PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Modifications de l'algorithme SELECT() appliquées à une problématique du cancer du sein
Auteur: Author:	Louis-Marc Mercier
Date:	2017
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Mercier, L.-M. (2017). Modifications de l'algorithme SELECT() appliquées à une problématique du cancer du sein [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2862/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/2862/
Directeurs de recherche: Advisors:	Richard Labib, & François Soumis
Programme: Program:	Maîtrise recherche en mathématiques appliquées

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

MODIFICATIONS DE L'ALGORITHME `SELECT()` APPLIQUÉES À UNE PROBLÉMATIQUE DU CANCER DU SEIN

LOUIS-MARC MERCIER
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2017

© Louis-Marc Mercier, 2017.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MODIFICATIONS DE L'ALGORITHME `SELECT()` APPLIQUÉES À UNE PROBLÉMATIQUE DU CANCER DU SEIN

présenté par : MERCIER Louis-Marc
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. ADJENGUE Luc, Ph. D., président
M. LABIB Richard, Ph. D., membre et directeur de recherche
M. SOUMIS François, Ph. D., membre et codirecteur de recherche
M. PARTOVI NIA Vahid, Doctorat, membre

DÉDICACE

À la mémoire de Raynald Gagné (1946-2017),
merci de m'avoir guidé...

REMERCIEMENTS

Mes sentiments de gratitude vont à mes directeurs François Soumis et Richard Labib. Me procurer un stage en industrie, m'accorder un financement au long de ma maîtrise et être constamment disponible sont des gestes que j'ai particulièrement appréciés. Vous m'avez fait confiance et rendu plus indépendant.

Mes remerciements vont également à mes collègues et amis. De mon enfance à ma vie d'adulte, j'ai rencontré une multitude d'individus fascinants et chacun unique à sa façon. Vous m'avez apporté beaucoup et je ne serais pas le même sans vous. Un merci particulier à ceux qui m'ont soutenu et encouragé, vous m'avez donné la force de me dépasser.

À mes anciens enseignants en mathématiques, vous m'avez aidé à développer mon intuition mathématique tout en me transmettant la passion. J'espère pouvoir un jour redonner à des étudiants ce que vous m'avez donné. Une pensée va inexorablement à François-Simon Fauteux-Chapleau et David Adjavon. Assis sur vos épaules de géants, j'ai appris à mieux naviguer à travers les vignes de la jungle des mathématiques et de la statistique.

La qualité de ce mémoire ne serait pas la même sans les commentaires de plusieurs amis. Mes remerciements vont donc à : Lucie Desfontaines, Guillaume Gazil, Philippe Racette et Antoine Lefebvre-Brossard.

Je tiens à remercier mes parents et mes frères pour le soutien moral inconditionnel qu'ils m'ont offert. Il est évident que sans leur présence, mon parcours scolaire aurait été bien plus ardu et peut-être plus éphémère. En cette fin de parcours, je suis particulièrement reconnaissant envers mes parents de m'avoir encouragé à toujours apprendre.

En dernier, mais non la moindre, je remercie Hannatou pour m'avoir encouragé durant les moments difficiles et m'avoir fait croire à nouveau en l'amour. ♡

RÉSUMÉ

Les puces à ADN sont fréquemment utilisées pour diagnostiquer diverses maladies, dont le cancer. L'analyse des données récoltées par ces appareils mène à l'obtention d'une quantité impressionnante d'information sur les gènes de chaque patient. Cela conduit alors à la *malédiction de la dimensionnalité*. En des termes simples, on dispose de trop peu d'observations pour le nombre de variables explicatives. Les méthodes d'apprentissage automatique se voient alors limitées. Étant donné le nombre élevé de gènes, une hypothèse plausible est que certains gènes contiennent moins d'informations ou de l'information redondante. L'idée de l'algorithme `SELECT()` est de sélectionner les gènes les plus informatifs à l'aide de l'analyse en composantes principales et de la régression logistique. La sélection des variables est une étape cruciale dans le développement d'un modèle de prédiction. Par conséquent, il est intéressant d'étudier des versions modifiées de cet algorithme de sélection de variables.

L'objet de ce travail est donc d'étudier deux variations de `SELECT()` sur un ensemble de données de patients en rétablissement d'un cancer du sein (Gravier, 2010). Le chapitre 3 traite des notions nécessaires pour avoir une compréhension profonde de `SELECT()` et ses modifications. Le chapitre 4 vise à reconstituer le plus fidèlement possible l'algorithme `SELECT()` à la version de Ocampo-Vega (2016). Puis, des classificateurs sont développés, chacun selon une méthodologie qui lui est propre. C'est dans le chapitre 5 que l'originalité de cette recherche est décomposée en deux parties. Pour la première partie du chapitre 5, l'analyse en composantes principales est remplacée par une analyse en composantes principales creuse. Cette substitution nécessite une estimation du vecteur de pénalités $\boldsymbol{\lambda}$ qui est effectuée en tenant compte de contraintes. La seconde partie du chapitre 5 propose une substitution de la régression logistique bayésienne par l'algorithme `Boruta`. Pour chacune de ces parties, des classificateurs sont entraînés de la même façon qu'au chapitre 4.

Les performances des classificateurs sont finalement analysées afin de montrer l'amélioration de performance engendrée (pour les critères d'AUC et de précision) par nos algorithmes modifiés. Sur notre ensemble de données, nos résultats ne battent pas les résultats obtenus dans la littérature et les modifications de `SELECT()` ne semblent pas mener à une amélioration des performances. Ces modifications ont également été appliquées sur d'autres jeux de données (Tian, 2003 ; Singh, 2002) et aucune amélioration n'a été constatée. Les désavantages et des améliorations potentielles sont par la suite traités. Finalement, une synthèse du mémoire est effectuée et des avenues de recherche sont proposées.

ABSTRACT

DNA microarrays are a commonly used technology to diagnose various diseases including cancer. The analysis of the collected data from these devices gives an impressive amount of information for each patient. This leads to the *curse of dimensionality*, there are not enough observations for the number of explanatory variables. This causes limitations in the use of machine learning methods. A plausible assumption is that given the high number of explanatory variables, some of them contain less information or redundant information. Thus, variable selection should be used. The idea of the `SELECT()` algorithm is to select the most informative genes by using principal components analysis and logistic regression. The selection of variables is a crucial step in the development of a prediction model. Therefore, studying modifications of this algorithm could lead to interesting discoveries.

The aim of this work is to study two modified versions of `SELECT()` on a set of data from patients recovering from breast cancer (Gravier, 2010). Chapter 3 deals with important concepts that are necessary to have a deep understanding of `SELECT()` and its modifications. Chapter 4 aims at reconstructing the `SELECT()` algorithm as closely as possible to the version created by Ocampo-Vega (2016). Then, classifiers are developed, each according to a methodology of its own. It is in Chapter 5 that the originality of this research is broken down into two parts. The first part of Chapter 5 focuses on the first modification of the algorithm. In order to modify the set of selected variables, the principal component analysis is replaced by a sparse principal component analysis. This substitution requires an estimate of the penalty vector $\boldsymbol{\lambda}$ which is performed taking into account constraints. The second part of Chapter 5 is about a substitution of the bayesian logistic regression by the `Boruta` algorithm. For each of these parts, classifiers are trained in the same way as in Chapter 4.

The performance of classifiers is finally analyzed in order to show the improvement of performance generated (for AUC and precision criteria) by our modified algorithms. On our data set, the results are not better than the ones from the literature and the modifications of `SELECT()` do not seem to lead to an improvement of the performances. These changes have also been applied to other datasets (Tian, 2003; Singh, 2002) and no improvement has been observed. Disadvantages and potential improvements are subsequently discussed. The most instructive potential improvement is to combine the two modifications to obtain a new algorithm. Finally, a synthesis of the dissertation is carried out and avenues of research are proposed.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS ET NOTATION

Abréviations

ACP	Analyse en composantes principales
ACPC	Analyse en composantes principales creuse
ACPCG	Analyse en composantes principales creuse groupée
AIC	*Akaike information criterion*
AUC	*Area under the curve*
BIC	*Bayesian information criterion*
C.S.	Cauchy-Swartz
CFS	Correlation based Feature Selection
lasso	*Least Absolute Shrinkage and Selection Operator*
MCO	Moindres carrés ordinaires
MCP	*Minimax Concave Penalty*
MGB	Machine à *Gradient Boosting*
MLDA	*Modified Linear Discriminant Analysis*
MLDA	*Minimum Redundancy-Maximum Relevance*
MVS	Machine à vecteurs de support
RLP	Régression logistique pénalisée
ROC	*Receiver operating characteristics graph*
SCAD	*Smoothly Clipped Absolute Deviation*
s.c	sous la(les) contrainte(s)
VNS	Variable Neighborhood Search
SVD	*Singular Value Decomposition*
SZMI	Score Z maximal parmi les imitations

Notation

Pour les lecteurs moins familiers avec les mathématiques, la section suivante décrit la notation mathématique utilisée dans ce mémoire.

a	Un scalaire (réel)
$\mathbf{a}$	Un vecteur
$\mathbf{A}$	Une matrice
$\mathbf{A}^{-1,\top}$	Une matrice inversée puis transposée
$\mathbf{A}_{m\times n}$	Une matrice composée de m lignes et n colonnes.
a_{ij}	Élément sur la i-ième ligne et la j-ième colonne de la matrice A
$\mathbf{I}_n$	Une matrice identité avec n lignes et n colonnes.
$\text{diag}(\mathbf{a})$	Une matrice diagonale dont la diagonale principale est le vecteur $\mathbf{a}$
$\mathbb{R}$	L'ensemble des réels
$\mathbb{N}$	L'ensemble des naturels
$\mathbb{1}_{a<x<b}$	Variable indicatrice de x ($x = 1$ si $a < x < b$, $x = 0$ sinon).
$\mathbb{1}$	Vecteur de 1.
$\{0, 1, \ldots, n\}$	L'ensemble des nombres naturels de 0 à n
$[a, b]$	L'intervalle des nombres réels entre a et b (incluant a et b)
$(a, b]$	L'ensemble des réels entre a et b (excluant a et incluant b)
$L, L(.)$	Le lagrangien d'un problème d'optimisation
$\mathbf{A}^\top$	La transposée de la matrice A
$\text{tr}(\mathbf{A})$	La trace de la matrice A
$\text{Sign}(.)$	Donne le signe de l'expression en argument (-1 ou 1)
$\|\|.\|\|$	Norme euclidienne
$\|\|.\|\|_F$	Norme de Frobenius
$\frac{\mathrm{d}y}{\mathrm{d}x}$	Dérivée de y en fonction de x
$\frac{\partial y}{\partial x}$	Dérivée partielle de y en fonction de x
$\nabla_{\mathbf{x}} y$	Gradient de y en fonction de $\mathbf{x}$.
$\text{Cov}(f(x), g(x))$	Covariance entre $g(x)$ de $f(x)$ sous la fonction $P(x)$
$f : \mathbb{A} \to \mathbb{B}$	Une fonction f au domaine $\mathbb{A}$ et image $\mathbb{B}$
$\underset{\mathbf{x}}{\text{argmin}}\ f(\mathbf{x})$	L'ensemble des solutions (en $\mathbf{x}$) minimisant la fonction $f(\mathbf{x})$
$(a)_+$	La partie positive de a, c'est-à-dire $\max(0, a)$

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Le cancer est une maladie grave répandue partout à travers le monde. De nombreux types de cancer sont présents et, afin de déterminer un traitement approprié, l'identification du type de cancer est cruciale. Le développement de l'équipement médical a entraîné un changement drastique dans la façon dont les données sont collectées et analysées par les scientifiques. On peut notamment penser à l'apparition de puces à ADN qui permettent de fixer des molécules d'ADN sur des plaques solides. Ces plaques contiennent des milliers d'emplacements pour mesurer l'expression des gènes. L'image scannée à partir de la puce à ADN est analysée afin d'associer une valeur d'intensité à chaque sonde fixée sur la biopuce et ainsi de déterminer s'il y a eu une hybridation pour chaque sonde. Cela mène à l'obtention d'une quantité impressionnante d'informations génétiques pour chaque patient.

En transposant les informations des puces à ADN vers un ordinateur, un vecteur de valeurs est obtenu pour chaque patient. Chaque valeur dans ce vecteur est un nombre réel. Généralement, la longueur de ce vecteur sera de l'ordre des milliers. Cela fait en sorte que l'on disposera fréquemment d'un plus grand nombre de gènes que de patients. En disposant d'un vecteur de gènes pour chaque patient et de la réalisation d'un phénomène (la présence d'un cancer dans ce cas), on fera face à ce que l'on appelle un *problème de données à hautes dimensions.*

Face à une telle situation, les intérêts en apprentissage machine sont la sélection de variables et la classification. La sélection de variables consiste à identifier un sous-ensemble de variables qui permet à un classificateur de mieux prédire la variable de réponse. Une bonne sélection de variables permet de réduire la variabilité du modèle et d'obtenir un modèle plus interprétable. De plus, cette identification peut être utile dans certains contextes. Par exemple, quelqu'un pourrait s'intéresser aux gènes en lien avec l'apparition de la maladie de Parkinson (Hill-Burns, 2016). La classification consiste à modéliser une fonction qui prédira la classe associée à la variable de réponse selon les variables explicatives. Dans les problèmes avec des puces à ADN, le grand nombre de gènes dont on dispose peut compliquer les choses. En effet, l'analyse de données liée aux puces à ADN nous confronte souvent à la malédiction de la dimensionnalité. Ce phénomène survient lorsque peu d'observations sont disponibles par rapport au nombre de variables. Pour bien généraliser une observation, des exemples similaires sont nécessaires dans l'ensemble d'entraînement. Pour contourner ce problème, deux solutions sont reconnues : construire un classificateur puissant ou réduire le nombre de variables utilisées. Étant donnée la difficulté de calculs de la première solution, la seconde option sera préconisée.

CHAPITRE 2 REVUE DE LITTÉRATURE

2.1 Sélection de variables : introduction

La sélection de variables est une étape essentielle dans le développement de modèles liés à l'apprentissage automatique. De nos jours, les ensembles de données sont fréquemment composés d'un grand nombre de variables. Souvent, beaucoup de ces variables ne sont pas nécessaires et peuvent affecter négativement le processus de modélisation. En effet, un nombre élevé de variables ralentit les algorithmes, demande de la puissance informatique et peut être néfaste à la précision de l'algorithme. Plus particulièrement, l'inconvénient provient du fait que plusieurs algorithmes d'apprentissage montrent une diminution de la précision lorsque le nombre de variables est supérieur au nombre de variables optimal (Kohavi et John, 1997). Ce problème est connu sous le nom de problème minimal optimal (ou *minimal optimal problem*, en anglais)(Nilsson et al., 2007).

Les méthodes de sélection de variables peuvent être catégorisées en deux groupes : les filtres (ou *filters*, en anglais) et les enveloppes (ou *wrappers*, en anglais). D'un côté, les méthodes filtres peuvent être interprétées comme un prétraitement des données. Les variables sont sélectionnées en se basant sur leur relation avec la variable de réponse. Ce sont des caractéristiques statistiques qui sont analysées. Par exemple, on pourrait analyser la variabilité d'une variable et décider de l'exclure si elle ne change pas suffisamment. Explorons un exemple de cette situation : imaginons un ensemble de données sur les passagers du Titanic pour lequel nous tentons de prédire qui survit selon des variables explicatives. Si l'on dispose de la variable âge et que tout le monde a le même âge, alors cette variable n'apporte aucune contribution pour la prédiction. La variable ne devrait donc pas être sélectionnée. De l'autre côté, les méthodes enveloppes sont associées à un algorithme d'apprentissage automatique. L'importance de la variable reposera sur son résultat obtenu avec sa méthode d'apprentissage. Par exemple, une forêt d'arbres aléatoires permet d'obtenir les rangs associés aux variables (en termes d'importance) pour la classification effectuée (Breiman, 2001). Par conséquent, une sélection de variables peut être faite.

Diverses méthodes de base peuvent être utilisées. Parmi les méthodes filtres, un filtre ANOVA est tel que si la distribution d'une variable varie peu par rapport à la variable de réponse, alors celle-ci doit avoir un faible pouvoir prédictif. Traditionnellement, la sélection de variables s'effectue en regardant les gènes individuellement. Mais cela est désavantageux, car un manque de corrélation directe entre une variable donnée et la décision n'est pas une preuve que cette variable n'est pas importante lorsqu'elle agit avec d'autres variables (Guyon et Elis-

seeff, 2003). Dans les cas de problèmes génétiques, il est difficile d'explorer les interactions entre les gènes étant donné leur grand nombre. Cependant, il existe des méthodes mesurant l'importance des variables en tenant compte de leurs interactions comme l'algorithme RELIEF (Kira et Rendel, 1992 ; Kononenko et al., 1997).

La littérature sur la sélection de variables est très vaste. Par conséquent, nous détaillons essentiellement la littérature se rapprochant de la problématique étudiée, c'est-à-dire les problèmes composés de données génétiques liées au cancer. C. Aliferis et al. (2001) ont utilisé l'élimination de variables par récursion (ou *recursive feature elimination*, en anglais) et l'association univariée (ou *univariate association*, en anglais) dans leur sélection de variables concernant le cancer du poumon. Dans le cadre d'un projet de classification de divers types de cancer, Ramaswany et al. (2001) ont combiné l'élimination de variables par récursion à des machines à support vectoriel. Un sélectionneur de variables basé sur la corrélation combiné à certaines méthodes de classification a été expérimenté par Wang et al. (2005). Sharma et al. (2012) ont d'abord groupé les gènes, puis ils ont identifié les gènes informatifs de chacun des groupes. Les gènes informatifs ont été extraits et combinés ensemble. Fakoor et al. (2013) ont proposé une méthode de classification du cancer qui se base sur l'apprentissage non supervisé et l'apprentissage profond.

D'autres approches moins conventionnelles ont aussi été employées. Ces approches consistent à développer des algorithmes composés de plusieurs étapes. Il est fréquent que ces algorithmes soient spécifiques à certains types de problèmes d'analyse de données. El Akadi et al. (2011) ont utilisé l'algorithme Redondance minimale - Pertinence maximale (ou *Minimum Redundancy-Maximum Relevance*, en anglais) et un algorithme génétique. Zhou et al. (2004) ont approché le problème de sélection de variables avec une approche bayésienne. Yu et al. (2009) ont utilisé une optimisation par colonie de fourmis pour sélectionner les gènes importants. Tong et Schierz (2011) ont combiné un algorithme génétique et un réseau de neurones pour effectuer leur sélection de variables.

2.1.1 Sélection de variables par la régression

Dans le cas d'un modèle linéaire, une sélection de variables naïve consisterait à expliciter tous les modèles possibles et à évaluer leur efficacité à l'aide d'un critère comme AIC (*Akaike information criterion*, en anglais) ou BIC (*Bayesian information criterion*, en anglais) de façon à choisir le meilleur sous-ensemble de variables (Furnival et Wilson, 1974). Malheureusement, le nombre de sous-ensembles est de 2^p et grandit exponentiellement à mesure que le nombre de variables p augmente, sans compter que ce type d'approche est numériquement impraticable à partir d'environ 30 ou 40 variables. Cette méthode sera donc qualifiée de méthode

numériquement instable. Pour cette raison, des procédures comme la sélection descendante (ou *backward selection*, en anglais) et la sélection ascendante (ou *forward selection*, en anglais) furent créées. Or, il se trouve que ces procédures violent des principes d'estimation statistique et de test d'hypothèses (Wilkinson et Dallal, 1981). En effet, les tests sont biaisés puisqu'ils sont basés sur les mêmes données, les valeurs-p sont faussement petites, les coefficients du modèle sont biaisés, etc. Sans compter qu'à de hautes dimensions, ces méthodes sont instables et conduisent à des résultats très variables.

Une façon de résoudre ces problèmes consiste à utiliser la régression pénalisée. La régression pénalisée consiste à appliquer un terme de régularisation à la fonction de perte de la régression linéaire multiple. La régression linéaire multiple est un modèle cherchant à établir une relation linéaire entre une variable de réponse et des variables explicatives. Explicitons les formules dans l'idée de mieux comprendre la régularisation :

$$M(\boldsymbol{\beta}) = L(\boldsymbol{\beta}|\mathbf{x}) + \lambda P(\boldsymbol{\beta}) \tag{2.1}$$

où $M(\boldsymbol{\beta})$ est la fonction objectif, $L(\boldsymbol{\beta}|\mathbf{x})$ est la fonction de perte, $P(\boldsymbol{\beta})$ est la fonction de pénalisation et λ est le terme de régularisation. La régularisation peut également être interprétée comme l'ajout d'une contrainte au problème de minimisation de la fonction de perte de la régression linéaire multiple.

$$\min_{\boldsymbol{\beta}} \quad L(\boldsymbol{\beta}|\mathbf{x}) + \lambda P(\boldsymbol{\beta}) \Leftrightarrow \min_{\boldsymbol{\beta}} \quad L(\boldsymbol{\beta}|\mathbf{x}) \quad \text{s.c} \quad P(\boldsymbol{\beta}) \in \mathcal{H} \tag{2.2}$$

où $\mathcal{H}$ est une région agissant sous la forme d'une contrainte sur $\boldsymbol{\beta}$. La solution obtenue à ce problème d'optimisation est un vecteur $\hat{\boldsymbol{\beta}}$ de scalaires qui représente les poids accordés aux variables. Cette méthode procure une sélection de variables continue et peut conduire à l'obtention d'un modèle plus interprétable. Un des plus anciens modèles de régression pénalisée est la régression de *Ridge* (Hoerl et Kennard, 1970). Cette méthode consiste à appliquer une pénalisation sur la valeur de la norme euclidienne appliquée sur $\boldsymbol{\beta}$. Cette technique mène à l'obtention de coefficients de $\boldsymbol{\beta}$ plus près de zéro par rapport aux coefficients obtenus avec la régression linéaire multiple. Également, la résolution de ce problème d'optimisation mène toujours à une solution unique, même lorsque le nombre de variables dépasse le nombre d'observations. Malgré ces très belles propriétés, il y a un vice caché ! Cette régression approche la taille des coefficients près de zéro, mais ceux-ci ne seront pas égaux à zéro. Par conséquent, la sélection de variables ne peut pas être effectuée efficacement.

Pour résoudre ce problème, Tibshirani (1996) a introduit la régression *lasso*. Au lieu d'appliquer la pénalisation en utilisant la norme euclidienne sur $\boldsymbol{\beta}$, elle est appliquée avec la norme

ℓ_1 sur $\boldsymbol{\beta}$. Les pénalisations de *Ridge* et de *lasso* sont semblables et il semble intéressant d'explorer leur généralisation. Il s'agit d'une famille de pénalités connue sous le nom de pénalités de *bridge*. Proposée par Frank et Friedman en 1993, l'idée est de baser la pénalisation sur la norme ℓ_λ, où λ est un hyperparamètre dont la valeur est positive. On retrouve plusieurs autres types de régressions pénalisées. Fan et Li (2001) ont introduit la pénalité SCAD. Zhang (2007) a proposé la MCP. Ces deux pénalités visent à choisir les variables influentes de façon à avoir un modèle de régression linéaire multiple qui exclut d'avance les variables de moindre importance (aussi appelé modèle oracle). Il se trouve que cette propriété est atteinte asymptotiquement par les modèles MCP et SCAD (Fan et Li, 2001 ; Zhang, 2007).

Dans certains domaines, comme la génétique, les variables peuvent être reliées en groupes et avoir un impact. Par exemple, certains gènes peuvent avoir une structure commune. Afin d'exploiter ces relations, des modèles de régression pénalisée à bisélection ont été développés. Yuan et Lin (2006) ont proposé le lasso groupé (ou *group lasso*, en anglais). Cependant, ce modèle comprend des défauts comme le biais des coefficients vers zéro et son incapacité à choisir des variables dans un groupe. Pour résoudre ce problème, le *group bridge* a été créé (Huang et al., 2007). L'approche proposée est une méthode de régularisation permettant de créer des groupes et de sélectionner des variables dans ces groupes. Cette approche a la propriété d'oracle, dans le sens qu'elle peut sélectionner correctement des groupes importants avec une probabilité convergente vers un. Le MCP groupé (ou *group MCP*, en anglais) a aussi été créé (Breheny, 2009). Les hypothèses de regroupement sont moins sévères que celles du groupe lasso et *group bridge*, et montrer qu'il fonctionne mieux que ces méthodes concurrentes dans des situations où l'intérieur des groupes de variables est creux ($> 50\%$) . Breheny (2015) a développé le lasso groupé exponentiel (ou *group exponential lasso*, en anglais). Celui-ci a plusieurs avantages statistiques et de calcul par rapport aux pénalités de groupe proposées précédemment, comme le groupe lasso, le *group bridge* et le MCP composite.

Les méthodes décrites permettent de sélectionner des variables dans des groupes. Or, il faut définir les groupes. L'idée est de localiser des groupes de gènes ayant des expressions similaires dans l'ensemble de données (Alon et al., 1999 ; Ben-Dor et al., 1999 ; Eisen et al., 1999 ; Michaels et al., 1998 ; Spellman et al., 1998). Étant donné le nombre important de variables, l'utilisation d'une méthode de groupage serait une stratégie intéressante. On peut penser à de nombreuses techniques : k-moyennes, PAM, CLARA, groupage hiérarchique, SOM, etc. Witten et Tibshirani (2010) ont présenté le groupage creux (*sparse clustering*) avec une application concernant la sélection de gènes dans une étude sur le cancer du sein. Plus tard, Tan et Witten (2013) ont généralisé le groupage creux pour en faire une procédure de bigroupage, c'est-à-dire en groupant les variables et les observations.

2.2 Classificateurs

Après la sélection de variables, l'implantation d'un ou de plusieurs classificateurs est nécessaire. Il est nécessaire de scruter les classificateurs fréquemment utilisés afin de développer une intuition pour guider nos choix. De nombreux classificateurs ont déjà été mis en oeuvre pour les problèmes de classification liés à divers types de cancer. On pense notamment à l'usage de machines à support vectoriel (Feng Chu et Lipo Wang, 2005 ; Furey et al., 2000 ; Mukherjee et al., 1999), de réseaux de neurones (Huynh et al., 2007 ; Linder et al., 2007 ; Liu et al., 2004), d'optimisation par colonies de fourmis (Chiang et al., 2008) et d'algorithmes génétiques (Diaz et al., 2014 ; Dolled-Filhart et al., 2006).

Furey et al. (2000) ont traité du cancer des ovaires. Ils ont trouvé que les machines à support vectoriel donnaient des résultats similaires à d'autres approches basées sur le perceptron. Chol et al. (2003) ont essayé différentes méthodes de classification : réseaux de neurones, k plus proches voisins, machines à support vectoriel et une carte auto-organisée (ou *self-organizing map*, en anglais) avec une structure adaptable. Leurs ensembles de données concernaient entre autres le cancer du côlon et la leucémie. Les meilleurs classificateurs obtenus sont les réseaux de neurones et la méthode des k plus proches voisins.

Chou et al. (2013) ont testé les réseaux de neurones, une forêt d'arbres aléatoires, la régression logistique et des hybrides de ces modèles. La meilleure précision a été obtenue à l'aide de réseaux de neurones. Une approche récente est l'apprentissage machine extrême (*Extreme Machine Learning*) (Revathi et Sumathi, 2014 ; Zhang et al., 2007). Les résultats trouvés sont semblables à ceux que l'on obtiendrait avec les méthodes traditionnelles. Le classificateur de Bayes naïf est populaire, mais exige des conditions fortes d'indépendance (Huang et al., 2003). Les réseaux bayésiens sont aussi utilisés. Il s'agit d'un type de modèle probabiliste utilisant des graphes acycliques (Friedman et al. 1997). Friedman et al. (2000) ont rédigé un article sur l'utilisation de réseaux bayésiens pour décrire les interactions entre les gènes. Trajdos et al. (2014) ont développé une méthode de sélection de variables (de type filtre) à trois étapes : un groupage selon les variables, l'utilisation d'une multitude de critères (*Borda count*) d'évaluation et l'utilisation de couverture de Markov. Pour des problèmes génétiques liés aux puces à ADN, Garcìa-Torres et al. (2015) ont développé l'algorithme de prédiction *GreedyPGG* basé sur un modèle de couverture de Markov. Witten et Tibshirani (2009) ont développé une technique de régression pénalisée et de classification pour des problèmes à haute dimensionnalité.

2.2.1 Recherche et données

L'ensemble de données est composé de 168 patients avec 2905 gènes et provient d'une étude du cancer du sein (Gravier et al., 2010). En 2010, Gravier et al. ont examiné des petits carcinomes canadairs invasifs n'atteignant pas les ganglions lymphatiques axillaires (T1T2N0) pour prédire les métastases du carcinome du sein ganglionnaire négatif. Les patients ont été examinés sur une période de cinq ans. 111 patients sans aucun événement après le diagnostic ont reçu une étiquette positive et 57 patients avec des métastases précoces ont reçu une étiquette négative. Les variables ont subi un changement d'échelle avec la transformation logarithmique en base 2. L'ensemble de données peut être récupéré en utilisant la librairie `datamicroarray` de R.

Le sujet de recherche consiste à construire un modèle de prédiction déterminant si un patient est atteint du cancer du sein en se basant sur une séquence de son ADN. Il s'agit d'un problème d'*apprentissage supervisé*. Plus précisément, c'est un problème de classification à deux classes : soit la personne est malade, soit elle est en bonne santé.

Cet ensemble de données a fréquemment été utilisé pour tester des modèles de prévision (voir Tableau 2.1). Ramey et Young (2013) ont utilisé des méthodes de régularisation appliquées à une fonction linéaire discriminante. En utilisant 500 variables explicatives, la meilleure méthode fut la méthode d'analyse discriminante modifiée (MLDA). Park et Kim (2014) ont sélectionné les variables à l'aide d'une méthode de sélection basée sur les composantes principales des vecteurs de poids et la prédiction a été faite avec la méthode des k plus proches voisins. Strobl et Visweswaran (2014) ont utilisé l'algorithme ExcD. Garcìa-Torres et al. (2015) ont développé des algorithmes de sélection de variables et de groupage à des fins de prédiction. La meilleure méthode fut CVNS qui est une variante de la recherche de variables par le voisinage (VNS). Arias-Michel et al. (2015) ont sélectionné des variables à l'aide de couvertures de Markov et avec la méthode de sélection de variables basée sur la corrélation (ou *Correlation based Feature Selection*, en anglais) . En effectuant ces expériences, la plus haute précision est obtenue en utilisant la méthode FCBF (*Fast Correlation based Filter*, en anglais). Huertas et Ramirez (2016) ont testé différentes méthodes de sélection de variables, mais leur meilleur résultat était obtenu sans sélection de variables et avec une machine à support vectoriel. Svensson (2016) a obtenu ses meilleurs résultats avec une machine à vecteurs de support et la régression *elastic net*.

En raison de ses multiples succès sur nos données, la machine à vecteurs de support est un candidat intéressant pour cette recherche. Un autre candidat potentiel aurait été un réseau de neurones. Cependant, ses résultats se sont révélés intéressants pour d'autres problématiques de cancer. Par conséquent, nous ne sommes pas en mesure de suspecter une grande efficacité

pour ce classificateur. De plus, un autre problème est son grand nombre d'hyperparamètres à optimiser (nombre de couches cachées, les fonctions d'activation, taux d'apprentissage, fonction de perte, etc.). On choisira d'employer une régression logistique ou une machine à *gradient boosting*. D'un côté, la régression logistique est un modèle linéaire et possède peu d'hyperparamètres à optimiser. De l'autre côté, la machine à *gradient boosting* est composée de peu d'hyperparamètres et détecte des relations non linéaires.

Tableau 2.1 Meilleurs résultats des modèles de prédictions de différents auteurs sur l'ensemble de données de Gravier (2010). La précision et l'aire sous la courbe ROC sont définies dans les sections 3.8.2 et 3.8.3.

Auteur	Année	Précision	Aire sous la courbe ROC
Ramey et Young	2013	**0.773**	–
Park et Kim	2014	0.665	–
Strobl et Visweswaran	2014	0.733	–
Garcìa-Torres	2015	0.744	–
Arias-Michel	2015	0.762	–
Huertas et Ramirez	2015	0.752	–
Svensson	2016	–	**0.880**
Lòpez et Maldonado	2017	–	0.795

2.2.2 Algorithme `SELECT()`

Développé par Ocampo-Vega et al. (2016), l'algorithme `SELECT()` a permis de réduire le nombre de variables et d'augmenter la précision de classification de plus de 10% dans tous les cas traités dans leur article (voir Algorithme 1). L'ensemble d'entraînement est composé de n observations qui sera représenté comme $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$. Le terme $\mathbf{x}_i$ représente la séquence génétique du $i^{\text{ème}}$ patient et y_i est une variable indicatrice sur la présence du cancer du sein chez ce patient. Dans l'algorithme, plusieurs valeurs de paramètres doivent être choisies. Le paramètre v est un nombre entier qui représente le pourcentage de la variance totale qui est retenue pour l'analyse en composantes principales (ACP). Le paramètre μ, un nombre réel, est un seuil. Si le poids d'un gène est inférieur à ce seuil, ce poids est considéré négligeable et est mis à zéro.

La première étape consiste à effectuer une analyse en composantes principales (ACP) et de conserver les q premières composantes responsables de v% de la variance. On note que chacune des composantes principales est une combinaison linéaire des variables. La deuxième étape est d'appliquer une régression logistique sur les nouvelles composantes et de conserver les d composantes les plus importantes de la matrice $\mathbf{Q}$. Au cours de la troisième étape, les d composantes sont analysées afin d'éliminer les gènes avec trop de poids égaux à zéro. Ainsi,

la sélection de variables est faite. Finalement, un nouvel ensemble de données $\{\mathbf{z}_i\}_{i=1}^{p}$ est créé à l'étape 4. Cet ensemble de données est le résultat de l'application de l'ACP (pour lequel q_2 composantes sont retenues) sur le nouvel ensemble d'entraînement.

Algorithme 1 : `SELECT()` trouve le sous-ensemble de gènes à utiliser pour la classification.

Entrée : $\mathcal{T}, v \in \mathbb{N}, \mu \in \mathbb{R}$
Sortie : $\mathbf{Q}_2$, l'ensemble des composantes choisies.
1 $\mathbf{Q} \leftarrow \texttt{ACP}(\{\mathbf{x}_i\}_{i=1}^{n}, v)$
2 $\mathbf{Q}_1 \leftarrow \texttt{LOGIT}(\mathbf{Q}, d)$
3 $\{\mathbf{z}_i\}_{i=1}^{n} \leftarrow \texttt{EnleverAttributs}(\mathbf{Q}_1, \mu)$
4 $\mathbf{Q}_2 \leftarrow \texttt{ACP}(\{\mathbf{z}_i\}_{i=1}^{n}, q_2)$
5 **Retourner** $\mathbf{Q}_2$

2.2.3 Modèle

L'algorithme décrit précédemment combine l'analyse en composantes principales et la régression logistique. Le succès de l'ACP est dû à deux éléments clés. Premièrement, les composantes capturent la variabilité en minimisant la perte d'information. Deuxièmement, les composantes sont non corrélées entre elles. L'ACP a comme inconvénient que chaque composante est une combinaison linéaire des variables, ce qui entraîne que les contributions associées aux variables sont difficiles à interpréter. À la suite de l'application de l'ACP, la régression logistique est utilisée et les d variables ayant les plus grands coefficients (en valeur absolue) sont choisies. Cette approche est discutable, car les combinaisons linéaires [1] des composantes principales sont différentes. Par conséquent, les échelles des composantes sont différentes. D'ailleurs, c'est l'une des raisons pour laquelle il n'est pas correct de déterminer l'importance des variables de la régression linéaire multiple à l'aide de ses coefficients. En raison de ces inconvénients, l'algorithme devrait être modifié.

Le but est d'effectuer une *meilleure* sélection de variables. Cela mènerait alors à de meilleures performances avec les classificateurs. Or, améliorer la performance des classificateurs est très utile et possède de nombreux avantages. Par exemple, un médecin pourrait adapter son traitement médical si son classificateur indique que le patient sera atteint du cancer. Les modifications suivantes sont appliquées à l'algorithme `SELECT()`.

Premièrement, Zou et al. (2004) ont développé une approche nommée analyse en composantes principales creuse (ACPC). Celle-ci est plus facile à interpréter et est basée sur le fait que l'ACP peut être écrite sous la forme d'un problème d'optimisation lié à la régres-

1. La combinaison linéaire associée à la $i^{\text{ième}}$ composante principale est $\mathbf{q}_i = \sum_j \alpha_j \mathbf{x}_j$

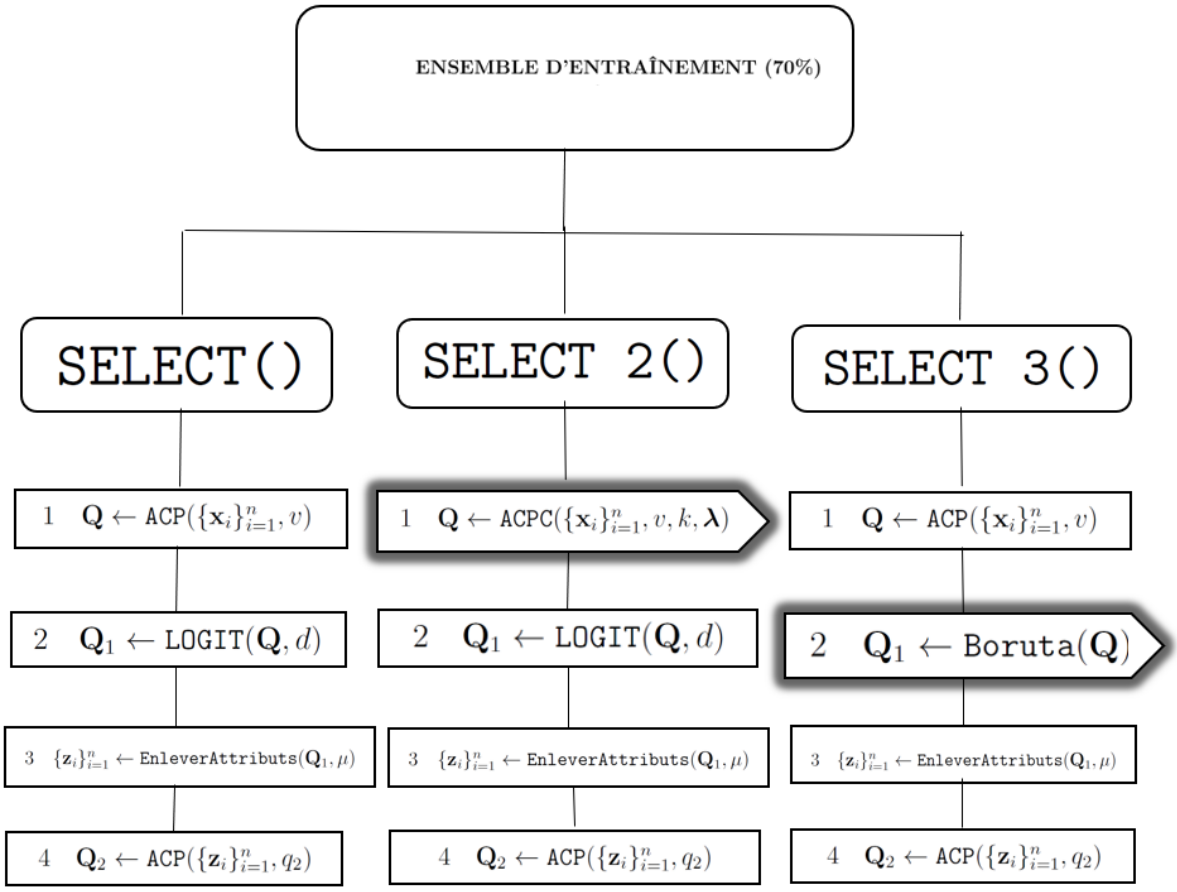
sion. Diverses approches ont été utilisées dans l'objectif de résoudre ce problème. Plusieurs implémentations ont également été faites sur R (librairie `elasticnet` (2012) ; librairie `PMA` (2013)). Développée par Zou et Hastie, c'est la librairie `elasticnet` qui sera utilisée dans cette recherche. D'ailleurs, Johnstone et Lu (2009) affirment qu'il est préférable d'effectuer une réduction de dimensionnalité dans une base supposant que les données ont une représentation creuse. Deuxièmement, Kursa et Rudnicki (2010) ont développé un l'algorithme de sélection de variables `Boruta` qui utilise une forêt d'arbres aléatoires. Cela mène à l'obtention de mesures d'importance pour les variables et l'algorithme détermine statistiquement les variables à conserver. Cette méthode est attrayante en raison de sa capacité de sélection qui est : sans biais, numériquement stable, capable de tenir compte des interactions entre les variables et de gérer les fluctuations liées aux mesures d'importance.

L'intérêt principal de cette recherche sera donc de comparer l'efficacité de deux versions modifiées de l'algorithme `SELECT()` avec la reconstitution de celui-ci. Il y aura donc trois algorithmes : la reconstitution de l'algorithme `SELECT()`, une première version modifiée (`SELECT 2()`) et une seconde version modifiée (`SELECT 3()`). La première modification d'algorithme reposera sur un remplacement de l'analyse en composantes principales par une analyse en composantes principales creuse. La deuxième version modifiée nécessite de remplacer la sélection de variables de la régression logistique (voir l'étape 2 de l'Algorithme 1) par la sélection de variables issue de la librairie `Boruta`. Chacune des modifications est appliquée individuellement de façon à voir si ces modifications sont efficaces. Certes, en combinant plusieurs modifications en même temps, il est difficile de comprendre d'où provient l'amélioration (ou la nuisance) qui survient. Si les modifications sont toutes les deux efficaces, alors il serait intéressant de les combiner. Pour chacune de ces modifications, des nouveaux ensembles de données sont obtenus. Une fois que ce changement de représentation de données est effectué, des modèles de prédiction seront développés sur les nouveaux ensembles de données obtenus. En s'inspirant des classificateurs efficaces de l'ensemble de données utilisé, les algorithmes d'apprentissage choisis seront : une machine à support de vecteurs, une régression logistique pénalisée et une machine à *gradient boosting* (ou *gradient boosting machine*, en anglais). L'ANOVA à blocs complets est utilisée pour regarder l'évolution de l'efficacité de chaque classificateur avec sa méthode d'optimisation. L'intérêt secondaire est d'obtenir un modèle surpassant la plus grande AUC et/ou la plus grande précision des résultats vus dans la littérature (voir Tableau 2.1). Un test d'hypothèses sur la différence de moyennes est préconisé pour vérifier si les résultats de la littérature sont battus.

L'entraînement de classificateurs nécessite de partitionner l'ensemble de données en trois sous-ensembles sans chevauchement : l'ensemble d'entraînement (70% des observations), l'ensemble de validation (15% des observations) et l'ensemble test (15% des observations). Il se

trouve que l'on a priorisé l'application d'une stratification pour que chacun de ces ensembles ait la même proportion d'individus malades. Cela entraîne que l'ensemble d'entraînement est composé de 116 observations, l'ensemble de validation repose sur 24 observations et l'ensemble test est composé de 28 observations...ce qui ne respecte pas tout à fait les pourcentages énoncés en début de paragraphe (70%, 15% et 15%). La validation croisée aurait également pu être appliquée, mais celle-ci est plus compliquée à programmer et l'on dispose d'assez d'observations pour séparer en trois partitions : entraînement, validation et test. L'ensemble d'entraînement sert à entraîner un classificateur doté d'hyperparamètres fixés. Un hyperparamètre est un paramètre de distribution a priori. Or, il se trouve que l'efficacité de l'algorithme (c'est-à-dire à quel point il réussit à bien déterminer les valeurs de la variable de réponse pour des nouvelles observations) dépend des valeurs attribuées aux hyperparamètres. Pour trouver des bonnes estimations d'hyperparamètres, diverses méthodes d'optimisation sont utilisées. Pour s'assurer que la qualité des résultats finaux obtenus n'est pas le fruit du hasard, l'expérience est effectuée à dix reprises selon des partitionnements de données différents.

La page suivante montre les sélections de variables qui sont effectuées dans cette recherche. Chaque étape de chaque algorithme est présentée. L'effet d'embrasement de noir et la boîte en pointe sont utilisés pour mettre l'accent sur l'étape, car celle-ci est une modification innovante de `SELECT()`.


ENSEMBLE D'ENTRAÎNEMENT (70%)
SELECT()
SELECT 2()
SELECT 3()
1 $\mathbf{Q} \leftarrow \texttt{ACP}(\{\mathbf{x}_i\}_{i=1}^n, v)$
1 $\mathbf{Q} \leftarrow \texttt{ACPC}(\{\mathbf{x}_i\}_{i=1}^n, v, k, \boldsymbol{\lambda})$
1 $\mathbf{Q} \leftarrow \texttt{ACP}(\{\mathbf{x}_i\}_{i=1}^n, v)$
2 $\mathbf{Q}_1 \leftarrow \texttt{LOGIT}(\mathbf{Q}, d)$
2 $\mathbf{Q}_1 \leftarrow \texttt{LOGIT}(\mathbf{Q}, d)$
2 $\mathbf{Q}_1 \leftarrow \texttt{Boruta}(\mathbf{Q})$
3 $\{\mathbf{z}_i\}_{i=1}^n \leftarrow \texttt{EnleverAttributs}(\mathbf{Q}_1, \mu)$
3 $\{\mathbf{z}_i\}_{i=1}^n \leftarrow \texttt{EnleverAttributs}(\mathbf{Q}_1, \mu)$
3 $\{\mathbf{z}_i\}_{i=1}^n \leftarrow \texttt{EnleverAttributs}(\mathbf{Q}_1, \mu)$
4 $\mathbf{Q}_2 \leftarrow \texttt{ACP}(\{\mathbf{z}_i\}_{i=1}^n, q_2)$
4 $\mathbf{Q}_2 \leftarrow \texttt{ACP}(\{\mathbf{z}_i\}_{i=1}^n, q_2)$
4 $\mathbf{Q}_2 \leftarrow \texttt{ACP}(\{\mathbf{z}_i\}_{i=1}^n, q_2)$

CHAPITRE 3 RÉVISION MATHÉMATIQUE ET STATISTIQUE PERTINENTE

3.1 Motivation

L'algorithme `SELECT()` comporte plusieurs étapes. Or, afin de pouvoir modifier ces étapes, il est impératif de bien comprendre l'essence des mathématiques derrière chacune de ces étapes afin de pouvoir offrir des modifications intéressantes. En consultant l'Algorithme 1, on remarque qu'il faut bien comprendre l'analyse en composantes principales et la régression logistique pénalisée. Il se trouve que la maîtrise de ces concepts n'est pas suffisante. D'un côté, une compréhension des modifications est également nécessaire. Par conséquent, il faut comprendre : l'analyse en composantes principales creuse (ACPC), des modèles de régressions (*Ridge, lasso* et *elastic net*) et l'algorithme `Boruta`. D'un autre côté, pour chaque classificateur, il faut choisir la meilleure méthode d'optimisation. Pour effectuer des comparaisons valides, la procédure statistique utilisée est l'ANOVA. Plus particulièrement, il s'agit d'un plan à blocs complets. De plus, les critères de performance pour les problèmes de classification seront étudiés.

3.2 Analyse en composantes principales (ACP)

Considérons une matrice de données composée de n lignes et de p colonnes. Un nombre de colonnes supérieur au nombre de lignes rend l'analyse plus difficile sur une multitude de perspectives comme l'interprétation du modèle, la rapidité d'exécution des algorithmes, etc. Il serait donc souhaitable d'employer une méthode réduisant la dimensionnalité de l'ensemble de données (c'est-à-dire qui réduit le nombre de colonnes) tout en conservant le maximum d'information.

L'analyse en composantes principales est une solution à ce problème. L'idée de cette méthode est de changer la représentation des données vers des nouvelles variables. Chacune de ces variables est une combinaison linéaire des variables originales. Le nombre de nouvelles variables q est inférieur au nombre de variables originales ($q < p$). De plus, ces nouvelles variables ne sont pas corrélées entre elles.

3.2.1 Motivation

L'analyse en composantes principales est la première étape de l'algorithme `SELECT()`. Par conséquent, il est impératif de la comprendre afin de bien comprendre l'algorithme. De plus, c'est cette étape de l'algorithme qui sera modifiée prochainement. Comprendre cette étape permet de comprendre l'intuition derrière les modifications qui seront apportées.

3.2.2 Développement théorique

Il a été mentionné que l'on désire effectuer un changement de représentation. Donc, pour la i-ième observation, on a $\mathbf{x}_i \in \mathbb{R}^p$ qui deviendra $\mathbf{z}_i \in \mathbb{R}^q$. L'intérêt est donc de trouver une fonction d'encodage $f : \mathbb{R}^p \leftarrow \mathbb{R}^q$ telle que $f(\mathbf{x}) = \mathbf{c}$. On voudra aussi une fonction de décodage $g : \mathbb{R}^q \leftarrow \mathbb{R}^p$ telle que $\mathbf{x} \approx g(f(\mathbf{c}))$. Sans perdre de généralité, on supposera que les variables ont été centrées sur leur moyenne. Il faut noter que des variables non centrées changeront le problème d'optimisation à une constante près. Or, cette constante disparaît après avoir dérivé le lagrangien. La fonction utilisée sera une application linéaire basée sur la multiplication matricielle. Soit la fonction $g(\mathbf{c}) = \mathbf{D}\mathbf{c}$ où $\mathbf{D} \in \mathbb{R}^{p \times q}$ est la matrice de décodage. Trouver la matrice de décodage optimale est difficile. Par conséquent, pour garder le problème simple, imposons la contrainte que les colonnes de la matrice $\mathbf{D}$ sont orthogonales. Cette condition n'est pas suffisante pour l'obtention d'une solution unique. En multipliant la matrice $\mathbf{D}$ par un nombre réel, la contrainte d'orthogonalité est toujours respectée. On imposera donc que les colonnes de la matrice $\mathbf{D}$ soient unitaires.

Il s'agit d'un problème d'optimisation. On cherche le recodage $g(\mathbf{c}^\star)$ qui est le plus *proche* d'une observation $\mathbf{x}$. Avec la norme euclidienne, ce problème d'optimisation devient :

$$\mathbf{c}^* = \underset{\mathbf{c}}{\operatorname{argmin}} \quad ||\mathbf{x} - g(\mathbf{c})||_2 = \quad \underset{\mathbf{c}}{\operatorname{argmin}} \quad ||\mathbf{x} - g(\mathbf{c})||_2^2 \tag{3.1}$$

On remarque que la norme euclidienne peut être mise au carré, car la norme euclidienne est non-négative et l'opération d'élever au carré est issue d'une fonction monotone croissante pour les arguments positifs. L'expression de droite de l'équation (3.1) peut être développée :

$$(\mathbf{x} - g(\mathbf{c}))^\top (\mathbf{x} - g(\mathbf{c})) = \mathbf{x}^\top \mathbf{x} - 2\mathbf{x}^\top g(\mathbf{c}) + g(\mathbf{c})^\top g(\mathbf{c})$$

Le terme quadratique $\mathbf{x}^\top \mathbf{x}$ ne dépend pas de $\mathbf{c}$ et peut être considéré comme une constante. En utilisant notre définition de $g(\mathbf{c})$, on arrive à :

$$\mathbf{c}^* = \underset{\mathbf{c}}{\operatorname{argmin}} \quad -2\mathbf{x}^\top g(\mathbf{c}) + g(\mathbf{c})^\top g(\mathbf{c}) = \quad \underset{\mathbf{c}}{\operatorname{argmin}} \quad -2\mathbf{x}^\top \mathbf{D}\mathbf{c} + \mathbf{c}^\top \mathbf{D}^\top \mathbf{D}\mathbf{c} \tag{3.2}$$

Puisque les colonnes de $\mathbf{D}$ sont orthogonales, on arrive au problème d'optimisation convexe suivant :

$$\mathbf{c}^* = \underset{\mathbf{c}}{\operatorname{argmin}} \quad -2\mathbf{x}^\top \mathbf{D}\mathbf{c} + \mathbf{c}^\top I_q \mathbf{c} \tag{3.3}$$

La solution est obtenue en dérivant par rapport à $\mathbf{c}$ et en égalisant cette dérivée à zéro :

$$\nabla_{\mathbf{c}}(-2\mathbf{x}^\top \mathbf{D}\mathbf{c} + \mathbf{c}^\top \mathbf{c}) = -2\mathbf{D}^\top \mathbf{x} + 2\mathbf{c} = 0 \Leftrightarrow \mathbf{c} = \mathbf{D}^\top \mathbf{x} \tag{3.4}$$

Il se trouve que la fonction d'encodage est $f(\mathbf{x}) = \mathbf{D}^\top \mathbf{x}$. Il s'ensuit que la fonction de décodage est $r(\mathbf{x}) = g(f(\mathbf{x})) = \mathbf{D}\mathbf{D}^\top \mathbf{x}$. Des contraintes sont imposées sur la matrice d'encodage $\mathbf{D}$, mais elle n'est pas encore définie. Pour ce faire, considérons la fonction de coût entre les valeurs d'entrées $\mathbf{x}$ et leur reconstruction respective $r(\mathbf{x})$. Puisque ce problème traite de toutes les entrées $\mathbf{x}$, la minimisation reposera sur la norme de Frobenius qui est une généralisation de la norme euclidienne appliquée sur les matrices. Le problème sera donc de la forme suivante :

$$\mathbf{D}^* = \underset{\mathbf{D}}{\operatorname{argmin}} \quad \sqrt{\sum_{i,j} \Big(x_{ij} - r(x_{ij}) \Big)^2} \quad \text{s.c. } \mathbf{D}^\top \mathbf{D} = \mathbf{I}_l \tag{3.5}$$

Le terme x_{ij} correspond à la $j^{\text{ième}}$ composante de la $i^{\text{ième}}$ observation dans la matrice des données. De façon similaire, $r(x_{ij})$ est à la reconstruction de la $j^{\text{ième}}$ composante de la $i^{\text{ième}}$ observation. Supposons que l'on souhaite résoudre ce problème, mais en considérant seulement une composante ($l = 1$). Le problème devient :

$$\mathbf{d}^* = \underset{\mathbf{d}}{\operatorname{argmin}} \quad \sum_i ||\mathbf{x}_i - \mathbf{d}\mathbf{d}^\top \mathbf{x}_i||_2^2 \quad \text{s.c. } ||\mathbf{d}||_2 = 1 \tag{3.6}$$

En écrivant ce problème en termes de la matrice des données, on obtient :

$$\mathbf{d}^* = \underset{\mathbf{d}}{\operatorname{argmin}} \quad ||\mathbf{X} - \mathbf{d}\mathbf{d}^\top \mathbf{X}||_F^2 \quad \text{s.c. } ||\mathbf{d}||_2 = 1 \tag{3.7}$$

L'équation (3.7) peut être vue comme une minimisation de la somme des résidus au carré. En utilisant la définition de la norme de Frobenius, l'expression à minimiser se simplifie :

$$||\mathbf{X} - \mathbf{d}\mathbf{d}^\top \mathbf{X}||_F^2 = \operatorname{tr}\Big((\mathbf{X} - \mathbf{d}\mathbf{d}^\top \mathbf{X})^\top (\mathbf{X} - \mathbf{d}\mathbf{d}^\top \mathbf{X}) \Big) = \operatorname{tr}\Big(\mathbf{X}^\top \mathbf{X} - \mathbf{X}^\top \mathbf{d}\mathbf{d}^\top \mathbf{X} \Big)$$

Le terme $\mathbf{X}^\top \mathbf{X}$ ne dépend pas de $\mathbf{d}$ et peut être considéré comme une constante.. Également, la trace est invariante aux permutations de l'ordre des matrices si le produit matriciel est bien défini.

$$\mathbf{d}^\star = \underset{\mathbf{d}}{\operatorname{argmin}} \quad \operatorname{tr}(-\mathbf{X}^\top \mathbf{d}\mathbf{d}^\top \mathbf{X}) \quad \text{s.c. } ||\mathbf{d}||_2 = 1$$

$$\Leftrightarrow \mathbf{d}^\star = \underset{\mathbf{d}}{\text{argmax}} \quad \text{tr}(\mathbf{d}^\top \mathbf{X}\mathbf{X}^\top \mathbf{d}) \quad \text{s.c. } ||\mathbf{d}||_2 = 1$$

Il se trouve que $\mathbf{d}^\top \mathbf{X}\mathbf{X}^\top \mathbf{d}$ est un scalaire. Donc, l'opérateur trace n'est plus nécessaire. Aussi, puisqu'il s'agit d'un scalaire, on peut transposer l'expression. Le problème est donc de la forme :

$$\mathbf{d}^\star = \underset{\mathbf{d}}{\text{argmax}} \quad \mathbf{d}^\top \mathbf{X}^\top \mathbf{X}\mathbf{d} = \underset{\mathbf{d}}{\text{argmax}} \quad ||\mathbf{X}\mathbf{d}||_2^2 \quad \text{s.c. } ||\mathbf{d}||_2 = 1 \tag{3.8}$$

Les colonnes de $\mathbf{X}$ génèrent un sous-espace de $\mathbb{R}^n$ que l'on nommera l'espace des colonnes de $\mathbf{X}$. Le vecteur $\mathbf{d}$ est projeté orthogonalement sur l'hyperplan des variables explicatives. Il est pertinent de noter que ce problème d'optimisation équivaut à trouver le vecteur $\mathbf{d}$ qui maximise la variance. La solution de l'équation est obtenue en résolvant le lagrangien suivant :

$$L(\lambda, \mathbf{X}) = \mathbf{d}^\top \mathbf{X}^\top \mathbf{X}\mathbf{d} - \lambda(\mathbf{d}^\top \mathbf{d} - 1) \tag{3.9}$$

Il s'agit d'une fonction quadratique en $\mathbf{d}$. La solution est obtenue en dérivant l'équation (3.9) par rapport à $\mathbf{d}$:

$$\nabla_{\mathbf{d}}(L(\lambda, \mathbf{X})) = 2\mathbf{X}^\top \mathbf{X}\mathbf{d} - 2\lambda\mathbf{d} = \mathbf{0} \Leftrightarrow \mathbf{X}^\top \mathbf{X}\mathbf{d} = \lambda\mathbf{d} \tag{3.10}$$

Il s'agit d'un problème lié aux valeurs propres. Les solutions à notre problème d'optimisation sont l'ensemble des vecteurs propres de la matrice $\mathbf{X}^\top \mathbf{X}$. Il faut choisir un vecteur propre lié à sa valeur propre respective. Le choix approprié correspond au vecteur propre ayant la plus grande valeur propre.

$$\mathbf{d}^\star = \underset{\mathbf{d}}{\text{argmax}} \quad \mathbf{d}^\top \mathbf{X}^\top \mathbf{X}\mathbf{d} = \underset{\mathbf{d}}{\text{argmax}} \quad \mathbf{d}^\top \lambda\mathbf{d} \quad \text{s.c. } \mathbf{d}^\top \mathbf{d} = 1 \Leftrightarrow \mathbf{d}^\star = \underset{\mathbf{d}}{\text{argmax}} \quad \lambda \tag{3.11}$$

Puisque la matrice $\mathbf{X}^\top \mathbf{X}$ est symétrique, les vecteurs propres de cette matrice sont orthogonaux et les valeurs propres sont supérieures ou égales à zéro. Ce résultat est valable pour $l = 1$ composante. En généralisant, on se rend compte que la matrice $\mathbf{D}$ correspond aux l vecteurs propres unitaires associés aux plus hautes valeurs propres de $\mathbf{X}^\top \mathbf{X}$. Le lecteur curieux peut prouver ce résultat par une preuve par induction.

3.2.3 Propriétés importantes et limitations

L'analyse en composantes principales a plusieurs propriétés intéressantes. En voici quelques-unes :

— La variance de chaque composante est égale à sa valeur propre. Par conséquent, la variance expliquée par la i-ième composante est donnée par $\lambda_i / \sum_j \lambda_j$.

— La somme de la variance des composantes est égale à la somme de la variance de chacune des variables originales.

Cependant, cette méthode n'est pas à l'abri de certaines limitations. On a entre autres que :

— La direction avec la plus grande variance est utilisée en supposant qu'elle est la plus intéressante.

— Cette méthode considère seulement les transformations orthogonales (rotation) des variables originales.

— L'analyse en composantes principales n'est pas invariante aux changements d'échelle et est sensible aux données aberrantes.

— L'analyste doit choisir le nombre de composantes à sélectionner.

3.2.4 Choisir le nombre de composantes

Dans le cadre de la réduction de dimensionnalité, il faudra choisir un sous-ensemble de m composantes ($m < p$). Ce choix est normalement basé sur la magnitude des valeurs propres. Les composantes avec une valeur propre grande sont gardées tandis que les autres sont négligées. Cependant, le seuil de décision pour garder une composante est subjectif. Voici quelques règles utilisées :

— Sélectionner le nombre minimal de composantes faisant en sorte que la proportion totale de la variance dépasse un certain seuil (Mardia et al., 1979).

— Utiliser un graphique en éboulis (ou *screeplot*, en anglais) (Mardia et al., 1979).

— Exclure les composantes dont la valeur propre est inférieure à la moyenne (Kaiser, 1960).

— L'utilisation du test d'isotropie (Mardia et al., 1979).

3.2.5 Lien entre l'analyse en composantes principales et la décomposition en valeurs singulières

La décomposition en valeurs singulières (ou SVD, en anglais) est une méthode pour factoriser des matrices, et ce, peu importe ses dimensions.

Théorème 1 *Soit* $\mathbf{X}$ *une matrice* $n \times p$ *de rang* r*. Alors, il existe une matrice* $\mathbf{D}$ *de dimensions* $n \times p$ *telles que les* r *premiers éléments sur sa diagonale correspond aux valeurs singulières de* $\mathbf{X}$*,*$d_1 \geq d_2 \geq ... \geq d_r > 0$*, et il existe une matrice orthogonale* $\mathbf{U}$ *de dimensions* $n \times n$ *ainsi qu'une matrice orthogonale* $\mathbf{V}$ *de dimensions* $p \times p$ *telle que :*

$$\mathbf{X} = \mathbf{UDV}^\top$$

Preuve 1 *La preuve peut être trouvée dans divers ouvrages d'algèbre linéaire.*

Les colonnes de la matrice $\mathbf{V}$ sont appelées directions (ou poids) des composantes principales. Les colonnes de la matrice $\mathbf{U}$ sont nommées les composantes principales normalisées (ou scores des composantes principales). Si les colonnes de la matrice $\mathbf{X}$ sont centrées, la matrice de covariance est $\mathbf{S} = {}^{\mathbf{X}^\top \mathbf{X}}/_{n-1}$ où n est le nombre d'observations. Il se trouve que la variance totale correspond à la trace de $\mathbf{S}$. En utilisant la décomposition en valeurs singulières, on obtient :

$$\mathrm{tr}(\mathbf{X}^\top\mathbf{X}) = \mathrm{tr}((\mathbf{U}\mathbf{D}\mathbf{V}^\top)^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top) = \mathrm{tr}(\mathbf{V}\mathbf{D}^\top\mathbf{U}^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top) = \mathrm{tr}(\mathbf{V}\mathbf{D}^2\mathbf{V}^\top) = \mathrm{tr}(\mathbf{D}^2) \quad (3.12)$$

Donc, la variance totale correspond à la somme des valeurs singulières de la matrice $\mathbf{X}^\top\mathbf{X}$. Ainsi, la $j^{\text{ième}}$ composante principale à une contribution $p_j = {}^{d_j^2}/_{\mathrm{tr}(\mathbf{X}^\top\mathbf{X})}$ dans la variation totale des données. Mais il se trouve que les valeurs propres de $\mathbf{X}^\top\mathbf{X}$ correspondent aux valeurs singulières élevées au carré. En effet, on obtient par des simplifications analogues à l'équation (3.12) :

$$\mathbf{X}^\top\mathbf{X}\mathbf{d} = (\mathbf{V}\mathbf{D}^2\mathbf{V}^\top)\mathbf{d} = \lambda\mathbf{d} \quad (3.13)$$

Pour l'équation (3.13), il s'avère que la décomposition de $\mathbf{X}$ est liée à la diagonalisation de $\mathbf{X}^\top\mathbf{X}$. Certes, on peut s'apercevoir que les valeurs propres de $\mathbf{X}^\top\mathbf{X}$ correspondent aux valeurs singulières de $\mathbf{X}$ élevées au carré.

3.2.6 Exemples

Considérons un problème de classification binaire où l'on dispose de deux variables explicatives et d'une variable de réponse. Pour ce problème, tracer un nuage de points et colorer les points selon la classe est une bonne initiative. En effet, il est possible qu'une séparation linéaire entre les deux classes soit visible. Une règle de classification simple peut alors être établie. En réalité, il est rare que l'on dispose de seulement deux variables explicatives. Fréquemment, il y en aura des centaines et même des milliers. Une solution permettant de réduire la dimensionnalité consiste à utiliser l'ACP. On trace alors un nuage de points selon les deux ou trois composantes expliquant le mieux la variance.

Un ensemble de données connu est celui des chiffres de codes postaux scannés des enveloppes du service de poste américain (*U.S Postal Service*, en anglais). Écrits manuellement, ces chiffres étaient originalement de différentes tailles et orientations, mais ces images ont été normalisées (Le Cun et al., 1990). Le résultat est un ensemble d'images en noir et blanc de format 16 pixels par 16 pixels (256 pixels par image). En évaluant l'intensité de la cou-

leur de chaque pixel, chaque image de chiffre peut être représentée par un vecteur de 256 composantes.

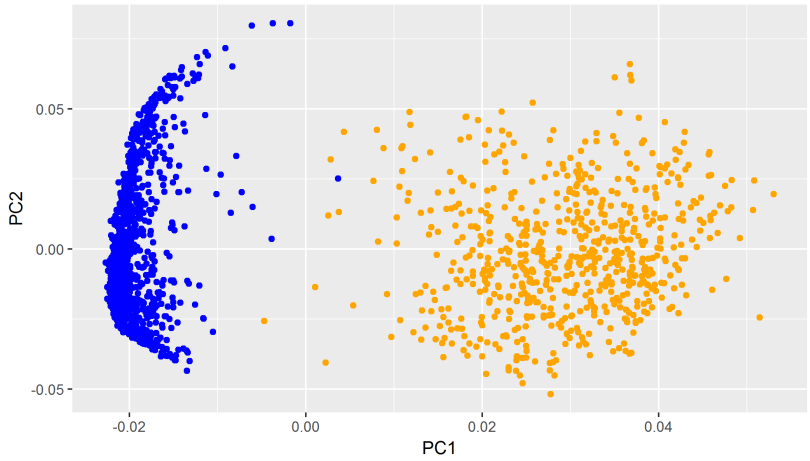


Figure 3.1 Nuage de points des deux premières composantes de l'ACP pour les chiffres 1 et 3 des codes postaux (zip.train, librairie `ElemStatLearn` de R). Les points en bleue représentent les observations qui sont le chiffre 1. Les points en orange correspondent au chiffre 3. La première composante explique 43.3% de la variance. La seconde composante explique 8.2% de la variance totale.

La séparation pour l'exemple des chiffres est très apparente (voir Figure 3.1). Il est presque possible de séparer ces points en deux classes en y faisant passer une droite.

Portons maintenant attention à l'ensemble de données de Gravier (2010) qui sera analysé dans ce mémoire. Malgré l'analyse en composantes principales, il ne semble pas possible de séparer les deux groupes dans ce nuage de points (voir Figure 3.2).

3.3 Régression linéaire multiple, de *Ridge*, *lasso* et *elastic net*

La régression linéaire multiple permet de modéliser la variable de réponse (un scalaire) comme une combinaison linéaire (selon les poids $\boldsymbol{\beta}$) des variables explicatives en résolvant un problème d'optimisation des moindres carrés ordinaires (MCO). Par exemple, on pourrait vouloir prédire le prix d'une maison selon des variables explicatives comme : le nombre d'étages, l'aire du terrain extérieur, le quartier, etc. Pour sélectionner des variables, une nouvelle procédure doit être appliquée après le calcul de $\hat{\boldsymbol{\beta}}_{MCO}$.

Malheureusement, la solution obtenue comporte certains inconvénients. Premièrement, si la matrice $\mathbf{X}^\top\mathbf{X}$ n'est pas de plein rang, alors la matrice est mal conditionnée et calculer son inverse est une procédure numériquement instable. Aussi, cela entraîne que les coefficients de

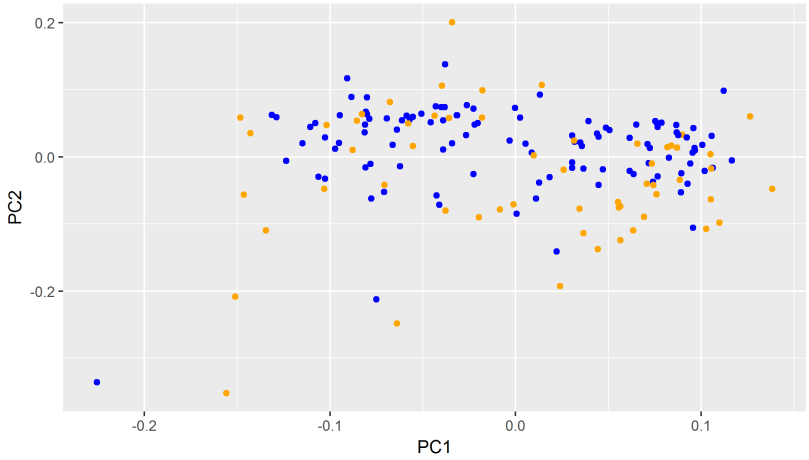


Figure 3.2 ACP pour l'ensemble de données de Gravier (2010). La première composante explique 9.58% de la variance et la seconde composante explique 7.08% de la variance.

$\hat{\boldsymbol{\beta}}_{MCO}$ sont associés à une haute variance. Finalement, il pourrait y avoir l'apparition d'un problème de multicolinéarité. Cela arrive lorsqu'au moins une variable peut être exprimée comme une combinaison linéaire des autres variables.

Une solution consiste à introduire une pénalité à la fonction de coût. Parmi les pénalités populaires, il y a celles de la régression de *Ridge*, de *lasso* et de l'*elastic net*. Le choix de certaines pénalités peut mener à une sélection de variables et permet ainsi l'obtention d'un modèle plus parcimonieux. Il s'avère que l'analyse en composantes principales peut être traitée dans un contexte de régression (Zou et al., 2004). Par conséquent, les méthodes de pénalisation nécessaires à la compréhension de l'analyse en composantes principales creuse (ou *sparse principal components analysis*, en anglais) seront explorées.

3.3.1 Motivation

Il existe un lien qui unissant l'analyse en composantes principales (ACP) et l'analyse en composantes principales creuse (ACPC). Le changement de l'ACP vers l'ACPC donnera naissance à la première version modifiée de l'algorithme `SELECT()`, que l'on nommera `SELECT 2()`. Or, la clé de l'ACPC repose dans l'utilisation de fonctions de coût pénalisées. Il se trouve que la régression de *Ridge*, la régression *lasso* et l'*elastic net* reposent sur des fonctions de coût pénalisées. Par conséquent, comprendre ces régressions permet de bien comprendre la théorie de l'ACPC.

3.3.2 Développement théorique

Considérons la matrice d'expérimentation $\mathbf{X}$ composée de n observations et de p variables. Considérons $\mathbf{y} = (y_1, y_2, ..., y_n)^\top$ le vecteur de la variable de réponse et $\mathbf{X}_j = (x_{1j}, ..., x_{nj})^\top, j = 1, ..., p$ les prédicteurs. Il est possible de centrer les prédicteurs $\mathbf{X}_j$ et le vecteur $\mathbf{y}$. La régression linéaire multiple est de la forme $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$ où $\boldsymbol{\beta}$ est le vecteur des poids obtenu en minimisant la fonction de coût.

$$\hat{\boldsymbol{\beta}}_{MCO} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{X}_j \beta_j \right\|_2^2 = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{3.14}$$

Cette solution correspond à la résolution d'un problème d'optimisation des moindres carrés ordinaires. La régression de *Ridge* consiste à utiliser la même fonction de coût que la régression linéaire multiple, mais en ajoutant un terme de pénalité. La pénalité est appliquée sur la norme euclidienne de $\boldsymbol{\beta}$.

$$\hat{\boldsymbol{\beta}}_{ridge} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{X}_j \beta_j \right\|_2^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \tag{3.15}$$

L'ajout d'un terme de pénalité correspond à ajouter une contrainte au problème d'optimisation en (voir Figure 3.3). La régression de *Ridge* réduit la taille des coefficients et n'est pas limitée par le fait que $p > n$. Malheureusement, ceux-ci n'atteignent pas zéro et la sélection de variables n'est pas effectuée. Si la sélection de variables doit être effectuée, une alternative intéressante consiste à utiliser la régression *lasso*.

De façon similaire, la régression *lasso* consiste à ajouter un terme de pénalité à la fonction de coût du problème de régression. La pénalité est appliquée sur la norme ℓ_1 de $\boldsymbol{\beta}$.

$$\hat{\boldsymbol{\beta}}_{lasso} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\| \mathbf{y} - \sum_{i=1}^{p} \mathbf{X}_j \beta_j \right\|_2^2 + \lambda \sum_{i=1}^{p} |\beta_i| \tag{3.16}$$

Malheureusement, la méthode *lasso* comporte plusieurs inconvénients. Notamment, la solution est obtenue numériquement. Dans la situation présentée, l'inconvénient le plus désagréable est le fait que le nombre de variables que l'on peut sélectionner est borné par le nombre d'observations. Étant donné que le jeu de données utilisé contient peu d'observations, une alternative doit être trouvée.

L'alternative utilisée est l'*elastic net*. Cette méthode est un mélange de la régression de *Ridge* et de *lasso*. Sa pénalité est une combinaison convexe de la pénalité de *Ridge* et de la pénalité

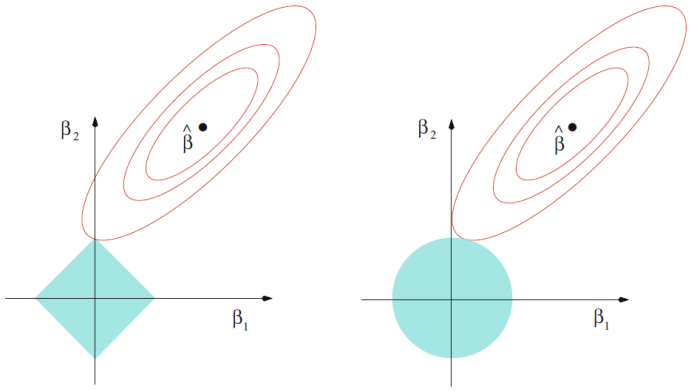


Figure 3.3 Estimations de *lasso* (gauche) et de la régression de *Ridge* (droite). Les courbes de niveau sont associées à la fonction de coût de la régression linéaire multiple et les régions turquoises correspondent aux pénalités. (*Elements of Statistical Learning*, p.71, Figure 3.11)

lasso. Pour tout λ_1 et λ_2 non négatifs, l'estimateur de l'*elastic net* est :

$$\hat{\boldsymbol{\beta}}_{en} = (1+\lambda_2)\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left| \mathbf{y} - \sum_{j=1}^{p} \mathbf{X}_j \beta_j \right|^2 + \lambda_2 \sum_{j=1}^{p} \beta_j^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| \tag{3.17}$$

Pour un λ_2 fixé, l'algorithme LARSEN (Zhou & Hastie, 2003) résout efficacement l'*elastic net* pour tout λ_1 avec le coût de calcul associé à la régression linéaire multiple. Puisque notre situation est telle que $p > n$, alors il faudra choisir $\lambda_2 > 0$. En procédant ainsi, l'*elastic net* peut inclure toutes les variables dans un modèle. Un avantage offert par cette méthode est l'effet de groupe. Plus précisément, il est probable qu'un groupe de variables fortement corrélées soit formé une fois qu'une variable de ce groupe est sélectionnée.

Par exemple, le Tableau 3.1 montre les résultats de l'application des régressions précédentes sur une étude (1989) examinant le lien entre un antigène présent dans la prostate (ou PSA, en anglais) et différentes mesures médicales pour des hommes sur le point de recevoir une prostatectomie radicale. Il s'avère que les coefficients égaux à zéro proviennent de la régression *lasso*.

3.4 Analyse en composantes principales creuse (ACPC)

3.4.1 Motivation

L'analyse en composantes principales consiste à diminuer la dimensionnalité en changeant la représentation des données. Or, les composantes principales sont une combinaison linéaire

Tableau 3.1 Coefficients obtenus à partir des différentes méthodes de régression (avec $\lambda_{lasso} = 0.0325$ et $\lambda_{ridge} = 6.8$). Le point symbolise un coefficient égale à zéro dans le modèle.

Régression	**Coefficients**							
	lcavol	**lweight**	**age**	**lbph**	**svi**	**lcp**	**gleason**	**pgg45**
Linéaire multiple	0.5643	0.6220	-0.0212	0.0967	0.7617	-0.1061	0.0492	0.0045
Ridge	0.4949	0.6050	-0.0169	0.0863	0.6885	-0.0420	0.0634	0.0034
lasso	0.5057	0.5387	-0.0074	0.0585	0.5855	.	.	0.0022

de toutes les variables [1]. L'interprétation est alors plus difficile et il serait souhaitable d'avoir des combinaisons linéaires composées de moins de variables. Dans cette optique, l'analyse en composantes principales peut être vue comme un problème de régression et une sélection de variables peut être effectuée pour la construction de chacune des composantes principales.

3.4.2 Développement théorique

Il a été prouvé précédemment que les composantes principales étaient les vecteurs propres de la matrice $\mathbf{X}^\top\mathbf{X}$. Par le Théorème 1, on a donc :

$$\mathbf{X}\mathbf{v}_1 = \mathbf{U}\mathbf{D}\mathbf{V}^\top\mathbf{v}_1 = d_1\mathbf{u}_1$$

Par conséquent, les vecteurs $\mathbf{v}_j$ (pour $j = 1, 2, ..., p$) peuvent être obtenus en effectuant une régression sur les composantes principales :

Théorème 2 *Pour tout $i \in \{1, 2, ..., p\}$, notons $\mathbf{Y}_i = \mathbf{U}_i D_i$ comme étant la i-ième composante principale. Il se trouve que $\forall\lambda > 0$, en supposant que $\hat{\boldsymbol{\beta}}_{ridge}$ est l'estimateur de la régression de* Ridge *donné par :*

$$\hat{\boldsymbol{\beta}}_{ridge} = \underset{\boldsymbol{\beta}}{argmin}\ |\mathbf{Y}_i - \mathbf{X}\boldsymbol{\beta}|^2 + \lambda\sum_{i=1}^{p}\beta_i^2 \tag{3.18}$$

Si $\hat{\mathbf{v}} = \frac{\hat{\boldsymbol{\beta}}_{ridge}}{|\hat{\boldsymbol{\beta}}_{ridge}|}$, on a que $\hat{v} = \mathbf{V}_i$.

1. $\mathbf{q}_i = \sum_{i=1}^{p}\alpha_i\mathbf{x}_i \quad \forall i \in \{1, 2, ..., p\}$

Preuve 2 *On sait que* $\mathbf{X}^\top\mathbf{X} = \mathbf{V}\mathbf{D}^2\mathbf{V}^\top$ *et* $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$. *On a donc que :*

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{ridge} &= (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}_i = (\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})\mathbf{V}^\top)^{-1}\mathbf{V}\mathbf{D}^2\mathbf{V}^\top\mathbf{V}_i \\ &= \mathbf{V}\left(\frac{\mathbf{D}^2}{\mathbf{D}^2 + \lambda\mathbf{I}}\right)\mathbf{V}^\top\mathbf{V}_i = \mathbf{V}_i\frac{\mathbf{D}_i^2}{\mathbf{D}_i^2 + \lambda\mathbf{I}}\end{aligned} \tag{3.19}$$

Maintenant, si l'on calcule $\hat{\mathbf{v}} = \frac{\hat{\boldsymbol{\beta}}_{ridge}}{|\hat{\boldsymbol{\beta}}_{ridge}|}$, *on obtient :*

$$\hat{\mathbf{v}} = \frac{\hat{\boldsymbol{\beta}}_{ridge}}{|\hat{\boldsymbol{\beta}}_{ridge}|} = \mathbf{V}_i\frac{\mathbf{D}_i^2}{\mathbf{D}_i^2 + \lambda\mathbf{I}} \cdot \frac{1}{|\mathbf{V}_i|}\frac{\mathbf{D}_i^2 + \lambda\mathbf{I}}{\mathbf{D}_i^2} = \frac{\mathbf{V}_i}{|\mathbf{V}_i|} = \mathbf{V}_i \quad \blacksquare$$

Il est intéressant de noter que $\hat{\mathbf{v}}$ ne dépend pas de λ et il n'y a donc pas d'effet de rétrécissement attribué au paramètre λ. Il se trouve que notre estimateur $\hat{\mathbf{v}}$ procure la valeur exacte de $\mathbf{V}_i$. Ajoutons un terme de pénalité ℓ_1 à l'expression (3.18) :

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; |\mathbf{Y}_i - \mathbf{X}\boldsymbol{\beta}|^2 + \lambda\sum_{i=1}^{p}\beta_i^2 + \lambda_1\sum_{i=1}^{p}|\beta_i| \tag{3.20}$$

Ce problème est presque de la même forme que le problème d'optimisation de l'*elastic net*. Il n'existe donc pas de solution explicite et une approximation est obtenue numériquement. On appelle $\hat{\mathbf{V}}_i$ l'approximation de $\mathbf{V}_i$ et $\mathbf{X}\hat{\mathbf{V}}_i$ l'approximation de la i-ième composante. L'expression (3.20) correspond à l'*elastic net* naïf (Zhou et Hastie, 2003) qui diffère de l'expression (3.17) par la facteur de changement d'échelle $(1+\lambda_2)$. Ce facteur est sans importance ici car les coefficients sont normalisés. Pour λ_1 suffisamment grand, $\hat{\boldsymbol{\beta}}$ est creux et par conséquent $\hat{\mathbf{V}}_i$ aussi.

Considérons $\boldsymbol{\alpha}, \boldsymbol{\beta}$ des matrices de format $p \times k$. Il a été vu précédemment (voir section 3.2) que l'optimisation de $\sum_{i=1}^{n}|\mathbf{X}_i - \boldsymbol{\alpha}\boldsymbol{\beta}^\top\mathbf{X}_i|^2$ (sous la contrainte d'orthogonalité et lorsque $\boldsymbol{\alpha} = \boldsymbol{\beta}$) correspond au problème d'optimisation de l'analyse en composantes principales. Or, il se trouve que l'on peut relaxer la contrainte $\boldsymbol{\alpha} = \boldsymbol{\beta}$ et ajouter le terme de pénalité de la régression de *Ridge* pour aboutir à la solution de l'ACP. En effet :

Théorème 3 *Soit $\mathbf{X}_i$, la i-ième rangée de la matrice $\mathbf{X}$ et $\boldsymbol{\alpha}, \boldsymbol{\beta}$ des matrices de dimension $p \times k$. Considérons les k premières composantes principales.Pour tout $\lambda > 0$, considérons :*

$$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = \underset{\boldsymbol{\beta}}{argmin} \sum_{i=1}^{n} |\mathbf{X}_i - \boldsymbol{\alpha}\boldsymbol{\beta}^\top \mathbf{X}_i|^2 + \lambda \sum_{j=1}^{k} |\boldsymbol{\beta}_j|^2 \quad s.c\ \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = \mathbf{I}_k \tag{3.21}$$

Alors, $\hat{\boldsymbol{\beta}}_i \propto \mathbf{V}_i$ pour $i = 1, 2, ..., k$.

Preuve 3 *Il se trouve que :*

$$\begin{aligned}
\sum_{i=1}^{n} |\mathbf{X}_i - \boldsymbol{\alpha}\boldsymbol{\beta}^\top \mathbf{X}_i|^2 &= \sum_{i=1}^{n} tr(\mathbf{X}_i^\top (\mathbf{I} - \boldsymbol{\beta}\boldsymbol{\alpha}^\top)(\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\beta}^\top)\mathbf{X}_i) \\
&= \sum_{i=1}^{n} tr((\mathbf{I} - \boldsymbol{\beta}\boldsymbol{\alpha}^\top)(\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\beta}^\top)\mathbf{X}_i\mathbf{X}_i^\top) \\
&= tr\Big((\mathbf{I} - \boldsymbol{\beta}\boldsymbol{\alpha}^\top)(\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\beta}^\top) \sum_{i=1}^{n} \mathbf{X}_i\mathbf{X}_i^\top\Big) \\
&= tr\Big((\mathbf{I} - \boldsymbol{\beta}\boldsymbol{\alpha}^\top)(\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\beta}^\top)\mathbf{X}^\top\mathbf{X}\Big) \\
&= tr\Big((\mathbf{I} - \boldsymbol{\beta}\boldsymbol{\alpha}^\top - \boldsymbol{\alpha}\boldsymbol{\beta}^\top + \boldsymbol{\beta}\mathbf{I}_k\boldsymbol{\beta}^\top)\mathbf{X}^\top\mathbf{X}\Big) \\
&= tr(\mathbf{X}^\top\mathbf{X}) - 2tr(\boldsymbol{\alpha}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}) + tr(\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta})
\end{aligned} \tag{3.22}$$

En ajoutant le terme de pénalité $\lambda \sum_{j=1}^{k} \beta_j^2$, on obtient :

$$\begin{aligned}
\sum_{i=1}^{n} |\mathbf{X}_i - \boldsymbol{\alpha}\boldsymbol{\beta}^\top \mathbf{X}_i|^2 + \lambda \sum_{j=1}^{k} \beta_j^2 &= tr(\mathbf{X}^\top\mathbf{X}) - 2tr(\boldsymbol{\alpha}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}) + tr(\boldsymbol{\beta}^\top(\mathbf{X}^\top\mathbf{X} + \lambda)\boldsymbol{\beta}) \\
&= tr(\mathbf{X}^\top\mathbf{X}) + \sum_{j=1}^{k} \Big(\boldsymbol{\beta}_j^\top(\mathbf{X}^\top\mathbf{X} + \lambda)\boldsymbol{\beta}_j - 2\boldsymbol{\alpha}_j^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}_j\Big)
\end{aligned} \tag{3.23}$$

Pour un $\boldsymbol{\alpha}_j$ fixé, la quantité est minimisée avec $\boldsymbol{\beta}_j = (\mathbf{X}^\top\mathbf{X} + \lambda)^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\alpha}_j$ pour tout $j \in \{1, 2, ..., k\}$. De façon équivalente, cela revient à avoir $\boldsymbol{\beta} = (\mathbf{X}^\top\mathbf{X} + \lambda)^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\alpha}$. En substituant cela dans l'équation matricielle (3.23), on obtient le problème d'optimisation suivant :

$$\begin{aligned}
\hat{\boldsymbol{\alpha}} &= \underset{\boldsymbol{\alpha}}{argmax}\ tr(\boldsymbol{\alpha}^\top\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda)^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\alpha}) \quad s.c\ \boldsymbol{\alpha}^\top\boldsymbol{\alpha} = \mathbf{I}_k \\
&= \underset{\boldsymbol{\alpha}}{argmax}\ tr\Big(\boldsymbol{\alpha}^\top\mathbf{V}\frac{\mathbf{D}^4}{\mathbf{D}^2 + \lambda}\mathbf{V}^\top\boldsymbol{\alpha}\Big) \quad s.c\ \boldsymbol{\alpha}^\top\boldsymbol{\alpha} = \mathbf{I}_k
\end{aligned} \tag{3.24}$$

Si l'on considère ce problème pour un $\boldsymbol{\alpha}_j$, on peut résoudre le problème en dérivant le lagran-

gien. La solution est $\hat{\boldsymbol{\alpha}}_j = s_j \mathbf{V}_j$ *avec* $s_j = 1$ *ou* -1 *pour tout* $j \in \{1, 2, ..., k\}$. *En substituant* $\hat{\boldsymbol{\alpha}}$ *dans* $\boldsymbol{\beta}_j$, *on obtient* $\hat{\boldsymbol{\beta}}_j = s_j \frac{\mathbf{D}_j^2}{\mathbf{D}_j^2 + \lambda} \mathbf{V}_j$ *pour tout* $j \in \{1, 2, ..., k\}$. ■

Il s'avère donc que l'on est en mesure de résoudre le problème d'ACP en le ramenant à un problème de régression. Cependant, on souhaiterait que nos vecteurs correspondant aux composantes principales soient composés de zéros comme pour la sélection de variables dans *lasso*. Pour ce faire, il faut ajouter une pénalité selon la norme ℓ_1 sur $\boldsymbol{\beta}$ au problème d'optimisation (3.21) :

$$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} |\mathbf{X}_i - \boldsymbol{\alpha}\boldsymbol{\beta}^\top \mathbf{X}_i|^2 + \lambda \sum_{j=1}^{k} |\boldsymbol{\beta}_j|^2 + \sum_{j=1}^{k} \lambda_{1,j} |\boldsymbol{\beta}_j|_1 \quad \text{s.c } \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = \mathbf{I}_k \tag{3.25}$$

Il est important de noter que le paramètre λ est le même pour toutes les composantes principales. Les paramètres $\lambda_{1,j}$ servent à pénaliser les différentes composantes principales. Dans notre situation, puisque $p > n$ alors il faut que $\lambda > 0$ (si $\lambda_{1,j} = 0$).

3.4.3 Lorsque $p \gg n$

Il se trouve que la fonction de coût exprimée par l'équation (3.25) peut être transformée de la suivante :

$$\operatorname{tr}(\mathbf{X}^\top \mathbf{X}) - 2\operatorname{tr}(\boldsymbol{\alpha}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta}) + \operatorname{tr}(\boldsymbol{\beta}^\top (\mathbf{X}^\top \mathbf{X} + \lambda)\boldsymbol{\beta}) + \sum_{j=1}^{k} \lambda_{1,j} |\boldsymbol{\beta}_j|_1 \tag{3.26}$$

Si $\boldsymbol{\beta}$ est fixé, il faudra maximiser $\operatorname{tr}(\boldsymbol{\alpha}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta})$ sous la contrainte $\boldsymbol{\alpha}^\top \boldsymbol{\alpha} = \mathbf{I}_k$. La solution est donnée par le théorème suivant :

Théorème 4 *Soient* $\boldsymbol{\alpha}$ *et* $\boldsymbol{\beta}$ *des matrices de dimensions* $m \times k$ *où* $\boldsymbol{\beta}$ *est de rang* k. *Considérons le problème d'optimisation sous contrainte suivant :*

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\textit{argmin}}\ \textit{tr}(\boldsymbol{\alpha}^\top \boldsymbol{\beta}) \quad \textit{s.c } \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = \mathbf{I}_k \tag{3.27}$$

Supposons que $\boldsymbol{\beta}$ *admet une décomposition SVD, c'est-à-dire* $\boldsymbol{\beta} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$, *alors* $\hat{\boldsymbol{\alpha}} = \mathbf{U}\mathbf{V}^\top$.

Preuve 4 *La contrainte* $\boldsymbol{\alpha}^\top \boldsymbol{\alpha} = \mathbf{I}_k$ *peut être exprimée de façon équivalente en* $k(k+1)/2$ *contraintes :*

$$\boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_i = 1 \quad \forall i \in \{1, 2, ..., k\} \tag{3.28}$$

$$\boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j = 0 \quad j > i \tag{3.29}$$

Le lagrangien du problème est donc :

$$L = -\sum_{i=1}^{k} \boldsymbol{\beta}_i^\top \boldsymbol{\alpha}_i + \sum_{i=1}^{k} \left(\frac{1}{2} \lambda_{i,i} (\boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_i - 1) + \sum_{j>i} \lambda_{i,j} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j \right) \tag{3.30}$$

En posant $\frac{\partial L}{\partial \boldsymbol{\alpha}_i}$*, on obtient que* $\boldsymbol{\beta}_i = \lambda_{i,i} \hat{\boldsymbol{\alpha}}_i + \sum_{j=l+1}^{k} \lambda_{ij} \boldsymbol{\alpha}_j$ *ou sous la forme matricielle* $\boldsymbol{\beta} = \hat{\boldsymbol{\alpha}} \boldsymbol{\Lambda}$ *avec* $\boldsymbol{\Lambda}_{i,j} = \lambda_{i,j}$*. Les matrices* $\boldsymbol{\alpha}$ *et* $\boldsymbol{\beta}$ *sont de plein rang. Par conséquent,* $\boldsymbol{\Lambda}$ *est inversible et* $\boldsymbol{\alpha} = \boldsymbol{\beta} \boldsymbol{\Lambda}^{-1}$*. En effet, puisque* $\boldsymbol{\Lambda}$ *est une matrice carrée, elle est inversible si et seulement si elle est surjective, ce qui dans notre cas veut dire que son rang est égal à* k*. Supposons au contraire que son rang soit inférieur à* k*. Son image est donc engendrée par des vecteurs* $v_1, ..., v_r$ *où* $r < k$*. Mais alors l'image de* $\boldsymbol{\beta} = \boldsymbol{\alpha} \boldsymbol{\Lambda}$ *est engendrée par* $\boldsymbol{\alpha} \mathbf{v}_1, ..., \boldsymbol{\alpha} \mathbf{v}_r$*, ce qui implique que le rang de* $\boldsymbol{\beta}$ *est au plus* r*. Ceci est une contradiction puisque le rang de* $\boldsymbol{\beta}$ *est égal à* k *(par hypothèse). Le système matriciel est alors de la forme suivante où* $\boldsymbol{\Lambda}$ *est triangulairement inférieure :*

$$\begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1k} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{m1} & \beta_{m2} & \dots & \beta_{mk} \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1k} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m1} & \alpha_{m2} & \dots & \alpha_{mk} \end{bmatrix} \begin{bmatrix} \lambda_{11} & 0 & \dots & 0 \\ \lambda_{21} & \lambda_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{k1} & \lambda_{k2} & \dots & \lambda_{kk} \end{bmatrix} \tag{3.31}$$

En utilisant cette représentation, les manipulations algébriques sont simplifiées. On a :

$$tr(\hat{\boldsymbol{\alpha}} \boldsymbol{\beta}) = tr(\boldsymbol{\Lambda}^{-1,\top} \boldsymbol{\beta}^\top \boldsymbol{\beta}) = tr(\boldsymbol{\Lambda}^{-1,\top} \mathbf{V} \mathbf{D}^2 \mathbf{V}^\top) \tag{3.32}$$

$$\mathbf{I}_k = \hat{\boldsymbol{\alpha}}^\top \hat{\boldsymbol{\alpha}} = \boldsymbol{\Lambda}^{-1,\top} \boldsymbol{\beta}^\top \boldsymbol{\beta} \boldsymbol{\Lambda}^{-1} = \boldsymbol{\Lambda}^{-1,\top} \mathbf{V} \mathbf{D}^2 \mathbf{V}^\top \boldsymbol{\Lambda}^{-1} \tag{3.33}$$

Posons $\mathbf{A} = \mathbf{V}^\top \boldsymbol{\Lambda}^{-1} \mathbf{V}$*, alors il se trouve que :*

$$tr(\boldsymbol{\Lambda}^{-1,\top} \mathbf{V} \mathbf{D}^2 \mathbf{V}^\top) = tr(\mathbf{V}^\top \boldsymbol{\Lambda}^{-1,\top} \mathbf{V} \mathbf{D}^2) = tr(\mathbf{A}^\top \mathbf{D}^2) = \sum_{j=1}^{k} A_{jj} D_{jj}^2 \tag{3.34}$$

$$\mathbf{A}^\top \mathbf{D}^2 \mathbf{A} = \mathbf{V}^\top \boldsymbol{\Lambda}^{-1,\top} \mathbf{V} \mathbf{D}^2 \mathbf{V}^\top \boldsymbol{\Lambda}^{-1} \mathbf{V} = \mathbf{V}^\top \hat{\boldsymbol{\alpha}}^\top \hat{\boldsymbol{\alpha}} \mathbf{V} = \mathbf{I}_k \tag{3.35}$$

Si l'on explicite les éléments sur la diagonale de la matrice $\mathbf{A}^\top \mathbf{D}^2 \mathbf{A}$*, on se rend compte que* $[\mathbf{A}^\top \mathbf{D}^2 \mathbf{A}]_{ll} = \sum_{j=1}^{k} A_{jl}^2 D_{jj}^2 = 1 \quad \forall i \in \{1, 2, ..., k\}$*. Donc, en particulier, on a que* $A_{ll}^2 D_{ll}^2 \leq 1$ *pour tout* $l \in \{1, 2, ..., k\}$*. On remarque que* D_{jj} *correspond aux valeurs singulières de* $\boldsymbol{\beta}$ *qui sont positives. Par conséquent, l'inégalité suivante est obtenue en utilisant l'inégalité de*

Cauchy-Swartz :

$$\sum_{j=1}^{k} A_{jj} D_{jj}^2 = \sum_{j=1}^{k} A_{jj} D_{jj}^{3/2} D_{jj}^{1/2} \overset{C.S.}{\leq} \sqrt{\sum_{j=1}^{k} A_{jj}^2 D_{jj}^3 \sum_{j=1}^{k} D_{jj}} \leq \sqrt{\sum_{j=1}^{k} D_{jj} \sum_{j=1}^{k} D_{jj}} = \sum_{j=1}^{k} D_{jj} \quad (3.36)$$

L'égalité est seulement obtenue si $\mathbf{A}$ *est diagonale et* $A_{jj} = D_{jj}^{-1}$ *pour tout* j*. Alors :*

$$\mathbf{\Lambda}^{-1} = \mathbf{V}\mathbf{A}\mathbf{V}^\top = \mathbf{V}\mathbf{D}^{-1}\mathbf{V}^\top \Rightarrow \hat{\boldsymbol{\alpha}} = \boldsymbol{\beta}\mathbf{\Lambda}^{-1} = \mathbf{U}\mathbf{D}\mathbf{V}^\top\mathbf{V}\mathbf{D}^{-1}\mathbf{V}^\top = \mathbf{U}\mathbf{V}^\top \quad \blacksquare \quad (3.37)$$

Le fait de désirer un grand nombre de composantes principales avec des poids nuls engendre un grand coût en termes de calculs. Par conséquent, la stratégie employée consistera à simplifier la fonction de coût à optimiser.

Théorème 5 *Soit* $\hat{\mathbf{V}}_i(\lambda) = \frac{\hat{\boldsymbol{\beta}}_i}{|\hat{\boldsymbol{\beta}}_i|}$ *les composantes principales provenant de (3.22). Soit* $(\hat{\boldsymbol{\alpha}}^\star, \hat{\boldsymbol{\beta}}^\star)$*, la solution du problème d'optimisation suivant :*

$$(\hat{\boldsymbol{\alpha}}^\star, \hat{\boldsymbol{\beta}}^\star) = \underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{argmin} \; -2tr(\boldsymbol{\alpha}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}) + \sum_{j=1}^{k}|\boldsymbol{\beta}_j|_2^2 + \sum_{j=1}^{k}\lambda_{1,j}|\boldsymbol{\beta}_j|_1 \quad s.c.\ \boldsymbol{\alpha}^\top\boldsymbol{\alpha} = \mathbf{I}_k \quad (3.38)$$

Lorsque $\lambda \to \infty, \hat{\mathbf{V}}_i(\lambda) \to \frac{\hat{\boldsymbol{\beta}}_i^\star}{|\hat{\boldsymbol{\beta}}_i^\star|}$.

L'algorithme est donc le suivant pour les k premières composantes principales :

Algorithme 2 : Algorithme de l'ACPC $p \gg n$.

1 Initialisons $\boldsymbol{\alpha}$ à partir de $\mathbf{V}[, 1:k]$, les k premières composantes principales.
2 Pour $\boldsymbol{\alpha}$ fixé, on résout le problème d'*elastic net* suivant pour $j = 1, 2, ..., k$:

$$\boldsymbol{\beta}_j = \underset{\boldsymbol{\beta}^\star}{\text{argmin}} \left(|\boldsymbol{\alpha}_j^\top\mathbf{X}^\top\mathbf{X}| - \frac{\lambda_{1,j}}{2}\right)_+ Sign(\boldsymbol{\alpha}_j^\top\mathbf{X}^\top\mathbf{X})$$

3 Répéter les étapes 2 et 3 jusqu'à la convergence de $\boldsymbol{\beta}_j$.
4 Normalisation : $\hat{\mathbf{V}}_j = \frac{\boldsymbol{\beta}_j}{|\boldsymbol{\beta}_j|}$ pour $j = 1, 2, ..., k$.

3.4.4 Propriétés intéressantes

D'un point de vue pratique, cette méthode comporte des propriétés intéressantes :

— Sans la contrainte similaire à *lasso*, la méthode se ramène au problème de l'ACP.
— Au niveau des calculs, elle devrait être efficace pour un petit p ou un grand p.
— Elle est capable de bien sélectionner les variables importantes.

3.4.5 Importance des variables

3.4.6 Algorithme `Boruta`

L'algorithme `Boruta` est une méthode de sélection de variables de type enveloppe basée sur un algorithme de classification de forêt d'arbres aléatoires implémentée dans la librairie `randomForest` (Liaw et Wiener, 2002) de R. Cet algorithme de classification est rapide, est normalement exécuté sans l'optimisation de paramètres et il procure une estimation numérique de l'importance des variables. C'est une méthode ensembliste qui effectue la classification en votant à partir de plusieurs classificateurs faibles et sans biais. Ces arbres sont développés indépendamment sur différents échantillons de l'ensemble l'entraînement. Ensuite, la mesure d'importance d'une variable est obtenue avec la diminution de la précision engendrée par la permutation aléatoire des valeurs de la variable entre les observations. Ceci est effectué séparément pour tous les arbres de la forêt aléatoire. En employant la terminologie de Achen (1982), cet algorithme mesure l'importance théorique de la variable.

`Boruta` est basé sur l'idée suivante. En ajoutant du bruit aléatoire au système et en recueillant des résultats des échantillons aléatoires, on peut réduire l'impact trompeur des fluctuations aléatoires de la précision et des corrélations. Le score Z calculé en divisant la perte moyenne de précision par son écart-type peut être utilisé comme mesure d'importance. Malheureusement, ce score ne peut pas être lié à la significativité statistique de la mesure d'importance de la variable (qui est renvoyée par l'algorithme de forêt aléatoire), car sa distribution ne suit pas une loi $\mathcal{N}(0,1)$ (Rudnicki et al, 2006). Néanmoins, l'algorithme utilise le score Z comme mesure d'importance, car il tient compte des fluctuations de la perte de précision moyenne parmi les arbres de la forêt.

Nous ne pouvons pas utiliser le score Z pour mesurer directement l'importance. Par conséquent, il faudrait une référence externe afin de décider si l'importance d'une variable donnée est significative, c'est-à-dire si l'importance est perceptible au-delà des bruits aléatoires. Pour ce faire, les créateurs `Boruta` ont créé un système d'information avec des variables construites aléatoirement. Pour chaque variable, une "imitation" lui est associée, dont les valeurs sont obtenues par permutation des valeurs de la variable originale par rapport aux observations. Une classification est ensuite effectuée en utilisant toutes les variables de ce système étendu

et l'importance des variables est calculée.

L'algorithme est composé des étapes suivantes :

1. Extension du système d'information en ajoutant des imitations de toutes les variables (le système d'information comporte toujours au moins 5 imitations, même si le nombre de variables dans le jeu de données est inférieur à 5).
2. Mélangez les variables ajoutées pour supprimer leurs corrélations avec la variable de réponse.
3. Exécutez un classificateur de forêt aléatoire sur le système d'information et rassemblez les scores Z calculés.
4. Trouvez le score Z maximal parmi les imitations (SZMI), puis affectez une récompense à chaque variable qui a mieux marqué que SZMI.
5. Pour chaque variable avec une importance indéterminée, effectuez un test d'égalité à deux côtés avec SZMI.
6. Considérez les variables qui ont une importance significativement inférieure à SZMI comme *sans importance* et les retirer définitivement du système d'information.
7. Considérez les variables qui ont une importance significativement plus élevée que SZMI comme *importante*.
8. Supprimez toutes les imitations.
9. Répétez la procédure jusqu'à ce que l'importance soit attribuée pour toutes les variables ou que l'algorithme atteigne le nombre maximal d'itérations.

En conclusion, cet algorithme mène à une sélection de variables sans biais et stable. De plus, cette méthode est robuste à la nature aléatoire des mesures d'importance d'une forêt aléatoire sans compter qu'elle tient compte des interactions entre les variables.

3.5 Classificateurs

3.5.1 Motivation

L'utilisation des classificateurs suivants peut être décomposée en deux parties. Premièrement, la régression logistique pénalisée servira à sélectionner les variables (deuxième étape de `SELECT()`). Deuxièmement, suite à la sélection de variables, il sera nécessaire d'évaluer la performance des classificateurs. En suivant des méthodologies claires et précises, les résultats obtenus permettront de comparer nos algorithmes.

3.5.2 Régression logistique pénalisée

L'idée de la régression logistique est de modéliser la probabilité d'appartenance à une des classes de la variable de réponse étant donné les variables explicatives fournies. Dans les modèles linéaires généraux, la fonction de lien (ou *link function*, en anglais) pour la famille binomiale est appelée *logit*. Son inverse est la fonction logistique qui prend une valeur réelle et la projette dans l'intervalle $[0, 1]$ de façon à obtenir une probabilité d'appartenance à une classe. La fonction sigmoïde est représentée à la Figure 3.4.

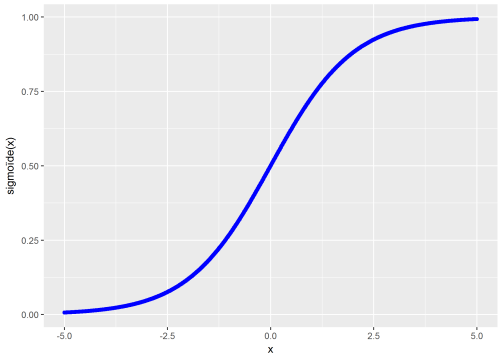


Figure 3.4 Représentation de la fonction sigmoïde.

La fonction sigmoïde a la forme suivante :

$$\hat{y} = P(y = 1|\mathbf{x}) = \frac{\mathrm{e}^{\mathbf{x}^\top \boldsymbol{\beta} + \beta_0}}{1 + \mathrm{e}^{\mathbf{x}^\top \boldsymbol{\beta} + \beta_0}} \tag{3.39}$$

Divers problèmes peuvent survenir lors de la modélisation de la régression logistique. Notre problème principal ici est le nombre élevé de prédicteurs par rapport au nombre d'observations. Pour remédier à ce problème, on ajoute une pénalité à la fonction de perte. Le modèle est alors obtenu en maximisant la fonction de coût suivante :

$$M(\boldsymbol{\beta}) = L(\boldsymbol{\beta}|\mathbf{x}) + \lambda P(\boldsymbol{\beta}) \tag{3.40}$$

où $M(\boldsymbol{\beta})$ est la fonction objectif, $L(\boldsymbol{\beta}|\mathbf{x})$ est la fonction de perte, $P(\boldsymbol{\beta})$ est la fonction de pénalisation et λ est un hyperparamètre contrôlant l'intensité de la pénalité. Plus précisément, la fonction de coût à maximiser pour la régression logistique pénalisée est :

$$M(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i (\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0) - \log(1 + \mathrm{e}^{\mathbf{x}^\top \boldsymbol{\beta} + \beta_0}) \right) - \lambda \overbrace{\left(\alpha ||\boldsymbol{\beta}||_1 + \frac{1}{2}(1 - \alpha) ||\boldsymbol{\beta}||_2^2 \right)}^{elastic\ net} \tag{3.41}$$

où $0 \leq \alpha \leq 1$ est un hyperparamètre donnant un compromis entre les deux pénalités sur $\boldsymbol{\beta}$. On note que $\lambda \geq 0$ et qu'il est l'hyperparamètre contrôlant la force de la pénalité sur l'*elastic net*. La librairie R utilisée pour résoudre ce problème se nomme `h2o` et permet de faire de la modélisation pour des problèmes d'analyse de données. Notre problème peut se résoudre en profitant des fonctions pour les modèles linéaires généraux implantés. Ces méthodes sont de nature similaire à celles employées par Friedman et al. (2009) concernant les chemins de régularisations pour les modèles linéaires généraux. Cette librairie calcule le chemin de régularisation en commençant à l'aide d'un modèle nul jusqu'à un modèle minimalement pénalisé (Nykodym, 2016). De plus, l'efficacité de la recherche de paramètres est améliorée par l'utilisation de règles fortes décrites par Bien et al. (2012) dans l'article *Strong Rules for Discarding Predictors in Lasso-type Problems*. De nombreux autres avantages existent et sont documentés dans le guide *Generalized Linear Modeling with* `h2o` par Nykodym et al. (2016).

3.5.3 Machine à vecteurs de support (MVS)

3.5.3.1 Théorie : données linéairement séparables

Un des classificateurs les plus populaires est la machine à vecteurs de support (Boser et al., 1992 ; Cortes et Vapnik, 1995). Ce modèle est basé sur la fonction linéaire suivante :

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \tag{3.42}$$

Le vecteur $\mathbf{w}$ correspond aux poids, b est le biais et $\mathbf{w}^\top \mathbf{x}$ est le produit scalaire entre les vecteurs $\mathbf{w}$ et $\mathbf{x}$. La classification binaire effectuée mène soit à la classe positive ($y_i = 1$) ou la classe négative ($y_i = -1$). Le classement dans la classe positive survient lorsque $\mathbf{w}^\top \mathbf{x} + b > 0$ tandis que lorsque $\mathbf{w}^\top \mathbf{x} + b < 0$, l'observation est dans la classe négative. Il s'agit de la règle de décision.

Il existe de multiples façons de séparer un ensemble de points de deux classes différentes. L'approche de la machine à vecteurs de support (MVS) consiste à maximiser la marge, c'est-à-dire la distance entre l'hyperplan séparant les classes et les points les plus proches de chaque classe (ou *support vectors*, en anglais). Ces points sont les vecteurs de support. Les deux droites contenant ces points sont nommées les droites de support.

Désignons $\mathbf{x}_+$ un point appartenant à la classe positive et $\mathbf{x}_-$ un point appartenant à la classe négative. On ajoutera les conditions suivantes au problème :

— Si le point appartient à la classe positive : $\mathbf{w}^\top \mathbf{x}_+ + b \geq 1$
— Si le point appartient à la classe négative : $\mathbf{w}^\top \mathbf{x}_- + b \leq -1$

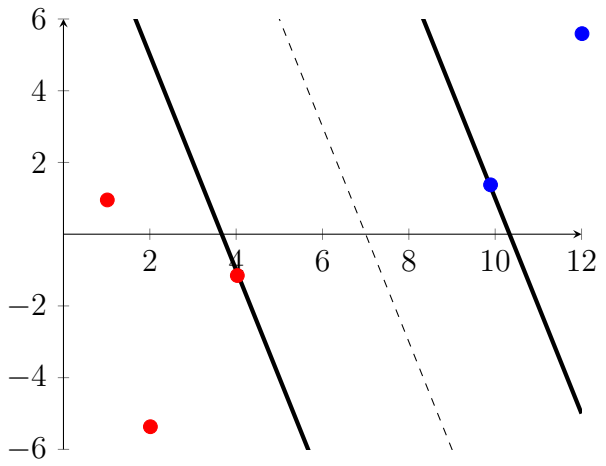


Figure 3.5 Application d'une MVS pour une classe négative (points rouges) et d'une classe positive (points bleus). L'axe des abscisses correspond à $\mathbf{x}_1$ et l'axe des ordonnées correspond à $\mathbf{x}_2$. La ligne pointillée correspond à la droite de séparation $\mathbf{w}^\top \mathbf{x} + b = 0$. Les droites tracées en gras correspondent aux droites de support.

On note que les inégalités sont transformées en égalités pour les vecteurs de support. Précédemment, la variable indicatrice de classe y_i a été définie et indique la classe de l'observation $\mathbf{x}_i$. En utilisant cette variable pour chacune des inégalités précédentes, on arrive à l'inégalité suivante : $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$. Cette expression peut être réécrite comme $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \geq 0$. En appliquant la remarque précédente, cette l'inégalité devient $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 = 0$ pour les vecteurs de support.

La marge correspond à la plus petite distance entre la droite de séparation et les vecteurs de support. Il faudrait maximiser cette marge. Considérons $\mathbf{w}$, un vecteur normal à l'hyperplan de séparation. On peut alors exprimer la marge de l'ensemble de données $\mathbf{D}$ par l'expression suivante :

$$m_{\mathbf{D}}(f) = (\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{||\mathbf{w}||} = \frac{(1 - b + b + 1)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||} \tag{3.43}$$

Maximiser la marge équivaut à minimiser $||\mathbf{w}||$ qui correspond à la norme euclidienne de $\mathbf{w}$. Les inégalités précédentes doivent aussi être considérées comme les contraintes dans ce problème d'optimisation. Ce problème d'optimisation peut être exprimé comme :

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^\top \mathbf{w} \quad \text{s.c } y_i(\mathbf{w}_i^\top \mathbf{x} + b) - 1 \geq 0 \quad (\text{pour } i = 1, 2, ..., n) \tag{3.44}$$

Le lagrangien du problème est donc :

$$L = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \sum_i \alpha_i \Big(y_i(\mathbf{w}_i^\top \mathbf{x} + b) - 1 \Big) \tag{3.45}$$

Les points de stationnarité sont obtenus en calculant les dérivées de premier ordre :

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{3.46}$$

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i \mathbf{x}_i = 0 \Rightarrow \sum_i \alpha_i \mathbf{x}_i = 0 \tag{3.47}$$

En substituant ces expressions dans le lagrangien initial, on obtient le résultat suivant :

$$L = \frac{1}{2}\Big(\sum_i \alpha_i y_i \mathbf{x}_i \Big)\Big(\sum_j \alpha_j y_j \mathbf{x}_j \Big) - \Big(\sum_i \alpha_i y_i \mathbf{x}_i \Big)\Big(\sum_j \alpha_j y_j \mathbf{x}_j \Big) - \sum_i \alpha_i \mathbf{x}_i b + \sum_i \alpha_i \tag{3.48}$$

$$\Rightarrow L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

Il se trouve que l'optimisation dépend des données par l'intermédiaire de produits scalaires entre les paires d'observations $\mathbf{x}$. En substituant la solution obtenue dans la règle de décision initiale, on trouve que l'on catégorise une observation $\mathbf{u}$ dans la classe positive si : $\sum_i \alpha_i y_i \mathbf{x}_i^\top \mathbf{u} + b \geq 0$. Ce problème doit alors être résolu de façon numérique.

3.5.3.2 Théorie : données linéairement inséparables

On a traité précédemment de l'utilisation d'une MVS dans le cas de données linéairement séparables. Mais qu'en est-il lorsque celles-ci ne le sont pas ? Dans cette nouvelle perspective, le problème d'optimisation traité est d'une nature similaire étant donné que l'on souhaite toujours maximiser la marge. Par contre, les contraintes doivent être relaxées afin de pouvoir permettre les mauvaises classifications. Les contraintes deviennent alors :

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad (\text{pour } i = 1, 2, ..., n) \tag{3.49}$$

où $\xi_i \geq 0$ (pour $i = 1, 2, ..., n$) sont les variables d'écarts. Ces variables permettent à n'importe quelle observation $\mathbf{x}_i$ d'être une erreur de marge (c'est-à-dire que $0 \leq \xi_i \leq 1$) ou une mauvaise classification ($\xi_i > 1$). Pour la fonction de perte, un terme de pénalisation $C \sum_i \xi_i$ est ajouté. L'hyperparamètre $C > 0$ contrôle l'intensité de la pénalité appliquée sur les mauvaises classifications et les erreurs de marge. Le problème d'optimisation est donc (Cortes et

Vapnik, 1995) :

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \xi_i \quad \text{s.c } y_i(\mathbf{w}_i^\top\mathbf{x} + b) - 1 - \xi_i \geq 0 \quad (\text{pour } i = 1, 2, ..., n) \tag{3.50}$$

Le dual de ce problème d'optimisation (Cortes et Vapnik, 1995 ; Cristianini et Shawes-Taylor, 2000 ; Schölkopf et Smola, 2002) est alors :

$$\min_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{s.c } \sum_i y_i\alpha_i = 0 \ , \ 0 \leq \alpha_i \leq C \quad (\text{pour } i = 1, 2, ..., n) \tag{3.51}$$

Comme pour les données séparables linéairement, le problème d'optimisation dépend des données par l'intermédiaire de produits scalaires. Le vecteur de poids $\mathbf{w}$ est exprimé en termes des observations. Plus précisément, on a que $\mathbf{w} = \sum_i y_i\alpha_i\mathbf{x}_i$.

3.5.3.3 Théorie : noyaux et leur(s) hyperparamètre(s)

La frontière de séparation dont il a été question précédemment est linéaire. Cela est restrictif en raison du fait que pour bien des problèmes un classificateur non linéaire serait plus efficace. L'avantage principal des classificateurs linéaires est principalement la facilité à les entraîner (Bishop, 2007 ; Hastie et al., 2001). Or, il existe une multitude de noyaux que l'on peut utiliser (voir le Tableau 3.2 pour des exemples de noyaux).

Une façon simpliste de transformer un classificateur linéaire en un classificateur non linéaire est de projeter nos données à partir de l'espace des données $\mathcal{X}$ vers un espace des variables $\mathcal{F}$ (c'est un espace d'Hilbert) en utilisant une fonction non linéaire $\phi : \mathcal{X} \to \mathcal{F}$. Il se trouve que dans l'espace $\mathcal{F}$, la fonction discriminante devient :

$$f(\mathbf{x}) = \mathbf{w}^\top\phi(\mathbf{x}) + b \tag{3.52}$$

Pour mettre en pratique cette fonction, il faut calculer les produits scalaires du noyau :

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top\phi(\mathbf{x}') \tag{3.53}$$

Il s'avère que l'on peut définir des noyaux de façon à ce que calculer $k(\mathbf{x}, \mathbf{x}')$ est plus efficace que de calculer $\phi(\mathbf{x})$ et $\phi(\mathbf{x}')$ puis ensuite effectuer $\phi(\mathbf{x})^\top\phi(\mathbf{x}')$. Ainsi, les méthodes de noyaux évitent de chercher une fonction projetant nos données vers un espace de variables à haute dimension. En tenant compte de la projection et de la solution de l'équation 3.51, on a que

$\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \phi(\mathbf{x}_i)$. Alors, on obtient que :

$$f(\mathbf{x}) = \sum_{i=1}^{n} y_i \alpha_i \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}) + b \tag{3.54}$$

Le problème est le fait que l'espace $\mathcal{F}$ est à haute dimension. Cela rend la transformation précédente ardue. Mais en utilisant l'équation 3.53, on peut reformuler la fonction discriminante de la façon suivante :

$$f(\mathbf{x}) = \sum_{i=1}^{n} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{3.55}$$

Tableau 3.2 Les noyaux fréquemment employés pour la machine à support de vecteurs ($d \in \mathbb{N}^\star, \gamma > 0$ et $r \in \mathbb{R}$).

Noyau	$k(\mathbf{x}, \mathbf{x}')$
Polynomial de degré d	$(\mathbf{x}^\top \mathbf{x}' + r)^d$
Gaussien (RBF)	$\exp(-\gamma \|\|\mathbf{x} - \mathbf{x}'\|\|^2)$
Linéaire	$\mathbf{x}^\top \mathbf{x}'$
Sigmoïde	$\tanh(\gamma \mathbf{x}^\top \mathbf{x}' + r)$

Il a été vu que l'hyper paramètre C (aussi appelé le coût) a un rôle important dans la MVS. Une valeur élevée de C signifie une pénalité plus forte sur les erreurs de classification et les erreurs de marge. Par conséquent, cela risque d'entraîner une diminution de la largeur de la marge. Pour le noyau polynomial, le degré d contrôle la flexibilité de la frontière de classification (Ben-Hur et Weston, 2010). En effet, une valeur plus élevée de d amène une frontière plus flexible. Il en va de même avec l'augmentation de γ pour un noyau gaussien.

Plusieurs hyperparamètres doivent être optimisés. Or, ces hyperparamètres diffèrent selon le choix du noyau. Il se trouve donc que le premier hyperparamètre à déterminer est le noyau. Ben-Hur et Weston (2010) recommandent l'exploration de divers noyaux en commençant par effectuer une évaluation de base avec le noyau linéaire. Ensuite, il faudrait tenter d'obtenir des meilleures performances en changeant de noyau. Ces auteurs mentionnent que la flexibilité d'un noyau gaussien et d'un noyau polynomial mène fréquemment à un problème de surapprentissage (ou *overfitting*, en anglais) pour des problèmes à hautes dimensions. Cependant, des expérimentations conduisent à croire que le noyau gaussien surpasse le noyau polynomial en termes de précision et de temps de convergence (Ben-Hur et Weston, 2010). Pour le noyau linéaire, seul le paramètre C affecte la performance de la MVS.

Pour cette recherche, les deux noyaux qui seront explorés sont le noyau linéaire et le noyau gaussien. D'un côté, le choix du noyau linéaire s'explique principalement par la simplicité de

son optimisation. Il suffit d'optimiser C qui pénalise les erreurs de marge et les mauvaises classifications. La recherche par coordonnées pour un seul hyperparamètre peut être effectuée, et ce, à un coût de calcul bas. De plus, il est recommandé d'employer un noyau linéaire pour un nombre de variables explicatives élevé (Hsu et al., 2016). D'un autre côté, le noyau gaussien est intéressant en raison de sa capacité à faire face à des relations non linéaires entre les variables explicatives et la variable de réponse. Il se trouve d'ailleurs que le noyau linéaire est un cas particulier du noyau gaussien (Keerthi et Lin, 2003). Aussi, le noyau sigmoïde agit comme le noyau gaussien pour certaines valeurs associées aux hyperparamètres (Lin et Lin, 2003). Finalement, ce noyau possède moins d'hyperparamètres à optimiser que le noyau polynomial et moins de difficultés numériques peuvent survenir (Hsu et al., 2016).

3.5.4 Machine à *gradient boosting* (MGB)

La machine à *gradient boosting* est un algorithme d'apprentissage automatique combinant deux concepts importants : l'optimisation par la méthode du gradient et le *boosting*.

En premier lieu, l'optimisation par la méthode du gradient consiste à minimiser une fonction de perte en utilisant des calculs du gradient. Ici, ces calculs sont effectués sur les données d'entraînement. En deuxième lieu, l'idée du *boosting* est d'assembler un groupe de modèles faibles (ou *weak learners*, en anglais) pour créer un modèle plus robuste (voir Figure 3.6). Un modèle faible est un modèle qui, quelle que soit la distribution sur les données d'apprentissage, fera toujours mieux que le hasard, lorsqu'il essaie d'étiqueter les données. Le *boosting* est particulièrement intéressant lorsqu'un modèle a une faible capacité, c'est-à-dire qu'il n'apprend pas vraiment les données «par coeur». Dans un tel cas, les prédictions sur l'ensemble d'entraînement sont généralement loin de correspondre aux vraies valeurs. Des modèles à faible capacité ont des petites variances. La clé réside dans le fait de diviser le travail. Les modèles sont entraînés en séquence, c'est-à-dire l'un après l'autre. Chaque modèle devra se concentrer sur les mauvaises prédictions des modèles précédents. Il s'agit donc de modifier les poids accordés aux prédictions.

L'algorithme MGB suivant sert à effectuer de la classification pour k classes. Il sera utilisé dans le cas de cette recherche pour le cas $k = 2$ avec l'aide de la librairie `h2o` du progiciel R.

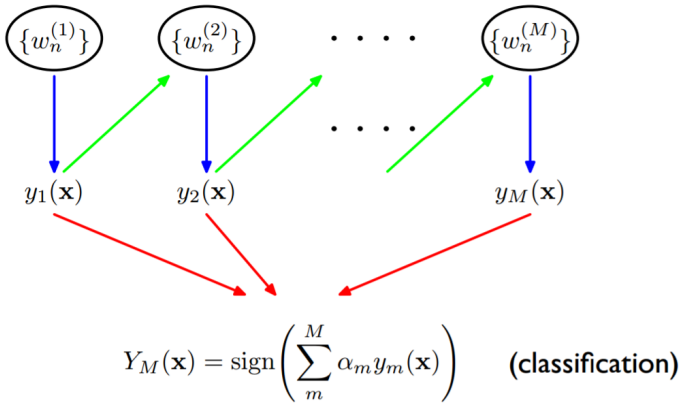


Figure 3.6 Illustration du *boosting* (tirée des notes de cours de Hugo Larochelle, 2015).

Algorithme 3 : MGB pour un problème de classification.

I. Initialiser $f_{k0} = 0$ pour $k = 1, 2, ..., K$

II. Pour $m = 1$ à M :

1. Poser $p_k(\mathbf{x}) = \frac{e^{f_k(\mathbf{x})}}{\sum_{l=1}^{K} e^{f_l(\mathbf{x})}}$ pour $k = 1, 2, ..., K$
2. Pour $k = 1$ à K :
   (a) Calculer $r_{ikm} = y_{ik} - p_k(\mathbf{x}_i)$ pour $i = 1, 2, ..., N$
   (b) Modéliser un arbre de régression sur les cibles r_{ikm} pour $i = 1, 2, ..., N$ étant donné les régions R_{jkm} pour $j = 1, 2, ..., J_m$
   (c) Calculer (pour $j = 1, 2, ..., J_m$) :

$$\gamma_{jkm} = \frac{(K-1)\sum_{\mathbf{x}_i \in R_{jkm}} r_{ikm}}{K \sum_{\mathbf{x}_i \in R_{jkm}} |r_{ikm}|(1 - |r_{ikm}|)}$$

   (d) Mettre à jour $f_{km}(\mathbf{x}) = f_{k,m-1}(\mathbf{x}) + \sum_{j=1}^{J_m} \gamma_{jkm} I(\mathbf{x} \in R_{jkm})$

III. Trouver $f_k(\mathbf{x}) = f_{kM}(\mathbf{x})$ pour $k = 1, 2, ..., K$

L'étape 2 consiste à construire K arbres de régression. Chaque arbre représente une classe de la variable de réponse. L'indice m (voir l'étape II) permet de savoir le nombre de modèles faibles ajoutés au modèle présent.

Dans cette boucle, il existe une autre boucle pour chacune des K classes. Dans cette boucle, on commence par calculer les résidus r_{ikm} (qui sont en fait les valeurs de gradient) pour chacune des N observations dans le modèle CART, puis on développe un arbre de régression avec ces calculs de gradient. Ce processus d'ajustement est parallélisé. Pour plus de détails

sur les aspects techniques de l'implantation de l'algorithme, un livret a été conçu à cet effet (Click et al., 2016).

Les hyperparamètres à optimiser sont : le taux d'apprentissage, le nombre d'arbres et la profondeur maximale des arbres. Le taux d'apprentissage détermine la contribution de chaque arbre au modèle en développement. De plus, il diminue la contribution de chaque arbre lorsqu'il est ajouté au modèle (Elith et al., 2008). Une diminution du taux d'apprentissage fait en sorte qu'un nombre d'arbres plus élevé est nécessaire. En général, un taux d'apprentissage bas et un grand nombre d'arbres sont préférables. Cependant, cette contrainte peut être difficile à respecter étant donné le nombre d'observations et le temps de calcul disponible. La profondeur des arbres est une contrainte sur l'espace des fonctions. Pour trouver des valeurs d'hyperparamètres intéressantes, le livret mentionné précédemment recommande deux méthodes : la recherche par grille et la recherche aléatoire.

3.6 Optimisation des hyperparamètres

3.6.1 Motivation

Trouver les hyperparamètres optimaux est un problème auquel les analystes de données sont souvent confrontés. La performance d'un modèle à généraliser sur des nouveaux exemples dépend des hyperparamètres. Par conséquent, pour surpasser les performances de nos prédécesseurs (voir Tableau 2.1), il faut déterminer de *bonnes valeurs* d'hyperparamètres. Or, ces derniers doivent être déterminés avant l'entraînement du modèle. Puisque l'on ne dispose pas de dérivées, la solution optimale (ou une approximation de celle-ci) est difficile à trouver. Pour approximer la solution optimale, la stratégie utilisée ici sera l'emploi de méthodes d'optimisation sans dérivées.

3.6.2 Recherche exhaustive

Une méthode rudimentaire dans un cadre général de problèmes d'optimisation est la recherche exhaustive aussi appelée recherche par force brute. L'idée est d'énumérer toutes les solutions candidates possibles et de les évaluer afin de trouver le candidat minimisant (ou maximisant) la fonction objectif. L'Algorithme 4 explicite le fonctionnement de la recherche exhaustive où Ω est l'espace des hyperparamètres.

Algorithme 4 : Recherche exhaustive.

Soit $f : \mathbb{R}^n \to \mathbb{R}$ et $\Omega \subseteq \mathbb{R}^n$.
0. Initialisation
$S = \{x^0, x^1, x^2, ...\}$
$k \leftarrow 0$
$x_{\text{meilleur}} = x^0 \in S$
$f_{\text{meilleur}} = f(x_{\text{meilleur}})$
1. Mise à jour
Évaluer $f(x^{k+1})$
Si $f(x^{k+1}) < f^k_{\text{meilleur}}$, alors
mettre à jour $x_{\text{meilleur}} \leftarrow x^{k+1}$, $f_{\text{meilleur}} = f(x_{\text{meilleur}})$
Incrémenter $k \leftarrow k + 1$ et retourner à l'étape 1.

La convergence de cet algorithme est facile à prouver, mais une limitation importante est lorsque l'ensemble des solutions candidates n'est pas dénombrable. En effet, il suffirait de faire face à un paramètre dont le support [2] est continu. Dans un tel cas, une infinité d'évaluations de la fonction objectif serait nécessaire.

3.6.3 Recherche par grille

Une des méthodes fréquemment utilisées est la recherche par grille. L'idée est que si l'espace des hyperparamètres Ω est borné, alors on peut évaluer la fonction objectif pour de nombreux points situés sur une grille de l'espace Ω. Par exemple, si l'on a $\Omega = [\ell, u] = \{x \in \mathbb{R}^n : \ell_i < x_i < u_i, i = 1, 2, ..., n\}$ qui est une boîte sur $\mathbb{R}^n$, alors on pourrait discrétiser x_i pour lui attribuer p valeurs équidistantes dans l'intervalle $[\ell_i, u_i]$. Puis, on sélectionne une valeur dans chacune des palettes de valeurs et l'on évalue le modèle pour toutes les combinaisons (p^n possibilités (voir l'Algorithme 5 pour plus de détails). Malheureusement, un problème qui survient est le fait que le nombre de modèles à créer augmente exponentiellement à mesure que le nombre d'hyperparamètres augmente. Par conséquent, si l'entraînement d'un modèle est long, on est limité dans le nombre de points de notre grille. Cette méthode est souvent employée en raison de sa simplicité.

2. Le support d'un paramètre est l'ensemble des valeurs que le paramètre peut prendre.

Algorithme 5 : Recherche par grille.

Soit $f : \mathbb{R}^n \to \mathbb{R}$ et $[\ell, u] \subseteq \mathbb{R}^n$.
0. Initialisation
$p \in \mathbb{N}$ et $p \geq 2$ (le nombre de points pour chaque hyperparamètre)
1. Évaluer f sur tous les points de la grille
Définir $\delta_i = {}^{(u_i - \ell_i)}/_{p-1}$ pour chaque $i \in \{1, 2, ..., n\}$
Créer $G = \{x \in [\ell, u] : x_i = \ell_i + z\delta_i, i = 1, 2, ..., n, z = 0, 1, ..., p-1\}$
Choisir $x_{\text{meilleur}} \in \text{argmin}\{f(t) : t \in G\}$

Il se trouve qu'il est possible de relaxer la contrainte sur l'équidistance des points. Une autre relaxation de contrainte est de prendre un nombre de points différent pour les hyperparamètres.

3.6.4 Recherche par coordonnées

Une alternative intéressante consisterait à utiliser un algorithme de recherche par coordonnées. Développé en 1952 par Fermi et Metropolis, cet algorithme se résume à partir d'un point initial $\mathbf{x}_0$, créer une sonde locale de quatre nouveaux points à évaluer puis se diriger dans la direction présentant la plus grande diminution (ou augmentation) selon l'évaluation de la fonction objectif (voir l'Algorithme 6 pour plus de détails). Il s'agit d'un algorithme parfois utilisé dans l'optimisation sans dérivée. Malheureusement, cette méthode est coûteuse en termes de calculs. Cependant, elle garantit une convergence vers un minimum local.

Algorithme 6 : Recherche par coordonnées pour deux hyperparamètres.

Entrée : $\mathbf{x}_0 \in \Omega_1 \times \Omega_2, \delta_0 > 0$ où Ω_i est le support du i-ième hyperparamètre .
Sorti : $\mathbf{x}^\star = (\alpha^\star, \lambda^\star)^\top$, les valeurs optimales trouvées par l'algorithme.
1 **Initialisation** : Choisir un point initial $\mathbf{x}_0 = (\alpha_0, \lambda_0)^\top \in \Omega_1 \times \Omega_2$.
2 **Sonde locale** : Pour $k \in \{0, 1, ...\}$, construire $P_k := \{\mathbf{x}_k \pm \delta_k \mathbf{e}_i : i \in \{1, 2, ..., n\}\}$ où $\mathbf{e}_i = (1, 0)^\top$ ou $\mathbf{e}_i = (0, 1)^\top$ et $\delta_k \in \mathbb{R}$.
— Si $f(\mathbf{t}) < f(\mathbf{x}_{k-1})$ pour un $t \in P_k$, alors $\mathbf{x}_{k+1} \leftarrow \mathbf{t}$.
— Sinon, on considèrera $\mathbf{x}_k$ comme un minimum local de P_k. Alors, on a que $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$ et $\delta_{k+1} \leftarrow \frac{\delta_k}{2}$.

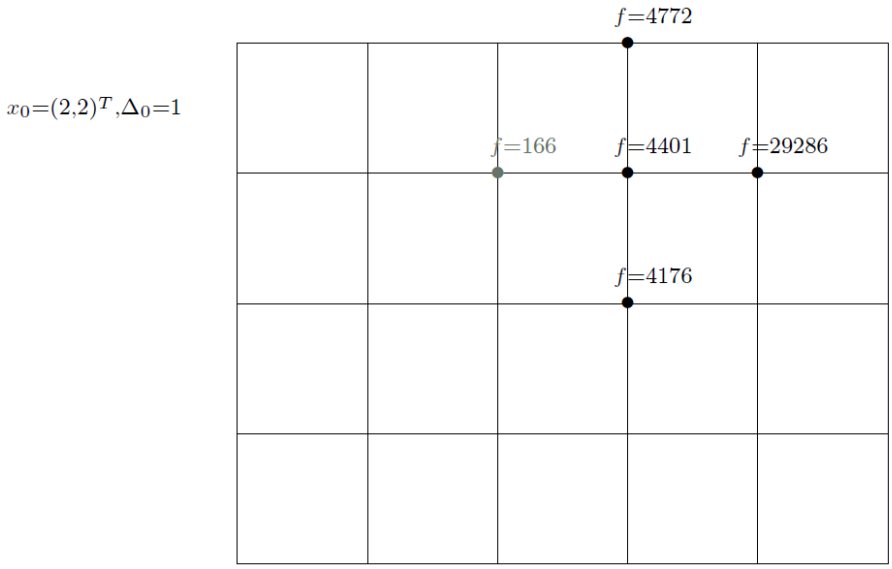


Figure 3.7 Exemple d'une itération initiale pour la méthode de recherche par coordonnées (Le Digabel, 2016). On a initialement $f(\mathbf{x}_0) = 4401$ et on calcule la fonction objectif dans les quatre directions. La direction de la plus grande diminution correspond au point à gauche de $\mathbf{x}_0$. Il s'agira de notre point initial à la prochaine itération.

3.6.5 Recherche aléatoire

La recherche aléatoire (Rastrigin, 1963) est une méthode d'optimisation consistant à se déplacer dans l'espace expérimental itérativement en se basant sur des points échantillonnés dans une hypersphère entourant le point actuel. La méthodologie est expliquée dans l'Algorithme 7.

Algorithme 7 : Recherche aléatoire (Rastrigin, 1963).

Soit $f : \mathbb{R}^n \to \mathbb{R}$ la fonction de coût à optimiser. Considérons $\mathbf{x} \in \mathbb{R}^n$ une position ou une solution candidate dans l'espace d'expérimentation. L'algorithme de recherche aléatoire peut être décrit de la façon suivante :
I. Initialiser $\mathbf{x}$ avec une position aléatoire sur l'espace d'expérimentation.
II. Jusqu'à ce qu'une règle d'arrêt soit atteinte (p. ex. un certain nombre d'itérations est accompli), répéter la procédure suivante :

1. Échantillonner une nouvelle position $\mathbf{y}$ de l'hypersphère d'un rayon fixé entourant l'emplacement actuel de $\mathbf{x}$.
2. Si $f(\mathbf{y}) < f(\mathbf{x})$ alors se déplacer à la nouvelle position en posant $\mathbf{x} = \mathbf{y}$.

3.7 Méthodes de comparaisons de traitement

3.7.1 Motivation

Si l'on fixe un classificateur, un algorithme de sélection de variables et une méthode d'optimisation, on obtient une valeur d'AUC et une valeur de précision. Choisissons l'AUC, sans perdre de généralité. Puisque l'expérience est répétée à dix reprises pour `SELECT()` et chacune de ses modifications, on obtient trois vecteurs d'AUC, chacun ayant dix composantes. Pour savoir si les modifications de l'algorithme `SELECT()` sont efficaces, il faudrait comparer le vecteur d'AUC de `SELECT()` aux vecteurs d'AUC des modifications. Pour ce faire, l'ANOVA à blocs complets est utilisée. Le classificateur est fixé, le facteur est l'algorithme de sélection de variables utilisé et la variable de blocage est le numéro de l'expérience.

3.7.2 Plans à blocs complets

Il est fréquent qu'il soit nécessaire de planifier l'expérience afin de minimiser la variabilité issue de variables nuisibles, c'est-à-dire des variables ayant une contribution sur la variable de réponse Y, mais qui n'intéressent pas l'expérimentateur.

Bloc 1	Bloc 2		Bloc b
Y_{11}	Y_{12}		Y_{1b}
Y_{21}	Y_{22}		Y_{2b}
Y_{31}	Y_{32}	$\cdots$	Y_{3b}
$\vdots$	$\vdots$		$\vdots$
Y_{a1}	Y_{a2}		Y_{ab}

Figure 3.8 Structure des données provenant d'un plan à blocs complets (pour a traitements et b blocs).

Le plan d'expérience en blocs est un prolongement du test t apparié qu'on utilise dans les situations où le facteur d'intérêt possède plus de deux modalités.

Traditionnellement, on choisit b blocs et l'on répète de façon identique l'expérience à un facteur dans chaque bloc (voir Figure 3.8). Il y a a observations dans chaque bloc, et l'ordre d'exécution de ces observations dans chaque bloc est assigné au hasard. On représente les observations par le modèle statistique linéaire :

$$Y_{ij} = \mu + \tau_i + \beta_j + E_{ij} \text{ pour } i = 1, 2, ..., a \text{ et } j = 1, 2, ..., b \tag{3.56}$$

où μ représente la moyenne globale, τ_i l'effet du $i^{\text{ième}}$ traitement, β_j est l'effet aléatoire du

$j^{\text{ième}}$ bloc et E_{ij}, le terme d'erreur aléatoire.

Par définition, les effets des traitements sont des écarts à la moyenne globale, de sorte que $\sum_{i=1}^{a} \tau_i = 0$. Le modèle suppose également que $\beta_j \sim \mathcal{N}(0, \sigma_\beta^2)$, que $E_{ij} \sim \mathcal{N}(0, \sigma^2)$ et que les β_j et les E_{ij} sont indépendants. Le but est de tester l'égalité des effets des traitements :

$$H_0 : \tau_1 = \tau_2 = ... = \tau_a = 0 \text{ vs } H_1 : \tau_i \neq 0 \text{ pour au moins un } i \tag{3.57}$$

Soient :

- $Y_{i.}$ et $\bar{Y}_{i.}$, la somme et la moyenne de toutes les observations pour le traitement i.
- $Y_{.j}$ et $\bar{Y}_{.j}$, la somme et la moyenne de toutes les observations dans le bloc j.
- $Y_{..}$ et $\bar{Y}_{..}$, la somme et la moyenne de toutes les observations.
- $N = ab$, le nombre total d'observations.

Sous forme simplifiée, la somme des carrés s'exprime comme :

$$\sum_{i=1}^{a}\sum_{j=1}^{b}(Y_{ij} - Y_{..})^2 = b\sum_{i=1}^{a}(\bar{Y}_{i.} - \bar{Y}_{..})^2 + a\sum_{j=1}^{b}(\bar{Y}_{.j} - \bar{Y}_{..})^2 + \sum_{i=1}^{a}\sum_{j=1}^{b}(Y_{ij} - \bar{Y}_{i.} - \bar{Y}_{.j} + \bar{Y}_{..})^2 \tag{3.58}$$

De façon symbolique, on l'exprime comme :

$$SS_T = SS_A + SS_{blocs} + SS_E \tag{3.59}$$

Les degrés de liberté sont respectivement :

$$N - 1 = (a - 1) + (b - 1) + (a - 1)(b - 1) \tag{3.60}$$

L'absence d'effets du facteur A, l'hypothèse H_0 est confrontée au moyen de la statistique :

$$F_0 = \frac{MS_A}{MS_E} \overset{\text{sous } H_0}{\sim} F_{a-1,(a-1)(b-1)} \tag{3.61}$$

Les résultats peuvent être résumés dans la table suivante :

Table ANOVA				
Source	**SS**	**df**	**MS**	F_0
facteur A	SS_A	$a - 1$	MS_A	$F_0 = \frac{MS_A}{MS_E}$
blocs	SS_{blocs}	$b - 1$	MS_{blocs}	
erreur	SS_E	$(a - 1)(b - 1)$	MS_E	
total	SS_T	$N - 1$		

3.7.3 Test d'hypothèses (forme générale)

Considérons un test d'hypothèse où $X \sim \text{Loi}(\theta)$. Un test d'hypothèses est une procédure basée sur un échantillon de données dans le but de rejeter ou ne pas rejeter l'hypothèse nulle. Normalement, on a que :

— Hypothèse nulle :$H_0 : \theta = \theta_0$

— Contre-hypothèse :

1. $H_1 : \theta \neq \theta_0$ (bilatéral)
2. $H_1 : \theta > \theta_0$ (unilatéral à droite)
3. $H_1 : \theta < \theta_0$ (unilatéral à gauche)

Tableau 3.3 Tableaux des résultats possibles pour un test d'hypothèses (où $\alpha = P(H_0 \text{ rejetée}|H_0 \text{ vraie})$ et $\beta = P(H_0 \text{ acceptée}|H_0 \text{ fausse})$).

	H_0 vraie	H_0 fausse
H_0 est rejetée	Erreur de type I (α)	$1 - \beta$
H_0 n'est pas rejetée	$1 - \alpha$	Erreur de type II (β)

Par exemple, supposons que l'on obtient différents résultats de AUC pour une régression logistique pénalisée entrainée une seule fois sur plusieurs permutations d'un jeu de données. On s'intéresse à savoir si ces résultats battent la meilleure AUC trouvée dans la littérature (appelons-la θ_0). Il serait alors pertinent d'effectuer un test d'hypothèses avec une contre-hypothèse unilatérale à droite. ($H_0 : \theta = \theta_0$ vs $H_1 : \theta > \theta_0$).

3.7.4 Comparaisons multiples

Lorsqu'on effectue plusieurs tests ou intervalles de confiance simultanément, un problème survient. Il s'agit du contrôle de la probabilité d'une erreur de première espèce. Supposons qu'on effectue K tests et (par simplicité) que ces tests sont indépendants. Soit R_i la région critique du test i effectué au niveau α pour l'hypothèse H_i. Chaque test satisfait individuellement $P(R_i|H_i) = \alpha$. Dans le cas des tests simultanés, la probabilité de commettre au moins une erreur de première espèce est :

$$P\Big(\bigcup_{i=1}^{K} R_i \Big| \bigcap_{i=1}^{K} H_i\Big) = 1 - \prod P(R_i^c|H_i) = 1 - (1-\alpha)^K \tag{3.62}$$

Pour fin d'illustration, en choisissant $\alpha = 0.05$ et $K = 5$, cette probabilité est de 0.23. Dans le cas plus réaliste où les tests sont possiblement dépendants, on peut utiliser l'inégalité de

Bonferroni :

$$P\Big(\bigcap_{i=1}^{K} R_i^c \Big| \bigcap_{i=1}^{K} H_i\Big) = 1 - P\Big(\bigcup_{i=1}^{K} R_i \Big| \bigcap_{i=1}^{K} H_i\Big) \geq 1 - \sum_{i=1}^{K} P(R_i|H_i) = 1 - K\alpha \tag{3.63}$$

Il suffit d'effectuer chaque test individuel au niveau $\frac{\alpha}{K}$ pour être assuré de ne commettre aucune erreur de première espèce avec une probabilité d'au moins $1 - \alpha$. C'est la méthode de Bonferroni qui contrôle le niveau global en présence de tests simultanés. Cette méthode procure de bons résultats lorsque le nombre de tests K est petit.

3.8 Critères de performance

La tâche de prédiction traitée ici est un problème de classification. Plus particulièrement, il s'agit d'un problème de classification binaire (à deux classes). Ce genre de problème est traditionnellement traité avec la précision (le taux de succès) ou son complément, le taux d'erreur. Cependant, il existe une panoplie d'autres critères de performance : la sensibilité et la spécificité, l'AUC, le score de Brier, etc.

3.8.1 Sensibilité et spécificité

Considérons un problème de classification binaire : soit positive ou négative. Pour chaque observation, il se trouve que la classe prédite ne correspond pas toujours à la véritable classe. Par conséquent, le classement d'une observation peut mener à quatre cas dont deux sont des erreurs. Les deux cas qui ne sont pas des erreurs correspondent à bien classifier une observation positive (ou négative). Ces deux cas sont appelés : vrai positif et vrai négatif. La première erreur est d'attribuer une étiquette négative à une observation de classe positive. Cette observation erronée se nomme faux négatif. La deuxième erreur est d'attribuer une étiquette positive à une observation de classe négative. Cette observation se nomme faux positif. Les cas décrits peuvent être regroupés dans un tableau de contingence (voir Tableau 3.4). En excluant la colonne totale du Tableau 3.4, on obtient la matrice de confusion du problème de classification binaire étudié.

Deux mesures de performances souvent utilisées sont la sensibilité et la spécificité. La sensibilité correspond au nombre de vrais positifs divisé par le nombre d'observations dans la classe positive. Il s'agit donc de la proportion de vrais positifs basée sur les observations de la classe positive. En considérant les nombres du Tableau 3.4, la sensibilité est écrite de la façon suivante : $S_N = {}^{a_{11}}/_{(a_{11}+a_{12})}$. De façon analogue, on peut considérer la proportion de faux positif comme le nombre de vrai négatif sur le nombre d'observations dans la classe négative. Cette

proportion fp peut être écrite comme $fp = {a_{22}}/{(a_{22} + a_{21})}$. Cette expression est utilisée dans le calcul de la spécificité. La spécificité peut être écrite comme : $S_P = 1 - fp = {a_{21}}/{(a_{21} + a_{22})}$. Ces critères peuvent être combinés pour obtenir l'AUC, qui est plus facile à analyser.

Tableau 3.4 Tableau de contingence des divers cas pour une classification binaire.

	Classification		
Classe	Positive	Négative	Total
Positive	a_{11}	a_{12}	$a_{11} + a_{12}$
Négative	a_{21}	a_{22}	$a_{21} + a_{22}$

3.8.2 Précision

La précision (ou *accuracy*, en anglais) correspond au nombre d'observations bien classé divisé par le nombre total d'observations. En se fiant au Tableau 3.3, on peut formuler la précision comme $\bar{prec} = {(a_{11} + a_{22})}/{(a_{11} + a_{12} + a_{21} + a_{22})}$. Une autre façon de l'écrire est :

$$\bar{prec} = 1 - \frac{1}{N} \sum_{i=1}^{N} L_{0/1}(y_i, \hat{y}_i) \tag{3.64}$$

où N est la taille de l'ensemble test et $L_{0/1}()$ est une fonction de perte définie de la façon suivante :

$$L_{0/1}(y_i, \hat{y}_i) = \begin{cases} 0 & \text{si } y_i = \hat{y}_i \\ 1 & \text{sinon} \end{cases} \tag{3.65}$$

Un des défauts de la précision est qu'elle peut donner une estimation biaisée vers le haut lorsqu'une des deux classes est surreprésentée dans l'ensemble d'entraînement. Ce critère n'est pas toujours une bonne mesure d'évaluation de performance (Provost et Fawcett, 1997 ; Provost et al., 1998). Il se trouve que la comparaison d'algorithme basée sur le critère de précision est insatisfaisante lorsqu'il n'y a pas de classificateur dominant. Cette mesure de performance suppose un coût de mauvaise classification égal (pour un faux négatif et un faux positif). Cette hypothèse est problématique, car pour la majeure partie des problèmes (dont celui étudié ici) l'une des erreurs de classification est pire que l'autre.

Cependant, la précision est fréquemment utilisée dans la littérature. Pour tous les problèmes de classification, sa valeur varie entre zéros et un et l'interprétation est toujours la même. Une valeur près de un signifie que le modèle classe très bien les observations. Une valeur près de zéro indique que le modèle est presque incapable de classer une observation correctement. La popularité de ce critère et son interprétation intuitive font en sorte que ce critère sera employé dans cette recherche.

3.8.3 Courbe ROC et AUC

La courbe de ROC (ou *receiver operating characteristics graph*, en anglais) est un graphique permettant la visualisation et la sélection de classificateurs. Il se trouve que l'analyse de courbe de ROC est très utilisée dans la littérature en lien avec la prise de décision médicale (Zou, 2002). D'ailleurs, l'utilisation de ce type de graphique a connu un essor dans la communauté des analystes de données.

Une courbe ROC est un graphique à deux dimensions exprimant généralement la sensibilité sur l'axe vertical et la proportion de faux positif (fp) sur l'axe horizontal. Étant donné la relation entre la proportion de faux positif et la spécificité, la courbe ROC est parfois exprimée en changeant le taux de faux positif par la spécificité. Cependant, ce changement nécessite une inversion de l'ordre des valeurs afin de préserver l'allure et l'interprétation de la courbe.

Un modèle dont la matrice de confusion présente une classification parfaite est tel que $S_N = 1$ et $S_P = 1$. Le point $(1, 1)$ correspond donc à une classification parfaite. Le problème de classification mène à l'obtention d'une classe prédite pour chaque observation de l'ensemble de validation. Les classes prédites sont exprimées sous la forme d'un vecteur de valeurs discrètes ou sous la forme d'une matrice de probabilités où chaque ligne désigne la probabilité d'appartenance à chacune des classes pour l'observation en question. Dans le cas de la matrice, il faut décider d'un seuil au-dessus duquel la classe sera la classe prédite. Différents seuils mènent à différents couples (S_P, S_N). Ce sont ces seuils qui sont présentés sur la courbe ROC (voir Figure 3.9).

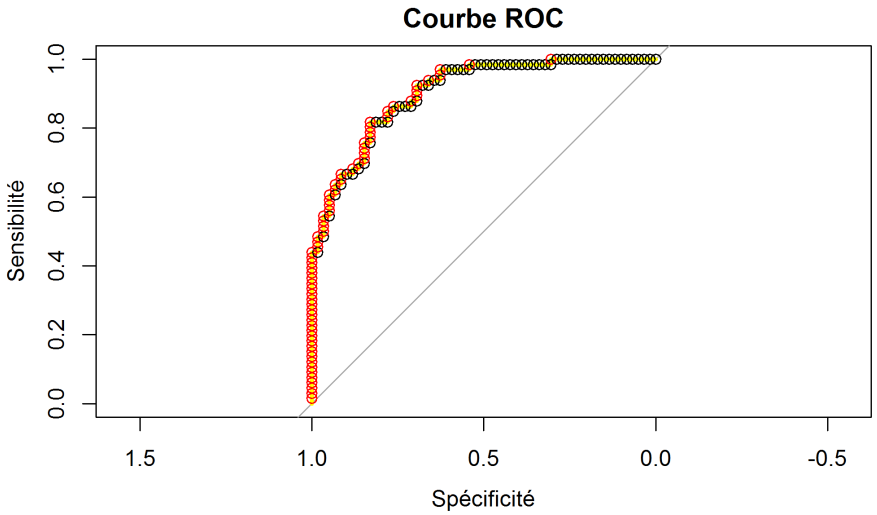


Figure 3.9 Courbe ROC avec un AUC de 0.9037.

Une mesure de performance fréquemment utilisée est l'aire sous la courbe ROC (ou *Area*

Under the Receiver Operating Characteristic curve, en anglais). Cette mesure est souvent appelée AUC (ou *Area Under the Curve*, en anglais) ce qui est ambigu en raison du fait que la courbe n'est pas spécifiée dans cette appellation. Malgré tout, cet acronyme sera utilisé pour la suite pour alléger le document. Cette mesure varie entre 0 et 1. Une valeur près de zéro correspond à un modèle exécrable et une valeur proche de un correspond à un modèle presque parfait (voir Tableau 3.5). Une valeur d'AUC de 0.5 (la courbe a alors l'allure d'une ligne droite) indique que le modèle prédit aléatoirement avec équiprobabilité pour chaque classe.

Malheureusement, cette mesure de performance comporte plusieurs inconvénients. Le plus important est que l'AUC utilise différentes fonctions de coûts pour les différents classificateurs (Hand, 2009). Cela revient à dire qu'employer l'AUC est comme utiliser différents critères de classification. Cependant, malgré cet inconvénient majeur, ce critère de performance est parmi les plus utilisés !

Tableau 3.5 Qualité attribuée à un modèle selon son AUC dans le domaine de la médecine (de Souza Lauretto, 2013).

AUC	**Qualité du modèle**
$]0.9, 1]$	Excellent
$]0.8, 0.9]$	Bon
$]0.7, 0.8]$	Acceptable
$]0.6, 0.7]$	Pauvre
$]0.5, 0.6]$	Mauvais

L'AUC s'interprète facilement et comme pour la précision, elle est souvent utilisée dans la littérature. Ces qualités et la popularité de ce critère dans la littérature font en sorte que l'on emploiera ce critère de performance.

3.8.4 Score de Brier

Le score de Brier est exprimé de la façon suivante :

$$B_s = \frac{1}{N}\sum_{t=1}^{N}(f(t) - o(t))^2 \tag{3.66}$$

où $f(t)$ est la probabilité d'appartenance à la bonne classe pour la prédiction de l'observation t et $o(t)$ est la classe de l'observation t. On note la ressemblance avec l'erreur quadratique moyenne. Cette fonction de coût est normalement utilisée pour des problèmes de classification binaire. Malheureusement, le score de cette fonction ne s'interprète pas aussi bien que l'AUC

et la précision. Sa valeur dépend des données utilisées. On ne l'utilisera donc pas.

3.8.5 Choix de critères

Afin de permettre la comparaison avec les résultats de la littérature en lien avec cet ensemble de données et d'avoir une interprétation plus intuitive, les critères de performance évalués seront l'AUC et la précision, faute de mieux.

CHAPITRE 4 RECONSTITUTION DE L'ALGORITHME SELECT()

Comme il a été mentionné précédemment, l'algorithme SELECT() (Occampo-Vego et al., 2016) a apporté des résultats prometteurs en augmentant de plus de 10% la précision de classification dans les problèmes traités dans leur article. Dans la section 3.4 de leur article, les auteurs expliquent brièvement la méthodologie de SELECT(). Cependant, certaines explications manquent de clarté et il semble y avoir certaines contradictions. Les étapes manquantes de clarté ont été reconstituées aussi rigoureusement que possible.

L'originalité de notre recherche provient des modifications apportées à cet algorithme. Afin de pouvoir mesurer l'efficacité de ces changements, la méthode de sélection de variables devra être implantée le plus fidèlement possible. Deux versions modifiées de l'algorithme sont créées, chacune avec une seule modification. Ensuite, les classificateurs choisis précédemment sont entraînés, chacun avec une méthodologie spécifique. Cette méthodologie sera également appliquée pour les différentes versions de l'algorithme modifié.

4.1 Application des étapes de la méthode

L'article de Ocompo-Vega et al. (2016) décrit les étapes de la méthode SELECT() (voir Algorithme 1). Chacune de ces étapes est exécutée sur le progiciel R et des explications sur la méthodologie sont fournies.

Étape 1 : $\mathbf{Q} \leftarrow ACP(\{\mathbf{x}_i\}_{i=1}^{n}, \upsilon)$

L'ensemble d'entraînement utilisé est composé de 116 observations et de 2905 variables explicatives. Il se trouve que l'on a plus de variables que d'observations. Par la décomposition en valeurs singulières, la matrice d'expérimentation $\mathbf{X}_{116\times2905}$ se décompose en des matrices : $\mathbf{V}^{\top}_{116\times2905}$, $\mathbf{D}_{116\times116}$ et $\mathbf{U}_{116\times116}$. La projection des données vers les composantes principales donne la matrice $\mathbf{XV}_{116\times116}$.

Le seuil υ correspond au pourcentage de la variance totale à conserver. Cela signifie qu'on choisira le nombre minimal de composantes principales expliquant υ% de la variance totale. Diverses méthodes existent pour sélectionner υ, mais le choix de υ sera simplement la même valeur que celle préconisée dans les expérimentations de Occampo-Vego et al. (2016). Cette valeur est $\upsilon = 90$ et mène à conserver les 55 premières composantes principales.

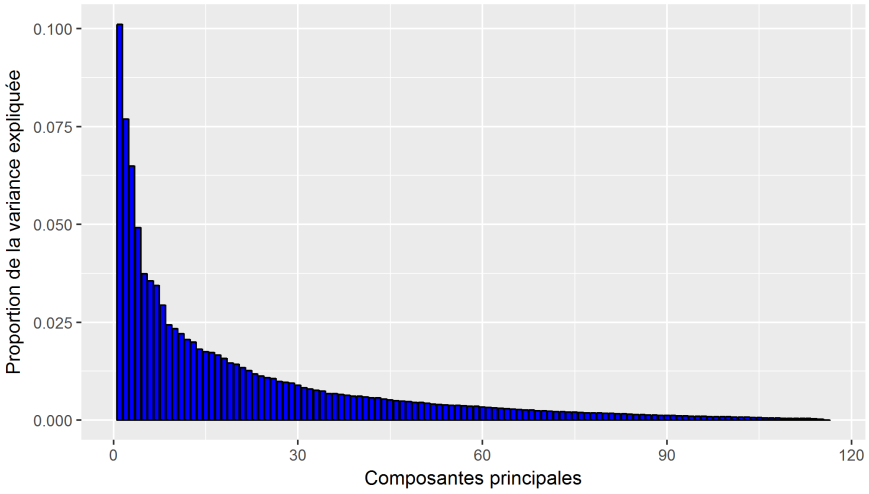


Figure 4.1 Proportion de la variance pour chaque composante (pour l'ensemble d'entraînement).

Étape 2 : $\mathbf{Q}_1 \leftarrow LOGIT(\mathbf{Q}, d)$

La sélection de variables devrait se faire en utilisant la régression logistique. L'article stipule qu'il faut conserver les d variables les plus pertinentes. Malheureusement, il est fréquent d'avoir des problèmes lors de la modélisation de la régression logistique. Des problèmes tels que la multicolinéarité, le problème de séparation et la non-identifiabilité (du maximum de vraisemblance) peuvent survenir (Albert et Anderson, 1984). Un problème d'une nature similaire est apparu dans ce mémoire. Gelman et al. (2008) ont proposé d'implanter une distribution a priori faiblement informative afin de régler ce problème. Cette a priori n'est pas hiérarchique et est construite de la façon suivante :

1. Centrer les variables (non binaires) à zéro et réduire l'écart-type à 0.5.
2. Placer des distributions a priori indépendantes de Student-t sur les coefficients.

L'option par défaut sera celle conseillée par Gelman, c'est-à-dire une distribution de Cauchy centrée en zéro avec une échelle de 2.5. Un des bénéfices de cette distribution est qu'une réponse peut toujours être obtenue et ce peu importe la nature des données du problème étudié. Cela est avantageux pour une automatisation de la procédure. L'application de cette régression mène alors à l'obtention d'un modèle dont on possède les coefficients et les erreurs types. Nous disposons maintenant des statistiques t. La magnitude du coefficient β_j est $d_j = |\hat{\beta}_j|$ où $\hat{\beta}_j$ est l'estimation obtenue par la régression bayésienne. L'article de Ocampo-Vega regarde la taille des coefficients puis décide empiriquement d'un seuil ℓ au-delà duquel

on doit conserver les variables. Cela équivaut à choisir les d composantes avec les plus grands coefficients (en valeur absolue). On posera $d = 50$, ce qui correspond environ à conserver 90% des variables. Il est important de mentionner que la valeur de 90% a été choisie arbitrairement.

Étape 3 : $\{\mathbf{z}_i\}_{i=1}^n \leftarrow EnleverAttributs(\mathbf{Q}_1, \mu)$

En se débarrassant de certaines composantes, il se trouve que nous venons de réduire la matrice des poids $\mathbf{V}$ en conservant les colonnes correspondant à chacune des 50 composantes les plus importantes. La prochaine étape consiste à arrondir à zéro les poids inférieurs (en valeur absolue) au seuil μ dans la matrice $\mathbf{V}$. Désignons $\mathbf{V}^{(i)}$ comme $\mathbf{V}$ après la $i^{\text{ième}}$ étape (pour $i = 1, 2$ et 3). Concentrons-nous sur $\mathbf{V}^{(2)}$. Chaque ligne de cette matrice correspond aux poids d'un gène sur les composantes conservées (voir Figure 4.2). Mathématiquement, on dira que l'élément $v_{i,j}^{(2)}$ correspond au poids du gène i pour la composante principale issue de la j-ième colonne de $\mathbf{Q}_1$.

$$\mathbf{V}^{(2)} = \begin{pmatrix} v_{1,1}^{(2)} & v_{1,2}^{(2)} & \dots & v_{1,50}^{(2)} \\ v_{2,1}^{(2)} & v_{2,2}^{(2)} & \dots & v_{2,50}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{2905,1}^{(2)} & v_{2905,2}^{(2)} & \dots & v_{2905,50}^{(2)} \end{pmatrix} \begin{matrix} \text{Poids du } 1^{\text{er}} \text{ gène} \\ \text{Poids du } 2^{\text{ième}} \text{ gène} \\ \vdots \\ \text{Poids du } 2905^{\text{ième}} \text{ gène} \end{matrix}$$

Figure 4.2 Matrice de rotation $\mathbf{V}$ après la deuxième étape.

Il se trouve que μ est un hyperparamètre à déterminer par l'expérimentateur. Plus la valeur de μ est grande, plus des poids de la matrice seront arrondis à zéro. Pour le cas où $\mu = 0$, aucun des éléments de $\mathbf{V}^{(2)}$ est arrondi à zéro. Cette méthode est simplement un seuillage simple. Après cette opération, la matrice $\mathbf{V}^{(3)}$ sera composée d'un plus grand nombre de zéros (par rapport à $\mathbf{V}^{(2)}$) et sera alors plus creuse. Ocampo-Vega et al. (2016) mentionne que le choix de μ se fait en regardant le diagramme en boîte des coefficients $\hat{\beta}$. Cependant, aucune méthodologie n'est spécifiée, ce qui limite la reproductibilité de leur méthode. Par conséquent, le choix de $\mu = 0.03$ de cette recherche est estimé de façon à avoir une valeur similaire[1] aux expérimentations (sur la leucémie et le lymphome) de Ocampo-Vega et al. (2016). Ce choix mène à l'obtention d'une matrice $\mathbf{V}^{(3)}$ composée de 90.35% de zéros.

Une fois cette opération effectuée, il faut regarder le nombre de poids non nul lié à chaque gène. La procédure de l'article étudié propose de conserver les gènes ayant le plus grand

1. En termes de proportion de zéros dans la matrice $\mathbf{V}^{(3)}$.

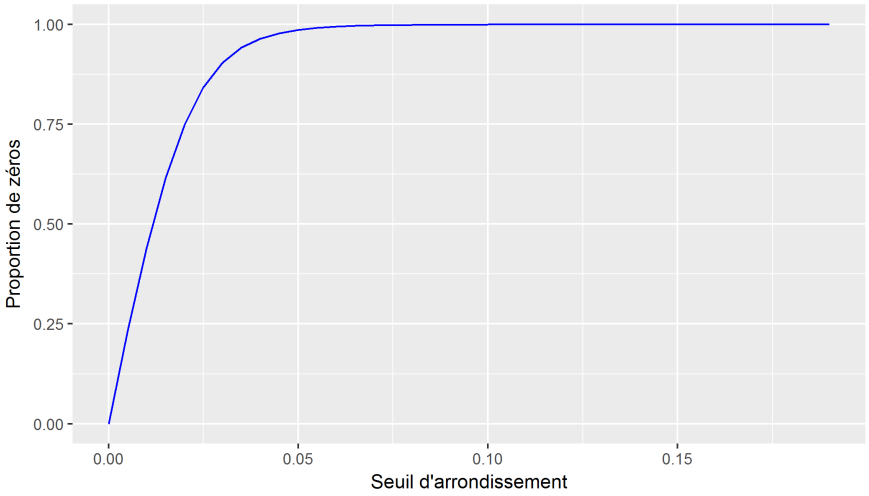


Figure 4.3 Proportion de zéros dans la matrice $\mathbf{V}^{(3)}$ selon le seuil d'arrondissement μ choisi.

nombre de poids. Par conséquent, il semble qu'un gène ayant un grand nombre de poids non nuls apporte plus d'informations au modèle. Par exemple, en observant le Tableau 4.1, on constate que l'expérience du lymphome a conduit à conserver 972 gènes et celle de la leucémie a permis la sélection de 422 gènes. Cela correspond à garder 13.63% des gènes (pour le lymphome) et 5.91% des gènes (pour la leucémie).

Tableau 4.1 Fréquence des poids pour les différents gènes (tirée de Ocampo-Vega et al., 2016).

Ensemble de données	μ	0	1	2	3	4
Lymphome	0.006	236	1117	2296	2508	972
Leucémie	0.01	1846	2861	2000	422	-

En considérant les proportions des problèmes précédents (5.91% et 13.63%), on est porté à considérer qu'il faudrait garder entre 172 et 395 gènes de notre ensemble de données. Malheureusement, on dispose d'une plus grande variété de nombres de poids, c'est-à-dire que l'on a beaucoup de gènes composés d'un nombre de poids différent (voir Figure 4.5). En se fiant au Tableau 4.2, on est contraint de conserver seulement les gènes ayant plus de neuf poids non nuls sur les composantes pour garder le maximum de gènes. Ce choix s'explique par le fait que l'on veut conserver le maximum d'information en conservant moins de 395 gènes. Cela mène à la sélection de 322 variables spécifiques pour chacun des ensembles de données (entraînement, validation et test)

L'histogramme de la Figure 4.4 diffère des histogrammes de la Figure 4.3. Une explication plausible est que Ocampo-Vega et al. (2016) ont choisi beaucoup moins de composantes à la première ou la deuxième étape.

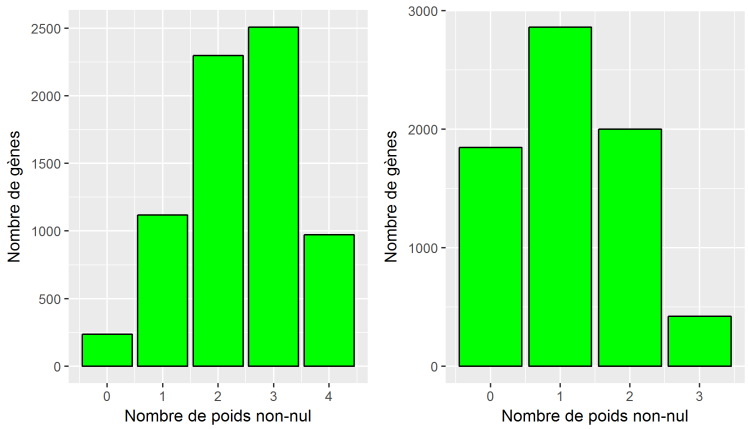


Figure 4.4 Histogrammes de la fréquence des gènes selon le nombre de poids non-nuls (à gauche : Lymphome, à droite : Leucémie).

Tableau 4.2 Nombre de gènes sélectionnés selon le nombre de poids non nuls minimal souhaité.

	i						
	0	1	2	...	8	9	10
Nombre de gènes avec plus de i poids non nuls	2686	2365	1943	...	426	**322**	242

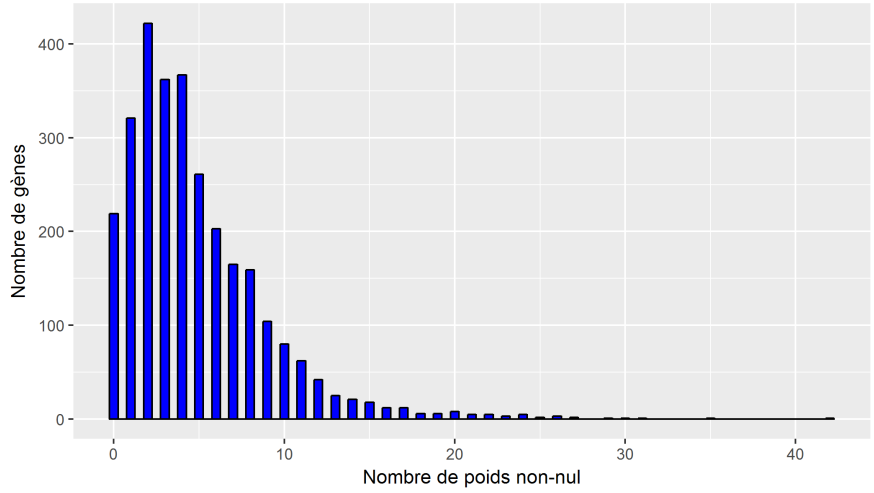


Figure 4.5 Histogramme de la fréquence des gènes selon le nombre de poids non nuls pour l'ensemble de données d'entraînement.

Étape 4 : $\mathbf{Q}_2 \leftarrow ACP(\{\mathbf{z}_i\}_{i=1}^n, q_2)$

La dernière étape consiste simplement à appliquer l'analyse en composantes principales sur notre nouvel ensemble de données. Il ne faut pas oublier que la valeur q_2 auquel fait référence

l'algorithme est simplement le nombre de composantes principales que l'on désire conserver. Dans notre cas, toutes les composantes principales sont conservées.

Cette étape permet de se débarrasser du problème de multicolinéarité en profitant de l'orthogonalité des composantes principales. Une fois cette analyse en composantes principales effectuée, on l'applique à l'ensemble de validation et à l'ensemble test.

4.2 Optimisation des hyperparamètres

La section suivante décrit l'optimisation dans le cas d'une seule expérience, c'est-à-dire lorsque l'on travaille seulement un ensemble d'entraînement, un ensemble de validation et un ensemble test. La démarche suivante est automatisée sous le fichier `Analytic Engine.R`[2]. Afin d'arriver à des résultats statistiquement valables, la démarche d'optimisation est effectuée 10 fois pour chaque classificateur, sa méthode d'optimisation respective et l'algorithme de sélection de variables (`SELECT ()`, `SELECT 2()` ou `SELECT 3()`) en question. Il faut noter que l'exemple suivant ne fait pas partie des 10 expériences mentionnées. Cette démarche vise à montrer au lecteur les idées derrière l'optimisation des modèles.

4.2.1 Optimisation de λ et α (régression logistique pénalisée)

Le problème étudié ici nous permet l'utilisation de la recherche par coordonnées sans être limité considérablement par ses inconvénients. En effet, on dispose de seulement deux hyperparamètres ce qui limite le nombre d'évaluations de la fonction objectif imposé par la sonde locale. Aussi, l'ensemble des valeurs que peut prendre l'hyperparamètre α correspond à l'intervalle $[0, 1]$ qui est facilement explorable. Étant donné les supports des hyperparamètres sont différents, deux tailles de pas différentes ont été définies pour chaque hyperparamètre. Cette initiative est tout à fait logique. Supposons que l'on choisisse d'utiliser un seul pas. Pour une grandeur de pas trop grande, les valeurs itérées de α seront fréquemment hors de sa région d'admissibilité ($0 \leq \alpha \leq 1$) limitant ainsi la capacité d'exploration de la sonde locale. Si l'on tente de remédier à ce problème choisissant un petit pas, on augmente le nombre d'itérations nécessaire pour évaluer la fonction objective pour des valeurs élevées de λ.

La librairie `h2o` modélise la régression logistique pénalisée en se basant sur un ensemble d'entraînement et un ensemble de validation. La configuration par défaut utilise l'ensemble de validation pour trouver des valeurs performantes d'hyperparamètres à l'aide d'une méthode de descente par coordonnées. Cependant, le guide d'utilisation mentionne qu'il est plus efficace de trouver une valeur de α avec une recherche par grille et d'utiliser une fonction

2. Le lien URL vers ce fichier est présenté dans l'annexe A

automatique de recherche (de la librairie `h2o`) pour trouver λ. Les trois méthodes suivantes sont employées pour trouver des valeurs d'hyperparamètres : la configuration par défaut de `h2o`, la recherche par coordonnées et la méthode proposée par `h2o`.

La configuration par défaut mène à l'obtention des valeurs suivantes : $\alpha = 0.5$, $\lambda = 0.0934$ et $\text{AUC}_{\text{validation}} = 0.6406$. Pour la recherche par coordonnées, un point de départ $\mathbf{x}_0$ doit d'abord être défini. Une approche intéressante inspirée de méthodes d'optimisation de gradient est de choisir $\mathbf{x}_0$ comme la solution optimale obtenue avec la méthode précédente (ou *warm-starting*, en anglais). Précédemment, il a été mentionné que l'on était en mesure d'explorer le support de α. Par conséquent, plusieurs analyses de sensibilité de α et λ sont effectuées. Ces différentes analyses reposent sur des valeurs de δ_{α_0} (longueur de pas initiale de α) et δ_{α_0} (longueur de pas initiale de λ) différentes. Plus précisément, on aura les palettes de valeurs suivantes : $\wp_{\delta_{\alpha_0}} = \{0.10, 0.15\}$ et $\wp_{\delta_{\lambda_0}} = \{\lambda_0/3, \lambda_0/2\}$. Il y aura quatre tableaux de forme similaire au Tableau 4.3. Il faudra prendre celui qui possède la cellule avec la plus grande AUC. En cas d'égalité, on en sélectionnera un arbitrairement parmi les meilleurs tableaux. Pour chaque recherche par coordonnées, la valeur initiale du paramètre α (c'est-à-dire α_0) sera prise dans la palette de valeurs $\wp_{\alpha_0} = \{0.20, 0.40, 0.60, 0.80\}$ et la valeur initiale du paramètre λ (c'est-à-dire λ_0) sera choisie dans la palette $\wp_{\lambda_0} = \{2^{-5}, 2^{-3}, 2^{-1}, 2^1, 2^3, 2^5\}$.

Tableau 4.3 Analyse de sensibilité de la recherche par coordonnées selon des variations des valeurs initiales des paramètres α et λ pour 15 itérations ($\delta_{\alpha_0} = 0.15$ et $\delta_{\lambda_0} = \lambda_0/3$). Les valeurs contenues dans le tableau correspondent à l'$\text{AUC}_{\text{validation}}$ pour la régression logistique pénalisée.

α_0	λ_0					
	2^{-5}	2^{-3}	2^{-1}	2^1	2^3	2^5
0.20	0.4688	0.7109	**0.7188**	**0.7188**	0.5000	0.5000
0.40	0.6172	**0.7188**	**0.7188**	0.5000	0.5000	0.5000
0.60	0.6172	**0.7188**	0.5000	0.5000	0.5000	0.5000
0.80	0.6250	**0.7188**	0.5000	0.5000	0.5000	0.5000

L'analyse de sensibilité effectuée indique une faible sensibilité à des variations de α_0 (voir Tableau 4.3). Certes, l'examen des colonnes indique que presque trois colonnes (les dernières) ont une variance nulle. Il s'avère que beaucoup de recherches par coordonnées arrivent au même résultat. Cette analyse de sensibilité a été conservée, car elle semblait la meilleure candidate pour l'obtention de la solution optimale.

La meilleure solution donne la solution de départ suivante : $(\alpha_0{}^\star, \lambda_0{}^\star) = (0.20, 2^{-1})$. Elle a été choisie en raison qu'il s'agit de la solution à la somme des hyperparamètres miniminale (selon la norme ℓ_1). Cette règle de décision sera appliquée pour la régression logistique pénalisée et la

MVS. Il faut ensuite analyser le processus lié au itérations de ce point de départ pour obtenir notre solution. La solution obtenue est $(\alpha^\star, \lambda^\star) = (0.403125, 0.16667)$ et donne $\text{AUC}_{\text{validation}} = 0.7188$.

En ce qui a trait à la recommandation de `h2o`, la recherche par grille de α se fera à l'aide de la palette de valeurs $\wp_\alpha^\star = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Les résultats sont présentés au Tableau 4.4. Dans le cas d'une égalité de performance, la solution avec la norme ℓ_1 la plus petite est choisie. On arrive à la conclusion que l'$\text{AUC}_{\text{validation}}$ est maximisée pour $(\alpha^\star, \lambda^\star) = (1.0, 0.6776)$.

Tableau 4.4 Valeurs finales $\alpha^\star$ et $\lambda^\star$ des hyperparamètres pour une recherche par grille de α et une recherche automatisée de λ.

$\alpha^\star$	$\lambda^\star$	$\lvert\alpha^\star\rvert+\lvert\lambda^\star\rvert$	$\text{AUC}_{\text{validation}}$
0	18.855	19.527	0.5000
0.1	**0.6776**	**0.7776**	**0.7109**
0.2	0.2947	0.4947	0.6875
0.3	0.1557	0.4557	0.6406
0.4	0.1168	0.5168	0.6406
0.5	0.09341	0.59341	0.6406
0.6	0.07784	0.67784	0.6406
0.7	0.06672	0.76672	0.6406
0.8	0.06407	0.86407	0.6484
0.9	0.05695	0.95695	0.6484
1.0	0.05126	1.05126	0.6563

En comparant les trois méthodes utilisées, le critère de l'AUC indique que les meilleures estimations des hyperparamètres sont $\alpha^\star = 0.40$ et $\lambda^\star = 0.16797$.

Tableau 4.5 Meilleurs résultats des différentes méthodes de sélection de valeurs d'hyperparamètres.

Méthode	$\alpha^\star$	$\lambda^\star$	$\text{AUC}_{\text{validation}}$
Configuration par défaut de `h2o`	0.5	0.0934	0.6406
Recherche par coordonnées	**0.403125**	**0.16667**	**0.7188**
Recommandation de `h2o`	0.1	0.6776	0.7109

4.2.2 Optimisation de C et γ (machine à vecteurs de support)

Pour cette recherche, les deux noyaux explorés sont le noyau linéaire et le noyau gaussien. D'un côté, le choix du noyau linéaire s'explique principalement par la simplicité de son optimisation. Il suffit d'optimiser C qui pénalise les erreurs de marge et les mauvaises classifica-

tions.[3] De plus, il est recommandé d'employer un noyau linéaire pour un nombre de variables explicatives élevé (Hsu et al., 2016). D'un autre côté, le noyau gaussien est intéressant en raison de sa capacité à traiter des relations non linéaires entre les variables explicatives et la variable de réponse. Il se trouve d'ailleurs que le noyau linéaire est un cas particulier du noyau gaussien (Keerthi et Lin, 2003). Aussi, le noyau sigmoïde agit comme le noyau gaussien pour certaines valeurs associées aux hyperparamètres (Lin et Lin, 2003). Finalement, ce noyau possède moins d'hyperparamètres à optimiser que le noyau polynomial et moins de difficultés numériques peuvent survenir (Hsu et al., 2016).

Pour l'expérimentation avec le noyau linéaire, la démarche employée consiste simplement à choisir une palette de valeurs de coût et à effectuer une recherche par coordonnées pour chacune de ces valeurs. Le choix de ces valeurs est effectué en suivant la recommandation de Hsu et al. (2016), c'est-à-dire en choisissant des valeurs à croissance exponentielle pour les hyperparamètres : $\wp_C = \{2^{-5}, 2^{-3}, 2^{-1}, ..., 2^5\}$.

Hsu et al. (2016) recommande d'effectuer une recherche par grille couvrant une grande région et utilisant peu de points d'évaluation initiale. Après l'identification d'une région optimale, une recherche plus fine doit être effectuée sur cette région. On peut s'apercevoir d'une similarité avec notre méthode proposée qui consiste en des recherches par coordonnées sur les couples de valeurs (C, γ) obtenus par une recherche par grille. Notre méthode a le désavantage d'être plus coûteuse, car une recherche locale est effectuée pour chaque point. Les palettes de valeurs choisies sont : $\wp_C = \{2^{-2}, 2^0, 2^2, 2^4, 2^6\}$ et $\wp_\gamma = \{2^{-5}, 2^{-3}, 2^{-1}, 2^1, 2^3, 2^5\}$.

Tableau 4.6 Analyse de sensibilité de la recherche par coordonnées selon des variations des valeurs initiales des paramètres C_0 et γ_0 pour 15 itérations ($\delta_{C_0} = {}^{C_0}/_3$ et $\delta_{\gamma_0} = {}^{\gamma_0}/_3$). Les valeurs contenues dans le tableau correspondent à l'$\text{AUC}_{\text{validation}}$ pour un MVS à noyau gaussien.

C_0	γ_0					
	2^{-5}	2^{-3}	2^{-1}	2^1	2^3	2^5
2^{-2}	**0.8594**	**0.8594**	0.7383	0.5	0.5	0.5
2^0	0.8437	0.7031	0.7188	0.5	0.5	0.5
2^2	0.8359	0.7031	0.7383	0.5	0.5	0.5
2^4	0.8438	0.8438	0.7188	0.5	0.5	0.5
2^6	0.8438	0.8438	0.7188	0.5	0.5	0.5

De façon similaire à ce qui a été fait pour la régression logistique pénalisée, diverses valeurs de pas δ_{C_0} et δ_{γ_0} (pour la recherche par coordonnées) ont été explorées pour le noyau gaussien.

3. La recherche par coordonnées pour un seul hyperparamètre peut être effectuée, et ce, à un coût de calcul bas.

Plus précisément, on avait les palettes de valeurs suivantes pour les pas :$\wp_{\delta_{C_0}} = \{{}^{C_0}/_2, {}^{C_0}/_3\}$ et $\wp_{\delta_{\gamma_0}} = \{{}^{\gamma_0}/_2, {}^{\gamma_0}/_3\}$.

Pour le noyau linéaire, divers pas ont aussi été expérimentés $\wp_{\delta_{C_0}} = \{{}^{C_0}/_2, {}^{C_0}/_3\}$. Les matrices de sensibilité obtenues étaient les mêmes. Le Tableau 4.9 montre ces résultats. En comparant les résultats du noyau linéaire et du noyau gaussien, on conclut qu'il est préférable d'opter pour le noyau linéaire en raison de sa performance selon l'$\text{AUC}_{\text{validation}}$.

Tableau 4.7 Analyse de sensibilité de la recherche par coordonnées selon des variations du paramètre C_0. Les valeurs du AUC de validation sont calculées pour 15 itérations ($\delta_{C_0} = {}^{C_0}/_2$).

C_0	$\text{AUC}_{\text{validation}}$
2^{-5}	**0.8281**
2^{-3}	0.8281
2^{-1}	0.8281
2^{1}	0.8281
2^{3}	0.8281
2^{5}	0.8281

Tableau 4.8 Meilleurs résultats des différentes méthodes de sélection de valeurs d'hyperparamètres.

Méthode	$C^\star$	$\gamma^\star$	$\text{AUC}_{\text{validation}}$
Recherche par coordonnées : noyau linéaire	0.03125	–	0.8281
Recherche par coordonnées : noyau gaussien	**0.25**	**0.01367**	**0.8594**

4.2.3 Optimisation des hyperparamètres (machine à *gradient boosting*)

Les hyperparamètres à optimiser sont le taux d'apprentissage, le nombre d'arbres et la profondeur maximale des arbres. Il serait tentant d'implémenter une fonction de recherche par coordonnées à trois hyperparamètres pour l'optimisation de la machine à *gradient boosting*. Cependant, la présentation des analyses de sensibilité serait complexe et lourde. En effet, il faudrait présenter des tableaux selon les variations de trois variables. Cela correspond à des tableaux à trois dimensions. De plus, la méthode serait très coûteuse en calculs. Par conséquent, les méthodes d'optimisation préconisées sont : la recherche par grille, la recherche automatique de la librairie `h2o` et une recherche aléatoire.

Premièrement, appliquons la recherche par grille. Les palettes de valeurs sont :

— $\wp_{TA} = \{0.01, 0.05, 0.1, 0.15, 0.25\}$ (taux d'apprentissage),

— $\wp_{NA} = \{25, 50, 75, 100, 125\}$ (nombre d'arbres),
— $\wp_{PA} = \{4, 5, 6\}$ (profondeur maximale des arbres)

Il y aura donc au total 75 modèles à construire. En se fiant au critère de l'$\text{AUC}_{\text{validation}}$, il semble que les meilleures valeurs pour les hyperparamètres correspondent à un taux d'apprentissage de 0.25, a 50 arbres et a une profondeur d'arbres maximale de 4. La valeur de l'AUC de l'ensemble de validation est alors de 0.8906.

Deuxièmement, utilisons la recherche automatique configurée par la librairie `h2o`. Le taux d'apprentissage obtenu est de 0.1, le nombre d'arbres est de 50 et la profondeur maximale des arbres est de 5. L'$\text{AUC}_{\text{validation}}$ est de 0.8594.

Finalement, la recherche aléatoire est effectuée. De façon similaire à la recherche par grille, une plage de valeurs doit être choisie pour chaque hyperparamètre. La différence majeure est que pour cet algorithme, certaines des valeurs aléatoires seront prises dans les plages, contrairement à la recherche par grille où chaque combinaison doit être explorée. Par conséquent, il serait avantageux d'élargir l'étendue de chacune des palettes précédentes. On obtient alors que les palettes de valeurs sont $\wp_{TA} = \{0.001, 0.006, ..., 0.101\}$, $\wp_{NA} = \{25, 30, .., 175\}$ et $\wp_{PA} = \{1, 2, ..., 10\}$. Un choix de nombre d'itérations s'impose. Par conséquent, 75 itérations sont effectuées. L'expérimentation mène à l'obtention d'un modèle ayant une $\text{AUC}_{\text{validation}}$ de 0.8906. Celui-ci est composé de 65 arbres, avec une profondeur maximale de 7 et un taux d'apprentissage de 0.101. Il se trouve que la recherche par grille et la recommandation de `h2o`

Tableau 4.9 Meilleur résultat des différentes méthodes de sélection de valeurs d'hyperparamètres pour la MGB.

Méthode	Nombre d'arbres	Profondeur	Taux	$\text{AUC}_{\text{validation}}$
Recherche par grille	**50**	**4**	**0.25**	**0.8906**
Configuration par défaut de `h2o`	50	5	0.10	0.8594
Recommandation de `h2o`	65	7	0.101	0.8906

mènent au même $\text{AUC}_{\text{validation}}$. Également, les matrices de confusion respectives de chaque méthode sont les mêmes. Il se trouve qu'il est préférable d'avoir un nombre peu élevé d'arbres pour éviter le surrapprentissage. Par conséquent, on choisit le modèle associé à la recherche par grille.

4.3 Résultats (pour une expérience)

4.3.1 Régression logistique pénalisée (Gravier, 2010)

En appliquant la solution trouvée par la recherche par coordonnées, on obtient $\text{AUC}_{\text{test}} = 0.6444$ et la matrice de confusion est présentée comme étant le Tableau 4.10. La précision obtenue correspond à 57.14%.

Tableau 4.10 Matrice de confusion de la classification du cancer du sein pour la régression logistique pénalisée.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	8	10
$y = 1$	2	8

4.3.2 Machine de support à vecteurs (Gravier, 2010)

Le modèle préconisé est la machine à vecteurs de support à noyau linéaire avec une valeur de coût $C = 0.03125$. On obtient $\text{AUC}_{\text{test}} = 0.8222$ et la matrice de confusion est présentée comme étant le Tableau 4.11. La précision obtenue correspond à 67.86%.

Tableau 4.11 Matrice de confusion de la classification du cancer du sein pour le modèle basé sur la machine de support à vecteurs avec $(C^\star, \gamma^\star) = (0.25, 0.0001627604)$.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	15	3
$y = 1$	6	4

4.3.3 Machine à *gradient boosting* (Gravier, 2010)

On obtient $\text{AUC}_{\text{test}} = 0.8278$ et la matrice de confusion est présentée comme étant la Tableau 4.12. La précision obtenue correspond à 82.14%.

Tableau 4.12 Matrice de confusion de la classification du cancer du sein pour la machine à *gradient boosting*.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	17	1
$y = 1$	4	6

4.4 Généralisation de l'optimisation des hyperparamètres

Pour chaque classificateur, on dispose de différentes méthodes d'optimisation des hyperparamètres. Or, le choix d'une méthode selon une seule valeur d'AUC est douteux. La dominance d'une méthode pourrait simplement être due au hasard de l'échantillonnage initial [4]. C'est pour cette raison que chaque méthode d'optimisation est testée sur dix échantillonnages (voir le Tableau 4.13).

Tableau 4.13 Représentation partielle des résultats de la régression logistique pénalisée après l'application de l'algorithme `SELECT()` et de la recherche par coordonnées.

Expérience	α	γ	$\text{AUC}_{\text{validation}}$
1	0.95	0.05468	0.7109
2	0.90	0.0729	0.9531
3	0.55	0.2188	0.8750
...	...	...	...
8	0.05	0.0417	0.7578
9	0.62	0.1855	0.5938
10	0.20	0.6875	0.6719

Il faut se souvenir que notre objectif de recherche se décompose en deux parties. En premier lieu, on veut tenter de voir si les modifications de `SELECT()` mènent à des meilleures performances pour les classificateurs. En deuxième lieu, il serait souhaitable d'obtenir des résultats (AUC et précision) meilleurs que ceux fournis par la littérature.

La solution consiste tout simplement à employer l'ANOVA à blocs complets. Les blocs correspondent à des chiffres associés à chacune des dix expériences. Le facteur correspond à la méthode d'optimisation utilisée (voir Figure 5.6). Si les résultats sont suffisamment différents, il devrait ressortir une moyenne significativement différente des autres. L'ANOVA nous permet de déceler la présence de cette moyenne différente. Cependant, il faut utiliser des comparaisons multiples pour savoir laquelle est la plus grande. Puisque l'on effectue plusieurs tests d'hypothèses, alors il faudra modifier notre seuil de significativité à l'aide de la méthode de Bonferonni. Brièvement, cette méthode consiste à diviser notre seuil de significativité par le nombre de tests effectués. On emploiera des intervalles de Tukey. Dans l'éventualité où l'ANOVA révèle que la méthode n'est pas un facteur significatif ou que les intervalles de Tukey sont difficiles à interpréter, on prendra la méthode ayant la moyenne de traitement la plus élevée, faute de mieux.

4. Cet échantillonnage correspond à la façon aléatoire dont un décompose le jeu de données initial (entraînement, validation et test).

Tableau 4.14 Tableau raccourci à utiliser pour l'ANOVA à blocs complets pour la régression logistique pénalisée après l'application de l'algorithme `SELECT()` pour l'ensemble de validation (pd : par défaut, r : recommandation et rpc : recherche par coordonnées).

Méthode	Bloc						
	1	2	3	...	8	9	10
pd	0.6875	0.8906	0.7031	...	0.5703	0.5313	0.6484
r	0.7031	0.9531	0.7266	...	0.5703	0.5781	0.6563
rpc	0.7109	0.9531	0.8750	...	0.7578	0.5938	0.6719

4.5 Détermination des meilleurs modèles

On obtient différentes tables dans lesquelles on regardera la significativité du facteur méthode. Une valeur-p inférieure au seuil de signification ($\alpha^\star = \frac{\alpha}{3} \approx 0.01666$) indique que la méthode d'optimisation des hyperparamètres est un facteur statistiquement significatif. En consultant le Tableau 4.15, il semble donc y avoir une différence significative entre les moyennes des méthodes pour la régression logistique pénalisée et la machine à *Gradient Boosting*.

Tableau 4.15 Meilleures méthodes d'optimisations selon les classificateurs.

	Classificateur		
	RLP	MVS	MGB
Valeurs-p	**0.002799**	0.20576	$\mathbf{5.385 \times 10^{-7}}$
Meilleure moyenne	0.7335	0.7953	0.7672
Meilleure méthode	rpc	nl	ra

Pour la régression logistique pénalisée, l'ANOVA à blocs complets a mise en évidence la supériorité de la recherche par coordonnées par rapport (rpc) à la configuration par défaut (pd). En effet, une valeur-p de 0.0026638 a été obtenue lors des comparaisons multiples, ce qui est inférieur à notre seuil de significativité $\alpha^\star = 0.01666$. Cependant, l'ANOVA à blocs complets n'a pas été en mesure de montrer que rpc est meilleure que la recommandation de `h2o` (r). Effectuer la comparaison multiple est donc inutile ici. Malgré le fait que rpc ne dépasse pas les deux autres méthodes, on la choisira, car la moyenne de celle-ci est supérieure aux autres.

Pour la machine à vecteurs de support, l'ANOVA à bloc complets n'indique pas une différence entre les moyennes des méthodes. Par conséquent, la comparaison multiple est inutile. Par le raisonnement mentionné précédemment, on trouve que la meilleure méthode d'optimisation semble être celle basée sur le noyau linéaire.

Pour la machine à *Gradient Boosting*, l'ANOVA à bloc complets indique une différence entre

les moyennes des méthodes (valeur-p $= 5.385 \times 10^{-7} < 0.01666 = \alpha^\star$). La méthode avec la plus grande moyenne est la recherche aléatoire. Les comparaisons multiples montrent que la recherche aléatoire (ra) et la recherche par grille (rpg) surpassent la configuration par défaut. Cependant, il n'est pas possible de déterminer statistiquement laquelle de ces méthodes a une meilleure moyenne (valeur-p $= 0.3482698 > 0.01666 = \alpha^\star$).

Une analyse des résidus a été faite et il ne semble pas y avoir de problèmes en lien avec la normalité des résidus.

4.6 Résultats (pour plusieurs expériences)

Les résultats obtenus sur l'ensemble de test sont exhibés dans le Tableau 4.16. Le temps moyen d'exécution de `SELECT 2()` est de 0.1181 minute avec une erreur type de 0.0024 minute (voir le Tableau B.7 pour plus de détails). Cette estimation est basée sur 10 observations.

Tableau 4.16 Résultats des prédictions selon les meilleurs classificateurs pour `SELECT()`.

	Classificateur		
	RLP	MVS	MGB
Méthode	Recherche par coordonnées	Noyau gaussien	Recherche aléatoire
$\text{AUC}_{\text{moyenne}}$	0.7206	0.7672	0.7347
Erreur standard (AUC)	0.0676	0.0416	0.0334
$\text{Précision}_{\text{moyenne}}$	0.6786	0.7179	0.7107
Erreur standard (précision)	0.0734	0.0299	0.0447

CHAPITRE 5 MODIFICATIONS DE L'ALGORITHME SELECT()

5.1 Première version modifiée de l'algorithme (SELECT 2())

Comme il a été mentionné, l'originalité de cette recherche provient de modifications apportées à l'algorithme SELECT(). La première modification consiste à changer l'analyse en composantes principales (ACP) par une analyse en composantes principales creuse (ACPC). Ce changement fait en sorte que l'expérimentateur n'a pas à choisir un seuil μ (voir l'étape 3 de SELECT()) pour arrondir à zéro des poids de la matrice de rotation $\mathbf{V}^{(2)}$. L'arrondissement est pratique, mais peut mener à une mauvaise identification des variables importantes (Zou et al., 2004). L'espoir issu de ce changement est d'obtenir des zéros dans $\mathbf{V}^{(2)}$ de façon plus naturelle et que les variables sélectionnées par la suite révèleront plus d'informations aux classificateurs. Également, ce changement de représentation par l'ACPC pourrait mener à une meilleure sélection de variables par la régression logistique bayésienne. Cependant, il n'est pas possible de déterminer si c'est la cas. Un des côtés intéressants de ce changement est le fait qu'il affecte simplement à l'étape 1 sans compliquer le reste de l'algorithme. Malheureusement, le prix à payer est le choix d'un vecteur $\boldsymbol{\lambda}$ de pénalités. La démarche suivante suggère une façon de choisir ce vecteur.

5.1.1 Application des étapes de la méthode

La version modifiée de l'algorithme est donc la suivante :

Algorithme 8 : SELECT 2() trouve le sous-ensemble de gènes à utiliser pour la classification et retourne l'analyse en composantes principales de ces gènes.

Entrées : $\mathcal{T}, v \in \mathbb{N}, k \in \mathbb{N}, \boldsymbol{\lambda} \in \mathbb{R}_+^k$
Sorti : $\mathbf{Q}_2$, l'ensemble des composantes principales suite à la sélection de variables.
1 $\mathbf{Q} \leftarrow \texttt{ACPC}(\{\mathbf{x}_i\}_{i=1}^n, v, k, \boldsymbol{\lambda})$
2 $\mathbf{Q}_1 \leftarrow \texttt{LOGIT}(\mathbf{Q}, d)$
3 $\{\mathbf{z}_i\}_{i=1}^n \leftarrow \texttt{EnleverAttributs}(\mathbf{Q}_1)$
4 $\mathbf{Q}_2 \leftarrow \texttt{ACP}(\{\mathbf{z}_i\}_{i=1}^n, q_2)$
5 **Retourner** $\mathbf{Q}_2$

Il se trouve qu'il n'y a plus le seuil u (à l'étape 3) et les paramètres k et $\boldsymbol{\lambda}$ ont été rajoutés à l'étape 1. Il s'agit respectivement du nombre de composantes principales à calculer et du vecteur de pénalités sur les composantes principales.

Étape 1 : Q $\leftarrow ACPC(\{\mathbf{x}_i\}_{i=1}^n, v)$

Rappelons-nous que l'ensemble d'entraînement utilisé est composé de 116 observations et de 2905 variables explicatives. L'ACPC s'effectue sur le logiciel R en utilisant la fonction `arrayspc` issue de la librairie `elasticnet`. La commande a la forme suivante :

```
arrayspc(x,K,para,use.corr,max.iter,trace,eps)
```

- `x` : La matrice des gènes (ou matrice de corrélation).
- `K` : Le nombre de composantes principales à calculer ($1 \leq$ `K` $\leq n$).
- `para` : Le vecteur des pénalités $\{\lambda_j\}$ (ou $\boldsymbol{\lambda}$, taille du vecteur : k).
- `use.corr` : Argument logique sur l'utilisation de la matrice de corrélation.
- `max.iter` : Nombre d'itérations maximal.

Parmi ces arguments, la détermination des suivants reposeront sur nous : `x` (ou $\mathbf{X}_{\text{entraînement}}$), `K` (ou k) et `para` (ou $\boldsymbol{\lambda}$). Déterminer $\mathbf{X}_{\text{entraînement}}$ est trivial. Il s'agit simplement de l'ensemble d'entraînement en excluant la variable de réponse. Pour k, on calculera le nombre maximal de composantes, c'est-à-dire le nombre de composantes de l'ACP. Le seul prix à payer est un temps d'attente plus long. Cependant, déterminer $\boldsymbol{\lambda}$ n'est pas évident. Il se trouve que ce vecteur peut être construit d'une infinité de façons différentes.

Une solution pour limiter ces choix est tout simplement de considérer les contraintes de ce problème. Premièrement, étant donné que l'on ne dispose pas de connaissances pour nous aider à choisir les poids des multiples pénalités λ_j (où $j \in \{1, 2, ..., k\}$), on décide que toutes les pénalités sont égales. Mathématiquement, on dira que $\lambda_j = c \geq 0 \quad \forall j \in \{1, 2, ..., k\}$. La première considération est le pourcentage de variance expliquée par nos composantes. Dans la première étape de la reconstitution de `SELECT()`, il a été décidé de garder le minimum de composantes principales expliquant 90% de la variance. Or, il se trouve que la proportion de la variance totale expliquée pour l'ACPC ne somme pas à un (sauf lorsque $\boldsymbol{\lambda} = 0$, ce cas correspond à l'ACP). Les poids de la pénalité affectent la proportion totale de la variance expliquée. Plus précisément, cette proportion diminue à mesure que la pénalité augmente (voir Figure 5.1). Donc, le prix pour avoir des éléments des composantes principales creuses est une perte d'une partie de la proportion totale de la variance expliquée.

En examinant la Figure 5.1, on peut constater que certaines courbes n'atteindront jamais le seuil de 90% que l'on souhaite atteindre. Un cas particulier de ce comportement est lorsque $\boldsymbol{\lambda} = (0.25, 0.25, ..., 0.25)^\top$ (voir la courbe rose à la Figure 5.1). Par conséquent, il existe une valeur de pénalité $p_s > 0$ telle que la proportion de variance totale expliquée est supérieure ou égale à 90% pour tout $\boldsymbol{\lambda} \in [\mathbf{0}, p_s \cdot \mathbb{1}]$. L'idée est d'estimer p_s. Puisque l'objectif global n'est pas

d'obtenir une solution exacte de p_s, alors on peut se contenter d'une approximation obtenue analytiquement. En utilisant la méthode de recherche par coordonnées (pour 15 itérations), on obtient l'estimation $\hat{p}_s = 0.2276367$. Il est important de noter que la fonction de coût est la proportion de variance expliquée et les contraintes sont de garder ce seuil au-dessus de 90% et avoir une pénalité positive.

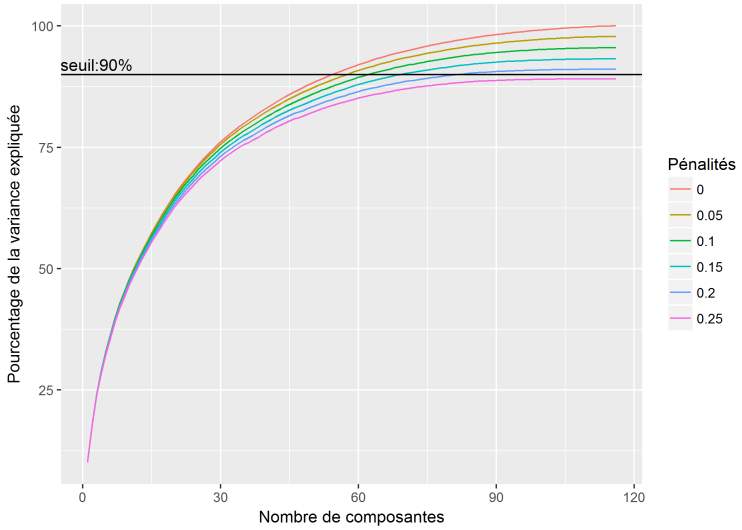


Figure 5.1 Pourcentage de la variance expliquée selon les différentes pénalités pour l'ACPC.

On disposera alors d'un intervalle de valeurs dans lequel il faudra choisir une valeur de pénalité. Or, il est important de se rappeler du rôle de l'estimation de μ dans l'algorithme original. Le but était d'obtenir une matrice creuse, c'est-à-dire une matrice composée de beaucoup de zéros. Afin d'évaluer à quel point la matrice $\mathbf{V}^{(2)}$ est creuse, la mesure suivante peut être utilisée :

$$\text{Sp}(\mathbf{V}^{(2)}) = \frac{\text{Nombre de zéros dans } \mathbf{V}^{(2)}}{\text{Nombre d'éléments dans } \mathbf{V}^{(2)}} \tag{5.1}$$

où $\mathbf{V}^{(2)}$ après la deuxième étape de l'algorithme. Cette mesure représente la proportion de zéro dans la matrice en question. Une valeur près de 1 signifie que la matrice comporte beaucoup de zéro. Une valeur proche de zéro signifie que la matrice comporte très peu de zéros. Précédemment, la valeur obtenue était de 90.35%. Afin de se rapprocher de la méthodologie précédente, il faudrait trouver une valeur de μ qui est telle que $\text{Sp}(\mathbf{V}^{\star}) \approx 90\%$.

Encore une fois, on emploiera la recherche par coordonnées. Il se trouve la valeur de 90.14% est obtenue à l'étape 3 dans `SELECT()`. Heureusement, l'étape 2 peut être automatisée. Le

problème d'optimisation est alors le suivant :

$$\min_{0 \leq p_s \leq 0.2276367} \quad |\text{Sp}(\mathbf{V}^\star) - 90.35| \tag{5.2}$$

La solution de l'expression 5.2 obtenue avec la recherche par coordonnées (pour 15 itérations) est $\hat{p}_s = 0.2275391$. Divers points de départ p_{s_0} ont été essayés et tous semblent converger même la même valeur (voir Tableau 5.1).

Tableau 5.1 Estimation de $\hat{p}_s$ à partir de divers points de départ.

p_{s_0}	$\delta_{p_{s_0}}$	Nombre d'itérations	$\hat{p}_s$	$\lvert\text{Sp}(\mathbf{V}^\star) - 0.9014\rvert$
0.05	0.05	10	0.2265625	0.4487345
0.10	0.05	10	0.2275391	0.4487345
0.15	0.05	10	0.2273437	0.4306429
0.20	0.05	10	0.2273438	0.4306429

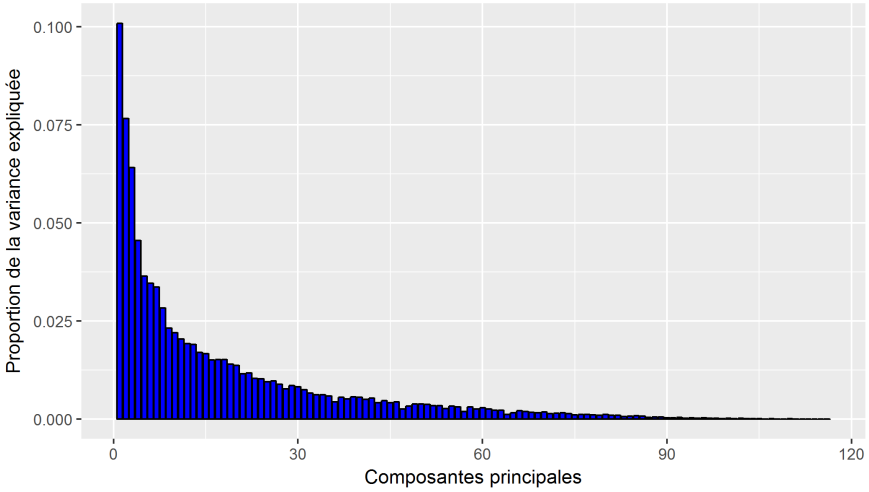


Figure 5.2 Pourcentage de la variance expliquée selon les composantes principales pour l'ACPC.

Étape 2 : $\mathbf{Q}_1 \leftarrow LOGIT(\mathbf{Q}, d)$

L'étape 2 est similaire à celle effectuée dans l'algorithme `SELECT()`. La sélection de variables est normalement basée sur la régression logistique. Pour contourner des problèmes potentiels, on utilise une distribution a priori faiblement informative afin de régler ce problème. Cette a priori n'est pas hiérarchique. L'option par défaut est une distribution de Cauchy centrée en zéro avec une échelle de 2.5. L'application de cette régression mène alors à l'obtention d'un

modèle dont on possède les coefficients et les erreurs types. La stratégie consiste à regarder la magnitude des coefficients de $\boldsymbol{\beta}$ pour sélectionner des variables.

On choisira $d = 100$, ce qui correspond à conserver environ 90% des composantes. Il se trouve que la valeur de 90% a été choisie afin de suivre la méthodologie de `SELECT()`.

Étape 3 : $\{\mathbf{z}_i\}_{i=1}^n \leftarrow EnleverAttributs(\mathbf{Q}_1)$

En se débarrassant de certaines composantes, la matrice des poids $\mathbf{V}$ devient plus petite. On conserve les colonnes correspondant à chacune des 103 composantes expliquant le plus la variance des données. Chaque ligne de cette matrice correspond aux poids d'un gène sur les composantes. Mathématiquement, l'élément v_{ij} correspond au poids du gène i pour la composante de la $j^{\text{ième}}$ colonne de $\mathbf{V}$.

Il n'est pas nécessaire d'effectuer un choix de μ, car l'analyse en composantes principales creuse rend la matrice de poids plus creuse. Dans notre situation, le choix de pénalité mène à l'obtention de la matrice $\mathbf{V}^{(3)}$ composée de 47.5% de zéros.

Une fois cette opération effectuée, il faut regarder le nombre de poids non nul lié à chaque gène. La procédure de l'article étudié propose de conserver les gènes ayant le plus grand nombre de poids. Par conséquent, il semble qu'un gène ayant un grand nombre de poids non nuls apporte une plus grande contribution au modèle.

Pour des raisons explicitées dans le dernier chapitre, on souhaiterait garder entre 172 et 395 gènes. On souhaite conserver le maximum d'information en conservant moins de 395 gènes. On conservera alors seulement les gènes ayant au moins 58 poids sur les composantes pour garder le maximum de gènes. Cela mène à la sélection de 304 variables dans l'ensemble de données originales.

Tableau 5.2 Nombre de gènes sélectionnés selon le nombre de poids non nuls minimal souhaité.

	i						
	...	56	57	58	59	60	...
Nombre de gènes avec plus de i poids non nuls	...	516	400	**304**	220	163	...

Étape 4 : $\mathbf{Q}_2 \leftarrow ACP(\{\mathbf{z}_i\}_{i=1}^n, q_2)$

La dernière étape consiste simplement à appliquer l'analyse en composantes principales sur notre nouvel ensemble de données, c'est-à-dire l'ensemble d'entraînement réduit. Ce nouvel ensemble de données est composé de 116 lignes et de 117 colonnes.

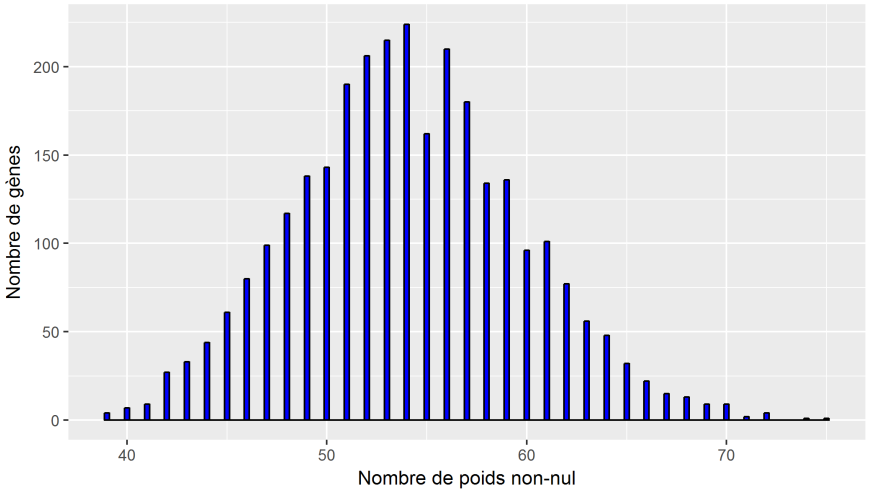


Figure 5.3 Histogramme de la fréquence des gènes selon le nombre de poids non nul pour l'ensemble de données d'entraînement.

5.2 Détermination des meilleurs modèles

On se rappelle que l'on choisit la méthode d'optimisation de chaque classificateur selon les résultats obtenus de l'ANOVA par blocs complets et les moyennes par méthode. C'est la significativité du facteur méthode qui nous intéresse. Une valeur-p inférieure au seuil de signification ($\alpha^\star = \frac{\alpha}{3} \approx 0.01666$) indique que la méthode d'optimisation des hyperparamètres est un facteur statistiquement significatif. Parmi les classificateurs, il semble donc y avoir une différence significative entre les moyennes des méthodes pour la régression logistique pénalisée et la machine à *gradient boosting* (voir Tableau 5.3).

Tableau 5.3 Meilleures méthodes d'optimisation selon les classificateurs.

	Classificateur		
	RLP	MVS	MGB
Valeur-p	11.55×10^{-5}	0.9290	$\mathbf{1.233 \times 10^{-5}}$
Meilleure moyenne	0.7867	0.7684	0.7679
Meilleure méthode	rpc	ng	rpg

Pour la régression logistique pénalisée, la valeur-p l'ANOVA à blocs complets (voir Tableau 5.3) indique qu'il y a une différence significative entre les moyennes des méthodes. En effectuant la comparaison multiple, on constate la supériorité de la recherche par coordonnées (rpc) par rapport à la configuration par défaut (pd) et la recommandation de `h2o` (r). En effet, des valeurs-p respectives de 0.0002616 (rpc vs pd) et 0.0005289 (rpc vs r) sont obtenues. Ces valeurs sont inférieures à notre seuil de significativité $\alpha^\star = 0.01666$.

Pour la machine à vecteurs de support, l'ANOVA à blocs complets n'indique pas une différence entre les moyennes des méthodes. La comparaison multiple est donc inutile. En raison de sa meilleure performance moyenne sur l'ensemble de validation, on supposera que la meilleure méthode d'optimisation est celle basée sur le noyau linéaire.

Pour la machine à *gradient boosting*, l'ANOVA à blocs complets indique une différence significative entre les moyennes des méthodes (valeur-p $= 1.233 \times 10^{-7} < 0.01666 = \alpha^\star$). La méthode avec la plus grande moyenne est la recherche par grille. Les comparaisons multiples montrent que la recherche aléatoire (ra) et la recherche par grille (rpg) surpassent la configuration par défaut. Cependant, il n'est pas possible de déterminer statistiquement laquelle de ces deux méthodes a une meilleure moyenne en raison de la valeur-p obtenue (valeur-p $= 0.3482698 > 0.01666 = \alpha^\star$). On choisira la recherche par grille en raison de sa meilleure performance moyenne.

Une analyse des résidus a été faite et il ne semble pas y avoir de problèmes en lien avec la normalité des résidus. Le temps moyen d'exécution de `SELECT 2()` est de 4584.70 minutes avec une erreur type de 118.69 minutes (voir le Tableau B.7 pour plus de détails). Cette estimation est basée sur 10 observations.

Tableau 5.4 Résultats des prédictions selon les meilleurs classificateurs pour `SELECT 2()`.

	Classificateur		
	RLP	MVS	MGB
Méthode	Recherche par coordonnées	Noyau linéaire	Recherche par grille
$\text{AUC}_{\text{moyenne}}$	0.7277	0.7927	0.6705
Erreur standard (AUC)	0.0212	0.0301	0.0374
$\text{Précision}_{\text{moyenne}}$	0.7250	0.7536	0.6964
Erreur standard (précision)	0.0287	0.0279	0.0245

5.3 Seconde version modifiée de l'algorithme (`SELECT 3()`)

Afin de créer une deuxième version modifiée de `SELECT()`, une autre modification importante peut être apportée à cet algorithme. Il suffit d'entraîner le modèle de régression logistique et de sélectionner les d composantes dont le coefficient β est supérieur à un certain seuil. Malheureusement, la méthodologie ne précise pas comment choisir ce seuil. Il a été décidé de garder 90% des composantes avec la plus grande importance. Puis, l'une des modifications présentes dans `SELECT 2()` est une sélection de variables à l'aide des statistiques t. Il se trouve que la sélection de variables est basée sur un modèle linéaire, mais il existe de multiples façons

de sélectionner des variables. Il pourrait être instructif d'employer une autre méthode. Il serait souhaitable d'avoir une procédure objective permettant de faire la sélection des composantes sans avoir à décider du nombre de composantes voulues. Pour ce faire, l'algorithme `Boruta` est utilisé.

On se rappelle qu'il s'agit d'une méthode de sélection de variables de type enveloppe utilisant une forêt d'arbres aléatoire. La nature différente de cette approche pourrait mener à la sélection d'autres composantes potentiellement informatives. Sans compter que l'on ne fait plus face à la nécessité de choisir la valeur d à l'étape 2.

5.3.1 Application des étapes de la méthode

La version modifiée de l'algorithme est donc la suivante :

Algorithme 9 : `SELECT 3()` trouve le sous-ensemble de gènes à utiliser pour la classification et retourne l'analyse en composantes principales de ces gènes.

Entrées : $\mathcal{T}, v \in \mathbb{N}, k \in \mathbb{N}, \boldsymbol{\lambda} \in \mathbb{R}_+^k$
Sorti : $\mathbf{Q}_2$, l'ensemble des composantes principales suite à la sélection de variables.
1 $\mathbf{Q} \leftarrow \texttt{ACP}(\{\mathbf{x}_i\}_{i=1}^n, v)$
2 $\mathbf{Q}_1 \leftarrow \texttt{Boruta}(\mathbf{Q})$
3 $\{\mathbf{z}_i\}_{i=1}^n \leftarrow \texttt{EnleverAttributs}(\mathbf{Q}_1, \mu)$
4 $\mathbf{Q}_2 \leftarrow \texttt{ACP}(\{\mathbf{z}_i\}_{i=1}^n, q_2)$
5 **Retourner** $\mathbf{Q}_2$

Étape 1 : $\mathbf{Q} \leftarrow ACP(\{\mathbf{x}_i\}_{i=1}^n, v)$

Cette étape est exactement la même que celle pratiquée pour l'algorithme `SELECT()`. La projection des données vers les composantes principales donne une matrice composée de 116 lignes et de 116 colonnes.

Le seuil $v = 90$ nous amène à conserver les variables conservant 90% de la variance expliquée. Concrètement, cela revient à garder les 55 premières composantes principales.

Étape 2 : $\mathbf{Q}_1 \leftarrow Boruta(\mathbf{Q})$

Pour la sélection de variables avec une forêt d'arbres aléatoires, le logiciel R est utilisé. Plus particulièrement, on utilise la fonction `Boruta` issue de la librairie `Boruta`. La fonction a la forme suivante :

`Boruta(x,y,data,maxRuns)`

- `x` : La matrice d'expérience.
- `y` : La variable de réponse.
- `maxRuns` : Nombre maximum d'itérations.

L'application de cette méthode détermine l'importance des composantes principales selon l'une des trois catégories suivantes : acceptée, refusée et non déterminée. Pour le nombre d'itérations, on choisira d'effectuer 2000 itérations. Ce grand nombre s'explique par le fait que `Boruta` est indécis face à l'importance de certaines composantes. En augmentant le nombre d'itérations, cette indécision peut être amenuisée jusqu'à une certaine limite.

Les arguments suivants doivent être déterminés : `x`, `y` et `maxRuns`. Déterminer `x` et `y` est très simple. Il s'agit simplement de la matrice des composantes principales conservées et le vecteur de la variable de réponse pour l'ensemble d'entraînement. Le paramètre `maxRuns` correspond simplement au nombre d'itérations maximales employé pour `Boruta`. Le niveau de confiance est laissé au seuil par défaut, c'est-à-dire 0.01. L'exécution de l'algorithme `Boruta` mène aux résultats suivants :

- 12 composantes sont importantes : PC13, PC15, PC17, PC18,...,PC2 ;
- 42 composantes ne sont pas importantes : PC1, PC10, PC11, PC12,...,PC14 ;
- Une composante a une importance indéterminée : PC31

Un graphique peut être tracé pour avoir un aperçu de l'importance des variables (voir Figure 5.4). Heureusement, une correction peut être appliquée pour trouver la catégorie associée à chacune des variables à l'importance indéterminée. Il en résulte que la $31^{\text{ième}}$ composante n'est pas importante.

Étape 3 : $\{\mathbf{z}_i\}_{i=1}^{n} \leftarrow \mathit{EnleverAttributs}(\mathbf{Q}_1, \mu)$

En se débarrassant de certaines composantes, on aboutit à la matrice $\mathbf{V}^{(2)}$ qui est une réduction de la matrice des poids $\mathbf{V}$. Certes, seules les colonnes correspondant à chacune des 12 composantes les plus importantes sont conservées.

Plus précisément, on aboutit à la matrice $\mathbf{V}^{(2)}_{2905\times 12}$. Ensuite, il faut déterminer une valeur de μ telle que $\mathrm{Sp}(\mathbf{V}^{(3)}) \approx 0.90\%$. On se rappelle que plus la valeur de μ est grande, plus les éléments de la matrice $\mathbf{V}^{(2)}$ sont arrondis à zéro. Il se trouve que pour $\mu = 0.03$, environ 90.5% de la matrice $\mathbf{V}^{(3)}$ est composée de zéros.

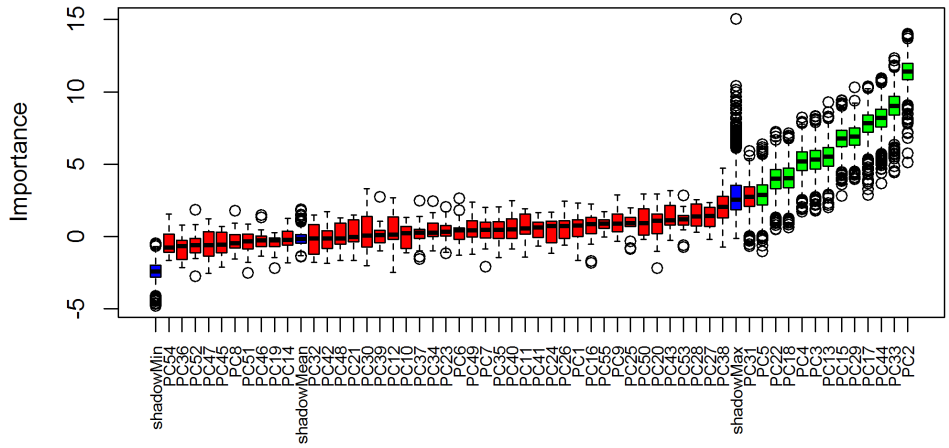


Figure 5.4 Graphique de l'importance des diverses composantes après la correction (rouge : non important, vert : important, bleu : minimum des imitations et maximum des imitations).

Il faut maintenant sélectionner les gènes ayant le plus de poids non nul dans la matrice $\mathbf{V}^{(3)}$. Pour des raisons mentionnées dans le chapitre précédent, on voudrait garder entre 172 et 395 gènes de notre ensemble de données. En consultant le Tableau 5.5, on remarque que le critère est respecté si l'on sélectionne les gènes ayant plus de deux poids non nuls.

Tableau 5.5 Nombre de gènes sélectionné selon le nombre de poids non nul minimal souhaité.

	i					
	0	1	2	3	4	5
Nombre de gènes avec plus de i poid(s) non nul(s)	1575	774	**367**	174	77	43

Étape 4 : $\mathbf{Q}_2 \leftarrow ACP(\{\mathbf{z}_i\}_{i=1}^n, q_2)$

Finalement, on applique l'analyse en composantes principales sur notre ensemble de données au nombre de gènes réduit. Après l'application de l'ACP, l'ensemble de données comporte 116 lignes et de 117 colonnes. Ensuite, cette transformation doit être appliquée sur l'ensemble de validation et sur l'ensemble test.

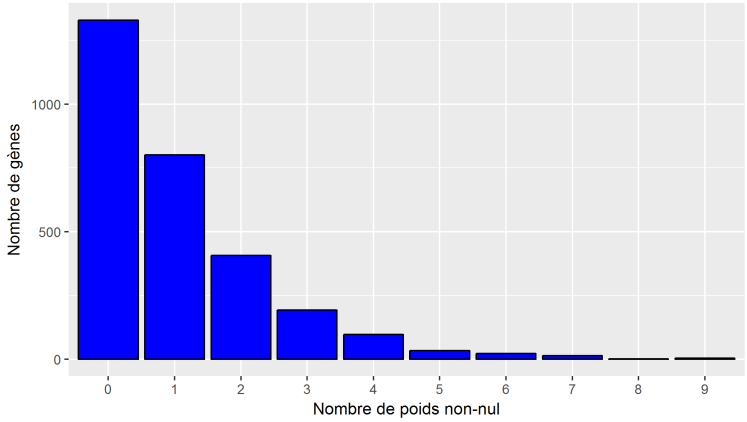


Figure 5.5 Histogramme de la fréquence des gènes selon le nombre poids non nul pour l'ensemble des données d'entraînement.

5.4 Détermination des meilleurs modèles

On obtient différentes tables dans lesquels on regardera la significativité du facteur méthode. Une valeur-p inférieure au seuil de signification ($\alpha^\star = \frac{\alpha}{3} \approx 0.01666$) indique que la méthode d'optimisation des hyperparamètres est un facteur statistiquement significatif. Il semble donc y avoir une différence significative entre les moyennes des méthodes pour la régression logistique pénalisée et la machine à *gradient boosting* (voir Tableau 5.6).

Tableau 5.6 Meilleures méthodes d'optimisations selon les classificateurs.

	Classificateur		
	RLP	MVS	MGB
Valeurs-p	11.55×10^{-5}	0.9169	$\mathbf{1.233 \times 10^{-5}}$
Meilleure moyenne	0.8125	0.8023	0.8219
Meilleure méthode	rpc	ng	rpg

Pour la régression logistique pénalisée, l'ANOVA à blocs complets a mise en évidence une différence de moyennes entre les méthodes (valeur-p $= 11.55 \times 10^{-5}$). En effectuant des comparaisons multiples, on constate la supériorité de la recherche par coordonnées par rapport (rpc) à la configuration par défaut (pd). D'ailleurs, une valeur-p de 0.0014458 a été obtenue lors des comparaisons multiples, ce qui est inférieur à notre seuil de significativité $\alpha^\star = 0.01666$. Aucune autre conclusion ne peut être tirée des comparaisons multiples. On choisit cette moyenne, car elle est supérieure aux autres.

Pour la machine à vecteurs de support, l'ANOVA à blocs complets n'indique pas une différence entre les moyennes des méthodes (valeur-p $= 0.9169$). Par conséquent, la comparaison

multiple est inutile. Par le raisonnement mentionné précédemment, on trouve que la meilleure méthode d'optimisation semble être celle basée sur le noyau gaussien.

Pour la machine à *gradient boosting*, l'ANOVA à blocs complets indique une différence entre les moyennes des méthodes (valeur-p $= 1.233 \times 10^{-5}$). La méthode avec la plus grande moyenne est la recherche aléatoire. Les comparaisons multiples montrent que la recherche aléatoire (ra) et la recherche par grille (rpg) surpassent la configuration par défaut. Cependant, il n'est pas possible de déterminer statistiquement laquelle de ces méthodes a une meilleure moyenne. On choisit la recherche par grille en raison de sa moyenne obtenue qui est plus élevée.

Une analyse des résidus a été faite et il ne semble pas y avoir de problèmes en lien avec la normalité des résidus.

5.5 Résultats (pour plusieurs expériences)

Les résultats obtenus sur l'ensemble test sont les suivants. Le temps moyen d'exécution de `SELECT 3()` est de 1.0645 minutes avec une erreur type de 0.0957 minute (voir le Tableau B.7 pour plus de détails). Cette estimation est basée sur 10 observations.

Tableau 5.7 Résultats des prédictions selon les meilleurs classificateurs pour `SELECT 3()`.

	Classificateur		
	RLP	MVS	MGB
Méthode	Recherche par coordonnées	Noyau linéaire	Recherche par grille
$\text{AUC}_{\text{moyenne}}$	0.7100	0.7711	0.7194
Erreur standard (AUC)	0.0486	0.0298	0.0272
$\text{Précision}_{\text{moyenne}}$	0.7071	0.7214	0.7035
Erreur standard (précision)	0.0484	0.0224	0.0324

CHAPITRE 6 ANALYSE DES RÉSULTATS

Un retour est effectué sur les composantes de ce travail. Dans un premier temps, on se penche sur les modifications de l'algorithme `SELECT()` afin de voir si celles-ci contribuent à améliorer les performances des classificateurs. Dans un deuxième temps, les AUC et les précisions sont comparées à celles de la littérature afin de savoir si ces résultats ont été battus.

6.1 Analyse des performances liées aux modifications de `SELECT()`

Pour chaque classificateur, plusieurs méthodes d'optimisation ont été mises en oeuvre afin d'optimiser nos hyperparamètres. Pour un classificateur donné et un algorithme fourni, il est possible de trouver une méthode d'optimisation qui semble mener à une plus grande performance moyenne. C'est ce qui a été fait dans les chapitres précédents. D'ailleurs, le Tableau 6.1 témoigne des résultats obtenus. Ce sont ces méthodes qui nous mènent aux résultats de la section 6.2.

Tableau 6.1 Méthodes d'optimisation choisies selon l'algorithme de sélection de variables et le classificateur.

Algorithme	Classificateur		
	RLP	MVS	MGB
`SELECT()`	rpc	ng	ra
`SELECT 2()`	rpc	nl	rpg
`SELECT 3()`	rpc	ng	rpg

Or, l'idée ici est d'analyser l'efficacité des modifications de `SELECT()`. Le fait d'effectuer des comparaisons basées sur des méthodes d'optimisation différentes ne semble pas approprié. Cela introduit du bruit dans l'ANOVA. Cependant, il est possible d'effectuer une ANOVA pour chaque méthode d'optimisation de chaque classificateur. Il suffit ensuite de regarder si le facteur qui correspond à l'algorithme de sélection de variables est significatif.

En ajustant le seuil avec la méthode de Bonferonni, on obtient un seuil de significativité $\alpha^\star = \frac{\alpha}{8} = 0.00625$. Cet ajustement du seuil de significativité résulte du fait que l'on effectue deux fois une série de huit ANOVA à blocs complets. Après avoir effectué ces ANOVA, aucune valeur-p n'est significative, et ce, pour l'AUC et la précision. Pour tous les classificateurs et les méthodes d'optimisation, nous n'avons pas assez de preuves pour affirmer que `SELECT 2()` ou `SELECT 3()` surpasse `SELECT()`. De façon similaire, il n'est pas possible ici d'énoncer les cas où `SELECT 2()` est meilleur que `SELECT 3()` et vice versa. Bref, il semble donc que

Tableau 6.2 Valeur-p associée à l'ANOVA à blocs complets pour le facteur algorithme. Le seuil de significativité est $\alpha = 0.00625$. Chaque valeur-p est associée à une ANOVA qui peut être retrouvée en annexe (du tableau B.8 au tableau B.23)

Critère	**RLP**			**MVS**		**MGB**		
	pd	**r**	**rpc**	**ng**	**nl**	**pd**	**ra**	**rpg**
AUC	0.8677	0.64112	0.9688	0.1897	0.8470	0.2111	0.4446	0.1550
Précision	0.8670	0.4552	0.8290	0.2945	0.7501	0.0325	0.8036	0.8861

nos modifications de `SELECT()` ne permettent pas une amélioration de la performance des classificateurs.

6.2 Comparaison avec les résultats de la littérature

Les résultats présentés à la fin de chaque algorithme doivent être regroupés afin d'avoir une perspective globale de notre recherche. Pour chacun des algorithmes, le Tableau 6.3 indique l'AUC moyenne des classificateurs. Le nombre de gènes conservé pour chaque algorithme est spécifié entre parenthèses dans la première colonne du tableau. Le Tableau 6.4 fournit des renseignements similaires, mais en utilisant la précision comme mesure de performance. Il faut se souvenir que les performances de `SELECT()` sont des références et que le but est de les surpasser.

En termes de performance moyenne sur l'ensemble test, la meilleure AUC est obtenue en sélectionnant les variables avec `SELECT 2()` et en utilisant une machine à support de vecteurs basée sur un noyau linéaire. L'AUC obtenue est de 0.7927 avec une erreur type de 0.0301. Malheureusement, il se trouve que ce résultat ne bat pas le seuil de 0.880 de la littérature (Huertas et Ramirez Svensson, 2016). Il est donc inutile d'effectuer un test d'hypothèses ici.

Tableau 6.3 AUC des classificateurs pour `SELECT()` et ses modifications. Le nombre entre parenthèses correspond à l'erreur type associée au classificateur et l'algorithme de sélection de variables en question.

Algorithme	Classificateur		
	RLP	MVS	MGB
`SELECT()` (323 gènes)	0.7206(0.0676)	0.7672(0.0416)	0.7347(0.0334)
`SELECT 2()` (390 gènes)	0.7277(0.0212)	**0.7927**(**0.0301**)	0.6705(0.0374)
`SELECT 3()` (367 gènes)	0.7100(0.0486)	0.7711(0.0224)	0.7194(0.0324)

Pour le critère de précision, la meilleure performance est obtenue en sélectionnant les variables avec `SELECT 2()` et en utilisant une machine à support de vecteurs basée sur un noyau

linéaire. L'AUC obtenue est de 0.7536 avec une erreur type de 0.0279. Il s'avère que ce résultat ne bat pas le seuil de 0.773 de la littérature (Ramey et Young, 2013). Encore une fois, il est donc inutile d'effectuer un test d'hypothèses.

Tableau 6.4 Précision des classificateurs pour `SELECT()` et ses modifications. Le nombre entre parenthèses correspond à l'erreur type.

Algorithme	Classificateur		
	RLP	MVS	MGB
`SELECT()` (323 gènes)	0.6786(0.0734)	0.7179(0.0299)	0.7107(0.0447)
`SELECT 2()` (390 gènes)	0.7250(0.0287)	**0.7536**(**0.0279**)	0.6964(0.0245)
`SELECT 3()` (367 gènes)	0.7071(0.0484)	0.7214(0.0224)	0.7035(0.0324)

Quelque chose d'important qui ressort de l'évaluation des performances est la présence de beaucoup d'AUC et de précision basses [1]. Les hautes mesures de performance de l'ensemble de validation suggèrent que les hyperparamètres obtenus mènent parfois à des problèmes de surapprentissage. Une cause plausible de ce phénomène est le grand nombre d'itérations utilisé pour optimiser les hyperparamètres.

Les résultats obtenus pour la précision et l'AUC ne sont pas très appréciables. En consultant ces résultats attentivement, on constate que les résultats sont parfois excellents (proche de un) et d'autres fois ils sont exécrables (près de 0.50). Cela s'explique par le fait qu'il y a du surapprentissage. Certains modèles apprennent les données *par coeur* lors de l'entraînement et cela mène à des pauvres performances sur l'ensemble test.

6.3 Désavantages et améliorations potentielles des algorithmes proposés

Les nouveaux algorithmes proposés comblent des lacunes de `SELECT()`. Cependant, ces méthodes ne sont pas à l'abri de certains problèmes. On élabore ici des solutions pour remédier à ces désavantages dont certaines sont des avenues de recherche. Puis, on traite des avantages des algorithmes `SELECT 2()` et `SELECT 3()`.

Un aspect de `SELECT 2()` qui laisse à désirer est la sélection de variables de l'étape 2, qui est un pilier de l'algorithme. Cette étape est basée sur une analyse de la magnitude [2] des coefficients de la régression logistique bayésienne. Les d variables ayant les plus grands coefficients sont conservées. Cette façon de faire est questionable, car les composantes sont basées sur des échelles différentes. Une meilleure façon de faire serait d'utiliser la statistique de Wald.

1. Plus précisément, il y a des expériences telles que : $\text{AUC}_{\text{test}} \approx 0.5000$ et/ou $\text{précision}_{\text{test}} \approx 0.5000$
2. La magnitude d'un coefficient β_j est définie comme $|\beta_j|$.

Les coefficients seraient ainsi normalisés par leur erreur type. On prendrait alors les d plus grandes statistiques de Wald (en valeur absolue). Une autre solution serait d'employer une méthode de mesure d'importance basée sur des modèles non linéaires comme les arbres de décisions. Par exemple, une forêt d'arbres aléatoire pourrait être employée pour mesurer l'importance des variables. Cette procédure a été utilisée pour `SELECT 3()`. Malheureusement, les résultats obtenus laissent à désirer. Ceux-ci ne montrent pas d'amélioration significative de l'algorithme `SELECT()`. Un autre désavantage est le long temps d'exécution de `SELECT 2()`. Cela s'explique par le fait que la recherche par coordonnées est une méthode coûteuse en terme d'itérations. Or, chaque itération de l'ACPC est longue à exécuter. Une avenue de recherche intéressante consisterait à combiner les modifications de `SELECT 2()` et de `SELECT 3()`.

Pour `SELECT 3()`, un des désavantages est le fait qu'il faille effectuer la mise à zéro de certains éléments de $\mathbf{V}^2$ en se basant sur un seuillage simple [3]. Il a été vu précédemment que cette méthode peut mener à une mauvaise identification des variables importantes. Pour éviter un tel problème, l'ACPC a été utilisée. Une solution intéressante serait d'employer l'analyse en composantes principales creuse groupé. En plus de permettre une mise à zéro naturelle des éléments de $\mathbf{V}$, elle permet une mise à zéro de groupes. Dans notre cas, un groupe pourrait correspondre aux poids d'un gène.

3. Mathématiquement, cela se résume à :$\forall i \in \{1, 2..., 2905\}, \forall j \in \{1, 2, ..., 116\}, v_{ij}^{(3)} = \mathbb{1}_{\{|v_{ij}^{(2)}|>\mu\}} v_{ij}^{(2)}$

CHAPITRE 7 EXPÉRIENCES SUR DIVERS ENSEMBLES DE DONNÉES

7.1 Motivation

Les améliorations potentielles entraînées par nos modifications d'algorithmes sont évaluées seulement sur un jeu de données lié au cancer du sein. Or, il est possible que les résultats obtenus soient d'une nature différente pour d'autres types de maladies. Par conséquent, cette section se concentre sur des analyses sommaires liées à d'autres jeux de données.

7.2 Myélome multiple (Tian, 2003)

7.2.1 Contexte

Les auteurs ont étudié les cellules plasmatiques purifiées de la moelle osseuse des patients témoins, ainsi que des patients atteints de myélome multiple nouvellement diagnostiqué. Les profils d'expression pour 12 625 gènes ont été obtenus à l'aide des puces à ADN. Les cellules plasmatiques ont été soumises à des analyses biochimiques et immunohistochimiques pour identifier les déterminants moléculaires des lésions ostéolytiques. Pour 36 patients atteints de myélome multiple, les lésions osseuses focales n'ont pas pu être détectées par imagerie par résonance magnétique (IRM). L'IRM a été utilisée pour détecter de telles lésions chez 137 patients.

Taille de l'échantillon	Nombre de gènes	Nombre de classes	Maladie
173	12625	2	Myélome multiple

7.2.2 Analyse

L'analyse consiste à effectuer diverses ANOVA à blocs complets. Le facteur est l'algorithme de sélection de variables, c'est-à-dire `SELECT()`, `SELECT 2()` ou `SELECT 3()`. Ce type d'ANOVA est effectué pour chaque méthode d'optimisation. Normalement, un seuil de signification $\alpha = 0.05$, mais en raison du fait que les ANOVA sont faites simultanément, un ajustement est nécessaire. Le seuil de signification à utiliser est donc $\alpha^\star = \frac{\alpha}{8} = 0.00625$.

En consultant le Tableau 7.1, il semble que les modifications de l'algorithme ne mènent pas à une amélioration des performances dans le cas des données provenant du myélome multiple.

Tableau 7.1 Valeur-p associée à l'ANOVA à blocs complets pour le facteur algorithme.

	RLP			**MVS**		**MGB**		
Critère	**pd**	**r**	**rpc**	**ng**	**nl**	**pd**	**ra**	**rpg**
AUC	0.5840	0.5738	0.8892	0.07334	0.5253	0.4487	0.8774	0.4510
Précision	0.7143	0.3707	0.4359	0.3612	0.4383	0.3253	0.2060	0.8044

7.3 Cancer de la prostate (Singh, 2002)

7.3.1 Contexte

Singh et al. (2002) ont examiné 235 cas de prostatectomie radicale chez des patients chirurgicaux entre 1995 et 1997. Les auteurs ont utilisé des puces à ADN d'oligonucléotides contenant des sondes pour environ 12 600 gènes. Ils ont signalé que 102 des échantillons de prostatectomie radicale sont de haute qualité : 52 échantillons de tumeur de la prostate et 50 échantillons de prostate non tumorale.

Taille de l'échantillon	Nombre de gènes	Nombre de classes	Maladie
102	12600	2	Cancer de la prostate

7.3.2 Analyse

La démarche effectuée est de la même nature que celle faite à la sous-section 7.2.2.

Tableau 7.2 Valeur-p associée à l'ANOVA à blocs complets pour le facteur algorithme (- : l'ANOVA ne peut pas être effectuée).

	RLP			**MVS**		**MGB**		
Critère	**pd**	**r**	**rpc**	**ng**	**nl**	**pd**	**ra**	**rpg**
AUC	0.5835	0.7685	0.08587	0.9785	0.86758	0.7949	0.8790	0.8646
Précision	0.6250	0.9036	0.1662	-	0.6131	0.8885	0.9792	0.6623

Il semble que les modifications de l'algorithme ne permettent pas d'obtenir une amélioration des performances dans le cas des données provenant du cancer de la prostate.

CHAPITRE 8 CONCLUSION

8.1 Synthèse du mémoire

Dans l'optique d'améliorer la sélection de variables issue de l'algorithme `SELECT()`, le présent travail s'est concentré sur deux piliers de l'algorithme : l'analyse en composantes principales et la sélection de composantes principales basée sur la régression logistique. Dans un premier temps, l'algorithme `SELECT()` a été reconstitué le plus fidèlement possible à des fins comparatives, et ce, avec une méthodologie objective. Pour chaque classificateur, une méthodologie spécifique a été élaborée.

Dans un deuxième temps, deux versions modifiées de l'algorithme ont été élaborées et ne mènent pas à des meilleurs résultats que `SELECT()`. Pour la première version de l'algorithme modifié, l'emploi de l'analyse en composantes principales creuse a permis une mise à zéro de certains éléments de la matrice de rotation de façon plus naturelle. Cette mise à zéro est effectuée grâce au fait que l'analyse en composantes principales creuse peut être interprétée comme un problème de régression. En profitant de cela et en employant l'*elastic net*, la naissance de l'analyse en composantes principales creuse est possible (Zou et al., 2004). Pour la seconde version de l'algorithme modifié, la substitution de la méthode de sélection de composantes [1] a mené à l'obtention d'un plus petit nombre de composantes. Il se trouve que la procédure traditionnelle consistait à conserver 90% des composantes avec les plus grands coefficients (selon la norme ℓ_1). Or, un désavantage de ce seuil est qu'il doit être fixé par l'expérimentateur. La nouvelle procédure utilise une mesure d'importance basée sur une forêt d'arbres aléatoire et le nombre de composantes à conserver est déterminé selon une approche statistiquement plus rigoureuse. Malheureusement, notre recherche ne montre aucune évidence que les versions modifiées sont meilleures que `SELECT()`. De plus, les résultats obtenus par la littérature restent invaincus.

Le présent mémoire contribue à la littérature et aux algorithmes de sélection de variables de plusieurs façons. Premièrement, l'ACPC a été mise en place et évite l'estimation du paramètre μ. Cependant, le prix à payer est l'estimation du vecteur de paramètres $\boldsymbol{\lambda}$. Le véritable avantage réside dans une identification de variables plus intéressantes qui résulte d'une matrice de rotation finale *naturellement* creuse. Deuxièmement, en plus de modifier l'ACP qui est un pilier de `SELECT()`, un autre pilier est modifié. Il s'agit d'un changement de la méthode de sélection de composantes de l'étape 2. Un algorithme basé sur une forêt

1. Cette dernière est originalement basée sur la régression logistique bayésienne.

d'arbres aléatoire a été mis en place afin de permettre une sélection stable et sans biais des variables importantes. Un aspect particulièrement intéressant est que l'estimation de d par l'expérimentateur n'est plus nécessaire. Malheureusement, ces développements n'ont pas mené à des améliorations. Dans les problèmes de grandes dimensions, les non-linéarités et des corrélations entre les variables font en sorte qu'il est ardu de comprendre ce qui se passe. Il est donc difficile d'expliquer pourquoi ces améliorations ne sont pas concluantes. Cependant, de nombreuses modifications potentielles peuvent être appliquées à l'algorithme `SELECT()`. Celles-ci pourraient mener à des améliorations intéressantes.

Enfin, il convient de rappeler que l'estimation des paramètres de `SELECT()` a été effectué en tentant de se rapprocher le plus des expériences d'Ocampo-Vegas et al. (2016). Lorsqu'un manque de clarté était présent [2], des estimations conservatrices, c'est-à-dire similaires à ce qu'on peut lire dans la littérature, ont été faites. L'idée derrière ces estimations est d'employer une méthodologie rigoureuse et reproductible. Pour l'algorithme `SELECT 2()`, il a été supposé que le vecteur $\boldsymbol{\lambda}$ contiendrait les mêmes valeurs pour tous ses éléments. Il aurait intéressant d'utiliser un jeu de données du site *GEO* (`https://www.ncbi.nlm.nih.gov/geo/`) afin de comparer nos résultats à ceux de divers chercheurs et ajouter nos résultats aux leurs.

8.2 Avenues de recherche

Plusieurs améliorations intéressantes ont déjà été proposées dans l'analyse (voir section 6.3). En voici quelques-unes de plus qui méritent un intérêt particulier :

- Pour la première modification, la valeur de $\mathrm{Sp}(\mathbf{V}^{(3)})$ obtenue par l'analyse en composantes principales creuse n'était pas suffisamment grande. Or, il est possible de diminuer encore plus la proportion de zéros dans $\mathbf{V}^{(3)}$ en déterminant un seuil d'arrondissement μ. Ce seuil agirait comme un complément dans le but d'atteindre la valeur de $\mathrm{Sp}(\mathbf{V}^{(3)})$ souhaitée. Cette modification nous permettrait de nous rapprocher des conditions de notre reconstitution de $\mathbf{V}^{(3)}$, mais au prix de l'estimation de μ. Peut-être que cela améliorerait `SELECT 2()`.
- La recherche par coordonnées a été effectuée en raison du fait que le temps de calcul était relativement petit. De plus, la fonction de coût était peu sensible à des petites variations des hyperparamètres. Cependant, pour des ensembles de données avec plus d'observations, les calculs numériques peuvent être un obstacle important. Il existe des méthodes d'optimisation sans dérivées qui sont plus performantes : MADS (ou *Mesh Adaptative Direct Search*, en anglais), Nelder Mead, etc.

2. Il y avait des manques de clarté par rapport aux choix de valeurs de certains paramètres de l'algorithme.

RÉFÉRENCES

C. Achen, “Interpreting and using regression”, *Sage*, 1982.

E. A. Ali, A. Aouatif, E. O. Abdeljalil, et A. Driss, “A two stage gene selection scheme utilizing MRMR filter and GA wrapper”, *Knowledge and Information Systems*, vol. 26, no. 3, pp. 487–500, 2011.

C. Aliferis, I. Tsamardinos, P. Massion, N. Fanananpazir, et D. Hardin, “Machine learning models for classification of lung cancer and selection of genotic markers using array gene expression data”, *FLAIRS Conference*, 2003.

U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, et A. Levine, “Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotidearrays”, *Proceedings of the National Academy of Sciences*, vol. 96, pp. 6745–6750, 1999.

R. Arias-Michel, M. Garcia-Torres, C. E. Schaerer, et D. Federico, “Feature selection via approximated markov blankets using the cfs method”, *Data Mining with Industrial Applications (DMIA)*, 2015.

C. Audet et M. Kokkolaras, “Blackbox and derivative-free optimization : theory, algorithms and applications”, *Optimization and Engineering*, vol. 17, no. 1, pp. 1–2, 2016. DOI : 10.1007/s11081-016-9307-4. En ligne : http://dx.doi.org/10.1007/s11081-016-9307-4

R. E. Bellman, “Adaptive control processes : A guided tour”, *Princeton Legacy Library*, pp. 499–511, 1961.

A. Ben-Dor, R. Shamir, et Z. Yakhini, “Clustering gene expression patterns”, *Journal of Computational Biology*, vol. 6, pp. 281–297, 1999.

A. Ben-Hur et J. Weston, “A user’s guide to support vector machines”, *Book, Data mining techniques for the life sciences*, 2010.

J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, R. Tibshirani, et R. J. Tibshirani, “Strong rules for discarding predictors in lasso-type problems”, *Journal of the Royal Statistical Society*, vol. Series B, 74 (1).

C. Bishop, “Pattern recognition and machine learning”, *Springer*, 2007.

P. Breheny, “The group exponential lasso for bi-level variable selection”, *Biometrics*, vol. 71, pp. 731–740, 2009.

P. J. Breheny, “Regularized methods for high-dimensional and bi-level variable selection”, Thèse de doctorat, University of Iowa, 2009.

H. Chih-Wei, C. Chih-Chung, et L. Chih-Jen, “A practical guide to support vector classification”, Technical report, Departement of Computer Science, National Taiwan University, Rapp. tech., 2010. En ligne : `https://www.cs.sfu.ca/~oschulte/teaching/726/spring11/svmguide.pdf`

C. Click, M. Malohlava, V. Parmar, H. Roark, et A. Candel, “Gradient boosting machine with h2o”, *http ://h2o.ai/ressources/*, 2016.

C. Cortes et V. Vapnik, “Support vector networks”, *Machine Learning*, pp. 20 :273–297, 1995.

N. Cristianini et J. Shawe-Taylor, “An introduction to support vector machines”, *Cambridge UP*, 2000.

M. Eisen, P. Spellman, P. Brown, et D. Botstein, “Cluster analysis and display of genome-wide expression patterns”, *Proceedings of the National Academy of Sciences*, vol. 95, pp. 14 863–14 868, 1998.

R. Fakoor, F. Ladhak, A. Nazi, et M. Huber, “Using deep learning to enhance cancer diagnosis and classification”, *Computer Science and Engineering Dep., University of Texas at Arlington*, juin 1998.

J. Fan et R. Li, “Variable selection via nonconcave penalized likelihood and its oracle properties.” *Journal of the American Statistical Association 96*, vol. 456, pp. 1348–1360, 2001.

E. Fermi et N. Metropolis, “Numerical solution of a minimum problem”, Los Alamos National Laboratory, Los Alamos, USA, Los Alamos Unclassified Report LA–1492, 1952.

I. E. Frank et J. H. Friedman, “A statistical view of some chemometrics regression tools”, *Technometrics*, vol. 35, pp. 109–135, 1993.

J. Friedman, T. Hastie, et R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent”, *Journal of Statistical Software*, vol. 33 (1).

G. M. Furnival et R. W. Wilson, “Regression by leaps and bounds”, *Technometrics*, vol. 16, pp. 499–511, novembre 1974.

M. Garcìa-Torres, F. Gomez-Vela, D. Becerra-Alonso, B. e. Melian-Batista, et J. M. Moreno-Vega, “Feature grouping and selection on high-dimensional microarray data”, *IEEE Computer Science*, pp. 30–37, 2015.

U. Gromping, “Relative importance for linear regression in r : The package relaimpo”, *Journal of Statistical Software, Volume 17, Issue 1*, 2006.

I. Guyon, S. Gunn, A. Ben-Hur, et G. Dror, “Result analysis of the nips 2003 feature selection challenge”, *Advances in Neural Information Processing Systems*, pp. 17 : 545–552, 2005.

T. Hastie, R. Tibshirani, et J. Friedman, “The elements of statistical learning”, *Springer*, 2001.

J. Huang, S. Ma, H. Xie, et C.-H. Zhang, “A group bridge approach for variable selection”, *Technical Report 376, Department of Statistics and Actuarial Science, University of Iowa*, 2007.

——, “Automatic threshold search for heat map based feature selection : A cancer dataset analysis”, *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, No 7, pp. 1308–1313, 2016.

I. M. Johnstone et A. Y. Lu, “On consistency and sparsity for principal components analysis in high dimensions”, *Journal of the American Statistical Association*, vol. 104 (486), pp. 682–693, 2010.

S. Keerthi et C.-J. Lin, “Asymptotic behaviors of support vector machines with gaussian kernel”, *Neural Computation*, vol. 15 (7), pp. 1667–1689, 2003.

K. Kira et L. Rendel, “The feature selection problem : Traditional methods and a new algorithm”, *In Proc. Tenth National Conference on Artificial Intelligence*, pp. 129–134, 1992.

R. Kohavi et G. John, “Wrappers for feature subset selection”, *Artificial Intelligence*, pp. 97 : 273–324, 2007.

I. Kononenko, E. Simec, et M. Robnik-Sikonja, “Overcoming the myopia of induction learning algorithms with relief”, *Applied Intelligence*, vol. 17 (1), pp. 39–55, 1997.

M. B. Kursa et W. R. Rudnicki, “Feature selection with the boruta package”, *Journal of Statistical Software, Volume 36, Issue 11*, 2010.

N. Kwak et C. Kim, “Dimensionality reduction based on ica for regression problems”, *ICANN 2006 : Lecture Notes in Computer Science*, pp. 1–10, 2006.

A. Liaw et M. Wiener, “Classification and regression by randomforest”, *R news*, pp. 2(3) : 18–22, 2002.

H.-T. S. Lin et C.-J. Lin, “A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods”, *Technical report, Departement of Computer Science, Naitonal Taiwan University*, 2003.

G. Michaels, D. Carr, M. Askenazi, S. Fuhrman, X. Wen, et R. Somogyi, “Cluster analysis and display of genome-wide expression patterns”, *Pac. Symp. Biocomputing*, pp. 42–53, 1998.

R. Nilsson, J. Pena, J. Björkegren, et J. Tegnér, “Consistent feature selection for pattern recognition in polynomial time”, *The Journal of Machine Learning Research*, p. 8 : 612, 2007.

T. Nykodym, T. Kraljevic, N. Hussami, A. Rao, et A. Wang, “Generalized linear modeling with h2o”, *http ://h2o.ai/ressources/*, 2016.

R. Ocampo-Vega, G. Sanchez-Ante, M. A. de Luna, R. Vega, L. Falcón-Morales, et H. Sossa, “Improving pattern classification of dna microarray data by using pca and logistic regression”, *Data Visualization and Pattern Recognition Lab, Tecnológico de Monterrey*, 2016.

Y. J. Park et S. B. Kim, “Unsupervised feature selection method based on principal component loading vectors”, *Journal of the Korean Institute of Industrial Engineers*, vol. 40, No. 3, pp. 275–282, 2014.

V. Partovi Nia, “Fast high-dimensional bayesian clasification and clustering”, Thèse de doctorat, PhD (Doctor of Philosophy) thesis, École Polytechnique Fédérale de Lausanne, 2009.

S. Powers, T. Hastie, et T. Robert, “Customized training with an application to mass spectrometric imaging of cancer tissue”, *Journal of the Royal Statistical Society*, vol. Series B 71(3), pp. 615–636, 2009.

S. Ramaswany, P. Tamayo, R. Rifkin, S. Mukherjee, D. Hardin, C. Yeang, C. Angelo M., Ladd, M. Reich, E. Latulippe, J. Mesirov, T. Poggio, W. Gerald, M. Loda, E. Lander, et T. Golub, "Multiclass cancer diagnosis using tumor gene expression signatures", *Proceedings of the National Academy of Sciences of United States of America*, vol. 98(26), pp. 15 149–15 154, 2001.

J. A. Ramey et P. D. Young, "A comparison of regularization methods applied to the linear discriminant function with highdimensional microarray data", *Journal of Statistical Computation and Simulation*, vol. 83 :3, pp. 581–596, 2013.

T. Robert, "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society, Serie B (Methodological)*, vol. 58, pp. 267–288, 1996.

W. Rudnicki, M. Kierczak, J. Koronacki, et J. Komorowski, "A statistical method for determining importance of variables in an information system", *Rough Sets and Current Trends in Computing, 5th International Conference, RSCTC 2006*, pp. 557–566, 2006.

B. Schölkopf et A. Smola, "Learning with kernels", *MIT Press, Cambridge, MA*, 2002.

A. Sharma, S. Imoto, et S. Miyano, "A top-r feature selection algorithm for microarray gene expression data", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, pp. 754–764, 2012.

P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, et B. Futcher, "Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization", *Molecular Biology of the Cell*, vol. 9, pp. 3273–3297, 1998.

E. V. Strobl et S. Visweswaran, "Dependence versus conditional dependence in local causal dis-covery from gene expression data." *arXiv :1407.7566 [q-bio.QM]*, 2014.

P. Svensson, "Machine learning techniques for binary classification of microarray data with correlation-based gene selection", Thèse de doctorat, Master thesis, Uppsala University, 2016.

D. L. Tong et A. C. Schierz, "Hybrid genetic algorithm-neural network : Feature extraction for unpreprocessed microarray data", *Artificial Intelligence in Medecine*, vol. 53(1), pp. 47–56, 2011.

Y. Wang, I. Tetko, M. Hall, E. Frank, A. Facius, K. Mayer, et H. Mewes, "Gene selection

from microarray data for cancer classification-a machine learning approach", *Computational Biology and Chemistry*, vol. 29(1), pp. 37–46, 2005.

K. Wei, R. V. Charles, G. Hiromi, T. R. Jack, et H. Xudong, "A review of independent component analysis application to microarray gene expression data", *Biotechniques*, vol. 45 (5), pp. 501–520, 2008.

L. Wilkinson et G. Dallal, "Tests of significance in forward selection regression with an f-to enter stopping rule", *Technometrics*, vol. 23, pp. 377–380, 1981.

D. M. Witten et R. Tibshirani, "A framework for feature selection in clustering", *Journal of the American Statistical Association*, vol. 105 (490), pp. 713–726, 2010.

D. M. Witten, R. Tibshirani, et T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis", *Biostatistics*, vol. 10, 3, pp. 515–534, 2009.

D. Witten, T. Hastie, et R. Tibshirani, "Discussion of "on consistency and sparsity of principal components analysis in high dimensions" by johnstone and lu", *Journal of the American Statistical Association*, vol. 104(486), pp. 698–699, 2009.

H. Yu, G. Gu, H. Liu, J. Shen, et J. Zhao, "A modified ant colony optimization algorithm for tumor marker gene selection", *Genomics, proteomics and bioinformatics*, vol. 7 (4), pp. 200–208, 2013.

M. Yuan et Y. Lin, "Model selection and estimation in regression with grouped variables", *Journal of the Royal Statistical Society, Series B : Statistical Methodology*, vol. 68 (1), pp. 49–67, 2006.

C.-H. Zhang, "Penalized linear unbiased selection", *Technical Report 2007-003, Department of Statistics and Biostatistics, Rutgers University*, 2007.

H. Zou, T. Hastie, et R. Tibshirani, "Sparse principal component analysis", *Journal of Computational and Graphical Statistics*, vol. Number 2, pp. 265–286, 2006.

ANNEXE A CODE

Le code peut être retrouvé sur le répertoire Gihub suivant : `https://github.com/TheDataMiningMap/Data-Mining/blob/master/SELECT.zip`.

Functions

- But : Mettre en place des fonctions auxiliaires pour la sélection de variables selon `SELECT()` et ses modifications.
- Nombre de lignes : 800
- Taille du fichier : 36 Ko

Data Extractor

- But : Exécuter la séparation de données et effectuer la sélection de variables.
- Nombre de lignes : 100
- Taille du fichier : 4 Ko

Analytic Engine

- But : Optimiser les hyperparamètres de chacun des modèles. Puis, retourner les résultats et les valeurs des hyperparamètres dans le dossier approprié.
- Nombre de lignes : 275
- Taille du fichier : 14 Ko

ANNEXE B TABLEAUX

Tableau B.1 Valeurs-p pour des comparaisons par paires de méthodes d'optimisation (pour la régression logistique pénalisée) selon différents algorithmes de sélection de variables.

Algorithme	Comparaisons de méthodes		
	r-pd	rpc-pd	rpc-r
`SELECT ()`	0.5549	0.9433	0.3055
`SELECT 2()`	0.0027	0.0003	0.0014
`SELECT 3()`	0.0256	0.0005	0.0377

Tableau B.2 Valeurs-p pour des comparaisons par paires de méthodes d'optimisation (pour la machine à *gradient boosting*) selon différents algorithmes de sélection de variables.

Comparaison de méthodes	Algorithme		
	`SELECT ()`	`SELECT 2()`	`SELECT 3()`
nl-ng	0.2058	0.9290	0.9169

Tableau B.3 Valeurs-p pour des comparaisons par paires de méthodes d'optimisation (pour la machine à *gradient boosting*) selon différents algorithmes de sélection de variables.

Algorithme	Comparaisons de méthodes		
	r-pd	rpc-pd	rpc-r
`SELECT ()`	<0.0000	0.0001	<0.000
`SELECT 2()`	<0.0000	<0.0000	<0.000
`SELECT 3()`	0.3483	0.5297	0.7707

Tableau B.4 AUC pour des modèles entraînés sur les données de Gravier et al. (2010) pour l'algorithme `SELECT ()`.

Expérience	**RLP**			**MVS**		**MGB**		
	pd	**r**	**rpc**	**ng**	**nl**	**pd**	**ra**	**rpg**
1	0.6875	0.7031	0.7109	0.7578	0.6797	0.6406	0.7188	0.7188
2	0.8906	0.9531	0.9531	0.8320	0.8281	0.8906	0.9375	0.9297
3	0.7031	0.7266	0.8750	0.8984	0.7578	0.7188	0.8750	0.8438
4	0.6797	0.7031	0.7422	0.6445	0.7265	0.7031	0.7500	0.7734
5	0.5000	0.5000	0.5000	0.7500	0.7344	0.6250	0.7109	0.6719
6	0.5313	0.5013	0.5859	0.7265	0.6484	0.5313	0.5781	0.5547
7	0.9219	0.9219	0.9453	0.9141	0.9219	0.7967	0.9140	0.9141
8	0.5703	0.5703	0.7578	0.9453	0.8047	0.6875	0.7734	0.7500
9	0.5313	0.5781	0.5938	0.7188	0.7422	0.6328	0.6797	0.6797
10	0.6484	0.6563	0.6719	0.7656	0.7891	0.6641	0.7344	0.6953

Tableau B.5 AUC pour des modèles entraînés sur les données de Gravier et al. (2010) pour l'algorithme `SELECT 2()`.

Expérience	**RLP**			**MVS**		**MGB**		
	pd	**r**	**rpc**	**ng**	**nl**	**pd**	**ra**	**rpg**
1	0.6641	0.6641	0.7500	0.5156	0.7656	0.5469	0.6250	0.6133
2	0.9610	0.9610	0.9766	0.9453	0.9219	0.7188	0.7969	0.8750
3	0.6250	0.6719	0.6953	0.8281	0.7266	0.6953	0.8281	0.8516
4	0.7266	0.7266	0.7344	0.7344	0.6875	0.6719	0.7266	0.7031
5	0.8359	0.8359	0.8672	0.7539	0.7500	0.7656	0.7969	0.7969
6	0.5000	0.5000	0.6563	0.6602	0.7031	0.5547	0.6953	0.7148
7	0.7109	0.7109	0.7813	0.7891	0.7695	0.6016	0.6953	0.7109
8	0.8359	0.8359	0.8750	0.8359	0.8047	0.7578	0.8594	0.8672
9	0.5000	0.5000	0.6641	0.7813	0.7813	0.6953	0.7344	0.7656
10	0.7891	0.7891	0.8672	0.8125	0.7734	0.7734	0.7656	0.7813

Tableau B.6 AUC pour des modèles entraînés sur les données de Gravier et al. (2010) pour l'algorithme `SELECT 3()`.

Expérience	**RLP**			**MVS**		**MGB**		
	pd	**r**	**rpc**	**ng**	**nl**	**pd**	**ra**	**rpg**
1	0.6094	0.7422	0.7500	0.8242	0.7578	0.7109	0.7578	0.7578
2	0.8438	0.8672	0.9063	0.8242	0.8594	0.7969	0.9531	0.9609
3	0.6406	0.7969	0.8047	0.8672	0.7891	0.7969	0.8203	0.8281
4	0.5000	0.5000	0.6406	0.7422	0.7891	0.7266	0.8203	0.8047
5	0.5000	0.5000	0.8359	0.7773	0.8477	0.7109	0.7891	0.7734
6	0.5391	0.6328	0.7813	0.6680	0.6250	0.6719	0.7344	0.7421
7	0.8750	0.9063	0.9141	0.9141	0.9063	0.9531	0.9838	0.9844
8	0.7734	0.7734	0.7813	0.9063	0.7813	0.8438	0.8906	0.9219
9	0.8203	0.8281	0.9063	0.7500	0.8984	0.6016	0.6835	0.6797
10	0.7891	0.7891	0.8047	0.7500	0.7422	0.6641	0.7109	0.7656

Tableau B.7 Temps d'exécution des algorithmes de sélection de variables.

Expérience	**Temps d'exécution (minutes)**		
	`SELECT()`	`SELECT 2()`	`SELECT 3()`
1	0.1321	4467.413	1.2023
2	0.1259	5336.320	0.4935
3	0.1272	4752.595	1.2969
4	0.1154	4925.637	1.2998
5	0.1133	4434.644	0.9850
6	0.1180	4081.806	1.1103
7	0.1097	4513.510	0.5530
8	0.1116	4813.893	1.3202
9	0.1147	4288.952	1.1573
10	0.1144	4232.186	1.2273

Tableau B.8 RLP : pd (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	$\Pr(> F)$
Bloc	0.0566	1	0.0566	0.1148
Algo	0.0061	2	0.0030	0.8677
Erreur	0.55241	26	0.0212	

Tableau B.9 RLP : pd (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.1144	1	0.1144	0.0857
Algo	0.0103	2	0.0051	0.8670
Erreur	0.9323	26	0.0359	

Tableau B.10 RLP : r (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0733	1	0.0733	0.0878
Algo	0.0211	2	0.0105	0.6411
Erreur	0.6055	26	0.0232	

Tableau B.11 RLP : r (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.1254	1	0.1254	0.0733
Algo	0.0584	2	0.0292	0.4552
Erreur	0.9361	26	0.0360	

Tableau B.12 RLP : rpg (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0646	1	0.0646	0.0097
Algo	0.0334	2	0.0167	0.1550
Erreur	0.2163	26	0.0083	

Tableau B.13 RLP : rpg (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0712	1	0.0712	0.0122
Algo	0.0024	2	0.0012	0.8861
Erreur	0.2547	26	0.0098	

Tableau B.14 MVS : ng (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0127	1	0.0127	0.1626
Algo	0.0219	2	0.0109	0.1897
Erreur	0.1605	26	0.0061	

Tableau B.15 MVS : ng (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0014	1	0.0014	0.1677
Algo	0.0018	2	0.0009	0.2945
Erreur	0.0181	26	0.0007	

Tableau B.16 MVS : nl (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0224	1	0.0224	0.1717
Algo	0.0038	2	0.0019	0.8470
Erreur	0.2951	26	0.0113	

Tableau B.17 MVS : nl (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0164	1	0.0165	0.1533
Algo	0.0044	2	0.0022	0.7501
Erreur	0.1977	26	0.0076	

Tableau B.18 MGB : pd (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0350	1	0.0350	0.0380
Algo	0.0242	2	0.0121	0.2111
Erreur	0.1902	26	0.0073	

Tableau B.19 MGB : pd (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0612	1	0.0612	0.0162
Algo	0.0725	2	0.0362	0.0325
Erreur	0.2406	26	0.0092	

Tableau B.20 MGB : ra (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0434	1	0.0434	0.0499
Algo	0.0172	2	0.0085	0.4446
Erreur	0.2672	26	0.0103	

Tableau B.21 MGB : ra (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0427	1	0.0427	0.0977
Algo	0.0064	2	0.0032	0.8036
Erreur	0.3760	26	0.0145	

Tableau B.22 MGB : rpg (AUC)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0646	1	0.0646	0.0097
Algo	0.0334	2	0.0167	0.1550
Erreur	0.2163	26	0.0083	

Tableau B.23 MGB : rpg (précision)

Table ANOVA				
Source	**SS**	**df**	**MS**	Pr($> F$)
Bloc	0.0711	1	0.0711	0.0121
Algo	0.0024	2	0.0012	0.8861
Erreur	0.2547	26	0.0098	