Rapport du projet DATA

Études de données synthétiques et réelles avec R

Projet réalisé par GUARDIA Quentin, Master 1 Cybersécurité FI, quentin.guardia@etu.u-paris.fr

Table des matières

Introduction

L'objectif de ce projet est de comparer différents modèles prédictifs. Nous avons décidé d'implémenter ces modèles en R. En voici la liste : classifieur bayésien naïf, randomForest, CART, régression logistique, KNN, LDA, QDA, SVM linéaire et SVM non-linéaires (polynomial, base radiale, simoïde).

Nous disposons de différents jeux de données. D'abord, des données synthétiques, à savoir Aggregation.txt, flame.txt et spiral.txt. Ces données sont analysées par le programme contenu dans synth.R. Puis, les données réelles, VisaPremier.txt et creditcard.csv, qui sont respectivement analysées par visa.R et fraude.R.

À l'issue des analyses, nous avons pu avoir un aperçu de l'efficacité des différentes méthodes. Ce qui nous a permis de formuler quelques conjectures, disponibles en conclusion.

En annexe se trouvent les codes que nous avons utilisés. Toutes les images du projet sont obtenues avec RStudio.

Données synthétiques (synth.R)

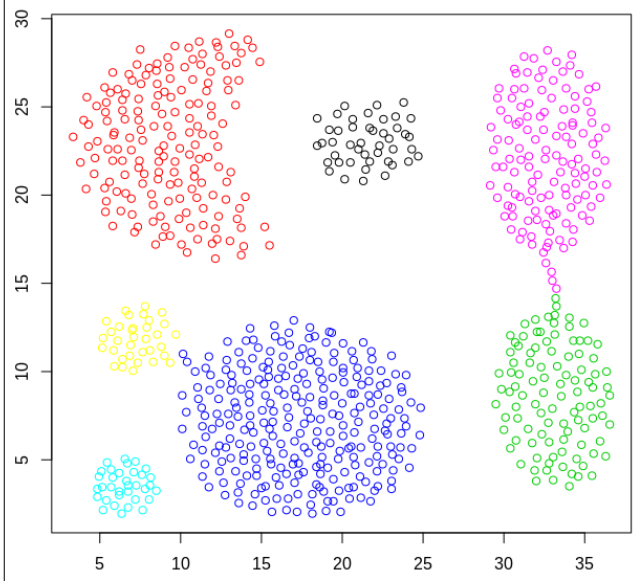
Étude préliminaire

On réalise une étude préliminaire des fichiers Aggregation.txt, flame.txt et spiral.txt. Grâce à la méthode summary(), on peut voir que les trois fichiers de données se ressemblent. Ils ont chacun 3 colonnes. Voici les détails concernant chacun des fichiers :

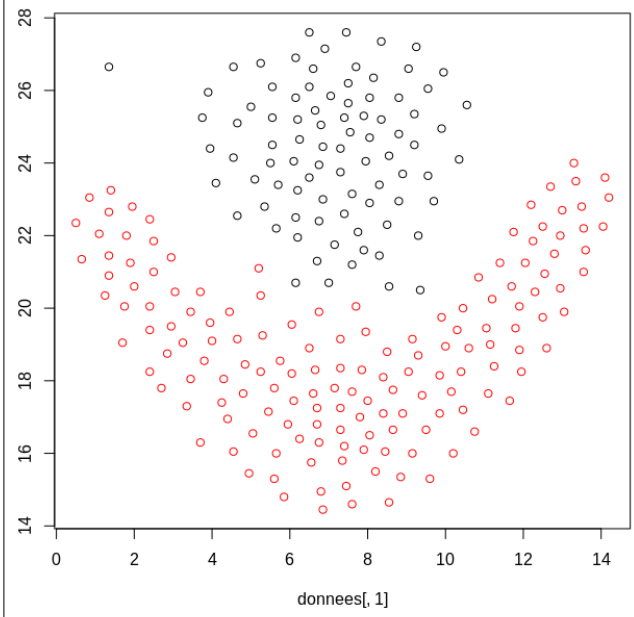
	Colonne 1	Colonne 2	Colonne 3
Aggregation.txt (788 entrées)	Min : 3,35 Médiane : 18,30 Max : 36,55	Min : 1,95 Médiane : 11,70 Max : 29,15	Entiers de 1 à 7
Flame.txt (96 entrées)	Min : 0,50 Médiane : 7,35 Max : 14,20	Min : 14,45 Médiane : 20,90 Max : 27,60	Entiers de 1 à 2
Spiral.txt (312 entrées)	Min: 3,00 Médiane : 18,20 Max : 31,65	Min : 2,90 Médiane : 16,05 Max : 31,65	Entiers de 1 à 3

Lorsque l'on représente la colonne 1 en fonction de la colonne 2, en colorant selon la colonne 3, on obtient les graphes ci-dessous, avec donnees[,1] en abscisse en donnees[,2] en ordonnée :

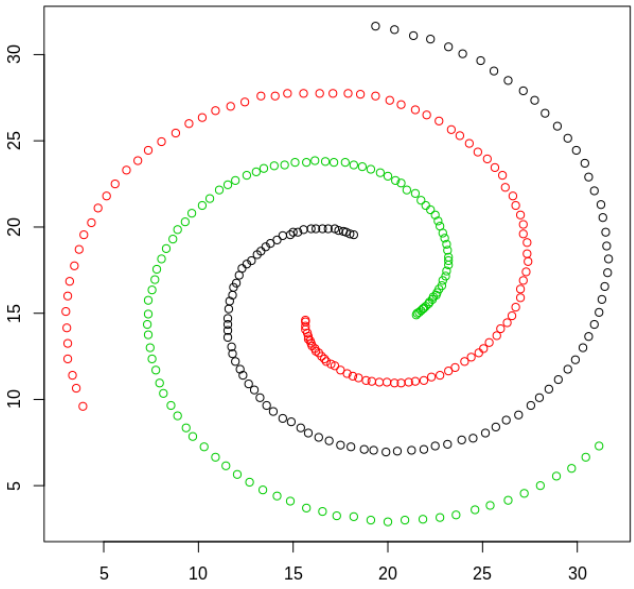
Aggregation.txt



flame.txt



spiral.txt



On va donc admettre que les variables explicatives seront celles des deux premières colonnes, et celles à expliquer celles de la troisième colonne. Le but serait donc de prédire la couleur d'un point en fonction de son abscisse et de son ordonnée.

Méthodes utilisées

On a réalise un échantillonnage. L'échantillon d'apprentissage contient 80 % des données, celui de test 20 %. Tout cela de manière aléatoire bien sûr, grâce à la méthode sample(). Nous avons sélectionné toutes les données, toujours dans le but d'exprimer la dernière colonne en fonction des deux autres.

On a répété chaque méthode 100 fois chacune, sauf pour la régression logistique, LDA, QDA et SVM linéaire et non-linéaires. On a en effet observé une constance dans les résultats. Au sujet de la méthode des KNN, nous avons pris soin de sélectionner le meilleur k à chaque fois. Soit k=1 ici.

Grâce à la fonction confusionMatrix() de la bibliothèque Caret, on a pu déterminer l'accuracy de chacune des méthodes, afin de la stocker dans un tableau. Voici un exemple avec le classifieur bayésien naïf :

```
modele <- naiveBayes(as.factor(apprentissage[,3])~., data=apprentissage[,-3])
prediction <- predict(modele, test)
resultats <- confusionMatrix(table(prediction,test[,3]))
accuracy[k] <- resultats$overall['Accuracy']
```

Cependant, plus de deux valeurs à prédire pouvaient être possibles pour Aggregation.txt et spiral.txt. Comme la bibliothèque Caret calcule la F-Measure à partir d'une matrice de confusion à deux variables uniquement, il a fallu calculer nous-même la F-Measure. Pour cela, nous avons calculé la moyenne pondérée de la F-Measure de chaque variable, avec le code suivant :

```
cm <- as.matrix(resultats)
n = sum(cm)
nc = nrow(cm)
rowsums = apply(cm, 1, sum)
colsums = apply(cm, 2, sum)
diag = diag(cm)
precision = diag / colsums
recall = diag / rowsums
fmeasure_multiclasses = 2 * precision * recall / (precision + recall)
fmeasure[k] <- sum(fmeasure_multiclasses*colsums)/sum(colsums)
```

On a ensuite réalisé les moyennes d'accuracy et de F-Measure par modèle, que l'on a stockées dans les tableaux de la partie suivante.

On pourrait optimiser le seuil de la régression logistique si le nombre de variables prédictibles était constant. Ce n'est pas le cas donc on a arbitrairement choisi d'arrondir. Cela ne fonctionnerait bien que si la variable à prédire ne pouvait prendre que deux valeurs.

Il y a une deuxième faiblesse : on n'a pas étudié quelle répartition des échantillons d'apprentissage et de test est optimale. On a mis aléatoirement 80 % des données dans celui d'apprentissage, et 20 % dans celui de test. Pour ne pas alourdir le code et comme c'est une répartition communément utilisée, on a choisi de garder celle-ci.

Comparaison entre les méthodes

Grâce à des tableaux on compare les méthodes et bilan :

Aggregation.txt	Accuracy	Weighted F-Measure
Classifieur Bayésien naïf	0.9981435	0.9981561
randomForest	0.9992405	0.9992361

CART	1	1
KNN	0.9988186	0.9988236
Régression logistique	0.3456163	NaN (divison par 0)
LDA	0.9923761	0.9922512
QDA	0.9987294	0.9987287
SVM Linéaire	0.9974587	0.9974674
SVM polynomial	0.9898348	0.9897977
SVM base radiale	0.9974587	0.9974674
SVM sigmoïde	0.9008895	NaN (division par 0)

flame.txt	Accuracy	Weighted F-Measure
Classifieur Bayésien naïf	0.9595833	0.9588759
randomForest	0.9883333	0.9883203
CART	0.9792887	0.9793155
KNN	0.99375	0.9937733
Régression logistique	0.8786611	0.8793671
LDA	0.8786611	0.8779551
QDA	0.9623431	0.9628447
SVM Linéaire	0.8828452	0.8828452
SVM polynomial	0.8661088	0.857145
SVM base radiale	0.9958159	0.9958105
SVM sigmoïde	0.8284519	0.8304093

spiral.txt	Accuracy	Weighted F-Measure
Classifieur Bayésien naïf	0.3178723	0.316688
randomForest	0.9895238	0.9869118
CART	0.977717	0.977731
KNN	0.9994681	0.9994686
Régression logistique	0.3376206	NaN (division par 0)
LDA	0.3247588	0.3247795
QDA	0.340836	0.3408673
SVM Linéaire	0.318328	0.3183417
SVM polynomial	0.437299	0.424945
SVM base radiale	0.9839228	0.9839722
SVM sigmoïde	0.1221865	NaN (division par 0)

Tous jeux confondus, les meilleurs modèles prédictifs semblent être fournis par randomForest, CART, KNN et SVM base radiale. De plus, on peut voir que les méthodes de prédiction sont très précises pour les données d'Aggregation.txt et flame.txt, mais pas pour spiral.txt, qui a probablement une répartition des valeurs non-optimale pour ce genre d'analyse, surtout le SVM à noyau sigmoïde. On peut suspecter un overfit, surtout au vu de l'accuracy et de la F-Measure de la méthode CART, qui atteignent 100 % sur le jeu Aggregation.txt

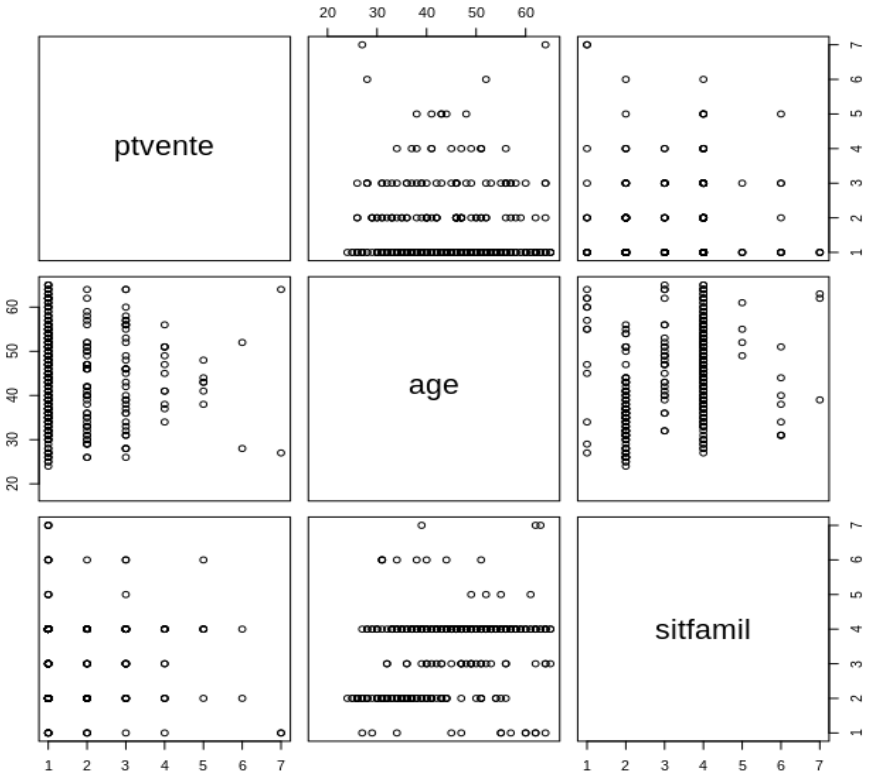
Données VisaPremier.txt (visa.R)

Étude préliminaire

On a pu observer que la variable a prédire (cartevpr) est un entier compris entre 0 et 1. Si la variable vaut 1, alors le client bénéficie d'une carte Visa Premier. 714 clients du jeu n'ont pas de carte Visa Premier (0) pour 359 qui en ont une (1). Il y a donc 1073 entrées.

```
length(which(donnees$cartevpr == 0))
length(which(donnees$cartevpr == 1))
```

Les variables prédictives sont souvent des entiers et apportent des informations sur le client et son comportement. L'objectif est donc ici de prédire si un client bénéficie d'une carte Visa Premier en fonction de ses informations.
Voici ci-dessous un graphique représentant les trois premières variables prédictives en fonction de cartevpr. On peut voir qu'il n'aide pas autant que pour les données synthétiques.



On a commencé par supprimer les variables non-prédictives, soit :

- matricul, qui est indépendante de la variable à prédire.
- departem dont certaines valeurs étaient vides.
- sexe, car la variable sexer contient déjà la valeur de sexe binarisée.
- nbimpaye car constant, uniquement 0 pour valeur.
- mteparte, nbbon, mtbon et nbparte car quasi constants, seule 1 à 3 variables valent 1 sur tout le dataset composé de 0.

```
donnees <- donnees[, !(names(donnees) %in% c("matricul","cartevp","sexe",
"departem", "nbimpaye", "mteparte", "nbbon", "mtbon", "nbeparte"))]
```

Puis, on a dans un premier temps essayé de numériser les variables littérales, soit sitfamil, csp et codeqlt.

```
  levels(donnees$sitfamil) <- c(1:length(levels(as.factor(donnees$sitfamil))))
  levels(donnees$csp) <- c(1:length(levels(as.factor(donnees$csp))))
  levels(donnees$codeqlt) <- c(1:length(levels(as.factor(donnees$codeqlt))))
```

Toutefois, en convertissant de la sorte chaque valeur possible -les levels-, certains membres du groupe ont rencontré des problèmes par la suite, car les anciens levels littéraux semblent être sauvegardés par défaut. Malgré différentes techniques essayées, la solution pour n'avoir aucune erreur dans ce cas reste de supprimer ces trois variables du jeu de données, en leur affectant NULL. Il faut alors décommenter la ligne 23 de visa.R.

Enfin, pour que chaque méthode puisse être comparable, il faut que les mêmes données soient disponibles pour chacune d'entre elle. Or, randomForest ne supporte que 52 levels au maximum. Par conséquent, on a supprimé les variables ne correspondant pas à ce critère grâce à la boucle suivante :

```
  i=1
  while(i < ncol(donnees)){
      if(length(levels(as.factor(donnees[,i])))>52){
          donnees<-donnees[,-c(i)]
      }else{
          i=i+1
    }
}
```

Ce qui laisse 17 variables prédictives au total, ou 14 si on supprime les variables littérales.

Méthodes utilisées

La même boucle que dans la partie précédente a été mise en place, et elle permet toujours de calculer une moyenne d'accuracy et de F-Measure si nécessaire. Sur 10 échantillons différents cette fois, à cause de la faible efficacité de nos machines. Ici, il n'y a pas de F-Measure pondérée car la variable à prédire vaut 0 ou 1. D'où le paramètre positive="1" dans la méthode confusionMatrix() de Caret, car sinon la valeur que l'on vérifie par défaut est la première, soit 0. Voici l'instruction pour récupérer la F-Measure de la matrice de confusion, aussi appelée score F1 :

```
  fmeasure[i] <- resultats$byClass['F1']
```

On a implémenté des boucles pour trouver le seuil optimal pour les méthodes glm et svm, car les prédictions ne sont pas binarisées dans notre code. Ainsi les boucles s'incrémentent de 0,30 à 0,70 avec un pas de 0,05. Voici un exemple de code pour le SVM linéaire :

```
modele <- svm(donnees$cartevpr ~ ., data=donnees, kernel="linear")
fiabilite <- vector(mode="integer", length=9)
j <- 1

for (seuil in seq(0.30, 0.70, by=0.05)) {
     prediction = ifelse(predict(modele,donnees) >=seuil, 1, 0)
     resultats <- confusionMatrix(table(prediction,donnees$cartevpr))
     fiabilite[j] <- resultats$overall['Accuracy'] + resultats$byClass['F1']
     j <- j+1
}

seuil1=which.max(fiabilite)
seuil2=0.30+0.05*(seuil1-1)
prediction = ifelse(predict(modele,donnees) >=seuil2, 1, 0)
```

Comparaison entre les méthodes

VisaPremier.txt	Accuracy	F-Measure
Classifieur Bayésien naïf	0.7621118	0.5498351
randomForest	0.8487578	0.7543894
CART	0.8676608	0.801676
KNN	0.6931677	0.4133681
Régression logistique	0.8229264	0.875
LDA	0.8639329	0.7820896
QDA	0.7576887	0.5578231
SVM Linéaire	0.8564772	0.7855153
SVM polynomial	0.7847158	0.563327
SVM base radiale	0.8946878	0.8437068
SVM sigmoïde	0.8126747	0.6912442

Ici, la méthode la plus adaptée semble être la méthode CART, avec la fonction rpart(). Ces résultats correspondent au cas où les trois variables littérales ont été converties puis utilisées avec succès. Sans quoi, l'accuracy et la F-Measure ont une baisse signifiante, allant d'un centième jusqu'au presque deux dixièmes selon les méthodes.

Données creditcard.csv (fraude.R)

Étude préliminaire

La quantité de données est colossale : 284 807 entrées pour 31 variables. Parmi ces 31 variables, Class est à prédire. Il s'agit d'une fraude lorsque la variable vaut 1. Elles sont extrêmement faibles, il n'y a que 492 fraudes sur tout le jeu de données, soit moins de 0,2 %. On enlève la variable Time. Le reste des variables n'ont pas de nom explicite et sont binarisées. Probablement par sécurité s'il s'agit de données réelles. Ainsi, on se retrouve avec 29 variables prédictives. L'objectif est clairement de prédire s'il y a fraude ou non en fonction des données prédictives, bien qu'on ne connaisse pas leur nature concrète.

Méthodes utilisées

Le jeu est très lourd et très déséquilibré. Donc quelques changements s'imposent. Premièrement, par faiblesse matérielle, on prend un échantillon aléatoire représentant un dixième du jeu de données :

```
  indice <- sample(1:nrow(donnees),0.1*nrow(donnees))
  donnees <- donnees[ indice,]
```

Avec une machine assez puissante, on peut bien entendu se permettre d'enlever ces deux lignes.

Ensuite, on ne va pas faire de boucle pour faire une moyenne des accuracy et F-Measure, le coût est bien trop lourd. On applique donc chaque méthode une à une, et une seule fois. Ensuite, on ne va pas utiliser de matrice de confusion pour calculer la précision des prédictions par F-Measure ou Accuracy. Le jeu est beaucoup trop déséquilibré. On va plutôt chercher à calculer l'aire sous la courbe de la précision et du rappel (AUCPRC), grâce à la méthode pr.curve(). Par exemple, pour le classifieur Bayésien naïf, on a calculé l'AUCPRC de la manière suivante :

```
modele<-naiveBayes(as.factor(apprentissage$Class)~.,data=apprentissage_feature)
prediction <- predict(modele, test)
pr <- pr.curve( prediction, test$Class, curve = TRUE )
pr$auc.integral
```

On utilise les mêmes méthodes que pour les jeux de données précédents, sans avoir de seuil à préciser pour glm car la binarisation est automatique.

Il faut donc faire attention, car la petite taille de l'échantillon par rapport au jeu étudié et sa seule itération peuvent empêcher de généraliser les résultats obtenus.

Comparaison entre les méthodes

creditcard.csv	Aire sous la courbe
Classifieur Bayésien naïf	0.998054
randomForest	0.9979129
CART	0.9993727
KNN	0.9979099
Régression logistique	0.9870125
LDA	0.9987827
QDA	0.9987876
SVM Linéaire	0.9565932
SVM polynomial	0.8515678
SVM base radiale	0.7434113
SVM sigmoïde	0.7098996

Ici, le meilleur modèle de prédiction est celui offert par CART, une fois de plus. Et la méthode SVM avec sigmoid pour noyau semble être la moins efficace.

Conclusion

Au cours du projet, nous avons pu tester différents modèles de prédiction sur plusieurs jeux de données. Nous avons comparé l'efficacité des méthodes en calculant l'accuracy, la F-Measure ; voire l'aire sous la courbe Precision-Recall pour les données déséquilibrées. Ainsi, nous avons constaté de nombreuses choses.

Déjà, CART semble faire partie des méthodes les plus fiables dans chacune de nos études de cas. Cependant, il nous faudrait plus d'essais pour affirmer qu'il s'agit d'une des meilleures méthodes dans l'absolu. Malgré cela, il ne semble pas y avoir de méthode inadaptée à tous les cas.

Ensuite, on suppose que la qualité de prédiction dépend : du nombre d'entrées, du nombre de variables prédictives, de la dispersion des variables et de la méthode utilisée. La dispersion pouvant être liée la disparité des variables, leurs ratios et leurs levels. Cela pourrait expliquer pourquoi spiral.txt, qui a une dispersion particulière des variables prédictives, obtient de moins bonnes prédictions. Ou pourquoi Aggregation.txt qui a plus d'entrées, a de meilleurs résultats. L'analyse de VisaPremier.txt montre l'importance de la suppression de peu de valeurs prédictives.

Enfin, il faut se méfier car notre échantillonnage semble rendre les méthodes de prédiction utilisées sujettes à l'overfitting. On pourrait éventuellement trouver un moyen plus fiable de valider la prédiction.

Annexes

Données synthétiques

```
library(caret) #confusionMatrix
library(e1071) #naiveBayes et svm
library(class) #knn
library(MASS) #lda et qda
library(rpart) #rpart pour CART
library(randomForest) #randomForest
#en cas d'absence du package, écrire: install.packages("nom_du_paquet")

donnees <- read.delim("spiral.txt") #Valable pour les données synthétiques:
flame, spiral, aggregation
attach(donnees)


#Aperçu textuel:
summary(donnees)
apply(donnees, 2, sd) #ecart-type


#Aperçu graphique:
boxplot(donnees[,1],main="Colomne 1")
boxplot(donnees[,2],main="Colomne 2")
boxplot(donnees[,3],main="Colomne 3")
plot(donnees[,1], donnees[,2], col=donnees[,3]) #Intéressant, il faut prédire la
3ème colonne


#Création du tableau bilan
stat<-matrix(list(), nrow=11, ncol=2)
colnames(stat) <- c("accuracy","F-measure")
rownames(stat) <- c("Bayésien naïf", "randomForest", "CART", "KNN", "Régression
logistique", "LDA", "QDA", "SVM linéaire", "SVM polynomial", "SVM base radiale",
"SVM sigmoide")


#On choisit préalablement le meilleur k pour knn
iterations= 50
indice <- sample(1:nrow(donnees),0.8*nrow(donnees))
apprentissage <- donnees[ indice,]
test <- donnees[-indice,]
erreur <- vector(mode="integer", length=iterations)
for(k1 in 1:iterations){ #On choisit le meilleur k
        prediction <- knn(apprentissage[,-3],test[,-3],cl=apprentissage[,3],k1)
        erreur[k1] <- sum(test[,3] != prediction)/length(donnees[,3])
}
k1=which.min(erreur)


#On fait 100 itérations pour chaque méthode de prédiction pour calculer accuracy
et F-measure moyen
n=407
accuracy <- vector(length = n)
fmeasure <- vector(length = n)

print("On attend quelques secondes: les résultats chargent")
for (k in 1:407) {
```

```
        #Échantillonage aléatoire
        set.seed(3*k)
        indice <- sample(1:nrow(donnees),0.7*nrow(donnees))
        apprentissage <- donnees[ indice,]
        test <- donnees[-indice,]

        if(k<201){
                if(k <=100){ #Classifieur Bayésien naïf
                        modele <- naiveBayes(as.factor(apprentissage[,3])~.,
data=apprentissage[,-3])
                }else if(k>100 ){#randomForest
                        modele <- randomForest(apprentissage[,1:2],
as.factor(apprentissage[,3]))
                }
                prediction <- predict(modele, test)
                resultats <- confusionMatrix(table(prediction,test[,3]))
        }else if(k>200 & k < 301){#CART
                        modele <- rpart(as.factor(donnees[,3]) ~
donnees[,1]+donnees[,2], data = donnees)
                        prediction <- predict(modele, donnees, type="class")
                        resultats <-
confusionMatrix(table(donnees[,3],prediction))
        }else if(k>300 & k < 401){ #knn
                prediction <- knn(apprentissage[,-3],test[,-
3],cl=apprentissage[,3],k1)
                resultats <- confusionMatrix(table(prediction,test[,3]))
        }else if(k==401){ #Régression logistique
                modele <- glm(donnees[,3]~.,data=donnees[,-3])
                prediction <- round(predict(modele, as.data.frame(donnees[,3])))
#Approximatif
                prediction <- factor(as.factor(prediction),
levels=levels(as.factor(donnees[,3])))
                resultats <- confusionMatrix(table(prediction,donnees[,3]))
        }else if(k>401 & k < 404){
                if(k==402){#LDA
                        modele <- lda(donnees[,3] ~ ., data = donnees[-3])
                }else if(k==403){#QDA
                        modele <- qda(donnees[,3] ~ ., data = donnees[-3])
                }
                prediction <- predict(modele, donnees[-3])
                resultats <-
confusionMatrix(table(donnees[,3],prediction$class))
        }else if(k>403){
                if(k==404){ #SVM linéaire
                        modele <- svm(donnees[,-
3],as.factor(as.character(donnees[,3])),kernel="linear")
                }else if(k==405){ #SVM polynomial
                        modele <- svm(donnees[,-
3],as.factor(as.character(donnees[,3])),kernel="polynomial")
                }else if(k==406){ #SVM base radiale
                        modele <- svm(donnees[,-
3],as.factor(as.character(donnees[,3])),kernel="radial")
                }else if(k==407){ #SVM sigmoide
                        modele <- svm(donnees[,-
3],as.factor(as.character(donnees[,3])),kernel="sigmoid")
                }
                prediction <- predict(modele, donnees[,-3])
                resultats <- confusionMatrix(table(prediction,donnees[,3]))
        }

        #Calculs: accuracy et F-Measure
        cm <- as.matrix(resultats)
        n = sum(cm)
        nc = nrow(cm)
```

```
        rowsums = apply(cm, 1, sum)
        colsums = apply(cm, 2, sum)
        diag = diag(cm)
        precision = diag / colsums
        recall = diag / rowsums
        fmeasure_multiclasses = 2 * precision * recall / (precision + recall)
        if(k<401){
                accuracy[k] <- resultats$overall['Accuracy']
                fmeasure[k] <- sum(fmeasure_multiclasses*colsums)/sum(colsums)
        }else{
                stat[k-396,1] <- resultats$overall['Accuracy']
                stat[k-396,2] <- sum(fmeasure_multiclasses*colsums)/sum(colsums)
        }
}


#Remplissage du reste du tableau bilan
for(i in 0:3){
        inf <- i*100+1
        sup <- i*100+100
        stat[i+1,1]<-mean(accuracy[inf:sup])
        stat[i+1,2]<-mean(fmeasure[inf:sup])
}
stat
```

Données réelles : VisaPremier.txt

```
library(caret) #confusionMatrix
library(e1071) #naiveBayes et svm
library(class) #knn
library(MASS) #lda et qda
library(rpart) #rpart pour CART
library(randomForest) #randomForest
#en cas d'absence du package, écrire: install.packages("nom_du_paquet")

donnees <- read.delim("VisaPremier.txt")

#On supprime les données non ou peu prédictives
donnees <- donnees[, !(names(donnees) %in% c("matricul","cartevp","sexe",
"departem", "nbimpaye", "mteparte", "nbbon", "mtbon", "nbeparte"))]
attach(donnees)


#Numérisation des données
levels(donnees$sitfamil) <- c(1:length(levels(as.factor(donnees$sitfamil))))
levels(donnees$csp) <- c(1:length(levels(as.factor(donnees$csp))))
levels(donnees$codeqlt) <- c(1:length(levels(as.factor(donnees$codeqlt))))

#ATTENTION
#En cas d'erreur de NA par la suite, relancer le programme en décommentant la
ligne dessous
#donnees <- donnees[, !(names(donnees) %in% c("sitfamil","csp","codeqlt"))]

#Aperçu textuel:
summary(donnees)


#Contenu de la colonne cartevpr
length(which(donnees$cartevpr == 0))
length(which(donnees$cartevpr == 1))
```

```r
#Aperçu graphique en exprimant les trois premières données en fonction de la
données à prédire (cartevpr)
pairs(donnees[,1:3], col=as.numeric(cartevpr))


#RandomForest ne supporte que 53 levels. On enlève colonnes avec plus de levels
i=1
while(i < ncol(donnees)){
        if(length(levels(as.factor(donnees[,i])))>52){
                donnees<-donnees[,-c(i)]
        }else{
                i=i+1
        }
}

#Tableau récapitulatif
stat<-matrix(list(), nrow=11, ncol=2)
colnames(stat) <- c("accuracy","F-measure")
rownames(stat) <- c("Bayésien naïf", "randomForest", "CART", "KNN", "Régression
logistique", "LDA", "QDA", "SVM linéaire", "SVM polynomial", "SVM base radiale",
"SVM sigmoide")

#On fait 10 itérations pour chaque méthode de prédiction pour calculer accuracy
et F-measure moyen
n=47
accuracy <- vector(length = n)
fmeasure <- vector(length = n)


#On choisit préalablement le meilleur k pour knn
iterations= 50
set.seed(3)
indice <- sample(1:nrow(donnees),0.7*nrow(donnees))
apprentissage <- donnees[ indice,]
test <- donnees[-indice,]
apprentissage_feature <- subset(apprentissage,select=-cartevpr)
donnees_feature <- subset(donnees,select=-cartevpr)
test_feature <- subset(test,select=-cartevpr)
erreur <- vector(mode="integer", length=iterations)
for(k1 in 1:iterations){ #On choisit le meilleur k
        prediction <-
knn(apprentissage_feature,test_feature,cl=apprentissage$cartevpr,k1)
        erreur[k1] <- sum(test$cartevpr != prediction)/length(donnees$cartevpr)
}
k1=which.min(erreur)

#La boucle
for(i in 1:47){
        set.seed(3*i)
        indice <- sample(1:nrow(donnees),0.7*nrow(donnees))
        apprentissage <- donnees[ indice,]
        test <- donnees[-indice,]
        apprentissage_feature <- subset(apprentissage,select=-cartevpr)
        donnees_feature <- subset(donnees,select=-cartevpr)
        test_feature <- subset(test,select=-cartevpr)
        if(i < 21){#Bayes et RF
                if(i<11){
                        modele <-
naiveBayes(as.factor(apprentissage$cartevpr)~., data=apprentissage_feature)
                }else{
                        modele <- randomForest(apprentissage_feature,
as.factor(apprentissage$cartevpr))
                }
                prediction <- predict(modele, test)
```

```
                resultats <- confusionMatrix(table(prediction,test$cartevpr),
positive="1")
        }else if(i > 20 & i < 31){ #CART
                modele <- rpart(factor(donnees$cartevpr) ~ ., data =
donnees_feature)
                prediction <- predict(modele, donnees, type="class")
                resultats <- confusionMatrix(table(donnees$cartevpr,prediction),
positive="1")
        }else if(i > 30 & i < 41){ #KNN
                prediction <-
knn(apprentissage_feature,test_feature,cl=apprentissage$cartevpr,k1)
                resultats <- confusionMatrix(table(prediction,test$cartevpr),
positive="1")
        }else if(i==41){ #Regression logitique

                #Sélection du meilleur seuil
                fiabilite <- vector(mode="integer", length=9)
                j <- 1
                for (seuil in seq(0.30, 0.70, by=0.05)) {
                        modele <- glm(donnees$cartevpr~.,data=donnees[,!
colnames(donnees) %in% c("sitfamil","csp","codeqlt")])
                        prediction <- ifelse(predict(modele,
as.data.frame(donnees$cartevpr)) >= seuil, 1, 0)
                        resultats <-
confusionMatrix(table(prediction,donnees$cartevpr))
                        fiabilite[j] <- resultats$overall['Accuracy'] +
resultats$byClass['F1']
                        j <- j+1
                }
                seuil1=which.max(fiabilite)
                seuil2=0.30+0.05*(seuil1-1)
                modele <- glm(donnees$cartevpr~.,data=donnees[,!
colnames(donnees) %in% c("sitfamil","csp","codeqlt")])
                prediction <- ifelse(predict(modele,
as.data.frame(donnees$cartevpr)) >= seuil2, 1, 0)
                resultats <- confusionMatrix(table(prediction,donnees$cartevpr))
        }else if(i>41 & i < 44){
                if(i==42){#LDA
                        modele <- lda(donnees$cartevpr ~ ., data =
donnees_feature[,-6])
                }else if(i==43){#QDA
                        modele <- qda(donnees$cartevpr ~ ., data =
donnees_feature[,-4])
                }
                prediction <- predict(modele, donnees_feature)
                resultats <-
confusionMatrix(table(donnees$cartevpr,prediction$class), positive="1")
        }else if(i>43){
                if(i==44){ #SVM linéaire
                        modele <- svm(donnees$cartevpr ~ ., data=donnees,
kernel="linear")
                }else if(i==45){ #SVM polynomial
                        modele <- svm(donnees$cartevpr ~ ., data=donnees,
kernel="polynomial")
                }else if(i==46){ #SVM base radiale
                        modele <- svm(donnees$cartevpr ~ ., data=donnees,
kernel="radial")
                }else if(i==47){ #SVM sigmoide
                        modele <- svm(donnees$cartevpr ~ ., data=donnees,
kernel="sigmoid")
                }
                #Selection du seuil
                fiabilite <- vector(mode="integer", length=9)
                j <- 1
```

```
                for (seuil in seq(0.30, 0.70, by=0.05)) {
                        prediction = ifelse(predict(modele,donnees) >=seuil, 1,
0)
                        resultats <-
confusionMatrix(table(prediction,donnees$cartevpr))
                        fiabilite[j] <- resultats$overall['Accuracy'] +
resultats$byClass['F1']
                        j <- j+1
                }
                seuil1=which.max(fiabilite)
                seuil2=0.30+0.05*(seuil1-1)
                prediction = ifelse(predict(modele,donnees) >=seuil2, 1, 0)
                resultats <- confusionMatrix(table(prediction,donnees$cartevpr),
positive="1")
        }

        #Accuracy et F-Measure
        if(i<41){
                accuracy[i] <- resultats$overall['Accuracy']
                fmeasure[i] <- resultats$byClass['F1']
        }else{
                stat[i-36,1] <- resultats$overall['Accuracy']
                stat[i-36,2] <- resultats$byClass['F1']
        }
}

#Remplissage du reste du tableau bilan
for(i in 0:3){
        inf <- i*10+1
        sup <- i*10+10
        stat[i+1,1]<-mean(accuracy[inf:sup])
        stat[i+1,2]<-mean(fmeasure[inf:sup])
}
stat
```

Données réelles : creditcard.csv

```
library(e1071) #naiveBayes et svm
library(class) #knn
library(MASS) #lda et qda
library(rpart) #rpart pour CART
library(randomForest) #randomForest
library(ROCR) #PRCurve
library(PRROC) #PRCurve
#en cas d'absence du package, écrire: install.packages("nom_du_paquet")

donnees <- read.csv(file = "creditcard.csv", head = TRUE)

#On enlève Time
donnees$Time <- NULL


#Contenu de la colonne Class
length(which(donnees$Class == 0))
length(which(donnees$Class == 1))


#Ces deux lignes sont à enlever si la machine le permet
indice <- sample(1:nrow(donnees),0.1*nrow(donnees))
donnees <- donnees[ indice,]

#Aperçu textuel:
```

```r
summary(donnees)


#Aperçu graphique en exprimant les trois premières données en fonction de la
données à prédire (cartevp)
pairs(donnees[,1:3], col=as.numeric(donnees$Class))


#Création du tableau bilan
stat<-matrix(list(), nrow=11, ncol=1)
colnames(stat) <- c("AUC")
rownames(stat) <- c("Bayésien naïf", "randomForest", "CART", "KNN" "Régression
logistique", "LDA", "QDA", "SVM linéaire", "SVM polynomial", "SVM base radiale",
"SVM sigmoide")


#Échantillonage
indice <- sample(1:nrow(donnees),0.7*nrow(donnees))
apprentissage <- donnees[ indice,]
test <- donnees[-indice,]
apprentissage_feature <- subset(apprentissage,select=-Class)
donnees_feature <- subset(donnees,select=-Class)
test_feature <- subset(test,select=-Class)

#Boucle affectant le résultat de chaque méthode au tableau stat
for(i in 1:11){
        if(i==1){ #Classifieur Bayésien naïf
                modele <- naiveBayes(as.factor(apprentissage$Class)~.,
data=apprentissage_feature)
                prediction <- predict(modele, test)
        }else if(i==2){#RF
                modele <- randomForest(apprentissage_feature,
as.factor(apprentissage$Class))
                prediction <- predict(modele, test)
        }else if(i==3){#CART
                modele <- rpart(factor(donnees$Class) ~ ., data =
donnees_feature)
                prediction <- predict(modele, donnees, type="class")
        }else if(i==4){#KNN
                iterations= 10
                erreur <- vector(mode="integer", length=iterations)
                for(k1 in 1:iterations){ #On choisit le meilleur k
                        prediction <-
knn(apprentissage_feature,test_feature,cl=apprentissage$Class,k1)
                        erreur[k1] <- sum(test$Class !=
prediction)/length(donnees$Class)
                }
                k1=which.min(erreur)
                prediction <-
knn(apprentissage_feature,test_feature,cl=apprentissage$Class,k1)
        }else if(i==5){#Régression logistique
                modele = glm(formula = apprentissage$Class ~.,family =
binomial,data = apprentissage_feature)
                prediction = predict(modele,type = 'response',newdata =
test_feature)
        }else if(i==6){#LDA
                modele <- lda(donnees$Class ~ ., data = donnees_feature)
                prediction <- predict(modele, donnees_feature)
                pr <- pr.curve( prediction$class, donnees$Class, curve = TRUE );
        }else if(i==7){#QDA
                modele <- qda(donnees$Class ~ ., data = donnees_feature)
                prediction <- predict(modele, donnees_feature)
                pr <- pr.curve( prediction$class, donnees$Class, curve = TRUE );
        }else if(i==8){#SVM
```

```
                modele <- svm(donnees$Class ~ ., data=donnees, kernel="linear")
        }else if(i==9){
                modele <- svm(donnees$Class ~ ., data=donnees,
kernel="polynomial")
        }else if(i==10){
                modele <- svm(donnees$Class ~ ., data=donnees, kernel="radial")
        }else if(i==11){
                modele <- svm(donnees$Class ~ ., data=donnees, kernel="sigmoid")
        }
        if(i > 7){
                prediction <- predict(modele,donnees)
                pr <- pr.curve( prediction, donnees$Class, curve = TRUE );
        }
        if(i < 6){
                pr <- pr.curve( prediction, test$Class, curve = TRUE )
        }
        stat[i,1] <-pr$auc.integral
}
stat
```