest neighbors model is much more accurate than the decision tree model. The nearest neighbor model with only one neighbor is already much more precise and the decision boundary seems to make more sense than for the decision tree model. For the first data set at least, all the orange objects are located in the orange region and every blue object is located in the blue region which has never been the case with the decision tree.
The outline of the orange region is white, which means that in this region the model can not predict whether a object is blue or orange. By increasing the number of neighbors, this white region gets bigger and one obtain blue and orange regions which are much lighter. This means that the model is less confident in those regions, since among the considered neighbors there are now a certain amount of objects of the right class and a certain non-negligible amount of objects of the wrong one. This could lead to better performances since the model takes more neighbors into account and intuitively gets more general and less glued to the LS. But if it takes too much neighbors into account, it could eventually consider objects that are not representative of the studied region, and one could end up with bad decision boundary.

Indeed, for a number of 75 neighbors, the model is performing much worse. Nearly none of the blue objects is now located in a strong blue region but all in a much lighter blue region or even in a white one.
For 100 neighbors the performance still seems to decrease a little but for 150 neighbors the performance gets really bad. Indeed, for the first data set, most of the blue objects are located in a white region and some of them in a lighter orange region.
In our opinion, for the first data set, the optimal number of neighbors has to be between 1 and 5.
For the second data set, one can observe a similar phenomenon. Again, the model with 100 and 150 neighbors perform very poorly. Especially for 150 neighbors, a big part of the orange objects are located in a blue region, and even in a strong blue region. Furthermore, one can observe a nearly binary division of the attribute space, which perfectly shows how considering too much nearest neighbors can lead to incoherent decision boundaries. As a matter of fact, the "nearest" attribute loses all its meaning.
This time we guess that the optimal number of neighbors will be between 1 and 10.

2.2

a. The five-fold cross validation strategy is implemented in the `knn.py` file, in particular in the `fold` and the `neighbor_optimizer` functions.

   The methodology is quite simple. The `fold` function gets as parameters the data set on which one wants to perform the cross validation. Thanks to the `split` method of the `numpy` package, the indexes of the data set are split in five parts (the number of folds is also passed as a parameter). The function then generates five tuples, each containing one of the sets of indexes as test set indexes (different from one tuple to another) and a concatenation of the four other ones as training set indexes for the cross validation.
   This set of tuples is then passed to the `neighbor_optimizer` function which uses it to divide five times the original data set into a test set and a training set.

   In order to determine an optimal value for the `n_neighbors` parameter, the algorithm then fits the model for different values of `n_neighbors` with the training sets and evaluate its accuracy with the test sets. For each `n_neighbors` value, it computes the mean accuracy over the five folds and eventually determine the best `n_neighbors` value as the one corresponding to the best mean accuracy.

b. In figure 23, one can observe that the optimal number of neighbors is 3 or 6 with a computed mean accuracy of 0.932 [2]. Furthermore, by increasing the number of neighbors, the performance of our model decreases significantly as previously discussed.

2. Please note that in the file `knn.py`, since only the first part of the graph is relevant for optimization, only the values from 1 to 10 are considered to lighten the computation.

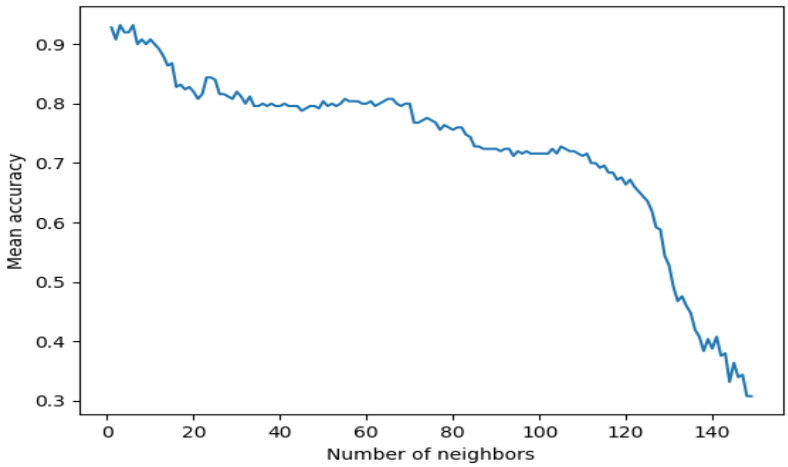

FIGURE 23 – Mean accuracies for all the numbers of neighbors

As expected earlier, the optimal number is between 1 and 10, which is not surprising as these values do not lead to incoherent decision boundary since they do not imply to consider far-located non-relevant objects. In addition, the optimal value(s) found is (are) sufficiently high to provide a good generalization property, since the model does not only base its prediction about one object on only one other object.

2.3

a. In the following figures, one can observe the test accuracy (on a TS of size 500) for every possible number of neighbors [3] for LS of sizes 50, 200, 250, 500 for the first data set.

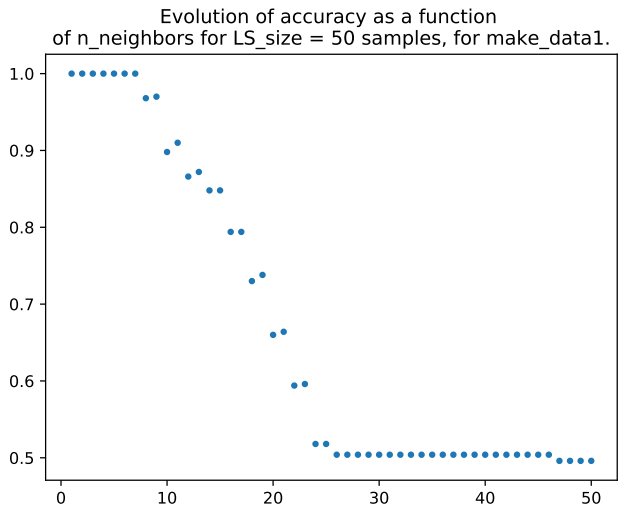

FIGURE 24

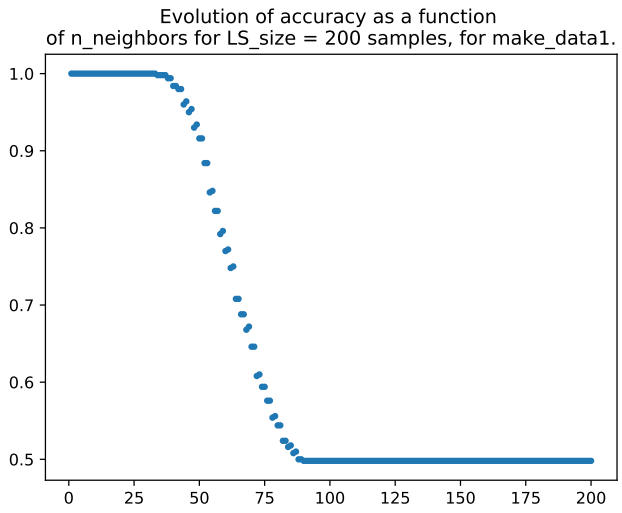

FIGURE 25

3. Meaning up to the size of the LS.

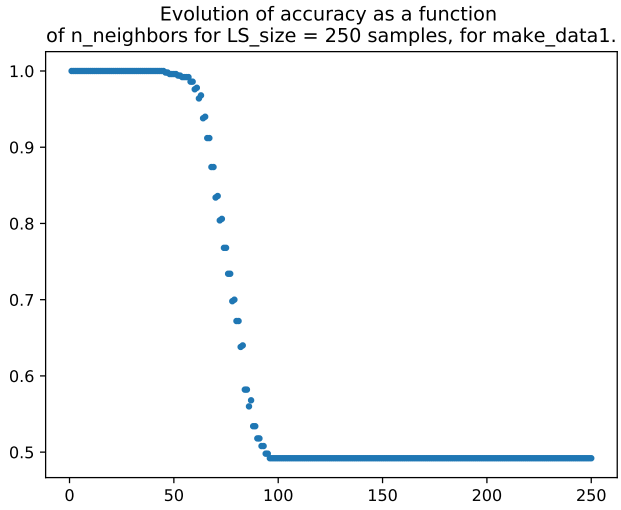

FIGURE 26

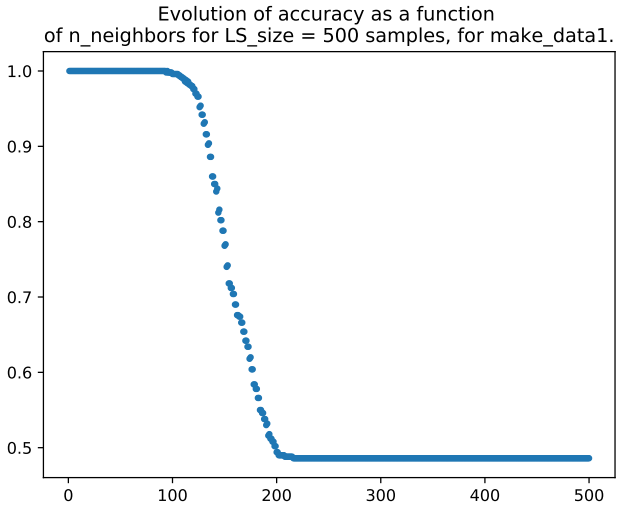

FIGURE 27

In the following figures, one can observe the test accuracy (on a TS of size 500) for every possible number of neighbors for LS of sizes 50, 200, 250, 500 for the second data set.

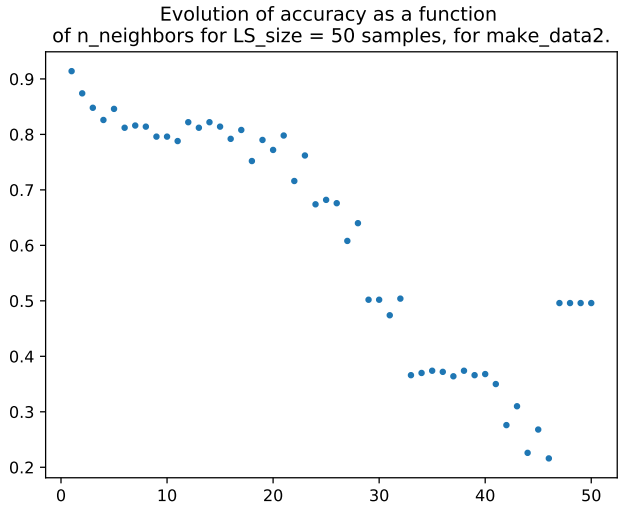

FIGURE 28

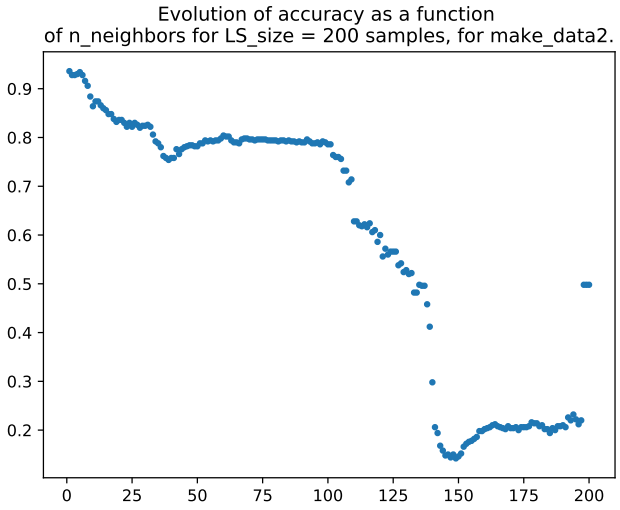

FIGURE 29

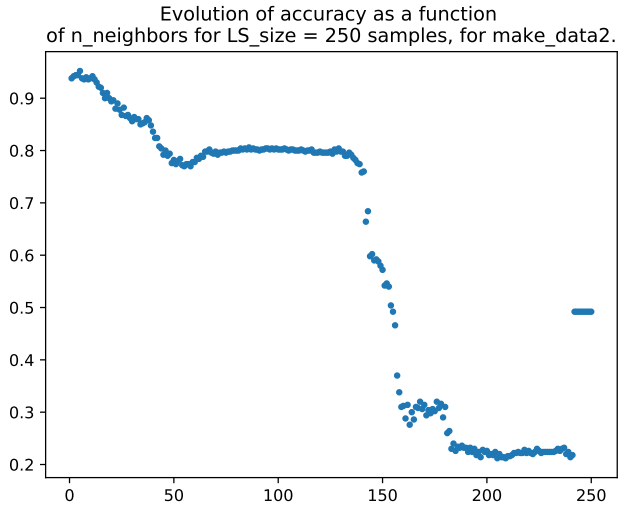

FIGURE 30

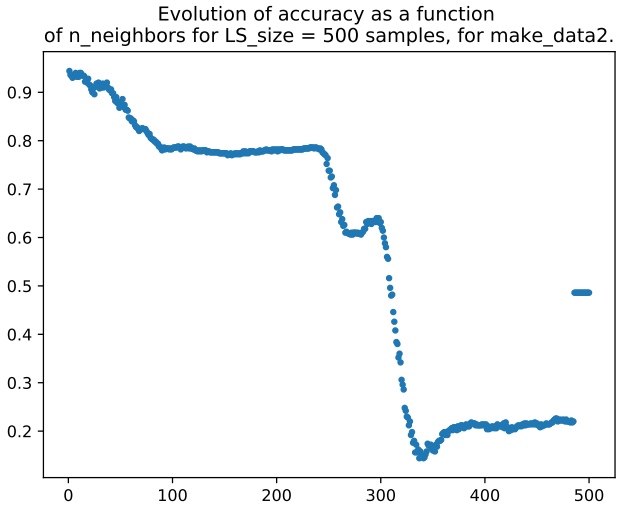

FIGURE 31

b. The evolution of the optimal value of the number of neighbors with respect to the LS size is shown in figure 32 for the first data set and in figure 33 for the second data set.

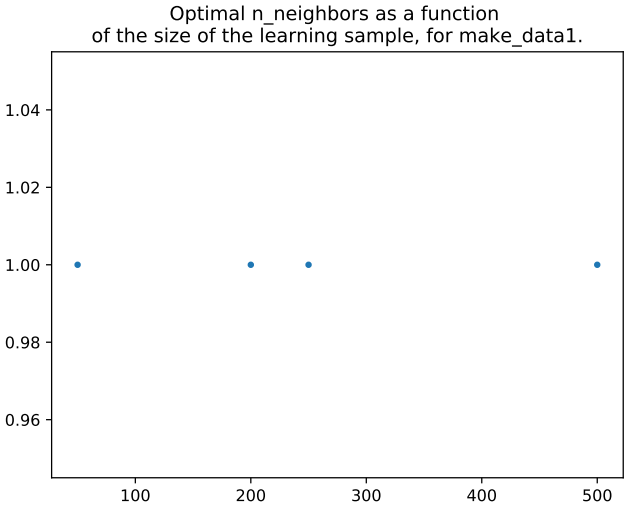

FIGURE 32

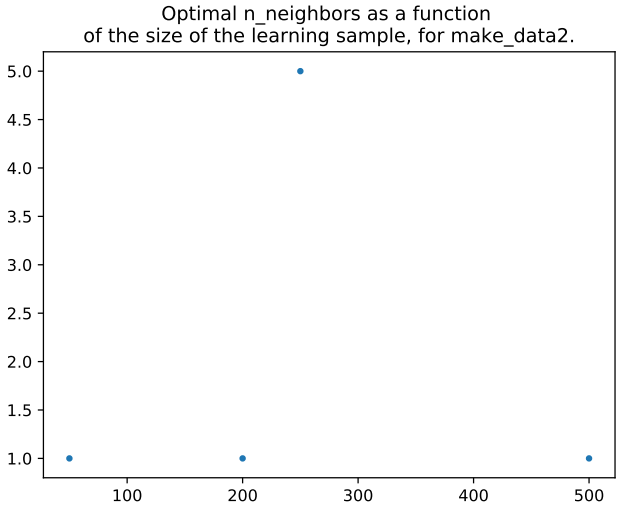

FIGURE 33

— **First data set**

For the first data set, one can observe four times the same sigmoid-like shape of graph. First of all, the graph describes a quite straight line where the mean accuracy seems equal or at least very close to 1. After that, the mean accuracy drops significantly until it reaches 0,5.
Finally, the graph describes a straight line again where the mean accuracy is equal to 0.5. To conclude, one can say that by increasing the size of the learning sample, the biggest number of neighbors for which the mean accuracy will still be close to 1 increases too. In other words, when increasing the size of the learning sample, a higher number of neighbors can be used without loosing performance or accuracy. For a learning set of 50 data points, the optimal number of neighbors is between 1 and 10, and for a learning set of 500 data points, it is already between 1 and about 135. This is quite logical since if the size of the

LS increases, one can expect it to become denser in the attribute space. So, in this case, even for a bigger number of neighbors taken into account, the majority of them will still be kind of close from the evaluated object, meaning that the majority of them is more likely to be from the same class as the evaluated object.
For every size of the learning set, the optimal number of neighbors is equal to 1.

— **Second data set**

For the second data set, one can observe that high mean accuracies (close to 1) are reached for a smaller range of number of neighbors than for the first data set. Furthermore, the highest mean accuracy is clearly obtained for only one value of the number of neighbors, while for the first data set, even if a number of neighbors equal to 1 is presented as the optimal one, the graphs show that a wide range of values could be considered as nearly optimal.
For high values of the number of neighbors, one has a mean accuracy of roughly 0.25 and suddenly, when the number of neighbors is approximately the size of the learning set, the mean accuracy jumps to 0.5. While surprising, this phenomenon could intuitively be guessed by saying that, since one consider nearly all or all objects of the LS, and if one consider the LS to be compound of nearly the same amount of objects of both classes, then the model will, for each object of the TS, give the same probability equal to 0.5 as each object has an equal number of orange and blue neighbors. The model will thus have an accuracy of 0.5 for each element and a global accuracy of 0.5 too. For every size of the learning set, the optimal number of neighbors is equal to 1, except for a size of 250 where the optimal number is 5.

2.4

A five-fold cross validation would not be adapted here. Indeed, for a LS of size 250, the five-fold cross validation would provide a mean accuracy over the five folds corresponding to a LS size of 200 (since $250 - \frac{250}{5} = 200$) rather than 250. Actually, it is impossible to use five-fold cross validation in order to determine the optimal value for a LS size of 250 since 250 is not a multiple of 4.

3 Residual fitting

3.1

The residual at step k is defined by

$$\Delta_k y(o) = y(o) - w_0 - \sum_{i=1}^{k-1} w_i a_i(o), \tag{1}$$

and the optimal weight w_k^* for the attribute a_k to be introduced in the model must be such that

$$\Delta_k y(o) - w_k^* a_k = y(o) - w_0 - \sum_{i=1}^{k-1} w_i a_i(o) - w_k^* a_k = 0 \tag{2}$$

in order to ensure the optimality of the linear approximation of the output $y(o)$.
Furthermore, the residual at step k can be considered as the new output in such a way that the

problem now has only one attribute $a_k(o)$ for which the optimal weight value w_k^* is sought. From now, the residual $\Delta_k y(o)$ can be substituted for $y(o)$ into the expression

$$\frac{y(o) - \overline{y}}{\sigma_y} = \rho_{a_1,y} \frac{a_1(o) - \overline{a}_1}{\sigma_{a_1}},$$

appearing in the least mean square error solution for dimension one [4], and the expression writes

$$\frac{\Delta_k y(o) - \overline{\Delta_k y}}{\sigma_{\Delta_k y}} = \rho_{a_k,\Delta_k y} \frac{a_k(o) - \overline{a}_k}{\sigma_{a_k}}. \tag{3}$$

Since residuals have zero mean and attributes are pre-whitened so that their mean and standard deviation are equal to 0 and 1 respectively, one can now write

$$\Delta_k y(o) = \rho_{a_k,\Delta_k y} \sigma_{\Delta_k y} a_k(o) \tag{4}$$

and, finally, inserting (2) into (4),

$$w_k^* = \rho_{a_k,\Delta_k y} \sigma_{\Delta_k y} \tag{5}$$

which is the expected expression for the optimal weight w_k^*.

3.2

The implementation of the residual fitting algorithm as described in the slides of the theoretical course can be found in the `residual_fitting.py` file.

3.3

a. The decision boundary of the residual fitting algorithm for both data sets is shown in figures 34 and 35.

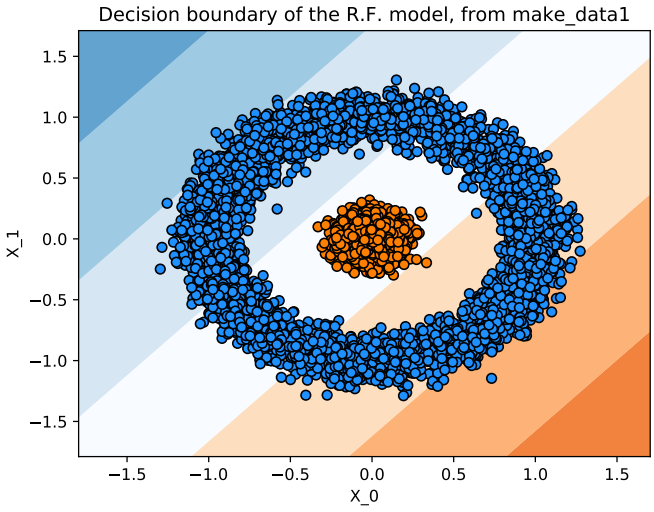

FIGURE 34

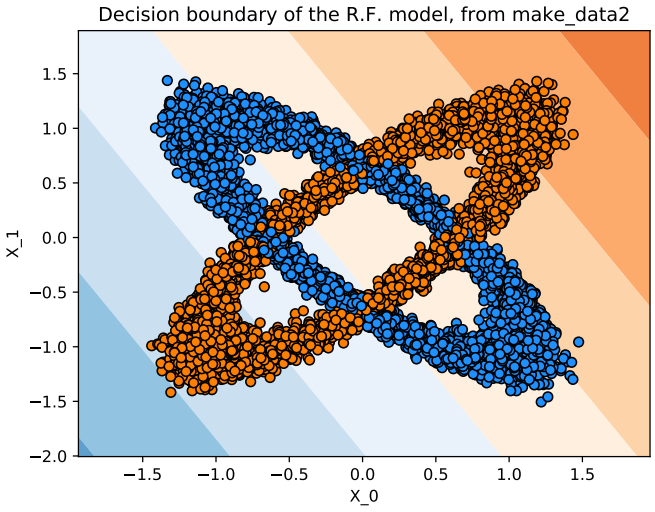

FIGURE 35

Since the residual fitting algorithm provides optimal weights for a *linear* approximation of the output with only two attributes X_0 and X_1, this algorithm is not expected to fit well both studied data sets. Indeed, as it can be seen in figures 34 and 35, one can clearly see the linear

4. This expression appears on slide 5 of the chapter 3 (*Linear regression*) of the theoretical course.

behavior of the classifier, since the decision boundaries look quite like variants of the identity function, roughly classifying objects in two distinct parts of the attribute space. Then, given the shape of both data sets in the attribute space, this model can not depict accurately the input-output relation of both problems and can not be used to make predictions with a sufficient accuracy.

b. The following table shows the test accuracy obtained by using the residual fitting algorithm on the two data sets.

Data set	Test accuracy
1	0.4856
2	0.5595

TABLE 3

For both data sets, it is clear that the test accuracy is not really good, for instance in comparison with the other algorithms previously studied in this project. Given the decision boundaries and the corresponding discussion that has been done, this is not surprising.

3.4

a. The decision boundary of the residual fitting algorithm for the modified version of the second data set is shown in figure 36.

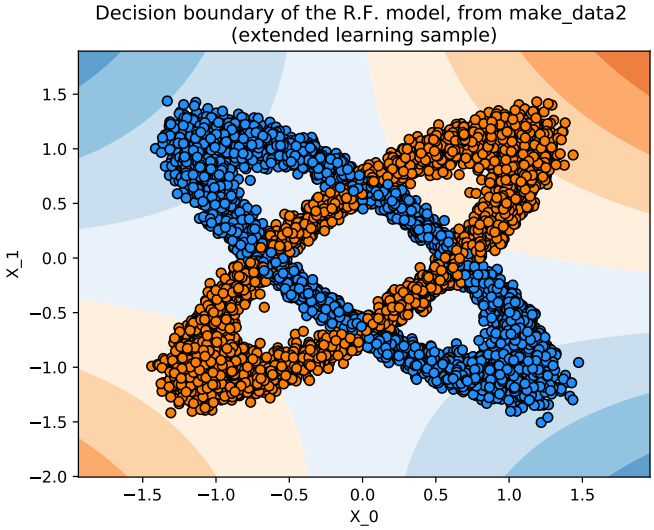

FIGURE 36

Since the residual fitting algorithm now provides more optimal weights corresponding to more attributes, and since those added attributes are quadratic combinations of the original ones, the fitted model's decision boundary now depict more accurately the attribute space. Indeed, those quadratic combinations allow quadratic decision boundary which, given the non linear shape of the second data set in the attribute space, are more suited. Still, the central part of the attribute space is not quite well described.

b. A test accuracy of 0.7881 is obtained when training the residual algorithm with the extended version of the second data set as asked in the statement. This result is way better than the previous one (obtained with the original attributes only), which is not surprising given the decision boundary. It is however still not perfect and one could imagine that adding more attributes consisting of more complex quadratic (or even cubic, etc.) combinations of the original ones might lead to even better results. Again, the intersection points of the ellipses may still end up to be tough regions to fit. It is finally important to notice that although the accuracy has increased, the central part of the attribute space is predicted to be of the 'blue' class, so while this may increase the accuracy by correctly predicting blue objects of the TS located in that particular area, the boundary miss all the orange predictions of that same area.