Université Grenoble Alpes

Master SSD, BDA et IC
Enseignant : Jonathan El Methni
Apprentissage Statistique

TP 5 : MODÈLES PROBABILISTES DE CLASSIFICATION

Exercice : Classifieurs

L'objectif de cet exercice est d'apprendre à manipuler les outils `R` permettant de mettre en oeuvre les modèles LDA, QDA, régression logistique et la classification naïve bayésienne.

Pour cela nous allons comparer leurs performances de prédiction sur un jeu de données simulées en deux dimensions.

1. Charger le jeu de données `tp5-exo2.Rdata` et le visualiser.
   - il est constitué d'un jeu d'apprentissage `(X.train,y.train)` contenant 400 observations en 2 dimensions appartenant à deux catégories ("1" et "2"), et d'un jeu de test `(X.test,y.test)` contenant 200 observations.
   - on représentera sur un même graphique les données d'apprentissage et de test, avec un code couleur indiquant la catégorie ("1" ou "2"), et un symbole indiquant le jeu de données (apprentissage ou test), via l'option `pch`.

2. Mettre en oeuvre l'approche LDA :
   (a) apprendre le modèle à partir des données d'apprentissage en utilisant la fonction `lda` du package `MASS`.
   (b) obtenir les prédictions sur les données de test grâce à la fonction `predict`. Quelles informations fournit cette fonction ?
   (c) représenter les probabilités a posteriori d'être dans la classe "1" en fonction des vrais classes définies par `y.test` en utilisant la fonction `boxplot`. Qu'observez-vous ?
   (d) construire la matrice de confusion obtenue et calculer le taux de bonne classification comme dans le TP précédent.
   (e) construire à nouveau la matrice de confusion avec l'instruction `confusionMatrix` du package `caret`.

3. Faire de même avec l'approche QDA.
   - Les performances sont-elles meilleures ? Est-ce surprenant au vu du graphique construit précédemment ?
   - pour cela on s'appuie sur la fonction `qda` du package `MASS` qui fonctionne exactement comme la fonction `lda`.

4. Faire de même avec la régression logistique. Comment ce modèle se positionne t'il par rapport aux approches génératives ?
   - on utilise pour cela la fonction `glm` et la fonction `predict` correspondante. Ces fonctions suivent la même syntaxe que celle de la fonction `lm` pour effectuer une régression linéaire.
   - pour constuire le modèle :
     `fit = glm(y.train ~ x1+x2, data = data.frame(X.train), family = "binomial")`
   - pour effectuer les prédictions :
     `p = predict(fit, newdata = data.frame(X.test), type = "response")`
   - notons que le vecteur `p` contient les probabilités a posteriori d'être dans la classe "positive". La classe positive est choisie automatiquement comme le deuxième "niveau"" considéré pour l'apprentissage : nous avions ici des "1"" et des "2"", la classe 2 est considérée comme étant la classe positive.

5. Faire de même avec la classification naïve bayésienne à l'aide du package `e1071`.
   - on utilise pour cela la fonction `naiveBayes` et la fonction `predict` correspondante. Ces fonctions suivent la même syntaxe que celle de la fonction `lm` pour effectuer une régression linéaire.
   - pour constuire le modèle :
     `model = naiveBayes(y.train ~ x1+x2, data = data.frame(X.train))`
   - pour effectuer les prédictions :
     `prednaiveBayes = predict(model, newdata = data.frame(X.test), type = "raw")`
   - que contient le vecteur `prednaiveBayes` ?

6. Enfin en faire de même à l'aide des k plus proches voisins à l'aide du package `class` comme vu dans le TP précédent. Vous êtes libre de choisir la valeur de k qui vous semble pertinente.

7. Comparer les résultats obtenus par les cinq approches en terme de courbe ROC et conclure.

   Afin d'extraire les "probabilités" à posterori d'être dans chacune des classes pour l'approche knn il faut utiliser l'instruction `knn3Train` avec l'option `prob=TRUE`. Il est alors possible de récupérer les valeurs à l'aide de l'instruction `attr( , "prob")` appliquée à l'instruction obtenue avec `knn3Train` comme suit :

```
knn_mod=knn3Train(X.train, X.test, y.train, k = 5, prob = TRUE)
score5= attr(knn_mod, "prob")[,2]
```

8. A l'aides des instructions suivantes représenter les 5 frontières définies par les approches LDA, QDA, k-PPV, Regression Logistique et le classifieur Naif de Bayes sur la figure générée à la question 1.

```
TailleGrille=300
grille.x=seq(from = min(X.train[,1]), to = max(X.train[,1]), length.out = TailleGrille)
grille.y=seq(from = min(X.train[,2]), to = max(X.train[,2]), length.out = TailleGrille)
grille.df=data.frame(expand.grid(x = grille.x, y = grille.y))
names(grille.df)=c("x1","x2")
```

```
# Graphique sur lequel rajouter les frontières
plot(X.train[,1],X.train[,2],xlab = "X1",ylab = "X2",col = y.train, pch =1,main = "Classifieurs")

# Ajout frontière LDA
# Attention il faut obtenir : modelda
prdlda=as.numeric(predict(modelda,grille.df)$class)

# Ajout Moyennes dans les groupes
points(modelda$means, pch = "+", cex = 3, col = c("black", "red"))

contour(x=grille.x,y=grille.y,z=matrix(prdlda, nrow = TailleGrille,ncol =TailleGrille)
,levels = c(1, 2), add=TRUE,drawlabels=FALSE,lwd=3,lty=2,col="green")

# Ajout frontière QDA
contour(x = grille.x, y = grille.y, z = matrix(prdqda, nrow = TailleGrille, ncol = TailleGrille),
levels = c(1, 2), add = TRUE, drawlabels = FALSE,lwd=3,lty=2,col="orange")

# Ajout frontière Knn
prdknn5=matrix(probknn5, nrow=length(grille.x), ncol=length(grille.y))

contour(x = grille.x, y = grille.y, prdknn5, levels = c(1, 2), add = TRUE,
drawlabels = FALSE,lwd=3,lty=2,col="darkgreen")

### Ajout frontière Bayes Naif
y_pred=predict(modelnaif, newdata=grille.df)
prdnaif=matrix(y_pred, nrow=length(grille.x), ncol=length(grille.y))

contour(x = grille.x, y = grille.y, prdnaif, levels = c(1, 2), add = TRUE,
drawlabels = FALSE,lwd=3,lty=2,col="darkgreen")

#### Ajout frontière Regression Logistique ####
y_pred=predict.glm(logreg.fit, newdata=grille.df, type="response")
prob_logreg=matrix(y_pred, nrow=length(grille.x), ncol=length(grille.y))

contour(x = grille.x, y = grille.y, prob_logreg, levels = 0.5, add = TRUE,
drawlabels = FALSE,lwd=3,lty=2,col="darkgreen")
```

9. Représenter la frontière définie par la régression logistique sur la figure générée à la question 1 en obtenant les coefficients de la droite séparatrice.

   — On rappelle que la frontière est donnée par l'équation $f(x) = 0$, où $f(x)$ est la fonction linéaire définie par les coefficients du modèle linéaire, soit ici

   $$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

   — Pour tracer la frontière on peut donc utiliser la fonction `abline` qui permet de faire figurer sur un graphique existant une droite d'équation $y = a + bx$ à partir des paramètres a et b.

   — Il faut donc trouver l'expression $x_2 = a + bx_1$ à partir des coefficients $(\beta_0, \beta_1, \beta_2)$ de la régression logistique.

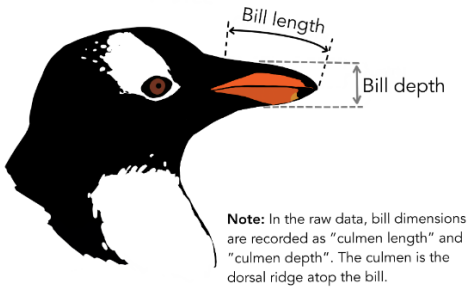
Application à un jeu de données réelles :

L'objectif de cet exercice est mettre en application les classifiers du cours sur le jeu de données `penguins` présent dans le package `palmerpenguins` de `R` . Cette fois nous nous intéresserons

à un problème de classification à 3 classes qui correspondent aux 3 espèces de pingouins présentées ci-dessous. Ces pingouins vivent dans l'archipel Palmer un groupe d'îles au large de la côte nord-ouest de la péninsule Antarctique.

La variable qui va donc superviser l'apprentissage est `species`. Nous allons essayer de classifier les 3 espèces pingouins des dimensions de leurs becs :

— bill_length_lmm : longueur du bec en millimètres,
— bill_depth_lmm : longueur du bec en millimètres.



Nous ne tiendrons par compte des variables restantes :

— flipper_length_mm : longueur de la nageoire en millimètres,
— body_mass_lg : poids du corps en grammes,
— island : île d'origine,
— sex : sexe du pingouin,
— year : année d'étude.

1. Pour cela on commencera par installer et charger le package `palmerpenguins` pour pouvoir travailler sur le jeu de données `penguins`. On aura également besoin d'installer et de charger les packages `dplyr` et `caret`.

2. Commençons par visualiser le jeu de données

   ```
   str(penguins)
   head(penguins)
   ```

3. On nettoye le jeu de données à l'aide de l'instruction :

   ```
   Pingouins=na.omit(penguins)[,c(-2,-8)]
   ```

4. On fixe le seed et on sépare nos données en Train (80%) et Test (20%) :

   ```
   set.seed(123)
   training.individuals=Pingouins$species%>%createDataPartition(p = 0.8,list = FALSE)
   TrainData=Pingouins[training.individuals,]
   TestData=Pingouins[-training.individuals,]
   ```

5. Mettez en oeuvre les différents classifieurs du cours en représentant leurs frontières sur le jeu données, donner les matrices de confusions correspondantes et conclure.

6. Vous pouvez reprendre l'étude en prenant d'autres variables.