Faculty of Applied Sciences

Introduction to machine learning

ELEN0062-1

Project 1: Classification algorithms

Authors :
Delcour Florian - s181063
Lewin Sacha - s181947
Makedonsky Aliocha - s171197

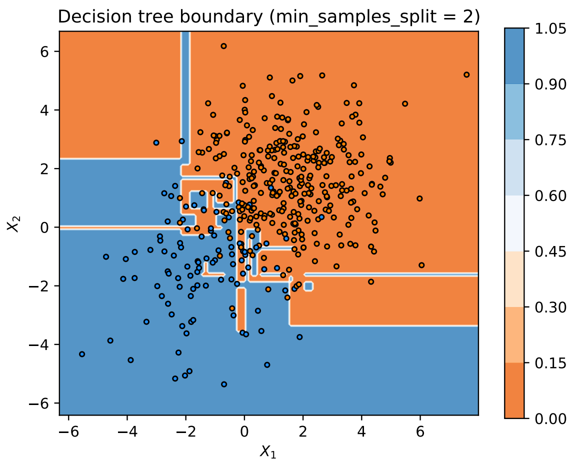
Instructors :
Wehenkel Louis
Geurts Pierre

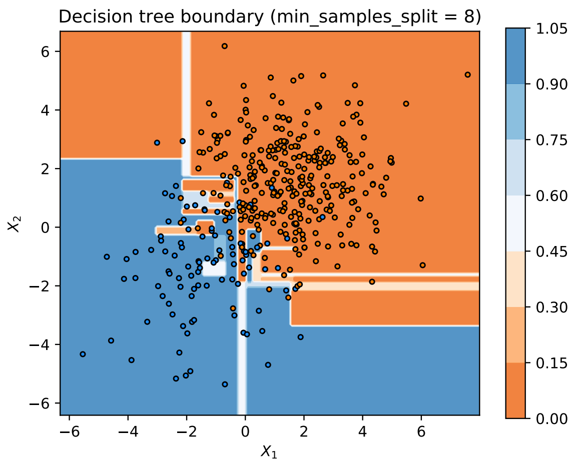
October 2021

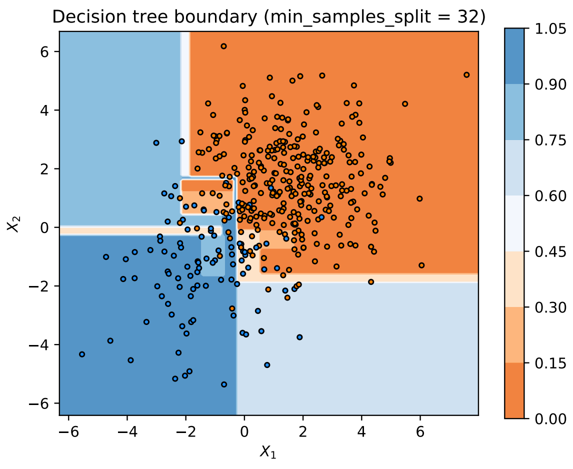
1 Decision tree

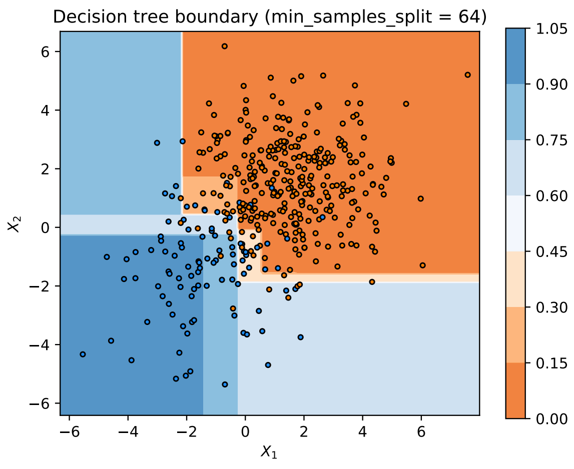
1.1 Decision boundary affected by tree complexity

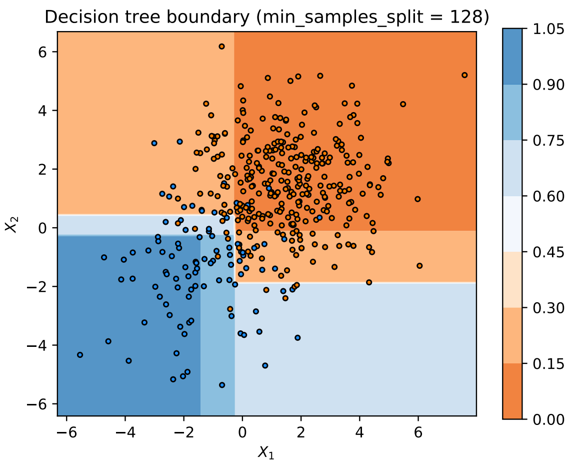
a) In a decision tree, points are split according to the value of their attributes. It leads to a decision boundary, which is the boundary between the different types of point.
In our case we have positive (blue) and negative (orange) points, and we want a boundary between those two types to have as few errors as possible. To do so, we create a decision tree with a certain value of `min_samples_split`, which is the number of objects needed to split an internal node. Lower this value, higher the accuracy, but with a risk of over-fitting the training set of samples, resulting with bad predictions for a new set of data. Boundaries are always aligned to the axes because decisions are made one variable at a time.

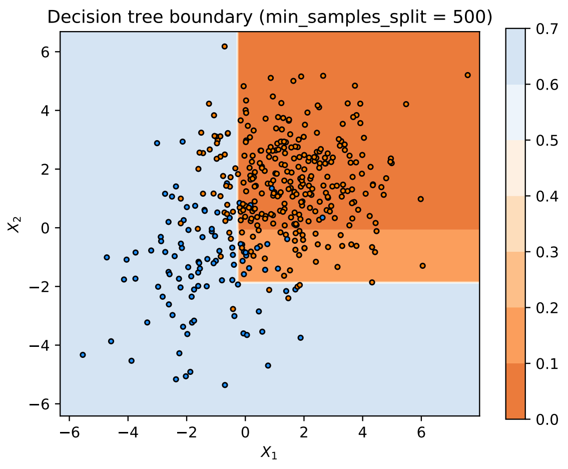












b) For `min_samples_split` equal to 500 we clearly have under-fitting since the boundary do not fit the samples at all, there are a huge number of errors, and the confidence is very low.
For this parameter equal to 2, we see over-fitting, because there are many zones where we have an orange boundary even though there are no orange point at all in this zone, and sometimes this boundary is far from the location of the orange points, it means that in the training sample there were some orange points there which moved the boundary, but the prediction is bad for other datasets. Indeed, in the training phase, noise is also fitted. Same for the blue zone at the bottom right. The confidence is also way too high seeing the errors.
For the value 8 it's better than for 2 but it's still not really good, there is over-fitting too.

c) The confidence is inversely proportional to the proportion of sample in a leaf. Indeed, if `min_samples_split` is lower, more and more internal nodes will expand and thus it will give more pure leaves which lead to more confidence in the decision.

1.2 Visual guesses

a) For the boundaries we can guess the shape of the decision tree with the levels of confidence in the plot (high level $\rightarrow$ lot of leaves), and with the shape of the boundaries (lot of small zones and thin zones $\rightarrow$ lot of leaves).

b) When the problem is unbalanced, the levels of confidence for the most populated part are higher than for the other one (in the plots we see that the levels of confidence for negative samples were generally higher than for positive ones). The density of points is also a clue for an unbalanced problem.

1.3 Test accuracies

The average accuracy over five generations (using 0, 1, 2, 3, 4 for the seeds) for each value of `min_samples_split` is respectively :

$$[0.8827, 0.8831, 0.9012, 0.9103, 0.9030, 0.8637]$$

and the standard deviation for each of those values is

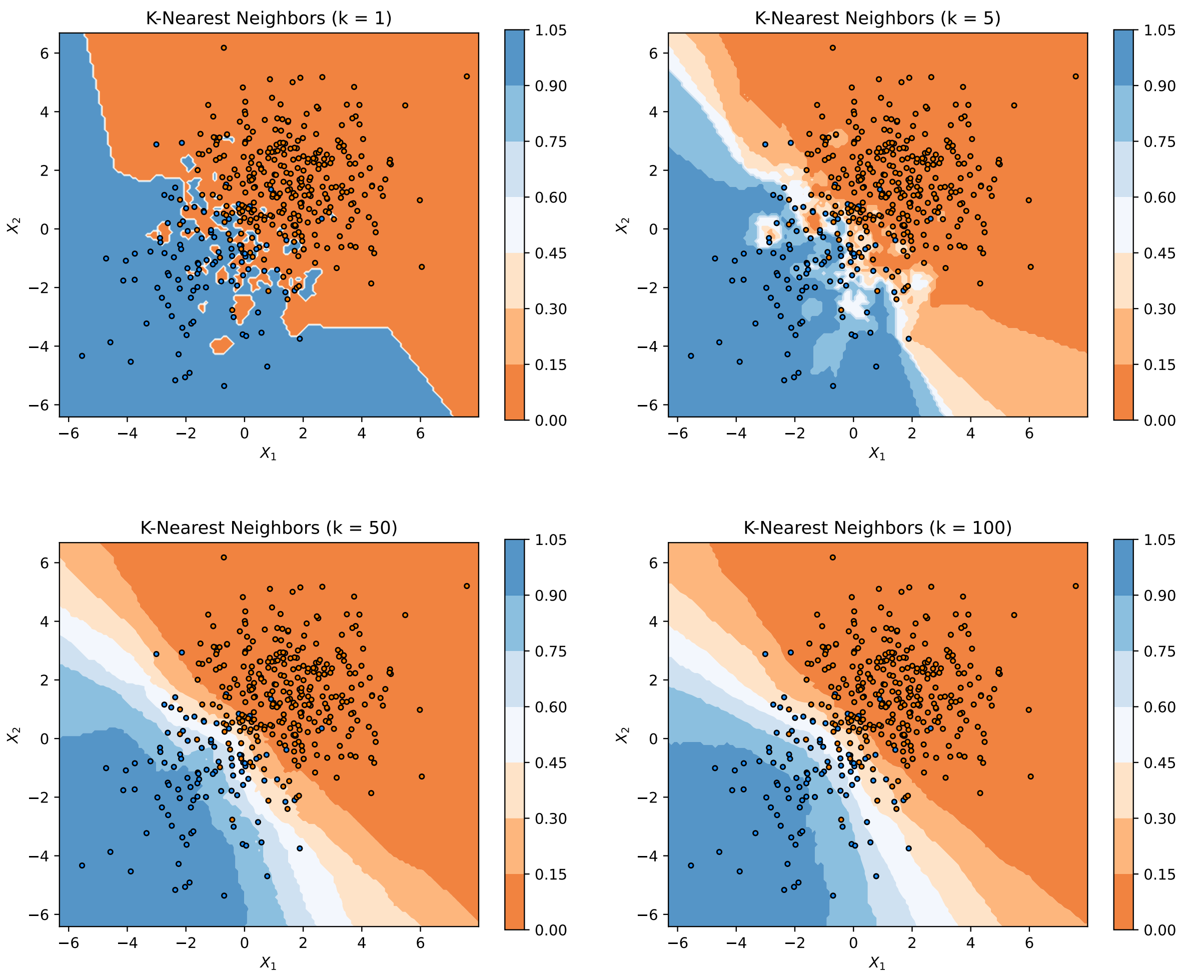
$$[0.01688, 0.01800, 0.01005, 0.01089, 0.01272, 0.01402]$$

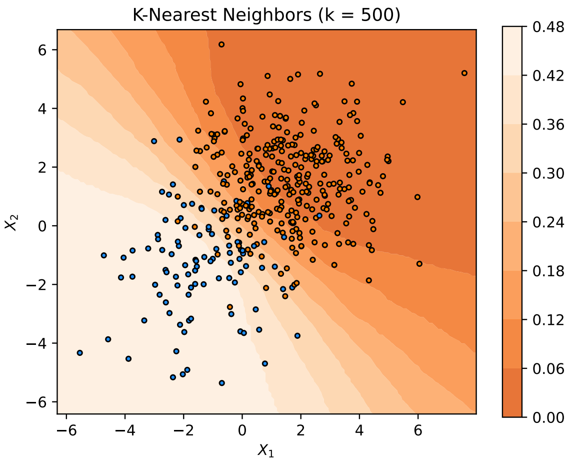
We can see that the accuracy score is lower for the values 2, 8, 500 of `min_samples_split`, as well as the standard deviation is higher for those values too. So it indeed represents the under-fitting and over-fitting we could see in the graphs of the question 1.a)

2 K-nearest neighbors

2.1 Decision boundary affected by the number of neighbors

a)





b) ✻ For $k = 1$, we see that the decision boundary is sharp and the predictions are confident. Indeed, the areas are either strong orange or strong blue. There is also some overfitting represented by small areas with no scatter points in it. Assigning the class of only one neighbor, the model is strongly confident and can also fit noise.

✻ For $k = 5$, the overall boundaries are reasonable and uncertainty is appearing near the boundaries (orange area mainly). We can still see a bit of overfitting.

✻ For $k = 50$ or 100 , uncertainty has increased a bit for blue area but decreased for the orange one (more confident for orange). Boundaries between each area are almost sharp. Blue area has decreased significantly because there are much more orange points in the dataset.

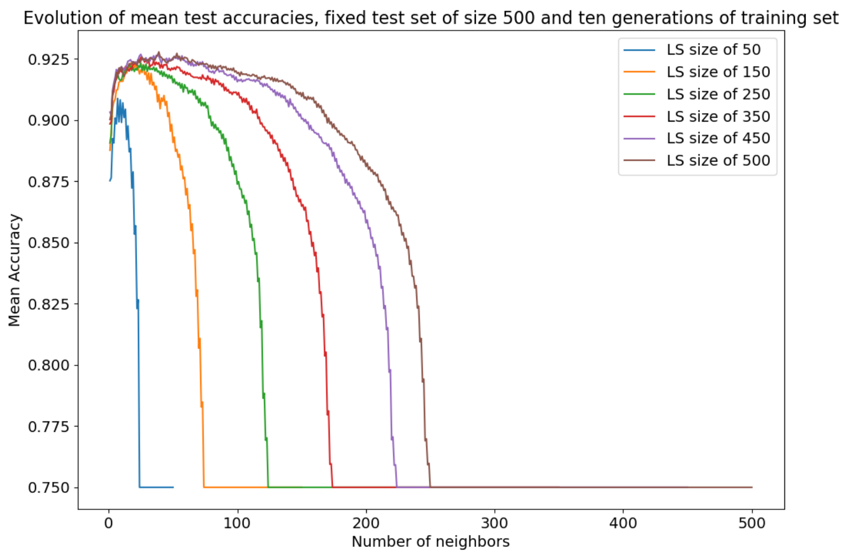
✻ For $k = 500$, Blue area has totally disappeared. At this point, if we increase k, the boundary will disappear leading to a uniform area corresponding to the orange class. Indeed, there are much more orange points than blue ones in the dataset, then if we use a lot of points to make a point prediction, chances are high of getting an orange point.

2.2 Cross Validation

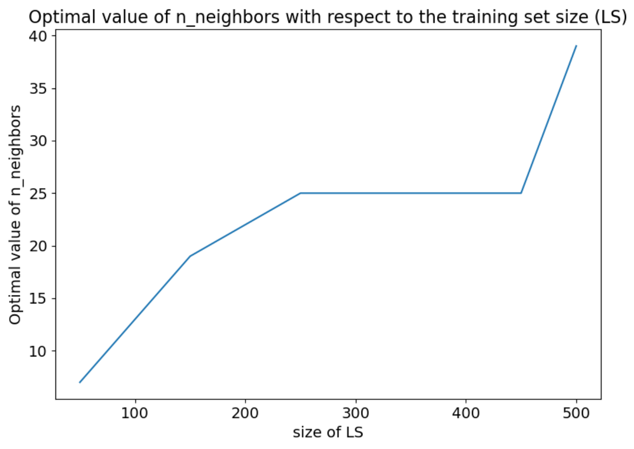
a) In 10-fold cross validation, the training set (1000 samples) is randomly partitioned into 10 equal sized data subset (100 samples). It is important to not use the whole dataset because we need an unbiased estimate for the final test accuracy. Then we choose the first data subset as the validation data for testing the model and the remaining nine data subsets are used to train the model. We build the KNN model using the training data and we get the accuracy of the obtained model using the testing data. Afterwards, this process is repeated until the 10 data subsets have been used as validation data. We obtained a list of 10 accuracies from which we can compute the mean. Finally, we execute this whole method for values $(1; 5; 50; 100; 500)$ of `n_neighbors` and select the optimal value that maximizes the mean accuracy. Then we can refit our model with this optimal value and compute the accuracy of the final model.

b) Using the algorithm above, we obtained the optimal value `n_neighbors` = 5 and a corresponding mean accuracy of about 92.9%. Refitting our model with `n_neighbors` = 5, we obtain an accuracy of 91.15%. As we seen in Q2.1.b, $k = 1$ is too confident on its predictions and makes errors (fits noise). For $k = 5$ and $k = 50$, there is some uncertainty but the model is still rather confident (blue area decreased but orange area increased. For these reasons, it is reasonable to think that the optimal value of `n_neighbors` is between 5 and 50 and this corroborates our result.

2.3 Optimal value of `n_neighbors`



We easily see that the optimal value of neighbors depends on the number of training samples. Indeed, on the first graph, the curves are offset : more the training set size is large, more the mean accuracy remains high while the parameter `n_neighbors` is increasing.
All the curves start to increase a bit to reach their maximum value and then decrease, the greater the neighbors, the less the slope. Finally they reach a similar point, whatever the number of neighbors, which is a mean accuracy of about 75%. This result is expected because they are three times more orange points than blue ones in the dataset. Then if the learning set size is equal to `n_neighbors`, the prediction will be an orange point, so we have 25% of errors for the whole dataset.



On this graph, we can see that the larger the size of the learning set, the larger the optimal value of `n_neighbors`.

3 Logistic regression

1. To prove that the boundary is linear, it suffices to show that the gradient of $\mathrm{P}(x_1, x_2)$, $\nabla\mathrm{P}$, is constant for all x_1, x_2.
   The expression of P in terms of x_1 and x_2 is given by:

   $$\mathrm{P}(x_1, x_2) = \frac{1}{1 + \exp\left(-w_0 - w_1 x_1 - w_2 x_2\right)}$$

   Differentiating this with respect to x_1 or x_2 gives:

   $$\frac{\partial \mathrm{P}}{\partial x_i} = \frac{w_i \exp\left(-w_0 - w_1 x_1 - w_2 x_2\right)}{\left(1 + \exp\left(-w_0 - w_1 x_1 - w_2 x_2\right)\right)^2} \text{ for } i \in \{1, 2\}$$

   These are the components of the gradient of P. Taking its slope therefore gives $\dfrac{w_2}{w_1}$ which is constant for each x_1, x_2. This means the direction of the gradient is the same over $\mathbb{R}^2$, resulting in the linear boundary.

2. Starting from the expression of $\mathcal{L}$, we have that:

   $$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log\Big(P\left(Y = y_i | \boldsymbol{x}_i, \theta\right)\Big) = -\frac{1}{N}\log\left(\prod_{i=1}^{N} P\left(Y = y_i | \boldsymbol{x}_i, \theta\right)\right)$$

   So minimizing the loss function is the same as maximizing the product inside the logarithm above. We also have that:

   $$P\left(D|\theta\right) = \prod_{i=1}^{N} P\left(Y = y_i | x_i, \theta\right)$$

Therefore, maximizing the likelihood of the observed data (P$(D|\theta)$), will maximize the argument of the logarithm in the expression of $\mathcal{L}$ described above, and will therefore minimize the negative log-likelihood function.

3. To compute the gradient of the negative log-likelihood, given by we have to derivate partially the equation with regard to each of the features of θ, which are w_0, w_1 and w_2. We assume that the first n samples are positive and the N-n last samples are negative so we can divide the sum into 2 parts, so we have

$$\mathcal{L}(\theta) = -\frac{1}{N}\left(\sum_{i=1}^{n} \log P\left(Y = +1|\boldsymbol{x}_i, \theta\right) + \sum_{i=n+1}^{N} \log P\left(Y = 0|\boldsymbol{x}_i, \theta\right)\right)$$

First we show the computation for $Y = +1$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_0} &= -\frac{1}{N}\sum_{i=1}^{n} \frac{\left(1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)\right)\exp\left(-w_0 - w_1x_1 - w_2x_2\right)}{\left(1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)\right)^2} \\
&= -\frac{1}{N}\sum_{i=1}^{n} \frac{-1 + 1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)}{1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)} \\
&= \frac{1}{N}\sum_{i=1}^{n}\left(\frac{1}{1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)} - 1\right) \\
&= \frac{1}{N}\sum_{i=1}^{n}\left(P(Y = +1|x_i, \theta) - y_i\right)1
\end{aligned}$$

With y_i being 1 since we computed for Y = +1 and the one at the end is the first feature of $\boldsymbol{x}_i'$. We quickly see that derivating with regard to w_1 we will get

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{N}\sum_{i=1}^{n}\left(P(Y = +1|x_i, \theta) - y_i\right)x_1$$

and the same with w_2 instead of w_1 and x_2 instead of x_1.
For $Y = 0$, we have the property $P(Y = +1|\boldsymbol{x}_i, \theta) = 1 - P(Y = 0|\boldsymbol{x}_i, \theta)$ since it's a binary problem, so we have the following computation :

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_0} &= -\frac{1}{N}\sum_{i=n}^{N} \frac{\left(1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)\right)}{\exp\left(-w_0 - w_1x_1 - w_2x_2\right)}\frac{\left(-\exp\left(-w_0 - w_1x_1 - w_2x_2\right)\right)}{\left(1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)\right)^2} \\
&= \frac{1}{N}\sum_{i=n}^{N} \frac{1}{1 + \exp\left(-w_0 - w_1x_1 - w_2x_2\right)} \\
&= \frac{1}{N}\sum_{i=n}^{N}\left(P(Y = +1|x_i, \theta) - y_i\right)1
\end{aligned}$$

with $y_i = 0$ since we are computing for the samples with $Y = 0$, and the 1 at the right is there for the same reason than before. We thus have the same results than for $Y = +1$, we can regroup the sums to have the global sum from 1 to N, and we thus find the result asked, which is that

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Big(\left(P(Y = +1|x_i, \theta) - y_i\right)\Big)\boldsymbol{x_i'}$$

with $\boldsymbol{x_i'} = (1, x_i^{(1)}, x_i^{(2)})^T$

4. Initially we want $P(Y = 0) = P(Y = 1) = \frac{1}{2}$, so every sample has the same probably to be positive or negative, because we don't want the classifier to be already influenced by one type since we want it to be general. The only way to assure that is to take $\theta = (w_0; \boldsymbol{w}^T) = (0; [0, 0])$ so the argument of the exponential is 0 and we have what we wanted.
5. Here is the boundary with a seed equal to 0, 15 iterations and a learning rate $\eta = 0.7$.

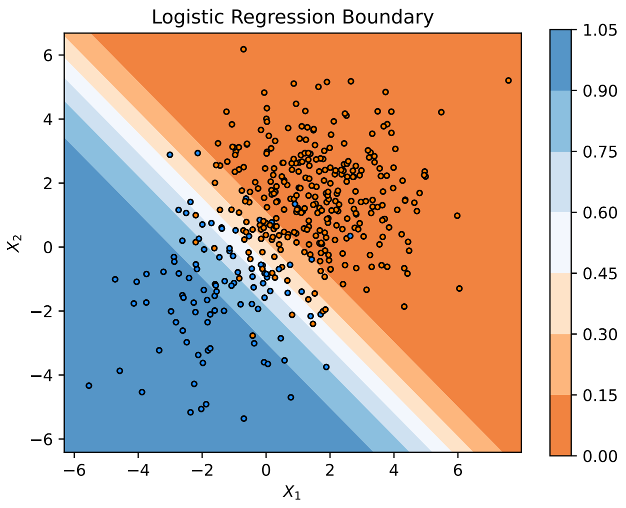


Figure 1: Boundary with 15 iterations and $\eta = 0.7$

6. Still using 15 iterations and $\eta = 0.7$, and 5 generations of the dataset (using seeds from 1 to 5), we obtain a mean accuracy score of 92.48% with a standard deviation of 0.45%.
7. On Figure 2 is represented the impact on the error rate of the number of iterations.

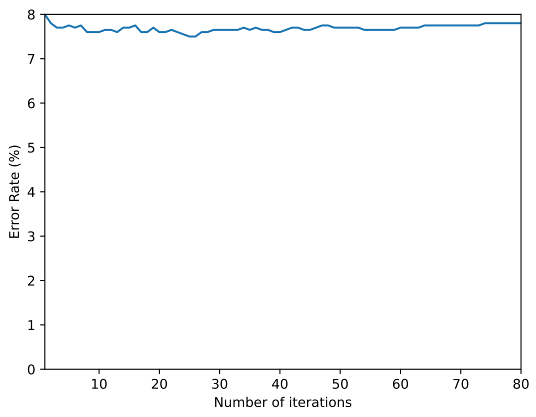
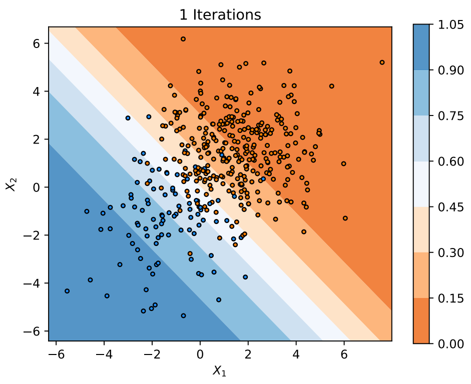


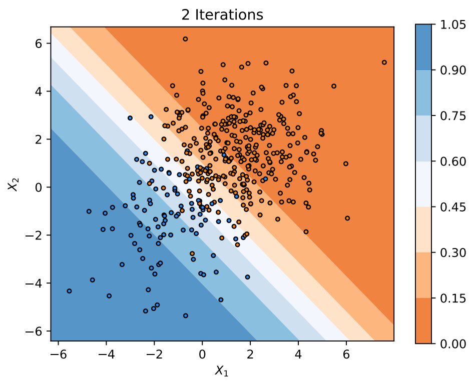
Figure 2: Error rate depending on the number of iterations

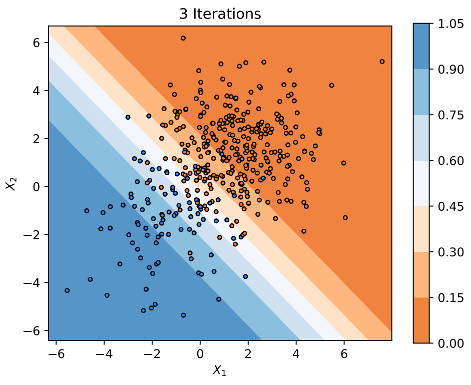
As one can see, the error rate does not change much depending on the number of iterations. As soon as there are multiple iterations, the error rate gets stable, hitting its lowest value, by not much, at around 25 iterations and then going slightly higher as iterations increase. Therefore, it is not necessary to use a high number of iterations.

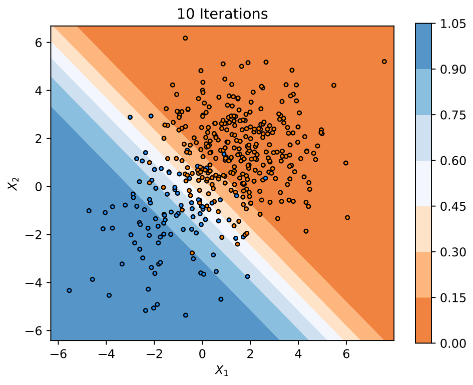
For the impact on the boundary, we can see on the figures below how a higher number of iterations makes the zone where it's more uncertain, so around the boundary, tighter. It thus makes the two very confident zones larger and more and more distinct.

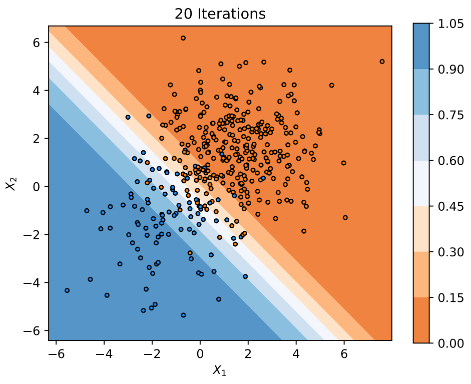
The figures correspond respectively to 1, 2, 3, 10, 20 and 30 iterations.

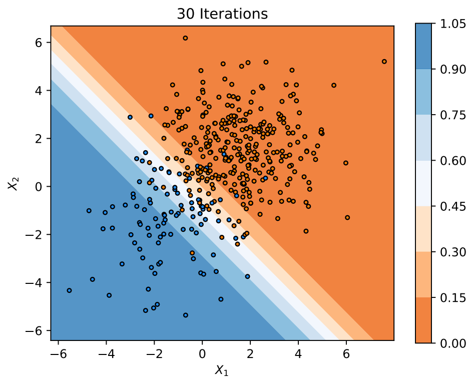












8. First, by looking at the accuracy scores, we can see that the three methods hit similar scores at around a bit more than 90%. The best mean accuracy we had was with logistic regression that reached 92,48%, then with kNN, and finally with decision trees.
   Also, by looking at the nature of the problems, a logical boundary is a linear one perpendicular to the line that joins the center of each distribution. This is achieved by the logistic regression methods.