UNIVERSITÉ DE LIÈGE

INGÉNIEUR CIVIL

October 18, 2020

ELEN 0062
Introduction to Machine Learning

Project 1

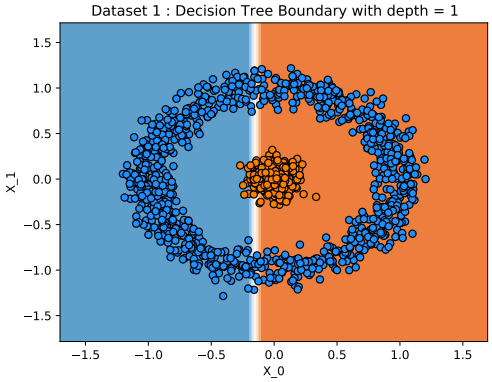
Classical algorithms

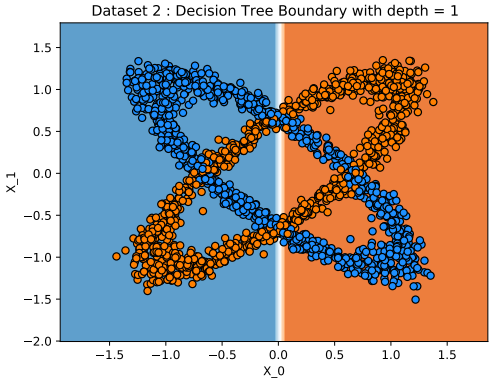
Maxime Lione - **s175606**
Cédric Siboyabasore - **s175202**

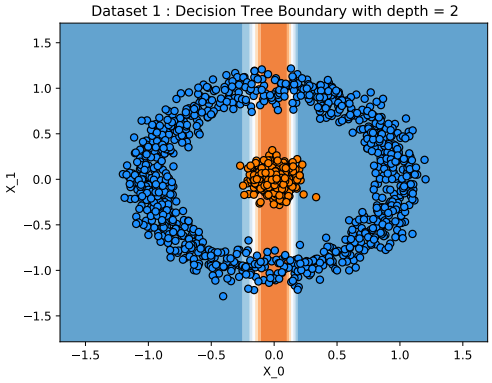
1 Decision tree (dt.py)

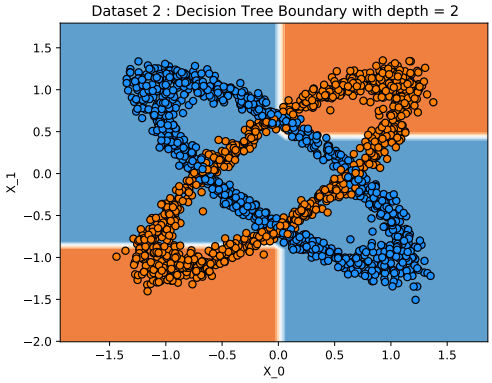
1.1 *For both problems, observe how the decision boundary is affected by tree depth)*

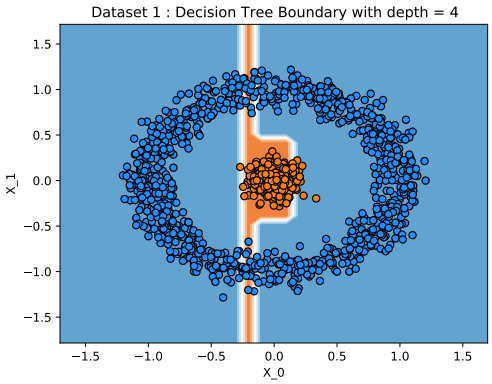
1.1. (a) *illustrate and explain the decision boundary for each depth*

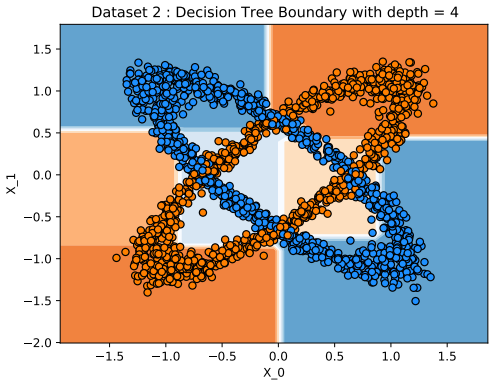


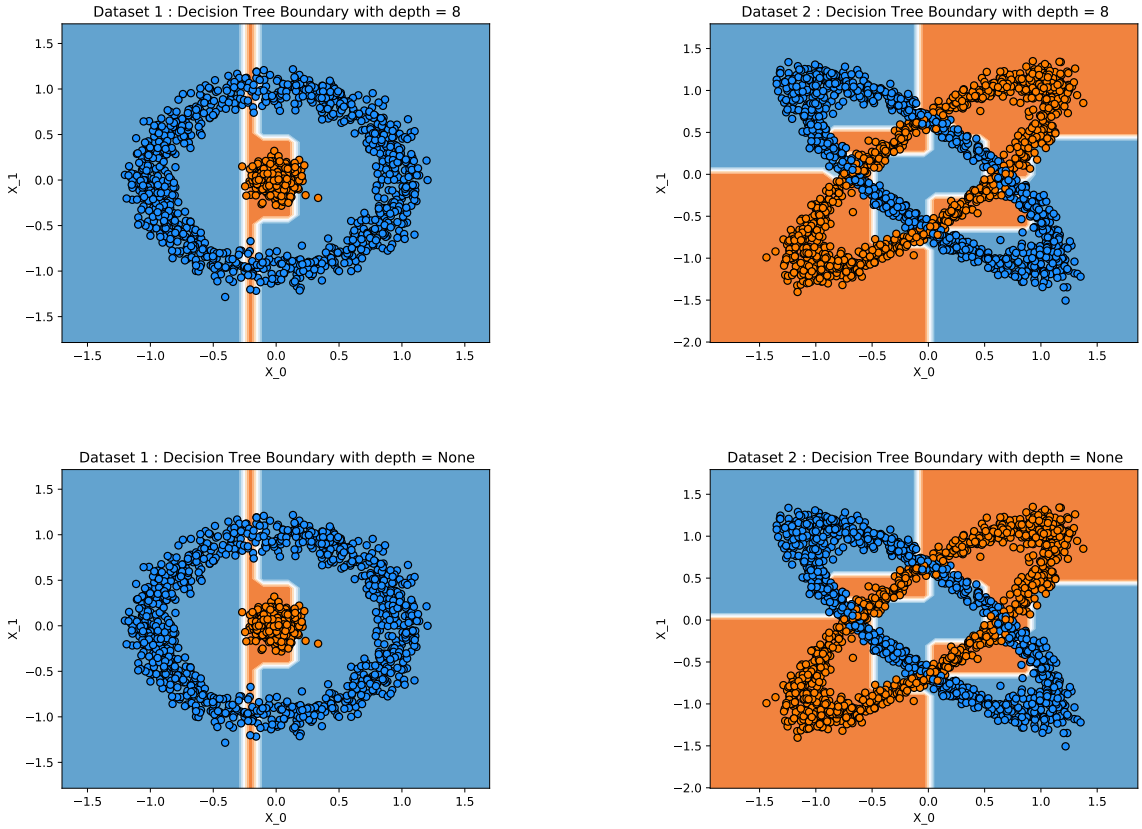












For each dataset and each depth $\{1, 2, 4, 8, None\}$, we built a decision tree model from the training set. For each of these trees we used the *plot_boundary* function to illustrate the decision boundary that can be seen on the 10 graphs above. To avoid overwhelming our figures with points and to have a better view of the classification boundaries, we only plotted 20% of the testing set.

We first notice that in every figure, the classification boundaries are aligned with the axis. This makes sense since a decision tree splits one attribute at a time. These axis-parallel splits partition the space into regions. We notice that the deeper the tree, the more regions we have in the space. This can be explained by the fact that at each depth, we split the nodes associated to that depth into subnodes. Therefore the number of regions can grow exponentially.

For dataset 1, we see that for a depth of 1 we have a lot of misclassifications of the test points (especially for the blue class test points). This is due to the fact that we only do 1 split partitioning the space in 2 regions. For a depth of 2, we have one more region and less misclassifications. As depth increases, we have better defined classification boundaries. Finally, we observe no difference between the decision boundary at depth 8 and the decision boundary with an unconstrained depth.

For dataset 2, we also observe major misclassification for depth 1 because we only have 1 splitting. For a depth of 2 we have less misclassification error. For a depth of 4, we see light blue and orange zones which means the decision tree model is not very confident about its predictions. For depths of 8 and more, the classification boundaries are better defined.

1.1. (b) *discuss when the model is clearly under- and over-fitting and detail your evidence for each claim*

For both datasets we observe underfitting when we have a depth of 1 thanks to the scatter plots. The model has not learned enough about the training sample. The model cannot partition the space into 2 regions with a single line to accurately classify the training points.
The underfitting is also present for depths 2 and 4 even though there are more delimitation lines.

We observe overfitting for dataset 2 when we have a depth greater or equal to 4 (depth of 8 or unconstrained); the choice of the regions and their delimitations in the features space feel unnatural, the model learned the training data "by heart" and may fail to predict future unlabeled observations.

1.1. (c) *explain why the model seems more confident when the depth is unconstrained*

When the depth is unconstrained, the decision tree has no limit to expand. So, it can grow until all the leaves are pure, which means that they contain only one class. Thus, the model can perfectly classify the training set : in a given region, the proportion of training objects of each class is pure : either 1 or 0. That's why the model is said to be "confident", even if, in practice, overfitting may interfere in the results.

1.2 *Report the average test set accuracies along with the standard deviation for each depth.*

	Dataset **1**		Dataset **2**	
Depth	mean	std	mean	std
1	71.026	0.656	49.996	0.185
2	92.514	0.762	65.678	12.429
4	98.142	0.487	79.744	2.012
8	98.218	0.490	87.054	3.218
None	98.052	0.541	89.160	2.804

Table 1: Average test set accuracies and standard deviation (%) over 5 generations for each depth

The average test accuracies of the model and the standard deviations over five generation of the data set confirm our observations for the underfitting and overfitting of the model :
Dataset 1 models have high accuracies (98%) for depths greater or equal to 4. We notice that the deeper the tree model the higher the accuracy average. The only exception is when we go from a tree of depth 8 to a tree of unconstrained depth. This means that 8 is the best depth we can get for a tree model on dataset 1.
For dataset 2 from depth of 8 to an unconstrained depth we observe a slight increase in average accuracy because the model is more confident for an unconstrained depth. However we don't have accuracies as high as the ones for dataset 1 for depths greater or equal to 4 because of the overfitting that we observed in **1.1. (b)**.

1.3 *Based on both the decision boundaries and the test accuracies, discuss the differences between the two problems.*

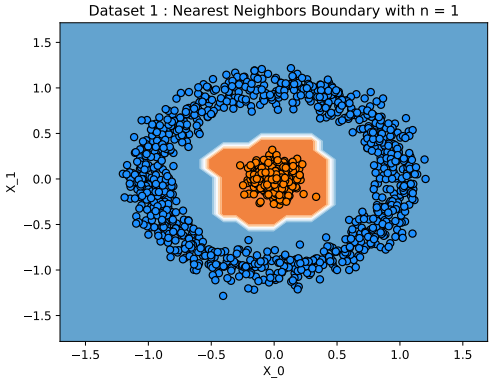
The main difference between the two datasets is the fact that dataset 2 is not aligned along the axes while dataset 1 is. This makes dataset 1 easier to manipulate since decision trees split the features space in axis-parallel lines. This is the reason why we reach higher accuracies with dataset 1 than with dataset 2.

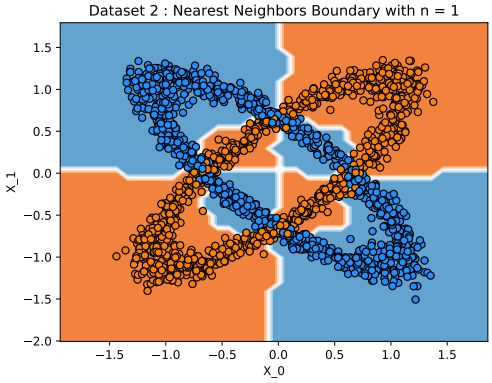
A larger learning sample could help the decision tree model reach higher accuracies on dataset 2. We arbitrarily raised the size of the learning sample in the *dt.py* file and immediately observed a higher average test accuracy for an unconstrained depth (average accuracy of 94.8% for a learning sample of size 25 000).

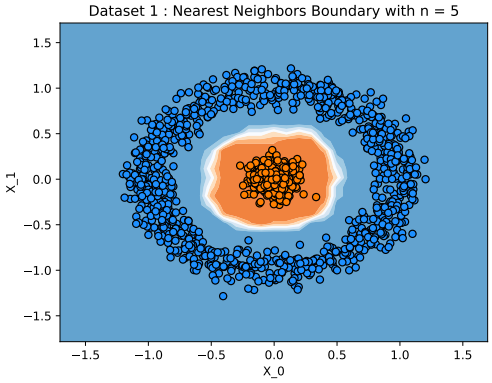
2 K-nearest neighbors (knn.py)

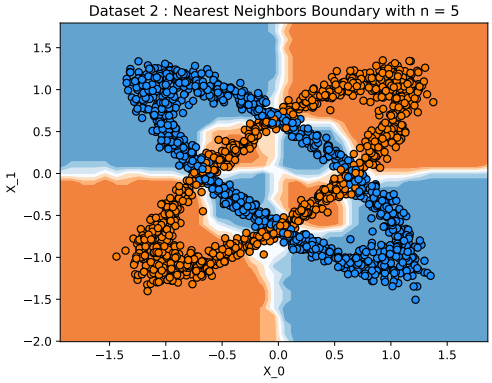
2.1 *For both datasets, observe how the decision boundary is affected by the number of neighbors*

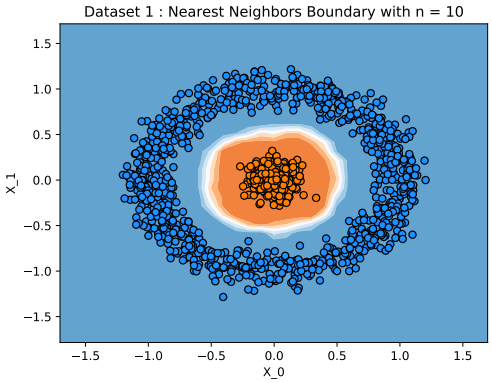
2.1. (a) *illustrate the decision boundary for* $\{1, 5, 10, 75, 100, 150\}$ *n_neighbors*

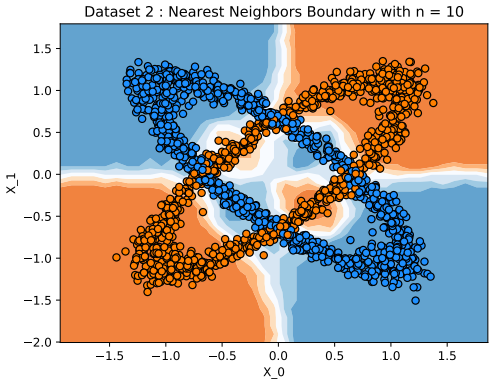


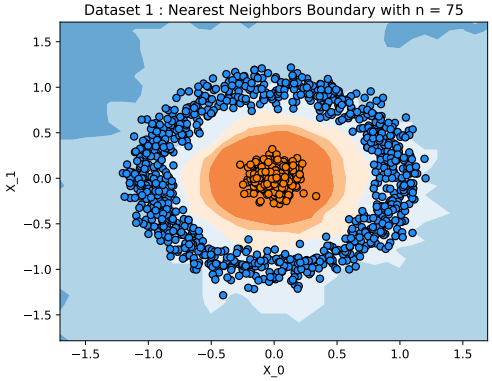


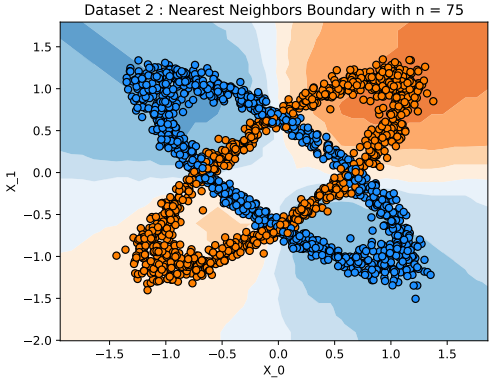


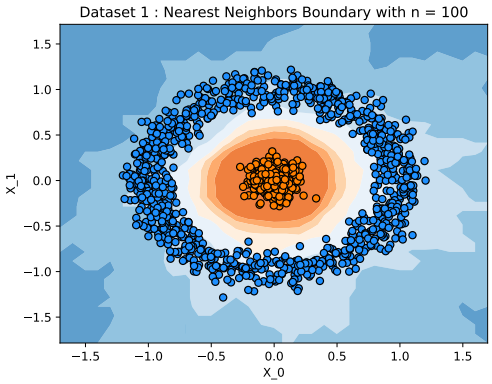


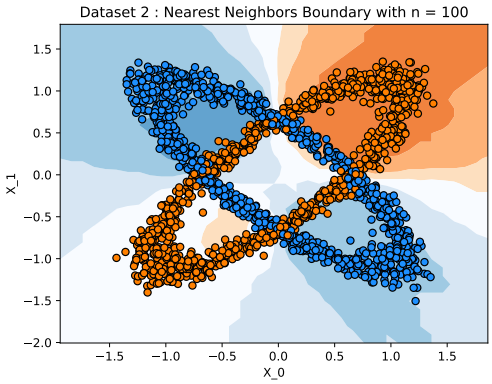

Dataset 1 : Nearest Neighbors Boundary with n = 10
X_0
X_1

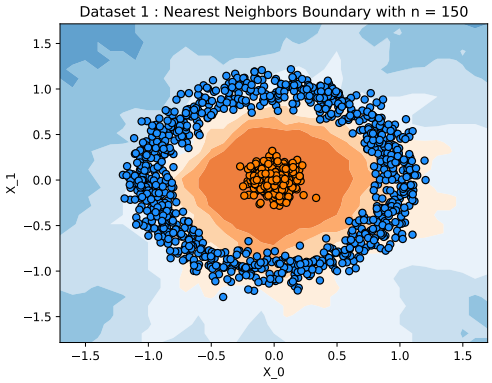

Dataset 2 : Nearest Neighbors Boundary with n = 10
X_0
X_1

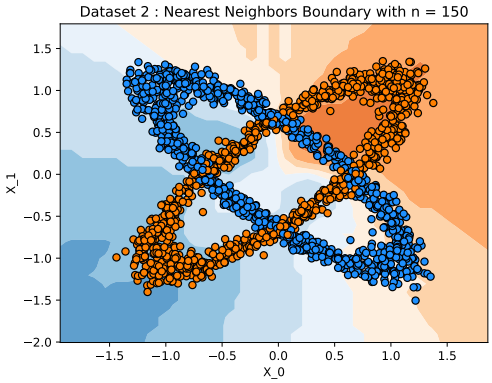

Dataset 1 : Nearest Neighbors Boundary with n = 75
X_0
X_1


Dataset 2 : Nearest Neighbors Boundary with n = 75
X_0
X_1


Dataset 1 : Nearest Neighbors Boundary with n = 100
X_0
X_1


Dataset 2 : Nearest Neighbors Boundary with n = 100
X_0
X_1





The twelve graphs above shows the decision boundary for several nearest neighbor models with different values of the n_neigbors parameter from 1 to 150, for both datasets. As we did in the Question 1, we only displayed 20% of the scatter points of the testing sample in order to make the graphs clearer.

2.1. (b) *comment on the evolution of the decision boundary wrt the number of neighbors*

As a major tendency, we can clearly see that as the number of neighbors is increasing, the models is becoming less confident about his predictions :

At first, as can be seen on the first two graphs above, when the model takes only 1 neighbor into account, it can only be 100% sure about the prediction even in zones where the classes intersect, which is a case of *overfitting*, because the model shouldn't be 100% certain in these zones.

Additionally, we can see on the graphs with n_neighbors = 5 and 10, that uncertainties start to appear. They appear earlier in zones where points from the 2 different classes are closer (and especially in zones that are equidistant to points from the 2 different classes, which seems logical). On graphs from Dataset 2, we can particularly see uncertainties that starts to appear soon around the intersections of the two ellipses. Otherwise, the overall boundary is accurate.

Furthermore, when n_neighbors is becoming too high, it takes into account too much neighbors. We can thus see on the graphs with n_neighbors = 75, 100 or 150 for both datasets that this leads to a drop in the certainty of the model (the zones are lighter). In this case, the model is taking into account neighbors than are too far and thus irrelevant for the current prediction.

Finally, the models using around 5/10 neighbors seem to be optimal for the number of training samples used here (250) and for both datasets, because they don't seem to be overfitting and the overall boundary seems accurate.

2.2 *Optimize the value of the n_neighbors parameter using a five-fold cross validation strategy and obtain an unbiased estimate of the test accuracy for the second dataset*

2.2. (a) *explain your methodology*

Since we perform a 5-cross validation we must divide dataset 2 into 5 consecutive subsets of equal length. To do so, we use the set of data X_train of size 250 generated by $make\_data2$. We store 5 in

a variable k and we iterate over it.

At each iteration $i \in [0,1,2,3,4]$ we consider the subset i as the test sample and the 4 other subsets as the learning sample. We copy the original dataset input into a *X_train* list and its output values into a *y_train* list. We pop the element $i*50$ of *X_train* corresponding to the first element of the test sample and append it to a list *X_test*. We repeat this action 50 times and end up with a *X_test* list of 50 elements containing the test sample. The *X_train* list now contains $250-50=200$ elements which correspond to the learning sample. We performed the same actions to get *y_train* list and *y_test* list.

We then build K-Neighbor classifiers for values of *n_neighbor* going from 1 to 150 (same upper bound as in **2.1. (a)**) and fit each model using *X_train* as training data and *y_train* as target values. At the next iteration i we use the next subset as training set and the 4 others as learning sample. At the beginning of the loop we copy the original dataset into *X_train* and its output values into *y_train* to get the test sample from the previous iteration back then use it as part of the training sample for the current iteration.

With the cross-validation we obtain an unbiased estimate of the test accuracy because we use the entire dataset instead of splitting the data into one learning set and one training set. The test sets we use are independent of the learning sets previously used.

2.2. (b) *report the optimal value of n_neighbors and the mean score over the five folds*

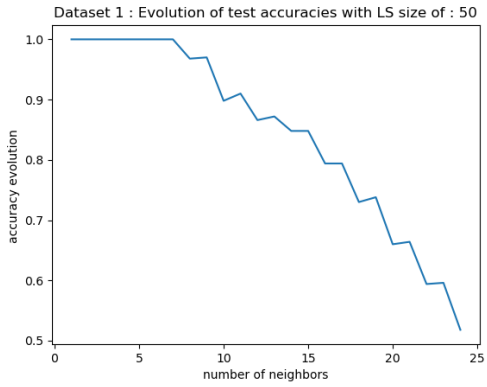
For each K-neighbors models we created while performing the cross validation we computed the mean accuracy of the model on the current *X_test* and *y_test* using the *score* method and store the result in an *accuracies* matrix for which the lines correspond to the considered test set and the columns correspond to the range of values of *n_neighbor* for the model. *accuracies* is thus a 5 *x* 149 matrix.
We perform the mean of this matrix over the considered training samples and obtain a list of 149 average test accuracies. We then take the maximum mean accuracy of that list which is **93.2%** and obtain the corresponding optimal *n_neighbor* which is **3**.
We notice on our graph below that for *n_neighbor* $= 5$ we have an average accuracy almost equal to the maximum value 93.2%. We have an accuracy of 92% for *n_neighbor* $= 5$.

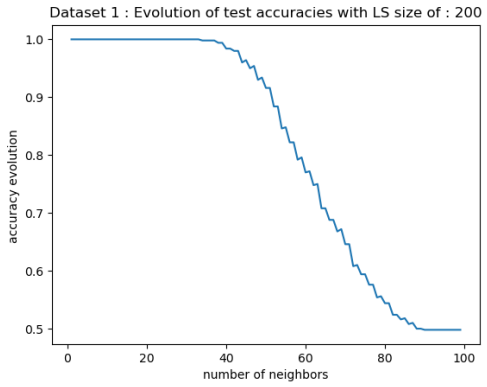
These results corroborate with our intuition in **2.1. (b)** because the optimal value $n\_neighbor = 3$ is neither too low to observe an overfitting like we had for $n\_neighbor = 1$ or too high to notice a lot of uncertainty like for $n\_neighbor = 75$.

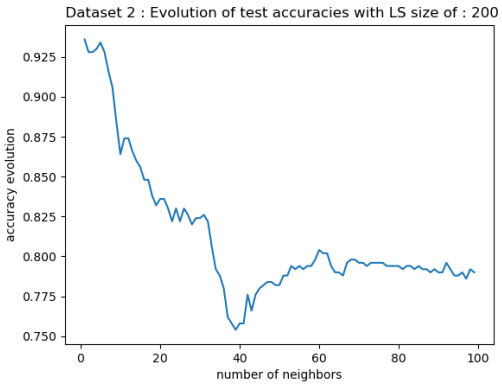
2.3 *For both datasets, observe the evolution of the optimal value of the number of neighbors with respect to the size of the learning sample set*

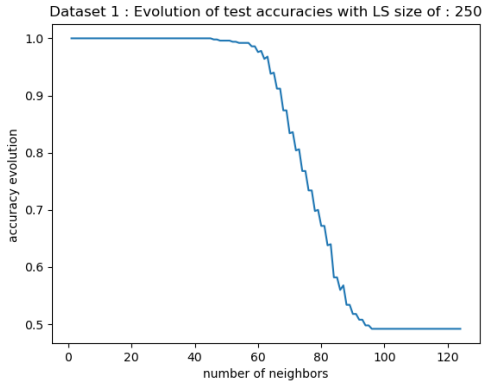
2.3. (a) *plot the evolution curve of test accuracies (on a TS of size 500) for every possible number of neighbors for LS of sizes* $\{50, 200, 250, 500\}$

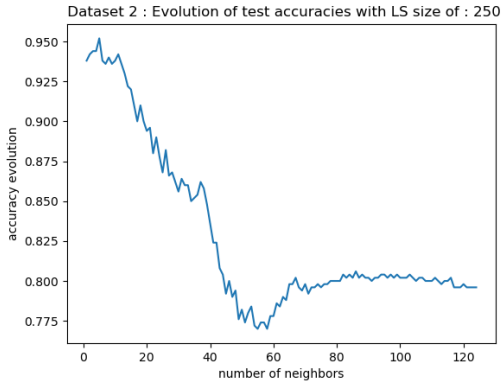
As we've seen in question 2.1, taking into account a too high number of neighbors leads to a lot of uncertainties. Thus, for this question, we only plot the evolution of test accuracies until a number of neighbors equal to the half of the learning sample size (it also prevents from doing useless consequent computations). We obtain the following graphs :

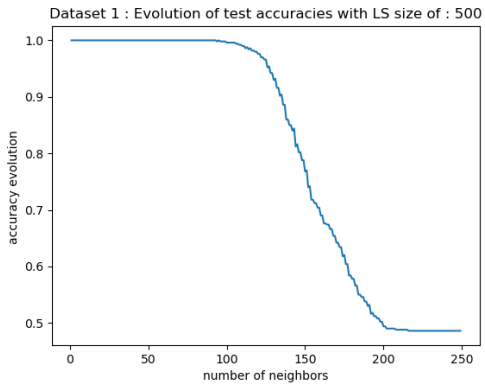


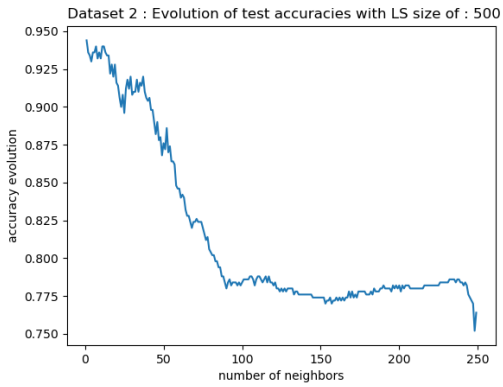










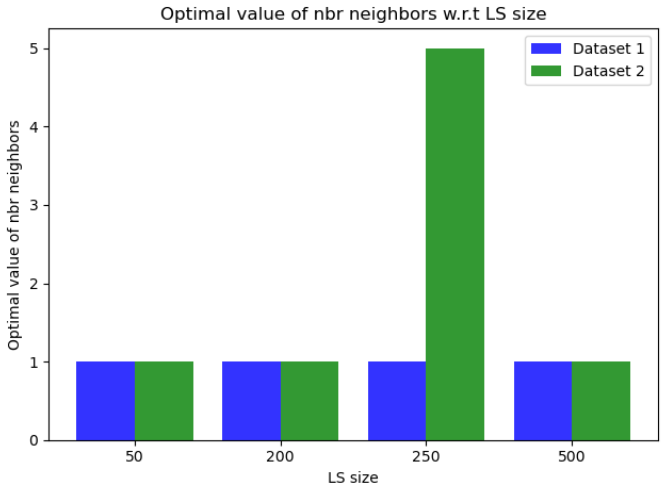


First of all, when the number of neighbors considered increases, for both dataset and any LS size, the accuracies globally decrease, which was expected following what we learned in question 2.1. Moreover, it can be seen that the curves for all LS sizes for dataset 1 remain high (the accuracies remain high) for higher numbers of neighbors than for dataset 2. This seems logical (and could also be observed on the graphs from question 2.1(a)) because in dataset 1, the scatter points corresponding to each class are better separated and don't cross each others, thus considering a number of neighbors that's a bit higher than one still leads to accurate boundaries. Uncertainties appears sooner in models for dataset 2 because the scatter plots for each class are mixed up.

Nevertheless, we can see that the drop in the accuracy appears for a higher number of neighbors when the LS size is higher too (which is easier noticeable on graphs for dataset 1), which also seems logical because as the learning set increases the global density of data points for a given domain is higher, thus more accurate neighbors are available.

2.3. (b) *plot the optimal value of n_neighbors with respect to the LS size*

For each dataset and each learning sample size we have an optimal value of $n\_neighbor = 1$. The only exception is the optimal $n\_neighbor = 5$ observed for dataset 2 when the learning sample has a size of 250. These results are presented on the following graph :



We can see that the optimal number of neighbors in most cases is equal to 1 because, as had also been noticed with the decision boundary graphs from question 2.1(a), when the model takes only 1 neighbor into account, it assigns the class of only one neighbor and is thus 100% sure about the prediction. The boundary is sharp, but the model is overfitting the data.

2.4 *Given the results of question 2.3 and a LS of size 250, what do you think of using five-fold cross-validation to determine the optimal value of n_neighbors as you did in question 2.2?*

We may have obtained an optimal $n\_neighbor = 3$ parameter with the cross-validation in question **2.2** but we had also noticed an another high accuracy of $n\_neighbor = 5$ when we displayed the evolution of the mean accuracy which corroborates with the optimal $n\_neighbor = 5$ found in question **2.3**.

With the cross-validation method we test on all the arrangements of the 5 consecutive folds while in question **2.3** we split the data into one learning set and one test set so we only get an accuracy for that arrangement.

For that reason the five-fold cross-validation seems more accurate because it is unbiased.

3 Residual fitting (residual.py)

3.1 *Proof that the best weight ω_k for the attribute a_k introduced in the model at step k is $\rho_{a_k,\Delta_k y}\sigma_{\Delta_k y}$*

The best weight ω_k for the attribute a_k introduced in the model at step k is the weight ω_k^*, that leads to the least mean square error solution. To be able to define properly this square error (SE) :

- let's pose : $a_0(o) = 1, \forall o$
- let's define :
  - $\mathbf{a'}(o_i) = (a_0(o_i), a_1(o_i), ..., a_n(o_i))^T$
    (we add a column in the dataset as if we were adding an attribute with value = 1 for all objects, and we have this new set, composed of the original set + the first element that we added)
  - $\omega' = (\omega_0, \omega_1, ..., \omega_n)^T$
    (which is the original vector ω "extended")

Thus the square error (SE) at o_i can be written as :

$$SE(o_i, \omega') = (y(o_i) - \widehat{y}(o_i))^2 = (y(o_i) - \omega'\mathbf{a'}(o_i))^2$$

where $y(o_i) - \widehat{y}(o_i)$ is the difference between the real output and the estimated output.

Then, by doing the sum of this error for all the objects of the dataset, we obtain the total squared error (TSE), written as :

$$TSE(LS, \omega') = \sum_{i=1}^{N} (y(o_i) - \omega'^T \mathbf{a'}(o_i))^2$$

So, we want ω' such as the TSE is minimum. Thus, assuming only one input, the solution is computed as :

$$(\omega_0^*, \omega_1^*) = \arg\min_{\omega_0, \omega_1} \sum_{i=1}^{N} (y(o_i) - \omega_0 - \omega_1 a_1(o_i))^2$$

By canceling the second derivative with respect to ω_0 and ω_1, we have :

$$\omega_1^* = \frac{\sum_{i=1}^{N} (a_1(o_i) - \overline{a}_1)(y(o_i) - \overline{y})}{\sum_{i=1}^{N} (a_1(o_i) - \overline{a}_1)^2} = \frac{cov(a_1, y)}{\sigma_{a_1}^2}$$

$$\omega_0^* = \overline{y} - \omega_1^* \overline{a}$$

where $\overline{a}_1 = N^{-1} \sum_{k=1}^{N} a_1(o_k)$ and $\overline{y} = N^{-1} \sum_{k=1}^{N} y(o_k)$

We see that when ω_1^* is defined, ω_0^* is easily found.

Finally, substituting the above into $y(o) = \omega_0^* + \omega_1^* a_1(o)$, we have :

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

with $\rho_{a_1,y}$ the Pearson correlation between a_1 and y, and σ_y, σ_{a_1} the standard deviations of y and a_1, all computed on the LS.

We can see that the dataset had been "standardized" : the new output has a mean of 0 and the standard deviation is equal to 1.

As we assume attributes of zero mean and unit variance, the equation above can be written such as :

$$y(o) - \bar{y} = \rho_{a_1,y}\sigma_y \ \ a_1(o)$$

$$(y(o) = \rho_{a_1,y}\sigma_y \ \ a_1(o))$$

$$\Delta = \rho_{a_1,y}\sigma_y \ \ a_1(o)$$

Where $\rho_{a_1,y}\sigma_y$ is the best weight ω_1 for the attribute a_1. This logic can be reproduced for any value of k.

3.2 *Implement the algorithm as described in the slides*

We implemented the algorithm of *Residual Fitting* as described on the slide 20 of Lecture 3, in the method *fit()*, in which the weight ω_k is computed for each step k. To do so :

- We started by computing ω_o for the no-variable case
- We introduced attributes with assumed zero mean and unit variance one at the time :
  - The residual is defined at step k by :

$$\Delta_k y(o) = y(o) - \omega_0 - \sum_{i=1}^{k} -1\omega_i a_i(o)$$

  - The best fit of residual with only attribute a_k is computed with :

$$\omega_k = \rho_{a_k,\Delta_k y}\sigma_{\Delta_k y}$$

Thanks to our implementation of the residual fitting algorithm we fitted linear regression models to our two datasets.
We obtained an intercept of $\omega_0 = 53.6\%$ for both models. We also got $\omega_1 = 23.2\%$ for both regression models but $\omega_2 = 1.4\%$ for dataset 1 and $\omega_2 = 10.37\%$ for dataset 2.
After being fitted, the model can then be used to predict the class of samples and alternatively, the probability of each class can be predicted.

3.3 *Learn a residual fitting model on both datasets*

3.3. (a) *illustrate the decision boundaries*

3.3. (b) *report the test accuracies*

3.4 *Learn a residual fitting model on a modified version of the second dataset that includes three new attributes in addition to the two original ones*

3.4. (a) *illustrate the decision boundaries*

3.4. (b) *report the test accuracies*