# Mini-Rapport

Étudiant :

Lileke Steve

Enseignant :

Fabien Panloup

Cours :

Statistique en Grande Dimension et Apprentissage

EXERCICE 1 :

# Implémentation de l'algorithme des k plus proches voisins baggé

Principe :

On fixe un nombre B. On tire aléatoirement avec remise B sous-échantillons $b_i$ dans l'échantillon mère, i allant de 1 à B.

Sur chaque $b_i$, on construit un prédicteur k-ppv $f_i$

Dans le cas d'une régression, pour un x donné, sa prédiction est la moyenne des B prédictions du bagging. En classification, la prédiction de x découle de la réponse majoritairement prédite dans le bagging.

Suivant ce principe, on conçoit deux programmes pour effectuer la prédiction d'un échantillon test et expérimenter la validation croisée.

Les données sur lesquelles on met en application ces deux algorithmes sont les données Iris de Fisher. On fabrique deux bases de données, l'une servant à appliquer la classification pour prédire l'appartenance des fleurs aux classes 'setosa', 'versicolor', 'virginica' à partir des longueurs et largeurs des sépales et pétales ; et l'autre base de données pour prédire la largeur sépales dans le cas de la régression.

Le code commenté pour la prédiction d'échantillon test du knn baggé est knn_ech_test_pred.py dont la plupart de fonctionnalités sont importées du fichier knnBagge.py

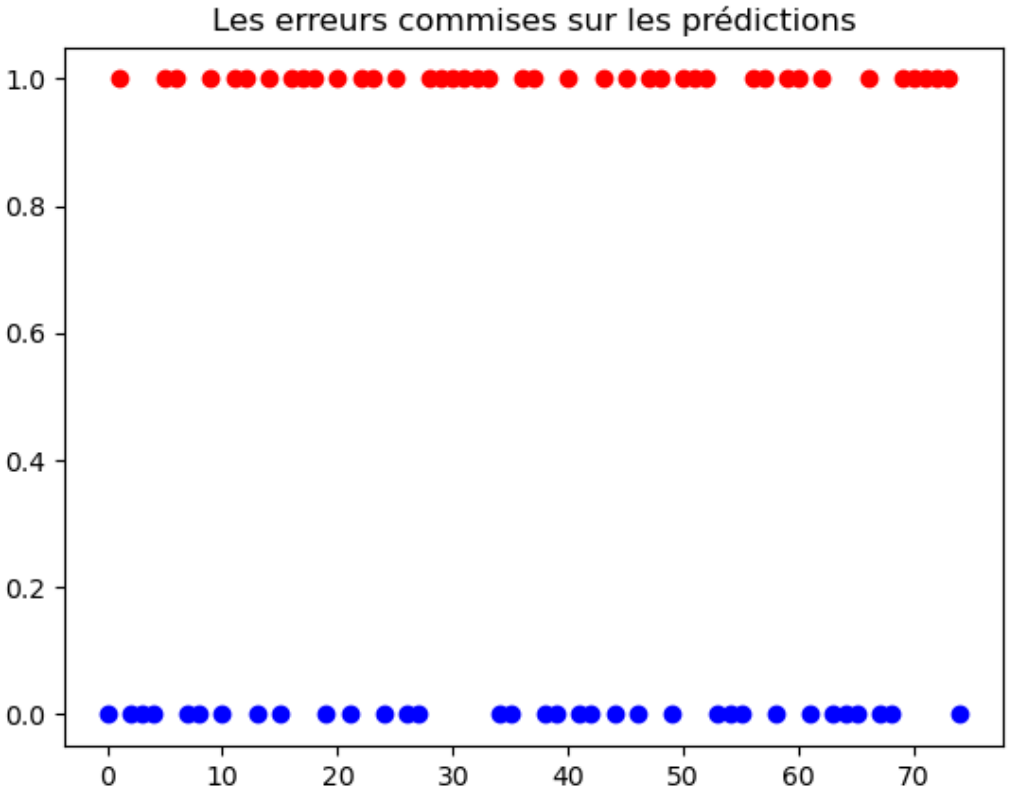
On rappelle que la base de données iris contient 150 individus, On en consacre la moitié pour entrainer le modèle baggé et l'autre moitié pour le test.

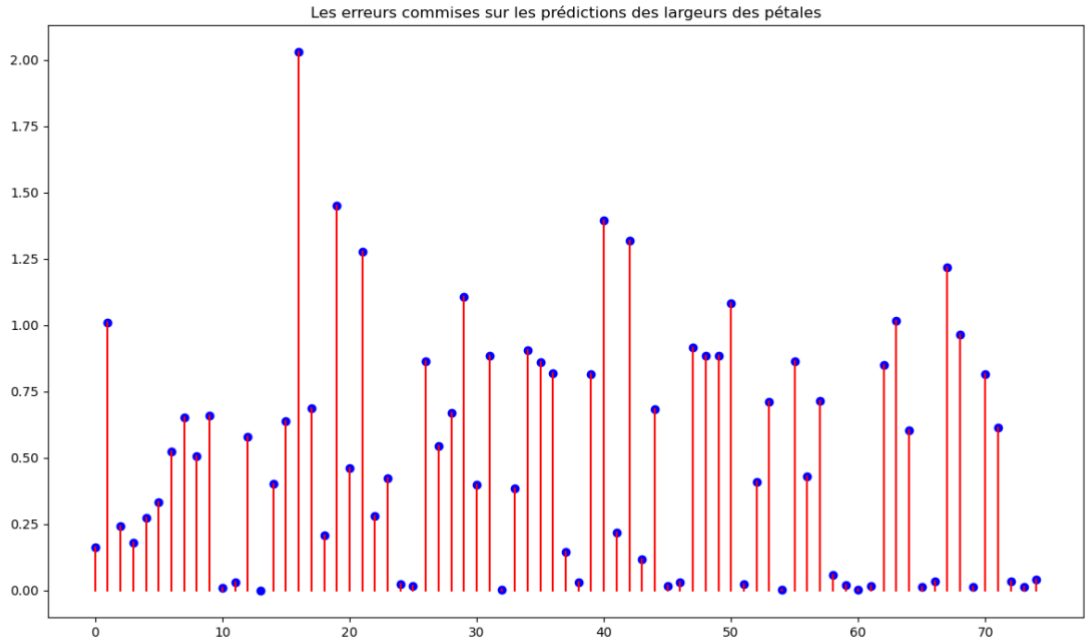
On, fixe B = 40, chaque sous échantillon dans le bagging est de taille 30 et on effectue les prédictions en fonctions des 3 plus proches voisins en utilisant la norme euclidienne.

On obtient les graphiques des erreurs suivants :

Les barres rouges sont les écarts entre la largeur des sépales prédites et les vraies largeurs. Le taux d'erreur = 51%

Dans le deuxième graphique, le points rouges indiquent les points mal prédits, avec un taux d'erreur de 54%



## Adaboost

En suivant le pseudo code du support de cours à la page 67, on construit un autre type de prédicteur des k plus proches voisins à la manière de l'adaboost.

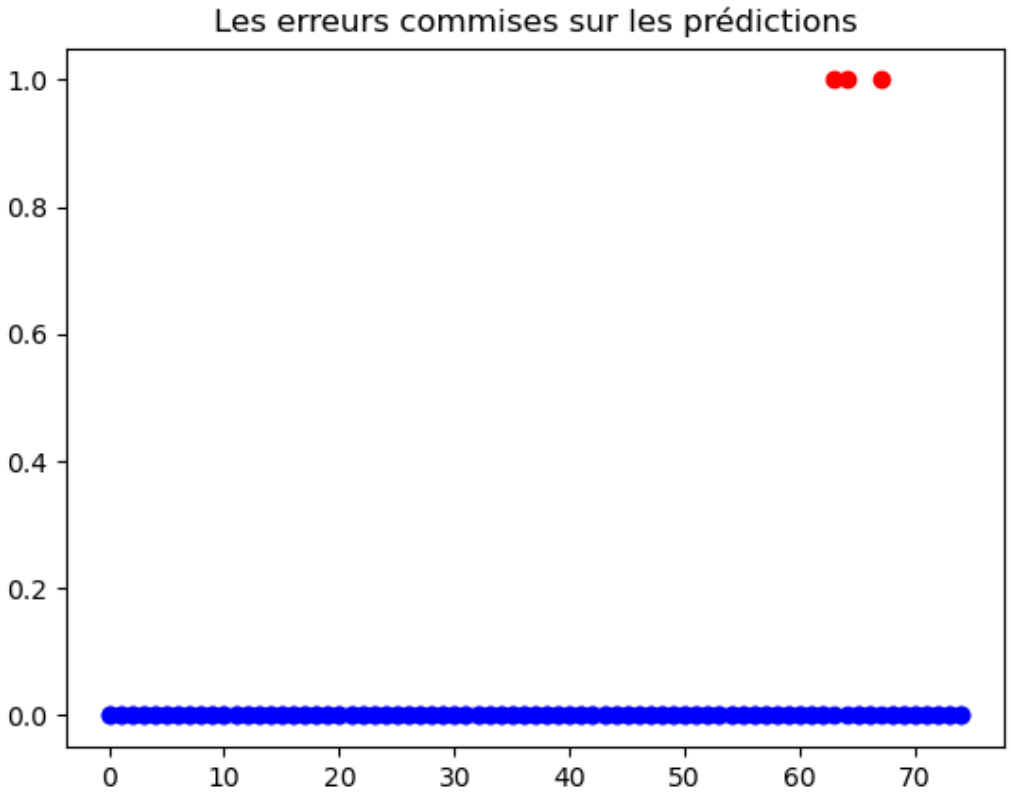
Ce modèle semble très performant, son taux d'erreur est au moins 25 fois plus petite que le modèle baggé.

Voici le graphique des erreurs sur un échantillon d'entrainement et de test :

Ici le nombre M d'itérations dans le boosting est de 4

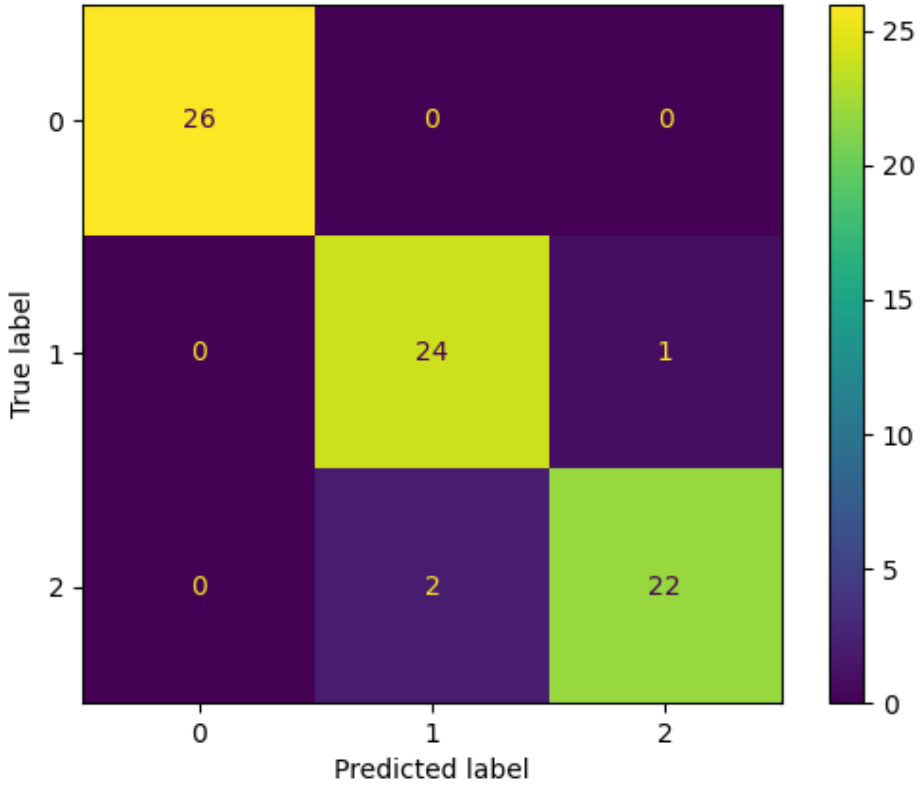
Taux d'erreurs de prédiction : 1.33%

Le code commenté pour ce programme est le knnAdaboost.py




Voyons ce que donne un algorithme des 3 plus proches voisins satndard sur un même jeu de données.

Taux d'erreur : 2 %




EXERCICE 2 :

# Prédiction de la réaction au traitement des patients souffrant du cancer de sein

Le code commenté pour l'intégralité de cet exercice est **exercice2.py**

1. Traitement de données

La faible représentativité des modalités 'pr_status: N', 'her2 status: N', 'pr_status: P' dans la variable réponse, respectivement 6, 3 et 1 sur les 278 enregistrements, nous amène à supprimer de la base de données les 10 patients qui correspondent à cette réaction.

On élimine également les variables peu importantes comme 'ethnicity' et 'ER_status: '.

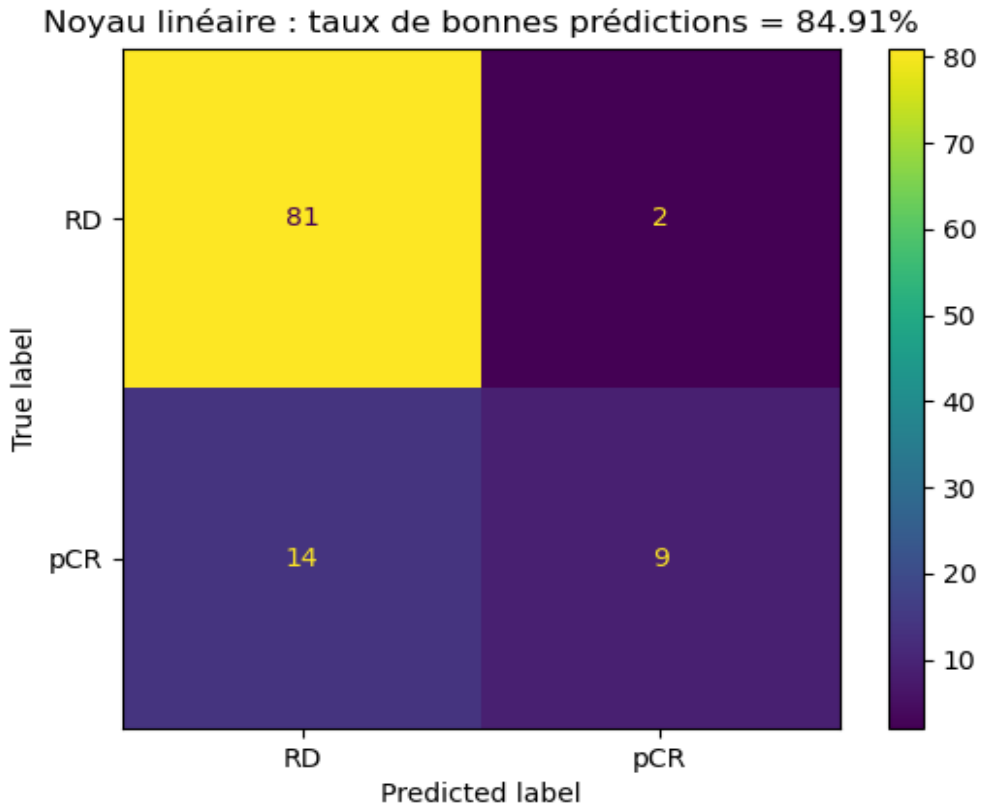
On se retrouve finalement devant un problème de classification binaire avec comme modalités sur la réaction des patients au traitement ‘RD’ et 'pCR’ sur un jeu de données de 265 individus.

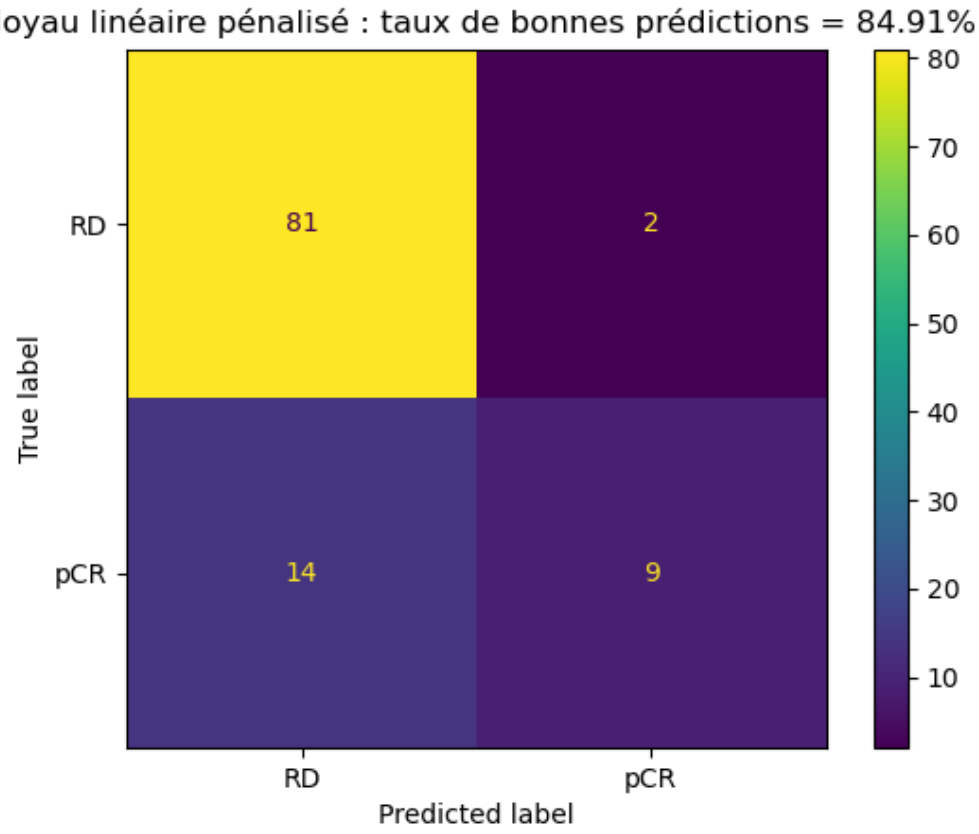
2. Modèles de classification

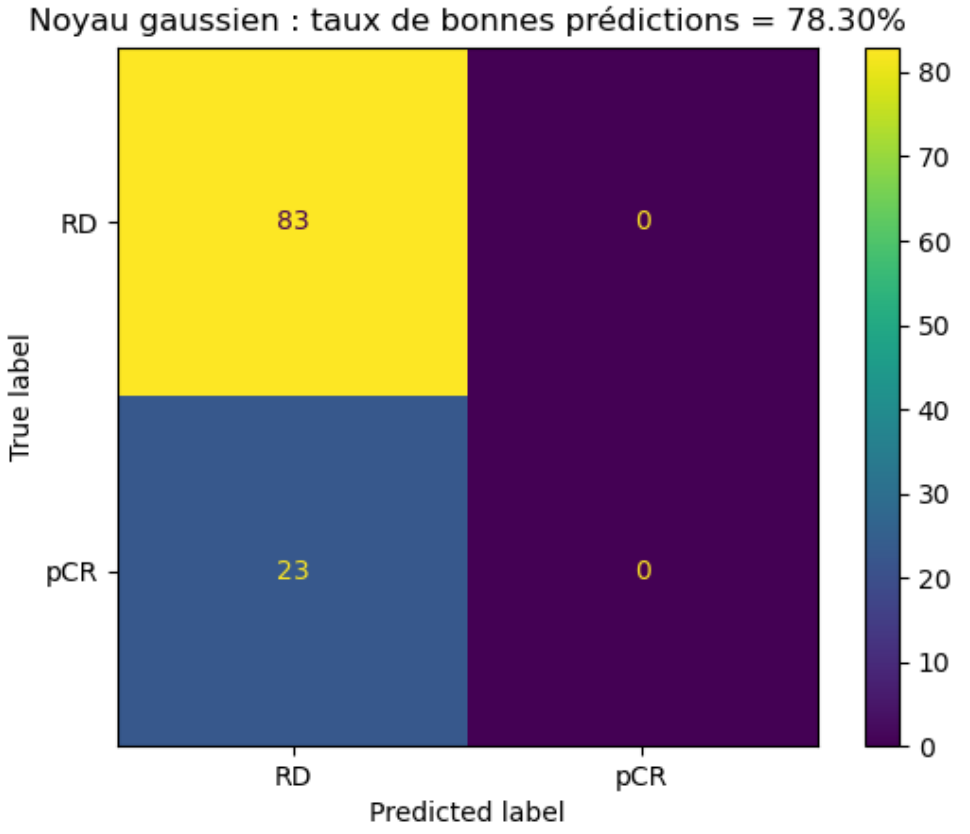
   On met en place deux modèles de classification : Les SVM et le Random forest

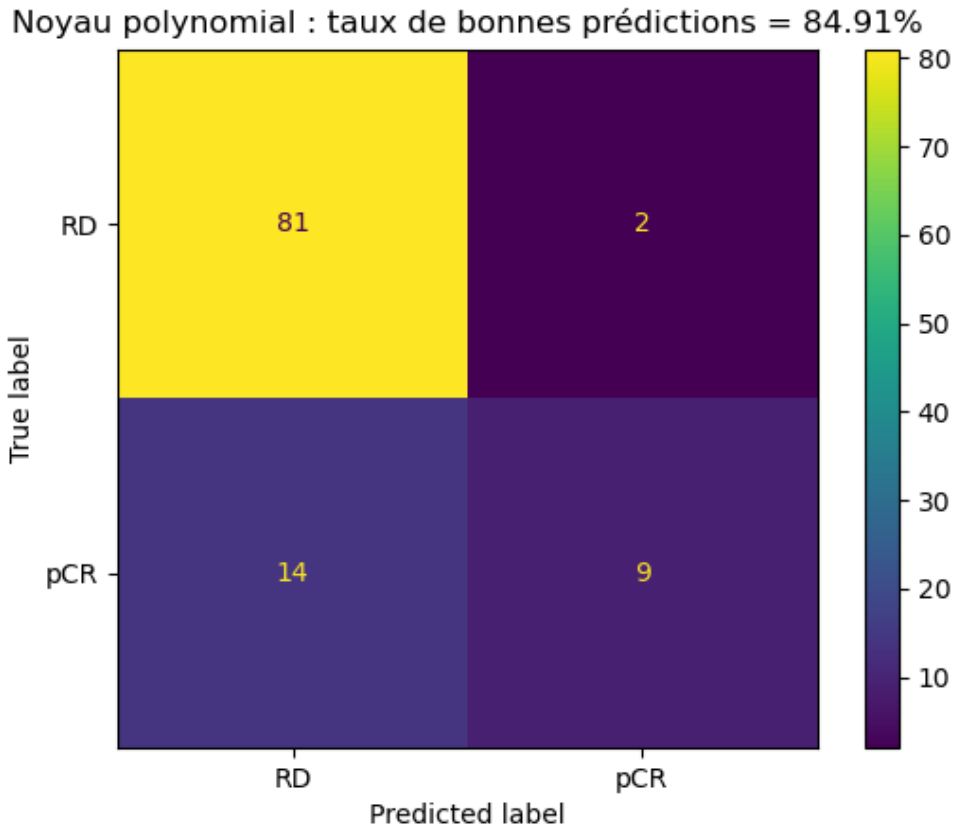
   Dans les deux situations, les modèles sont entrainés avec les 60% des individus et testés avec les 40 restants.

   Pour les svm, je teste un noyau linéaire standard c’est-à-dire le produit scalaire usuel, un noyau linéaire pénalisé, un noyau gaussien et un noyau polynomial de degré 2. Voici les résultats obtenus :

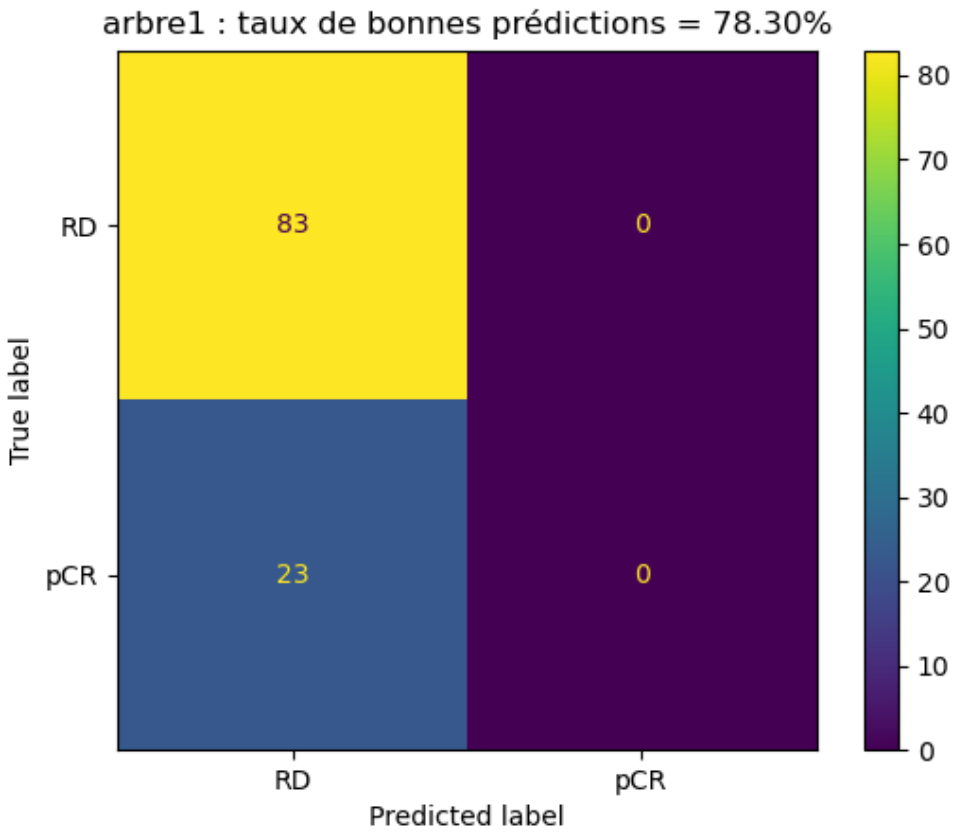


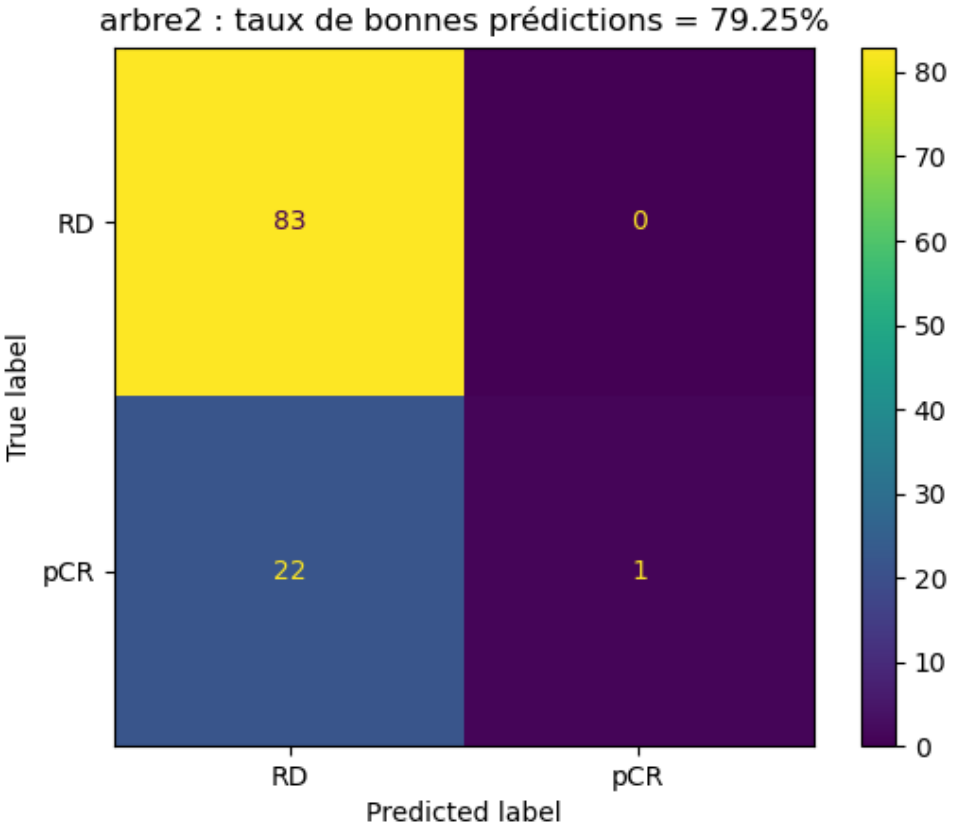

Noyau linéaire pénalisé : taux de bonnes prédictions = 84.91%
RD
pCR
True label
81
2
14
9
RD
pCR
Predicted label
80
70
60
50
40
30
20
10

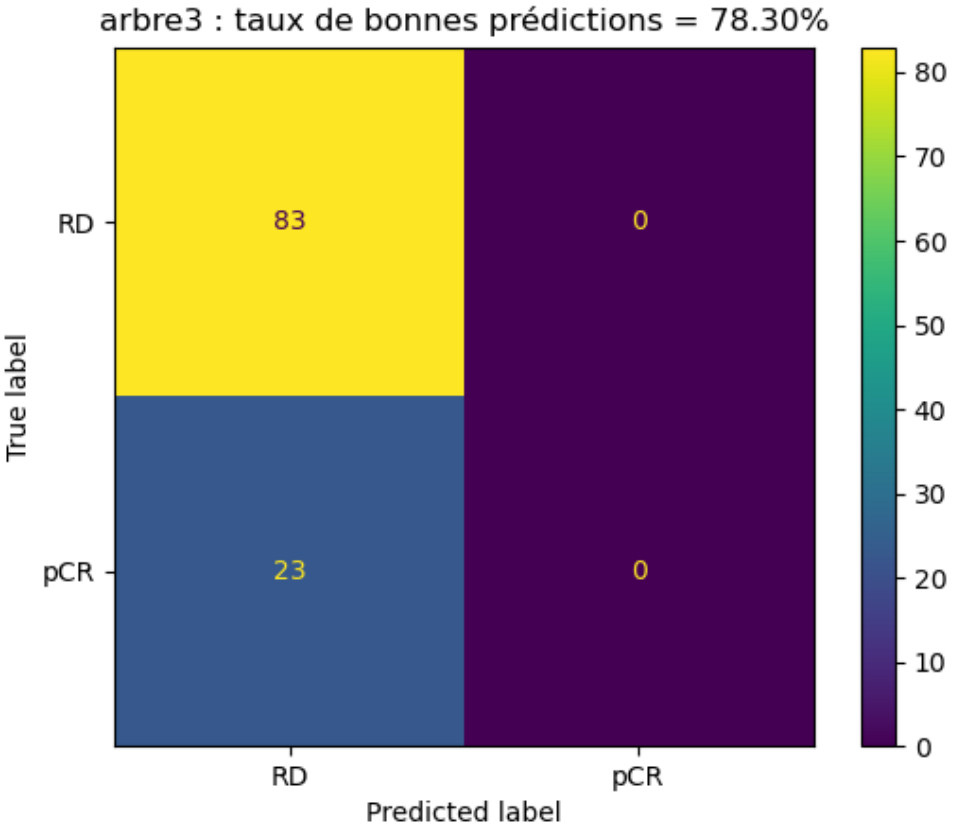

Noyau gaussien : taux de bonnes prédictions = 78.30%
RD
pCR
True label
83
0
23
0
RD
pCR
Predicted label
80
70
60
50
40
30
20
10
0




Pour les random forest, je teste 3 modélisations en variant la profondeur des arbres. Voici les résultats obtenus :




arbre2 : taux de bonnes prédictions = 79.25%
RD
pCR
True label
83
0
22
1
RD
pCR
Predicted label
80
70
60
50
40
30
20
10
0


arbre3 : taux de bonnes prédictions = 78.30%
RD
pCR
True label
83
0
23
0
RD
pCR
Predicted label
80
70
60
50
40
30
20
10
0

La modalité pCR est souvent mal prédite, même si les svm donnent 9 bonnes prédictions sur les 23 possibles.

## Exercice 3

Projet - Exercice 3 | page 1/4 | Lileke Bowaka Steve

Question 2.

On suppose $\mathcal{Y} = \{1, 2, \dots, K\}$ et que la fonction perte $\ell(y, y') = \mathbb{1}_{\{y \neq y'\}}$.

$L^* = \inf_{f : \mathcal{X} \to \mathcal{Y}} L(f)$ où $L(f) = \mathbb{E}[\ell(f(X), Y)]$.

a) Que représente $L^*$ ? Quelle interprétation peut-on en donner ?

$L^*$ représente le risque optimal de Bayes.

Etant donné une classe de prédicteurs $\mathcal{F} := \{f : \mathcal{X} \to \mathcal{Y}\}$, $L^*$ est le plus petit risque associé à la fonction perte $\ell$ : $(y, y') \mapsto \mathbb{1}_{\{y \neq y'\}}$.

b) Montrons que $L^*$ est un minimum et déterminons (par démonstration) sa valeur. Il suffit de prendre un prédicteur quelconque $f \in \mathcal{F}$ et de montrer que son risque est au moins égal à $L^*$.

Soit $f$ un prédicteur défini de $\mathcal{X}$ vers $\mathcal{Y}$.

$$\mathbb{E}[\ell(f(X), Y)] = \mathbb{E}[\mathbb{1}_{\{f(X) \neq Y\}}] = R_f.$$

$$= \sum_{k=1}^{K} \mathbb{E}[\mathbb{1}_{\{f(X) \neq k\}} \mathbb{1}_{\{Y = k\}}],$$

Appelons $P_{X,k} = \mathbb{P}(Y = k \mid X = x)$.

Pour un $x \in \mathbb{R}^p$ ponctuel donné,

$$\mathbb{E}[\mathbb{1}_{\{f(x) \neq k\}} \mathbb{1}_{\{Y = k\}}] = \mathbb{1}_{\{f(x) \neq k\}} \mathbb{P}(Y = k \mid X = x)$$

$= \mathbb{1}_{\{f(x) \neq k\}} P_{k,x}$.

Ainsi, $R_f = \mathbb{E}\left[\mathbb{1}_{\{f(x) \neq k\}} P_{X,k}\right]$.

Posons $\phi(x) = \min_{f: X \to Y} \mathbb{1}_{\{f(x) \neq k\}} P_{k,x} \quad \forall x, \forall k$.

$\phi = \inf_{f \in \mathcal{F}} \left\{ \mathbb{1}_{\{f(x) \neq k} P_{x,k}, x \in \mathbb{R}^p \right\}, \forall k$.

$= \inf_{f \in \mathcal{F}} \mathbb{1}_{\{f(x) = k\}} P_{X,k}$.

Par construction $f^* = \phi = \inf\{f \neq Y\}$.

$\Leftrightarrow \mathbb{1}_{\{f^* \neq Y\}} P_{X,X} \leq \mathbb{1}_{\{f \neq Y\}} P_{X,k}$

$\Leftrightarrow \mathbb{E}\left[\mathbb{1}_{\{f^* \neq Y\}}\right] \leq \mathbb{E}\left[\mathbb{1}_{\{f \neq Y\}}\right]$.

$\Leftrightarrow \mathbb{E}\left[\mathbb{1}_{\{f^* \neq Y\}}\right] \leq R_f$.

$L^*$ est le risque associé à $f^*$. $L^* = \mathbb{E}\left[\ell(f^*(X), Y)\right]$.

Donc $L^* \leq R_f, \forall f \in \mathcal{F}$.

Question 4

a) On décompose la quantité $L_n(g_n^*) - L^*$ en deux parties :

$L_n(g_n^*) - L^* = \underbrace{L_n(g_n^*) - \inf_{g \in \mathcal{G}} L(g)}_{\varepsilon_1} + \underbrace{\inf_{g \in \mathcal{G}}(L(g)) - L^*}_{\varepsilon_2}$

Lorsque $\mathcal{G}$ augmente, $\varepsilon_2$ décroit.

En effet, $\mathcal{G}$ augmente signifie que l'on ajoute des prédicteurs supplémentaires. On peut donc trouver un nouvel infimum qui minimise mieux la fonction $L$ que le faisait l'ancien infimum.

Concrètement, soit la classe $\mathcal{G}_2$ et $g_1^* = \inf_{g \in \mathcal{G}_2} L(g)$
et $\mathcal{G}_2$ contenant $\mathcal{G}_2$ et $g_2^* = \inf_{g \in \mathcal{G}_2} L(g)$.

page 3/4

$$g_1^* = \inf_{g \in \mathcal{G}_1} L(g) \geq g_2^* = \inf_{g \in \mathcal{G}_2} L(g)$$

$$\underbrace{g_1^* - L^*}_{\varepsilon_2 \text{ de départ.}} \geq \underbrace{g_2^* - L^*}_{\varepsilon_2 \text{ lorsque } \mathcal{G} \text{ a augmenté}}$$

$\varepsilon_2$ décroît lorsque $\mathcal{G}$ augmente.

b) $\varepsilon_1 = L_n(g_{\mathcal{G}n}^*) - \inf_{g \in \mathcal{G}} L(g)$

$$= \frac{1}{n} \sum_{k=1}^{n} \ell(g(X_i), Y_i) - \inf_{g \in \mathcal{G}} L(g).$$

$$\xrightarrow[n \to +\infty]{\text{loi des grands Nombres}} \mathbb{E}[\ell(g(X), Y)] - \inf_{g \in \mathcal{G}} L(g).$$

$$= L(g) - \inf_{g \in \mathcal{G}} L(g) = \underline{L(g) - L^*}$$

et si $\mathcal{G}$ est fini;

c) on a maintenant $\ell(y, y') = (y - y')^2$ et que $Y = f(X)$ où $f$ est une fonction quelconque de $\mathbb{R}^p$ vers $\mathbb{R}$.

$$\varepsilon_1 = L_n(g_n^*) = \frac{1}{n} \sum_{i=1}^{n} \ell(g_n^*(X_i), Y_i)$$

$$= \frac{1}{n} \sum_{i=1}^{n} (g_n^*(X_i) - Y_i)^2.$$

$$L^* = \inf_{f : X \to Y} L(f) = \inf_{f : X \to Y} \mathbb{E}[\ell(f(X), Y)]$$

$$= \inf_{f : X \to Y} \mathbb{E}[(f(X) - Y)^2].$$

$$\inf_{g\in\mathcal{G}} L(g) = \inf_{g\in\mathcal{G}} \mathbb{E}[\ell(g(X),Y)] = \inf_{g\in\mathcal{G}} \mathbb{E}[(g(X)-Y)^2].$$



Dans le cadre de la régression, $\mathcal{G} = L^2$.

d) Dans le cadre des méthodes pénalisées de Lasso, $\mathcal{G}$ correspond à un ensemble de fonctions convexes.

e) Dans le cadre des arbres de décision, plus la profondeur de l'arbre augmente, plus $\mathcal{G}$ augmente car l'arbre tend vers sa forme maximale. Ainsi, comme on a la question a, $\mathcal{E}_2$ décroit. $\inf_{g\in\mathcal{G}} L(g) \geq \inf_{g\in\mathcal{G}_{max}} L(g)$.

Ainsi, $\mathcal{E}_1$ croit.

f). Dans le cadre des réseaux de neurones, la méthode qu'on utilise pour calculer $f_n^*$ est la descente de gradient stochastique sur $L_n(g) = \frac{1}{n}\sum_{i=1}^{n} \ell(g(X_i), Y_i)$.