LIÈGE université
Sciences Appliquées

Université de Liège
Faculté des Sciences Appliquées

Project 1: Classification algorithms

ELEN0062-1 : Introduction to machine learning

Andrei Galan (s191903)
Tom Moës(s191408)
Jérôme Pierre (s190979)

October 23, 2022

Decision tree

Decision trees are among the machine learning (ML) algorithms used for classification. In this project, we will mainly work with a dataset of 1500 instances sampled from two gaussians. The goal is to discuss the performance of different kinds of classification algorithms.

1. Decision boundary

(a) Decision boundary for every hyperparameter

This classifier uses provided data to establish conditions on the different coordinates at each split of the decision tree in order to find the different classes. It explains why the decision boundaries are composed of different straight lines. For a maximum depth of 1, the classifier will simply fit a horizontal or vertical line in order to classify as best as possible. Then, the number of possible regions increase with the maximum depth which also results in an increase in complexity. This is quite visible for the plot of depth = 8 and depth = None (Figure 4 and 5). In those two cases, it can be seen that the overlapping region is most chaotic part of the plot[1]. It can also be seen that extreme values will affect the color of a large part of the plot as in the upper orange area.
Those results were predictable since the deeper the tree, the more precise the classifier gets. However, being too good at predicting the class of the different instances of the training set is not always a great thing. The over-adaptation can indeed imply a loss in generality resulting from fitting noise from the learning set and then induce errors during the classification of unseen data[2]. It doesn't mean that having the simplest model possible is a great idea though. Indeed, it would forbid the classifier from finding anything relevant [3] and then also increases the generalization error (error made on unseen data). We're then facing a tradeoff ! Those concepts are actually quite general and can be directly applied to KNN for example.

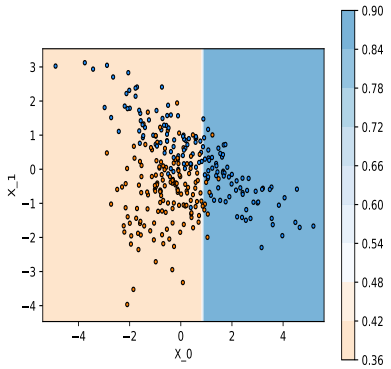

Figure 1: Classification for a maximum depth of 1

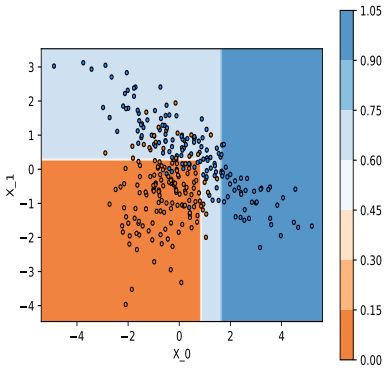

Figure 2: Classification for a maximum depth of 2

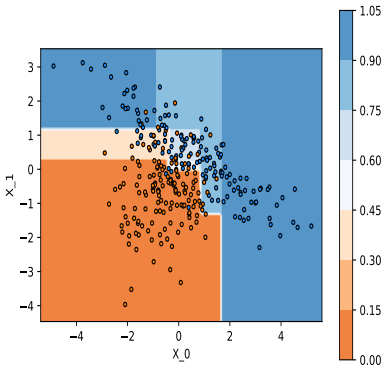

Figure 3: Classification for a maximum depth of 4

[1] In the case of depth = 8, regions with probability around 50% lie in this overlapping area.
[2] This phenomenon is called overfitting
[3] this phenomenon is called underfitting

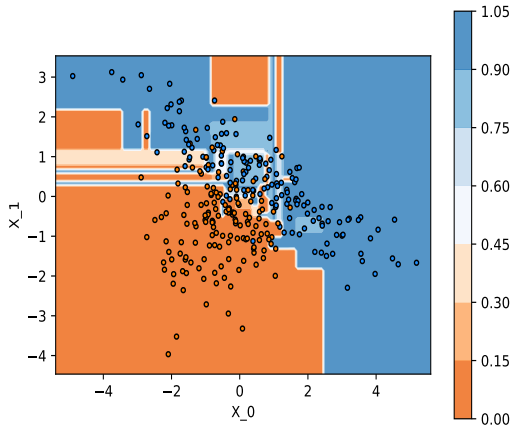

Figure 4: Classification for a maximum depth of 8

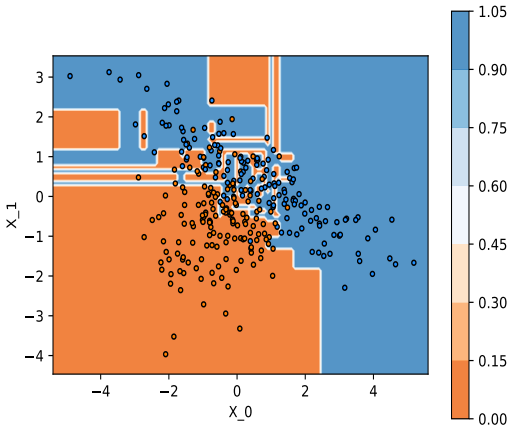

Figure 5: Classification for a depth not limited

(b) Underfitting/Overfitting

By observing the complexity of the decision boundaries, it can be understood that Figure 4 and Figure 5 are examples of overfitting because numerous sub regions independent from the general shape of the classes appear. On the other hand, Figure 1 represents underfiting since no relevant classification is provided by the output straight line. Even though a depth of 2 is OK, the best classifier from the plots seems to correspond to a depth of 4. Indeed, the resulting decision boundary is neither too simple nor too complex and incorporates most of the waited attributes. It is possible to observe an uncertainty area where the two gaussians overlap and where the orange points are slightly above the blue point cloud.

(c) Model confidence when max_depth is the largest

Another thing worth pointing out is that the deeper the decision tree, the smaller transition regions between extreme certitude areas are. This means that the model is way more confident with largest values of depth. It seems pretty logic because working with large value of the depth parameter means not having a lot (or any) counter-examples in each subset of the 2D plane for the learning phase. This directly leads to a very high certitude for the model.

2. Accuracy

Depth	1	2	4	8	None
Accuracy	0.6408	0.7806	0.8180	0.8013	0.7786
Standard deviation	0.0141	0.0212	0.0199	0.0098	0.0124

Let's observe that the standard deviation equals at most 2.5% of the accuracy which means that the accuracy values are pretty precise.

Those values also support what we just talked about. In the beginning, the model is too simple resulting in an increase of the accuracy as the depth is increased. However, once the depth has reached a certain value, increasing it will only decrease the efficiency of the model. Each accuracy obtained also supports the development we had just above.

KNN

KNN is an algorithm which works by looking at nearby points and computing the proportion of the different classes in a definite region of space. This principle will generally make decision boundaries follow the shape of the data a bit more than the ones of the decision tree methods.

1. Decision boundary

(a) Decision boundary for every value of n_neighbours

The new objective is to apply the KNN methods on similar datasets. This leads to the following results:

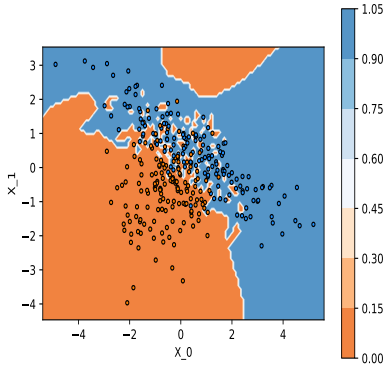

Figure 6: Selection of the closest neighbour

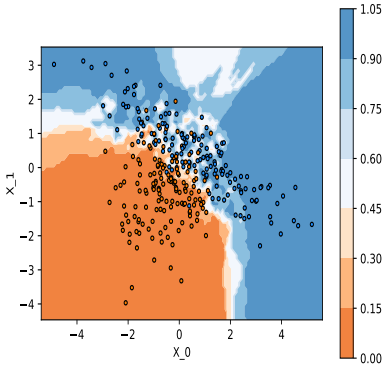

Figure 7: Selection of the 5 closest neighbours

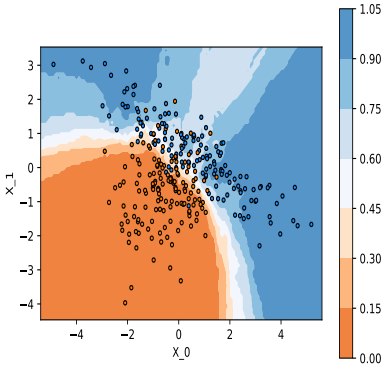

Figure 8: Selection of the 25 closest neighbours

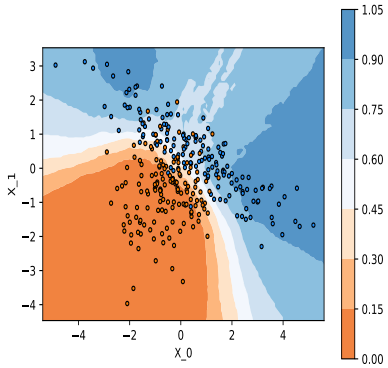

Figure 9: Selection of the 125 closest neighbours

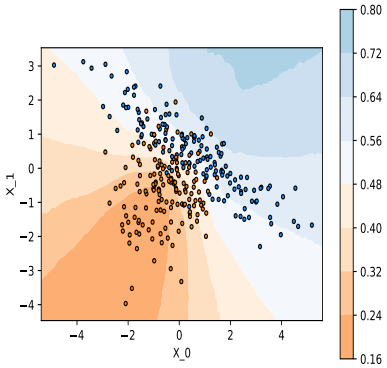

Figure 10: Selection of the 625 closest neighbours

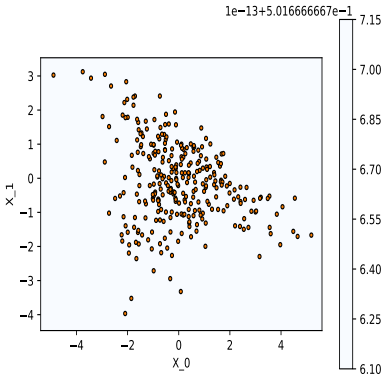

Figure 11: Selection fo the 1200 closest neighbours

(b) Decision boundary's evolution with respect to the number of neighbours

It can be observed that the general trend of the decision boundaries is to become more complex if k decreases.
In the most complex case possible (k=1), each point of the training set is considered individually by the model such that none of them is badly classified in the learning phase. It leads to some chaotic boundaries in the overlapping region of the two distributions. It induces an extremely small intermediate zone between the area associated with the different classes. The classifier is extremely confident and overfits the data. Another important thing to point out is that extreme points dictate the color on large part of the plane. For example, there are some orange points slightly above on the upper right of the blue point cloud. Those few points create the huge orange area in the upper part of the plot.

As k increases, the model starts grouping more point together and then stops fitting the noise of the learning sample. In k=5, k=25 and k=125, the classifier understands that it cannot create simple decision boundary in the overlapping region. It then associates a probability close to one half to there. It has difficulty to understand that the few orange points above the blue point data should be classified as being orange and then treat them as if there were in the overlapping region[4]. We can see that the shapes of decision boundaries in those 3 graphs are roughly the same. The main difference is that the shape are a bit smoother and better adapted when k is increased. A larger uncertainty zone can also be observed. However, the difference between the generalization error is expected to be really small.

In the case of k=625, we can clearly see that the model groups too many points together resulting in a way too simple model: two regions that could be separated by a line with an uncertainty zone slightly below the blue data point. We are facing underfitting.

Finally, for k=1200, since 1200 is the size of the training set, the classifier associates all the point of the learning set to a single class which is the one with the largest number of points. Underfitting is so important that we could argue that no classification task has been performed here...

2. Accuracy

Neighbors	1	5	25	125	625	1200
Accuracy	0.7672	0.8027	0.8375	0.8381	0.8067	0.4910
Standard deviation	0.0117	0.0176	0.0165	0.0159	0.0236	0.0065

The results agree with what has been said up until now: overfitting and underfitting both implies a lower accuracy. Since the two classes have the same number of points initially, the accuracy is close to 50% in the last case. We can also point out that the standard deviation is once again pretty small compared to the accuracy. It means that our values are rather precise.

[4]Note: there are so few points there that it won't greatly change the generalization error

LDA/QDA

1. Decision boundary's mathematical expression

It is first asked to prove that the decision boundary of LDA is linear and the one corresponding to QDA is quadratic.

Since it is assumed that the input vector is distributed according to a multivariate Gaussian $\mathrm{N}(\mu, \Sigma)$, the Bayes' theorem yields:

$$P(y = k|X = x) = \frac{f_k(x)}{\sum_{l=1}^{K} f_l(x)\,\pi_l} \tag{1}$$

where f_k is the probability density function of the multivariate Gaussian and π_l is the class l prior.

A decision boundary is the locus x^* where the prediction model is unable to predict the class that should be assigned. In such places, the conditional probability of belonging to two[5] different classes is equivalent. Let G be the set of possible classes,

$$x^* : \exists\, i, j \in G \text{ with } \mathrm{P(y = i|X = x^*) = P(y = j|X = x^*);\ (i \neq j)} \tag{2}$$

Hence in p dimensions, and noting $\det(\Sigma_i)$ as $|\Sigma_i|$:

$$P(y = i|X = x^*) = P(y = j|X = x^*) \tag{3}$$

$$\frac{1}{(2\pi)^{p/2}\,|\Sigma_i|^{1/2}}\, e^{\frac{-1}{2}(x-\mu_i)^T\,\Sigma_i^{-1}(x-\mu_i)}\,\pi_i = \frac{1}{(2\pi)^{p/2}\,|\Sigma_j|^{1/2}}\, e^{\frac{-1}{2}(x-\mu_j)^T\,\Sigma_j^{-1}(x-\mu_j)}\,\pi_j \tag{4}$$

If the case of identical Σ (hypothesis for LDA) is considered:

$$\begin{aligned} &ln(\pi_i) - \frac{1}{2}(x^T\,\Sigma^{-1}\,x - x^T\,\Sigma^{-1}\,\mu_i - \mu_i^T\,\Sigma^{-1}\,x + \mu_i^T\,\Sigma^{-1}\,\mu_i) \\ &= ln(\pi_j) - \frac{1}{2}(x^T\,\Sigma^{-1}\,x - x^T\,\Sigma^{-1}\,\mu_j - \mu_j^T\,\Sigma^{-1}\,x + \mu_j^T\,\Sigma^{-1}\,\mu_j) \end{aligned} \tag{5}$$

Let's simplify this equation. Since, each term is a scalar, it is allowed to freely take the transpose of one of them. What's more, it is known that the covariance matrix is symmetric, implying that its inverse is also symmetric.

$$-x^T\,\Sigma^{-1}\,\mu_i - \mu_i^T\,\Sigma^{-1}\,x = -x^T\,\Sigma^{-1}\,\mu_i - (\mu_i^T\,\Sigma^{-1}\,x)^T = -2\,x^T\,\Sigma^{-1}\,\mu_i \tag{6}$$

In a similar manner:

$$\begin{aligned} (\mu_i + \mu_j)^T\,\Sigma^{-1}(\mu_i - \mu_j) &= \mu_i^T\Sigma^{-1}\mu_i - \mu_i^T\Sigma^{-1}\mu_j + (\mu_j^T\Sigma^{-1}\mu_i)^T - \mu_j^T\Sigma^{-1}\mu_j \\ &= \mu_i^T\Sigma^{-1}\mu_i - \mu_j^T\Sigma^{-1}\mu_j \end{aligned} \tag{7}$$

[5] It can obviously be more than two different classes but since this work only focuses on binary classification, the definition is adapted.

Introducing the linear discriminant function δ, the following linear function characterizing the decision boundary is thus obtained from 5:

$$\delta_i - \delta_j = x^T\Sigma^{-1}(\mu_i - \mu_j) - \frac{1}{2}(\mu_i + \mu_j)^T\Sigma^{-1}(\mu_i - \mu_j) + ln(\frac{\pi_i}{\pi_j}) = 0 \tag{8}$$

However, if a specific covariance matrix is now given to each class (for QDA), the last result doesn't hold anymore. Indeed, a quadratic equation is obtained in such a case. It can indeed be seen in (5) that the quadratic terms won't cancel each other out anymore.

2. Decision boundary on second data set

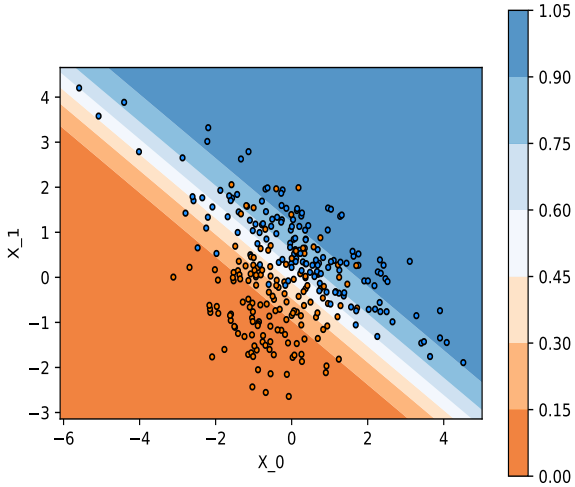

Figure 12: Example of the decision boundary from LDA

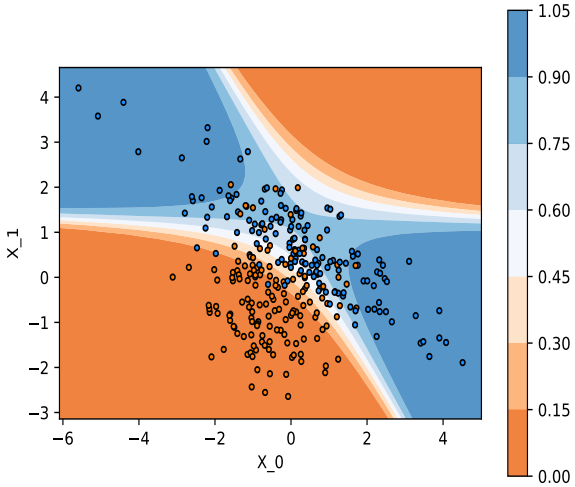

Figure 13: Example of the decision boundary from QDA

As demonstrated above, the decision boundary of LDA is linear and the one corresponding to QDA is quadratic. It can also be seen that the overlapping region has probability around 50%. What's more, it can be observed that the decision boundary of QDA is a bit more adapted to the data than the one of LDA.

3. Accuracy

Make_dataset	Classifier	Accuracies					Mean	Standard deviation
		n°1	n°2	n°3	n°4	n°5		
Make_dataset 1	LDA	0.933	0.890	0.906	0.913	0.926	0.914	0.0153
	QDA	0.933	0.896	0.903	0.913	0.926	0.914	0.0138
Make_dataset 2	LDA	0.836	0.799	0.809	0.846	0.823	0.823	0.0171
	QDA	0.870	0.823	0.826	0.853	0.836	0.841	0.0175

Just as in the previous classifiers, the standard deviation is significantly smaller than the accuracy which means that the obtained values are pretty precise.

Generally speaking, a model whose hypotheses are accurate will work better than a model whose hypotheses are not well adapted. In the case of make_dataset1, the two gaussians have the same covariance matrix which means that the assumptions of LDA are exactly true. In the case of make_dataset2, it is QDA which makes the correct assumptions. However, this simple situation doesn't show huge differences between LDA and QDA (they got almost exactly the same accuracies). Let's still observe that LDA performs significantly better in make_dataset1 than in make_dataset2 and that QDA is slightly better than LDA in make_dataset2.

Comparaison

1. Tuning hyperparameters

In both the KNN and decision tree classifier, we have a finite number of possible parameters. Let S be the number of point in our learning set. In KNN, k belongs to $[1, S]$ and in decision tree, the depth of the tree is bounded[6] by $S - 1$. A natural way of finding the best hyper-parameter is to go through all the possible cases and select the best one. To do so, we need to separate our data into the learning set and the test set to compute the accuracy on the latest. This technique is called simple cross-validation[7].

In our case, it seems convenient to apply cross-validation on different datasets in order to have the average accuracy and the associated standard deviation. As required, the following results have been obtained by using only 5 generations of data. In order to have more reliable results (especially on the standard deviation), more generations should be performed.

2. Finding the best model

(a) Make_dataset1

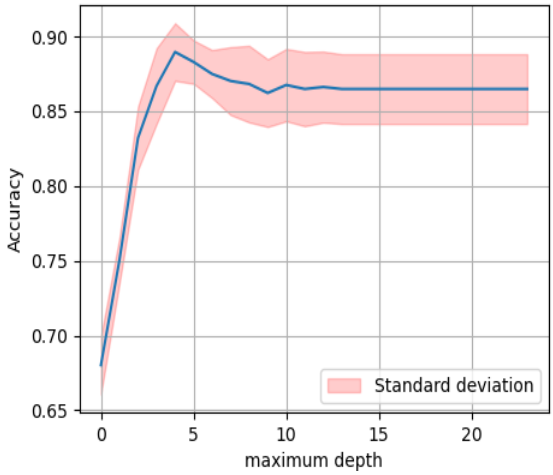

Figure 14: Plot of the average accuracy for decision tree

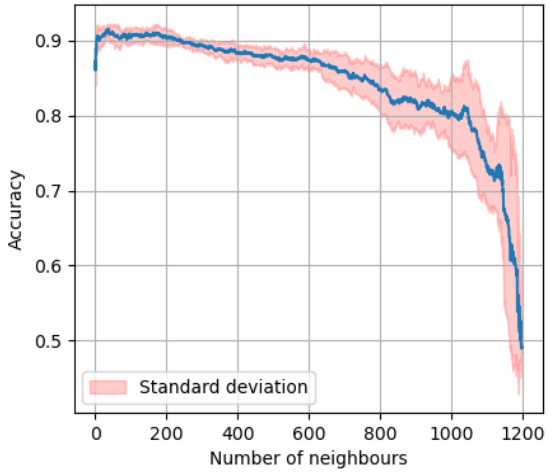

Figure 15: Plot of the average accuracy for knn

Using cross-validation, we found that the best parameter for KNN is k = 37 and for decision tree is depth = 4.

Classifier	Decision tree	KNN	LDA	QDA
Accuracy	0.889	0.915	0.914	0.914
Standard deviation	0.0193	0.008	0.0153	0.0138

[6] In practice, some classifiers forbid to go that far in order to avoid overfiting. This is why the plot is limited to a depth of 24.

[7] It would also be possible to use k-fold cross validation which consists in dividing our data into k sets and repeat k times simple cross-validation by considering one of those sets as the test set.

(b) Make_dataset2

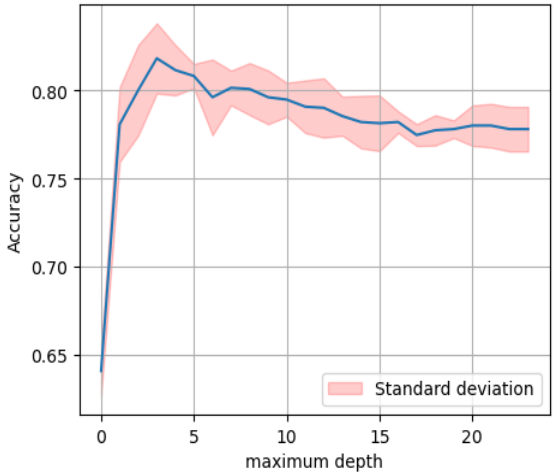

Figure 16: Plot of the average accuracy for decision tree

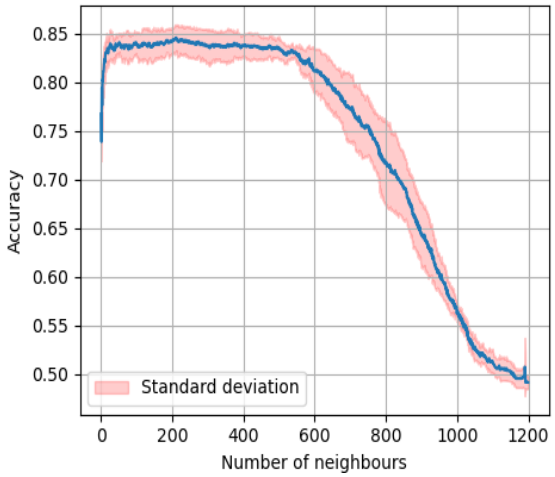

Figure 17: Plot of the average accuracy for KNN

Using cross-validation, we found that the best parameter for KNN is k = 211 and for decision tree is depth = 3.

Classifier	Decision tree	KNN	LDA	QDA
Accuracy	0.818	0.845	0.823	0.841
Standard deviation	0.0199	0.013	0.0171	0.0157

3. Comparison

In both cases, decision trees are a bit worse. It can be explained by the fact that a precise linear decision boundary with each part parallel to the axes is not a great idea here.

Otherwise, the three other methods have comparable results in make_dataset1. It is true that LDA performs a bit worse with the second type of data (make_dataset 2).

It is quite interesting that there is no big difference between KNN (a non-parametric model) and QDA (a parametric model). It should be pointed out that KNN required way more time than QDA in order to tune the parameter. This means that here, QDA should be privileged. Nevertheless, in situations where the decision boundary cannot be approximated by a quadratic functions, KNN might perform way better. This is the big strength of non-parametric model: they require no assumption on the data.