Lab 3 - Supervised Learning and Validation

# Part 1

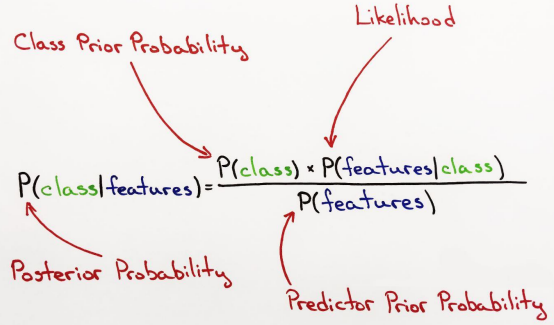
Read the tutorial Gaussian Naive Bayes Classification available on the course website. This tutorial explains NB classification and how you can implement it without using sklearn library. The objective is to dive into the algorithm and understand its parts. Execute the Python scripts GaussNB.py and look how the probabilities are implemented and used.

**To pass this part explain the concepts and Gaussian NB algorithm to the lab assistant.**

Gaussian Naïve Bayes Classification:

* Suitable for high-dimensional datasets
* Fast
* Few tunable parameters

The algorithm is based on Bayes Theorem, a way for calculating a conditional probability. The word “Naïve” comes from making naive assumptions about the generative model, which makes it possible to find an approximation of it. For example, if you are classifying a sentence, the word order does not mean anything. After this approximation we can proceed with the Bayesian classification. Gaussian comes from the fact that the Likelihood parameter is calculated based on the Gaussian Distribution for all PDFs.



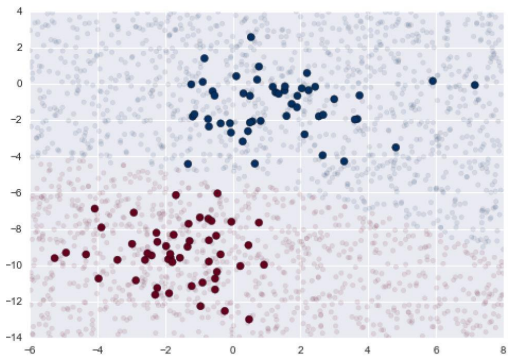
The use of Native Bayes is perfect when you need an algorithm who:

* Is fast in both training and prediction.
* Provide straightforward probabilistic prediction.
* Is easy to interpret.
* Have few tunable parameters.

…which makes it perfect for when the complexity of the model is less important and:

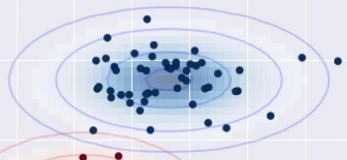
* It contains high dimensional data
* Have well separated categories

**The algorithm**



The image shows that the decision boundary is curved, this means that the boundary is quadratic for Gaussian distribution.

The algorithm takes every feature into account. For example, if the distance to the center of the clusters (the likelihood) is the same, the algorithm compares the features to see which one is more important for the query. The shape of the ellipse for 2D-examples points out the most important feature.



# Part 2

Random forest algorithms have been successfully used in solving facial recognition problems. A very well-known face recognition issue is to complete unknown parts of the face from the known parts.

Please read the code in the link carefully:

**1. Explain what the program does.**

- The program predicts how the lower part of the face looks based on the top part. It uses different algorithms to give the user a chance to rank them depending on the visual result.

**2. What is your interpretation of the final plot? Which algorithm has better performance in building the unknown parts of the face?**

- Extra Trees gives the best visual result while Linear Regression gives the worst.

**3. Download the code from the link above and modify it by adding the results of the following algorithms to the final plot:**

**(a) Regression decision tree with max depth of 10 and max number of features 50**

**(b) Regression decision tree with max depth of 20 and max number of features 50**

**(c) Regression decision tree with max depth of 20 and max number of features 25**

**(d) Random forest with max depth of 10 and max number of features 50**

**(e) Random forest with max depth of 20 and max number of features 50**

**(f) Random forest with max depth of 20 and max number of features 25**

**How do you interpret the results?**

- The regression decision tree gives a result where another bottom half from the training data set is applied. This is not a desired result!

- The random forest algorithm gives a good result, max depth 20 and max features 50 gives a slightly better result than the others. May be too low feature/depth numbers?

**4. How could performance of random forest be improved? (Hint: have a look at the example of using Haar-like feature in face detection here: https://realpython.com/ traditional-face-detection-python/)**

- You could use Haar features to detect edges. For example, a vertical mask should be used to find the bridge of the nose.

# Part 3

Start by reading the Validation Metrics tutorial (available on the course website) which presents validation metrics for regression and classification models.

Go through the three questions below and explain your solution to the lab assistant.

**1. In the script of the Regression section (the one before the section RFE), we apply cross validation with 10 folds. Note that the script does not make any change to the dataset. Modify the script to reshuffle the rows of the data set to randomize the cross-validation folds before applying the cross validation.**

**Run again the script but on the reshuffled data set and re-calculate the MSE and R2 scores. Do you obtain a better performance?**

- Worse performance in every method besides KNN which performs a little bit better. None of them has an acceptable performance.

**2. What happens if you do reshuffle and RFE? do you get better results than only reshuffling?**

- Better result but not acceptable. Lower average MSE.

**3. In the section Car Evaluation Quality, we performed the evaluation metrics for linear support vector machine, naive bayes, logistic regression and k nearest neighbors. As you can see at page 18, they do have a poor performance. Find out if there are ML algorithms that perform better on the data cars.csv data set. You may test decision trees and random forest as well as other type of SVM.**

- Decision trees perform better than all the others. We could not get random forest to work. Decision trees performs better because the method is considered non-parametric, making no assumption on the distribution of data and the structure of the true model. The other ones has linear kernels, and that’s why they are not suited for this.