UNIVERSITÉ DE LORRAINE

MÉMOIRE DE STATISTIQUES EN GRANDE DIMENSION

Master 2 Ingénierie Mathématique pour la Science des Données

2022-2023

---

gene expression cancer RNA-Seq Data Set

Réalisé par :
Marcel MOUDILA

Enseignante :
Anne Gégout-Petit

12 décembre 2022

Table des matières

1 Présentation du jeu de données **2**

2 Prétraitement des données **3**
2.1 gestion des variables qui ne varient pas . . . 3
2.2 test d'Anova . . . 3
2.3 test multiple . . . 3
2.4 recherche des corrélations dans les variables prédictives . . . 4
2.5 analyse des composantes principales parcimonieuses . . . 4
2.6 données d'apprentissage et données de test . . . 4

3 Entrainement des modèles **6**
3.1 choix des modèles . . . 6
3.2 stratégie de comparaison des modèles . . . 6
3.3 meilleur modèle . . . 6
3.4 autres alternatives . . . 6

4 Codes R **9**
4.1 importation des données . . . 9
4.2 exemple d'entrainement et validation (package caret) . . . 9

Bibliographie **10**

Chapitre 1

Présentation du jeu de données

Issu du site archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq, le dataset étudié présente différents types de tumeurs en fonction de leurs caractéristiques génétiques.
Il a pour dimension (801,20532). Le but du projet est de prédire les labels de la variable cible.
La variable cible est qualitative, il s'agit donc de réaliser une classification.

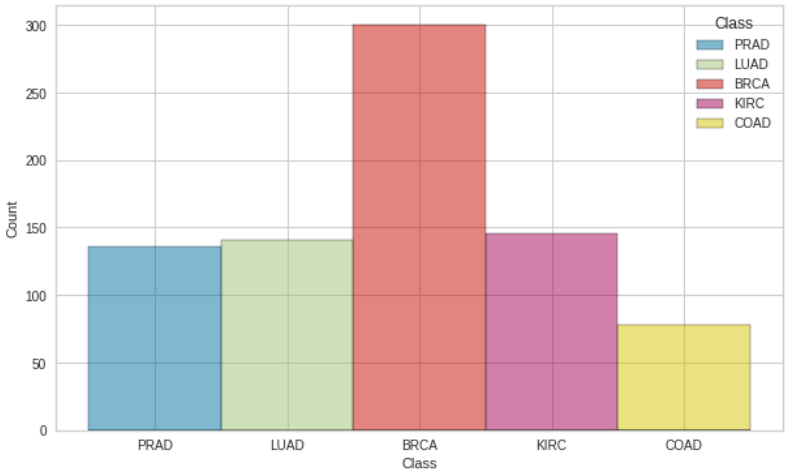

FIGURE 1.1 – effectif des labels

Les variables prédictives sont toutes quantitatives, l'étude descriptive de ces variables a mis en évidence que 267 entre elles ne varient pas. La première étape du prétraitement (chapitre suivant) a consisté à supprimer ces 267 variables. Ces variables ont pu être dénombrés grâce au code ci-dessous :

```
p = dim(df)[2]
ind = c()
for (j in 2:p){
  if (min(df[,j])== max(df[,j])){
    ind = c(ind,j)
  }
}
length(ind)
```

C'est très curieux qu'il ait jusqu'à 267 variables qui ne varient pas. L'explication, possible, trouvée est que le dataset est une sélection aléatoire des instances de tumeurs chez les patients, et donc cette sélection (notre dataset) n'a tout simplement pas de tumeurs qui aurait au moins une des 267 caractéristiques génétiques.

Chapitre 2

Prétraitement des données

Le prétraitement des données est une étape très importante en science des données, et plus de la moitié du temps d'un data scientist est consacré à cette étape.

2.1 gestion des variables qui ne varient pas

Une variable qui ne prend qu'une seule valeur pour chaque observation a peu d'intérêt. Dans notre dataset, 267 variables ne prennent que la valeur 0 pour chaque instance. Ces variables ont été écartés. Le tableau des statistiques descriptives de ces variables est donné ci-dessous (Python)

Out[11]:

	gene_5	gene_23	gene_4370	gene_4808	gene_4809	gene_4814	gene_48
count	801.0	801.0	801.0	801.0	801.0	801.0	801
mean	0.0	0.0	0.0	0.0	0.0	0.0	0
std	0.0	0.0	0.0	0.0	0.0	0.0	0
min	0.0	0.0	0.0	0.0	0.0	0.0	0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0
75%	0.0	0.0	0.0	0.0	0.0	0.0	0
max	0.0	0.0	0.0	0.0	0.0	0.0	0

8 rows × 267 columns

FIGURE 2.1 – statistiques descriptives

2.2 test d'Anova

Ce test étudie la dépendance entre une variable qualitative et une variable quantitative. L'hypothèse nulle d'indépendance est rejetée si la p-value est inférieure à 10 % (seul fixé). Dans notre cas, ce test a été mis en place pour étudier le lien entre chaque variable prédictive (quantitative) et la variable cible (qualitative).

2.3 test multiple

La méthode de Bonferroni a été utilisée, car, le but recherché est d'augmenter considérablement le nombre de test d'Anova non significatif. La p-value ajustée, calculée à partir des p-value trouvés pour le test d'Anova, a permis d'en dénombrer que 1886 tests non significatifs.
Les autres méthodes ont été aussi testées, mais donnait moins de tests non significatifs. La méthode de Bonferroni a donc été maintenue.

```r
# created pvalues's vector
vecteur_pv = c()
p = dim(df2)[2]
for (i in 2:p){
  pv = anova(lm(df2[,i] ~ df2$Class))$`Pr(>F)`[1]
  vecteur_pv = c(vecteur_pv, pv)
}
```

```r
{r echo=TRUE}
length(vecteur_pv)
```

```
[1] 20264
```

```r
{r}
summary(vecteur_pv)
```

```
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.00000 0.00000 0.01209 0.00000 0.99551
```

```r
{r}
# calculated the adjust pvalues
adj.pv = p.adjust(vecteur_pv,"bonferroni")
indCol_adj = which(adj.pv >=0.1)
length(indCol_adj)    # number of not significant test by Bonferroni
```

```
[1] 1886
```

FIGURE 2.2 – Mise en œuvre du test multiple de Bonferroni

2.4 recherche des corrélations dans les variables prédictives

Passant notre dataset de 20264 variables prédictives à 18378 variables prédictives grâce aux tests multiples, 3052 autres variables prédictives ont été écartés des données quand la somme des coefficients de corrélation de la variable avec les autres variables dépassait 0.75

2.5 analyse des composantes principales parcimonieuses

Avec 15326 variables prédictives désormais, il était très lent avec mon équipement informatique (intel core i5) de faire tourner des modèles d'apprentissage automatique. D'ailleurs, même pour exécuter l'analyse des composantes principales parcimonieuses en Python, j'ai été confronté par la lenteur de mon équipement, mais j'ai réussi à le faire sous R. Je me suis ainsi retrouvé avec 30 variables prédictives. Les hyperparamètres renseignés sont : K=3, type="predictor",sparse ="varnum",para =c(10,10,10)
La première colonne (fig 2.3) indique les numéros de colonnes des variables dont le coefficient est non nul dans les résultats de Sparse PCA. C'est ainsi, je récupère les 30 variables prédictives, et j'ai pu entrainer mes modèles d'apprentissage supervisé.

2.6 données d'apprentissage et données de test

600 instances dans les données d'apprentissage et 201 instances dans les données de test. Les dimensions sont donc (600,30), (600,1), (201,30), et (201,1) respectivement pour X_train, y_train, X_test, et y_test.

X	PC1	PC2	PC3
106	-0.422210531	0.0000000000	0.00000000
346	0.281202734	0.0000000000	0.00000000
430	-0.116618966	0.0000000000	0.00000000
2154	0.218526868	0.0000000000	0.00000000
3482	0.002822877	0.0000000000	0.00000000
5736	-0.429687502	0.0000000000	0.00000000
9402	-0.238397203	0.0000000000	0.00000000
10130	0.508410842	0.0000000000	0.00000000
13274	0.387022209	0.0000000000	0.00000000
14461	0.177676659	0.0000000000	0.00000000
4418	0.000000000	-0.2172867083	0.00000000
5654	0.000000000	-0.3890156810	0.00000000
7191	0.000000000	-0.3368061694	0.00000000
7807	0.000000000	0.3563807055	0.00000000
8671	0.000000000	-0.2215505941	0.00000000
9729	0.000000000	0.4543601210	0.00000000
9753	0.000000000	0.0016906349	0.00000000
9845	0.000000000	-0.0505433738	0.00000000
13037	0.000000000	0.0009332494	0.00000000
13645	0.000000000	-0.5503830813	0.00000000
1232	0.000000000	0.0000000000	-0.19001777
2960	0.000000000	0.0000000000	-0.15431428

1 to 23 of 30 entries, 4 total columns

FIGURE 2.3 – 10 coefficients dans PC1 sont non nuls
10 coefficients dans PC2 sont non nuls
10 coefficients dans PC3 sont nuls

Chapitre 3

Entrainement des modèles

3.1 choix des modèles

Trois modèles mis en comparaison : la régression logistique (multinomiale), les forêts aléatoires de classification, et l'analyse discriminante linéaire.

3.2 stratégie de comparaison des modèles

Les modèles ont été entrainés avec le même X_train, et le même y_train, et avec la même stratégie à savoir la validation croisée à 10 folds. La métrique d'évaluation est l'accuracy. Pour chaque modèle, une prédiction a été calculée à partir de X_test. Puis, l'accuracy sur les données de test a été calculé à partir de y_test et de la prédiction. Le package Caret a été utilisé pour cette tâche. Les résultats des modèles sont dans les figures ci-dessous :
Sur les données de test, les résultats des trois modèles sont dans le tableau ci-dessous :

	accuracy
régression logistique	0.97
forêts aléatoires	0.98
analyse discriminante linéaire	0.98

Table 3.1 – Accuracy sur les données de test

3.3 meilleur modèle

Le meilleur modèle est Random Forest (forêts aléatoires).

3.4 autres alternatives

Le modèle SPLS-DA du package mixOmics a été le plus rapide de par son éxécution à partir des données d'entrainement avec 15326 prédicteurs (trouvé tout juste après les tests multiples et les corrélations)
je donne ci-dessous une projection des données dans le premier plan factoriel, laissant entrevoir des clusters bien nets.

```r
regLogistique
```

```
Penalized Multinomial Regression

600 samples
 20 predictor
  5 classes: 'BRCA', 'COAD', 'KIRC', 'LUAD', 'PRAD'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 540, 539, 540, 539, 539, 540, ...
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.9248900  0.9012643
  1e-04  0.9563709  0.9425078
  1e-01  0.9749888  0.9670313

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 0.1.
```

FIGURE 3.1 – cross validation pour la régression logistique

```r
randomForest
```

```
Random Forest

600 samples
 30 predictor
  5 classes: 'BRCA', 'COAD', 'KIRC', 'LUAD', 'PRAD'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 541, 541, 541, 539, 541, 538, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9949718  0.9933873
  16    0.9867168  0.9824656
  30    0.9701029  0.9605840

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

FIGURE 3.2 – cross validation pour les forêts aléatoires

```r
{r}
ADL
```

```
Linear Discriminant Analysis

600 samples
 30 predictor
  5 classes: 'BRCA', 'COAD', 'KIRC', 'LUAD', 'PRAD'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 540, 541, 540, 540, 537, 540, ...
Resampling results across tuning parameters:

  dimen  Accuracy   Kappa
  1      0.7466088  0.6592939
  2      0.9549610  0.9404917
  3      0.9865819  0.9823708
  4      0.9949718  0.9934124

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was dimen = 4.
```

FIGURE 3.3 – cross validation pour l'analyse discriminante linéaire

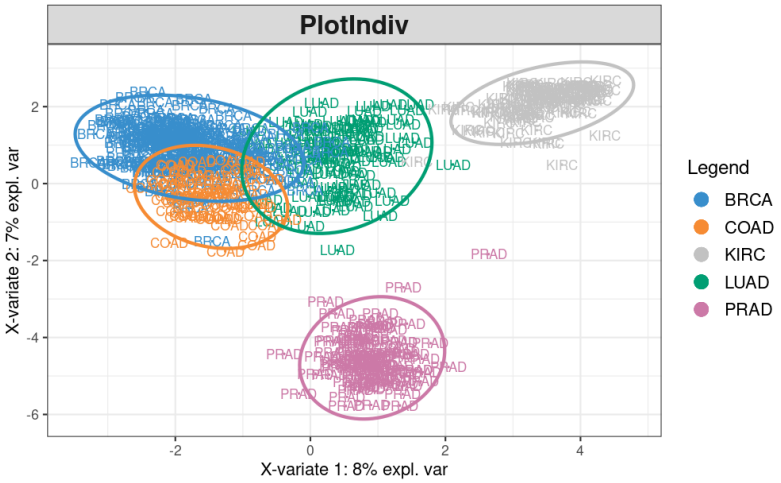

FIGURE 3.4 – Projection des données dans un espace 2D

Chapitre 4

Codes R

4.1 importation des données

```r
library(data.table)
data = fread("/home/moudilamarcel/Bureau/data.csv")
labels = fread("/home/moudilamarcel/Bureau/labels.csv", header = TRUE)
data = as.data.frame(data[,-1])
labels = as.data.frame(labels[,-1])

# quelques manipulations
names(labels) = "Class"
labels$Class = as.factor(labels$Class)
df = cbind.data.frame(labels,data)
```

4.2 exemple d'entrainement et validation (package caret)

```r
# cross-validation avec 10 folds
fitControl <- trainControl(method = "cv", number = 10)

# régression logistique multinomiale

regLogistique <- train(X_trainFinal, y_trainFinal,
                  method = "multinom",
                  trControl = fitControl,
                  metric = "Accuracy",
                  trace = FALSE)

# prediction
pred_regLogistique <- predict(regLogistique, X_test)

# score on testing dataset
postResample(pred = pred_regLogistique, obs = y_test)

```

Bibliographie

[Lan19] Brett Lantz. *Machine learning with R : expert techniques for predictive modeling*. Packt publishing ltd, 2019.