Université De Lille
Faculté des Sciences et Technologies
Département de Mathématiques

Compte rendu du projet
Analyse de données

Réalisé par :

- Abdelmonssif *Oufaska*
- Kamal *Samnouni*

Enseignant :

- *Prof.* Charlotte *Baey*

Année Universitaire :
2020/2021
28 Décembre 2020

Introduction

On s'intéresse dans ce projet à la classification de différentes espèces animales. Il y a deux parties, portant sur deux jeux de données distincts. Dans la première partie, on cherche à construire un modèle permettant de prédire la classe d'un animal en fonction d'un certain nombre de caractéristiques, et dans la deuxième partie on cherche à regrouper l'ensemble d'animaux en plusieurs sous catégories à l'aide des données stockées sous forme de fichier.jpg.
Les packages utilisés dans ce projet sont les suivants :
—**DBSCAN**
—**FactoMineR**
—**factoextra**
—**ggplot2**
—**nnet**

Table des matières

Prédiction de la classe d'un animal

L'objectif de cette partie est de prédire la classe d'un animal en fonction de plusieurs de ses caractéristiques. Sur le jeu de données considéré, la classe d'un animal peut être : mammifère, oiseau, reptile, poisson, amphibien, insecte ou invertébré. Nous avons 17 variables et 101 observations. Les variables sont toutes booléennes (1 : vrai, 0 : faux) sauf mention contraire :

—**animal_name** : le nom de l'animal.
—**hair** :la présence de poils.
—**feathers** :la présence de plumes.
—**eggs** :indique si l'animal pond des œufs.
—**milk** :indique si l'animal produit du lait.
—**airborne** :la possibilité de voler.
—**aquatic** :indique si c'est un animal aquatique.
—**predator** :indique si l'animal est un prédateur.
—**toothed** :la présence de dents.
—**backbone** :la présence d'une colonne vertébrale.
—**breathes** :la possibilité de respirer.
—**venomous** :indique si l'animal est venimeux.
—**fins** :la présence de palmes.
—**legs** :le nombre de pattes.
—**tail** :la présence d'une queue.
—**domestic** :indique s'il s'agit d'un animal domestique.
—**catsize** :indique si l'animal a la taille d'un chat.

Dans un premier temps on va recoder les variables qualitatives en facteurs, à l'aide de la fonction as.factor, et pour plus de lisibilité, on va renommer les niveaux de la variable cible class_type en toutes lettres. Voici un aperçu de ces données :

animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	class_type
aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	mammifère
antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	mammifère
bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	poisson
bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	mammifère
boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	mammifère
buffalo	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	mammifère

On propose maintenant de faire une analyse descriptive à notre base de données, on a les résultats suivants :

	Hair	Feathers	Eggs	Milk	Airborne	Aquatic	Predator	Toothed
Classe 0	58	81	42	60	77	65	45	40
Classe 1	46	20	59	41	24	36	56	61

	Backbone	Breathes	Venomous	Fins	Tail	Domestic	Catsize
Classe 0	18	21	93	84	26	88	57
Classe 1	83	80	8	17	75	13	44

On note également que dans notre base de données, on dispose de 41 mammifères, 20 oiseaux, 5 reptiles, 13 poissons, 4 amphibiens, 8 insectes et 10 invertébrés. Et 23 animaux n'ont pas de pattes, 27 animaux ont 2 pattes, 38 animaux ont 4 pattes, 1 animal a 5 pattes, 10 animaux ont 6 pattes et 2 animaux ont 8 pattes.
Nous allons proposer un modèle permettant de prédire la classe d'un animal à l'aide des différentes variables de la table.

1 Modèle de prédiction

Afin de construire un modèle permettant de prédire la classe d'un animal et d'évaluer ses performances, on va séparer la base de données en une base d'apprentissage (85% de l'échantillon) et une base de test (15% de l'échantillon). Les variables de notre base sont quantitatives. On veut expliquer la variable "class_type" en fonction des autres variables, on suggère d'ajuster un modèle de régression logistique multinomial, à l'aide de la fonction multinom de la librairie nnet . Nous avons les résultats suivants :
Si on utilise toutes les variables dans notre modèle de régression, ensuite on calcule

des prédictions, on trouve un taux d'erreur qui varie entre 0% et 20% pour chaque recompilation, le graphe ci-dessous représente les résultats qu'on a trouvé. Le taux d'erreur moyen trouvé est 6%

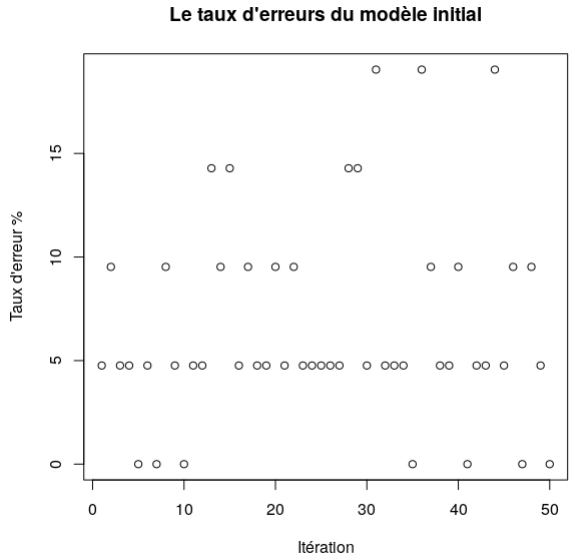

Figure 1 : Le taux d'erreur du modèle initial

Dans un premier temps, on va appliquer la fonction step-AIC à notre modèle de régression pour déterminer les caractéristiques qui influencent le plus l'affectation d'un animal dans chacune des 7 classes. On obtient que les variables qui influencent l'affectation d'un animal dans chacune des classes au sens du critère d'AIC sont : feathers, milk, backbone et breathes1.
Si on calcule le taux d'erreur de prédiction avec ce nouveau modèle on trouve : 4.76%.
Maintenant, si on recompile notre programme, on trouve un nouveau modèle au sens du critère d'AIC avec les variables suivantes : hair, feathers, aquatic, breathes, venomous et tail. On calcule le taux d'erreur de prédiction avec ce nouveau modèle on trouve : 9.52%.
À chaque recompilation on trouve un nouveau modèle avec un taux d'erreur diffèrent, on s'intéresse maintenant à faire une étude sur la nature de la liaison des variables afin de trouver un modèle performant avec moins de variables qui influencent le plus l'affectation d'un animal dans chacune des 7 classes.

1.1 Étude sur les variables

Comme on a vu dans la section précédente la fonction step-AIC nous propose un modèle différents avec des variables différentes à chaque recompilation. La raison qui nous a poussé de poser la question sur la nature de la liaison entre les variables de notre jeu de données.

On propose de faire l'analyse des correspondances multiples à l'aide de la fonction MCA de la librairie FactoMineR, pour visualiser les différentes liaisons entre les variables.
Le graphe ci-dessous représente la projection des variables sur le premier plan principal d'inertie égale à 44.8%

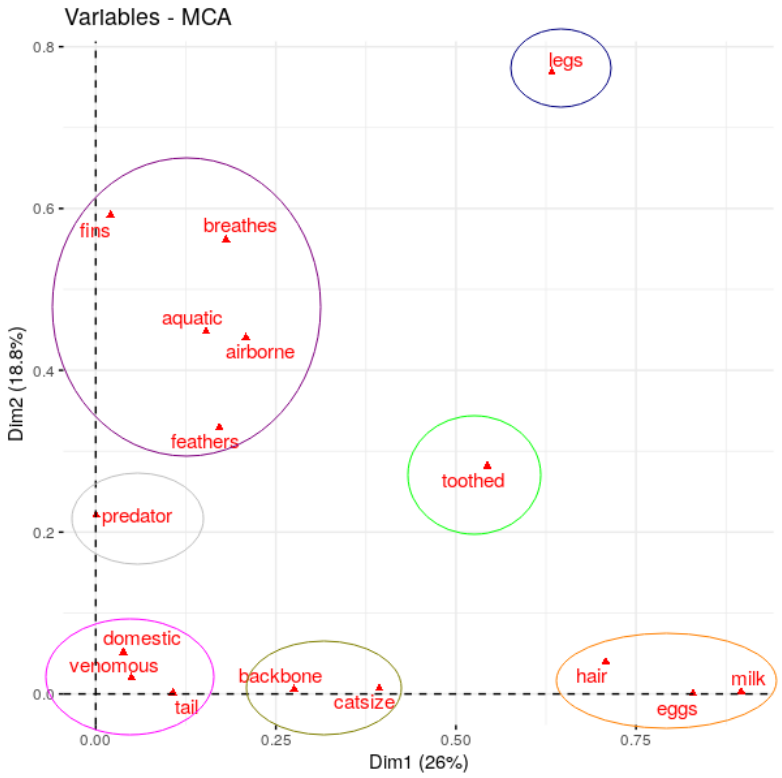

Figure 2 : Projection des variables sur le premier plan principal

On remarque qu'on peut identifier 7 différents groupes séparables contenant des variables liés entre eux :
- Groupe 1 :{legs}
- Groupe 2 :{fins,breathes,aquatic,airborne,feathers}
- Groupe 3 :{toothed}
- Groupe 4 :{hair,eggs,milk}
- Groupe 5 :{backbone,catsize}
- Groupe 6 :{predator}
- Groupe 7 :{domestic,venomous,tail}

On retire une variable qui se situe le plus possible au milieu de chaque groupe, sauf le groupe 7 qui est proche au centre du premier plan, les variables de ce groupe ont pas un grand effet sur l'étude. Les variables de notre modèle trouvé à l'aide de l'ACM sont : eggs, aquatic, legs, toothed, catsize et predator. Avec ce nouveau modèle on trouve un

taux d'erreur égale à 9%, si on applique la fonction Step-AIC au modèle plusieurs fois, on trouve un seul modèle final avec des variables qui influencent le plus l'affectation d'un animal dans chacune des 7 classes :eggs, aquatic, legs, toothed et catsize.
Le graphe ci-dessus représente les taux d'erreurs trouvés par ce modèle final en chaque recompilation.

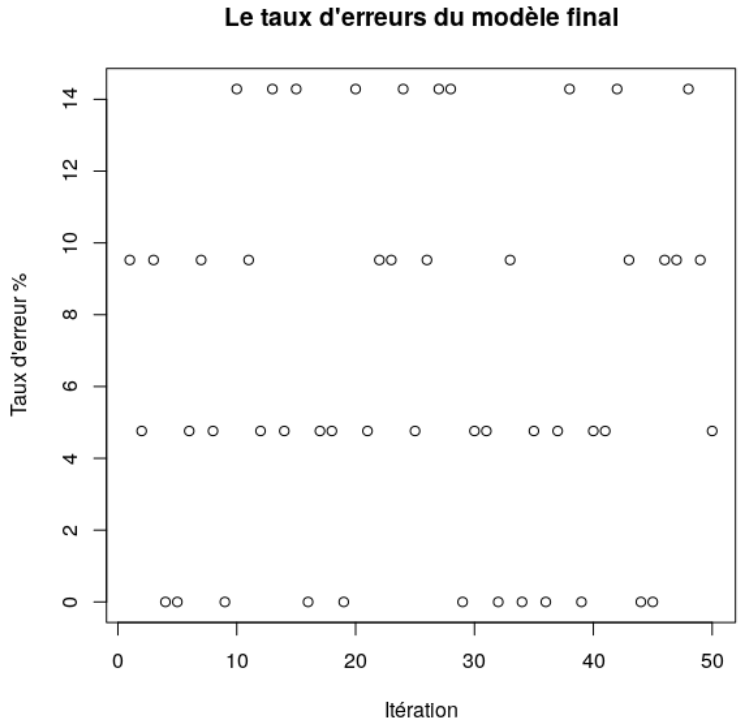

Figure 3 : Le taux d'erreurs du modèle final

Le taux d'erreur moyen trouvé est égale à 6%, on trouve le même taux d'erreur du premier modèle avec moins de variables. On note qu'on a trouvé les mêmes résultants, si on change un variable par un autre variable de son même groupe. Donc pour construire un modèle de régression performant avec moins de variables et qui influencent le plus l'affectation d'un animal dans une classe, il suffit de choisir une seule variable dans chacun de 5 groupes qu'on a déterminé.

Classification de photos d'animaux

Ici, on s'intéresse à la classification d'animaux en plusieurs sous-catégories à l'aide de photographies. Les données sont stockées sous forme de fichiers.jpg (un fichier par animal).

Dans un premier temps, on va importer sous R les données et les mettre en forme afin de pouvoir les analyser, c'est-à-dire en le codant comme un vecteur. Pour pouvoir traiter tous les fichiers de façon homogène, on ne gardera que les 3 premières dimensions de l'image,correspondant aux couches de couleurs RGB (Rouge, Vert, Bleu). Certaines images contiennent une quatrième couche, qui correspond à la transparence de l'image, et à laquelle on ne s'intéresse pas dans ce projet. On doit obtenir une base de données contenant autant de lignes que de photos,et autant de colonnes que de pixels dans les trois canaux R, G et B (donc 3 fois le nombre de pixels de la photo).

On dispose d'une base de données contenant 49152 variables (les pixels) et 4738 individus (les images). On cherche à proposer une classification en sous-catégories, c'est-à-dire de déterminer le nombre de sous-catégorie d'animaux dans notre base. Ensuite on va comparer différentes approches et évaluer leurs performances. En raison de la faible puissance de notre ordinateur, on va prendre aléatoirement 2004 images.

On note qu'on dispose beaucoup de variables par rapport au nombre d'individus, et on va donc chercher à réduire la dimension de notre base de données. Pour cela, on propose à faire une analyse en composantes principales(ACP), réalisée sur notre base. On gardons les axes dont les valeurs propres sont supérieures à la moyenne et qui correspond au plus de 90% de l'inertie totale, pour garder le minimum d'information. Pour ce choix on retient 44 axes qui correspond à 92% de l'inertie totale. On peut ensuite afficher les images compressées, c'est-à-dire l'approximation de chaque image initiale par sa projection dans le sous-espace engendré par les 44 premiers axes factoriels. On a le graphique ci-dissous pour le choix de 44 axes :

On remarque que les 44 axes ne semblent pas suffisants pour reconstituer l'image initiale. On propose de choisir plus d'axe pour mieux visualiser les images, on décide de garder 200 axes, on a le graphique ci-dessous :

On constate une différence entre les deux choix, avec 200 axes factoriels on visualise un peu mieux l'image.
Le premier plan factoriel correspond à 83% de l'inertie, on suppose que ce premier plan contient suffisamment d'information, on va visualiser les différents résultats de notre étude sur ce premier plan factoriel. Le graphique ci-dessous représente la distribution des individus dans ce plan

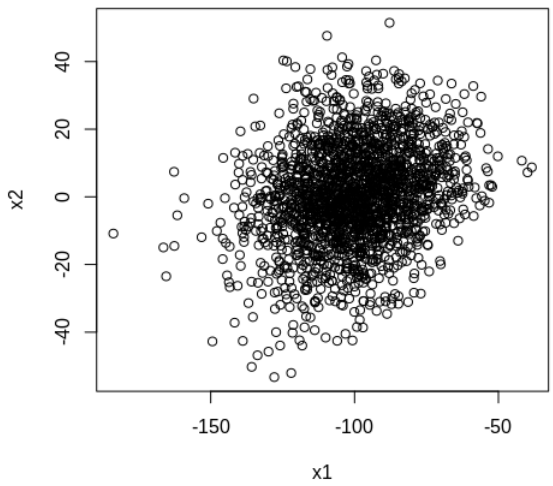

Figure 1 : Distribution des individus dans le premier plan factoriel

Maintenant, on veut créer une typologie des individus, à partir de l'ensemble des des axes factoriels, on cherche à regrouper les individus en sous-groupes ou classes homogènes, ce qu'on appelle classification non supervisée.
On commence par la méthode de partitionnement qui identifie les zones de forte densité du nuage de points, plus précisément on s'intéresse à l'algorithme de DBSCAN. On se fixe $m = 10$, on a le graphique ci-dessous

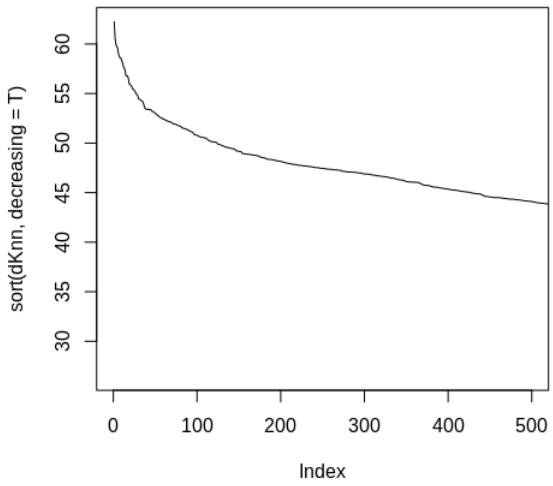

Figure 2 : Graphe de la distance entre chaque point et son (m-1)-ème voisin

D'après ce graphique on détecte un coude correspondant à une distance de 54, on pourra alors fixer $\epsilon = 54$. Avec ce choix, on trouve les résultats suivants :

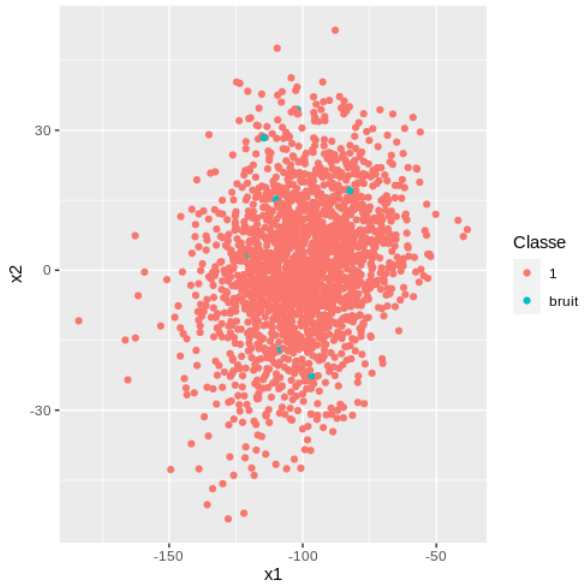

Figure 3 : Illustration des résultats de l'algorithme DBSCAN

La distance entre les individus dans le premier plan factoriel est très grande, les points sont classés dans un seul cluster sauf 7 points qui sont considérés comme du bruit. Cela signifie que l'algorithme de DBSCAN ne pourra pas trouver des clusters de densités différentes.

En revanche, on s'intéresse à la classification ascendante hiérarchique (CAH) et la méthode des k-means, qui ne font aucune hypothèse sur la distribution des données, et qui reposent plutôt sur la géométrie du nuage de points.
Premièrement avec CAH on choisit la distance de Ward, qui permet d'agréger les classes entre elles. On présente le graphique ci-dessous :

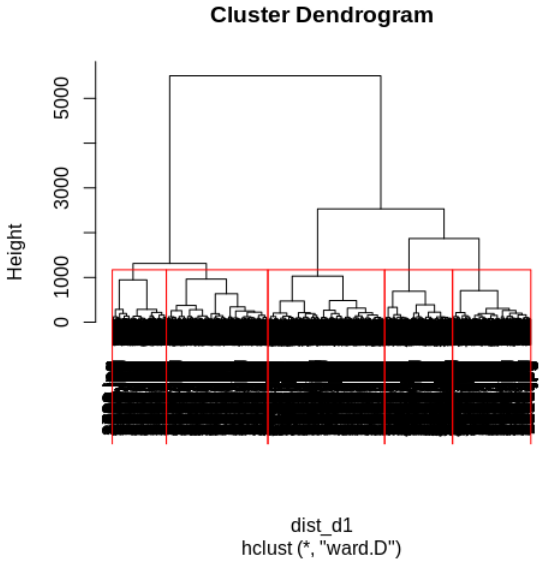

Figure 4 : Dendrogramme obtenu par le critère de Ward

D'après ce dendrogramme, on constate que le choix de cinq classes est le plus pertinent. Le graphique ci-dessous représente la distribution des individus par classe dans le premier plan factoriel.

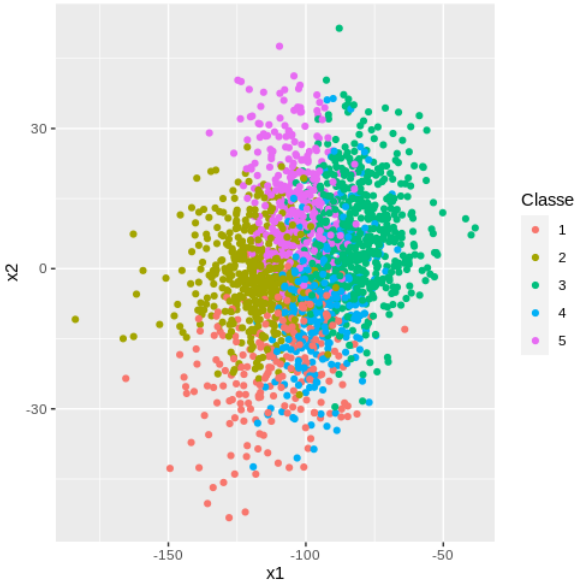

Figure 5 : Résultat de l'algorithme CAH sur le jeu de données

On constate que avec 5 clusters, la méthode de CAH ne donne pas un bon un résultat vu la visualisation du partitionnement des individus sur le premier plan factoriel qui porte plus de 83% de l'information n'est pas très convaincant.

Maintenant, on va procéder avec la méthode des K-means. On commence par tracer l'inertie intra-classes en fonction du nombre de classes, on a le graphique ci-dessous :

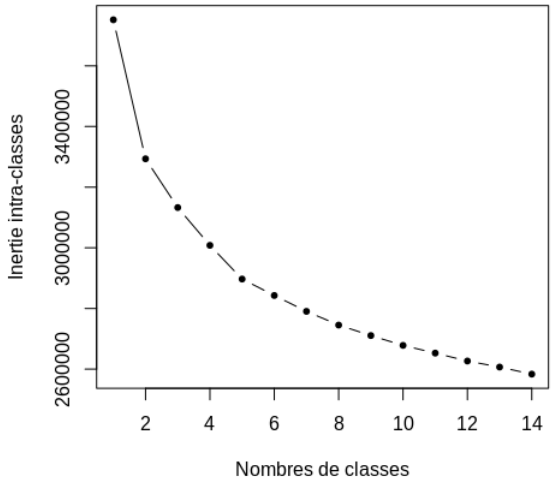

Figure 6 : Inertie intra-classes en fonction du nombre de classes

D'après ce graphique, le critère du coude nous conduit à choisir $k = 4$ classes, car l'inertie intra-classes ne diminue quasiment pas en passant à 5 classes. On présente maintenant les résultats de cette méthode :

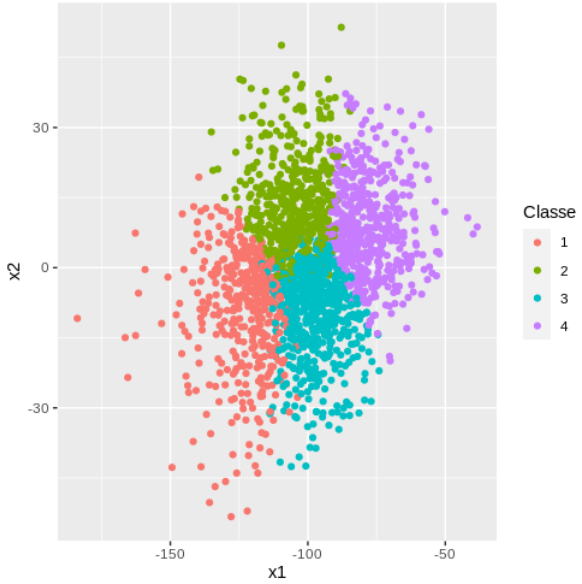

Figure 5 : Résultat de l'algorithme K-means sur le jeu de données

Avec 4 clusters, la méthode de K-means donne des bons résultats. En vérifiant le partitionnement trouvé par cette méthode, on retire en hasard quelque image réelle dans chaque cluster, on trouve que la méthode K-means regroupe les individus en 4 classes quasiment homogènes. Donc, notre base de données de 2004 images comporte 4 catégories d'animaux différentes.

Conclusion

Au cours de ce projet, nous avons eu l'occasion de travailler sur deux types du jeu de données. Le premier jeu de données on s'est intéressé à construire un modèle de classification permettant de prédire la classe d'un animal en fonction d'un certain nombre de caractéristiques qui influencent le plus l'affectation d'un animal dans chacune des classes, il s'agit d'une classification supervisée. On a trouvé un taux d'erreur moyen de 6%. Le deuxième jeu de données, on s'est intéressé à la classification de différentes espèces animales. Après avoir effectué les différentes tâches pour que notre base soit prête et de dimension petit à l'aide de l'ACP, nous avons décidé d'étudier trois méthodes. La première méthode de DBSCAN qui n'a pas donné un bon résultat, la deuxième méthode de CAH nous dit de choisir 5 classes, mais le partitionnement des individus n'a pas été convaincant, et la troisième méthode K-means à son tour a donné 4 classes avec des bons résultats du partitionnement. Donc en terme de performance, on voit que la méthode de K-means est la meilleure méthode pour ce jeu de données.

Annexe : Code R

```
1
2  #-------------------------------------------
3  # chargement des packages et fonctions utilises
4  #-------------------------------------------
5
6  library(nnet)
7  library(ggplot2)
8  library("FactoMineR")
9  library("factoextra")
10
11 #-------------------------------------------------------------------------------
12                                          # Exercice 1
13 #-------------------------------------------------------------------------------
14
15 #--------------------------------
16 # chargement du fichier de donnees
17 #--------------------------------
18
19 animal <- read.csv("/home/samnouni/Bureau/M2 ISN/methode daprentissage/projet/animaux
       _I.csv",sep=",", header=TRUE)
20
21 #--------------------------------
22 # Variable en factor
23 #--------------------------------
24
25 for (i in 2:ncol(animal)){animal[,i]=as.factor(animal[,i])}
26
27 #-----------------------------------
28 # Recodage de la variable class_type
29 #-----------------------------------
30
31 levels(animal$class_type)=c( "mammif re","oiseau","reptile"," poisson","amphibien","
       insecte","invert br  ")
32 animal=animal[,-1]
33
34 #-----------------------------------
35 # refaire une analyse descriptive
36 #-----------------------------------
37
38 str(animal)
39 summary(animal)
40 head(animal)
41
42 #-----------------------------------
43 # Calcul errreur
44 #-----------------------------------
```

```r

tx_er=function(pred, vrais)
{
  mc=table(pred, vrais)
  1-sum(diag(mc))/sum(mc)
}

#----------------------------------------
# Calcule Taux d'erreur du modle initial
#----------------------------------------

Erreurs=function(animal){
  N <- nrow(animal)
  ntrain <- floor(0.80*N) # partie entier
  ntest <- N - ntrain

  indices <- sample(1:N, ntrain, replace = FALSE)

  animal_train <- animal[indices,]
  animal_test <- animal[-indices,]
  log.animal= multinom(class_type~.,data=animal_train)
  predrf <- predict(log.animal, newdata=animal_test, type="class")
  te_rf <- tx_er(predrf, animal_test$class_type)
  return(te_rf)
}

#-----------------------------------------
# Graphique pour l'erreur en chaque itration
#-----------------------------------------

Erreurs_modele_initial=sapply(1:50 ,FUN=function(i){Erreurs(animal)})
plot(100*Erreurs_modele_initial, main="Le taux d'erreurs du modle initial",ylab="
    Taux d'erreur %",xlab="Itration")
mean(Erreurs_modele_initial)

#-----------------------------------------
# Choix d'un modle avec moins de variables
#-----------------------------------------

log.animal= multinom(class_type~.,data=animal_train)
summary(log.animal) # Pour afficher les resultats du modele
stepAIC(log.animal, trace = TRUE, data = animal_train) # pour choisir le modle le
    plus pertinent au snes du critre d'AIC

#-----------------------------------------
# Premier modle trouv
#-----------------------------------------

log.animal1= multinom(class_type~feathers+milk+backbone+breathes,data=animal_train)
predrf1 <- predict(log.animal1, newdata=animal_test, type="class")
te_rf1 <- tx_er(predrf1, animal_test$class_type)
te_rf1

#-----------------------------------------
# Deuxime modle trouv
#-----------------------------------------

log.animal2= multinom(class_type~hair+feathers+aquatic+breathes+venomous+tail,data=
    animal_train)
predrf2 <- predict(log.animal2, newdata=animal_test, type="class")
te_rf2 <- tx_er(predrf2, animal_test$class_type)
te_rf2
```

```
104
105 #-----------------------------------------
106 # ACM ( tude  de la liaison entre les variables)
107 #-----------------------------------------
108
109 animal=animal[,-18]
110 res.mca <- MCA (animal, graph = FALSE)
111 eig.val <- get_eigenvalue(res.mca) # Valeur propre
112 fviz_screeplot (res.mca, addlabels = TRUE, ylim = c (0, 45))
113 fviz_mca_var (res.mca, choice = "mca.cor",repel = TRUE,ggtheme = theme_minimal (),
        axes = c(3,2))
114 fviz_mca_var (res.mca,repel = TRUE,ggtheme = theme_minimal (),head=T,axes = c(1,2))
115
116 #-----------------------------------------
117 # Mod le final
118 #-----------------------------------------
119
120 log.animal_final1= multinom(class_type~eggs+aquatic+legs+toothed+catsize+predator,
        data=animal_train)
121 predrf_final1 <- predict(log.animal_final1,newdata=animal_test,type="class")
122 te_rf_final1 <- tx_er(predrf_final1,animal_test$class_type)
123 te_rf_final1
124 stepAIC(log.animal_final1, trace = TRUE, data = animal_train) # pour choisir le
        mod le le plus pertinent au snes du crit re d'AIC
125
126 #-----------------------------------------
127 #Fonction pour calculer l'erreur
128 #-----------------------------------------
129
130 Erreurs_modele_final=function(animal){
131   N <- nrow(animal)
132    ntrain <- floor(0.80*N) # partie entier
133    ntest <- N - ntrain
134
135    indices <- sample(1:N,ntrain,replace = FALSE)
136
137    animal_train <- animal[indices,]
138    animal_test <- animal[-indices,]
139    log.animal= multinom(class_type~eggs+aquatic+legs+toothed+catsize,data=animal_train
         )
140    predrf <- predict(log.animal,newdata=animal_test,type="class")
141    te_rf <- tx_er(predrf,animal_test$class_type)
142    return(te_rf)
143 }
144
145 Erreurs_modele_final=sapply(1:50 ,FUN=function(i){Erreurs_modele_final(animal)})
146 plot(100*Erreurs_modele_final,main="Le taux d'erreurs du mod le final",ylab="Taux d'
        erreur %",xlab="It ration")
147 mean(Erreurs_modele_final)
148
149
150 #### Partie 2 ####
151
152 data = data.frame()
153 # On r cup re les noms des fichiers
154 setwd("~/Bureau/M thodes_Dapprentissag/Projet")
155 animaux = list.files(path="animaux_II",full.names = F, recursive = F)
156 # choisir en hasard 2004 images
157 indices_image =sample(1:length(animaux),0.423*length(animaux))
158 names_animaux=animaux[indices_image]
159 nbimages=length(names_animaux)
160
```

```
for (i in 1:nbimages)
{
  files=paste0("animaux_II/",names_animaux[i])
  img=readJPEG(files)
  # transformation de chaque image en vecteur
  var1=as.vector(img[,,1])
  var2=as.vector(img[,,2])
  var3=as.vector(img[,,3])
  v=c(var1,var2,var3)
  v=as.data.frame(t(v))

  data =rbind(data,v )# on combine les vecteurs


}
# on renomme chaque ligne par nom de fichier
rownames(data)=names_animaux


# Application de l'acp sur notre base
acp = prcomp(data, scale. = F, center = F, retx = TRUE)

# on garde les axes dont les valeurs propres sont sup rieures     la moyenne
nbaxe = sum(acp$sdev**2 >= mean(acp$sdev**2))
pourcentage_inertie =sum(acp$sdev[1:nbaxe]**2)/sum(acp$sdev**2) # Pourcentage d'
    inertie
pourcentage_inertie


source("~/Bureau/M thodes_Dapprentissag/DBSCAN/color_utils.R")
# On teste la performance de l'Acp en fonction de nombre d'axes
img_reelle = array(unlist(data[1,]),dim=c(128,128,3))
display(img_reelle)

# les coordonn es des images dans la base form e par les 44 premier axes
data_acp = as.data.frame(acp$x[,1:nbaxe])
# Transformation     la base initiale
mat_pass = acp$rotation[,1:nbaxe] # Matrice de passage
images_reduite <- as.matrix(data_acp) %*% t(mat_pass)
img_reduite <- array(unlist(images_reduite[1,]),dim=c(128,128,3))
img_reduite[img_reduite<0] <- 0
img_reduite[img_reduite>1] <- 1
display(img_reduite)

# les coordonn es des images dans la base form e par les 200 premiers axes
nbaxe=200
data_acp = as.data.frame(acp$x[,1:nbaxe])
# Transformation     la base initiale
mat_pass = acp$rotation[,1:nbaxe] # Matrice de passage
images_reduite <- as.matrix(data_acp) %*% t(mat_pass)
img_reduite <- array(unlist(images_reduite[1,]),dim=c(128,128,3))
img_reduite[img_reduite<0] <- 0
img_reduite[img_reduite>1] <- 1
display(img_reduite)

### On va travailler sur les 200 premiers axes ####
data_acp=as.data.frame(acp$x[,1:nbaxe])
data_acp_plan1=data_acp[,1:2] # La projection de donn es sur le premier plan
colnames(data_acp_plan1)=c("x1","x2")
### DBSCAN ###

dKnn=kNNdist(data_acp,k=10)
```

```
222
223 plot(sort(dKnn,decreasing = T),type = 'l',xlim = c(0,500))
224 dev.off()
225 eps=54
226 dbing=dbscan(data_acp,eps=eps,minPts = 10)
227 clust=dbing$cluster
228 clust[clust==0]="bruit"
229 data_acp_plan1_DBSCAN=cbind(data_acp_plan1,clust)
230 ggplot(data = data_acp_plan1_DBSCAN,aes(x1,x2,color=as.factor(clust))) + geom_point()
    +labs(color = "Classe")
231
232
233 ### CAH ###
234 dist_d1 <- dist(data_acp,method = "euclidean")
235 hclust_1 <- hclust(dist_d1,method="ward.D")
236 plot(hclust_1)
237 # on garde 5 classes
238 clusters <- rect.hclust(hclust_1, k=5)
239 cah_clust=rep(1,nrow(data_acp_plan1))
240 cah_clust[clusters[[2]]]=2
241 cah_clust[clusters[[3]]]=3
242 cah_clust[clusters[[4]]]=4
243 cah_clust[clusters[[5]]]=5
244 data_acp_plan1_CAH=cbind(data_acp_plan1,cah_clust)
245 ggplot(data = data_acp_plan1_CAH,aes(x1,x2,color=as.factor(cah_clust))) + geom_point
    ()+labs(color = "Classe")
246
247 ### K-NN ###
248 sse <- numeric()
249 for(i in 1:14){
250   sse[i]=sum(kmeans(data_acp,centers=i,nstart = 2)$withinss)
251 }
252 plot(1:14,sse,pch=20,type="b",xlab = "Nombres de classes",ylab = "Inertie intra-
    classes")
253 # on choisit par exemple 4 ou 5
254 clust_KNN =kmeans(data_acp,centers=5, nstart = 2)
255 data_acp_plan1_KNN = cbind(data_acp_plan1,clust=clust_KNN$cluster)
256 ggplot(data = data_acp_plan1_KNN,aes(x1,x2,color=as.factor(clust))) + geom_point()+
    labs(color = "Classe")
```