31/10/2018

Project 1 - Classification algorithms

Introduction to Machine Learning

Delaunoy Arnaud s153059
Minne Adrien s145340
Master Civil Engineer

1 Decision tree

1.

(a)

Decision boundaries are shown in figures 2-6. We can see that the higher the depth, the higher the number of sections in the decision plot. This is due to the fact that if the depth is higher, the decision tree has more leafs and therefore more sections.

(b)

Figure 7 shows that the optimal depth is 4. For depth lower than 4, the accuracies on the training set are decreasing; the model is underfitting. For depth higher than 4, the accurcies on the training set are also decreasing; the model is overfitting. One can see on table 1 that for an unconstrained depth, the model is even more overfitting.

(c)

When the depth is unconstrained, the mean accuracy computed as in subsection 2. for the training set is 1. The model thus make the tree such that it fits the training set perfectly.

2.

Average test set accuracies are shown on table 1. As explained in subsection 1., The depth that leads to the higher accuracy is 4. If the depth is higher, the model is overfitting and if the depth is lower, the model is underfitting. It's also shown that the depth has no impact on the standard deviation of the accuracy.

depth	mean	std
1	0.67	0.01445299
2	0.802	0.02777689
4	0.83733333	0.02480143
8	0.81133333	0.02050474
Unconstrained	0.77533333	0.02817406

Table 1: Mean and standard deviation of test set accuracies for each depth

2 K-nearest neighbors

1.

(a)

The boundary plots are the Figures 8-13 in the annex.

(b)

The boundaries are represented on the Figures 8-13. Notice that we used the seed "687" to generate our dataset, and that for this seed there's more blue dots(y = 1) than red dots(y = 0) in our learning sample (the dataset is separated in LS and TS with the *sklearn.model_selection.train_test_split()* function with the same random seed as the generated dataset). When there's only 1 Neighbour used to determine the "color" of the point, only the nearest neighbour is considered. Consequently, the probability for a point to be blue or red is always 100% or 0%, there's no in between. That's why the boundary on figure 8 is only red or blue, with no shades in between. The boundary is chaotic when the two distributions intersect because each isolated point delimit it's own area of influence, in which it's the closest point.

When the number of neighbours considered grows, this chaoticity disappear. Isolated points have less of an impact over the whole model. As soon as 5 (fig. 9) neighbours, a tendency is beginning to draw clear. The more concentrated red points have their area of influence where they are more numerous, and the area of influence of the more spread out blue dots fill the remainder of the plane. It's as if the two distribution were separated by a curve, and the KNN algorithm is charged to find this curve. This tendency continues to grow for the following number of neighbours (fig. 10-11) and the curve becomes smoother as the number of neighbours grows.

But when the number of neighbours reach 625 (fig. 12), the model is clearly trying to overfit it's learning sample. It's clear by looking at the shades of color in the boundaries : the colors get paler and paler, meaning that the probabilities of being blue or red are quite close to the other, and the prediction aren't accurate. In the end, when the model checks every point in the learning sample to determine the color of the new dot, (figure 13), it predict that every new point is blue, simply because there's more blue dots in it's learning sample than red ones. It's clearly a result of overfitting.

2.

(a)

The cross validation can be done quite simply by using the *cross_val_score()* function from *sklearn.model_selection*. This function takes an argument an estimator (*KNeighborsClassifier* in our case), the dataset in X and Y, and the "KFold" argument of our cross validation (10 in our case). It returns an array containing the score for each of the 10 learning and testing samples from our dataset, and simply computing the mean of these 10 score gives the score of KNeighbours for a given number of neighbours for our dataset.

(b)

The expected result are : underfitting for a number N of neighbours too small, overfitting for N too big, and a curve with a slope not too great in between. Experimentally, the results of figure 1 shows the expected results. The curve isn't the most smooth, because a cross validation 10 fold isn't enough to totally remove the uncertainty factor in our model. The best number of neighbours seems to be around 200. This number can slightly vary depending of the result of the cross validation, once again because a cross validation 10 fold isn't enough to totally remove the uncertainty factor in our model. The score for this number of neighbour is 0.8545, which mean that 85,45% of the training samples were predicted correctly, that's already quite good, a bit better than a decision tree.

The graph could have been inferred from the boundaries : for 1 and 5 neighbours, the model underfit the dataset, it gets better as the number of neighbours increase, and finally overfitting becomes a problem from 625 neighbours onwards.

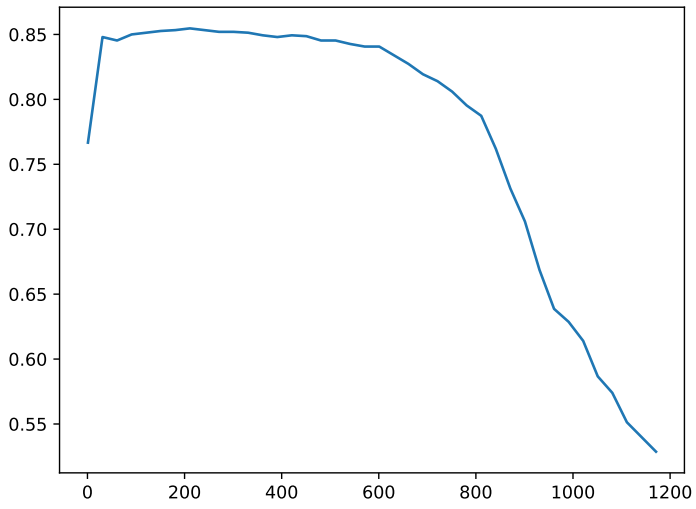

Figure 1: Scores of the KNN algorithm with respect to the number of neighbours considered

3 Linear discriminant analysis

1.

(a) two classes case

The decision boundary is said to be linear if the decision surface is a hyperplane. We must thus show that the decision boundary is of the form

$$\sum_{i=1}^{n} \alpha_i x_i = \beta$$

A sample is classified as the classe that has the higher probability $P(y = k|x)$. The decision boundary has thus the equation:

$$\begin{aligned}
&P(y = 0|x) = P(y = 1|x) \\
\Leftrightarrow &\frac{f_0(x)\pi_0}{f_0(x)\pi_0 + f_1(x)\pi_1} = \frac{f_1(x)\pi_1}{f_0(x)\pi_0 + f_1(x)\pi_1} \\
\Leftrightarrow &f_0(x)\pi_0 = f_1(x)\pi_1 \\
\Leftrightarrow &\frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)}\pi_0 = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)}\pi_1 \\
\Leftrightarrow &-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) = ln(\frac{\pi_1}{\pi_0}) \\
\Leftrightarrow &(\boldsymbol{x}.\boldsymbol{\Sigma}^{-1}.\boldsymbol{x}) - 2(\boldsymbol{x}.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_0) + (\boldsymbol{\mu}_0.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_0) - (\boldsymbol{x}.\boldsymbol{\Sigma}^{-1}.\boldsymbol{x}) + 2(\boldsymbol{x}.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_1) - (\boldsymbol{\mu}_1.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_1) = -2ln(\frac{\pi_1}{\pi_0}) \\
\Leftrightarrow &2\boldsymbol{x}.(\boldsymbol{\Sigma}^{-1}.(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)) = (\boldsymbol{\mu}_1.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_1) - (\boldsymbol{\mu}_0.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_0) - 2ln(\frac{\pi_1}{\pi_0})
\end{aligned}$$

$$\text{Let} \begin{cases} \boldsymbol{\alpha} = 2\boldsymbol{\Sigma}^{-1}.(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \\ \beta = (\boldsymbol{\mu}_1.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_1) - (\boldsymbol{\mu}_0.\boldsymbol{\Sigma}^{-1}.\boldsymbol{\mu}_0) - 2ln(\frac{\pi_1}{\pi_0}) \end{cases}$$

The equation thus become

$$\boldsymbol{x}.\boldsymbol{\alpha} = \beta$$

This equation is the equation of a hyperplane, the decision boundary is thus linear.

(b) multiple classes case

In the multiple classes case, the decision frontier will be a set of hyperplanes separating the different classes.

2.

Those decision boundaries are shown on figures 14 and 15. Both of those datasets contains 2 classes of samples. The decision boundary is linear, as demonstrated in subsection 1.

3.

Average accuracy and standard deviation are shown table 2.

dataset	mean	std
1	0.91	0.02403701
2	0.824	0.03309246

Table 2: Mean and standard deviation of test set accuracies

4.

Table 2 shows that the lda estimator has a better accuracy and a lower standard deviation for the first dataset. This is due to the fact that in the first dataset, both Gaussians are the same and thus have the same covariance matrix. The homoscedasticity hypothesis used to fit our lda estimator is thus verified for the first dataset and not for the second one.

It can be explained intuitively by noticing that when the two distribution have the same shape, it's easier to separate them by drawing a straight line. But when the distribution are different, a simple line can fail to grasp an accurate separation between the two. That's why KNN gives us better results for the dataset2, because it "draws" a curve, and not a line, to separate the two distribution, which will create a more accurate separation between the distributions.

4 Annex

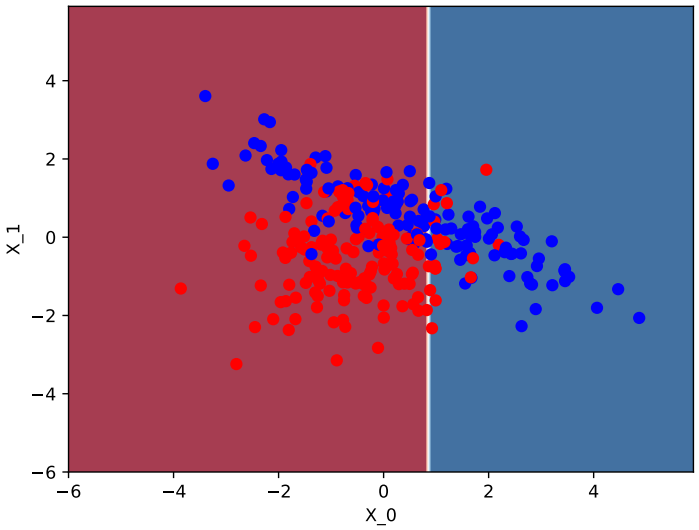

Figure 2: Decision boundary for depth = 1

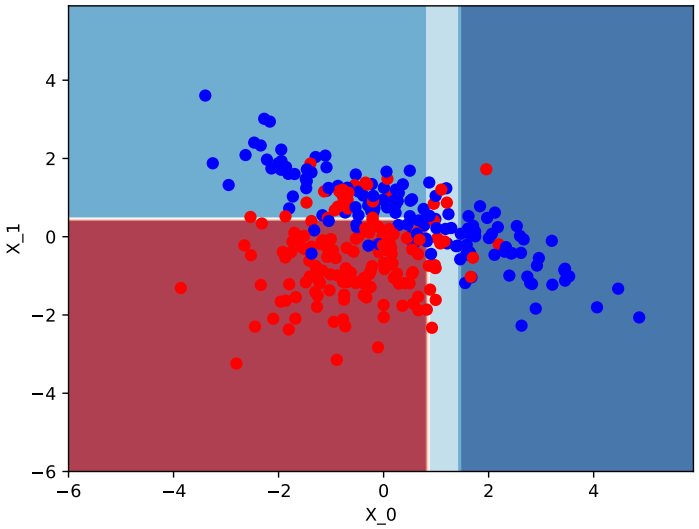

Figure 3: Decision boundary for depth = 2

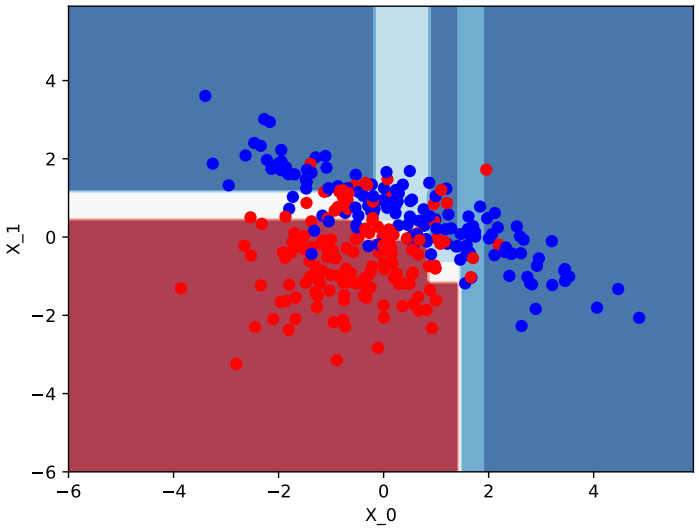

Figure 4: Decision boundary for depth = 4

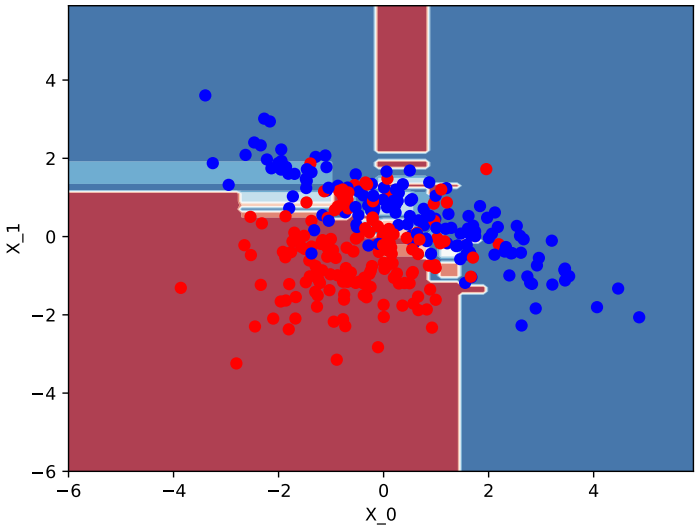

Figure 5: Decision boundary for depth = 8

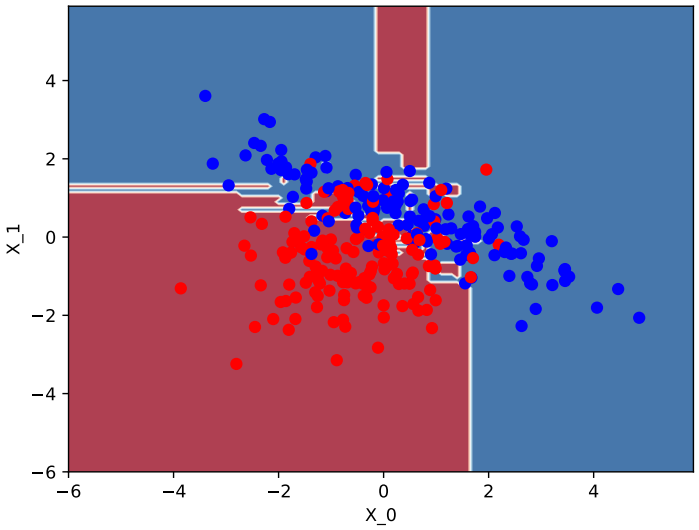

Figure 6: Decision boundary for depth = None

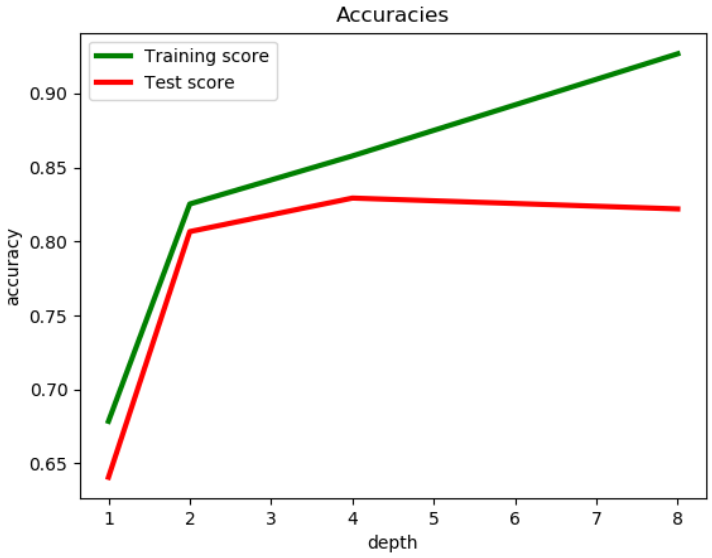

Figure 7: Train and test set accuracy

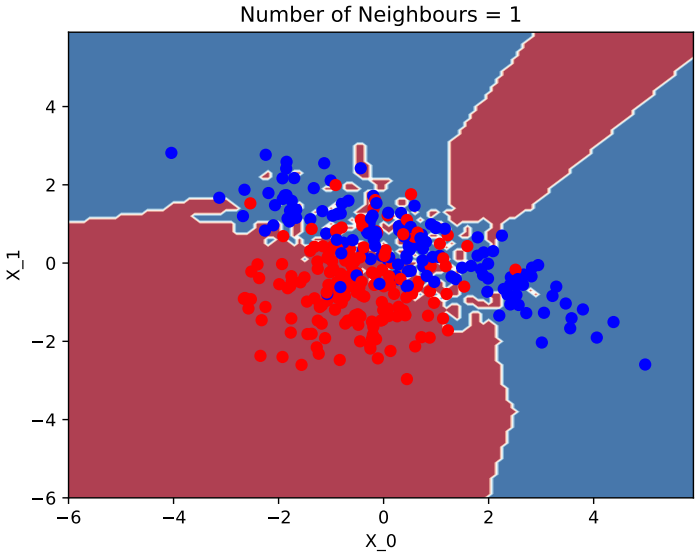

Figure 8: Decision boundary for 1 neighbour

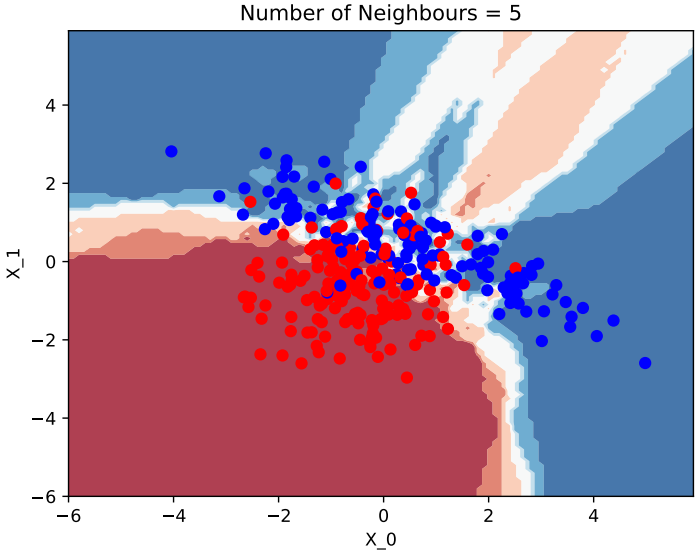

Figure 9: Decision boundary for 5 neighbour

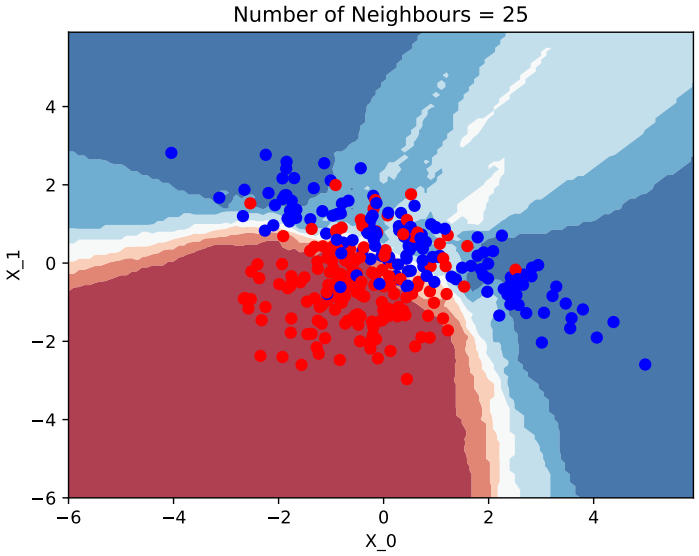

Figure 10: Decision boundary for 25 neighbour

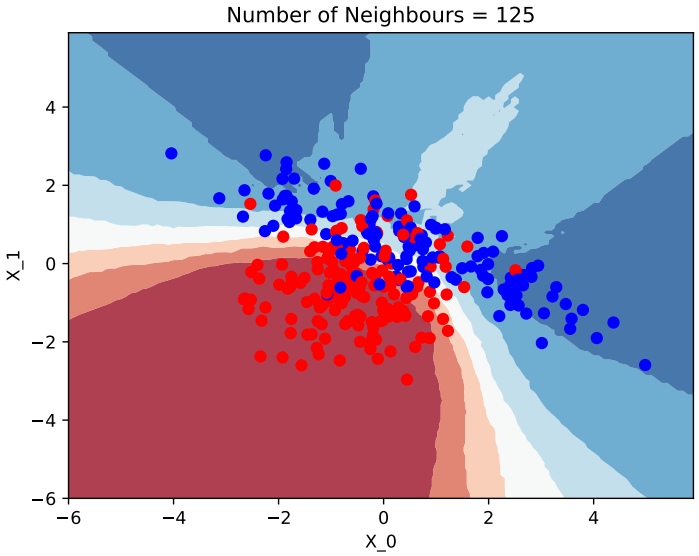

Figure 11: Decision boundary for 125 neighbour

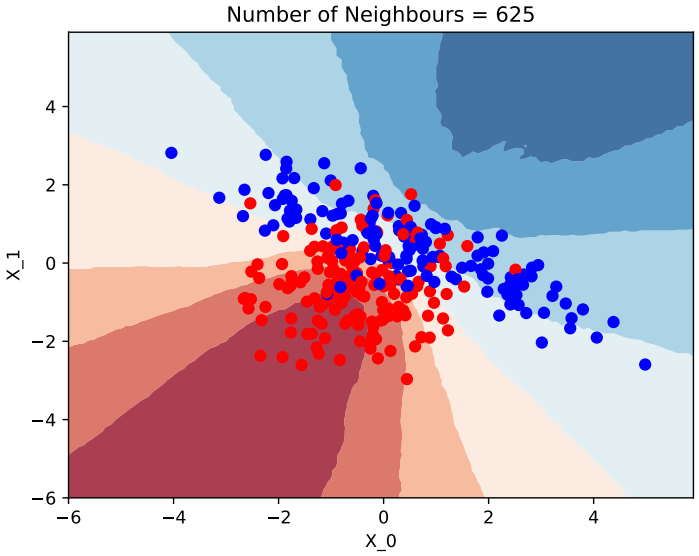

Figure 12: Decision boundary for 625 neighbour

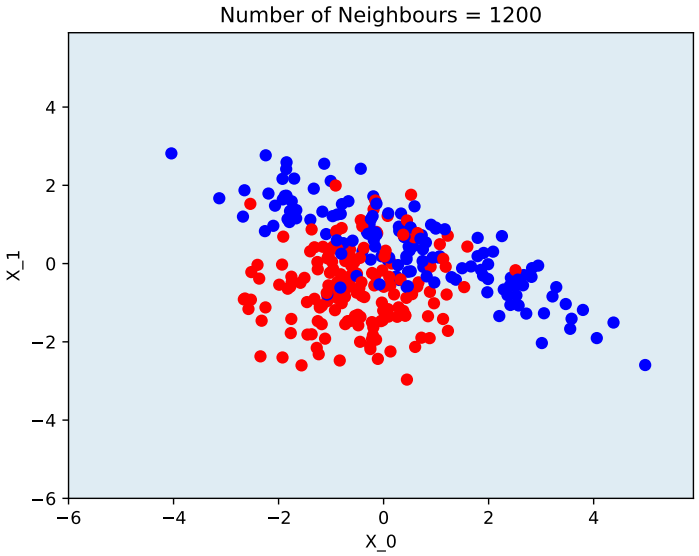

Figure 13: Decision boundary for 1200 neighbour

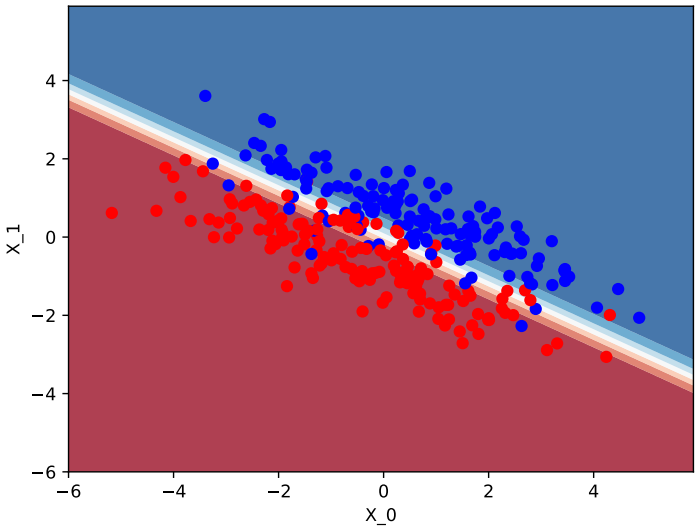

Figure 14: Decision boundary of linear discriminant analysis estimator for the first dataset

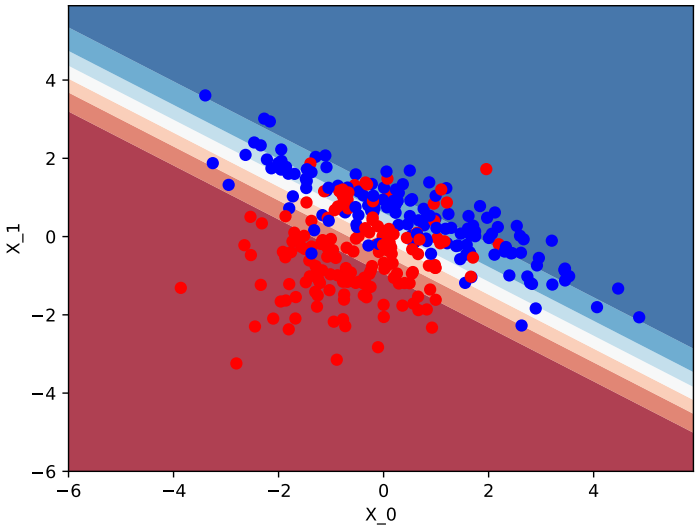

Figure 15: Decision boundary of linear discriminant analysis estimator for the second dataset