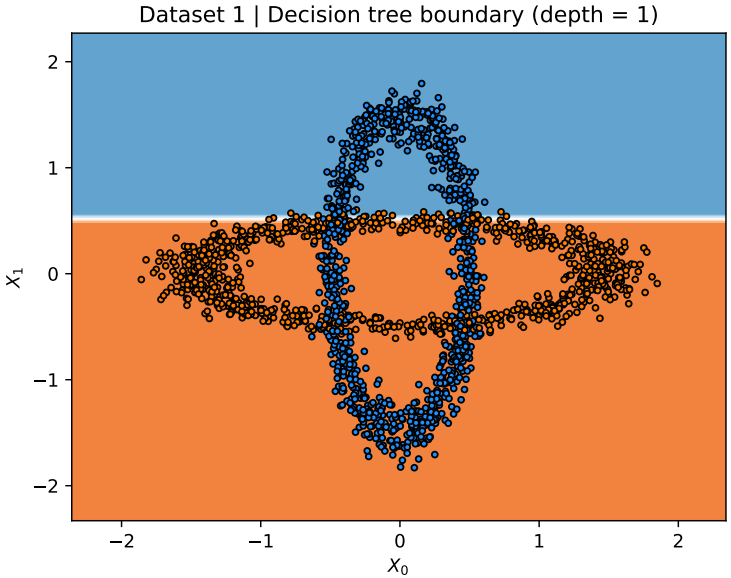
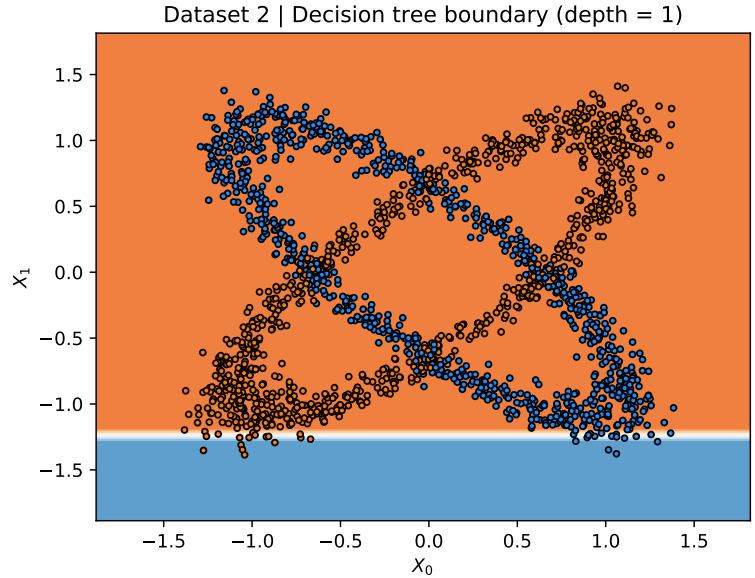
iML - Feedback of the first assignment

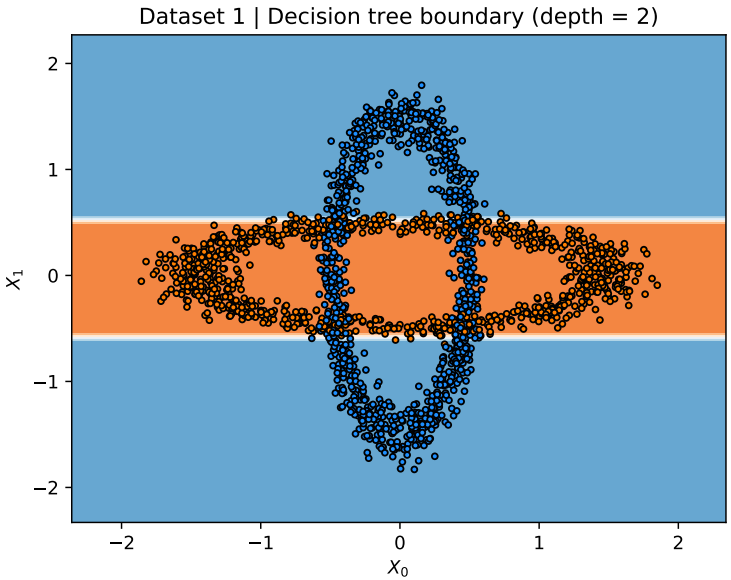
12 novembre 2019

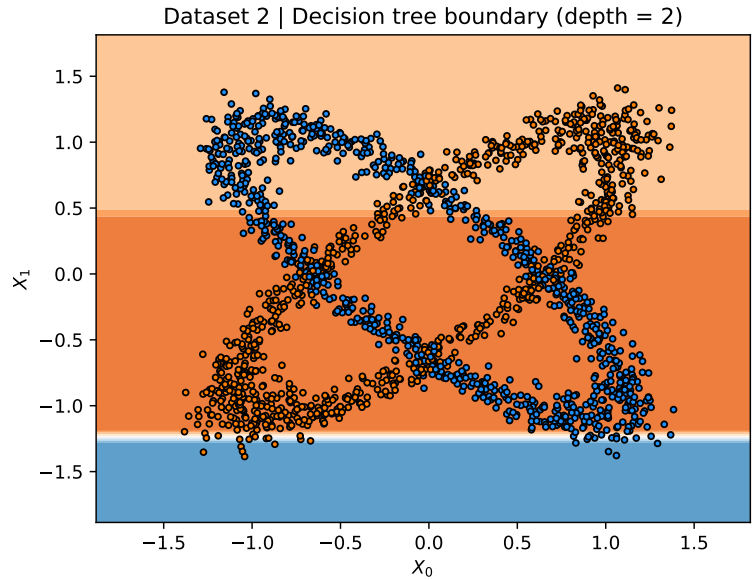
Decision tree

Decision tree – Decision boundary

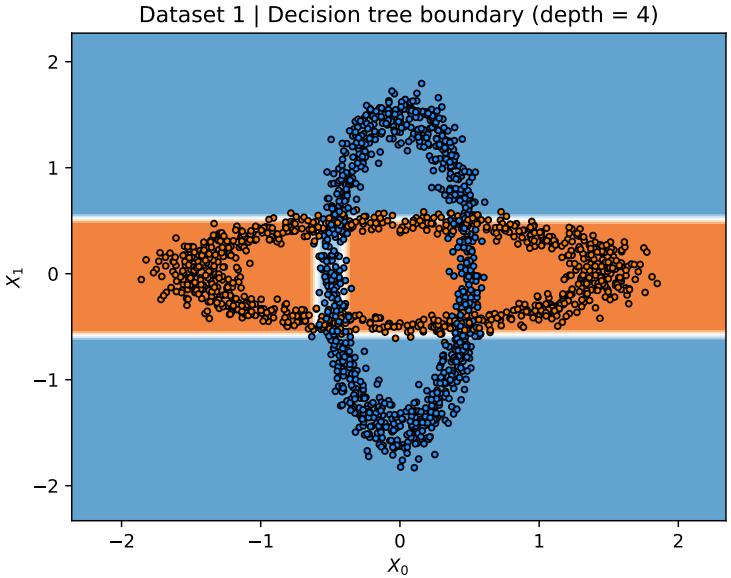


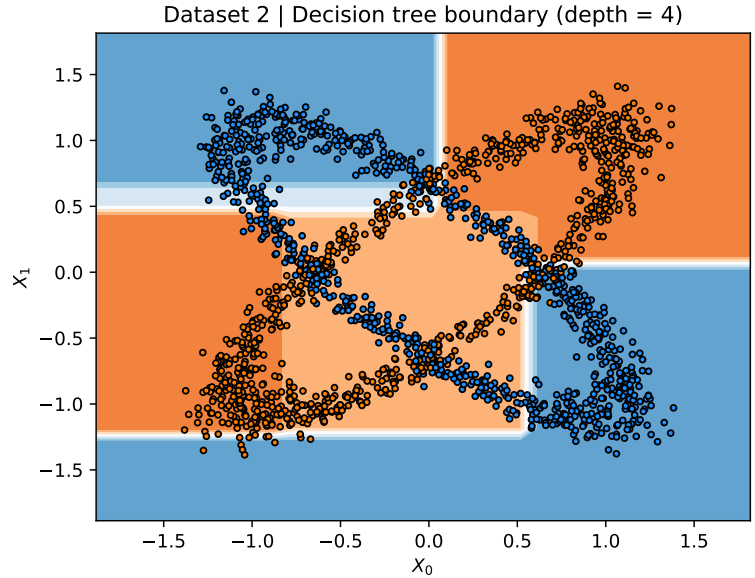


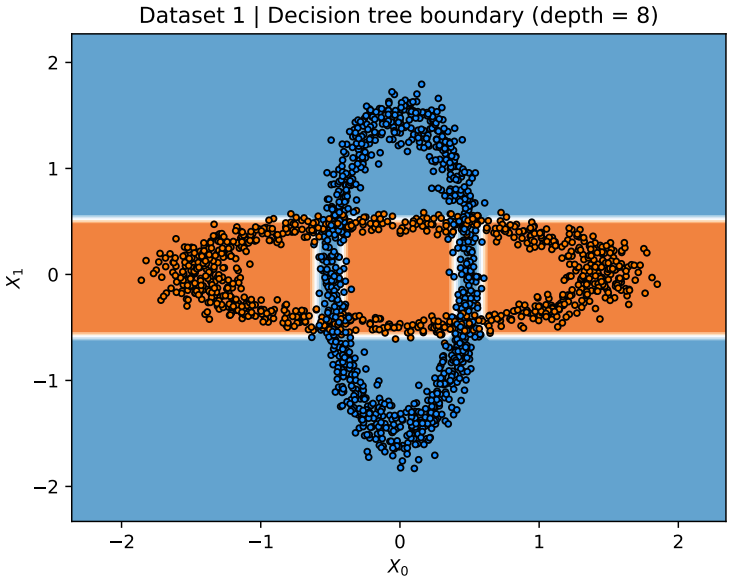


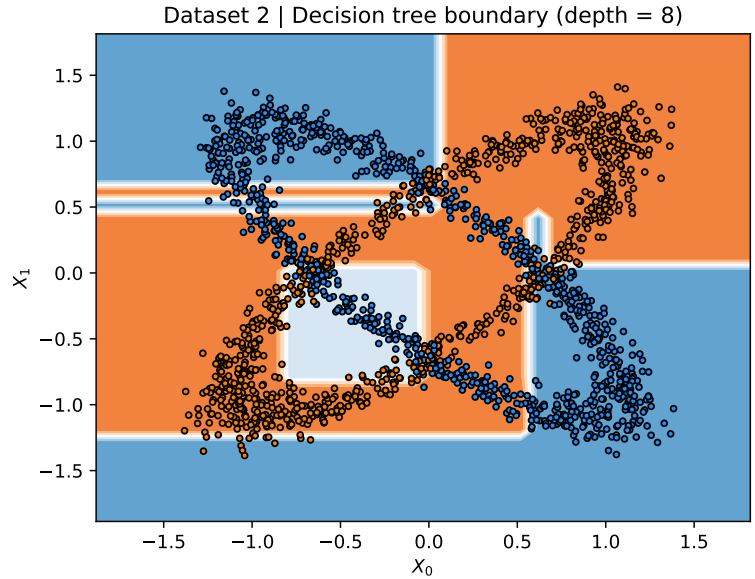


Decision tree – Decision boundary

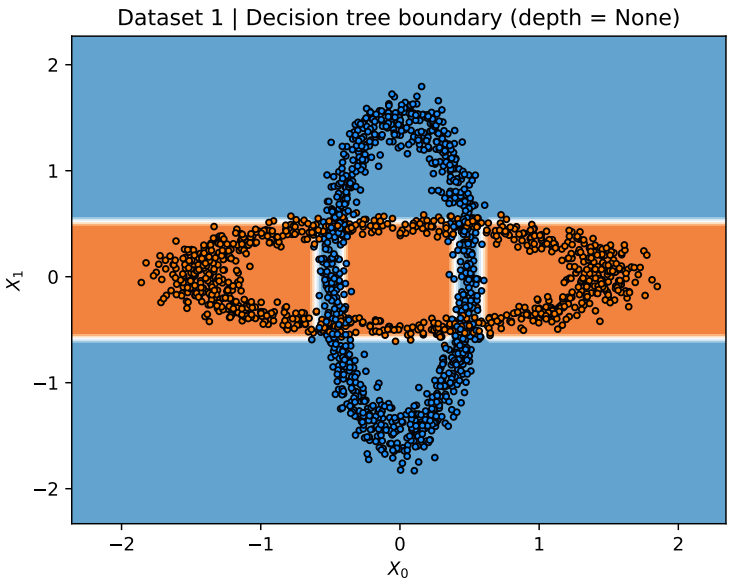


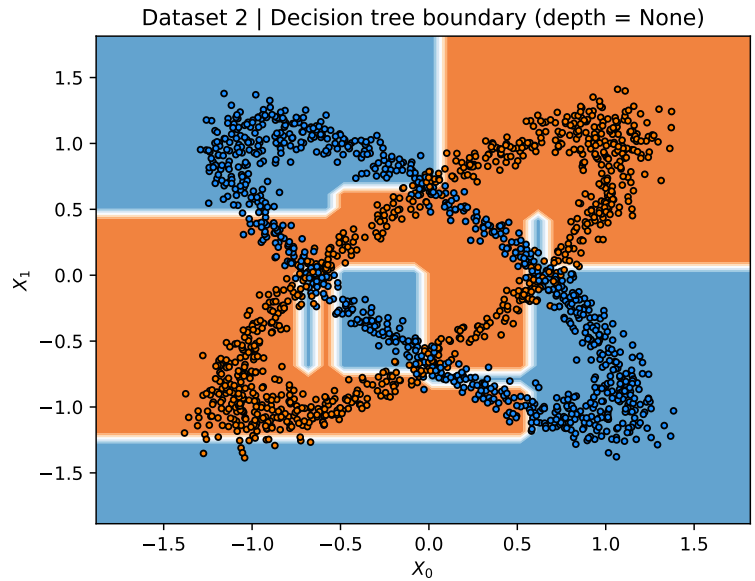






Decision tree – Decision boundary





- Boundaries are axis-aligned and define rectangular regions (because a DT splits on one variable at a time).
- A deeper model entails (many) more regions (because regions are split recursively and the number of them can grow exponentially).
- Dataset 1 is well handled because it is axis-aligned as well.
- Dataset 2 needs a more complex (*i.e.* deeper) model.

Decision tree – Underfitting/overfitting

General comments

- The goal was to get some intuition about under/overfitting. Do not look at the error curve.
- Under/overfitting is a model-independent concept
  - You should not compare to what the best decision tree model would do but to what the *overall* best model could do.
- Since the dataset was small, it was easier to observe underfitting than overfitting.
  - Clear signs of overfitting was dataset-dependent

Clear signs of underfitting (U) and overfitting (O)

Depth	1	2	4	8	-
Dataset 1	U	U	U		
Dataset 2	U	U	U	U (+ O)	(U +) O

Decision tree – test accuracy

Depth	Dataset 1	Dataset 2
1	68.15 ± 0.97	49.88 ± 1.32
2	86.22 ± 0.56	59.57 ± 10.09
4	86.75 ± 0.97	77.87 ± 1.29
8	91.41 ± 1.05	84.92 ± 2.34
Unconstrained	91.58 ± 0.85	87.96 ± 1.88

TABLE – Test accuracy (in percent) with respect to the maximum decision tree depth for dataset 1 and 2.

Decision tree – test accuracy

Depth	Dataset 1	Dataset 2
1	68.15 ± 0.97	49.88 ± 1.32
2	86.22 ± 0.56	59.57 ± 10.09
4	86.75 ± 0.97	77.87 ± 1.29
8	91.41 ± 1.05	84.92 ± 2.34
Unconstrained	91.58 ± 0.85	87.96 ± 1.88

TABLE – Test accuracy (in percent) with respect to the maximum decision tree depth for dataset 1 and 2.

About formatting

Data set \ Depth	1	2
1	0.684	0.866
2	0.499	0.653

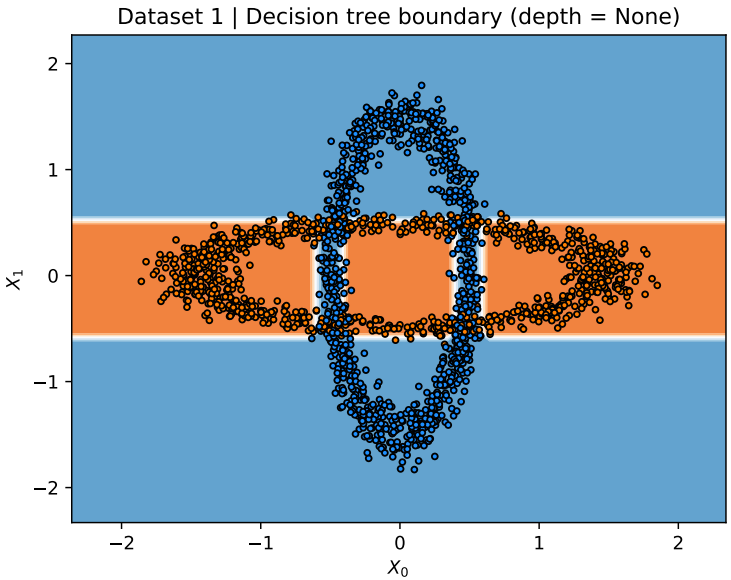
TABLE 1 – Average accuracies

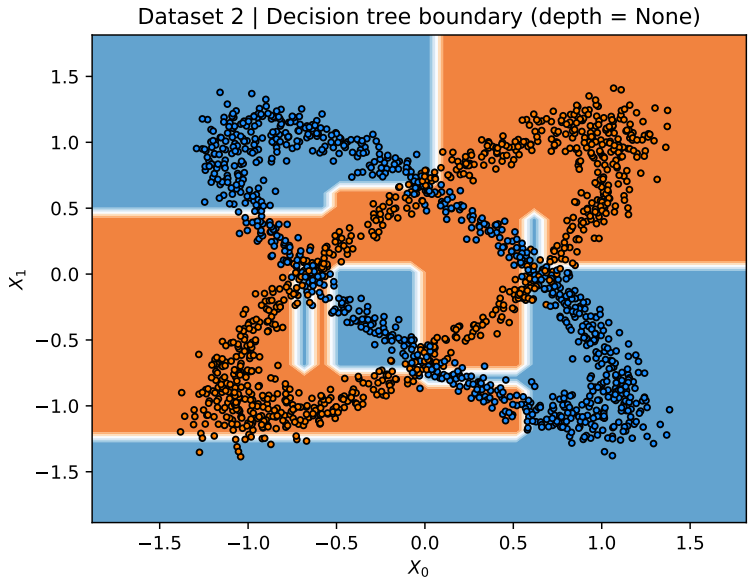
Data set \ Depth	1	2
1	0.00519	0.00639
2	0.00711	0.11084

TABLE 2 – Standard deviations of the a

Number of neighbors	Scores
1	0.9499999999999998
5	0.95
10	0.9549999999999998
75	0.952
100	0.9425000000000001
150	0.932

Decision tree – Confidence





- An unconstrained decision tree (DT) will expand until all its leaves are pure.
- The DT associates the proportion vector of the leaf in which an example fall.
- Therefore, the DT make a confident (*i.e* pure) prediction.

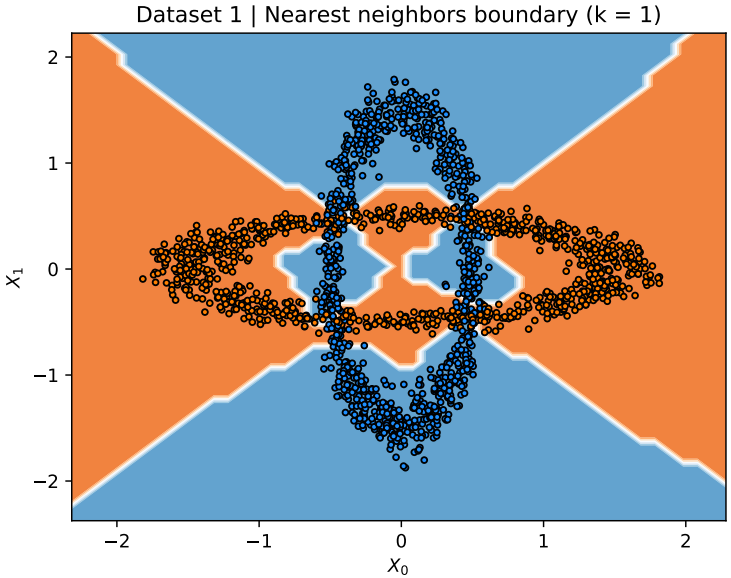
Decision tree – Differences between datasets

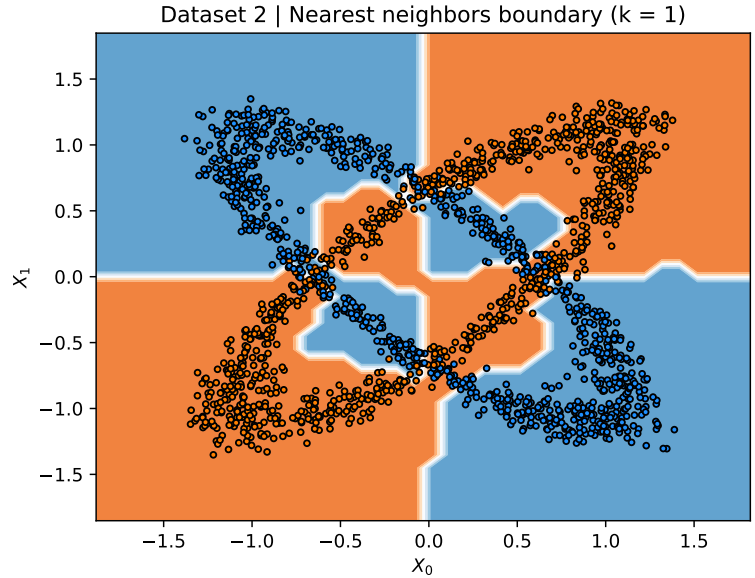
- The only difference between the datasets is the rotation.
- Decision trees split the feature space with axis-aligned cuts.
- The first dataset is also axis-aligned, therefore it is easier to handle.
- The learning set is too small to reach good accuracy on the second dataset.

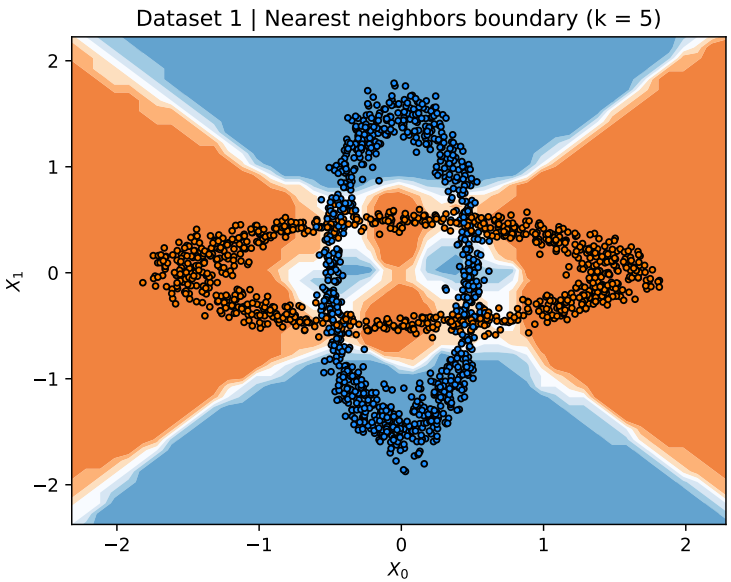
Note : stating that the first set is easier because it is axis-aligned without discussion why with respect to decision tree is not sufficient.

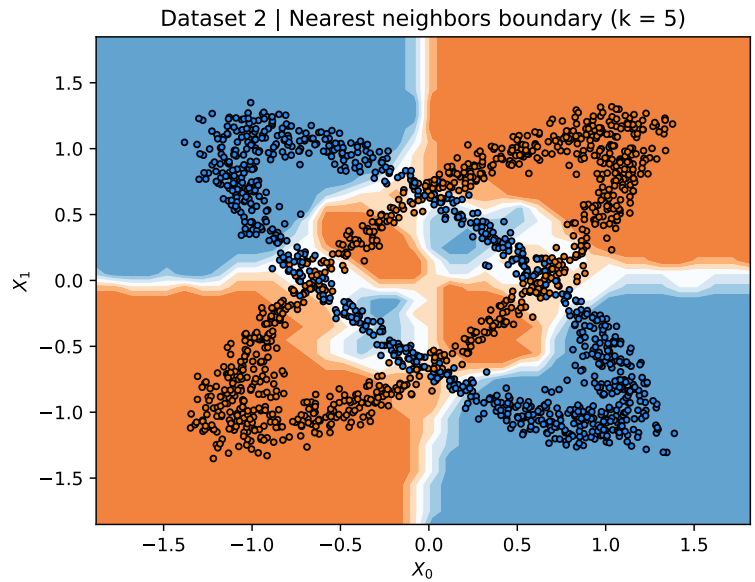
k Nearest neighbors

k-nearest neighbors – Decision boundary

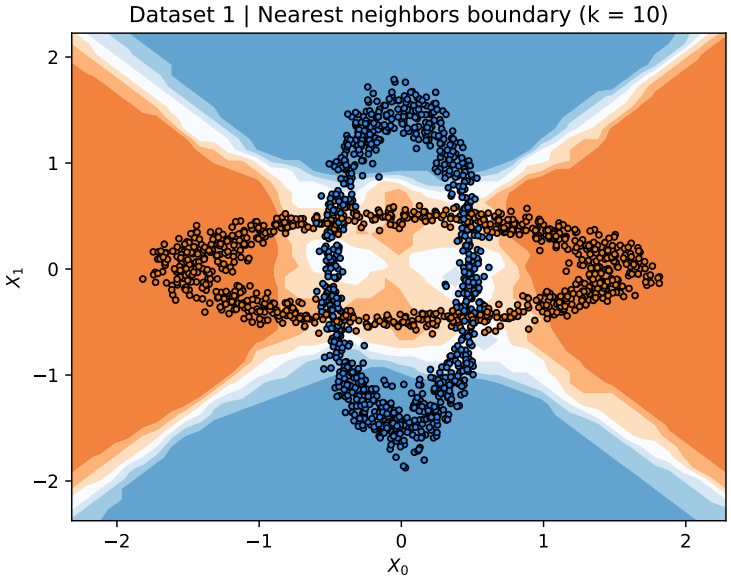


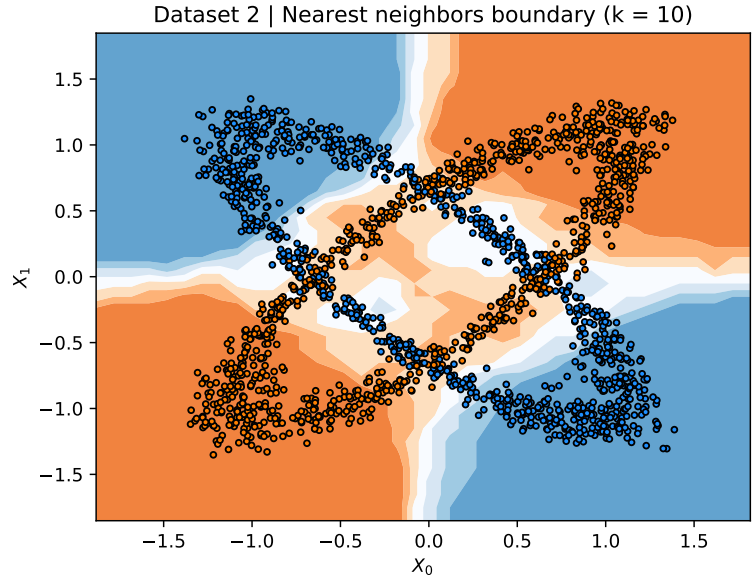


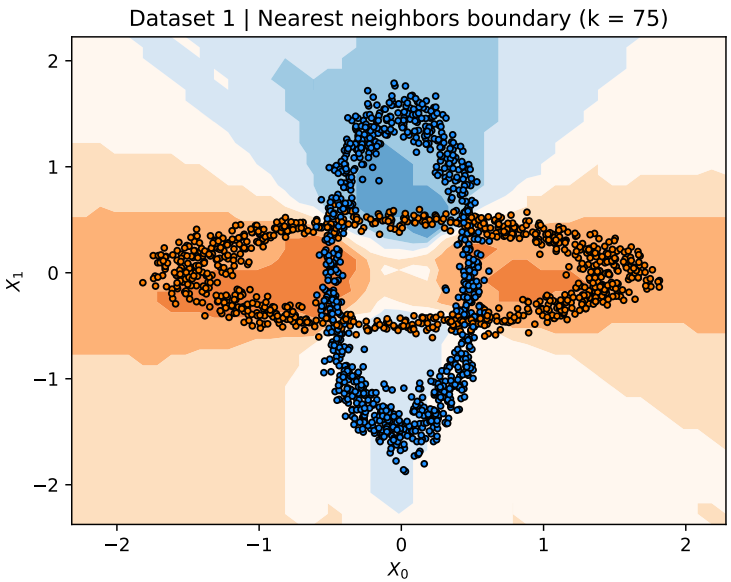


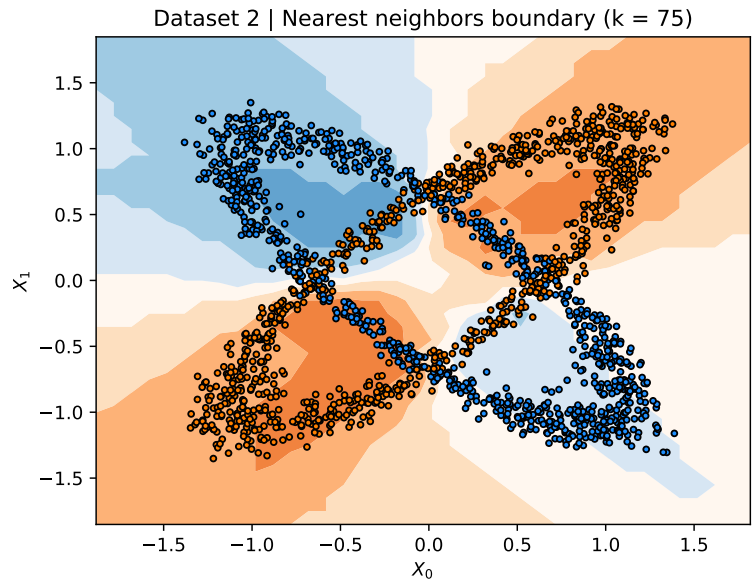


k-nearest neighbors – Decision boundary

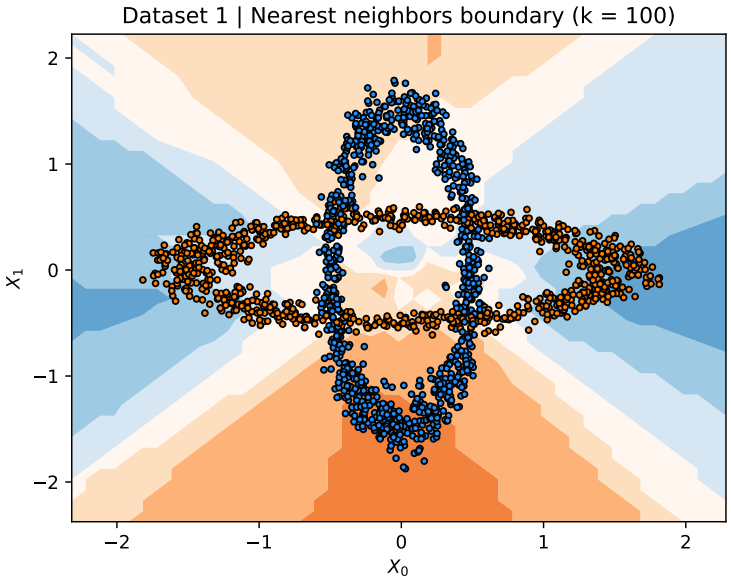


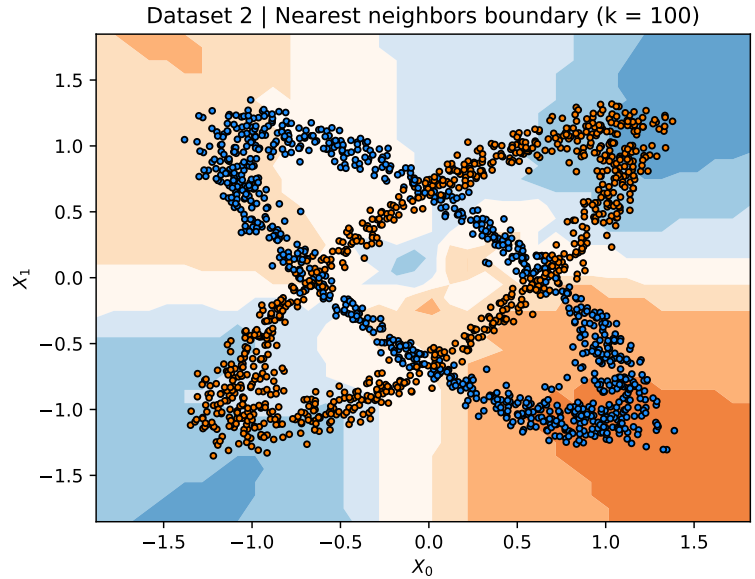


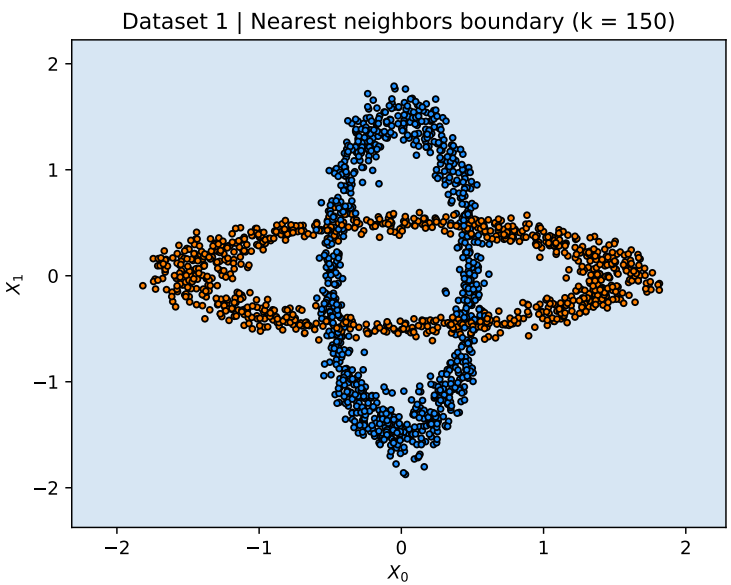


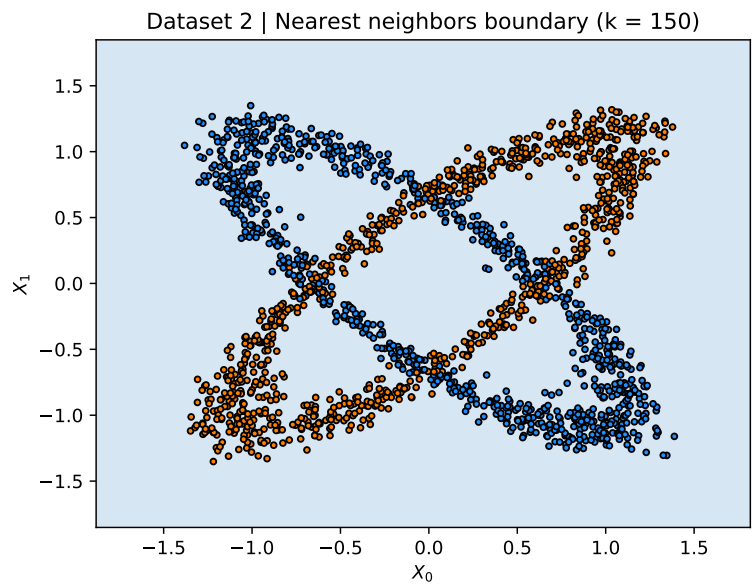


k-nearest neighbors – Decision boundary









k-nearest neighbors – Decision boundary

$k = 1$ The boundary is sharp ; the model is confident. There might be some overfitting

- This is because we assign the class of only one neighbor.

$k = 5, 10$ Uncertainty starts to appear near the crossing zone and on the bisectors. The overall boundary seems reasonable.

$k = 75$ The boundaries are not so great anymore and there is a lot of uncertainty.

$k = 100$ There is a inversion at the tips of ellipses. The geometry of the problem is such that we take into account more points of the other class. The overall classification is really bad.

$k = 150$ Wherever a point is, it uses all the points of the learning set to make its prediction. It is uniform and should tend only slightly toward one class.

k-nearest neigbhors – Cross validation

- Optimal value of k between 1 and 50. Great variability.
- Optimal accuracy $\approx 93\%$.
- For a truly unbiased estimate of the test accuracy, the final test set must be *independent* of all the learning sets used during cross-validation.
- Did you refit your model with all the data once the optimal k was found ?
- The optimal number of neighbors depends on the number of training samples !

Note : it was not enough to say you used the `cross_val_score` function of scikit-learn.

k-nearest neigbhors – Optimal k transferability

- If we disregard the variability due to the sampling, the only distinction between the datasets is the rotation.
- Since the kNN uses the Euclidean distance, which is isotropic, its performance are not impacted by a rotation.
- As such, the optimal value of k should be consistent on both datasets.

Note : Saying the optimal value of k would be the same because the datasets only differ by the rotation is not enough.

Naive Bayes classifier

Naive Bayes – Derivation

$$\arg\max_y Pr(y|x_1,\ldots,x_p) = \arg\max_y \frac{Pr(y)Pr(x_1,\ldots,x_p|y)}{Pr(x_1,\ldots,x_p)} \quad (1)$$

$$= \arg\max_y Pr(y)Pr(x_1,\ldots,x_p|y) \quad (2)$$

$$= \arg\max_y Pr(y)\prod_{i=1}^{p} Pr(x_i|x_{i-1},\ldots x_1,y) \quad (3)$$

$$= \arg\max_y Pr(y)\prod_{i=1}^{p} Pr(x_i|y) \quad (4)$$

(1) Bayes theorem. (2) The denominator is constant for a given input and can be ignored since it will not affect the arg max. (3) Chain rule. (4) Naive Bayes assumption.

Naive Bayes – Implementation : Fit method

```
...
# Get the shapes parameters
n_instances, n_features = X.shape
n_classes = len(np.unique(y))
# Instantiate the classifier parameters
priors = np.zeros(n_classes)
means = np.zeros((n_classes, n_features))
variances = np.ones((n_classes, n_features))
# Compute the the classifier parameters
for target in range(n_classes):
    indices = y == target
    priors[target] = np.sum(indices)/float(n_instances)
    means[target, :] = X[indices].mean(axis=0)
    variances[target, :] = X[indices].var(axis=0)
...
```

Naive Bayes – Implementation : log_predict method

computes

$$\log\left(Pr(y)\prod_{i=1}^{p}\frac{1}{\sqrt{2\pi\sigma_i^2}}exp\left(-\frac{x-\mu_i}{2\sigma_i^2}\right)\right) \quad \forall y, \forall x$$

```
...
if self.priors is None:
    raise ValueError("Estimator not fitted.")

# Instantiate and initialise output with priors
n_instances, n_features = X.shape,
n_classes = len(self.priors)
log_preds = np.zeros((n_instances, n_classes))
# Initialize output with priors
log_preds[:] = np.log(self.priors)
```

Naive Bayes – Implementation : log_predict method

```
# Compute the log_prediction
for target in range(n_classes):
    for feature in range(n_features):
        mean = self.means[target, feature]
        var = self.variances[target, feature]
        # exponential part
        log_norm = -((X[:, feature] - mean)**2)/(2*var)
        # constant part
        log_norm -= 0.5*np.log(2*np.pi*var)
        # total log_preds
        log_preds[:, target] += log_norm
return log_preds
```

Naive Bayes – Implementation : predict methods

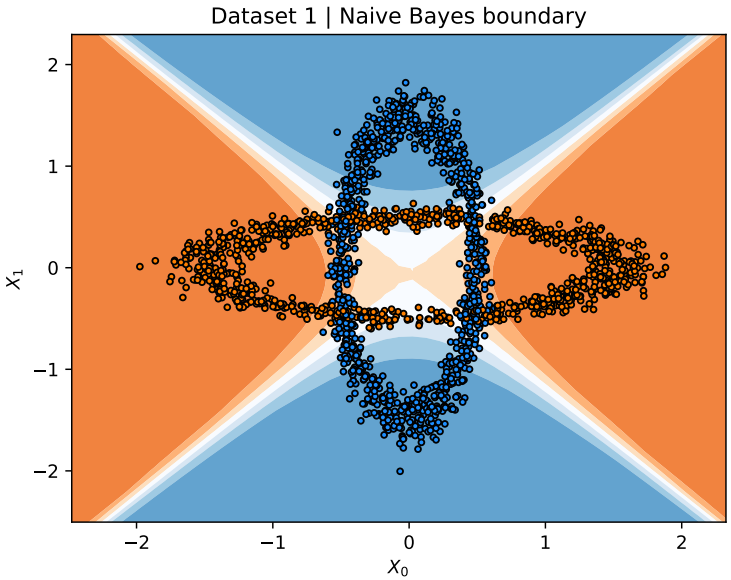
predict

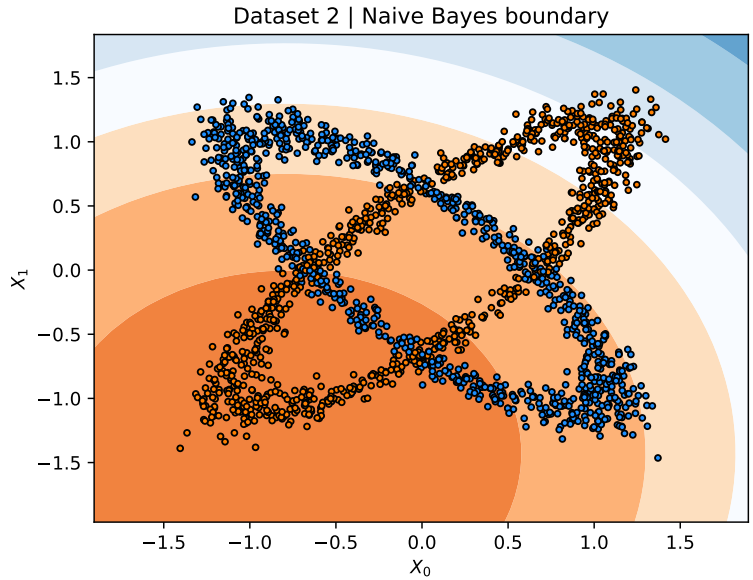
```
return self.log_predict(X).argmax(axis=1)
```

predict_proba

```
log_preds = np.exp(self.log_predict(X))
sums = log_preds.sum(axis=1)
for target in range(log_preds.shape[1]):
    log_preds[:, target] /= sums
return log_preds
```

Naive Bayes – Interpretation of accuracies





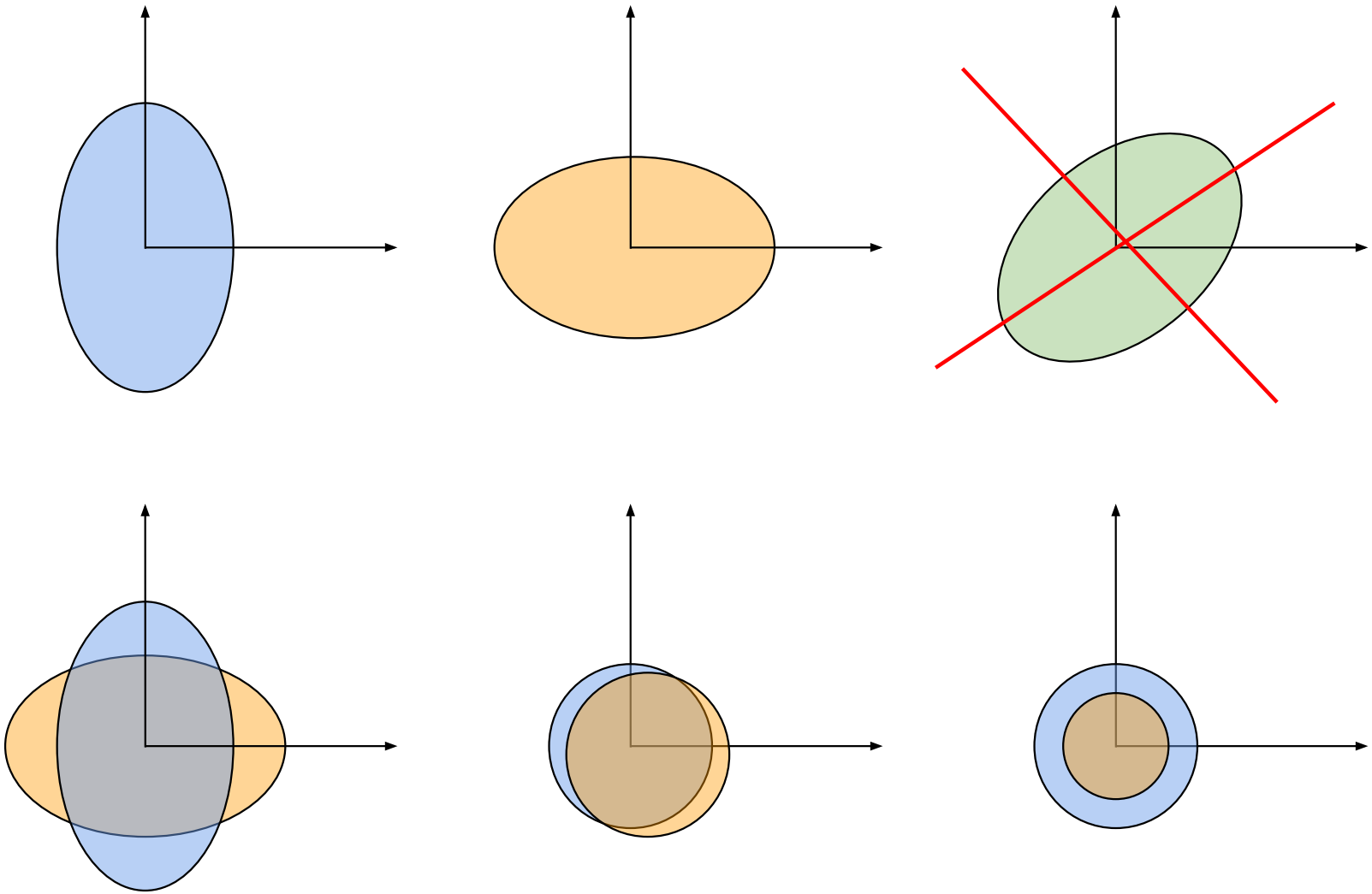
Accuracy (%) : 79.76 ± 0.98

Accuracy (%) : 46.45 ± 5.78

$$\mu_{\mathbf{b}} \approx \mu_{\mathbf{o}}$$
$$\sigma_b^{(0)} \neq \sigma_o^{(0)}$$
$$\sigma_b^{(1)} \neq \sigma_o^{(1)}$$

$$\mu_{\mathbf{b}} \approx \mu_{\mathbf{o}}$$
$$\sigma_b^{(0)} \approx \sigma_o^{(0)}$$
$$\sigma_b^{(1)} \approx \sigma_o^{(1)}$$

Naive Bayes – Gaussianity and conditional independence



Naive Bayes – Gaussianity and conditional independence

- Gaussianity implies that the model will consider elliptical distribution.
- NB assumption implies that the major/minor axes will be aligned with the X_0/X_1 axes.
- Having (approximately) the same means μ_i implies that the ellipses will be (approximately) centered on the same point.
- Having the same variances σ_i means we will have circles.
  - The boundary will try to separate almost completely overlapping circles ; the performance are expected to be random.
- Note that the crossing section is also fuzzy because the gaussians are not holed.
  - This is why the classification is not better for the first dataset.

Conclusion

Concluding remarks

- I hope it was fun and you got some intuition about the basics of machine learning.
- It is not because some assumptions of a model are not verified that the model is not useful.
  - In practice you do not know the valid assumption and you usually have to make some to get a practical solution.
  - A model is not "right" or "wrong" ; it is either useful or not.
- Try not to forget to answer some questions.
- (In a master course) you should try to supplement your observations with some links to the theory and the learning algorithms machinery.
- Do not forget some parts of the explanation (in a report, even if it feels obvious).
- Do not forget the style.