TDM : Classification

Concernant les TDM ou TP qui ont lieu en B141, vous utiliserez l'image **Deb-NF-MA431-2021-img**. Vous commencez par ouvrir Rstudio, créez un nouveau projet dans lequel **vous déposerez les données**. Créez ensuite un fichier RMarkdown en précisant une sortie en html. Vous enregistrerez vos commandes dans ce R Markdown en respectant la syntaxe ainsi que vos commentaires entre les différentes commandes. Pour compiler le fichier R Markdown, vous devez utilisez le bouton knit (tricoter).
Rstudio a l'avantage d'être composé de quatre consoles, une, en haut à gauche, pour les R scripts ou R MArkdown, une pour l'environnement et l'historique (en haut à droite, avec les différentes variables, données, etc.), une pour les résultats de vos commandes validées (en bas à gauche) et une pour l'aide, les packages et les graphiques notamment (en bas à droite).

Dans la suite de l'énoncé, toutes les commandes R seront indiquées en *italique*.
Vous devez répondre en commentaire soit en dehors de la zone de commandes R, soit sur la ligne de la commande en utilisant le #.
Vous pouvez vous inspirer de la syntaxe proposée lorsque vous ouvrez un fichier RMarkdown. Les titres sont précédés de ## et les zones de commandes R sont entre "'{r} et "'.
L'objectif de ce TDM est de pratiquer, via R, les techniques d'apprentissage supervisé étudiées en cours.
Les données que vous allez étudier sont issues du site des archives de l'UCI Machine Learning Repository. Il s'agit de la SPAMbase Data set. Les données comportent des observations de 58 variables sur 4601 messages, dont certains sont des Spams et d'autres non. Les non-Spams sont identifiés à l'aide la présence de certains mots comme "George" ou bien de code comme "650".

1 Préparation des données

1. Afin de gagner un peu de temps, cette partie est directement disponible sur Chamilo. Regardez les instructions et interprétez les résultats obtenus.
2. Pour construire la base d'apprentissage et la base test, on choisit de garder les trois quarts des données dans la base d'apprentissage et le reste dans la base test. Renseignez l'information puis construire les deux bases.
3. Réalisez le test d'homogénéité. Si les bases ne sont pas homogènes, recommencez jusqu'à obtenir deux bases que vous jugerez homogènes.
4. Réalisez les commandes *Train<-Strain* et *Test<-Stest*. Compilez le document puis effacez ces commandes (vous garderez ainsi vos deux bases qui ne seront pas modifiées à chaque compilation).

2 Classement par l'algorithme des k plus proches voisins

Pour utiliser l'algorithme des k plus proches voisins, on aura besoin de la librairie class. Installez-la.

1. On fixe $k = 5$. La base d'apprentissage servira à l'apprentissage supervisé puis nous le testerons sur la base test. On utilise la commande *knn* pour réaliser un classement à l'aide de l'algorithme. Réalisez une classification pour 5 voisins.
2. Déterminer la matrice de confusion de la prévision sur la base d'apprentissage puis interprétez.
3. Le problème, dans les calculs précédents, c'est que nous avons fixé arbitrairement k à 5. Nous pouvons tester plusieurs valeurs de k pour le déterminer. Choisissons par exemple k en fonction du pourcentage d'erreur.
   Testez différentes valeurs de k (entre 2 et 50 max) puis représentez le taux d'erreurs en fonction de k.
   Quelle valeur de k conseillez-vous ?
   Refaire une prévision avec le k choisi et comparer les matrices de confusion sur la base d'apprentissage.
4. Réaliser la matrice de confusion sur la base test.

3 Prévision de spams à l'aide du SVM

Nous allons maintenant tenter de prévoir si un mail est un spam à l'aide du SVM.

1. Nous allons avoir besoin du package e1071, il faut donc l'installer puis le charger : *library(e1071)*.
   Le package *e1071* permet d'implémenter le SVM. Dans la fonction *svm*, il faut préciser les données, la classe à prédire, le type de noyau, le paramètre de coûts (ou de relaxation) et s'il faut réduire les données ou non. Nous allons ici déterminer un classifieur linéaire (comme spécifié dans le noyau, ou kernel) sans

ajouter de paramètre de coûts et sans centrer ni réduire (scale) les données.
Combien de points supports ont été nécessaires ? A votre avis, ce nombre doit-il être faible ou important ?

2. Déterminer la matrice de confusion du modèle sur la base d'apprentissage :
contrainsvm =table(actual=Strain$y,predicted=nomdusvm$fitted).
Afficher la matrice puis interprétez le modèle sur la base d'apprentissage ?
3. Tester ce modèle sur la base test : *mlintest= predict(mlin,Stest).* Déterminer la matrice de confusion sur la base test et interprétez.
4. Recommencer un nouveau SVM en utilisant un noyau radial et non linéaire. Pour cela, il suffit remplacer linear par radial dans la commande du SVM.
Ce modèle est-il meilleur que le précédent ?
5. Faites un nouvel essai avec cette fois-ci un noyau polynomial de degré 5 et en introduisant la notion de coûts à 1000.
Ce modèle est-il meilleur que le linéaire ? que le radial ?

4 Prévision de spams à l'aide d'une regression logistique

Nous allons maintenant tenter de prévoir si un mail est un spam à l'aide d'une regression logistique.

1. Déterminer les coefficients de la regression logistique.
2. Peut-on valider le modèle ? Peut-on améliorer le modèle ?
3. Déterminer la matrice de confusion du modèle sur la base d'apprentissage.
Interpréter les résultats obtenus ici. Déterminer les proportions de faux positifs et de faux négatifs.
4. Tracer la courbe ROC.
5. Tester maintenant le modèle sur la base test et établir la matrice de confusion.
6. Quel classifieur est-il préférable d'utiliser ? Celui construit avec la régression logistique, celui avec le SVM ou bien celui construit avec les k plus proches voisins ?