SY09 - Analyse des données et Data-Mining

Printemps 2017 (P17)

TP4

Analyse discriminante, régression logistique et arbre de décision

Maxime Churin GI04 - Louis Denoyelle GI04

Introduction

Dans ce dernier TP, l'objectif est d'étudier différents modèles d'apprentissage des probabilités à posteriori comme l'analyse discriminante, la régression logistique et les arbres de décision. La première partie se consacre à la réalisation de fonctions sur `R` permettant l'apprentissage des différents modèles. Ensuite, ces fonctions seront testées sur différents jeux de données afin d'évaluer leurs performances en s'appuyant également sur les frontières de décision. Enfin, un dernier jeu de données, plus complexe, correspondant à un problème de détection de spams, sera analysé.

1 Programmation

Dans cette première partie, nous avons complété les fonctions permettant la réalisation d'analyses discriminantes et de régressions logistiques à la fois pour des cas linéaires et quadratiques. Nous détaillerons, ci-dessous, leur fonctionnement général et le code ainsi que les lignes manquantes, plus longuement annotées, seront donnés en annexe : `discriminante.R` et `regression-logistique.R`.

1.1 Analyse discriminante

L'analyse discriminante est basé sur la règle de Bayes. Elle repose sur le fait que le vecteur de caractéristiques suit, conditionnellement à chaque classe ω_k, une loi normale multidimensionnelle d'espérance μ_k et de variance Σ_k. Elle permet, entre autres, d'obtenir des estimations des probabilités à posteriori d'appartenance aux classes.

Les trois fonctions `adq.app, adl.app` et `nba.app` prennent en paramètres les données d'apprentissage sous la forme d'un tableau individus-variables et d'un vecteur associant à chaque individu sa classe. Elles retournent les paramètres estimés du modèle comme les proportions, les vecteurs de moyennes et la matrice de covariance des deux classes. La fonction `ad.val` calcule les probabilités à posteriori et les étiquettes associées à chaque individu pour un ensemble de données test sous la forme d'un tableau individus-variables. Ce calcul est réalisé grâce aux paramètres estimés par les fonctions précédentes.

1.2 Régression logistique

La régression logistique se base sur l'estimation directe des probabilités d'appartenance aux classes. Nous n'implémentons ici que son algorithme dans le cas d'un jeu de données binaire.

La fonction `log.app` prend en paramètres les données d'apprentissage sous la forme d'un tableau individus-variables, un vecteur associant à chaque individu sa classe, la présence d'une ordonnée à l'origine et le seuil qui servira de condition d'arrêt pour l'algorithme. Elle retourne l'estimateur du maximum de vraisemblance, le nombre d'itérations effectuées par l'algorithme de Newton-Raphson, et la valeur de la vraisemblance à l'optimum. La fonction `log.val` calcule les probabilités à posteriori et les étiquettes associées à chaque individu pour un ensemble de données test sous la forme d'un tableau individus-variables. Ce calcul est réalisé grâce l'estimateur du maximum de vraisemblance estimé par la fonction précédente.

Ces deux fonctions précédentes utilisent la fonction `postprob` qui retourne les probabilités à posteriori de la classe 1 d'un tableau individus-variables grâce à l'estimateur du maximum de vraisemblance. Une variante de la régression logistique linéaire consiste à modifier l'espace du modèle pour l'adapter à la régression logistique quadratique. C'est le rôle de la fonction `transform.quadratique` qui est donnée plus tard dans le sujet.

1.3 Vérification des fonctions

On utilise alors le script `script-1.R` joint au rapport pour vérifier nos fonctions. On observe alors des frontières de décision identiques à celles données dans le sujet du TP sur le jeu de données `Synth1-40`.

2 Application

La deuxième partie consiste en l'application des fonctions complétées dans la première partie. On étudie trois jeux de données simulés et deux jeux de données réels. Les taux d'erreurs de ces modèles ainsi que les frontières de décision de ces modèles seront étudiés. Les fonctions d'estimations des taux d'erreurs se retrouvent dans le fichier `functions-2.R` et le script d'exécution se nomme `script-2.R`, tous deux disponibles en annexe.

2.1 Test sur données simulées

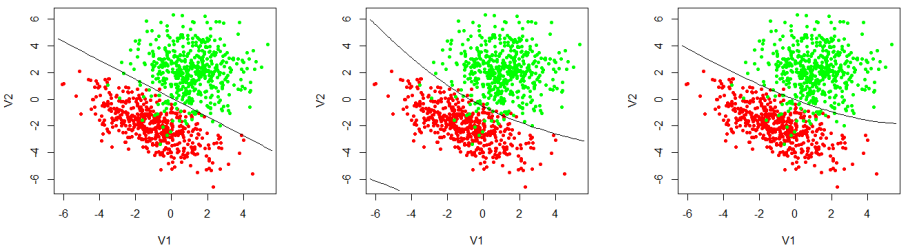
Les jeux de données étudiés ici sont des données simulées avec 1000 individus répartis en deux classes et modélisés par deux variables.

2.1.1 Données Synth1-1000

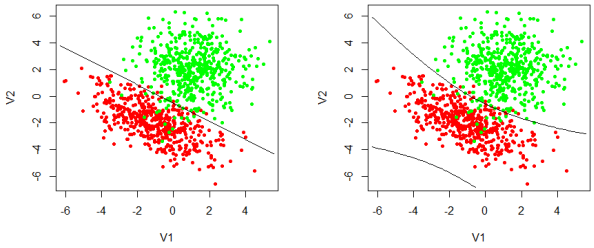
Nous étudions ici le jeu de données `Synth1-1000`.

Modèle	Taux d'erreur
ADL	4,06%
ADQ	3,16%
NBA	4,28%
LOG	3,28%
LOG2	3,19%
ARBRE	4,93%

Table 2.1 – Taux d'erreur moyen des modèles sur 20 répétitions.



(a) ADL **(b)** ADQ **(c)** NBA



(d) LOG **(e)** LOG2

Figure 2.1 – Frontières de décision des modèles.

Les taux d'erreurs pour les différents modèles sont relativement faibles ce qui nous indique que les deux classes sont assez distinctes. Cependant, la répartition des données nous apprend que les matrices de variances $\sum_k$ des lois normales bivariées de ces classes sont assez différentes. Or, l'hypothèse d'homoscédasticité de l'analyse discriminante linéaire suppose que : $\sum_k = \sum, k \in \{1, ..., g\}$. L'hypothèse n'étant pas respectée, cela explique pourquoi l'analyse discriminante linéaire obtient un taux d'erreur plus important que l'analyse discriminante quadratique. Le classifieur bayésien naïf suppose également cette hypothèse, son taux d'erreur est donc lui aussi important.

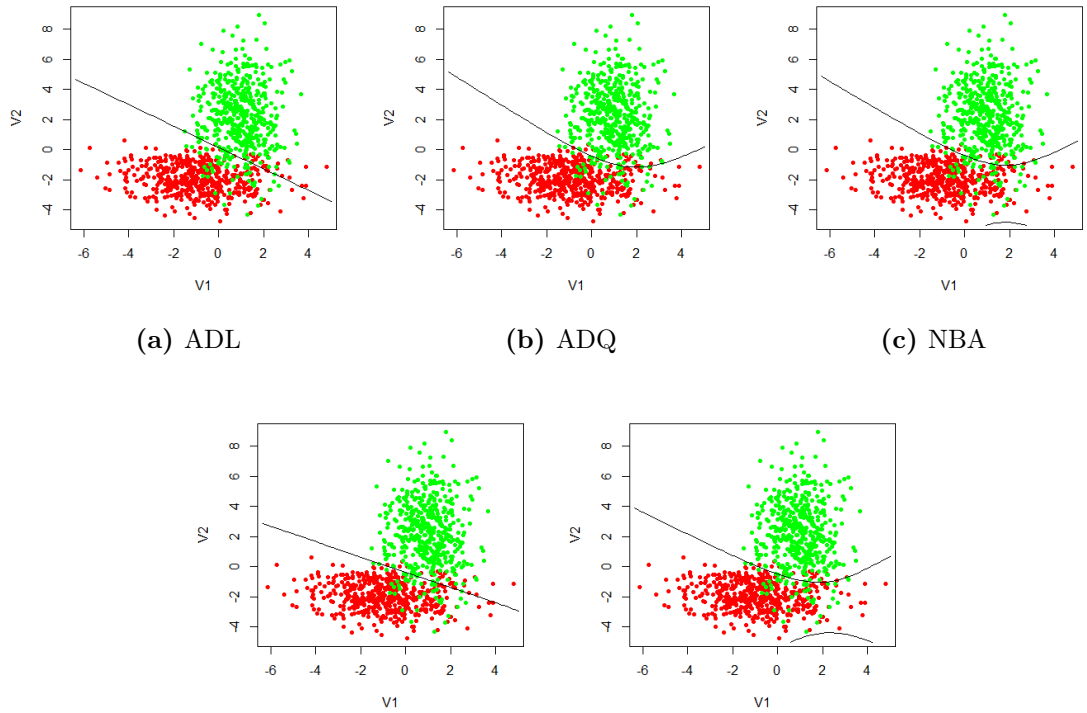
On remarque également que la régression logistique simple et la régression logistique quadratique donnent d'aussi bons résultats que l'analyse discriminante quadratique. Le modèle de l'arbre binaire obtient lui le plus haut taux d'erreur.

2.1.2 Données Synth2-1000

Nous étudions ici le jeu de données `Synth2-1000`.

Modèle	Taux d'erreur
ADL	8,06%
ADQ	6,22%
NBA	6,17%
LOG	7,19%
LOG2	6,43%
ARBRE	7,10%

Table 2.2 – Taux d'erreur moyen des modèles sur 20 répétitions.



(a) ADL **(b)** ADQ **(c)** NBA

(d) LOG **(e)** LOG2

Figure 2.2 – Frontières de décision des modèles.

L'analyse graphique de la répartition des données de `Synth2-1000` montre que les matrices de variances $\sum_k$ sont très différentes, de façon encore plus marquée que dans le jeu de données `Synth1-1000`. Cela se traduit par une frontière de décision difficile à déterminer, et encore une fois l'analyse discriminante quadratique donne des meilleurs résultats que l'analyse discriminante linéaire.

Étant donné les formes des nuages de points répartis de façon relativement uniforme autour de leur centre, cela nous laisse supposer que les matrices de variances $\sum_k$ sont diagonales. Cela respecte donc l'hypothèse du classifieur bayésien naïf qui suppose que $\sum_k$ est diagonale d'où son taux d'erreur assez faible.

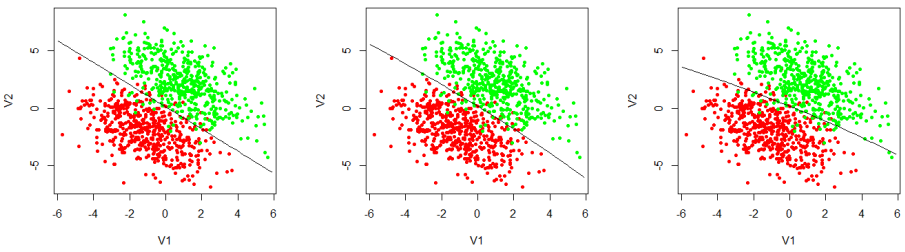
La régression logistique quadratique nous donne un taux d'erreur très proche de l'analyse discriminante quadratique, alors que ceux des arbres et de la régression logistique simple sont légèrement plus élevés.

2.1.3 Données Synth3-1000

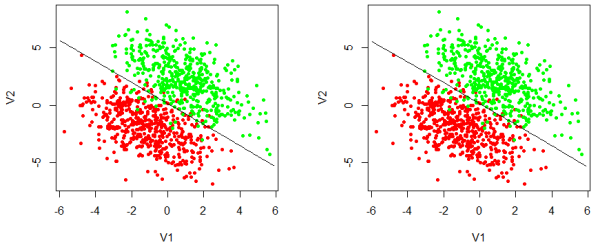
Nous étudions ici le jeu de données `Synth3-1000`.

Modèle	Taux d'erreur
ADL	4,04%
ADQ	4,07%
NBA	4,71%
LOG	4,28%
LOG2	4,25%
ARBRE	6,08%

Table 2.3 – Taux d'erreur moyen des modèles sur 20 répétitions.



(a) ADL **(b)** ADQ **(c)** NBA



(d) LOG **(e)** LOG2

Figure 2.3 – Frontières de décision des modèles.

On remarque ici que la répartition des nuages de points est assez similaire entre les deux classes, on peut alors supposer que les matrices de variances $\sum_k$ sont égales. Les frontières de décision sont assez évidentes et les analyses discriminante linéaire et quadratique, ainsi que les classifieur bayésien naïf donnent de bons résultats.

On obtient également des bonnes performances avec les deux méthodes de régression logistique, et le modèle des arbre de décision donne, encore une fois, un taux d'erreur légèrement plus élevé.

2.2 Test sur données réelles

Après avoir étudié des données simulées, nous analysons maintenant les performances des modèles sur deux jeux de données réels.

2.2.1 Données « Pima »

Le jeu de données correspond ici à une population de femmes atteintes de diabète. Les variables correspondent à des caractéristiques physiques comme le taux de glucose et à des données comme l'âge ou le nombre de grossesses.

Modèle	Taux d'erreur
ADL	22,32%
ADQ	23,88%
NBA	23,29%
LOG	22,12%
LOG2	24,75%

Table 2.4 – Taux d'erreur moyen des modèles sur 100 répétitions.

Les taux d'erreurs obtenus pour chacun des modèles sont très élevés. Ce résultat n'est pas surprenant au vu des analyses du jeu de données à travers les précédents TPs effectués. Malgré que le jeu de données comporte 7 variables, soit 5 de plus que dans les données simulées étudiées précédemment, aucune des variables impliquent une séparation nette entre les deux classes (diabète ou non-diabète). Ceci réduit donc l'efficacité des modèles, et explique un apprentissage moins performant pour différencier pour les deux classes.

2.2.2 Données « breast cancer Wisconsin »

On étudie la prédiction du niveau de gravité d'une tumeur à partir de descripteurs physiologiques. Le jeu de données comporte 683 individus répartis en deux classes et définis par 9 variables.

Modèle	Taux d'erreur
ADL	4,52%
ADQ	5,30%
NBA	4,00%
LOG	4,02%
ARBRE	5,33%

Table 2.5 – Taux d'erreur moyen des modèles sur 100 répétitions.

On remarque que les modèles donnant les meilleurs résultats sont l'analyse discriminante linéaire, le classifieur bayésien naïf et la régression logistique classique. Globalement, tous ces résultats sont assez bons, ce qui montre que les données des deux classes sont assez distinctes. Cependant, compte tenu de l'étude de cas, ce taux d'erreur est tout de même trop élevé lorsqu'il s'agit d'identifier un problème aussi délicat que la prédiction de la gravité d'une tumeur.

3 Challenge : données « Spam »

Le jeu de données `Spam` est composé de 4601 messages électroniques décrits par 57 variables quantitatives et une variable qualitative. Ces 57 variables représentent des indicateurs relatifs à la composition des messages électroniques comme par exemple l'occurrence et la proportion de certains mots et caractères, la taille du plus long mot, etc. On observe que de nombreuses réalisations sont égales à 0.00 et cela va nous compliquer l'analyse. L'unique variable qualitative référence l'appartenance à la classe 1 ou 2 qui équivaut à un booléen pour décrire le type de courriel : spam ou non spam.

On tente, dans un premier temps, d'utiliser les modèles précédemment étudiés mais seulement la méthode des arbres de décision nous donne un taux d'erreur de l'ordre de $\varepsilon = 7.87$ %. Les modèles d'analyses discriminantes précédents ne fonctionnent pas car on obtient des cas où le calcul des probabilités à posteriori donne une valeur indéfinie avec un dénominateur et un numérateur nuls ce qui empêchent l'affectation d'une étiquette au courriel. On a aussi des problèmes de surapprentissage avec ces modèles, en effet, l'ensemble d'apprentissage possède un nombre trop important de variables et de données qui empêche la généralisation des caractéristiques des données et perturbe donc la prédiction sur de nouveaux jeux de données. Une technique pour lutter contre le surapprentissage est la validation croisée par exemple. Pour ce qui est des modèles de régression logistique on se retrouve dans l'incapacité de calculer la matrice inverse dans l'algorithme de Newton-Raphson car le déterminant est nul.

Nous avons décidé de mettre en place une stratégie par sélection de variables de type `RandomForest`. Ce modèle est basé sur le *bagging* appliqué aux arbres de décision. Le *bagging* permet de faire la moyenne des prévisions de plusieurs modèles construits à l'aide d'échantillons bootstrap (tirages avec remises). On réduit ainsi la variance et le taux d'erreur. Le terme "forêts aléatoires" se caractérise par l'aléa au niveau des variables. En effet, l'ajout d'un noeud aux arbres construits avec un échantillon bootstrap d'individus se fait sur un sous-ensemble de variables tirées aléatoirement. Pour finir, on obtient plusieurs arbres et donc plusieurs prédictions pour chaque individu, on choisit alors la classe la plus fréquente. Ceci s'applique uniquement à la classification ou prédiction d'une variable qualitative comme c'est le cas ici.

On obtient finalement un plus faible taux d'erreur/*Out of Bag error* avec 500 arbres générés et 7 variables testées à chaque division : $\varepsilon = 4.46$ %.

Ce modèle, en plus de nous donner un très bon taux d'erreur, permet de repérer efficacement les facteurs qui impactent le plus la classe à déterminer : ici le fait qu'un mail soit un spam ou non.

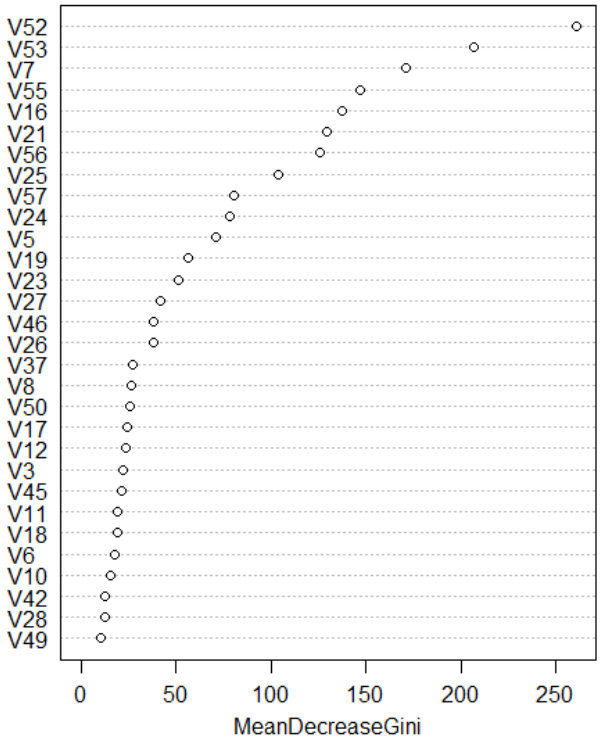


Figure 3.1 – Importance des variables explicatives sur la variable à prédire

En appliquant la méthode du coude on ne garde que les variables avec un indice supérieur ou égal à $V26 = 38$. On peut alors retenter nos précédents modèles en ayant réduit la dimension. Les modèles d'analyses discriminantes restent malheureusement inutilisables sauf si on enlève les valeurs indéfinies. Pour ce qui est des modèles de régression logistique, la version linéaire nous donne maintenant un taux d'erreur de $\varepsilon = 9$ % et le modèle quadratique reste inutilisable.

Pour conclure sur cette étude, ces résultats sont plutôt cohérents. On obtient un taux d'erreur faible via le modèle `RamdomForest` qui est adapté aux données, un taux moyen d'erreur intermédiaire pour les arbres de décision qui est amélioré par le `RamdomForest` et, enfin, le pire taux moyen pour la régression logistique. Ce dernier est tout à fait logique car en réduisant la dimension aux variables les plus significatives on perd de l'information ce qui rend la prédiction plus difficile.

Conclusion

Lors de ce TP, nous avons pu observer que les résultats de nos fonctions sur différents jeux de données peuvent être très variables. C'est valable pour des données réelles ou simulées et cela montre l'importance de choisir le bon modèle. La dernière partie, sur les données `Spam`, nous laissait plus de liberté quant à leur étude. Nous avons alors pu explorer d'autres modèles pour pallier à la défaillance des modèles précédents. Ce TP, ainsi que tous les TPs de SY09 de façon générale, nous a ouvert au problème complexe qu'est la classification et aux différents modèles qui permettent d'y répondre à travers des jeux de données variés et l'utilisation du logiciel `R`.