Université de technologie de Compiègne

Rapport TP4 - SY19

Régression et Classification - Sélection de modèles

24 NOVEMBRE 2016

Marceau LERICHE et Valentin PAUL

Contents

1 Introduction

Ce TP a pour but de mettre en pratique les différentes connaissances accumulées depuis le début de l'année à l'intérieur de cette UV pour deux problèmes bien distincts : un problème de régression et un problème de classification. De plus, nous allons mettre en place les différentes techniques vues et étudiées en cours sur la sélection des modèles afin de pouvoir trouver les meilleurs modèles pour chacun de nos problèmes. Ce rapport se divisera donc en deux parties correspondant à deux problèmes distincts ainsi qu'à deux jeux de données.

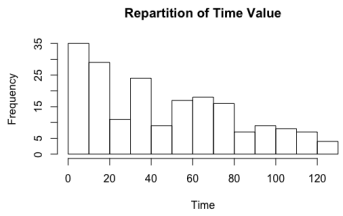
2 Les données Breast Cancer

2.1 Problématique et explication des données

Les données Breast Cancer utilisées correspondent à une analyse médicale de 194 individus ayant le cancer du sein. Pour chacun de ces individus, 33 paramètres ont été enregistrés tels que la taille moyenne de la tumeur mais aussi le nombre de jour que la cure a mise avant de rendre cette tumeur bénigne ou la faire disparaître totalement. Ainsi, la problématique que nous allons chercher à résoudre dans cette partie est de trouver le meilleur modèle d'estimation du nombre de jour qu'il faudra pour un nouveau patient avant d'être guéri via ce traitement. Ce problème est donc un problème de régression car nous allons essayer de prédire la valeur de la variable "Time" d'un nouvel individu en fonction des données que nous avons déjà collecté auparavant.

2.2 Analyse des données

Figure 1: Histogramme de "Time"



Avant de se lancer dans la recherche du meilleur modèle possible, il est intéressant de regarder d'un peu plus près le jeux de données que nous avons à disposition. Comme dit précédemment, celui-ci correspond à 194 individus et nous avons différents paramètres pour chaque individu. L'observation de la répartition de nos valeurs à prédire mais aussi des différentes corrélations entre chacune de nos variables explicatives peuvent être des informations très intéressantes à retenir. Lorsque l'on regarde la répartition de nos données en fonction du temps de guérison, nous pouvons constater que nous avons une répartition plutôt linéaire ainsi, on peut donc voir que nous avons beaucoup d'individus ayant une guérison rapide et de moins en moins avec une guérison lente allant jusqu'a 120 jours. Cette repartition inégale du nombre de données en fonction du temps de guérison est donc important car elle influencera notre modèle.

Si nous nous intéressons dorénavant aux variables qui nous serviront à construire notre modèle, il est cruciale de regarder la corrélation entre chacune de ces variables. Cette information est très importante pour la compréhension du choix de notre modèle finale car si certaines de nos variables sont corrélées alors, avoir les 2 variables pour prédire notre modèle n'est peut-être pas utile et peut même augmenter le taux d'erreur de ce modèle.

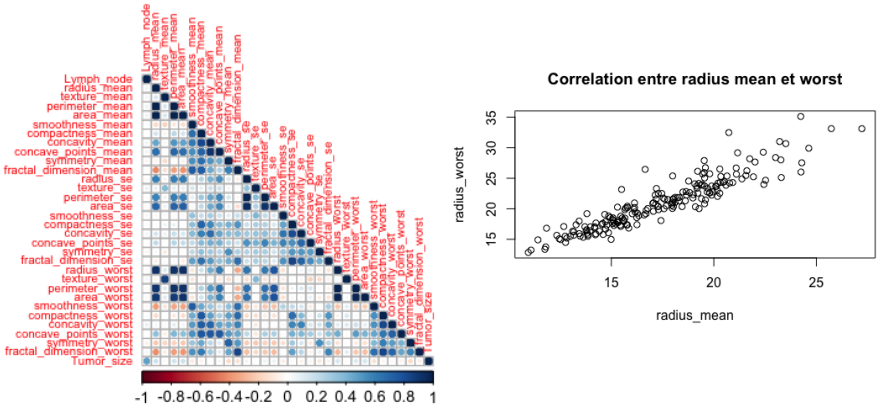


Figure 2: Observation des corrélations

On peut ainsi observer sur la matrice ci-dessus que certains de nos prédicteurs sont en effet corrélés, notamment le radius_mean et le radius_worst, comme le montre schéma de droite. Au vue de leurs noms, il semble logique d'avoir cette corrélation car nous avons certaines variables représentant la moyenne d'une valeur et une autre variable représentant la pire valeur prise... Il sera donc intéressant de voir si pour ces variables fortement corrélées, il ne serait pas plus correct de n'en sélectionner qu'une seule afin de prédire notre modèle final.

Pour finir,lorsque l'on regarde de plus près les intervalles de valeurs pris par certaines variables, on peut s'apercevoir que nous avons des échelles totalement différentes. Ainsi, afin d'accorder la même importance à chacun de nos prédicteurs, nous allons avant toutes choses centrer-réduire ceux-ci :

Variable	Min	1sr Qu.	Median	Mean	3rd Qu.	Max
Fractal_dimention_worst	0.001087	0.002753	0.003719	0.004004	0.004636	0.012560
area_worst	508.1	940.6	1295.0	1402.0	1694.0	3903.0

Malheureusement, ces différentes visualisations de données ne nous permettent pas d'identifier de manières précises les facteurs représentant au mieux le temps de guérison d'un patient. Ainsi, nous allons pousser notre analyse en comparant différents modèles obtenus via des méthodes de régression. Grâce à ces différents modèles, nous seront alors plus en mesure de déterminer le temps de guérison d'un nouveau patient.

2.3 Sélection des modèles

Avant de passer à l'explication de nos différents modèles, nous allons expliquer dans cette parties, les différentes étapes appliqués sur nos modèles et nos données. Chacune de ces 3 étapes a été réalisée pour chacun de nos modèles afin qu'ils soient évalués dans des conditions identiques.

2.3.1 Préparation des données

Une fois les données observées, nous les avons alors divisées en deux ensembles (un ensemble d'apprentissage correspondant à 80% de nos données et un ensemble de test correspondant aux 20% restants). Ainsi, la sélection du modèle se fera sur notre ensemble d'apprentissage. Chaque modèle sera alors évalué sur celui-ci et nous ferons dans un dernier temps un test final du meilleur modèle avec nos données de test afin de pouvoir évaluer la performance de celui-ci de manière non biaisée. Par ailleurs, nous avons préféré mettre le seuil à 80% car nous n'avons que peu d'individus pour apprendre notre modèle et de plus, la répartition des variables "Time" est linéaire décroissante. De ce fait, avec un seuil à 80%, nous avons plus de probabilités d'avoir un panel complet de nos données et ainsi avoir un modèle plus précis.

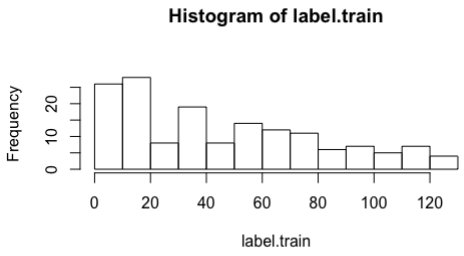


Figure 3: Répartition de nos données d'apprentissage

Une fois les données séparées, nous avons alors centré-réduit notre ensemble d'apprentissage, tout en récupérant les coefficients de centrage-réduction afin de pouvoir ensuite les utiliser pour centrer réduire notre ensemble de test en fonction. Voici ainsi le centrage réduction que nous avons effectué :

$$x^{app} = \frac{x^{app} - \mu^{app}}{\sigma^{app}}$$

$$x^{test} = \frac{x^{test} - \mu^{app}}{\sigma^{app}}$$

Pour finir, il est notamment important de préciser que nous avons réparti de manière aléatoire nos deux jeux de données et que ce jeux de tirage a été

effectué une unique fois pour tester l'ensemble de nos modèles. Ainsi, le choix ne sera pas influé par l'influence de plusieurs tirages aléatoire et sera fait en en fonction de l'erreur quadratique obtenue sur chacun de nos modèles.

2.3.2 Utilisation de la validation croisée

Afin de gérer la parcimonie de nos données, nous allons utiliser avec chacun de nos modèles la validation croisée. Il s'agit donc de découper le jeu de données en K groupes (dans notre cas, nous avons pris 10) tirés aléatoirement et qui sont alors successivement pris comme ensemble de test. Ainsi, nous pourrons avoir une estimation de l'erreur de nos modèles qui sera d'autant plus précise car nous calculerons non pas une seule erreur, mais une moyenne sur les 10 itérations, ce qui constituera alors l'estimateur de l'erreur de test par validation croisée. Afin de choisir notre modèle final, nous comparerons alors cet estimateur pour chacun de nos modèles et nous choisirons ainsi l'estimateur minimum.

2.3.3 Sélection de sous-ensembles

Ayant plusieurs variables corrélées dans notre jeu de données, nous avons voulu tester le plus grands nombres de modèles possible. Pour cela, nous avons décidé d'utiliser la méthode du "forward stepwise". Cette méthode commence avec un modèle n'ayant aucun prédicteur, puis ajoute des prédicteurs un à un au modèle jusqu'à ce que toutes nos variables soient présentes dans celui-ci. Nous avons notamment décidé de prendre cette méthode plutôt que d'autres du fait du nombre peu importants de données que nous avons et du nombre élevés de paramètres a prendre en compte dans notre modèle (pouvant donc aller jusqu'à 33 paramètres).

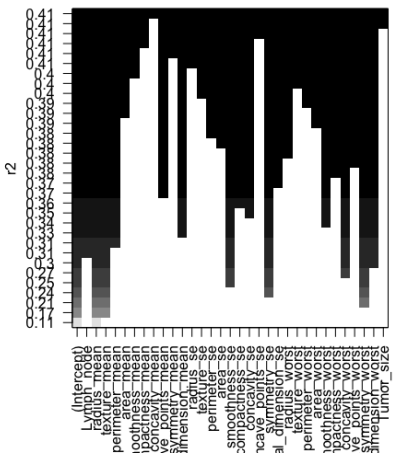


Figure 4: Graphe regsubset obtenu

Afin de pouvoir tester chaque modèle avec l'ordre d'ajout des variables dans

regsubset, nous avons créé une fonction récupérant cette ordre établie et créant ainsi les 33 différentes formules. Chaque formule tentera donc de prédire la valeur de la variable "Time" avec un nombre de prédicteurs allant de 1 à 33. Voici le code R de la fonction utilisée pour la création des formules :

```r
getFormulas <- function(col, order, label) {
  result <- vector(mode="character", length=length(order))
  for(i in 1:length(order)){
    if(i == 1){
      result[i] <- paste(c(as.character(label)," ~ ",col[order[i]]),collapse = '')
    } else {
      result[i] <- paste(c(result[i-1], col[order[i]]), collapse = ' + ')
    }
  }
  return(result)
}
```

Figure 5: Fonction R de création de nos formules

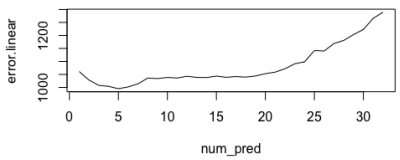
2.4 Les différents modèles utilisés

L'explication de la sélection des modèles étant faite, nous allons dans cette partie expliquer les trois différents modèles que nous avons testé afin de résoudre ce problème : la régression linéaire simple , la régression linéaire avec régularisation et la réduction de dimension avec le PCR (Principal Component Regression).

2.4.1 Régression Linéaire

```r
# Creation des partitions pour la crossValidation sous linear
folds = sample(1:10,data.train.dim[1], replace=TRUE)
for(i in (1:data.train.dim[2])){
  # 1 - Linear
  for(k in (1:10)){
    label.cv <- label.train[folds!=k]
    reg <- lm(formula = Formula[i],data=data.train[folds!=k,])
    pred <- predict(reg, newdata=data.train[folds==k,])
    error.linear[i] <- error.linear[i] + sum((label.train[folds==k]-pred)^2)
  }
  error.linear[i]<-error.linear[i]/data.train.dim[1]
```

(a) Régression Linéaire sous R



(b) MSE en fonction du nombre de prédicteurs

Figure 6: Observation de la regréssion linéaire

Dans un premier temps, nous avons décidé de tester une régression linéaire simple sur l'ensemble de nos formules. Ainsi, nous commençons donc avec une régression linéaire n'ayant qu'un seul prédicteur, puis nous les ajoutons un à un. Nous n'avons pas oublié d'utiliser la validation croisée pour calculer nos erreurs sur chacun de nos modèles et nous nous retrouvons ainsi avec le graphe ci-dessus qui représente l'erreur moyenne quadratique en fonction du nombre de prédicteurs. Comme nous pouvons le voir, plus le nombre de prédicteurs

est grand et plus le taux d'erreurs augmentent. Ceci est notamment du aux corrélations fortes que nous avons entre certaines de nos variables. Ainsi, le meilleur modèle obtenue par une régression linéaire simple est obtenue à l'aide de seulement cinq à sept prédicteurs tels que: "smoothness_se", "fractal_dimension_mean","area_se", "perimeter_mean", "symmetry_mean". (Le nombre de prédicteurs variant en fonction de la séparation des données qui est aléatoire).

```
Call:
lm(formula = FormulaTest[numberOfParameters], data = data.train)

Residuals:
    Min      1Q  Median      3Q     Max
-66.141 -22.013   0.316  20.957  73.429

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)               45.5097     2.4712  18.416  < 2e-16 ***
perimeter_mean           -24.5256    11.2211  -2.186  0.03051 *
texture_mean              -4.2572     3.0302  -1.405  0.16227
radius_worst              55.0946    21.4918   2.564  0.01142 *
symmetry_mean              5.7543     3.3853   1.700  0.09141 .
texture_se                -6.8577     3.7233  -1.842  0.06763 .
smoothness_se             17.4569     5.9196   2.949  0.00374 **
concavity_se              -9.5861     5.1710  -1.854  0.06588 .
fractal_dimension_se      -5.9413     6.4604  -0.920  0.35935
perimeter_worst          -29.9044    20.8072  -1.437  0.15290
concave_points_mean        0.9062     9.1492   0.099  0.92124
smoothness_worst         -17.1483     6.6224  -2.589  0.01064 *
fractal_dimension_worst   12.7678     7.2218   1.768  0.07926 .
area_se                  -22.9945    10.6637  -2.156  0.03278 *
perimeter_se              16.2650     9.8773   1.647  0.10188
smoothness_mean           11.6820     7.6862   1.520  0.13082
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 30.77 on 139 degrees of freedom
Multiple R-squared:  0.3208,    Adjusted R-squared:  0.2476
F-statistic: 4.378 on 15 and 139 DF,  p-value: 1.126e-06
```

Figure 7: Explication du modèle linéaire

Lorsque l'on utilise la régression linéaire sur l'ensemble de nos variables,on peut constater que la F-statistic est légèrement différente de 1, et la p-value est petite ($< 1.12e-06$). On peut donc conclure qu'il y a bien une relation entre la variable à expliquer (Time) et les autres variables, cependant nous avons tout de même une erreur standard résiduelle assez élévée et une valeur de R2 faible n'atteignant que 0.32. On doit donc chercher un meilleur modèle, mais le fait de sélectionner nos variables et donc de réduire le nombre de variables s'est montré efficace.

2.4.2 Régularisation

Comme nous avons pu le faire précédemment, nous avons une fois de plus ajouter un à un des prédicteurs à notre modèle afin d'en calculer les erreurs. Cependant, cette fois-ci, nous avons ajouté un nouveau terme à notre modèle qui est dorénavant comme ci dessus :

$$\hat{\beta} = argmin_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + RegTerm \right\}$$

$$RegTerm = \lambda \sum_{j=1}^{p} (\alpha\beta_j^2 + (1 - \alpha|\beta_j|)$$

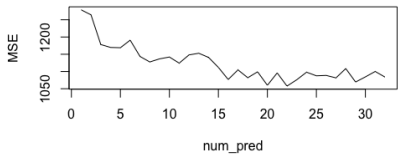
Le terme RegTerm correspond aux termes de l'elastic-net et nous allons donc cherche la meilleure régularisation possible pour nos modèles en testant pour des valeurs de la variable α allant de 0 (ce qui correspond alors à la régularisation de Ridge) jusqu'à 1 (régularisation Lasso). Nous pouvons alors observer une diminution de l'estimation en fonction du nombre de paramètres. Contrairement au modèle linéaire simple, nous avons donc une réduction de la variance des valeurs prises par nos estimateurs bêtas grâce à la régularisation, ainsi nos variables corrélées affectent moins notre modèle et il est donc intéressant d'avoir plus de 7 estimateurs.

```
for(i in (1:data.train.dim[2])){
  # ...
  # 2 - Preparation de la matrix pour les regularisations
  if( i == 1){
    data.regression <- as.matrix(data.train[,which(reg.order == i),])
  } else {
    data.regression <- cbind(data.regression, data.train[,which(reg.order == i)])
  }

  data.regression <- as.data.frame(data.regression)
  colnames(data.regression)[i] <- i
  data.regression.train <- model.matrix(label.train~.,data.regression)

  # 3 - From Ridge to Lasso
  for(l in 1:length(valueOfLambda)){
    cv.elastic.out <- cv.glmnet(data.regression.train, label.train, alpha = valueOfLambda[l])
    error.elasticNet[i,l]<-min(cv.elastic.out$cvm)
  }
}
```

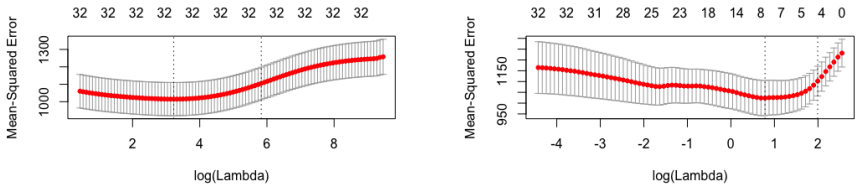
(a) Régularisation sous R



(b) MSE en fonction du nombre de prédicteurs

Figure 8: Observation de la régularisation

Nous avons ainsi à l'intérieur de cette partie véritablement testé l'ensemble des régularisations possibles vus en cours afin de pouvoir trouver la régularisation qui correspondait le mieux à nos données. En effet, il était tout d'abord intéressant de tester la régularisation car elle permet de réduire la variance de nos modèles tout en gardant tous les paramètres au lieu d'en sélectionner directement et donc de perdre de l'information. Par ailleurs, chacune des régularisations étaient intéressantes à observer. La régression sous Ridge permet de pénaliser les coefficients ayant des valeurs extrêmes alors que la régression sous Lasso nous permettait de pouvoir forcer certains paramètres à s'annuler donc ne pas influer sur notre modèle.



(a) Ridge Régularisation

(b) La méthode du lasso

Figure 9: Évolution des erreurs via différentes régularisation

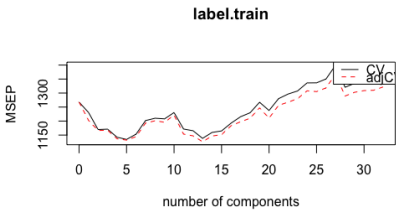
La méthode de l'elastic-net fait alors apparaitre une nouvelle variable α qui va nous permettre de pouvoir trouver le bon compromis entre ces deux différents types de régularisation. C'est pourquoi nous avons préféré le mettre en place, ainsi, aucune régularisation vu en cours n'a été mise de côté.

2.4.3 PCR (Principal Component Regression)

Pour finir, nous avons voulu tester un dernier modèle du fait des fortes corrélations de nos variables. En effet, si nous pouvions réduire la dimension de nos variables en agrégeant plusieurs variables sous une unique composante, alors nous aurions peut-être un meilleur modèle que celui obtenu via la régression linéaire régularisée. Pour ce faire, nous utilisons le package "pcr" sous R.

```
76 # Dernier modele : PCR (Principal Component regression)
77 model.pcr <- pcr(label.train~.,data=data.train, validation="CV")
78 validMSEP = MSEP(model.pcr)
```

(a) PCR sous R



(b) MSE en fonction du nombre de prédicteurs

Figure 10: Observation des résultats du PCR

Malheureusement, les résultats sont moins intéressants que ceux de la régression linéaire simple et régularisée. De plus, un des désavantages de cette méthode et que l'on ne sait plus réellement quels facteurs influent sur la guérison du patient et nous perdons donc en lisibilité alors que la prédiction n'en ait pas pour autant amélioré. Nous ne retiendrons donc pas ce modèle.

2.5 Observation du meilleur modèle

2.5.1 Choix du meilleur modèle

Afin de choisir notre meilleur modèle, nous allons simplement observer les différentes erreurs quadratiques obtenues par chacun de nos modèles de test. Nous retrouvons donc une erreur minimale obtenue grâce à la méthode de régression linéaire regularisée. Cependant, nous n'avons pris en variables que 13 prédicteurs et non les 33 présents au début.

Voici les différents meilleurs modèles que nous avons :

Modèle	nombre de paramètres	alpha	MSE
Reg. Linéaire	9		1044.533
Regularisation	13	0.5	1007.914
PCR	12		1047.533

Ainsi, nous pouvons voir qu'il aura été très intéressant de ne pas se limiter à une régression logistique régularisée avec tous nos paramètres et/ou qu'en fonction de Rigde ou de la méthode du Lasso car c'est la méthode de l'Elastic-Net avec un alpha de 0.5 et une sélection de 13 prédicteurs qui nous donne le meilleur modèle.

2.5.2 Résultats obtenus

Afin de pouvoir véritablement évalué notre modèle, nous régénérons alors un modèle avec les paramètres trouvés lors de nos test, soit avec les 13 paramètres sélectionnés ainsi qu'un alpha de 0.5. Nous trouvons alors notre lambda min de 2.32 grâce à cet ensemble d'apprentissage et calculons notre taux d'erreurs sur les données de test qui n'ont encore jamais été connu par notre modèle.

Voici tout d'abord les différents prédicteurs de notre modèle ainsi que les coefficients qui leur ont été attribués :

Prédicteurs	Coefficients
(intercept)	46.2967
perimeter_mean	-5.5079
lymph_node	-3.8742
smoothness_mean	0
symmetry_mean	2.0421
perimeter_se	0
fractal_dimension_worst	-4.3450
area_se	5.4418
texture_se	0
concave_point_worst	-1.6498
tumor_size	-4.1914
texture_mean	-0.4938545
area_worst	0
radius_mean	-0.3296

Nous obtenons alors une erreur quadratique moyenne de 814.237. Cette erreur est donc même inférieure à nos prédictions. Le fait d'avoir une erreur de test inférieur à une erreur d'apprentissage n'est pas rare, notamment lorsque l'on est sur une nombre de données peu important comme ce que nous avons ici dans cet exercice. Ainsi, il aurait été intéressant de pouvoir avoir un plus grand nombre de données afin de pouvoir relancer cette analyse et cette recherche du meilleur modèle.

2.5.3 Idée d'améliorations

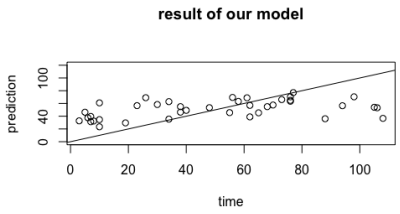


Figure 11: Prédictions en fonction des valeurs de tests

Au vue de la projection de nos estimations face aux véritables valeurs de la variable Time, il est facilement observable que nous avons définis un modèle linéaire ici alors qu'il ne l'était pas véritablement. En effet, beaucoup de problèmes ne sont malheureusement pas aussi simplement ré-solvable que via une simple régression linéaire. Ainsi, ici, on observe que l'on a une bonne estimation sur nos prédictions entre 30 et 80j (donc pour les valeurs moyennes) mais que nous n'avons pas du tout les bonnes estimations pour les valeurs extrêmes de "Time". Il aurait alors été intéressant de décomposer notre modèle en différents intervalles notamment via la méthode des splines afin d'avoir un modèle qui soit plus flexible et ainsi avoir un taux d'erreur plus faible. Cette méthode pourrait notamment nous donner de bons résultats car elle permettrait alors de réduire énormement la valeur de l'erreur quadratique.

3 Les données Phonème

3.1 Problématique et présentation des données

Les données Phoneme utilisées dans cette partie du TP sont des données sur des enregistrements vocaux d'individus fournis par la base de données TIMIT. Ces données sont utilisées pour de la reconnaissance vocale. Lors de ce TP nous allons donc entrainer notre modèle pour réussir à reconnaitre n'importe quel phoneme (élément sonore du langage) selon les prédicteurs enregistrés.

Nous disposons de 5 phonèmes différents : aa ao sh dcl iy. Nous disposons de données quantitatives et la variable à prédire est une valeur qualitative. Nous sommes donc bien dans un problème de classement et non de regression.

En s'informant sur les orgines de ces données nous avons pu apprendre que ces phonèmes sont énnoncés par 50 dicteurs différents venant d'origines différentes. Chaque dicteur a répété plusieurs fois une phrase bien définie, son enregistrement a été découpé en frames, et chaque frame a subi un log-periodogram qui comprend 256 spécifications. Nous disposons donc d'environ 5000 observations sur 256 variables différentes. Nous avons également un paramètre portant sur l'identité de l'individu qui a énnoncé le phoneme.

Lors de cette étude, l'objectif va etre de trouver la classe à laquelle appartient un phoneme en fonction de ses paramètres.

3.2 Analyse des données

Avant de nous lancer dans la recherche de notre meilleur modèle nous avons préféré analyser les données sur lesquelles nous allons travailler pour mieux les comprendre.

Tout d'abord nous pouvons regarder les proportions des classes présentes pour savoir si chaque phoneme est assez représenté dans notre jeu de données.

	aa	ao	dcl	iy	sh
Nombre d'observations	695	1022	757	1163	872
Probabilité a priori	15.41%	22.67%	16.78%	25.79%	19.34%

Selon ce tableau, nous pouvons voir que les 5 phonemes sont dans des proportions acceptables. Nous disposons donc d'assez d'observations de chaque phoneme pour travailler dessus et chercher un modèle efficace.

Nous pouvons ensuite réaliser une analyse en composante principale (ACP), méthode non supervisée qui nous donnera une visualisation des données dans une dimension 2. Cette ACP consiste à transformer des variables liées entre elles (dites " corrélées ") en nouvelles variables décorrélées les unes des autres. On va donc pouvoir visualiser les données dans un plan en éliminant le maximum de redondance et donc en affichant le plus d'information.

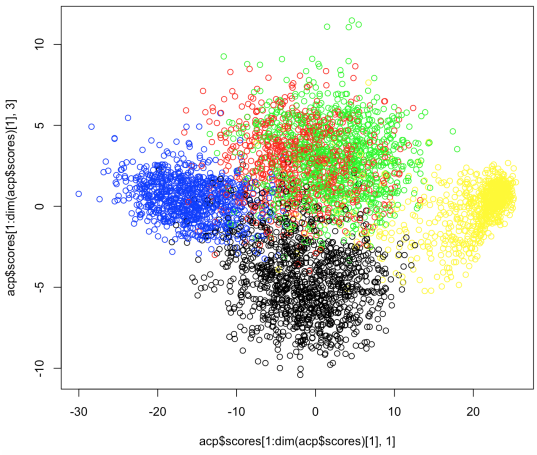


Figure 12: ACP - Données phoneme selon 2 composantes principales

Grâce à l'observation de ce graphique nous pouvons dèjà supposer que les 2 phonemes aa (rouge) et ao (vert) vont être difficiles à différencier. Il vont très certainement être a l'origine des erreurs lors du test de nos modèles.

3.3 Approche stratifiée

3.3.1 Séparation des données

De part la connaissance du jeu de données Phoneme sur lequel on travaille, nous avons décidé dans un premier temps de séparer les données de la maniere suivante :

- Un ensemble de test (1/3 des données)
- Un ensemble d'apprentissage (2/3 des données)

Cet ensemble d'apprentissage sera constitué de manière "stratifiée" c'est à dire qu'on a découpé les données et respectant certains critères:

- Aucun speaker ne doit apparaitre a la fois dans l'ensemble d'apprentissage et à la fois dans l'ensemble de test
- Les speakers venant de régions différentes et donc avec des accents différents, il faudra faire attention à ce que toutes les régions apparaissent dans nos ensemble de test et d'apprentissage, avec au moins un speaker de chaque sexe
- Tous les phonemes doivent etre présents dans les deux enesembles

Nous avons pris le soin de centrer réduire nos données pour s'affranchir des possibles écarts d'unité.

Dans la suite du TP nous construirons donc nos modèles sur nos ensembles d'apprentissage et nous évaluerons les performances en appliquant les modèles à l'ensemble de test

3.3.2 Selection de sous ensembles

Comme pour la première partie de ce TP nous avons pu remarquer que certaines de nos variables sont corrélées. De plus le nombre de variables est élevé ainsi il serait intéressant de mettre en place une méthode pour tester une selection de sous ensembles de variables. C'est la méthode du "forward stepwise selection", elle consiste a classer les variables de la plus significative à la moins significative. Nous pourrons tester les performances de chaque modèle en prenant itérativement un nombre de variables croissant. Ainsi nous allons choisir pour chaque modèle testé un ensemble de variables qui va être gardé.

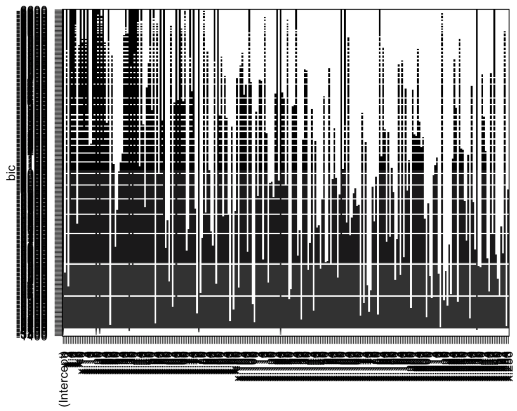


Figure 13: Graphe regsubset obtenu

De la même maniere que dans la premiere partie du TP avec les données Breast Cancer nous allons prédire la classe de nos phonemes avec un nombre de prédicteurs allant de 1 à 256.

3.3.3 Construction des modèles

Lors de notre étude nous avons utilisé 6 modèles différents : - L'analyse discriminante quadratique (ADQ)
- L'analyse discriminante linéaire (ADL)
- La méthode des K plus proches voisins (KNN)
- La régression logistique (GLMNET)
- Les arbres de décision (TREE)
- Le classifieur bayésien naif (BAYES)

Nous allons donc exposer les résultats obtenus avec chaque modèle.

Analyse discriminante linéaire

Pour l'analyse discriminante linéaire nous pouvons voir qu'à partir d'environ 50 variables nous avons des résultats qui se stabilisent. L'erreur minimale est observée à 132 variables, on remarque cependant qu'à partir de 50 variables nous obtenons une erreur stable d'environ 7.9%.

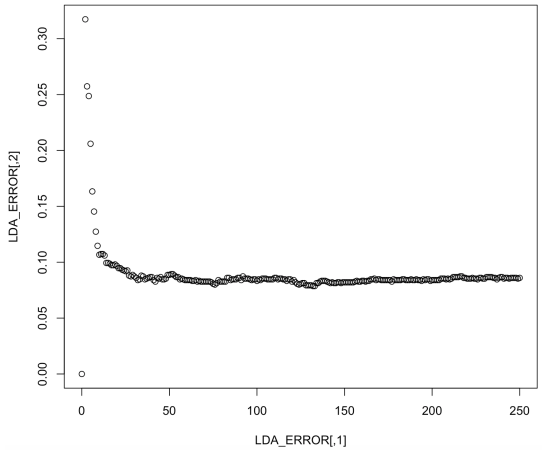


Figure 14: Variation du taux d'erreur en fonction du nombre de variables retenues

L'erreur minimale observée à 132 variables est de 7.87%. Voici la matrice de confusion associée :

Table de confusion	aa	ao	dcl	iy	sh
aa	171	61	0	0	0
ao	48	290	0	2	0
dcl	0	0	245	6	0
iy	0	0	1	386	0
sh	0	0	0	0	290

parler de la subset selection et montrer que pas forcement utile ici

Analyse discriminante quadratique

Pour l'analyse discriminante quadratique nous pouvons voir que les résultats sont meilleurs pour un nombre limité de variables. Nous obtenons donc un nombre de variables optimal à 37. L'analyse discriminante quadratique subit le "curse of dimentionality", plus on considere de dimensions, plus le modele subit

le sur apprentissage ce qui se traduit par une erreur élevée sur l'ensemble de test.

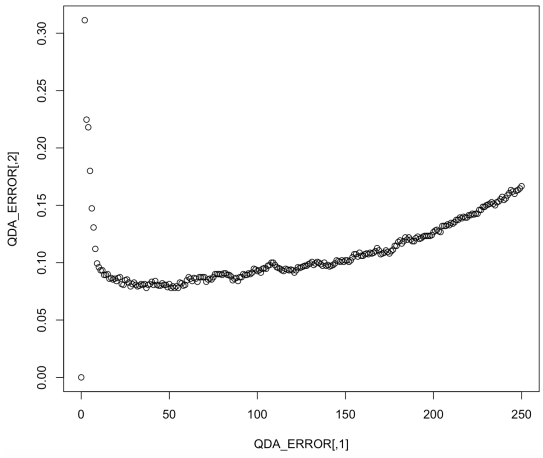


Figure 15: Variation du taux d'erreur en fonction du nombre de variables retenues

L'erreur minimale observée à 37 variables est de 7.8%. voici la matrice de confusion associée :

Table de confusion	aa	ao	dcl	iy	sh
aa	167	65	0	0	0
ao	45	294	0	1	0
dcl	0	0	246	5	0
iy	0	0	1	386	0
sh	0	0	0	0	290

Nous pouvons voir ici que la subset selection est intéressante, si nous choisissons l'analyse discriminnante quadratique comme modèle final il sera important de l'appliquer.

K plus proches voisins

Nous avons testé l'algorithme des K plus proches voisins. Cet algorithme permet d'assigner la classe la plus en présence dans un rayon de K données. Nous avons fait tourner cet algorithme sur différentes valeurs de K, et nous remarquons qu'il devient tres intéressant pour un K=8 lorsqu'on garde 48 variables.

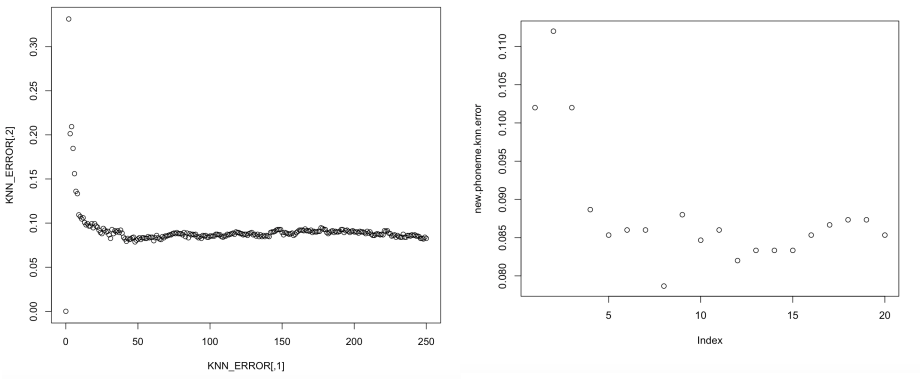


Figure 16: Erreur de KNN en fonction du nombre de variables retenues

L'erreur minimale observée avec 48 variables et un k optimal égal à 8, est de 7.87%.

Régression logistique

Nous avons mis en place une regression logistique multinomiale grâce à la fonction GLMNET, la grande différence avec la regression logistique utilsée dans d'autres exercices, est le nombre de classes à prédire. Nous devons ici prédire le pourcentage de chance d'appartenir à chacune des 5 classes présentes.

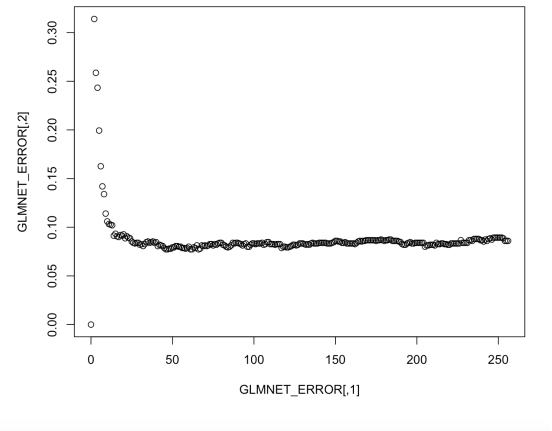


Figure 17: Variation du taux d'erreur en fonction du nombre de variables retenues et des k plus proches voisins

Nous pouvons voir ici que le nombre de variables optimale est 46 et que

le taux d'erreur sur l'ensemble de test est 7.73. Voici la matrice de confusion associée :

Table de confusion	aa	ao	dcl	iy	sh
aa	168	64	0	0	0
ao	47	293	0	0	0
dcl	0	0	248	3	0
iy	0	0	1	386	0
sh	0	0	0	1	289

Arbre de décision

Pour l'arbre de décision nous pouvons voir que le nombre de variables optimal retenu est 60 avec un taux d'erreur de 12.53%. Nous remarquons que son taux d'erreur evolue peu à partir d'environ 15 variables considérées.

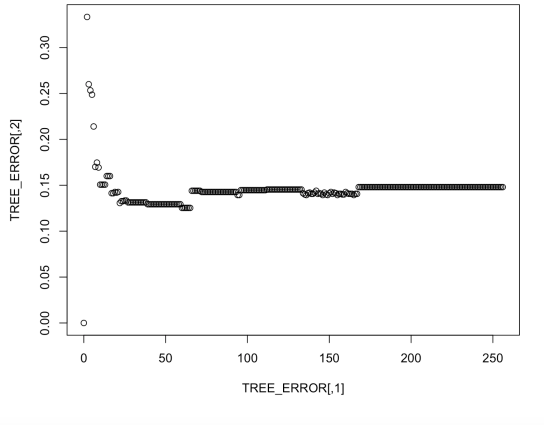


Figure 18: Variation du taux d'erreur en fonction du nombre de variables retenues

Classifieur bayésien naif

Pour le classifieur bayésien naif nous pouvons voir que le nombre de variables optimal retenues est 57 avec un taux d'erreur de 10.27%. Nous remarquons que son taux d'erreur augmente légèrement de maniere linéaire à partir d'environ 50 variables considérées.

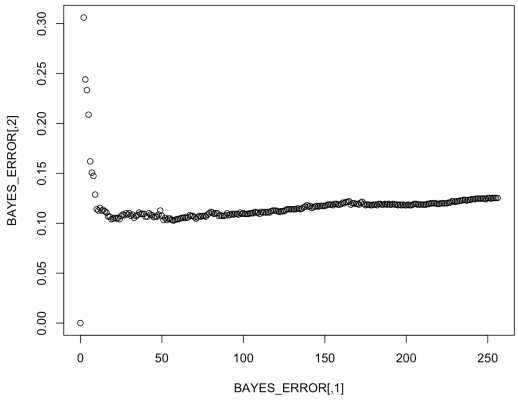


Figure 19: Variation du taux d'erreur en fonction du nombre de variables retenues

Pour conclure nous pouvons voir que la sélection d'un sous ensemble de variables n'est pas toujours intéressante en matière de taux d'erreur, elle est cependant intéressante en terme de temps de calcul (en effet moins un jeu de données a de parametres, plus rapides seront les calculs de modèles). De plus, on peut voir en observant les matrices de confusion que la plupart des erreurs de classement sont liées à la proximité des classes aa et ao.

Nous pouvons rassembler tous nos résultats dans un tableau pour avoir un premier aperçu de la performance des modèles :

	ADL	ADQ	KNN	GLMNET	TREE	BAYES
Erreur	7.87%	7.8%	7.87%	7.73%	12.53%	10.27%
Nombre de variables gardées	132	37	48	46	60	57
Erreur si toutes variables gardées	8.6%	18.93%	9.27%	11.13%	14.8	12.53%

Nous avons maintenant voulu utiliser l'Analyse Factoriel Discriminante (FDA). C'est une méthode factorielle de réduction de dimensions pour l'exploration statistique de variables quantitatives et d'une variables qualitative. Cette méthode est utilisée également de maniere non sueprvisée pour obtenir une représentation graphique des données en 2 dimensions et repérer les clusters. On peut repérer une deuxieme similarité entre deux phonemes dcl et iy, qui pourrait être une source d'erreur dans nos modèles.

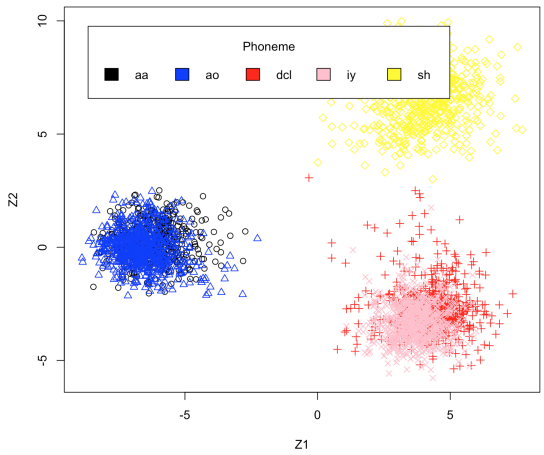


Figure 20: Analyse factorielle discriminante

Nous allons ici utiliser une analyse factorielle disciminante linéaire (comme les résultats précédents ont été meilleurs avec les modèles linéaires et non quadratiques). Nous avons de la même maniere que précédement effectué la construction des 6 modèles avec notre nouveau jeu de données, ici rapporté à 4 dimensions. Voici les taux d'erreur trouvés :

	ADL	ADQ	KNN	GLMNET	TREE	BAYES
Taux d'erreur	5.47%	5.67%	5%	5.27%	5.2%	5.6%

On remarque rapidement ici que nos résultats se sont netemment améliorés. Nous avons réussi à descendre notre erreur de test à 5% en utilisant la méthode des K plus proches voisins avec un k optimal à 8. Ce résultat n'est pas surprenant car la méthode des K plus proches voisins est plus performante dans les petites dimensions.

3.4 Approche validation croisée

3.4.1 Séparation des données

Dans un deuxième temps nous avons voulu efectuer un deuxieme type de séparation de données : la validation croisée. L'approche stratifiée présente des limites sur certains points. En effet les observations peuvent influencer beaucoup la construction des modèles selon s'ils se trouvent dans l'ensemble d'apprentissage ou de test. On a donc décidé de lisser cette erreur en effectuant une validation croisée.

Pour cela j'ai créé deux ensembles , un ensemble d'apprentissage contenant 2/3 des données, et un ensemble de validation contenant 1/3 des données. Je vais laisser cet ensemble de validation de coté jusqu'a ma sélection finale de

modèle. Dans mon ensemble d'apprentissage, Nous choisissons arbitrairement de découper en 5 sections de 600 données chacunes. Sur ces 10 sections nous choisissons tour à tour 1 section qui representera notre ensemble de test. Nous allons donc effectuer la construction de notre modèle 5 fois et ensuite choisir celui qui aura les meilleures performances.

3.4.2 Construction du modèle

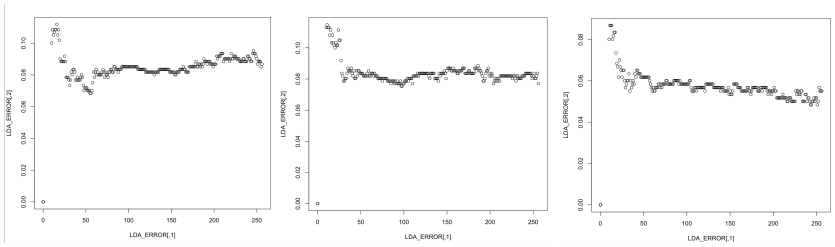
De la même maniere que pour la premiere séparation de données nous testons 6 modèles :
- L'analyse discriminante quadratique (ADQ)
- L'analyse discriminante linéaire (ADL)
- La méthode des K plus proches voisins (KNN)
- La régression logistique (GLMNET)
- Les arbres de décision (TREE)
- Le classifieur bayésien naif (BAYES)

Pour ces 6 modèles nous allons effectuer la même démarche que la section précédente. A savoir nous allons commencer par une selection d'un sous ensemble optimal et estimer le taux d'erreur

Analyse Discriminante Linéaire

Nous obtenons un taux d'erreur moyen de 6,61%. Comme on s'en doutait car vu précédement, il n'y a pas de sous ensembles de variables idéals. Nous pouvons le voir dans les 5 graphes qui ont été construit pour chaque K fold de notre ensemble d'apprentissage.

itération :	1	2	3	4	5	moyenne
Taux d'erreur	6,84%	7,54%	4,84%	6,30%	7,52%	6.61%



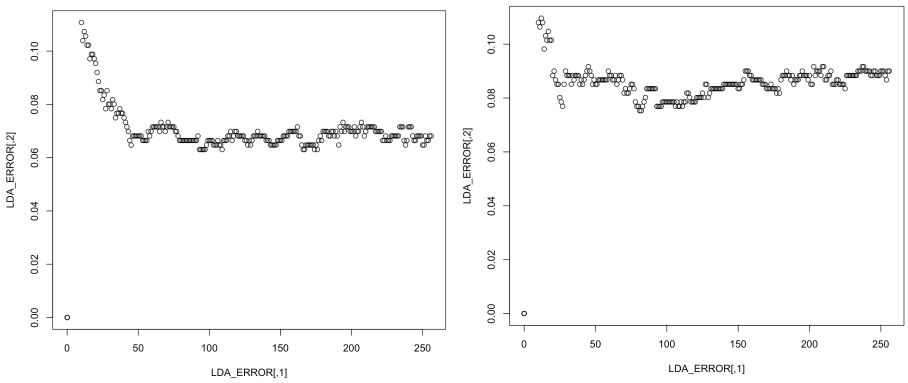
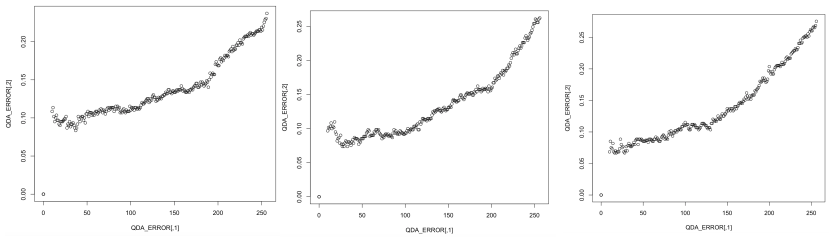


Figure 22: Variation du taux d'erreur en fonction du nombre de variables retenues pour K folds

Analyse discriminante quadratique

Pour l'analyse discriminante quadratique on peut voir se dessiner un nombre de variable optimal autour de 25. Nous obtenons grâce à ceci une erreur moyenne de 7,75%.

itération :	1	2	3	4	5	moyenne
Taux d'erreur	8,4%	7,38%	6.68%	8.01%	8.35%	7.75%



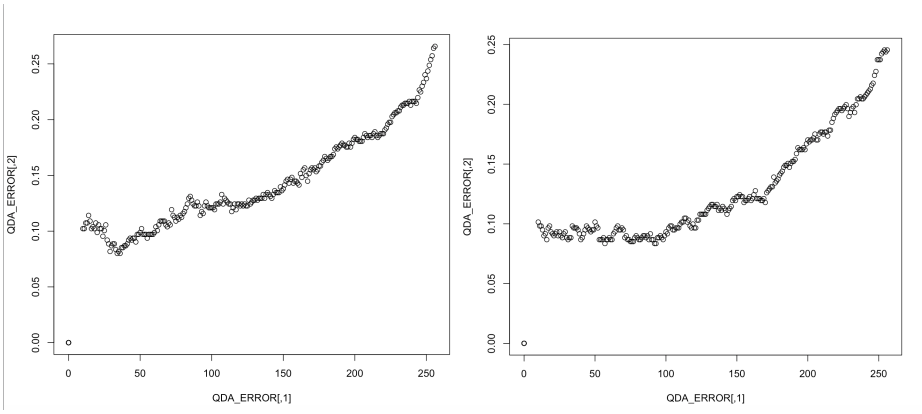


Figure 24: Variation du taux d'erreur en fonction du nombre de variables retenues pour K folds

K plus proches voisins

Lorsqu'on applique la méthode des K plus proches voisins, nous obtenons toujours un K optimal égal à 8. L'erreur moyenne de ce modèle est de 7,55%.

itération :	1	2	3	4	5	moyenne
Taux d'erreur	8.51%	7,38%	6.01%	8.01%	7,86%	7.55%

Régression logistique

Pour la régression logistique nous n'avons pas effectué de selection de sous ensemble. En effet le temps de traitement était tres long et comme nous avons pu le voir dans la section précédente la selection d'un sous ensemble de variables n'augmentait pas les performances. Nous obtenons un taux d'erreur moyen de 10,57%

Arbre de décision

Nous avons des performances moins intéressantes sur l'arbre de décision avec une erreur moyenne de 13,21%

itération :	1	2	3	4	5	moyenne
Taux d'erreur	14.02%	14.92%	11.02%	13.63%	12.44%	13.21%

Classifieur Bayésien Naif

La performance moyenne du Classifieur Bayésien Naif est de 8,75%.

itération :	1	2	3	4	5	moyenne
Taux d'erreur	10,02%	9,34%	6,84%	9,03%	8,51%	8,75%

Nous avons dans cette partie effectué tous les algorithmes de classement K fois pour obtenir une erreur de classement moyenne.

Lorsque nous avons voulu efectuer une analyse factorielle discriminante nous avons obtenu de très mauvais résultats (autour de 30% d'erreur) comparés à ceux obtenus avec l'approche de séparation de données différente. Nous pouvons supposer que le choix de l'ensemble d'apprentissage influence trop sur la construction du modèle. En étudiant les matrices de confusions nous avons remarqué que les erreurs de classement étaient principalement entre les aa et ao. Il se peut que certains phonèmes soient trop peu représentés.

3.4.3 Choix du meilleur modèle et résultats

Nous pouvons donc rassembler les erreurs moyennes de chaque modele dans un tableau :

	ADL	ADQ	KNN	GLMNET	TREE	BAYES
Taux d'erreur	6,61%	7,75%	7,55%	10,57%	13,21%	8,75%

Nous allons choisir logiquement l'Analyse Discriminante Linéaire car c'est elle qui donne le taux d'erreur le plus bas parmis nos modèles.

Une fois le modèle choisi il faut maintenant l'appliquer à notre ensemble de validation ce qui va nous permettre de connaitre notre vrai taux d'erreur. Pour ce la on reconstruit notre ensemble d'apprentissage en entier. Ensuite on applique le modèle de l'analyse discriminante liénaire avec comme ensmeble de test notre ensemble de validation. Nous obtenons une erreur de 7,85%.

Table de confusion	aa	ao	dcl	iy	sh
aa	177	56	0	0	0
ao	54	284	0	0	0
dcl	0	0	266	8	0
iy	0	0	0	369	0
sh	0	0	0	0	289

En étudiant notre matrice de confusion nous pouvons sans surprise voir que nos erreurs sont liées au erreurs de classement entre le données aa et ao.

3.5 Conclusion et interpretation

Lors de ce probleme de classification nous avons mis en oeuvre 6 techniques d'apprentissage supervisé. En utilisant deux méthodes de séparation de données différentes nous avons trouvé 2 méthodes optimales différentes. Pour l'approche stratifiée la méthode des K plus proches voisins avec un K optimal à 8 et en ayant au préalable réalisé une analyse factorielle discriminante. Pour l'approche en validation croisée cette fois ci c'etait l'analyse discriminante linéaire la plus efficace.

La différence de résultat est intéressante et nous amène à plusieurs reflexions. Premièrement on peut voir que la sélection minutieuse de l'ensemble sur

lequel on va apprendre nos données est importante car peut faire baisser de presque 2% notre erreur de test. Dans le cas d'une classification, il est donc important de comprendre les données sur lesquelles on travaille avant de lancer des algorithmes. Deuxiemement lorsqu'on regarde les résultats de la validation croisée, on peut remarquer que le KNN est le deuxieme modèle plus performant. Lorsqu'on regarde à l'inverse les résultats de l'approche stratifiée on remarque l'analyse discriminante linéaire est aussi une méthode avec une bonne performance. Nous pouvons donc dire que les KNN et l'ADL ont globalement une bonne performance sur notre jeu de données dans ce probleme de classification. Dernierement, comme on en a fait la remarque au début de cette partie, la plupart de nos erreurs de classification ont été la cause la ressemblance entre les phonèmes aa et ao. Nous pourrions tomber sur des performances accrues si on réussissait à trouver un modèle qui permettrait de traiter plus efficacement ces données.