TP.2 Classification supervisée
Régression linéaire - K plus proches voisins – Bayésien naïf

Binôme : NGUYEN Ngoc Tu – 21710268
SHAYESTEH NIA Vahid – 21512735

1. Données réelles (prostate)

1.1. Description des données prostate

Les données prostate examinent la corrélation entre le niveau de l'antigène spécifique de la prostate et un certain nombre de mesures cliniques chez les hommes qui étaient sur le point de recevoir une prostatectomie radicale. La variable lpsa est la réponse au traitement qui sera discrétiser en deux catégories ; high si lpsa > median(lpsa) et 0 sinon. On veut construire un score de détection de la réponse applicable aux patients. Pour chaque patient on a mesuré une batterie de critères et finalement $p = 3$ critères (lcavol, lweight et age) ont étés retenus pour construire le score.

Les données prostate ont 97 individus avec 10 variables suivantes :
lcavol :log du volume du cancer
lweight :log du poids de la prostate
age :en années
lbph : log de la quantité d'hyperplasie bénigne de la prostate
svi : invasion de vésicules séminales
lcp : log de pénétration capsulaire
gleason : un vecteur numérique
pgg45 : pour cent du score de Gleason 4 ou 5
lpsa : réponse
train : un vecteur logique

On va utiliser les 3 variables explicatives lcavol, lweight and age pour modéliser les classes g et tracer le nuage des points de ces trois variables en indiquant les classes.

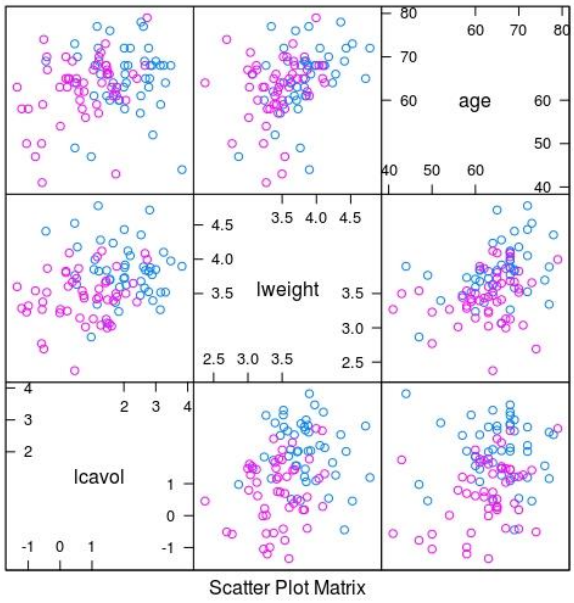

FIGURE 1 : Le scatterplot des trois variables lcavol, lweight, age

1.2. Régression linéaire

Pourquoi la régression linéaire n'est pas adaptée !?

Si on fait le classement, on veut optimiser quelque chose lié à des erreurs de classement. On se soucie seulement de prédire la bonne classe. Lorsqu'on fait la régression, on veut minimiser une certaine distorsion entre la prédiction et la valeur réelle (par exemple : erreur quadratique moyenne). Dans cette situation (le jeu de données prostate, on fait la régression pour prédire des valeurs binaires.

En faisant la regression lineaire, on obtient :

Coefficients de régression :

(Intercept)	*lcavol*	*lweight*	*age*
0.943044969	*0.200877054*	*0.388765006*	*-0.003983511*

Le coefficient de *age* montre que cette variable contribue faiblement dans ce modèle.

On tracer le modèle estimé *lcavol* et *age* pour *lweight* moyen :

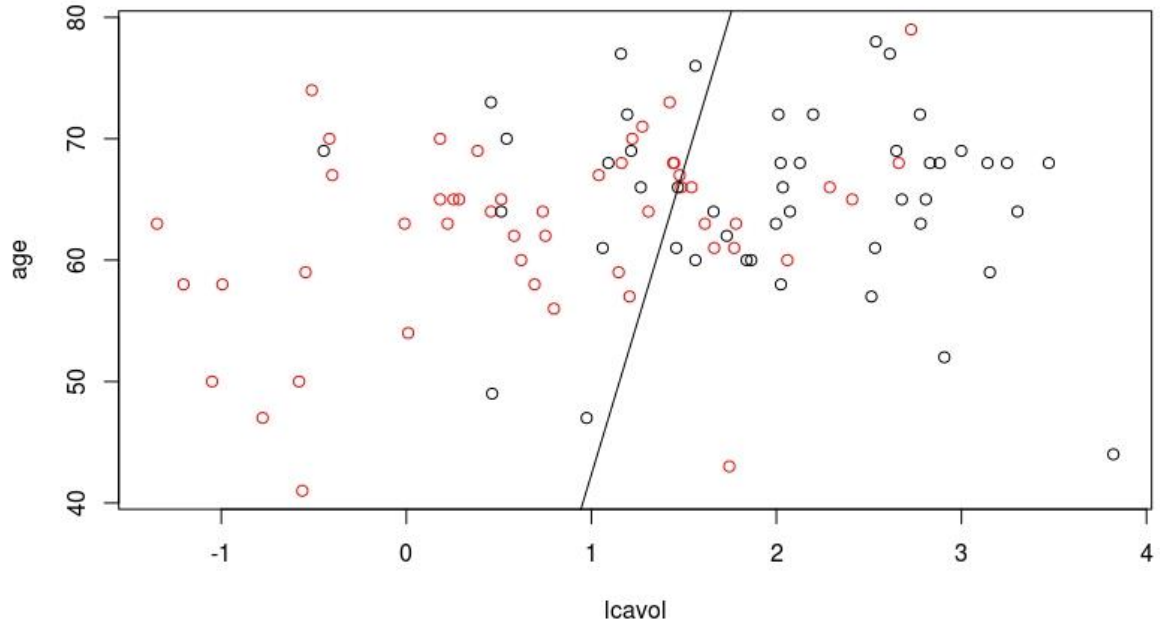

FIGURE 2 : Modèle de régression linéaire

Nombre d'exemples mal classés : *19*

Erreur de classification : *0.1958763*

Matrice de confusion :

		Vrai labels	
		High	Low
Prédiction	High	39	11
	Low	8	39

1.3. KNN

(1) : Créer un jeu de données de données d'apprentissage (75% des données) et un jeu de données test (25% des données) avec random see

(2) : Calculer les taux d'erreur sur les données test pour *k* variant de 1 à 100. Avec la fonction plot, représenter ce taux d'erreur test en fonction de *k* (contrôler que l'abscisse du graphique partde 0). Avec la fonction which.min, trouver le nombre de voisins qui donne la plus petite erreur test.

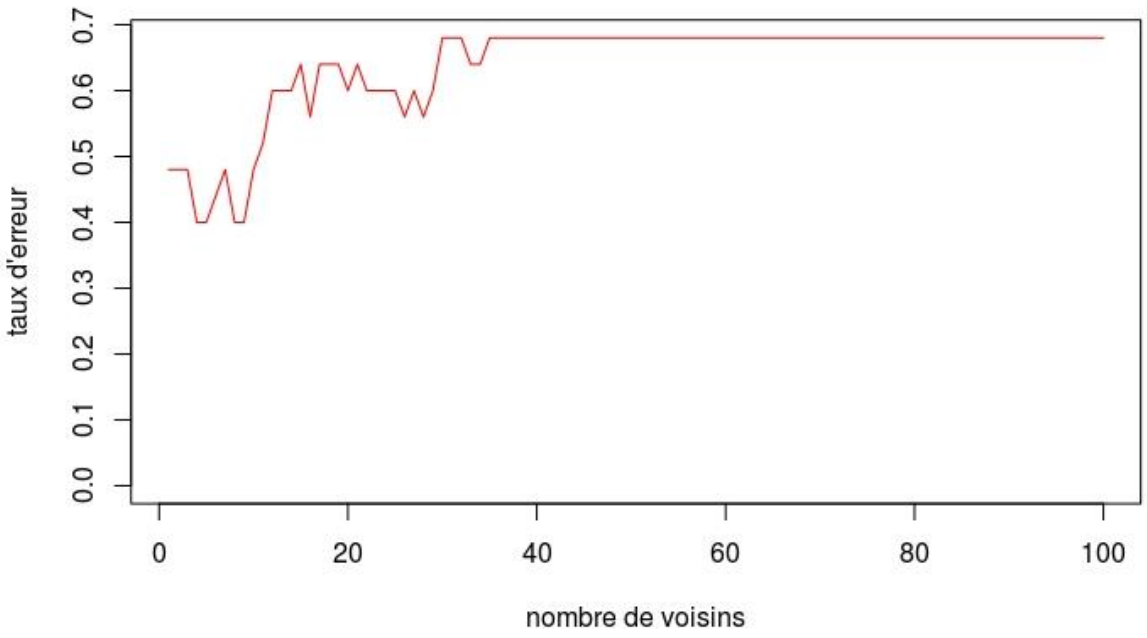

FIGURE 3 : Taux d'erreur pour k variant de 1 à 100

On va choisir le nombre K qui donne le taux d'erreur minimum : K = 4.

(3) Recommencer avec un autre découpage aléatoire apprentissage/test et représenter la courbe d'évolution du taux d'erreur test sur le même graphique qu'à la question précédente.

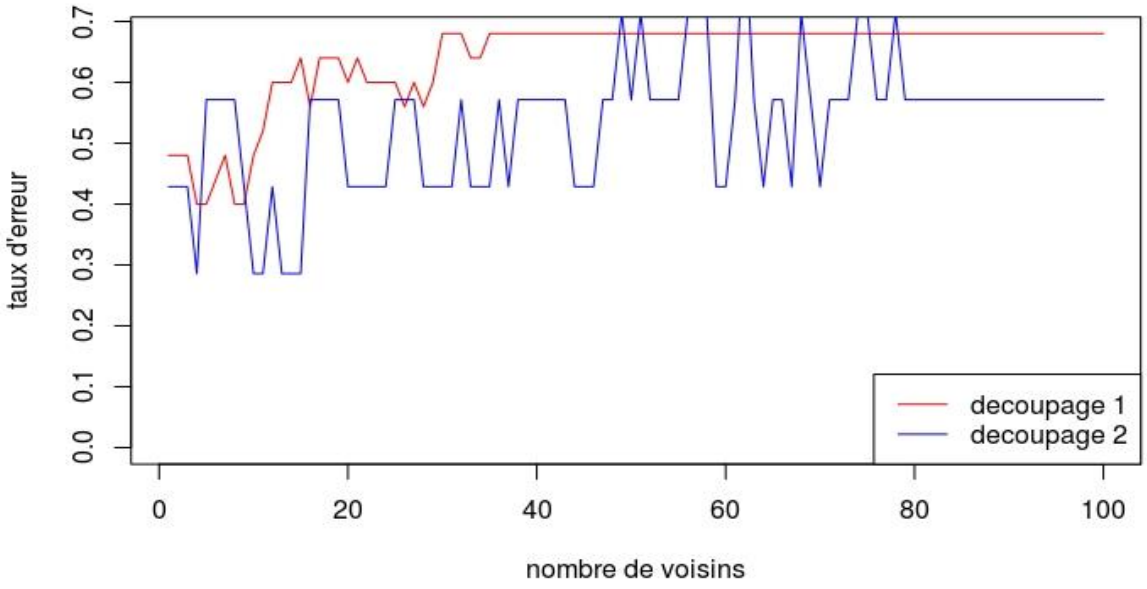

FIGURE 4 : Taux d'erreur pour k variant de 1 à 100 avec 2 découpages

Dans le deuxième essai, le taux de données d'apprentissage est 90%, on peut observer que le taux d'erreur est meilleur. On conclut que : plus d'exemples d'apprentissage → meilleure généralisation .

(4) On va faire 20 découpages, pour chaque découpage on va tester avec k variant de 1 à 100.
Voilà le plot qui montre les erreurs conditionnelles et l'erreur moyenne :

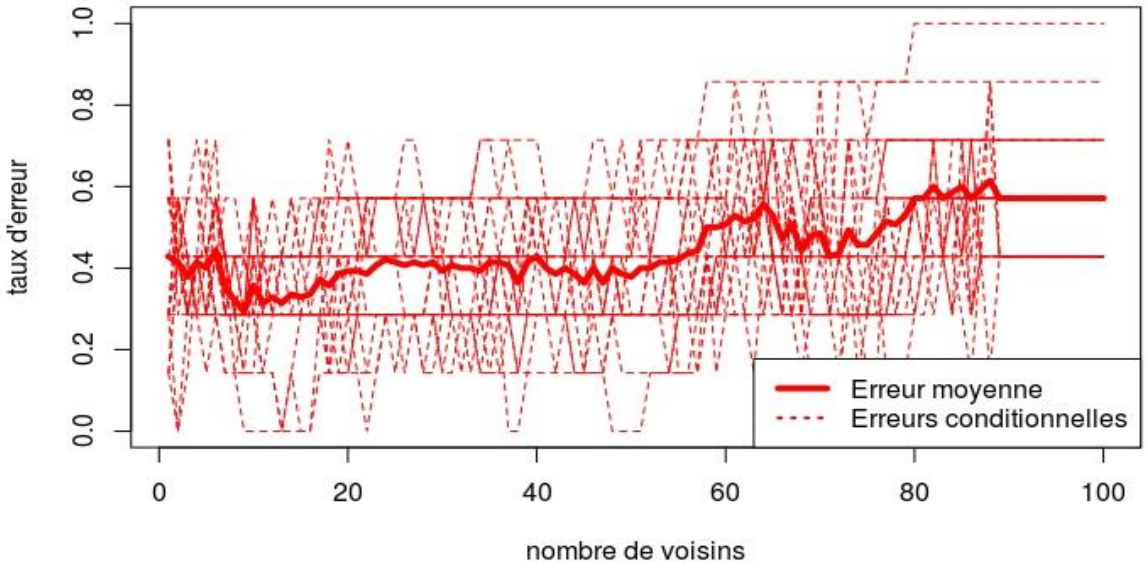

FIGURE 5 : les erreurs conditionnelles et l'erreur moyenne

Ce plot donne le k = 9 qui donne l'erreur moyenne minimum.

(5) Maintenant, on choit le nombre *k* de voisin en utilisant par validation croisée (cross validation) leave-one-out (LOO) avec la fonction knn.cv. Voici le plot :

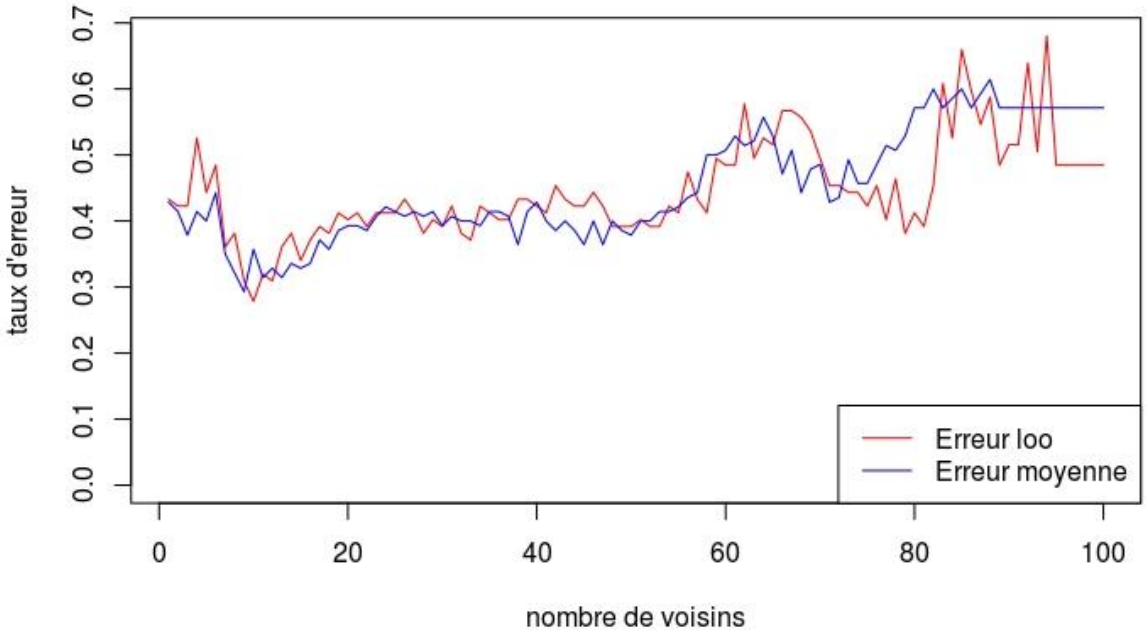

FIGURE 6 : l'erreur Leave-One-Out et l'erreur moyenne pour le k variant de 1 à 100

En utilisant le résultat de la KNN – Validation croisée (LOO), le k optimal est 10.

(6) Faire un petit bilan méthodologique concernant le choix du paramètre *k* :
On choit toujours le k qui donne l'erreur de test minimum

On veut maintenant non seulement choisir *k* mais également avoir une idée de l'erreur de prédiction de ce classifieur. Pour cela, il faut utiliser des données n'ayant jamais été utilisées. Les données doivent donc être découpées en trois parties : apprentissage/validation/test.

Choisir *k* en découpant les 97 données de l'ensemble "apprentissage-validation" en deux parties : une partie "apprentissage" (50% des données) et une partie "validation" (25 % des données). Choisir *k* qui minimise le taux d'erreur moyen sur les ensembles de validations de $B = 25$ découpages.
Dans ce cas :
La partie apprentissage : 49 individus
La partie validation : 24 individus
La partie test : 24 individus

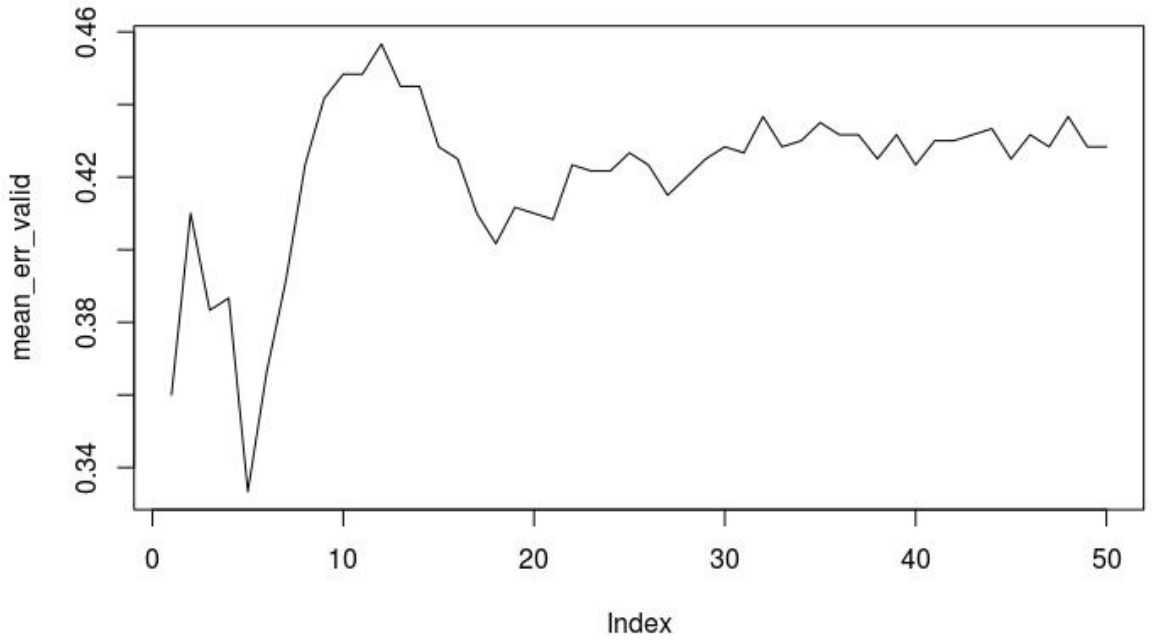

FIGURE 7 : l'erreur moyenne de validation pour k variant de 1 à 100, avec 25 découpages

Le K qui donne le taux d'erreur moyenne de validation est 5. On va construire le KNN avec k = 5 pour obtenir l'erreur de test : 0.416667

Pour les courageux, on pourrait recommencer avec plusieurs découpages des données en deux parties "apprentissage-validation" et "test". Cela permettrait d'avoir une erreur test moyenne, et une idée de sa variabilité. C'est assez rapide à faire avec la méthode de LOO pour le choix de *k*.
L'erreur test moyenne : 0.364
Voici le boxplot des erreur test :

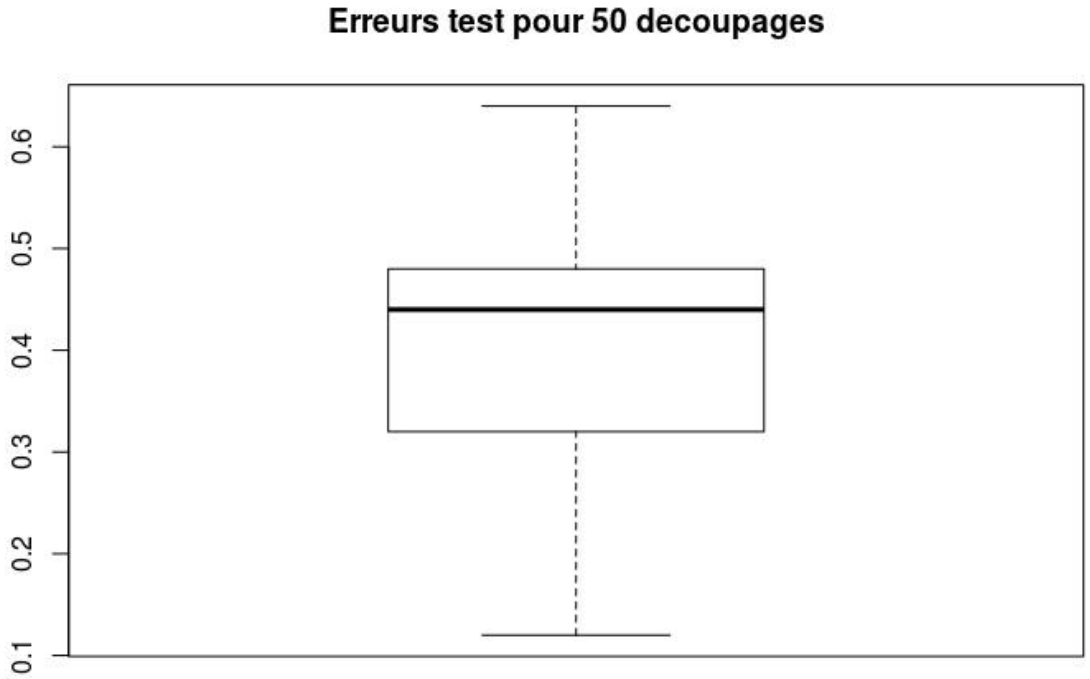

FIGURE 8 : boxplot des erreurs test

1.4. Bayésien naïf

On applique le classifieur bayésien naïf sur les données prostate et on obtient la matrice de confusion suivant :

	Vrais labels	
	High	Low
High	31	6
Low	16	44

Le taux d'erreur : 0.2268

2. Données spam

2.1. Description de la table spam

C'est un ensemble de données collectées chez Hewlett-Packard Labs, qui classe les 4601 e-mails comme spam ou non-spam. En plus de cette étiquette de classe, il y a 57 variables indiquant la fréquence de certains mots et caractères dans l'e-mail.

Les 48 premières variables contiennent la fréquence du nom de la variable (par exemple, entreprise) dans l'e-mail. Si le nom de la variable commence par « num » (par exemple, num650), il indique la fréquence du nombre correspondant (par exemple, 650). Les variables 49-54 indiquent la fréquence des caractères ';', '(', '[', '!', '\ $' Et '\ #'. Les variables 55-57 contiennent la moyenne, la plus longue et la totale des lettres majuscules. La variable 58 indique le type de courrier est "non spam" ou "spam".

2.2. Analyse statistique

Boxplot(spam)

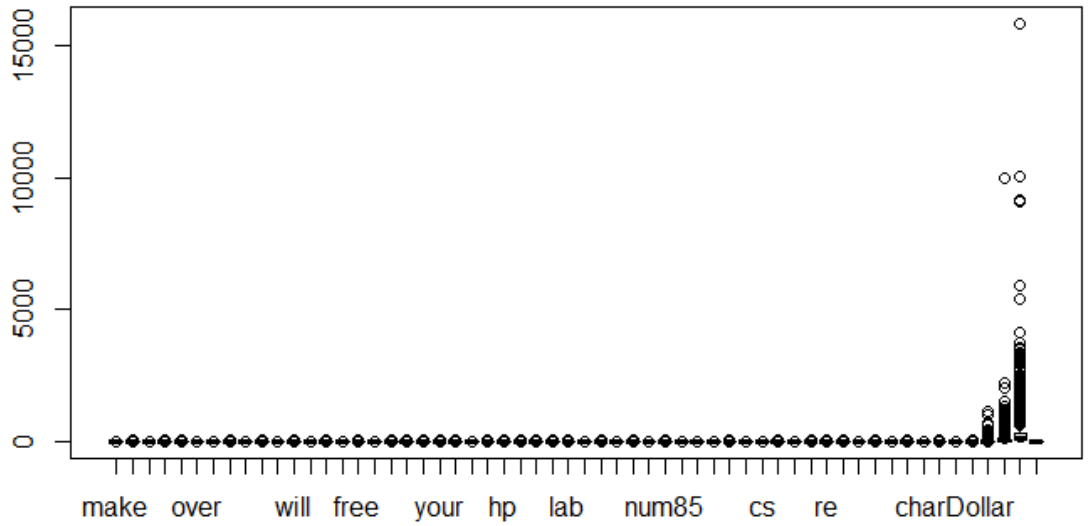

FIGURE 9 : Boxplot de la table spam

On trouve que les variables 55-57 de la table spam contiennent de nombreux valeurs extrêmes. Car le KNN est basé sur la distance euclidienne, ces trois variables vont dominer les autres, et elles peuvent donner des mauvais résultats. Alors, on va non seulement tester avec la table spam d'origine, la table de spam après la normalisation, mais également on va laisser tomber les variables 55-57 pour vérifier cette hypothèse.

2.3. Résultats

On va réaliser une étude comparative des méthodes de classification suivantes : régression linéaire, k plus proches voisins et le classifieur bayésien naïf sur le jeu de données spam/spam après la normalisation, et spam sans 55-57 variables. (Pour la normalisation, c'est diviser chaque case nij de la table spam par la racine carrée du produit des sommes marginales ni. et n.j).

Voici les résultats :

	Spam	**Spam avec la normalisation**	**Spam sans 55-57 variables**
Régression linéaire	- L'erreur de classification : 0.1119	- L'erreur de classification : 0.1389	- L'erreur de classification : 0.1141
KNN seed(30) 75% train, 25% test K : 1→ 100	- L'erreur de test minimum : 0.1903 - K = 1	- L'erreur de test minimum : 0.0851 - K = 9	- L'erreur de test minimum : 0.0799 - K = 5
KNN seed(10) 75% train, 25% test K : 1→ 100	- L'erreur de test minimum : 0.172 - K = 1	- L'erreur de test minimum : 0.0895 - K = 9	- L'erreur de test minimum : 0.0938 - K = 6

KNN 20 découpages 75 %train, 25% test K : 1→ 100	- L'erreur moyenne de test minimum : 0.1849 - K = 1	- L'erreur moyenne de test minimum : 0.0904 - K = 1	- L'erreur moyenne de test minimum : 0.0891 - K = 1
KNN CV LOO utilisation 100% données K : 1→ 100	- L'erreur de test minimum : 0.1691 - K = 1	- L'erreur de test minimum : 0.0904 - K = 1	- L'erreur de test minimum : 0.0891 - K = 1
KNN seed(30) 50 % train, 25 % val, 25 % test, 25 découpages K : 1→ 50	- L'erreur moyenne de validation minimum : 0.2022 - K = 1 - L'erreur de test : 0.1903	- L'erreur moyenne de validation minimum : 0.098 - K = 3 - L'erreur de test : 0.1034	- L'erreur moyenne de validation minimum : 0.1036 - K = 3 - L'erreur de test : 0.0843
KNN CV LOO 75 % train, 25 % test K : 1→ 50	- L'erreur de validation minimum : 0.18 - K = 1 - L'erreur de test : 0.1903	- L'erreur de validation minimum : 0.088 - K = 1 - L'erreur de test : 0.095	- L'erreur de validation minimum : 0.0933 - K = 1 - L'erreur de test : 0.0869
KNN CV LOO utilisation 75 % données 10 découpages K : 1→ 50	- L'erreur moyenne de test : 0.1875	- L'erreur moyenne de test : 0.0918	- L'erreur moyenne de test : 0.0939
Classifieur Bayésien Naïf	-L'erreur moyenne : 0.2865	-L'erreur moyenne : 0.3251	-L'erreur moyenne : 0.2936

On trouve que l'impact de la normalisation est significatif. Le taux d'erreur de validation et de test de KNN seront meilleurs si on fait la normalisation. En plus, on peut conclure que : si on utilise la table de spam sans les variables 55-57, le résultat sera meilleur en comparaison de l'utilisation de la spam d'origine. Et on trouve que le classifieur Bayésien Naïf et la régression linéaire ne sont pas améliorés avec la normalisation de la table spam.