Algorithme k-NN

Exercice 1 Détection de cancer

1. Collecte des données
   Nous utiliserons des données du UCI Machine Learning Repository [1]. Récupérez le fichier `wisc_bc_data.csv` et enregistrez-le dans votre répertoire de travail. Ce fichier contient des mesures de biopsies de cellules cancéreuses. Les valeurs représentent les caractéristiques des noyaux des cellules.
2. Exploration et préparation des données

   (a) Importez et affichez les données.

```
patients <- read.csv("wisc_bc_data.csv", stringsAsFactors
    = FALSE)
str(patients)
```

   En déduire le nombre d'exemples et de caractéristiques.
   La première variable est une variable entière nommée `id`. Il s'agit d'un identifiant unique pour chaque patient. Cette variable ne fournit pas d'information utile, il faut donc l'exclure du modèle. Il suffit de faire une copie du dataframe `patients` sans la colonne 1.
   La variable suivante `diagnosis` est le résultat que l'on souhaite prédire. Elle indique si le cas est bénin ou malin. La fonction `table()` indique le nombre de cas bénins ou malins. La plupart des classifieurs en R ont besoin que la caractéristique cible soit codée comme un "factor", il faut donc recoder la variable `diagnosis`.

```
# supprimer les id
patients <- patients[-1]

table(patients$diagnosis)

# recoder la variable diagnostic
patients$diagnosis <- factor(patients$diagnosis, levels =
    c("B", "M"), labels = c("Benign", "Malignant"))

# apercu des donnees et de leur repartition
round(prop.table(table(patients$diagnosis)) * 100, digits
    = 1)

# quelques caracteristiques
summary(patients[c("radius_mean", "area_mean", "smoothness
    _mean")])
```

1. `http://archive.ics.uci.edu/ml`

Affichez quelques informations sur les caractéristiques à l'aide de la fonction `summary`. Qu'en déduisez-vous ? Que faut-il faire avant d'entraîner votre modèle ?

(b) normalisation des données : dans un premier temps, on utilisera une normalisation min_max.

```
normalize <- function(x) {
 return ((x - min(x)) / (max(x) - min(x)))
}

patients_n <- as.data.frame(lapply(patients[2:31],
    normalize))

summary(patients_n$area_mean)
```

Que constatez-vous pour les nouvelles valeurs ?

(c) préparation des données : création d'un jeu d'entraînement et d'un jeu de tests
On va garder 469 enregistrements pour l'entraînement et 100 pour le test. Les jeux d'entraînement et de tests normalisés ne contiennent pas la variable cible `diagnosis`. Pour entraîner le k-NN, il faut sauvegarder ces labels dans des vecteurs `patients_train_labels` et `patients_test_labels` .

```
# creation jeu d'entrainement / jeu de test
patients_train <- patients_n[1:469, ]
patients_test <- patients_n[470:569, ]

patients_train_labels <- patients[1:469, 1]
patients_test_labels <- patients[470:569, 1]
```

3. entraînement du modèle
Appliquer kNN : construire le modèle à partir du jeu d'entraînement, spécifier les labels des données annotées, spécifier le nombre de voisins à considérer, tester le modèle avec le jeu de test.

`p <- knn(train, test, class, k)` avec
— `train` : dataframe contenant les données d'entraînement
— `test` : dataframe contenant les données tests
— `class` : vecteur contenant la classe de chaque donnée d'entraînement
— `k` : entier indiquant le nombre de plus proches voisins
La fonction retourne un vecteur avec les classes prédites pour chaque ligne du dataframe test.

Le jeu d'entraînement contient 469 exemples. On peut prendre $k = 21$, nombre impair proche de la racine carrée de 469. Comme on a 2 catégories possibles pour le résultat, le fait d'avoir un nombre impair de voisins permet d'éliminer le risque d'avoir un vote égalitaire (50-50) à la fin.

```
library(class)

patients_test_pred <- knn(train = patients_train, test =
   patients_test, cl = patients_train_labels, k=21)
```

4. évaluation des performances du modèle
   On peut utiliser la fonction `CrossTable()` du package `gmodels`

```
install.packages("gmodels", dependencies=TRUE)
library(gmodels)

# prediction vs verite de terrain
CrossTable(x = patients_test_labels, y = patients_test_pred,
   prop.chisq=FALSE)
```

   En déduire le nombre de vrais positifs TP, vrais négatifs TN, faux positifs FP, faux négatifs FN et le pourcentage de masses mal classées.

5. amélioration des performances du modèle
   (a) Testez d'autres valeurs de k
   (b) Testez une autre méthode de normalisation, comme la standardisation z-score avec la fonction `scale()`. Refaites les expériences avec différentes valeurs de k.
   (c) Pour chaque méthode calculez précision, rappel et F-score.