

UE ANALYSE FOUILLE DE DONNÉES

M2 MIASHS SSD 2019 - DATA CHALLENGE

Jean-Baptiste FICHES
Hugo MARTHINET
Komi AGBLODOE

Pierre MAHE

Table des matières

1	Introduction	2
1.1	Statistiques descriptives	2
2	Traitement des données	2
2.1	Harmonisation des notations	2
3	Analyse des données	2
3.1	Individus outliers	3
3.2	Inégalité de représentation des classes	3
3.3	ACP	3
4	Sélection de variables	4
4.1	Variables les plus importantes	4
4.2	Lasso	5
5	Création du modèle	5
5.1	Comparaison des modèles	5
5.2	Prédictions	6
5.3	Conclusion	6

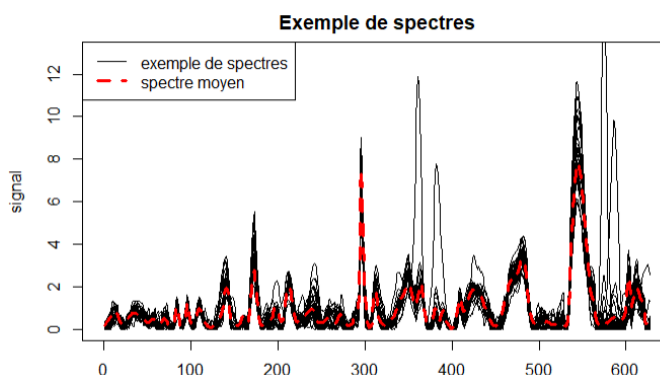
1 Introduction

Ce *Data Challenge* a pour but d'analyser des spectres afin de prédire l'espèce des bactéries. Nous avons à disposition une matrice d'entraînement de dimension 3114 lignes par 627 colonnes. Les 3114 lignes sont les 9 spectres \times 346 souches bactériennes. Nous avons aussi une matrice dont nous devrons prédire les classes comportant 1197 lignes (133 souches \times 9 spectres).

Le but de ce *data challenge* est donc de prédire l'espèce des 133 souches. Il faudra réaliser quelques statistiques descriptives, pré-traiter les spectres, gérer les réplicats au sein d'une même souche, sélectionner des variables et construire un modèle de classification.

1.1 Statistiques descriptives

Nous sommes en présence de variables qui ne varient pas beaucoup. Il existe cependant trois souches qui ont des variances élevées. Regarder uniquement la moyenne par variable ou la variance n'est pas très intéressant. On va plutôt représenter graphiquement quelques spectres avec un spectre moyen du jeu de données.



Sur ce graphique, on remarque plusieurs pics différents entre variables aux alentours des coordonnées 150, 180, 370, 480, 550 et 600. Il faudra faire attention que ces variables soient sélectionnées dans le modèle choisi car elles doivent avoir une influence sur la classification.

2 Traitement des données

2.1 Harmonisation des notations

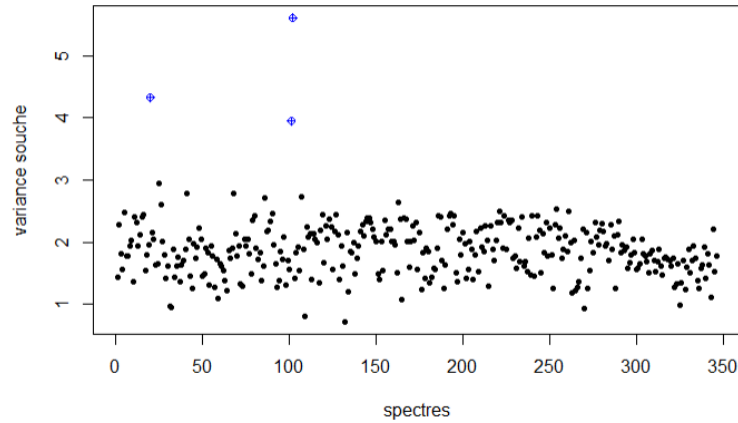
Afin de pouvoir mener à bien ce projet, nous avons dû modifier certaines données, comme celles du fichier *meta_train.txt* il contenait les facteurs des types de bactéries. Nous avons passé par exemple de *sp_12* à *12*, ou encore *sp_40* à *40*.

3 Analyse des données

Avant de commencer à programmer la construction d'un modèle, il nous a fallu réfléchir à plusieurs choses. Nous avons commencé par identifier les *outliers*, puis contrôlé la répartition des classes et avons choisi de réaliser une *Analyse en Composante Principale* - ACP.

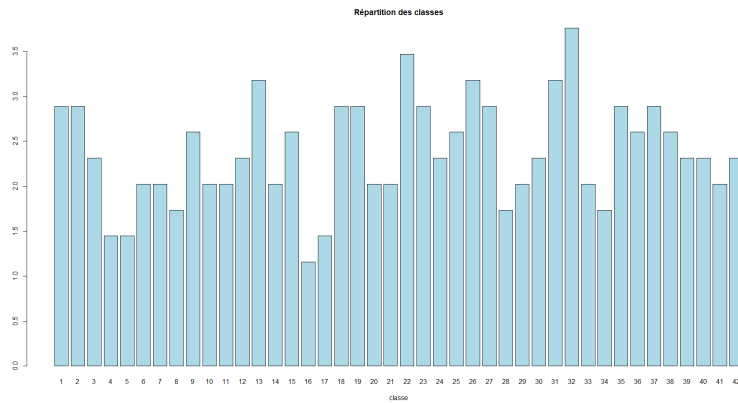
3.1 Individus outliers

Pour récupérer les individus que nous considérons comme *outliers*, nous commençons par calculer la variance de chaque (3114) individus. On fait ensuite la moyenne de ces variances par souche (on rassemble donc les individus neuf par neuf). Nous nous retrouvons donc avec un vecteur contenant la variance par souche. Si une variance est élevée par rapport aux autres, cela veut surment dire qu'il est un *outlier*. Dans notre cas, nous récupérons les souches 20, 101 et 102(c'est à dire 3×9 individus). On peut maintenant représenter ces 27 spectres en comparaison avec la moyenne des spectres.



On remarque bien ici qu'elles sont les valeurs abberantes.

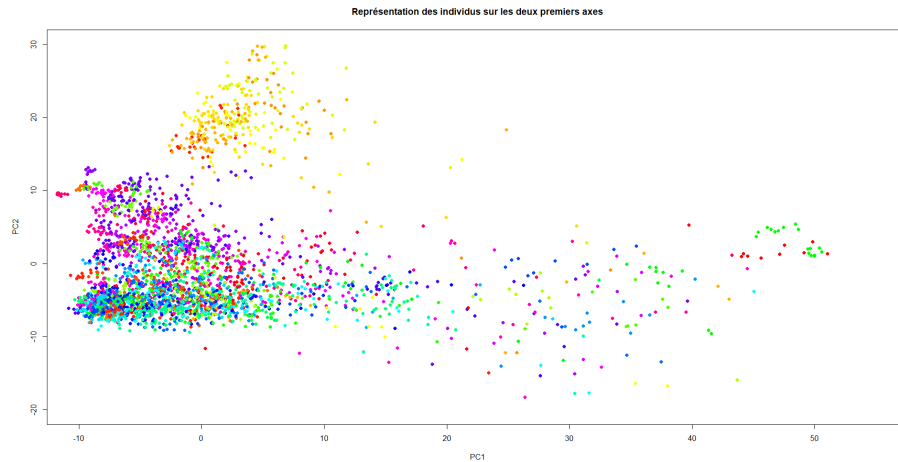
3.2 Inégalité de représentation des classes



Nous avons aussi remarqué que les classes n'étaient pas totalement bien équilibrées (représentation minimum : 1.12% ; représentation maximum : 3.76%). Pour palier à ce problème, nous avons précisé dans le modèle que toutes les classes étaient équiprobables. Nous avons alors utilisé le paramètre *classwt* en lui affectant une valeur égale pour chaque classes possible (nous avons 42 classes possibles donc on lui affecte la valeur $\frac{1}{42}$).

3.3 ACP

Notre première idée était de récupérer les variables les plus importantes à partir d'une ACP. Nous n'avons cependant pas continué dans cette voie une fois que nous avons trouvé une meilleure méthode que nous présenterons plus tard. Cependant, cela nous a quand même permis de remarquer quelques rapprochements. Le pourcentage de variance expliqué par chaque axe n'est pas très élevé. En effet, si l'on fait la somme des 5 premiers axes, on obtient moins de 42% des informations représentées. Ci-dessous, le graphique des individus sur les deux premières composantes principales.



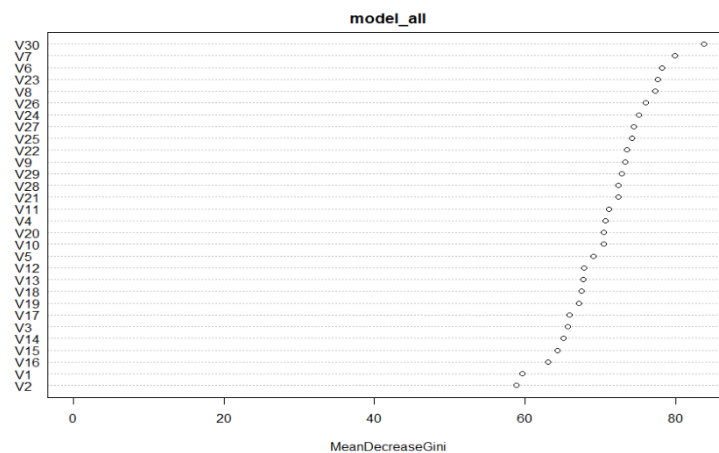
On remarque sur ce graphique qu'il n'est pas possible de différencier les 42 classes sur le plan principal.

4 Sélection de variables

La sélection de variable est toujours une étape très importante dans un projet. En effet, les modèles que nous sommes susceptibles d'utiliser ont un coût calculatoire assez important. Le but de cette partie est donc de minimiser ce coût en minimisant le nombre de variables.

4.1 Variables les plus importantes

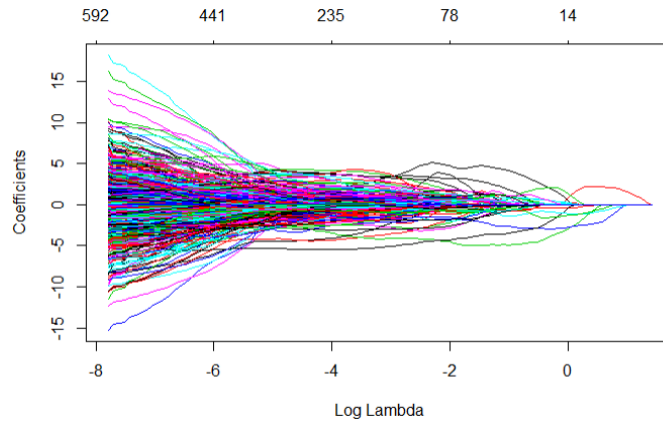
Nous avons commencé par créer un modèle *RandomForest* avec toutes les variables que nous disposons. Ensuite, la fonction `varImpPlot()` permet de faire un graphique des variables les plus importantes du modèle. Les résultats que nous obtenons est le suivant :



On récupère les variables de ce graphique et on relance le modèle. Les résultats que nous obtenons sont assez satisfaisants.

4.2 Lasso

Nous avons également testé une sélection de variables par la méthode *LASSO*. Les résultats que cette méthode nous a donné se présente sous cette forme :



5 Création du modèle

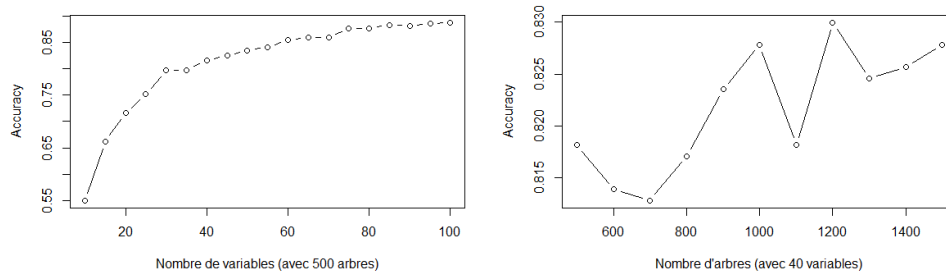
On décide de créer un modèle de type *Random Forest*. Même si il est possible de créer un modèle SVM multiclasse, le *Random Forest* est plus simple à mettre en oeuvre et plus robuste au sur-apprentissage. On essaye donc dans un premier temps de créer un modèle avec les variables sélectionnées les plus importantes (voir 4.1).

On utilise alors la méthode Lasso pour la sélection de variables car elle présentait de meilleurs résultats. On en fait varier les paramètres afin d'améliorer les résultats. Pour cela, on divise le jeu de données d'apprentissage afin de tester nos modèles.

Les paramètres qui varient sont les nombres de variables à inclure dans le modèle, et le nombre d'arbres créés dans l'algorithme.

5.1 Comparaison des modèles

Pour comparer les modèles, nous avons choisis de représenter la variation de l'*Accuracy*, qui nous donne le taux de bonnes prédictions, lorsque l'on fait varier le nombre d'arbres et le nombre de variables utilisées.



Nous avons donc opté pour une soixantaine de variables (correspondant à l'étape 35 du *LASSO*), car en ajouter plus n'influencait pas vraiment sur l'*accuracy* et nous permet d'obtenir un modèle parcimonieux. Un

nombre d'arbres trop élevé n'est pas nécessaire. Nous avons ici choisit de créer notre modèle avec 800 arbres car le temps de traitement n'était pas trop long, mais un nombre plus élevé n'est pas utile.

Nous avons pu caluler le taux d'*accuracy* de ces modèles (sachant que nous avons séparé avec un 70-30 sur le jeu de train). Nous obtenons donc :

- avec 43 variables et 800 arbres : 0.8503
- avec 43 variables et 1500 arbres : 0.8471
- avec 59 variables et 800 arbres : 0.8791
- avec 59 variables et 1500 arbres : 0.8663

5.2 Prédiction


Pour la prédiction, on applique donc un modèle *Random Forest* avec 800 arbres et les 43 variables les plus significatives du Lasso. On valide ensuite en prédiction pour chaque souche la classe revenant le plus souvent sur les 9 spectres. Si deux classes sont à égalité, on force une décision en regardant les probabilités d'appartenance à ces classes. Voici quelques exemples de notre méthode :

Souche A		
Classe	4	42
Nombre	7	2

Cas ou une classe est majoritaire

Dans le cas ou nous avons plusieurs classes pour la même souche mais que les occurences des classes sont différentes, nous choisissons la classe majoritaire. Ici nous prendrons donc la classe 4 car $7 > 2$.

Souche B			
Classe	2	37	39
Nombre	4	4	1



Souche B		
Classe	2	37
Nombre	5	4

Cas ou l'on doit *forcer* un choix

Dans le cas de l'égalité, on *force* le choix par probabilité du spectre classé dans la classe 39. Il est alors affecté à la classe la plus probable entre la 2 et la 37.

5.3 Conclusion

Nous aurions pu construire notre modèle avec toutes les variables que nous avons à disposition mais faire un modèle contenant plus de 600 variables permet de bonnes prédictions sur un jeu d'apprentissage mais ne pourra pas être adapter à d'autres jeux de données.

Le réseau de neurones aurait surement permis de meilleurs résultats mais nous n'avons pas eu le temps de le paramétrer pour avoir des résultats satisfaisant. Cependant, notre modèle est assez satisfaisant compte tenu des taux de prédiction obtenues.