SY19 - Regression et Classification - Selection de modèles

JoseMaria COLINA et Paul GOUJON

Automne 2016

Table des matières

Chapitre 1

Avant Propos

1.1 Introduction

Introduction Au cours de ce TP4, nous sommes confrontés à deux problèmes typiques du Data Scientist : un problème de classification, et un problème de régression. Deux jeux de données d'entrainement inconnus sont nos inputs, et nous devons y appliquer notre bagage scientifique naissant en techniques de la Data Science.

Objectif Nous avons pour objectif de fournir en output de ce TP des modèles de prédiction ayant les meilleures performances possible, aussi bien pour le problème de régression que pour le problème de classification.

Difficultés escomptées La principale difficulté de cet exercice vient du fait que contrairement à ce à quoi nous avons été habitués, les données qui nous sont fournies aujourd'hui sont des données réelles, donc potentiellement bruitées, et également présentant des espaces de prédicteurs (/inputs) de grandes dimensions (ex : 256 inputs pour le problème des phonèmes !). Nous ne sommes pas guidés, comme nous en avons l'habitude, et allons devoir gérer ces problématiques de manière autonome.

Chapitre 2

Classification de Phonemes

2.1 Caractérisation du dataset

Origine Données tirées de la base de données du "TIMIT Acoustic-Phonetic Continuous Speech Corpus", corpus très utilisé en recherche dans le domaine de la "Speech Recognition".

Formation Le dataset a été formé en séléctionnant 5 phonèmes pour des tâches de classification, basés sur les discours digitalisés de la database. Les phonèmes sont transcrits comme suit :

1. "sh" : pour le "sh" de "she"
2. "dcl" : pour le "dcl" de "dark"
3. "iy" : pour le "iy" de "she"
4. "aa" : pour le "aa" de "dark"
5. "ao" : pour le "ao" de "water"

A partir de discours continus de 50 orateurs males, 4509 segments de 32 msec ont été sélectionnés, soit approximativement 2 exemples de chaque phonème pour chaque orateur. Chaque segment est représenté par 512 échantillons, échantillonnés à une fréquence de 16kHz, et chacun des segments représente un des cinq phonèmes sus-cités.

Répartition La répartition des 4509 segments est la suivante :

TABLE 2.1 – Répartition des phonèmes

aa	ao	dcl	iy	sh
695	1022	757	1163	872

Traitement Pour chacun des segments a été calculé un log-periodogramme, l'une des méthodes des plus utilisées afin de transformer des données oratoires en données utilisables à des fins de "speech recognition". Ainsi les données finales contenues dans le dataset consistent en 4509 log-periodogrammes de longueur 256, caractérisées par leur classe.

Formellement Le dataset contient 4509 observations, chacune caractérisée par 256 prédicteurs, une étiquette de classe, et un identifiant correspondant à l'orateur.

2.2 Etude préliminaire

2.2.1 Analyse des Composantes Principales - PCA

PCA Notre premier réflexe, face à ce dataset aux nombreux inputs, a été d'effectuer une ACP, afin d'essayer de tirer des conclusions quant aux proportions de variance expliquée par les principales composantes. Ci-dessous sont représentées respectivement les courbes des deux principales composantes, de proportions de variance expliquée et de proportions cumulées de variance expliquée, dans cet ordre.

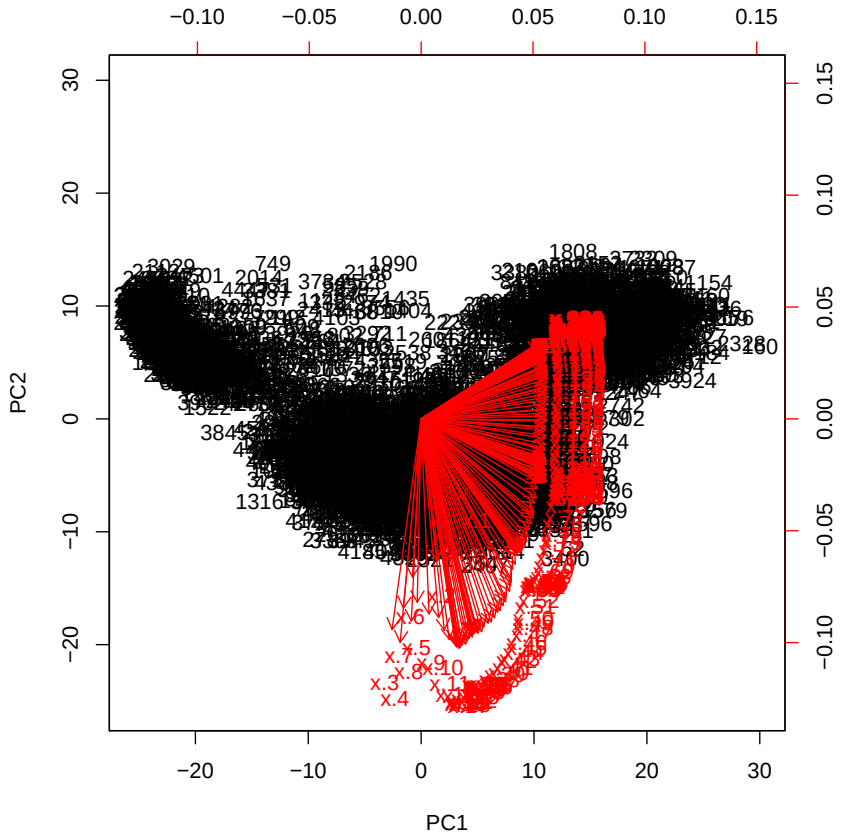

FIGURE 2.1 – Représentation des deux principales composantes du dataset Phoneme

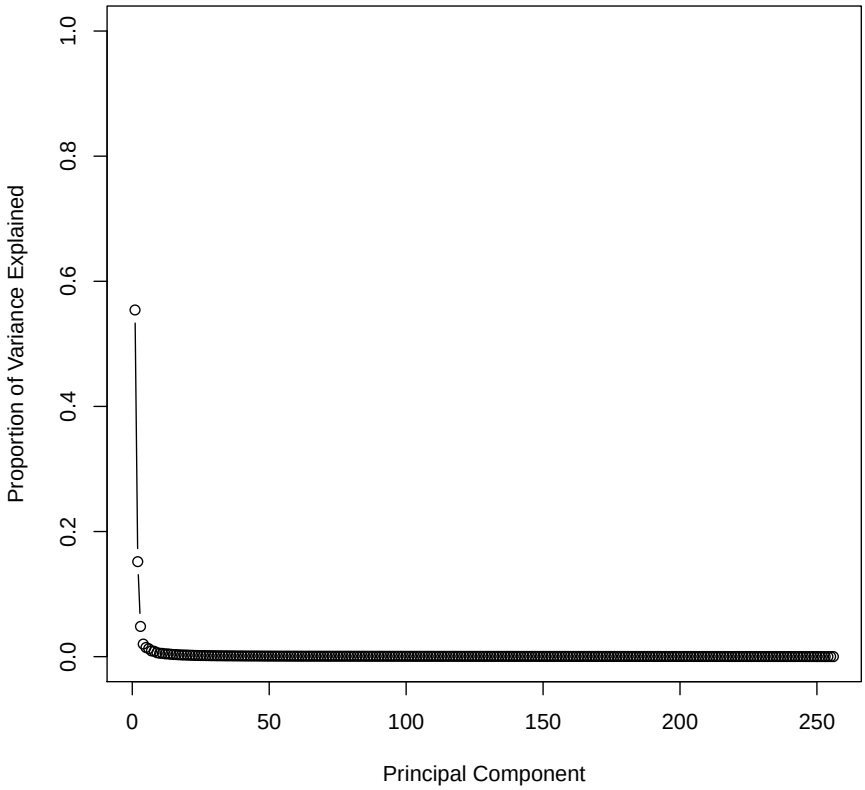

FIGURE 2.2 – Proportion de variance expliquée par les composantes principales du dataset

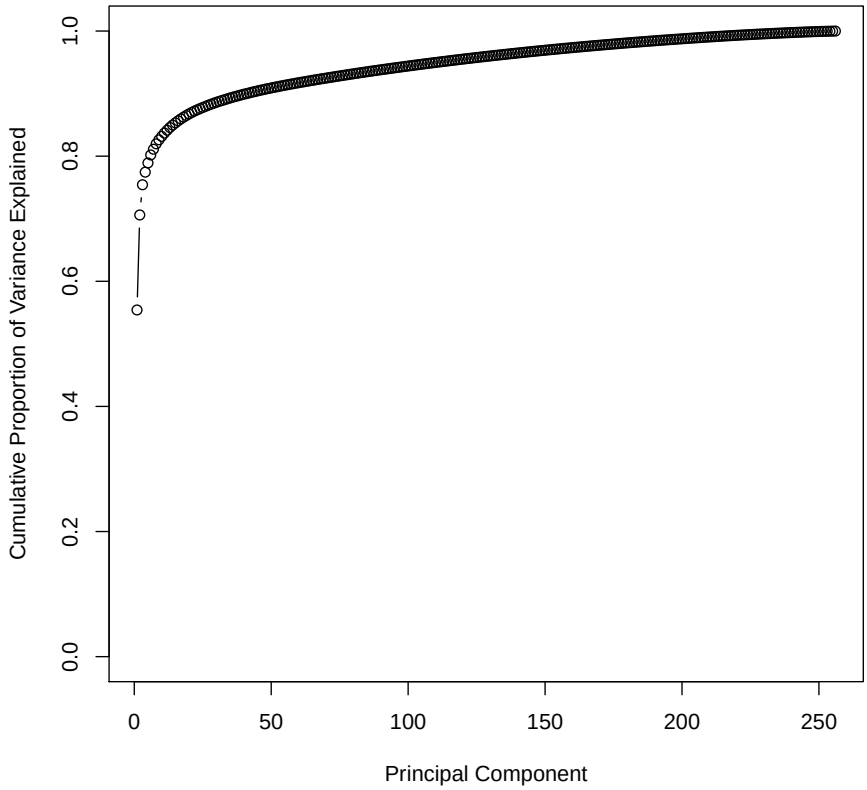

FIGURE 2.3 – Proportion cumulée de variance expliquée par les composantes principales du dataset

Interprétation Bien que la représentation des deux principales composantes du dataset ne soit pas très explicite. La consultation des proportions de variance expliquée des composantes principales nous montre qu'un nombre très peu élevé de composantes principales est suffisant pour expliquer une très importante proportion de variance. Nous pouvons ainsi émettre l'hypothèse que les méthodes de réduction de dimension seront efficaces sur ce dataset.

Etude de la corrélation au sein du dataset Nous réalisons également une "Heatmap" (ci-dessous), nous permettant de visualiser assez facilement la corrélation entre les différents prédicteurs de notre dataset. Une couleur claire représente une corrélation faible entre les variables, tandis qu'une couleur se rapprochant du rouge représente un corrélation relativement élevée entre les variables (Note : les labels en abcisse et en ordonnées se superposent, mais globalement il s'agit de la représentation de la matrice de corrélation, il suffit de s'imaginer les valeurs entre 0 et 1 remplacées par une échelle de couleur entre le blanc et le rouge en passant par le jaune).

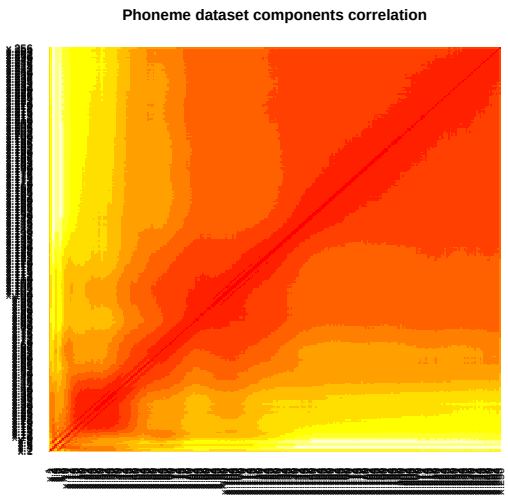

FIGURE 2.4 – Heatmap représentant la corrélation entre les prédicteurs

Interprétation Cette heatmap nous permet de voir que les prédicteur (/inputs) sont relativement corrélés, ce qui corrobore les résultats constatés lors de l'étude de la PCA. Ayant eu un premier aperçu du dataset, attaquons nous au problème de classification.

2.3 Multinomial Classification - Caractérisation du problème

2.3.1 La Multinomial ou Multiclass Classification

Généralités Nous devons effectuer une tâche de classification supervisée des phonèmes parmi $k = 5$ classes différentes. Lorsque $k \neq 2$, on parle de classification multiclasse ou multinomiale. Différentes stratégies peuvent être utilisées afin d'accomplir des tâches de multinomial classification. Nous allons les présenter dans la section qui suit.

2.3.2 Stratégies de classification multinomiale

One vs Rest (ou One vs All) Stratégie impliquant l'entrainement d'un classifieur par classe, en utilisant les échantillons étiquetés par la classe en question comme positifs, et les échantillons des autres classes comme négatifs. Le classifieur choisi doit impérativement produire un indice de confiance pour l'appartenance ou non d'un individu à une classe. La classe choisie est ainsi celle présentant l'indice de confiance le plus élevé.

One vs One Stratégie consistante à entrainer $\frac{k(k-1)}{2}$ classifieurs binaires. Chacun reçoit les positifs d'une paire de classes du dataset d'origine, et doit apprendre à les distinguer. Au moment de la prédiction, un "schéma de vote" est appliqué : les $\frac{k(k-1)}{2}$ classifieurs sont appliqués à l'individu de test, et la classe ayant reçu le plus de "+1" est attribuée par le "classifieur combiné".

Extension des méthodes de classification binaire Certaines méthodes de classification binaires, telles que la régression logistique, l'analyse discriminante linéaire ou quadratique, ou encore la méthode des k plus proches voisins peuvent être aisément étendues au cadre de la classification multinomiale.

Classification Hierarchique La classification hiérarchique s'attaque au problème de classification multinomiale en divisant l'espace de décision sous la forme d'un arbre. Chaque noeud père est une division de l'espace en de multiples noeuds fils, jusqu'à ce qu'il y ait une feuille par classe. Des mesures de dissimilarités sont généralement utilisées afin de diviser l'espace de décision en sous espaces.

2.3.3 Dans le cadre de ce TP

Choix de stratégies pour notre étude Dans le cadre de ce TP, nous allons dans un premier temps utiliser et optimiser les extensions de la régression logistique, de l'analyse discriminante et des k plus proches voisins au cas multiclasse. Dans un second temps nous utiliserons la méthode dite de "Forward Stepwise Selection" afin de réduire la dimension de l'espace de prédicteurs à la base de notre modèle.
Finalement, s'il nous reste du temps après avoir traité le problème de régression, nous espèrons avoir l'occasion d'implémenter à la main une stratégie "One vs All", et pourquoi pas, pour finir, des classifieurs utilisant les arbres de décision, et les SVM.

2.4 Evaluation des performances

2.4.1 Taux d'erreur

Performances d'un classifieur Plusieurs indicateurs permettent de mesurer les performances d'un classifieur. Nous allons pour notre part utiliser le taux d'erreur de nos classifieurs. L'expression de ce dernier est la suivante :

$$Err = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

Avec y_i la classe théorique de l'individu, $\hat{y}_i$ la classe prédite par notre classifieur, I la fonction indicatrice, et n le nombre d'individus.

2.4.2 Estimation du taux d'erreur

Taux d'erreur de test Notre objectif est d'obtenir une mesure la plus précise possible de taux d'erreur de test de notre classifieur, afin de ne pas sur-estimer ni sous-estimer le taux d'erreur réel de ce dernier. Plusieurs méthodes sont envisageable à ces fins.

Méthodes envisagées Les méthodes les plus connues (et que nous avons ainsi envisagées) sont les suivantes.

1. Validation Set : consiste à séparer notre dataset en un ensemble d'apprentissage, et un ensemble de validation de tailles aléatoires, et mesurer le taux d'erreur du classifieur entrainé sur l'ensemble d'apprentissage lorsque l'on l'applique à l'ensemble de validation.
2. Leave-One-Out Cross-Validation : consiste à "mettre de côté" chacun des individus du dataset initial uns à uns, en entrainant le classifieur sur les individus restants, puis calculer le taux d'erreur sur l'individu "isolé" (taux d'erreur final = nombre d'erreur divisé par nombre d'individus).

3. k-Fold Cross-Validation : consiste à diviser le dataset initial en k ensembles de taille égale. Puis entrainer le classifieur sur les individus de $k-1$ ensembles puis le tester et calculer son taux d'erreur sur l'ensemble restant (répéter l'opération de manière à ce que chaque ensemble ait servi d'ensemble de test).
4. Bootstrap : consiste à calculer le taux d'erreur du classifieur en tirant plusieurs fois aléatoirement des individus du dataset initial pour composer l'ensemble d'apprentissage et l'ensemble de test.

Choix Chacune des méthodes vues précédemment a ses avantages et ses inconvénients (que nous ne détaillerons pas ici). Dans la pratique (empiriquement), il se trouve que la 5-fold et la 10-fold Cross Validation, donnant d'excellents résultats (bon bias-variance trade-off), il s'agisse de méthodes extrêmement plébiscitées. Nous avons donc décidé d'utiliser la 10-fold Cross Validation dans le cadre de notre TP. L'expression de l'estimation du taux d'erreur par 10-fold Cross Validation est la suivante.

$$CV_{10} = \frac{1}{10} \sum_{i=1}^{10} Err_i$$

Avec Err_i le taux d'erreur dont nous avons donné l'expression précédemment.

2.5 Regression Logistique Multinomiale

2.5.1 Principe

Modèle La régression logistique modélise la probabilité qu'a un individu d'appartenir à une classe en fonction de la valeur de ses prédicteurs. La régression logistique utilise la "fonction logistique" pour déterminer ainsi cette probabilité :

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}$$

Estimation des coefficients La détermination de cette fonction logistique repose sur l'estimation des coefficients $\beta_0, ...\beta_p$ de cette dernière via la méthode du maximum de vraissemblance.

Cas multi-classe La regression logistique est usuellement utilisée dans le cas binaire, c'est à dire lorsque l'on cherche à déterminer si un individu appartient, ou n'appartient pas à une classe. Cependant elle peut être utilisée dans le cas multinomial, c'est pourquoi nous avons décidé de l'utiliser au cours de ce TP.

2.5.2 Résultats expérimentaux

Méthode Nous avons utilisé la méthode multinom du package "nnet" afin de réaliser l'entrainement du modèle (fichier "multinomial_logreg.R").

Modèle Souhaitant tout d'abord avoir une vue d'ensemble des performances de plusieurs classifieurs, nous avons choisi pour ce premier essai de nous en tenir au modèle :

$$Y \sim \beta_0 + \beta_1 X_1 + ... + \beta_{256} X_{256}$$

Performances L'estimation du taux d'erreur de ce classifieur par la méthode des 10-fold Cross Validation nous donne le résultat suivant.

TABLE 2.2 – Performances de la regression logistique multinomiale

$CV_{10}(mult.log.reg.)$	9.14%

2.6 k plus proches voisins

2.6.1 Principe

knn La méthode des k plus proches voisins (ou knn) attribue à l'individu considéré la classe la plus représentée parmis les k plus proches voisins de l'individu en question.

Modèle La méthode des knn peut s'exprimmer comme suit :

$$\hat{f}(x_0) = \frac{1}{k} \sum_{x_i \in \mathcal{N}_0} y_i$$

Avec x_0 l'individu de test, x_i les individus d'entrainement, y_i la classe des individus d'entrainement, et $\mathcal{N}_0$ l'ensemble des k individus les plus proches de x_0.

2.6.2 Résultats expérimentaux

Script Notre script est disponible en tant que "multinomial_knn.R"

Méthode Cette fois ci, il a été nécessaire d'optimiser la valeur de k pour laquelle le taux d'erreur de classification était optimal. Nous avons de nouveau utilisé la méthode de 10-folds Cross Validation, afin d'estimer au mieux la valeur de notre taux d'erreur pour chaque valeur du paramètre k

Performances La figure 2.1 ci-après représente la courbe de performance du classifieur, c'est à dire son taux d'erreur en fonction de la valeur de k considérée.

Interprétation Nous pouvons voir que les performances du classifieur en question s'améliorent initialement rapidement en augmentant la valeur de k pour atteindre leur optimum pour $k = 13$, avant de se stabiliser pour des valeurs plus élevées de k

TABLE 2.3 – Performances optimum de la méthode des KNN

$CV_{10}(knn, k = 13)$	8.43 %

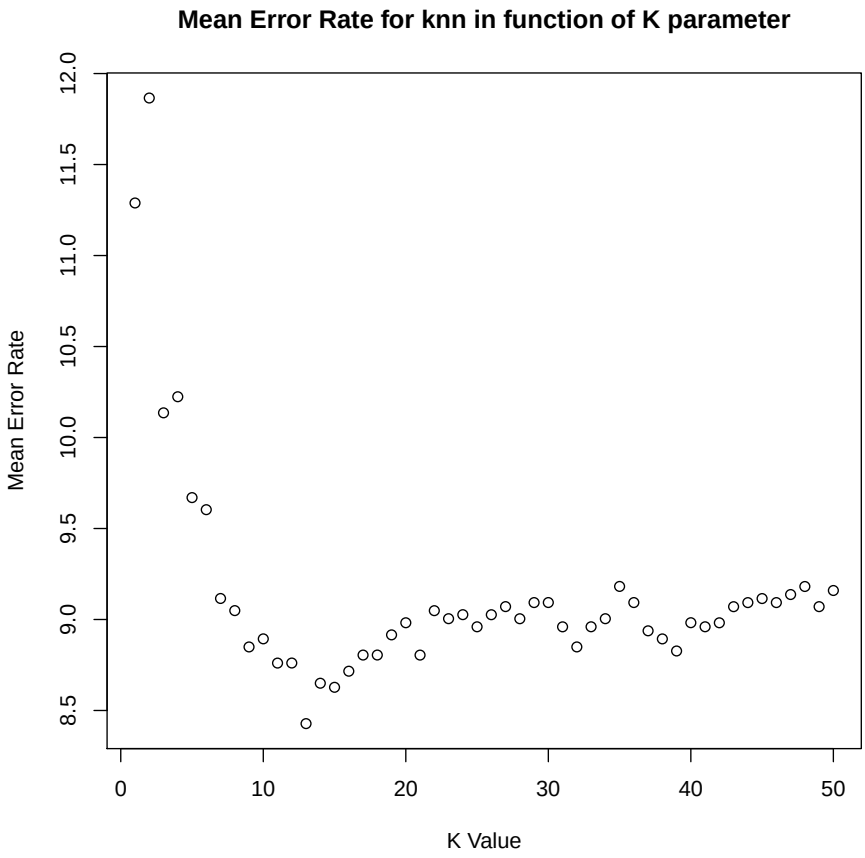

FIGURE 2.5 – Performances du classifieur "KNN" en fonction de la valeur de K

2.7 Analyse Discriminante Linéaire - LDA

2.7.1 Principe

LDA - Généralités L'Analyse Discriminante Linéaire (ou LDA), à l'image de la régression logistique, consiste à essayer d'estimer la probabilité d'appartenance d'un individu à une classe. La différence principale entre ces deux méthodes réside dans le fait que via la LDA, nous essayons de déterminer quelle classe présente la **probabilité a posteriori** le plus élevée, calculée en prenant notamment en compte les **probabilités à priori** de chacune d'entre elles.

LDA - Hypothèses La LDA assume vraies les hypothèses suivantes.

1. Les distributions des classes sont Gaussiennes
2. Ainsi la distribution de $X = (X_1, X_2, ...X_p)$ est Gaussienne Multi-variée
3. Les classes ont des moyennes μ_k différentes mais une matrice de covariance Σ commune
4. On peut estimer π_k, les probabilités à posteriori des différentes classes

Ainsi une observation $X = x$ est assignée à la classe pour laquelle la valeur de :

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + log \pi_k$$

est la plus grande.

2.7.2 Résultats expérimentaux

Méthode Une nouvelle fois, en appliquant la méthode de 10-folds Cross Validation à l'apprentissage d'un classifieur de type LDA, au sein de notre script "multinomial_lda.R", nous obtenons le taux d'erreur suivant.

TABLE 2.4 – Performances du classifieur LDA

$CV_{10}(LDA)$	7.43 %

2.8 Analyse Discriminante Quadratique - QDA

2.8.1 Principe

QDA - Généralités L'Analyse Discriminante Quadratique (ou QDA) repose sur les mêmes principes que l'Analyse Discriminante Linéaire. La principale différence entre les deux méthodes vient des hypothèses assumées par cette dernière méthode.

QDA - Hypothèses L'QDA assume vraies les hypothèses suivantes.

1. Les distributions des classes sont Gaussiennes
2. Ainsi la distribution de $X = (X_1, X_2, ...X_p)$ est Gaussienne Multi-variée
3. Les classes ont des moyennes μ_k différentes
4. **Les classes ont une matrice de covariance Σ différente**
5. On peut estimer π_k, les probabilités à posteriori des différentes classes

Ainsi une observation $X = x$ est assignée à la classe pour laquelle la valeur de :

$$\delta_k(x) = -\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}log|\Sigma_k| + log\pi_k$$

est la plus grande.

2.8.2 Résultats Expérimentaux

Méthode Renouvellant l'expérience précédente, appliquant la méthode de 10-folds Cross Validation à l'apprentissage du classifieur QDA au sein de notre script "multinomial_qda.R", nous obtenons pour ce dernier le taux d'erreur suivant.

TABLE 2.5 – Performances du classifieur LDA

$CV_{10}(QDA)$	13.15 %

2.9 Première phase de sélection de modèle - Bilan

Récapitulatif Le tableau ci-dessous récapitule les performances obtenues jusqu'à maintenant avec les différents classifieurs.

TABLE 2.6 – Performances des différents classifieurs

$CV_{10}(MULT.LOG.REG.)$	9.14 %
$CV_{10}(KNN, K = 13)$	8.43 %
$CV_{10}(LDA)$	7.43 %
$CV_{10}(QDA)$	13.15 %

Modèle - Interprétation Ces résultats sont obtenus en utilisant à chaque fois l'ensemble des prédicteurs, sans leur appliquer la moindre transformation, c'est à dire en prenant en paramètre à chaque fois le modèle suivant :

$$Y \sim \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_{256} X_{256}$$

Suite de l'étude Afin d'approfondir notre étude, nous avons par la suite décidé d'appliquer des transformations à notre ensemble de prédicteurs, afin d'étudier les conséquences que cela pouvait avoir sur les performances de nos classifieurs.

Choix de la LDA Nous décidons d'appliquer ces transformations au modèle en séléctionnant pour la suite de n'utiliser qu'un classifieur de type LDA, nous ayant fourni les meilleures performances de classification jusque là.

2.10 Forward Stepwise Selection

2.10.1 Principe et motivations

Motivations Dans un premier temps, nous aimerions réduire le nombre de prédicteurs utilisés pour notre classification. En effet, le dataset présente une dimension $p = 256$ très élevée d'espace de prédicteurs (= nombreux inputs). Déterminer un nombre restreint de prédicteurs suffisants pour obtenir de bonnes performances de classification nous simplifierait la vie pour la suite de nos expériences.

Principe La Forward Stepwise Selection consiste à partir du modèle nul, et y ajouter un à un les prédicteurs améliorant le plus les performances du classifieur (en les testant tous un par un). Une fois tous les prédicteurs ajoutés, étudier par Cross Validation les performances des meilleurs modèles successifs, afin de choisir le subset de prédicteurs nous semblant le plus adapté, en prenant en considération non seulement les performances du classifieur, mais également la dimension de l'espace de prédicteurs de laquelle nous essayons de nous rapprocher (généralement la plus petite possible).

2.10.2 Application

Script Nous avons codé notre propre méthode de Forward Stepwise Selection, disponible dans le fichier "stepwise_selection.R" de notre TP.

Méthode & Résultats Pour des raisons de performances, nous n'avons pas fait tourner l'algorithme de Forward Stepwise Selection jusqu'au bout (c'est à dire jusqu'à l'ajout du dernier des 256 prédicteurs au modèle). Au lieu de cela nous l'avons arrêté à 40 itérations, ayant remarqué que passé environ vingt itérations, les performances de notre classifieur ne s'amélioraient plus suffisamment. Nous obtenons le meilleur taux d'erreur pour le modèle ci-dessous. La figure ci-après représente l'évolution des performances de notre classifieur LDA en fonction du nombre de prédicteurs pris en compte.
Modèle : y $\sim$ X19 + X4 + X39 + X66 + X10 + X31 + X156 + X73 + X45 + X12 + X254 + X102 + X34 + X57 + X80 + X5 + X23 + X233

TABLE 2.7 – Paramètres optimum

Taux d'erreur optimal	7.52 %
Nombre de prédicteurs optimal	18

Interprétation Notons que nous atteignons un taux d'erreur de seulement 7.52%, en utilisant seulement 18 de nos prédicteurs sur 256 ! Cela confirme notre hypothèse initiale, prévoyant de bonnes performances des méthodes de

réduction de dimension, et égale quasiment les performances du classifieur LDA obtenu en utilisant les 256 prédicteurs en input !!!

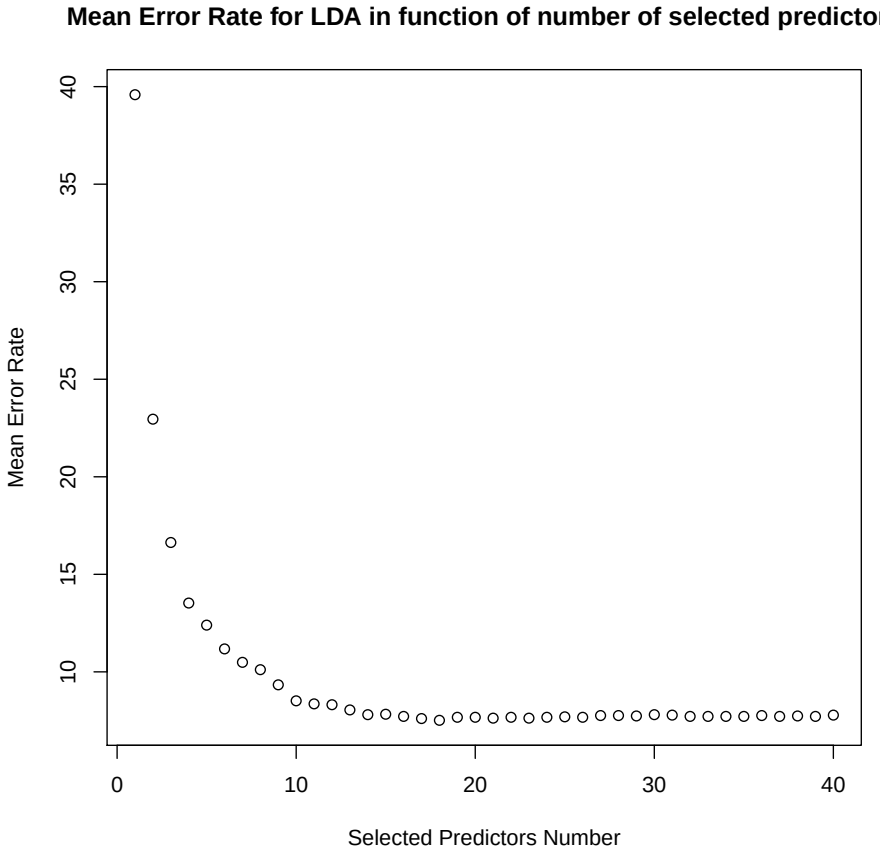

FIGURE 2.6 – Performances du classifieur en fonction du nombre de predicteurs séléctionné par la Forward Stepwise Selection

2.11 Aller plus loin

2.11.1 Perspectives d'amélioration

Perspectives Malheureusement, contrairement à ce que nous avions annoncé en introduction, nous n'avons pu implémenter autant de méthodes de classification que nous aurions aimé. Voici cependant quelques unes des études que nous aurions aimé mener.

1. Appliquer de plus nombreuses transformations à notre modèle : nous nous en sommes tenus à un modèle linéaire, sur lequel nous avons basé nos classifieurs. Nous aurions aimé avoir le temps de tester des modèles quadratiques, appliquer des transformation à certains de nos prédicteurs (en utilisant par exemple le logarithme, ...), ou tenter de créer des synergies entre ces derniers. Toujours dans le but de mesurer l'impact sur les performances de nos classifieurs
2. Implémenter à la main une stratégie "1 vs All" : optimiser individuellement les classifieurs binaires pour chacune de nos classes aurait été enrichissant. Nous aurions par exemple aimé tracer au moins une courbe ROC (pour le sport).
3. Essayer de faire de la classification à l'aide des composantes principales, plutôt qu'avec les prédicteurs bruts

Chapitre 3

Prédictions sur des données relatives au cancer du sein

3.1 Caractérisation du dataset

Présentation Les données représentent différentes mesures sur des patients présentant des cas de cancer du sein. Ces mesures sont présentées dans le data set en tant que Moyennes, Standard Error, et "Pires" valeurs, afin de réduire la quantité de données.
On possède ainsi 30 valeurs différentes (qui correspondent à chaque donnée mesurée) pour chaque patient. Le temps de récurrence ou sans maladie nous est fourni aussi ainsi que le statut du ganglion lymphatique. Il est évident, à première vue, que ces données sont corrélées. En effet, on possède par exemple le rayon, périmètre et aire pour chaque cellule des mesures.

Forme Les données fournies sont très hétérogènes et ne permettent pas une analyse purement visuelle. De plus, étant corrélées, un modèle direct ne peut être effectué sans marge d'erreur importante.

La figure 3.1 montre bien la relation entre l'aire, le périmètre et le rayon des cellules.
On peut, de même, voir qu'aucun prédicteur ne présente une relation directe avec le Temps, on ne peut donc pas extraire un modèle direct.

Objectifs L'objectif de cette analyse est de trouver un modèle le plus précis possible afin de prédire le temps de récurrence du cancer du sein. On utilise plusieurs variables relatives à des mesures sur différents patients, ce qui implique une analyse dimensionnelle et relative des données. Cette analyse permettra d'extraire les données les plus importantes, et un modèle relativement précis.

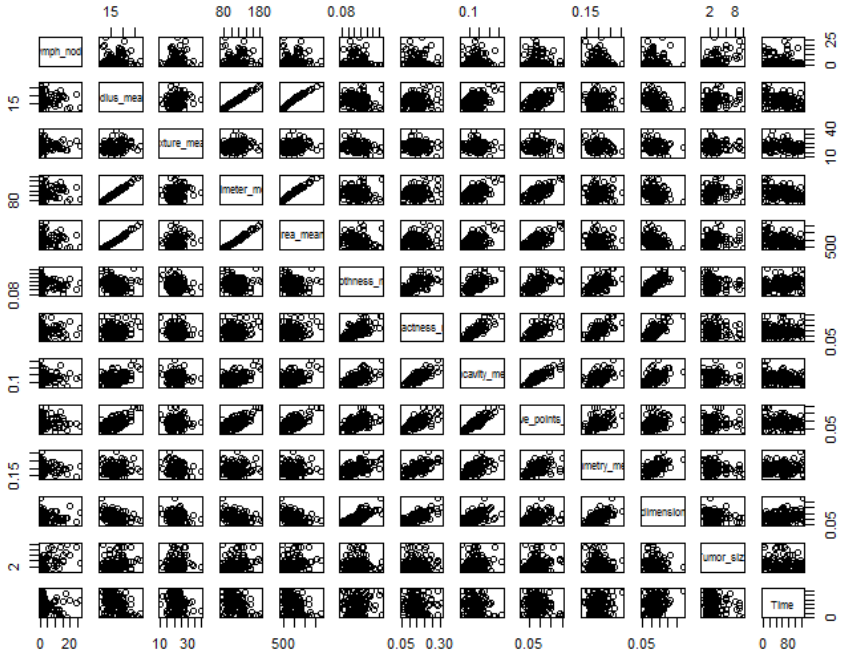

FIGURE 3.1 – Graphique de relation entre les valeurs moyennes de toutes les variables

3.2 Régression linéaire - Caractérisation du problème

3.2.1 Introduction

Une méthode courante pour la prédiction de données est la Régression Linéaire. Cette méthode suppose un modèle linéaire entre les prédicteurs et la variable à prédire. On peut poser par exemple :

$$V = \beta + \sum_{i=0}^{k} \beta_i P_i$$

Où P_i sont les prédicteurs, $_i$ les poids de chaque predicteur, et k le nombre de prédicteurs, avec β une valeur de base (parfois β_0.
Ce modèle n'est cependant pas adapté à tous les problèmes, qui nécessitent parfois une formule multinomiale, ou un modèle qui prend en compte la dépendance des variables entre elles.

3.2.2 Taux d'Erreur

Dans notre cas, on utilisera les résidus comme témoins de l'erreur du modèle. En effet, les r^2 définis comme :

$$r^2 = \sum_{i=0}^{n} \frac{(y - \hat{y})^2}{n}$$

permettent de comprendre l'erreur du modèle. Ainsi, la comparaison de chaque r^2 pour chaque modèle nous permettra, de manière relativement simple, de choisir le modèle le plus approprié.

3.2.3 Stratégie

Si l'on applique un modèle linéaire direct à nos données (dépendance de Time sur la somme des 32 autre variables différentes), le modèle ne peut prédire correctement le temps de récurrence.

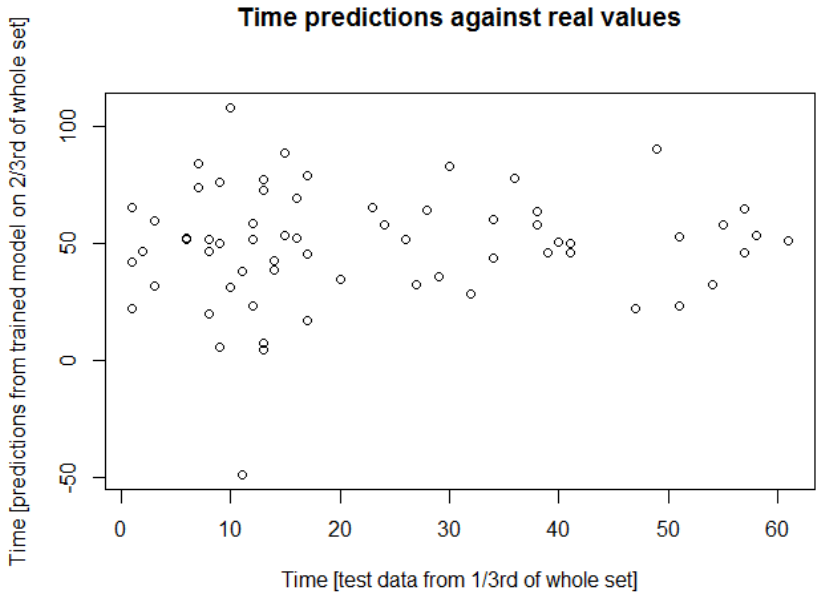

FIGURE 3.2 – Prédictions du modèle contre les valeurs réelles

Si le modèle était capable de prédire les données correctement, le résidu serait minimum, ce qui implique que la courbe précédente devrait tendre vers une fonction linéaire (c'est-à-dire que les valeurs prédites sont pratiquement

égales aux valeurs réelles). Ce n'est pas le cas du modele qui comprend toutes les donnees.

On doit donc modifier le set de données. Plusieurs méthodes existent, dont la Subset Selection, qui consiste à effectuer des méthodes *stepwise*, qui peuvent être *forward stepwise* ou *backward stepwise*, la méthode de Régularisation, ou encore les méthodes de Diminution de Dimensions. Dans ce TP on appliquera les trois méthodes, en en précisant le r^2 donné pour chacune.

3.3 Subset Selection

3.3.1 Principe

Une fois explorée la méthode directe, il est évident que le choix spécifique des prédicteurs est nécessaire. On utilisera donc une méthode de Subset Selection, qui consiste à choisir les prédicteurs un à un afin de trouver le modèle le mieux adapté. Il existe différentes échelles de mesure pour la sélection de subset, dans lesquelles on trouve : BIC, R^2, Cp et $adjustedr^2$ (r^2 ajusté). Elles représentent chacune un calcul différent pour l'erreur du modèle, pour la plupart la valeur la plus petite est celle qui représente le modèle le plus adapté, cependant pour r^2 et $adjustedr^2$ c'est la plus grande.

- $Cp = \frac{RSS}{\hat{\sigma}^2} - N + 2d$
- $AIC = -2l + 2d$
- $BIC = -2l = dlog(N)$
- $AR^2 = 1 - \frac{VAR_res}{VAR_tot}$

On peut alors éxécuter un algorithme de sélection de subset et dessiner le graphique de chaque modèle, ce qui nous permettra de définir les prédicteurs à utiliser. L'algorithme consiste à créer un modèle sans prédicteur ou avec tous les prédicteurs et les ajouter *(forward stepwise)* ou à les enlever *(backward stepwise)* afin de trouver le modèle présentant le taux d'erreur (en dépendant des échelles présentées ci-dessus) le plus petit.

3.3.2 Résultats Expérimentaux

Pour cela, on utilise la fonction *regsubsets* du package *leaps*, qui éxécute une recherche exhaustive (*forward* et *backward*) dans le set de données. Cette fonction retourne les différentes échelles de mesure pour chaque modèle testé et prend en paramètre une formule, pour laquelle on utilisera une somme comme dans la formule précédente.

```
regsubsets(formula, data=train_data, method='exhaustive', nvmax=NULL, nbest=1)
```

Les résultats pour les 2/3s du set entier (notre *training set*) sont les histogrammes dans la figure 3.9.
Les résultats nous permettent l'extraction de 4 formules différentes pour la régression linéaire :

- `Time~.`
- `Time ~ radius_mean + perimeter_mean + smoothness_mean + symmetry_mean + fractal_dimension_mean + radius_se + texture_se + perimeter_se + area_se + smoothness_se + symmetry_se + smoothness_worst + Tumor_size`
- `Time ~ radius_mean + perimeter_mean + fractal_dimension_mean`
- `Time ~ radius_mean + perimeter_mean + fractal_dimension_mean + texture_se`

3.3.3 Cross-Validation

Motivations La méthode des subset selection nous permet de reconnaître des modèles avec des grandes possibilités de prédiction, cependant cela reste relatif aux données de training. Avec cette méthode il est fort possible qu'il existe de l'overfitting, et donc que notre modèle ne puisse finalement pas prédire les résultats sur d'autres sets de données.
Afin de palier à ce problème, on éxécute un algorithme de Cross-Validation.

Principe L'algorithme de Cross-Validation prend un modèle donné, et l'applique à un set de données en le découpant, ou pliant, en k morceaux. On répète cette application pour chacun des k morceaux afin de valider le modèle en sous-divisant le set en training et test.

Résultats On peut ainsi procéder à la Cross-Validation, en effectuant k plis dans le set de données. De manière pratique, on impose $k = 5$, et on effectue le calcul 10 fois pour chaque méthode. On obtient ainsi un histogramme présentant les résultats de chaque CV, et de même, un tableau récapitulatif :

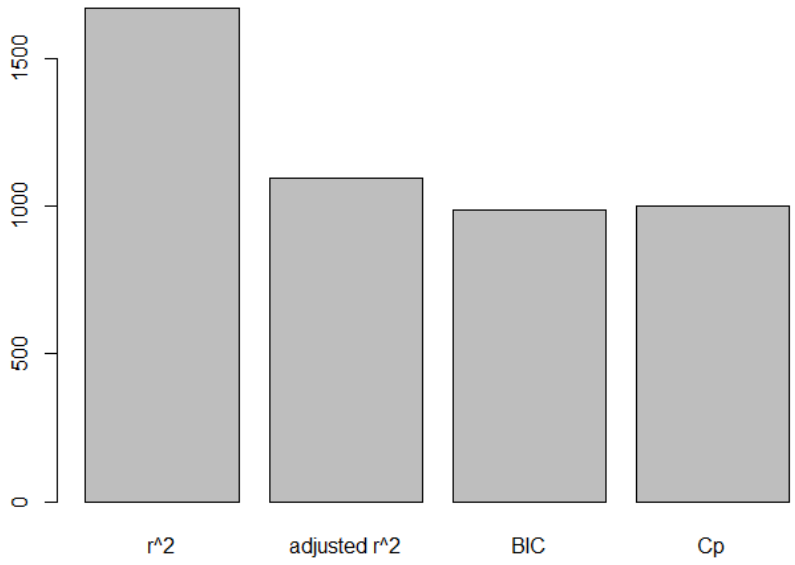

FIGURE 3.3 – Exemple de valeurs de CV pour chaque methode

r^2	$adjustedr^2$	BIC	Cp
1428.185	1010.8581	1021.189	998.0554
1570.229	1058.9614	1021.189	998.0554
1570.229	1058.9614	1021.189	998.0554
1386.481	980.5896	1021.189	998.0554
1637.932	1012.9833	1021.189	998.0554
1610.227	1096.5843	1021.189	998.0554
1727.982	1015.1380	1021.189	998.0554
1489.949	1018.1485	1021.189	998.0554
1667.519	1092.2323	1021.189	998.0554
1740.686	1065.5497	1021.189	998.0554

Et qui présentent ainsi en moyenne :

r^2	$adjustedr^2$	BIC	Cp
1582.942	1041.001	1021.189	998.055

Grâce à ces résultats on aurait tendance à conserver le modèle donné par le calcul de Cp.

$$\text{Cp} \quad r^2 = 998.055$$

3.4 Régularisation

3.4.1 Principe

Le modèle conservé avec la méthode de subset selection est celui de :

```
Time ~ radius_mean + perimeter_mean + fractal_dimension_mean + texture_se
```

Cependant, puisqu'il a été calculé à partir de petits sets de données, il se peut qu'il ne possède pas la meilleure capacité de prédiction. Afin de palier à ce problème on peut utiliser des méthodes de régularisation, telles que Ridge et le Lasso.

Le principe des méthodes de régularisation est de réduire les poids de chaque prédicteur en pénalisant leur taille. Cependant, la méthode de Ridge garde tous les p prédicteurs dans le modèle, alors que la méthode du Lasso permet de se passer de ceci. En effet, le Lasso est capable de réduire les poids de quelques prédicteurs à 0, étant donné un λ assez grand.

3.4.2 Résultats Expérimentaux

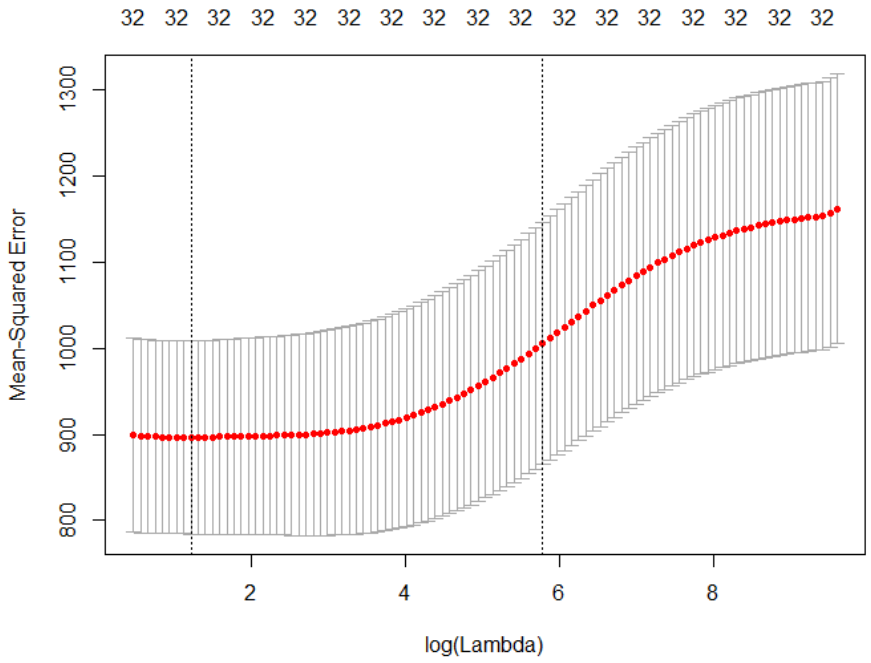

FIGURE 3.4 – CV error as function of λ (Ridge)

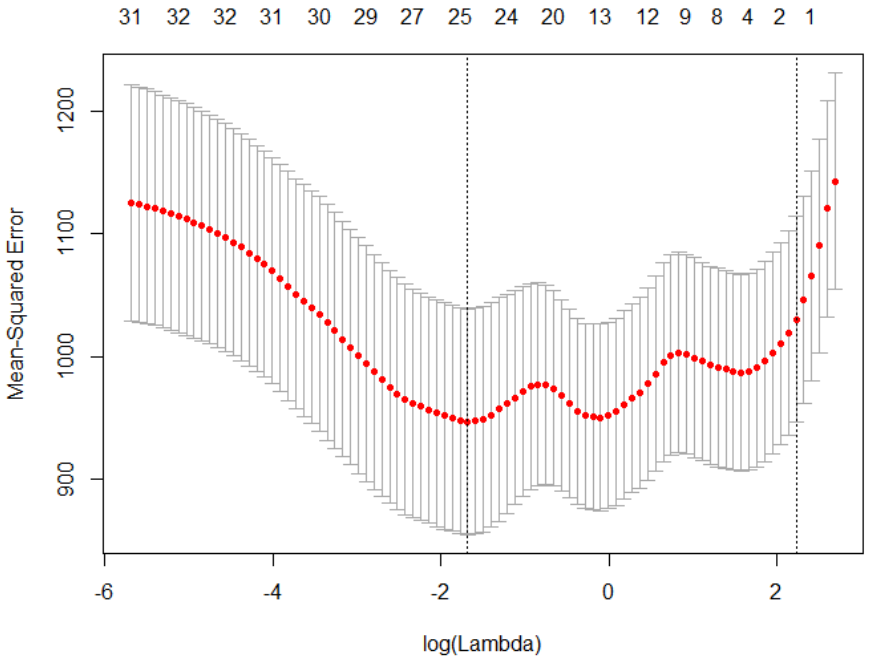

FIGURE 3.5 – CV error as function of λ (Lasso)

La méthode de Ridge nous permet de produire un $r^2 = 1333.044$ et celle du Lasso $r^2 = 1578.521$

Ridge $r^2 = 1333.044$
Lasso $r^2 = 1578.521$

3.5 Principal Component Analysis - PCA

3.5.1 Principe

Le PCA (Analyse de Composant Principal), est une méthode utilisée afin de trouver les composantes principales dans un set de prédicteurs. Elle est utilisée souvent quand les données sont très correlées entre elles, et on doit procéder à une réduction de dimension.

On peut, en effet, voir dans l'image 3.1 que les variables possèdent une certaine corrélation. Il serait donc judicieux d'éliminer les variables ne permettant pas d'expliquer toute la variation.

Principe La méthode de PCA consiste à transformer les prédicteurs X_i en de nouveaux prédicteurs. Dans la même idée que le Lasso, le but est de réduire l'importance des facteurs (ou la dimension) tout en conservant l'information permettant de prédire Y_i. Pour le PCA le but est de trouver des vecteurs horthogonaux dans l'espace des prédicteurs, ce qui implique une variance maximum et ainsi des combinaisons linéaires des X_i qui permettent de trouver le maximum d'informations.

3.5.2 Résultats Expérimentaux

Comme on a pu le voir dans la première figure, les prédicteurs de nos données sont souvent corrélés, et puisque la plupart viennent des mêmes mesures, possèdent souvent une relation directe. Afin d'effectuer la PCA, on utilisera *princomp* dans R :

```
> summary(pca)
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4    Comp.5     Comp.6     Comp.7     Comp.8     Comp.9    Comp.10    Comp.11
Standard deviation     3.1572035 2.8557779 1.8764471 1.54643893 1.2578441 1.21000513 1.08635482 0.93920451 0.87429388 0.7194753 0.69096025
Proportion of Variance 0.3020586 0.2471354 0.1066986 0.07246889 0.0479446 0.04436704 0.03576263 0.02673046 0.02316333 0.0156862 0.01446746
Cumulative Proportion  0.3020586 0.5491940 0.6558926 0.72836146 0.7763061 0.82067310 0.85643574 0.88316619 0.90632952 0.9220157 0.93648318
                         Comp.12   Comp.13    Comp.14    Comp.15     Comp.16     Comp.17     Comp.18     Comp.19     Comp.20     Comp.21
Standard deviation     0.68266695 0.5823696 0.532558500 0.48822161 0.413301552 0.353876388 0.303181688 0.282513201 0.253136586 0.233059052
Proportion of Variance 0.01412225 0.0102774 0.008594502 0.00722304 0.005176308 0.003794803 0.002785428 0.002418597 0.001941762 0.001645955
Cumulative Proportion  0.95060543 0.9608828 0.969477333 0.97670037 0.981876681 0.985671484 0.988456913 0.990875510 0.992817272 0.994463227
                         Comp.22     Comp.23     Comp.24     Comp.25     Comp.26     Comp.27     Comp.28     Comp.29     Comp.30
Standard deviation     0.20899234 0.1813999191 0.1670154211 0.1381269429 0.1286147455 0.1157082554 0.1034620116 0.0955407668 0.0758853700
Proportion of Variance 0.00132357 0.0009971494 0.0008452773 0.0005781531 0.0005012652 0.0004057091 0.0003243754 0.0002766072 0.0001745027
Cumulative Proportion  0.99578680 0.9967839458 0.9976292231 0.9982073762 0.9987086415 0.9991143506 0.9994387260 0.9997153332 0.9998898359
                          Comp.31      Comp.32      Comp.33
Standard deviation     4.889791e-02 3.146767e-02 1.594351e-02
Proportion of Variance 7.245472e-05 3.000649e-05 7.702896e-06
Cumulative Proportion  9.999623e-01 9.999923e-01 1.000000e+00
```

FIGURE 3.6 – Summary de la fonction *princomp*

Grâce à cette analyse, on remarque que les 12 premiers composants permettent d'expliquer 95% de la variation. On peut ainsi construire un modèle à partir de ces composants.

Ce dernier génère des prédictions avec un

$$r^2 = 911.2256$$

qui est la meilleure valeur de r^2 parmi les méthodes analysées.

En utilisant la fonction *pcr*, on peut dessiner la courbe qui représente le MSE en fonction de M, le nombre de prédicteurs avec $M < p$.

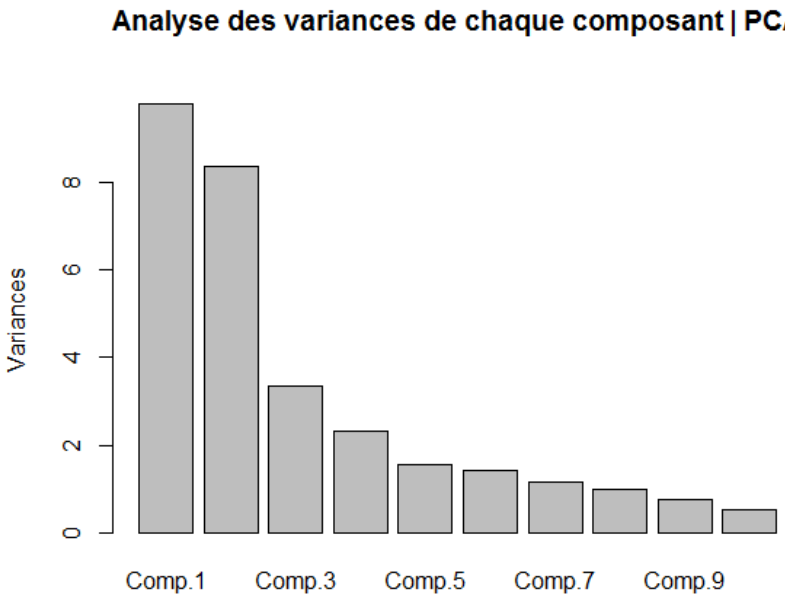

FIGURE 3.7 – Résultat de l'analyse par PCA des données

3.6 Sélection du modèle

Récapitulatif On peut rappeler les résultats des différents modèles dans le tableau suivant

TABLE 3.1 – Performances des différents modeles (r^2)

Cp	998.055 %
Ridge	1333.044 %
Lasso	1578.521 %
PCA	911.2256 %

Interprétation Les modèles présentès dépendent de différents prédicteurs. Il est donc difficile de les comparer directement. Cependant, la distance aux valeurs prédites du modèle PCA est inférieure à toutes les autres, ce qui lui permet de

se distinguer. Il aurait été possible de pousser l'étude plus loin, en analysant les données PCA pour différents composants, et ainsi obtenir différents modèles. Cependant, dans notre étude on choisit de garder ce modèle grâce à ce résultat.

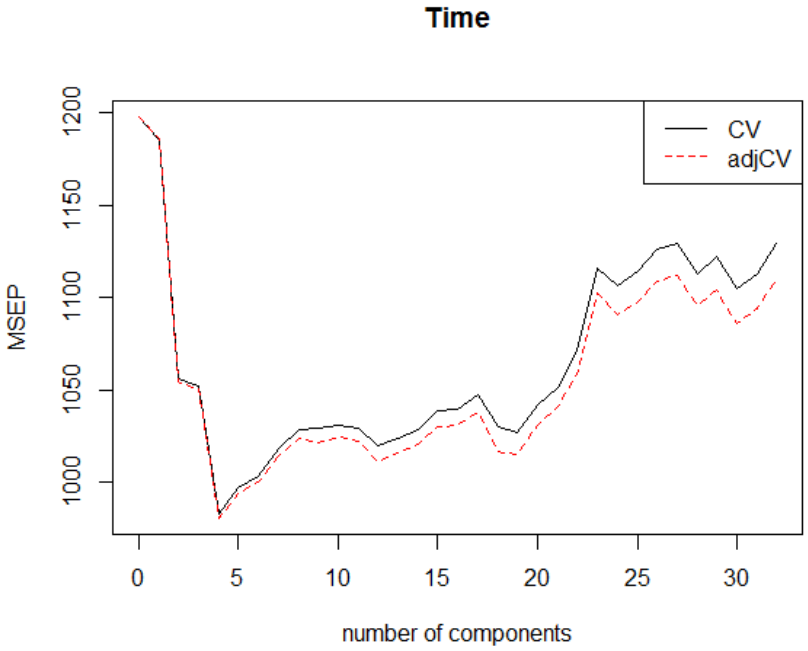

FIGURE 3.8 – MSE en fonction de M, $M < p$

3.7 Conclusion

Lors de ce TP, nous avons pu analyser deux sets de données complètement différents, et grâce à cela nous confronter à deux domaines très différents du Machine Learning, la Classification et la Régression, dans le but unique de prédire des résultats à partir de sets de données.

Ce TP a été l'occasion d'aborder différentes stratégies d'optimisation et de sélection de modèles prédictifs, telles que les méthodes du Lasso, la Ridge Regression, ou encore les méthodes de réduction de dimensions.

Nous nous sommes également familiarisés avec les concepts importants nous permettant de mesurer les performances de nos classifieurs et modèles de régression, afin de justement choisir ceux qui conviennent le mieux à nos besoins.

Finalement, ce TP a été une excellente introduction pour nous au monde du Machine Learning, à la régression, la classification et la sélection de modèle de manière générale, bien que légèrement rude, de par son aspect "autonomie totale", c'est-à-dire le fait qu'il a fallu pour nous découvrir tout cela "à tatons".

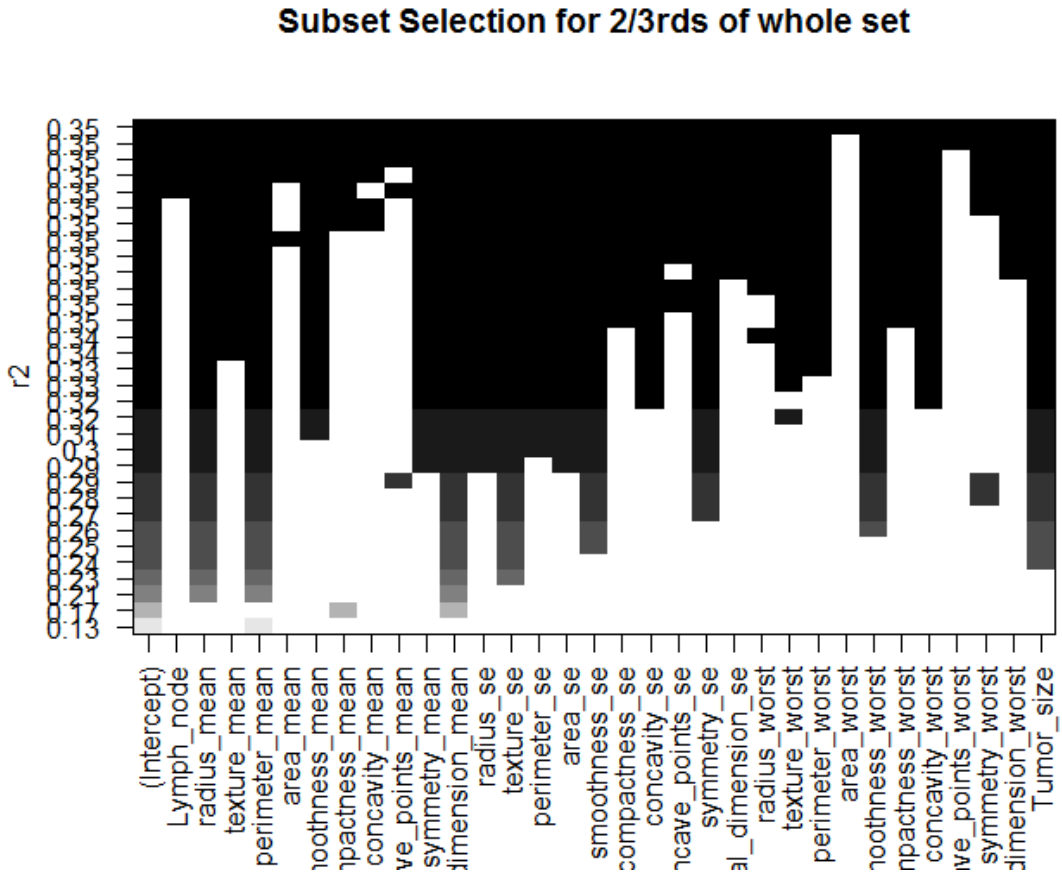

FIGURE 3.9 – Valeurs de r^2 pour la séléction du subset (r^2)

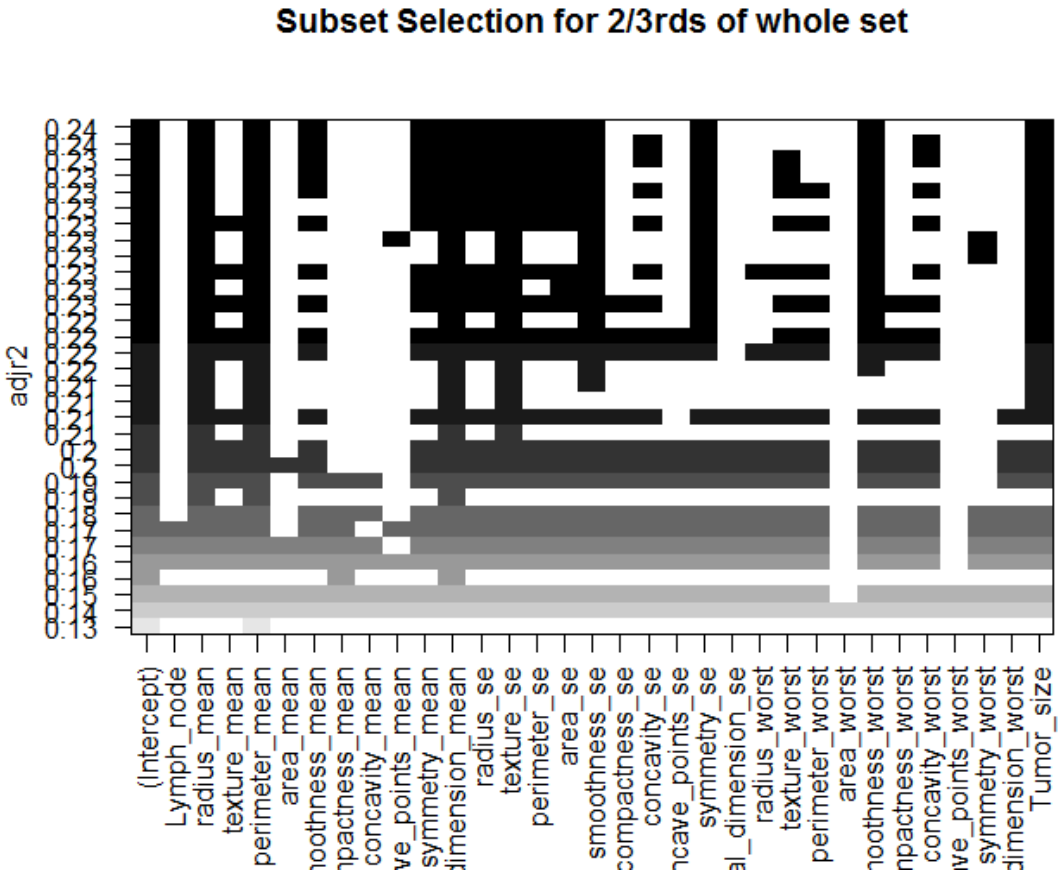

FIGURE 3.10 – Valeurs de r^2 pour la séléction du subset ($adjustedr^2$)

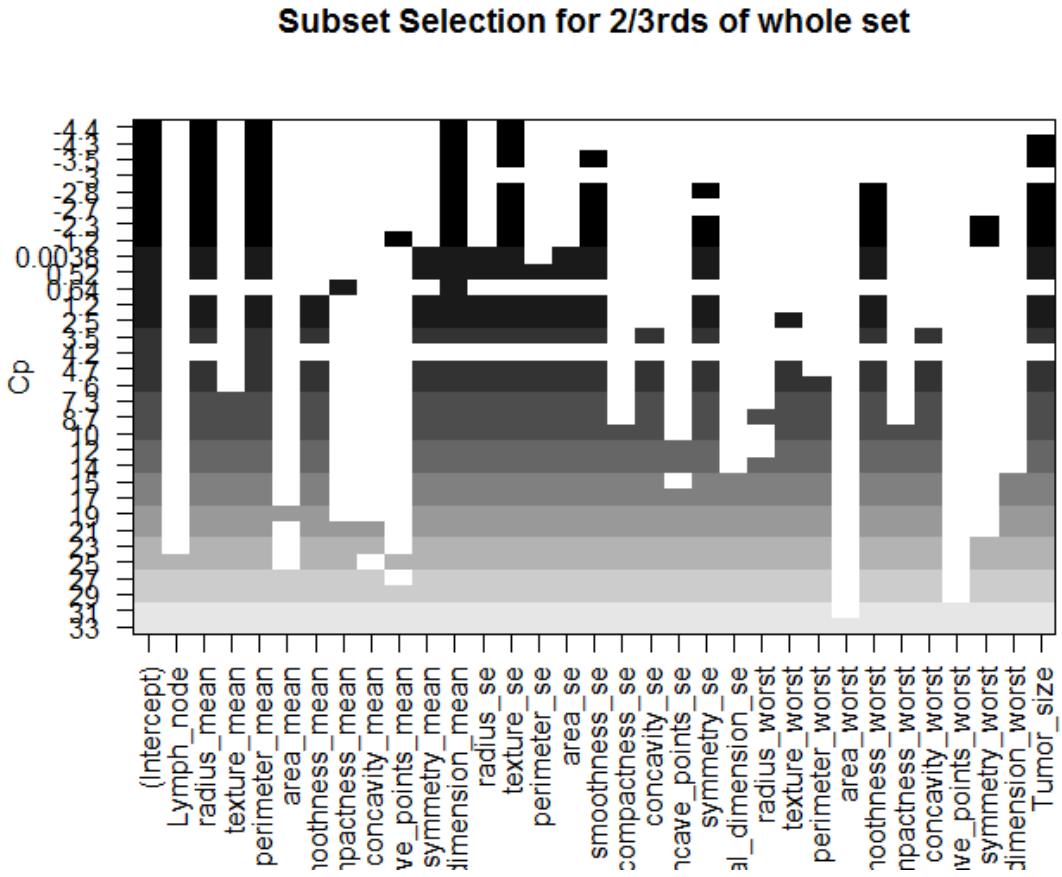

FIGURE 3.11 – Valeurs de r^2 pour la séléction du subset (Cp)

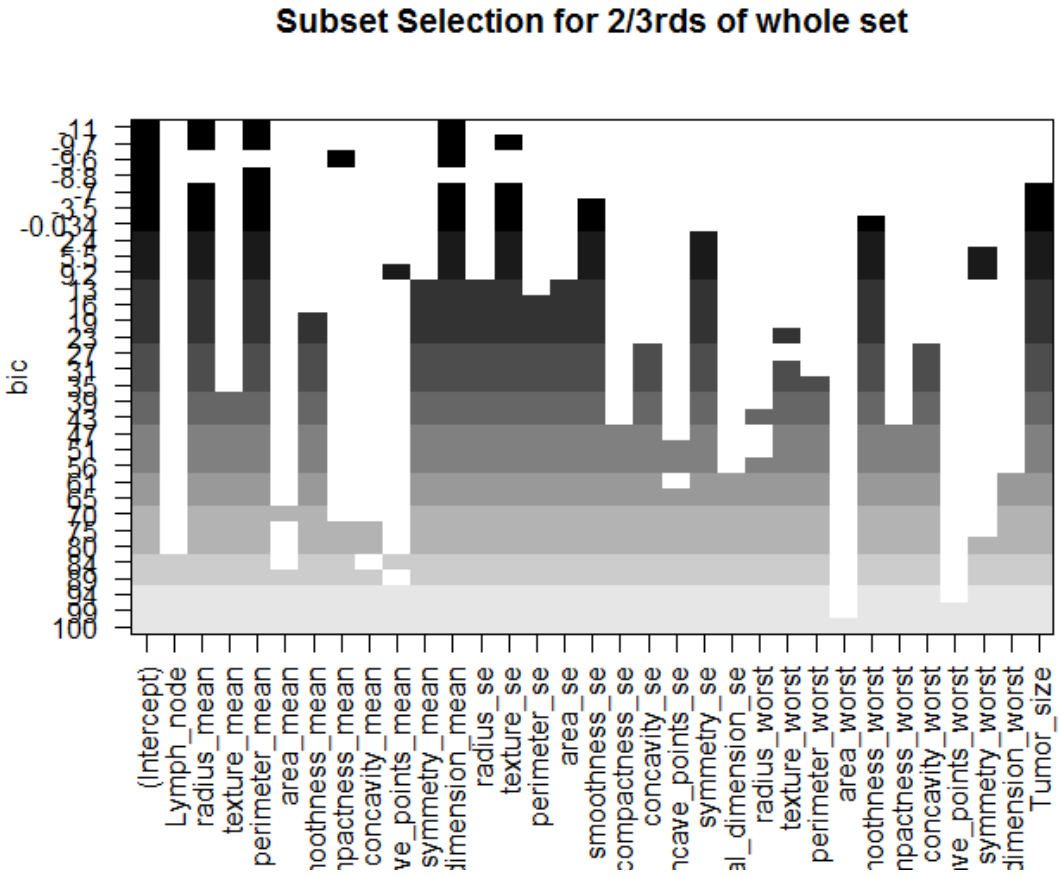

FIGURE 3.12 – Valeurs de r^2 pour la séléction du subset (BIC)