Master MAS
Classification supervisé

TP3 : Calibrer/évaluer une méthode et la comparer à d'autres

Les données concernent $n = 1260$ exploitations agricoles réparties en $K = 2$ classes : la classe des exploitations saines et la classe des exploitations défaillantes. Pour chaque exploitation agricole on a mesuré une batterie de critères économiques et financiers et finalement $p = 4$ ratios financiers ont étés retenus :

- **R2** : capitaux propres / capitaux permanents,
- **R7** : dette à long et moyen terme / produit brut,
- **R17** : frais financiers / dette totale,
- **R32** : (excédent brut d'exploitation - frais financiers) / produit brut.

La variable qualitative à expliquer est la variable difficulté de paiement codée de la manière suivante :
- 0=sain,
- 1=défaillant,

```
load("../data/Desbois.rda")
head(data)
```

```
##   DIFF    R2   R14    R17   R32
## 1    0 0.622 0.232 0.0884 0.431
## 2    0 0.617 0.150 0.0671 0.399
## 3    0 0.819 0.485 0.0445 0.319
## 4    0 0.733 0.373 0.0621 0.431
## 5    0 0.650 0.256 0.0489 0.431
## 6    0 0.755 0.186 0.0243 0.426
```

```
table(data$DIFF)
```

```
##
##   0   1
## 653 607
```

Il y a donc 607 exploitations agricoles défaillantes et 653 exploitations agricoles saines dans les données.

On souhaite constuire une régle de décision permettant de prédire la classe d'une nouvelle exploitation agricole. Nous allons comparer certaines méthodes de classification afin de choisir la "meilleure".

Exercice 1. Evaluer la méthode `lda`.

La qualité d'un règle de classification est souvent mesurée par son taux d'erreur. Il existe (au moins) trois approches pour estimer le taux d'erreur d'une méthode :
- le découpage apprentissage/test répété,
- la validation croisée LOO,
- la validation croisée K-folds (éventuellement répérée).

Nous allons appliquer ces trois approches pour **estimer le taux d'erreur de la méthode `lda`** sur ce jeu de données.

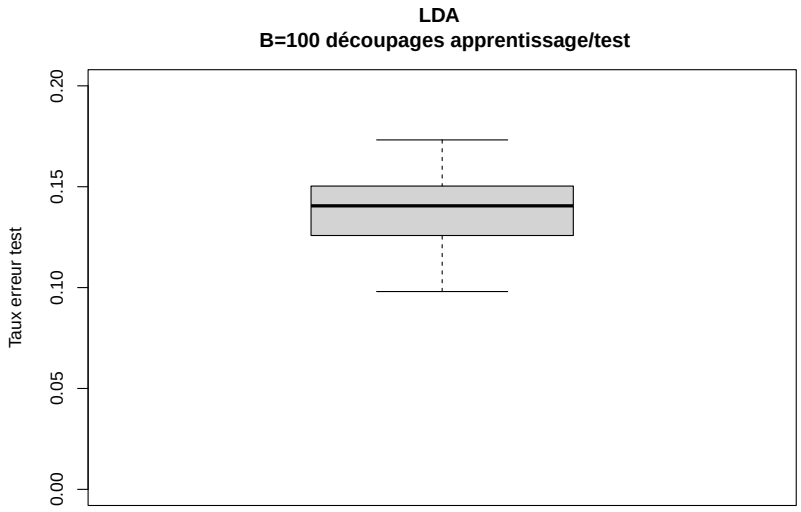
1. L'approche par **découpages apprentissage/test répétés**.

   (a) Calculer le **taux d'erreur test** de la méthode `lda` pour $B = 100$ découpages des $n = 1260$ en 75% de données d'apprentissage et 25% de données test. Vous devez retrouver (à peu prés) les résultats ci-dessous avec la fonction `summary` pour résumer les 100 valeurs obtenues et la fonction `boxplot` pour les représenter.

```
summary(err_test_lda)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.098   0.127   0.141   0.138   0.150   0.173
```

```
boxplot(err_test_lda, ylab="Taux erreur test", ylim=c(0,0.2),
        main=sprintf("LDA \n B=%s découpages apprentissage/test",B))
```



   (b) Commentez ces résultats ?

2. L'approche par **validation croisée 5-folds répétée**.

   (a) Calculer le **taux d'erreur de validation croisée 5-folds** pour $B = 100$ découpages des $n = 1260$ données en 5 folds d'effectifs égaux. Vous devez retrouver (à peu prés) les résultats ci-dessous avec la fonction `summary` pour résumer les 100 valeurs obtenues.

```
summary(vec_err_cv)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.132   0.136   0.137   0.137   0.138   0.141
```

   (b) Commentez ces résultats ?

3. L'approche par **validation croisée LOO** (Leave One Out).

   (a) Calculer le **taux d'erreur de validation croisée LOO**.

```
# Erreur de validation croisée LOO
sum(pred!=data$DIFF)/n
```

```
## [1] 0.137
```

   (b) Commentez ce résultat.

Exercice 2. Calibrer et évaluer la méthode `knn`

On souhaite maintenant **estimer le taux d'erreur de la méthode `knn`** sur ces données. Mais les performances de cette méthode dépendent de la valeur de l'hyperparamètre k. Il faut donc évaluer la méthode `knn` en utilisant une procédure qui choisit automatiquement le nombre k de voisins.

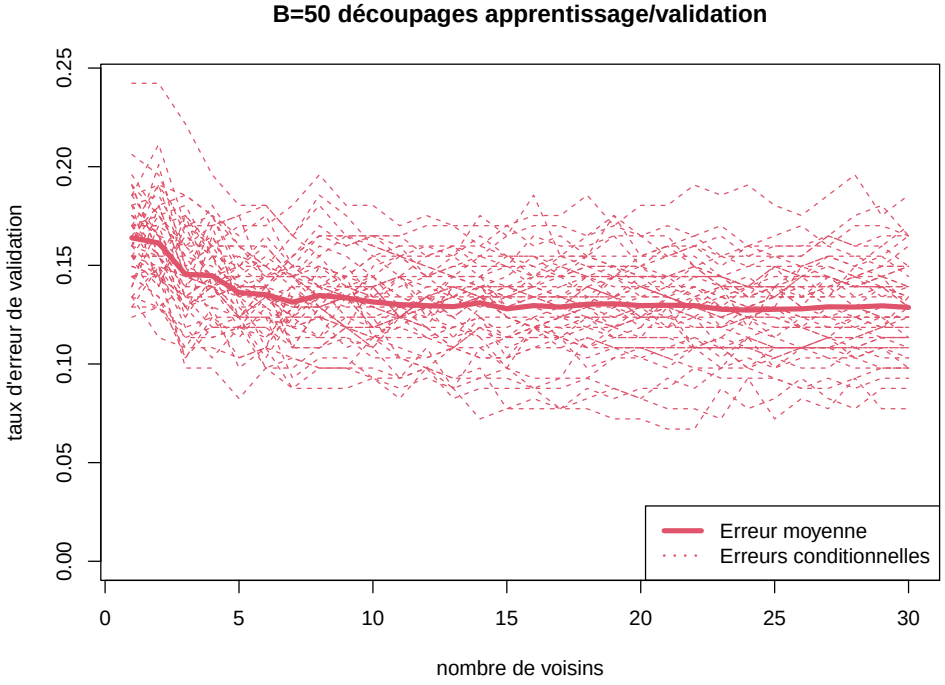
Cette procédure est la suivante :

1. Constuire une grille de valeurs de k.
2. Estimer pour chaque valeur de k dans cette grille le taux d'erreur de la méthode `knn` sur des données de validations (ni des données d'apprentissage, ni des données test).
3. Choisir la valeur de k qui donne la plus petite **erreur validation**.

On utilisera donc trois échantillons :
— un échantillon d'apprentissage,
— un **échantillon de validation** (pour choisir k),
— un échantillon test.

1. Mettre de côté 25% de données test.

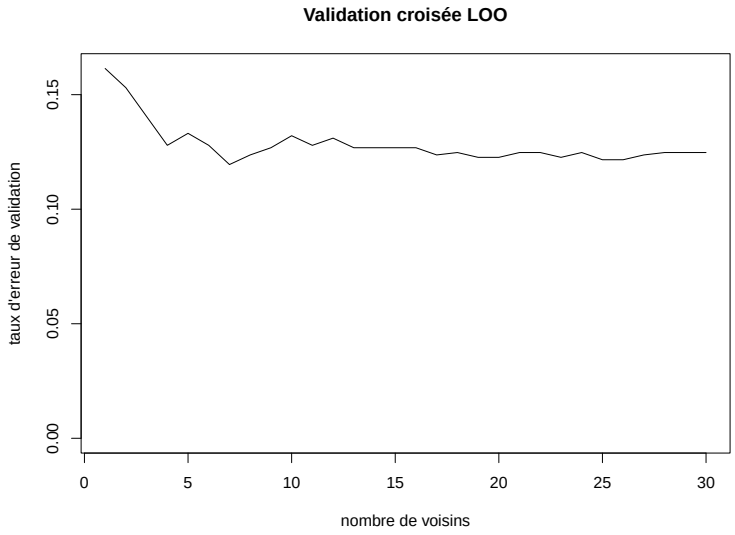
2. Découper $B = 50$ fois les données restantes en 80% pour l'apprentissage et 20% pour la validation et choisir automatiquement k qui minimise l'**erreur moyenne de validation**. Pour cela, vous devez retrouver (à peu prés) le graphique ci-dessous des erreurs moyennes de validation et des erreurs conditionnelles (selon le découpage). Quelle valeur de k optimale trouvez-vous ? Commenter ce graphique et ce résultat.




```
kopt <- grille_k[which.min(mean_err_valid)]
kopt

## [1] 24
```

3. Choisir maintenant automatiquement k qui minimise l'erreur de validation croisée LOO (vous pouvez utilsier la fonction `knn.cv`). Pour cela vous devez retrouver (à peu prés) le graphique ci-dessous. Quelle valeur de k optimale trouvez-vous ? Commenter ce graphique et ce résultat.




```
kopt <- grille_k[which.min(err_valid)]
kopt
```

```
## [1] 7
```

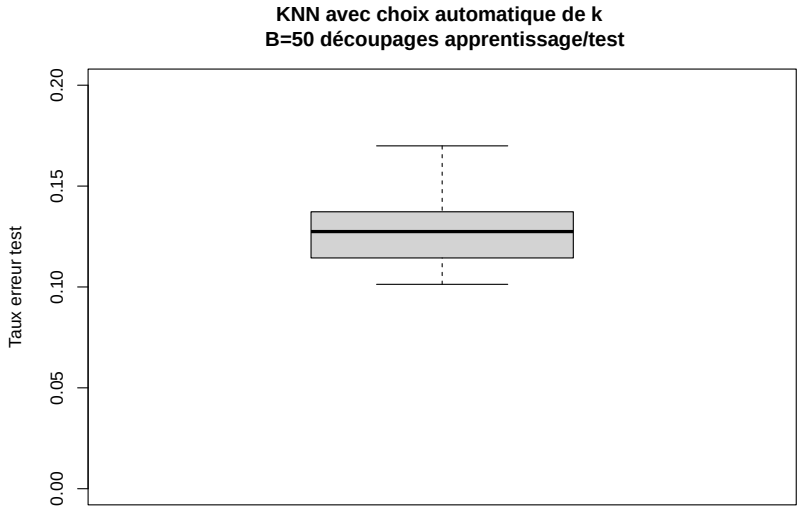
Remarque : on pourrait également choisir le nombre de voisins par validation croisée 5-folds ou 10-folds.

4. Prédire les données test avec la méthode `knn` et le nombre k choisi automatiquement par validation croisée L00. Quelles données utilisez-vous pour l'apprentissage ? Quel est le taux d'erreur test pour ce découpage (et cette valeur de k) ?
5. Si les données test son modifiées, on obtient un autre taux d'erreur test avec la même procédure (choix de k par validation LOO et prédiction des données test avec cette valeur de k). Il faut donc recommencer la procédure plusieurs fois pour évaluer cette méthode.

   (a) Recommencez toute la procédure avec $B = 50$ échantillons test différents. Vous devez retrouver (à peu prés) les résultats ci-dessous avec la fonction `summary` pour résumer les erreurs test obtenues et la fonction `boxplot`.

```
summary(err_test_knn)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.101   0.115   0.128   0.129   0.136   0.170
```



(b) Commenter ces résultats

(c) A votre avis, peut-on comprer ces résultats avec ceux obtenus avec de la méthode `lda` à la fin de l'exercice 1 ?

(d) A votre avis, pourquoi a-t-on pris ici $B = 50$ découpages et non pas $B = 100$ découpages comme pour la méthode `lda` ?

Remarque : Une valeur différente de k peut être obtenue à chaque découpage. Cela n'est pas un problème car l'objectif est d'évaluer la performance de la méthode `knn` (avec calibration de k par validation croisée LOO) afin de la comparer à d'autres méthodes (par exemple LDA, logistique, forêts aléatoires).

Exercice 3. Comparer les performances de `lda` et `knn`

On veut savoir maintenant laquelle des deux méthodes (`lda` ou `knn`) est la meilleure (a le plus petit taux d'erreur). Pour cela, il faut calculer les taux d'erreur test obtenus sur **les mêmes découpages** par les deux méthodes.

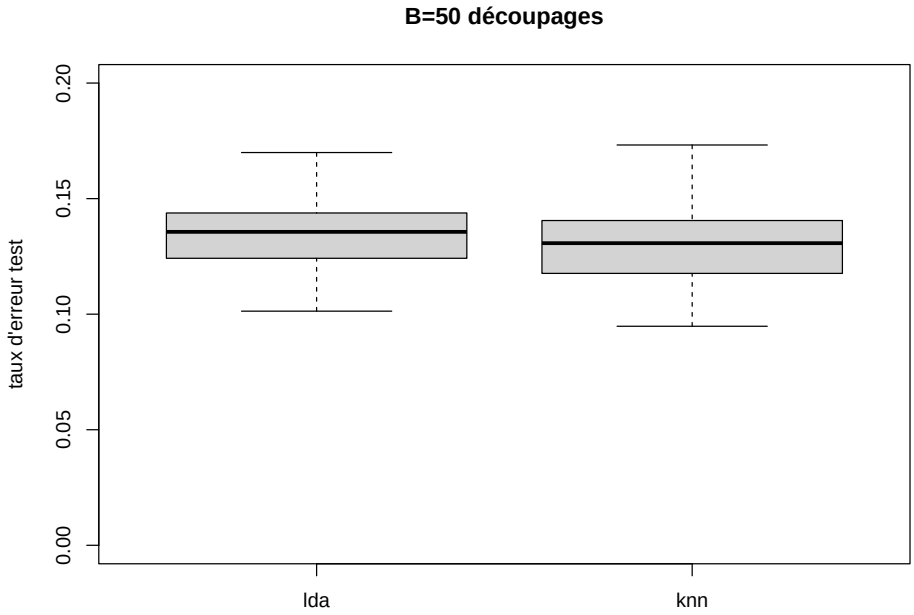
1. Calculer sur les mêmes $B = 50$ découpages apprentissage/test, le taux d'erreur test de la méthode (`lda` et le taux d'erreur test de la méthode `knn` avec choix automatique de k par validation croisée L00). Vous devez obtenir (à peu prés) les boxplots ci-dessous.

```
summary(err_test_lda)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.101   0.124   0.136   0.134   0.144   0.170

summary(err_test_knn)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0948  0.1176  0.1307  0.1295  0.1397  0.1732
```



2. Quelle méthode préférez-vous ? Pourquoi ?
3. On suppose que vous avez choisi la méthode `knn` avec choix automatique de k. Quel est finalement le nombre de voisins utilisé ? Prédire avec cette méthode la classe de la nouvelle exploitation agricole ayant les valeurs suivantes sur les 4 ratios financiers :

```
R2    R14    R17    R32
0.700 0.300 0.100 0.500
```

Quelle confiance a-t-on dans cette peédiction ?