University of Liège
Faculty of applied sciences
1^{st} Master in data science engineering
October 2020

Introduction to machine learning

Project 1 - Classical algorithms

ELEN0062 - Introduction to Machine Learning
GEURTS Pierre
WEHENKEL Louis
SUTERA Antonio

S153809 BACKES Lucas
S143401 LEERSCHOOL Adrien

1 Decision tree

1.a Decision boundary

In the first set of data, we could imagine that differentiating the two different classes is not a hard problem since they are distinctively separated. Indeed, at a glance we can estimate that the blue dots are in the interval $x, y \in [-0.5; 0.5]$ and the orange dots are the remaining ones. Nevertheless, the decision tree algorithm tries to minimize its error at each step as it is going deeper and deeper, which result in a non-optimal classification although it seems possible, in this case, to find a perfect one.

With a depth equal to 1, the classifier simply divide the points with a vertical line since the latter bases its separation on only one inequality, which is $X\_0 \leq -0.169$ (see the top square in figure 1). As we adding more depths in the decision tree, the algorithm is becoming more accurate and can first, for a depth of 2, create an interval on the x-axis and then, for a depth of 4, create an interval on the y-axis too (see the two middle schemes in figure 2). Since we can encompass the blue dots in a rectangular box, the additional depths do not bring better classifying performances as we can see on figure 1 and figure 2. Indeed, the figure 1 shows the decision tree generated by the `DecisionTreeClassifier` algorithm of the `Scikit-Learn` library and we observe that even without constraining the maximum depth, the latter does not generate deeper nodes than with previous constraints.
In this first data set, there is no over-fitting since, as explain above, the decision tree goes at a maximum depth of four, which cannot generate this kind of problem as, by definition, it is caused by an exaggeration when fitting the data. Nevertheless, we can clearly see that the model is under-fitting when there is a constraint on the depth equal or less than 2. In that case, the model cannot operate as its best and so roughly classify the data.

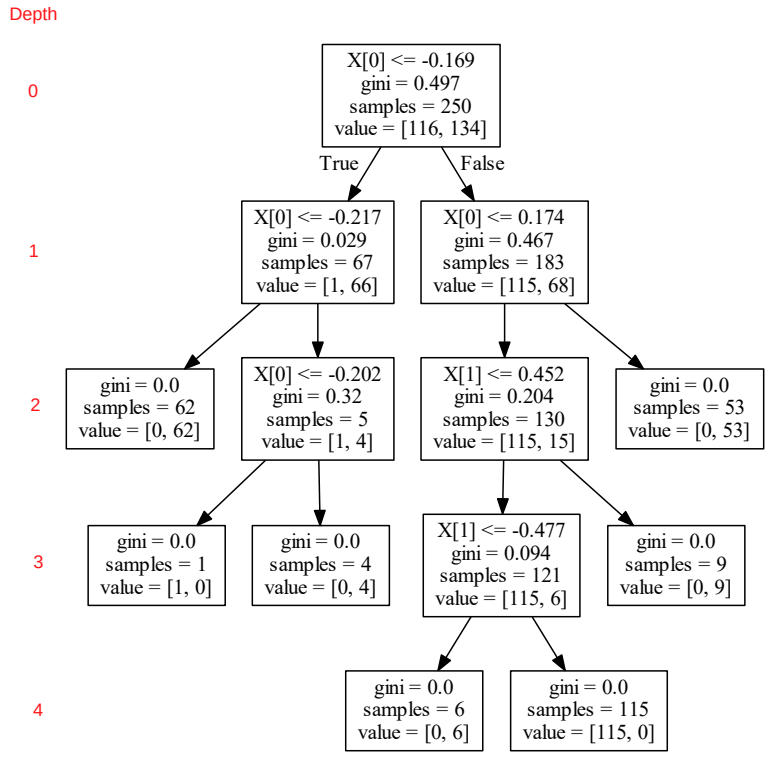

FIGURE 1 – Decision tree without depth constraint for the first set of data

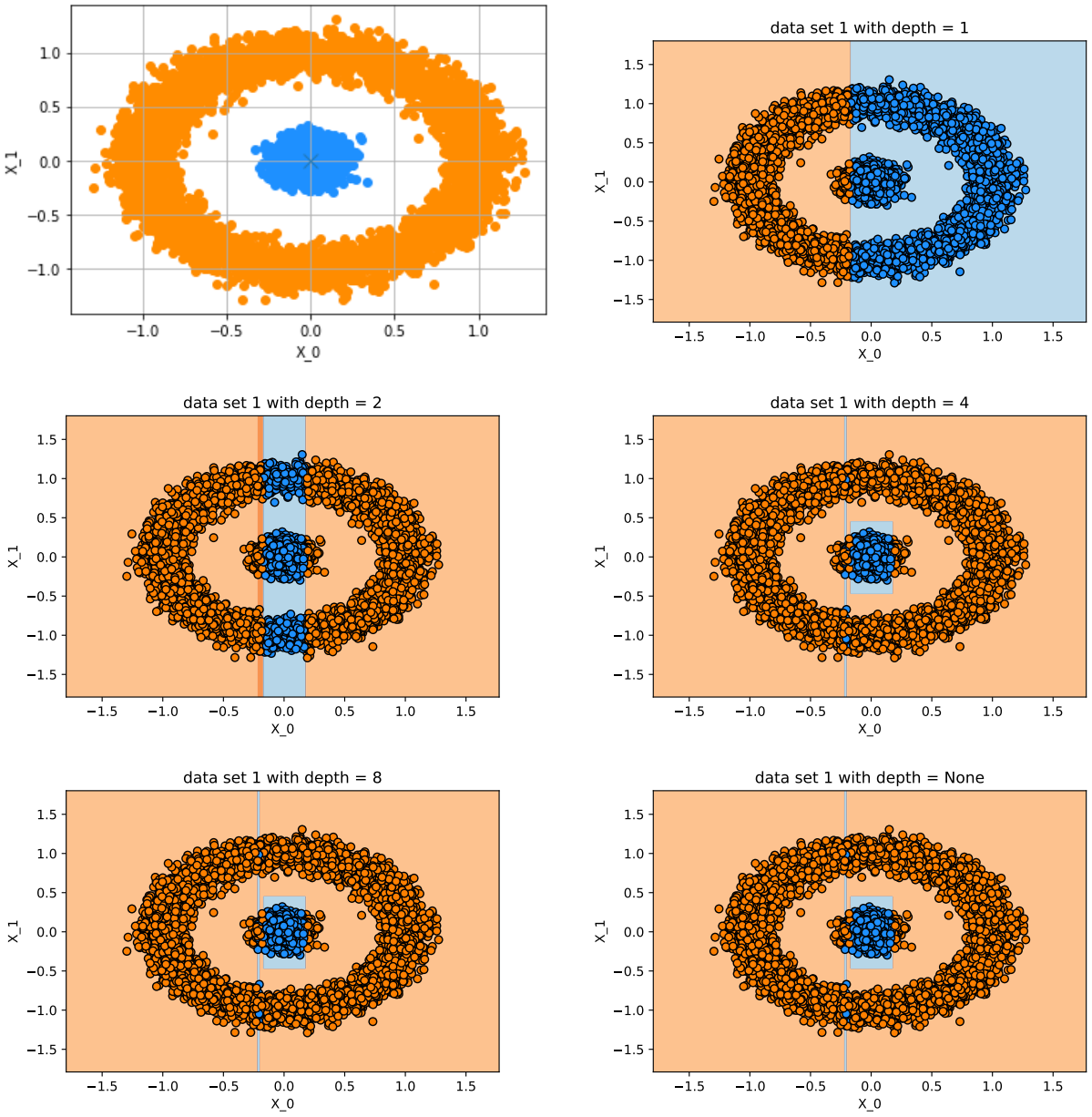

FIGURE 2 – Decision boundary for the first set of data with a depth of 1, 2, 4, 8 and none

In the second set of data, the classification seems to be a little bit more tricky because of the '*intersection*' of the two classes (see top left scheme on figure 6).
For a depth constraint of 1, we can realize, as before, that the algorithm simply divides the point set in half. For a constraint on the depth of 2, the latter reacts slightly differently from before and creates directly intervals on the y-axis too (because it minimizes the most the error) (see middle left scheme in figure 6) to frame at best the blue dots. As we relaxing the constraints, we see appearing more '*little rectangles*' which fit the data more accurately up to a certain ceiling. As a matter of fact, when we run the algorithm without constraints, the latter stops at a depth of 12 (see the corresponding decision tree in figure 3).

In contrast with the first set of data, the second one leaves possibilities to over-fitting. Actually, we observe on figure 6 that some points are not in the ellipses and may lead to an **little** over-fitting of the data. We use the term **little** because we see on figure 3 that despite the distances of some points

from the ellipses, the decision tree remains quite shallow (max depth = 12). That means that the algorithm did probably not created too much leaves to perfectly fit the training data. On the other hand, like with the first set of data, the model clearly under-fits the second one when we constraint the maximum depth to 4 or less.

As we expect it, if we leave the algorithm unconstrained, it will be able to fully exploit its '*precision*' by creating more `if` statement/inequalities to separate the points in an effective way and so create a more confident model.

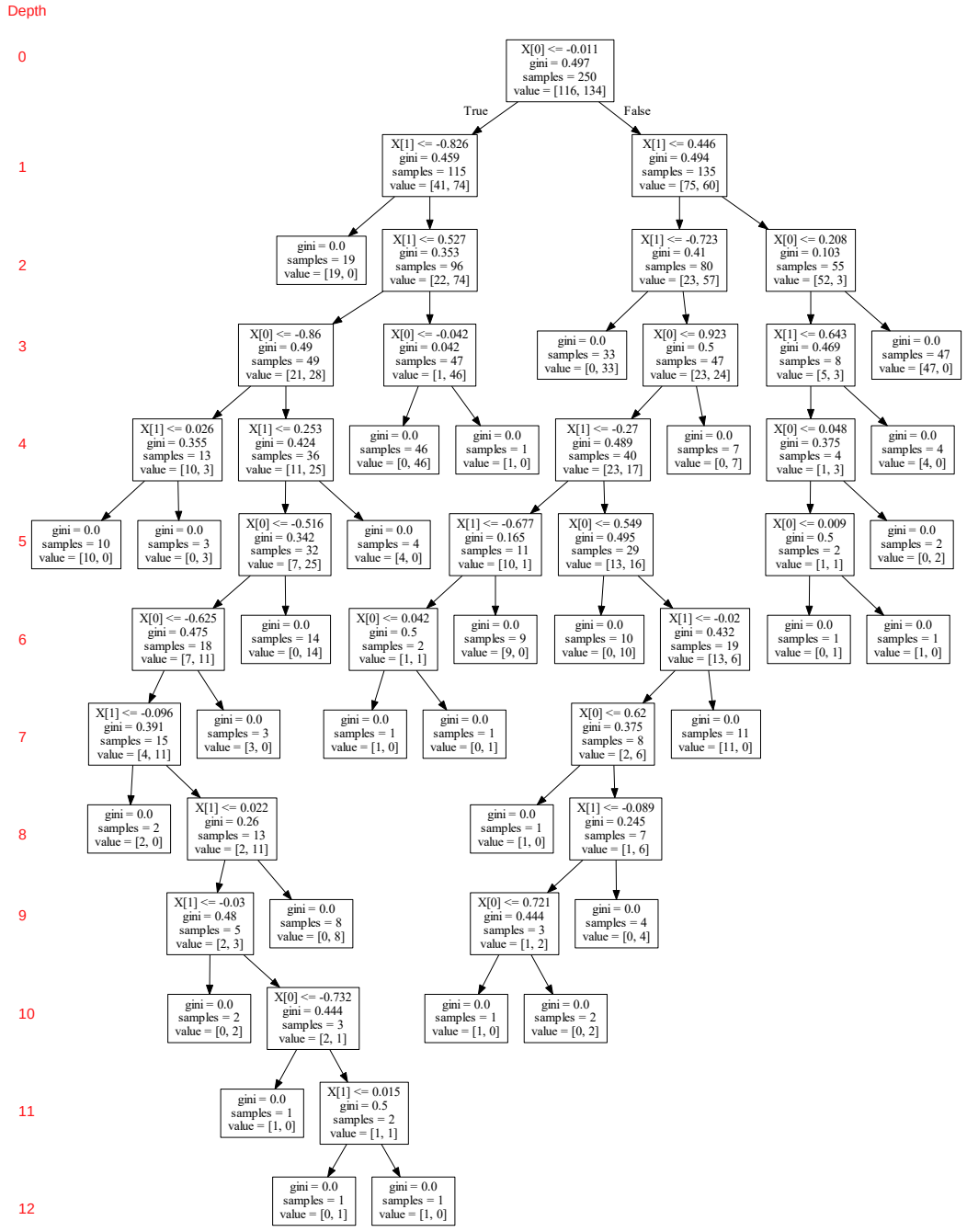

FIGURE 3 – Decision tree without depth constraint for the second set of data

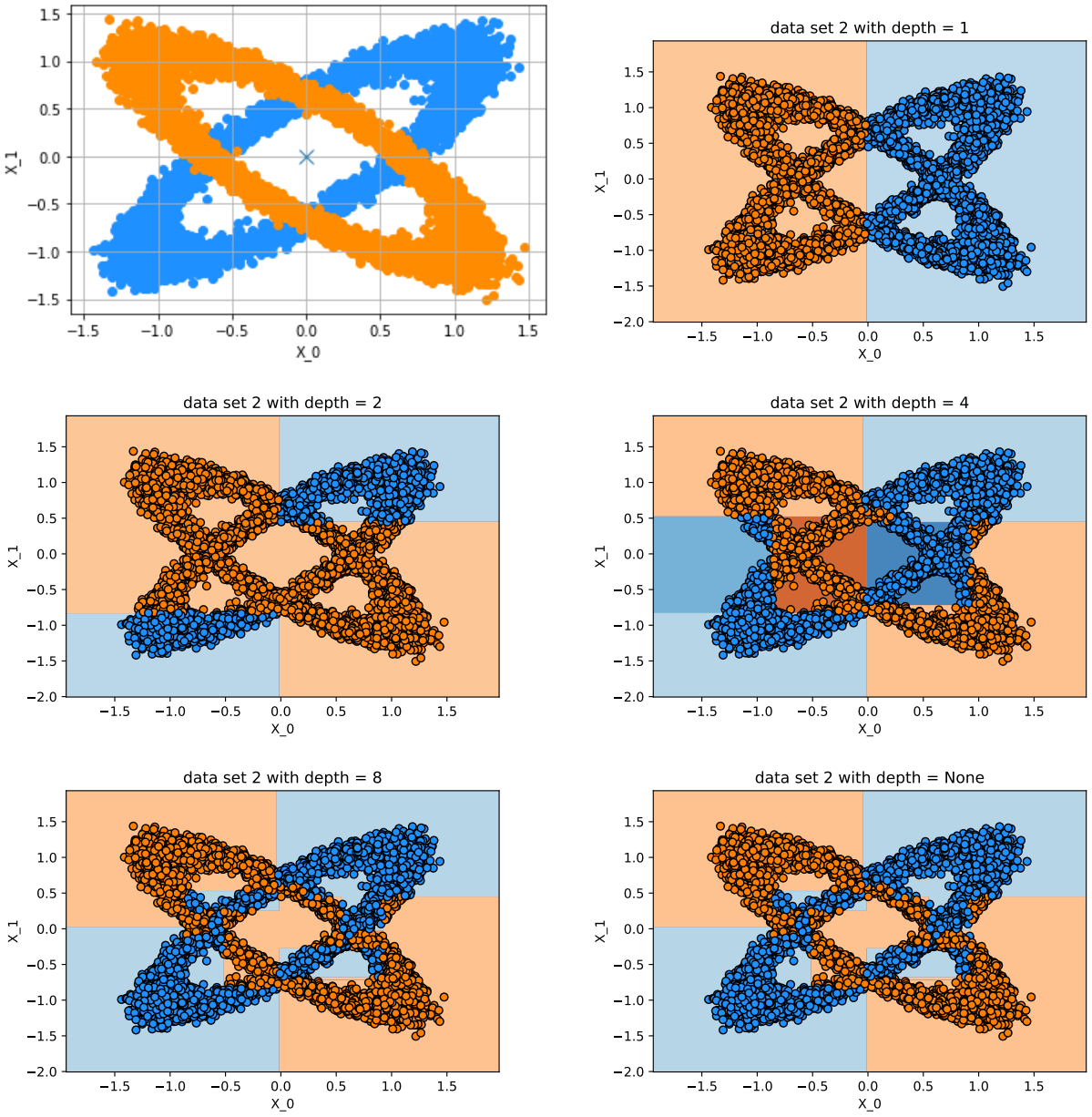

FIGURE 4 – Decision boundary for the second set of data with a depth of 1, 2, 4, 8 and none

1.b Accuracies and standard variations

Below in table 1 is gathered the accuracies and the standard deviations (averages over 5 generations) of the two data sets with different depth constraints. First thing to notice, is the fact that no matter which set we are talking about, the accuracy increase with the depth constraint as does so for the standard deviation. Second thing to note is the difference of accuracy between the two sets. We clearly observe that the first set (the one with the two concentric circles) has a better precision than the other, which can be said to be logical since the two classes are well separated (see top left scheme on figure 2).

The one interesting conclusion to make is that the standard deviation increases when we relax the constraint, which can be interpreted as the fact that the deeper the tree can go, the more variable the tree can be over multiple generations of the data.

Maximum depth		First data set	Second data set
1	Accuracy	0.7138	0.5026
	Standard deviation	0.0	0.0
2	Accuracy	0.9319	0.7574
	Standard deviation	0.0	0.0
4	Accuracy	0.9834	0.8026
	Standard deviation	1.11×10^{-16}	0.0
8	Accuracy	0.9834	0.8781
	Standard deviation	1.11×10^{-16}	2.87×10^{-4}
None	Accuracy	0.9834	0.8758
	Standard deviation	1.11×10^{-16}	7.12×10^{-4}

TABLE 1 – Accuracies and standard deviations for different depths of the two data sets

1.c Differences between the two problems

We spotted two main differences between the two decision tree classifiers :

1. **The decision tree :** We can easily notice by comparing figure 1 and figure 3 that the two trees have not the same depth. As a matter of fact, the first data set being easier to classify (by the non-ambiguity of the separation of the two classes), it results in a shallower tree than the one of the second data set.

2. **The accuracy :** As discussed just above, the accuracies of the two models are obviously different and the first model converge quicker with the constraint relaxing than the second one. This result comes from the fact that, again, the two classes of dots are more distinct in the first data set, making the classification easier for the corresponding model.

2 K-nearest neighbors

2.a Decision boundary

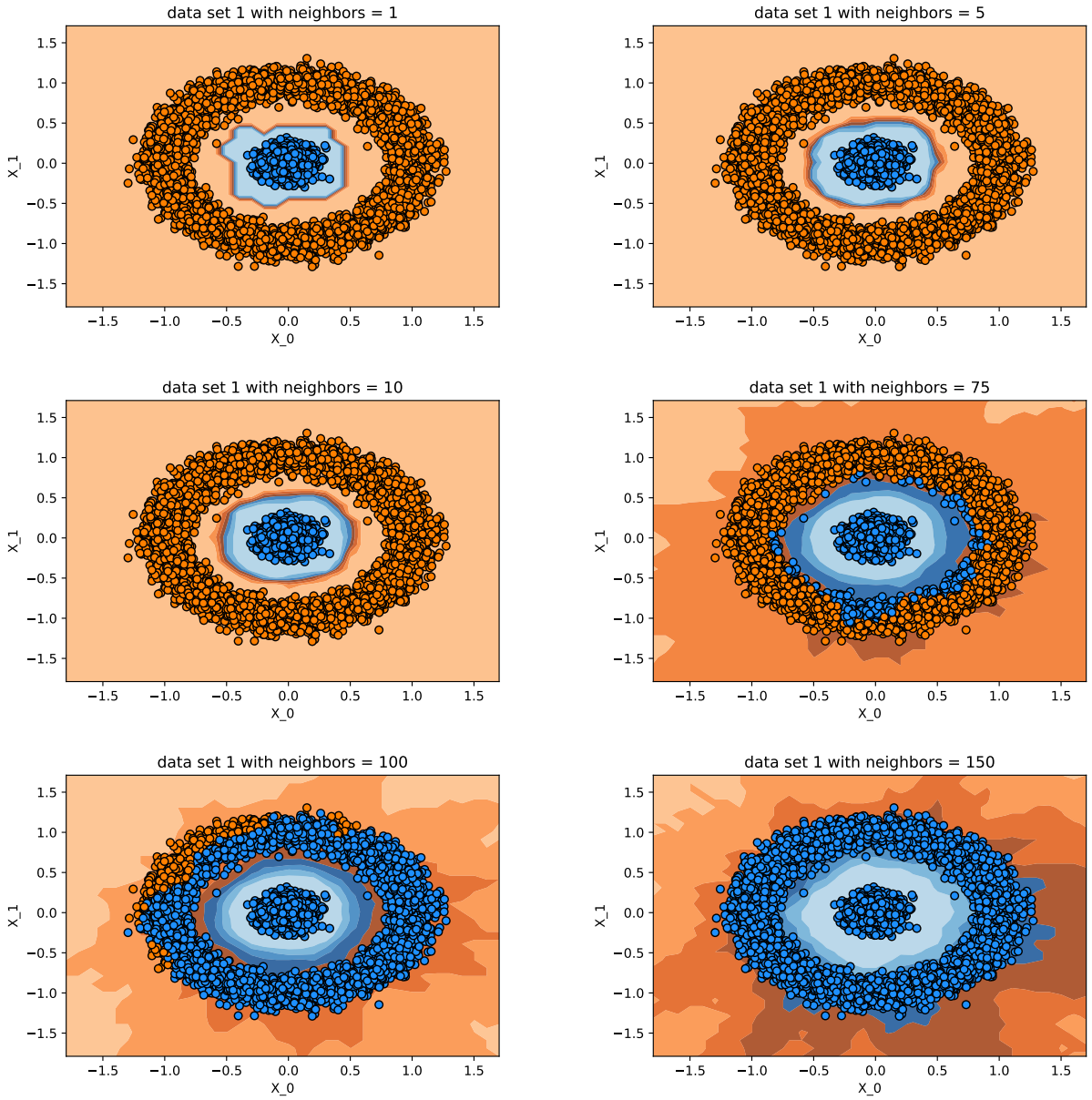

FIGURE 5 – Decision boundary for the first set of data with n_neighbors of 1, 5, 10, 75, 100, 150.

Neighbors	1	5	10	75	100	150
Accuracies	1.0	1.0	1.0	0.9573	0.5547	0.5009

As the graphs and the table suggest, the decision boundary gets wider and wider as the number of neighbors increases. It seems logical knowing that the entire set of the blue points is in the middle. Indeed, if the number of neighbors is extremely large, like the total number of data points, a 'orange' test point will immediately be closer to the center than to the other side of the ellipse. Therefore we get a 50% accuracy rate when the number is large since all points are predicted as blue ones.

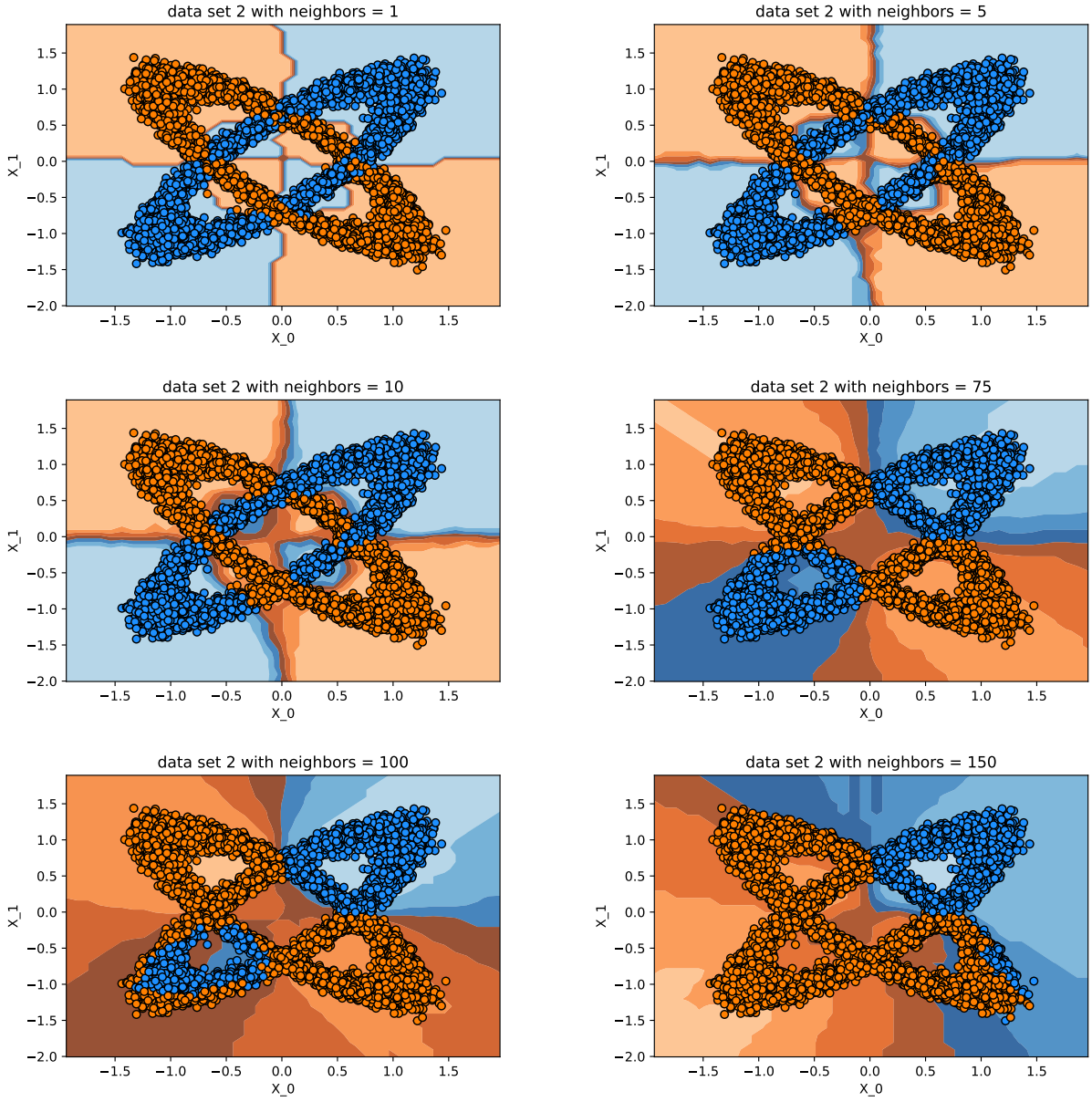

FIGURE 6 – Decision boundary for the first set of data with n_neighbors of 1, 5, 10, 75, 100, 150.

Neighbors	1	5	10	75	100	150
Accuracies	0.9387	0.923	0.9114	0.7927	0.7424	0.6225

For the second data set, we notice a couple of things. First, We never get a total accuracy like the first one. Secondly, The decision boundaries are a bit more tricky mostly because the data overlaps (which is a big difference with the first data set). We can also observe that for the last two `n_neighbors`, the accuracies for the second data set are higher than for the first one.

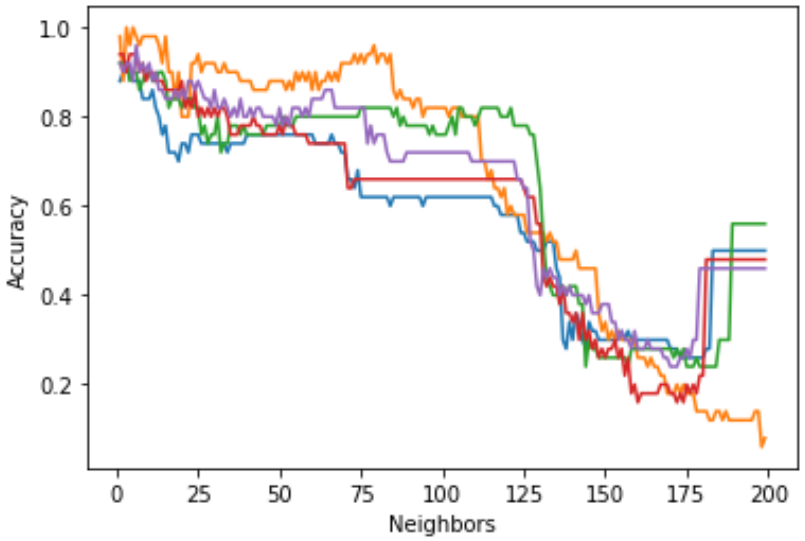

FIGURE 7

2.b Optimization of the *n_neighbors* parameter

Our methodology in order to perform a five-fold cross validation is the following. Once the data is created, it is directly split into 5 parts. Since only the learning sample is used for the cross-validation, each sections has fifty data points. Each section is then used, in turn, as a testing set for the model built on the learning of the remaining 200 data points.

For each testing set, we measure the accuracy of the 200 potential neighbors in a `KNeighborsClassifier`. The neighbor with the highest mean score over the five portions of data is considered the optimal one. His average accuracy over the five folds is also computed.

When it comes to our code, a function `cross_validation_function` was created in order to lighten the reading of the main function. This function takes into argument the integer k, as well as a training set of the second data set. It returns the optimal neighbors of that particular training set over a k-fold cross validation, as well as the mean score over the five folds of that optimal neighbor.

On figure 7 is a example of the accuracy evolution depending on the number of neighbors taken from a specific 5-fold cross validation. Each color corresponds to a different section of the data being tested.

Keeping in mind not to depend on a particular instance of the `makedata2` function, a hundred iterations were made. See below for the results.

Most frequent neighbor	1
Mean value of the neighbor	1.9898

2.c Optimal value of *n_neighbor* depending on the size of the learning set

For the first dataset (Figure 8) the behavior of each curves is really similar. The accuracy is at a maximum value of 1 for more or less 25% of the learning samples and then drastically decreases

until reaching a plateau of 0.5 accuracy.

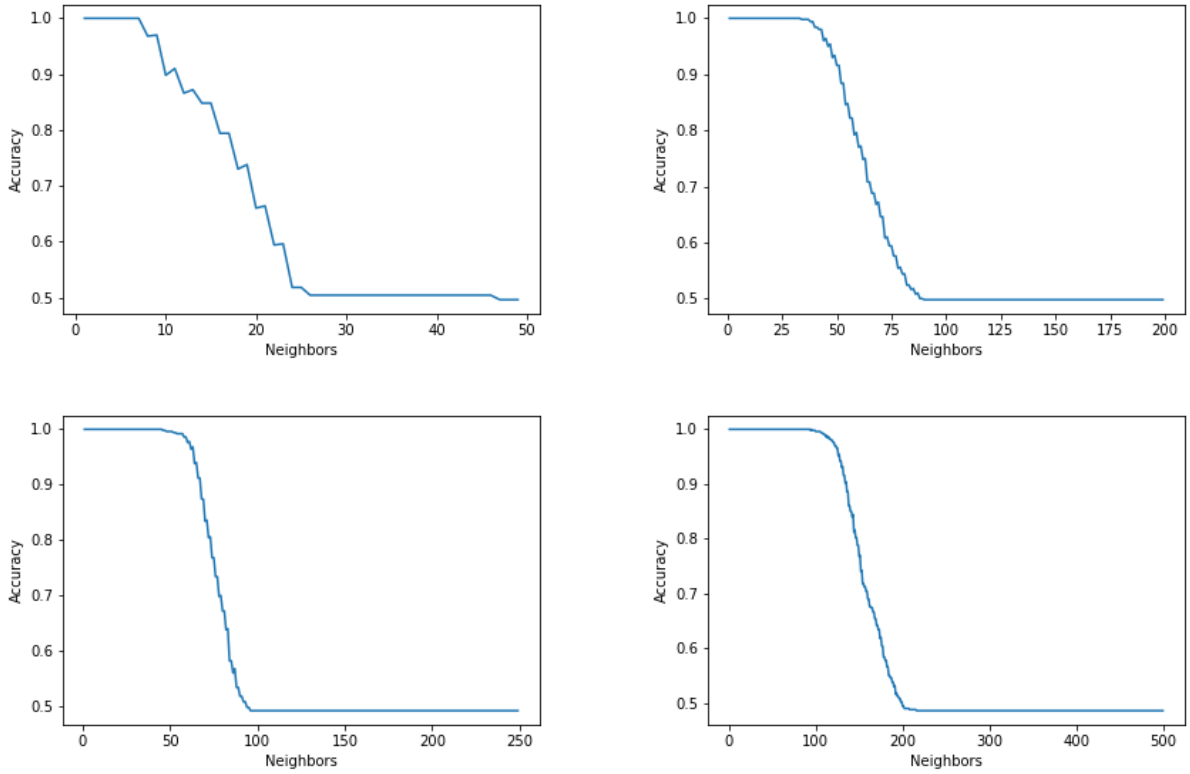

FIGURE 8 – Curves of test accuracies for every neighbors of a learning sample of sizes 50, 200, 250 and 500 for the first data set.

For the second data set (figure 9), even though the maximum is still between 1 and 3, the graphs have a different evolution.

2.d Discussion

Usually, when it comes to the K-nearest neighbors classifier, overfitting is very likely to happen when the number of neighbors is low. The different empirical experiments proved that, in this case and for those data sets, the optimal number of neighbor is very close to one. Considering this, our accuracies were very satisfiable on testing sets. From which we can only assume overfitting did not occur.

We believe it was reasonable to use a five-fold cross validation in order to determine the optimal value of `n_neighbor` and that we would have obtained similar results with a bigger learning sample as the behavior of the curves are almost the same.

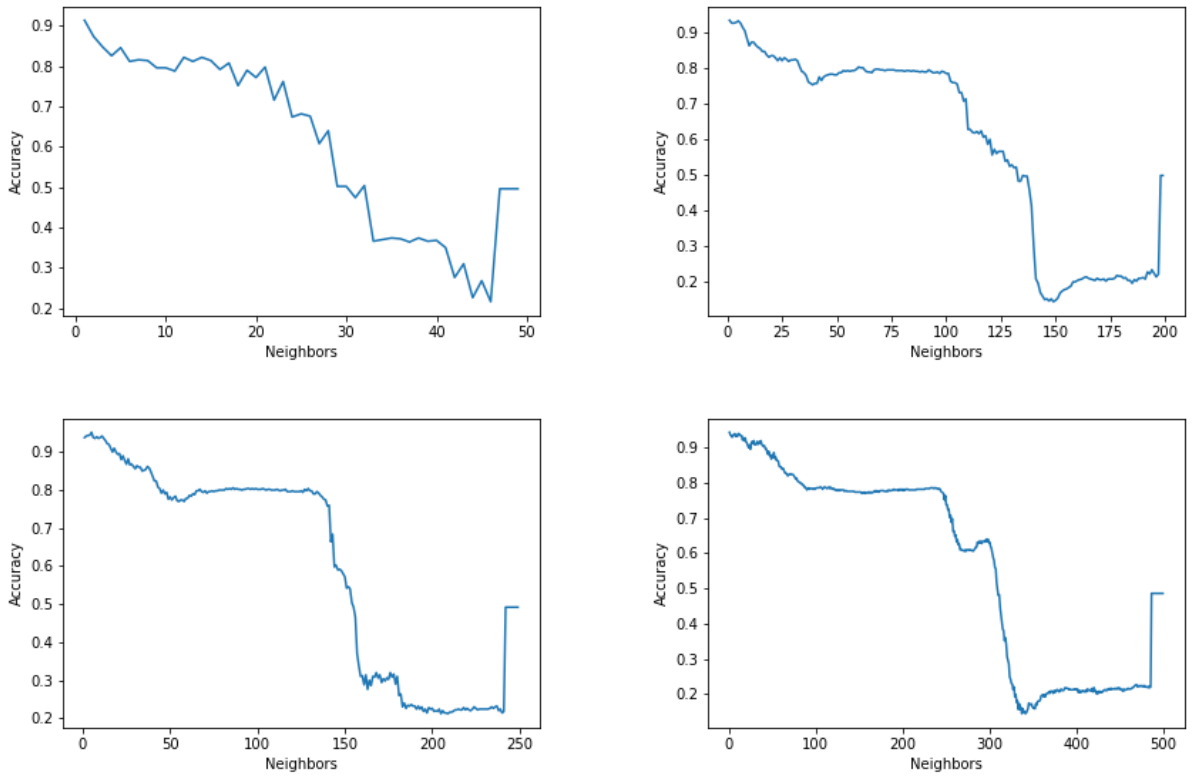

FIGURE 9 – Curves of test accuracies for every neighbors of a learning sample of sizes 50, 200, 250 and 500 for the second data set.

3 Residual fitting

In this section, we computed the prediction as described in the slides by iteratively calculating the different weights and then respectively multiply them with the attributes of the objects of the testing set. As expected with obtained values that are roughly in the interval $[0; 1]$ (some of them are above 1). To classify them, we calculated the sum of the squared weights and the points that have a value below that *limit* were considered as part of the first class and the ones above as part of the second class.

3.a Best weight proof

3.b Residual fitting

The two figures representing the probabilities for fitting the data do not represent valid decision boundaries as we can see in figure 10 below.

We did not managed to use it correctly and obtained a coherent rendering since our `predict_proba` function is obviously not well implemented. Nevertheless we achieved to compute coherent probabilities for each point classification by normalizing the output values.
From the numerical point of view, we obtain an accuracy of 57.72% for the first dataset and an accuracy of 50.09% on average. These accuracies tell us that the used method is clearly not suitable for this kind of classification problem, at least with this few number of attributes. Indeed, we can imagine that the three weights w_0, w_1 and w_2 are not enough to classify a whole set of points.

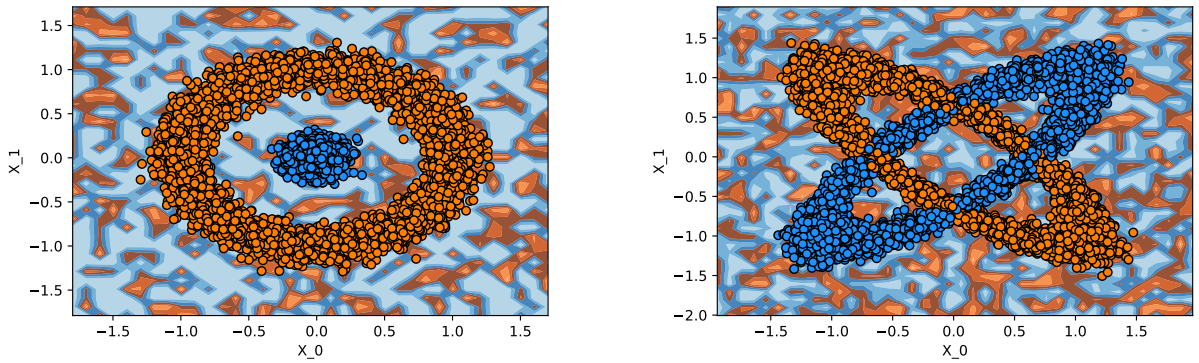

FIGURE 10 – Decision boundaries of the residual fitting

3.c Residual fitting on modified datasets

The illustrations of the decision boundaries for the two modified datasets are given below in figure 11.

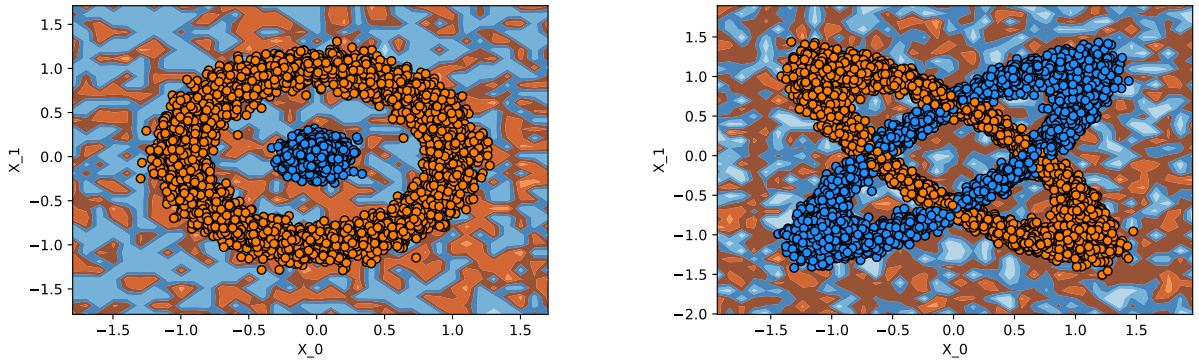

FIGURE 11 – Decision boundaries of the residual fitting of the two modified datasets

As we can see, the decision boundaries plotted by the given function `plot_boundary_extended` do not give a conclusive rendering as well. However, we obtained numerical results which suggest that we have created models that can more or less fit the data. Indeed, for the first dataset we obtained an accuracy of 99.88% and an accuracy of 78.75% in the second one.
We easily conclude that the additional weights offer more room to operate and so a greater precision to the fitting function.