Master MAS
Classification supervisé

TP5 : Score, courbe ROC et courbe Lift en classification binaire

Exercice 1. On reprend les données sur les exploitations agricoles.

```
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) # DIFF est la variable à expliquer (1=défaillant, 0=sain)

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

On veut construire un score de détection du risque financier des exploitations agricoles. Un score en classification binaire est associé à une des classes (généralement celle considérée comme positive). Ici on prend :

- prédire 1 (défaillant)=positif,
- prédire 0 (sain)=négatif.

Dans cet exercice le score (note) d'une exploitation agricole sera donc sa probabilité (à posteriori) d'être défaillante. Il prendra ses valeurs dans $[0, 1]$ et le risque (d'être défaillant) sera d'autant plus grand que le score sera proche de 1. Un score de 0 correspondra à un risque nul. On utilisera les méthodes `knn` et `lda` pour estimer ces probabilités à posteriori et donc les scores.

1. On commence par découper aléatoirement les $n = 1260$ exploitations agricoles en 954 données d'apprentissage et 306 données test.

   ```
   set.seed(10)
   n <- nrow(data)
   tr <- sample(1:n, 954)
   Xtrain <- data[tr, -1]
   Ytrain <- data$DIFF[tr]
   Xtest <- data[-tr, -1]
   Ytest <- data$DIFF[-tr]
   ```

2. Le score d'une exploitation agricole est sa probabilité à posteriori d'être défaillante. Calculons le score des 306 exploitations de l'échantillon test.

(a) On utilise d'abord la fonction `knn` pour estimer les probabilités à posteriori.

```
 # méthode knn avec choix de k par validation croisée LOO
kmax <- 40
grille_k <- 1:kmax
err <- rep(NA, length(grille_k))
for (k in 1:length(grille_k))
{
  pred <- class::knn.cv(Xtrain, Ytrain, grille_k[k])
  err[k] <- sum(pred!=Ytrain)/length(Ytrain)
}
kopt <- which.min(err)
res <- class::knn(Xtrain, Xtest, Ytrain, k=18, prob=TRUE)
# probabilités à posteriori
prob <- rep(NA, nrow(Xtest)) # probabilités d'être défaillant
prob[res==1] <- attr(res,"prob")[res==1]
prob[res==0] <- 1-attr(res,"prob")[res==0]
prob_knn_test <- data.frame("sain"=1-prob, "def"=prob)
prob_knn_test[1:4,]
```

```
##    sain   def
## 1 1.000 0.000
## 2 0.611 0.389
## 3 0.667 0.333
## 4 0.833 0.167
```

La première exploitation agricole a une probabilité à posteriori (estimée avec `knn`) d'être défaillant de 0 et donc un risque nul. La seconde a une probabilité d'être défaillante de 0.38 soit donc un peut moins d'une chance sur deux d'être défaillante. Finalement les scores des 306 exploitations agricoles de l'échantillon test se trouvent dans seconde colonne de `prob_knn_test`.

```
score_knn_test <- prob_knn_test[,2]
```

(b) On utilise maintenant la fonction `lda` pour estimer les probabilités à posteriori.

```
library(MASS)
g <- lda(Xtrain, grouping=as.factor(Ytrain)) # apprentissage
prob_lda_test <- predict(g, Xtest)$posterior # test
prob_lda_test[1:4,]
```

```
##        0      1
## 14 0.931 0.0691
## 27 0.683 0.3174
## 30 0.791 0.2093
## 38 0.880 0.1200
```

La première exploitation agricole a une probabilité à posteriori (estimée avec `lda`) d'être défaillant de 0.06. C'est cohérent avec l'estimation trouvée avec `knn`. La seconde a une probabilité d'être défaillante de 0.31 ce qui là encore est cohérent avec l'estimation obtenue avec `knn`. Finalement les scores des 306 exploitations agricoles de l'échantillon test se trouvent dans seconde colonne de `prob_lda_test`.

```
score_lda_test <-prob_lda_test[,2]
```

3. Nous voulons maintenant évaluer la capacité d'un score à bien discriminer les exploitations saines des exploitations défaillantes. Pour cela nous allons constuire la courbe ROC associée au score obtenu avec `knn` (`score_knn_test`) et évaluer l'aire sous cette courbe avec le package `ROCR`.

```
library(ROCR)
```

Pour rappel, si on se fixe un seuil $s \in [0,1]$, on obtient un vecteur de prédictions en affectant les observations qui ont un score supérieur ou égal au seuil s à la classe positive (ici 1=défaillant) et les autres à la classe négative (ici 0=sain). Jusqu'à présent, on a toujours utilisé un seuil (cutoff) de 0.5 pour obtenir les prédictions. La courbe ROC apporte plus d'informations qu'un seul vecteur de prédiction obtenu avec $s = 0.5$. En effet elle est construite à partir d'une grille de seuils (grille de valeurs dans $[0,1]$). Pour chaque seuil s dans cette grille, le taux de faux positifs et le taux de vrais positifs de la prédiction obtenue avec ce seuil donnent l'abscisse et l'ordonnée d'un point de la courbe ROC. Il y aura autant de points que de seuils dans la grille.

(a) La fonction `prediction` du package `ROCR` renvoie de nombreux résultats en fonction d'une grille de seuils.

```
pred <- prediction(score_knn_test, Ytest, label.ordering=c("0","1"))
# label.ordering : indique le libellé de la classe negative puis positive
```

`pred` est un objet de classe S4. Le symbole @ est donc utilisé pour récupérer les résultats. On peut ainsi récupérer la grille des valeurs de seuils (cutoffs).

```
pred@cutoffs[[1]]
```

```
##  [1]    Inf 1.0000 0.9444 0.8889 0.8333 0.7778 0.7222 0.6667 0.6111 0.5556
## [11] 0.5000 0.4444 0.3889 0.3333 0.2778 0.2222 0.2105 0.1667 0.1111 0.1053
## [21] 0.0556 0.0000
```

Ces seuils fournissent des tableaux de contingence obtenus en croisant les vraies classes et les classes prédites. On récupère avec le code ci-dessous les nombre de vrais positifs (VP) et le nombre de faux positifs (FP) associés à chaque seuil.

```
pred@fp[[1]] # nombre de faux positif pour chaque seuil
```

```
##  [1]   0   1   4   7   7  11  12  13  14  20  24  25  30  34  37  39  40  51  69
## [20]  70  97 145
```

```
pred@tp[[1]] # nombre de vrais positifs pour chaque seuil
```

```
##  [1]   0  67  97 109 118 124 129 132 135 139 141 142 145 146 146 150 150 152 159
## [20] 159 160 161
```

Avec le seuil 0, les 306 exploitations agricoles de l'échantillon test sont prédites défaillantes (classe 1). Or il y a 145 exploitations saines et 161 défaillantes.

```
table(Ytest)
```

```
## Ytest
##   0   1
## 145 161
```

Le seuil 0 donne donc 161 vrais positifs et 145 faux positifs. Cela correspond à un taux de vrais positifs de 1 (TVP) et à un taux de faux positif (TFP) de 1. Il s'agira donc du point $(1,1)$ (en haut à droite) de la courbe ROC de ce score.

Le seuil 0.07 donne 91 faux positifs et 160 vrais positifs. En d'autres termes 160 (sur 161) exploitations agricoles défaillantes sont prédites défaillantes et 90 (sur 145) exploitations agricoles saines sont prédites défaillantes. Le seuil 0.07 correspond donc à un taux de vrais positifs (TVP) de 160/161=0.99 et à un taux de faux positifs de 90/145=0.62. Ce seuil correspondra donc au point $(0.62, 0.99)$ sur la courbe ROC. On comprend que lorsque le seuil augmente, les points sur la courbe ROC se décalent vers la gauche. Tracons maintenant cette courbe en utilisant la fonction `performance` pour obtenir les TFP et les TVP qui formeront les abscisses et les ordonnées de cette courbe.

(b) La fonction `performance` permet d'obtenir de nombreuses mesures de performances d'un score.

i. Le code ci-dessous permet de récupérer les taux de vrais positifs (true positive rate) et les taux de faux positifs (false positive rate) qui seront les abscisses et les ordonnées des points de la courbe ROC.

```
perf <- performance(pred,  "tpr", "fpr")
perf@x.values[[1]] # fpr en abscisse
```

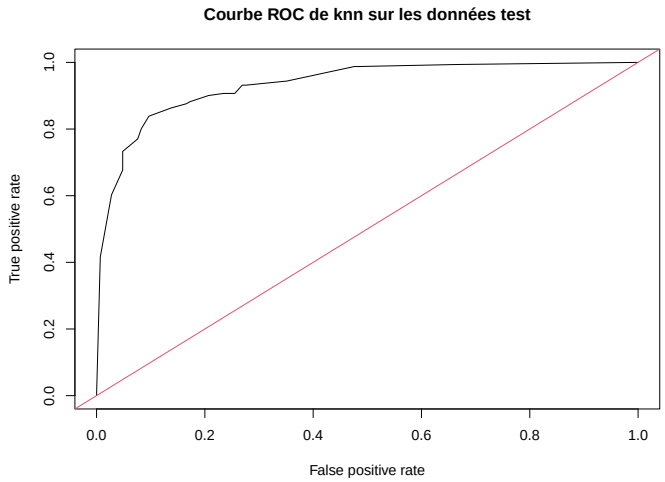
```
##  [1] 0.0000 0.0069 0.0276 0.0483 0.0483 0.0759 0.0828 0.0897 0.0966 0.1379
## [11] 0.1655 0.1724 0.2069 0.2345 0.2552 0.2690 0.2759 0.3517 0.4759 0.4828
## [21] 0.6690 1.0000
```

```
perf@y.values[[1]] # tpr en ordonnée
```

```
##  [1] 0.000 0.416 0.602 0.677 0.733 0.770 0.801 0.820 0.839 0.863 0.876 0.882
## [13] 0.901 0.907 0.907 0.932 0.932 0.944 0.988 0.988 0.994 1.000
```

La méthode `plot` permet alors tracer la courbe ROC.

```
plot(perf, main="Courbe ROC de knn sur les données test")
abline(a=0, b=1, col=2)
```



La première bissectrice symbolise la courbe ROC que l'on obtiendrait avec un très mauvais score (classes mélangées aléatoirement le long du score). Cette situation correspond à une aire sous la courbe ROC de 0.5.

Si cette courbe passait par le point (0,1) (en haut à gauche), il existerait un seuil permettrant d'avoir 100% de vrais positifs (toute les exploitations défaillantes prédites défaillantes) et 0% de faux positifs (aucune exploitation saine prédite défaillante). Une courbe ROC passant par ce point correspondrait à un score parfait et à une aire sous la courbe de 1.

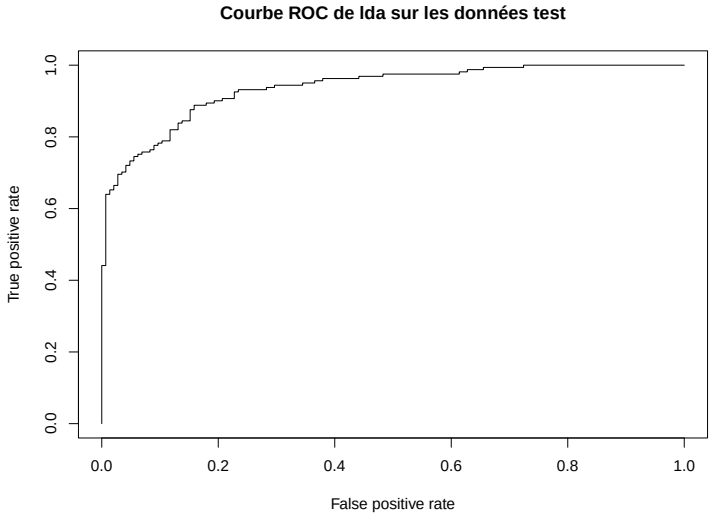
ii. Le code ci-dessous permet de récupérer la valeur de l'aire sous la courbe ROC.

```
perf <- performance(pred, "auc")
auc <- perf@y.values[[1]]
auc
```

```
## [1] 0.934
```

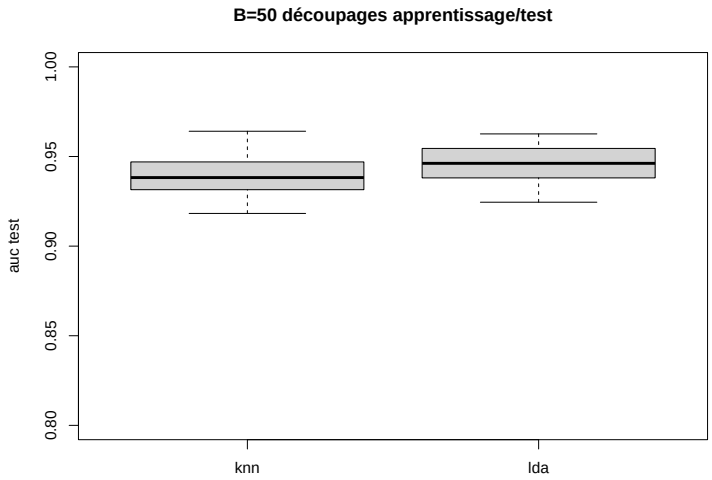
L'air sous la courbe ROC de la méthode `knn` (sur données test) vaut 0.93. Une valeur de 1 correspondrait à un score parfait capable de prédire parfaitement les données. Une valeur de 0.5 correspondrait à la pire des situations (score absolument pas discriminant).

4. En vous aidant du code de la question précédente, constuire la courbe ROC associée au score obtenu avec la méthode `lda` (`score_lda_test`).



Vous devez trouver que l'aire sour la courbe ROC est environ de 0.93 soit une valeur très proche de celle obtenu avec le score de `knn`.

5. Pour finir, on veut comparer les deux méthodes `knn` et `lda` en comparant les performances de leurs scores. Pour cela, on calcule l'aire sous la courbe ROC des deux méthodes pour $B = 50$ découpages aléatoires apprentissage/test. Vous devez retrouver un graphique proche de celui-ci :



Les boxplot des valeurs de `auc` (area under curve) permettent de comparer de manière équitable les scores des deux méthodes, pourquoi ? Commenter ces résultats

6. La méthode `knn` (avec choix de k par validation L00) semble donner des résultats légèrement moins bons que la méthode `lda`. On réalise donc un petit test de comparaison de moyennes. Commenter les résultats.

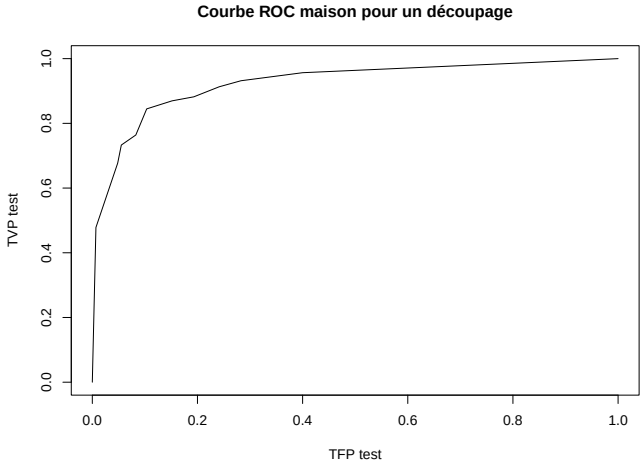
```
meth <- c(rep("lda", 50), rep("knn", 50))
auc <- c(auc_lda, auc_knn)
t.test(auc~meth)

##
##  Welch Two Sample t-test
##
## data:  auc by meth
## t = -3, df = 98, p-value = 0.005
## alternative hypothesis: true difference in means between group knn and group lda is not
## 95 percent confidence interval:
##  -0.01026 -0.00185
## sample estimates:
## mean in group knn mean in group lda
##             0.940             0.946
```

Exercice 2. On reprend les données de l'exercice 1 avec le même découpage apprentissage/test.

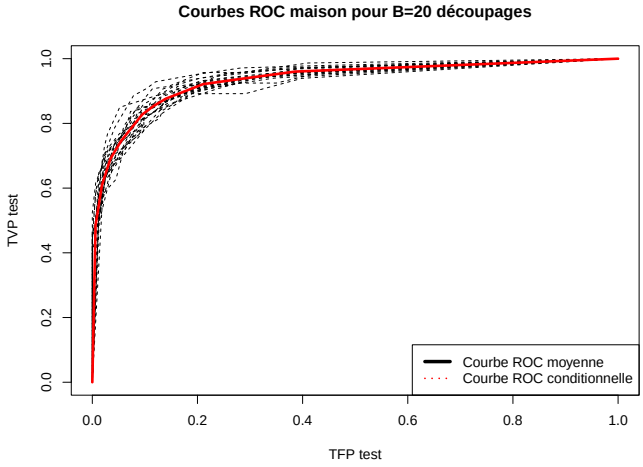
```
load("../data/Desbois.rda")
set.seed(10) # pour avoir des résultats reproductibles
n <- nrow(data)
tr <- sample(1:n, 954)
Xtrain_val <- data[tr, -1]
Ytrain_val <- data$DIFF[tr]
Xtest <- data[-tr, -1]
Ytest <- data$DIFF[-tr]
```

1. Constuire pour ce découpage la courbe ROC de la méthode `knn` (avec $k = 13$ voisins) sans utiliser le package `ROCR` et en utilisant de la grille de seuils ci-dessous :

```
s <- seq(from=0, to=1, by=0.1)
s <- c(s, Inf)
```



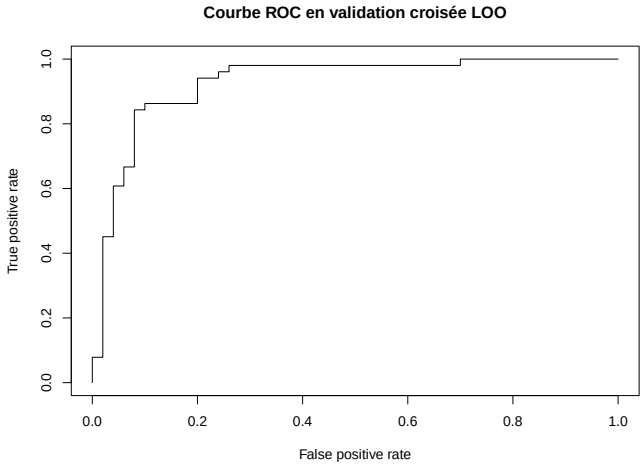
2. Recommencer pour $B = 20$ découpages apprentissage/test et représenter les 20 courbes ROC et la courbe ROC moyenne.



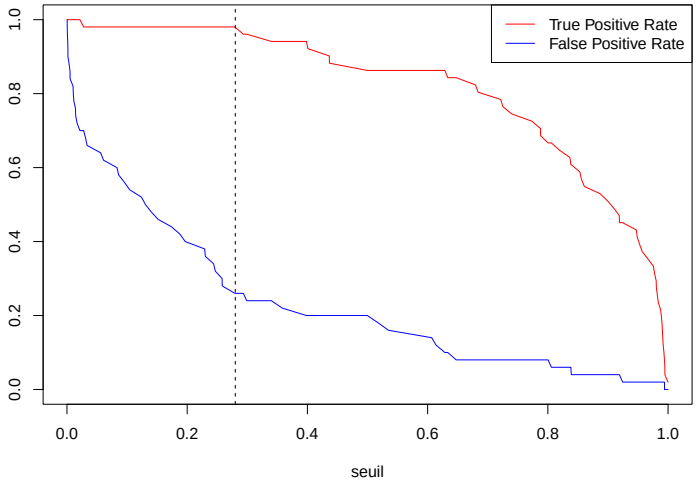
Exercice 3. On reprend dans cet exercice les données `infarctus` pour évaluer la capacité de la méthode `lda` à bien prédire la variable "PRONO".

```
load("../data/infarctus.rda")
X <- infarctus[,-1]
Y <- infarctus$PRONO
```

1. Quelle sera la modalité de la variable 'PRONO' qui sera considérée comme positive ?
2. Constuire avec le package `ROCR` la courbe ROC de la méthode `lda` par validation croisée LOO. Vous devez retrouver le graphique ci-dessous.



   Commenter ce graphique. Donne-t-il des informations sur la capacité de la méthode `lda` à bien prédire la modalité "DECES" sans trop se tromper pour la modalité "SURVIE" ?
3. Utiliser maintenant le package `ROCR` pour constuire le graphique ci-dessous.



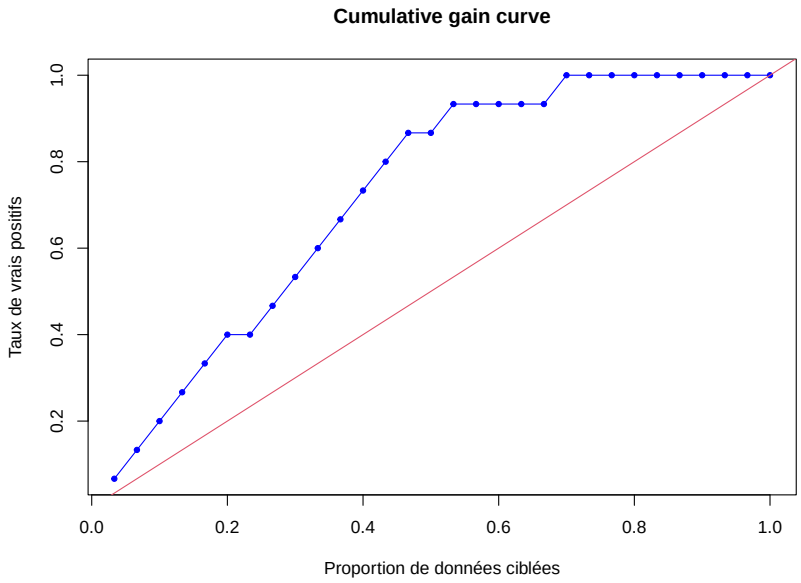
Commenter ce graphique. Quelle nouvelle règle de classification permettrait d'obtenir un taux de vrai positif proche de 100%, avec le taux de faux positifs le plus petit possible (autour de 20%) ?

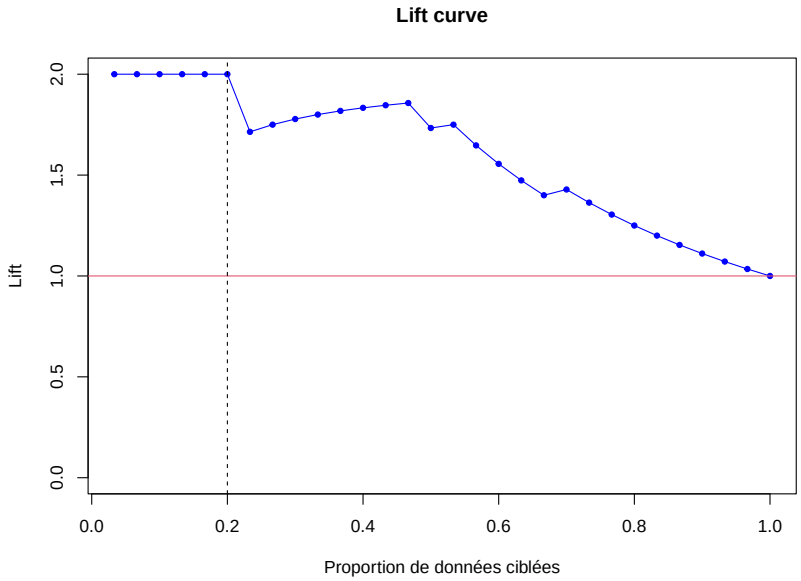
Exercice 4. Les données de l'exemple du cours sur le publipostage sont dans le fichier `ex_publipostage.txt`.

```
tab <- read.table("../data/ex_publipostage.txt", header = TRUE, sep="")
```

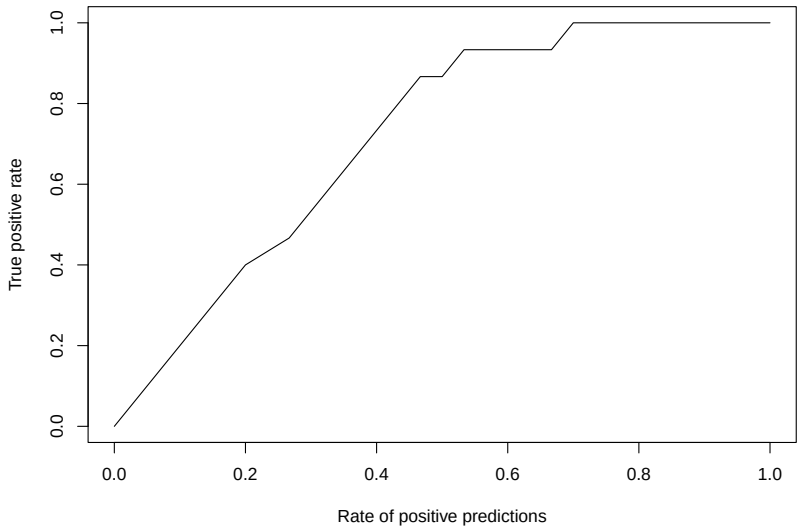
1. A partir de ces données reproduire les graphiques du cours représentant la courbe des gains cumulés et la courbe Lift :

   (a) en calculant vous même les coordonnées des points et en utilisant ensuite la fonction `plot`,

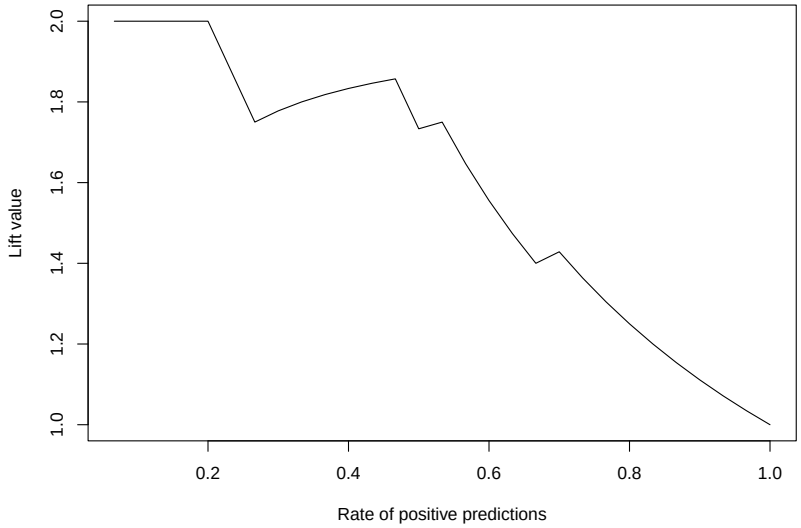




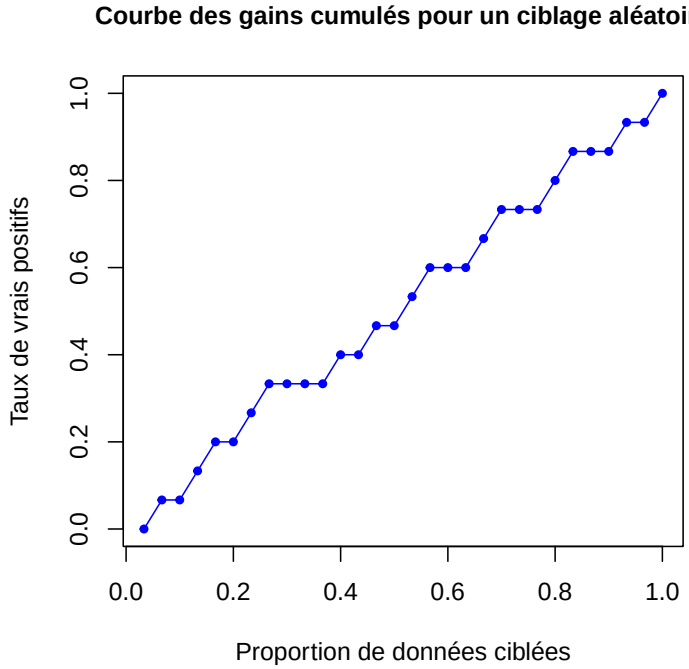
(b) en utilisant le package `ROCR` [1].



1. `https://cran.r-project.org/web/packages/ROCR/ROCR.pdf`



2. Tracer la courbe des gains cumulés correspondant à un ciblage aléatoire (données non triées).

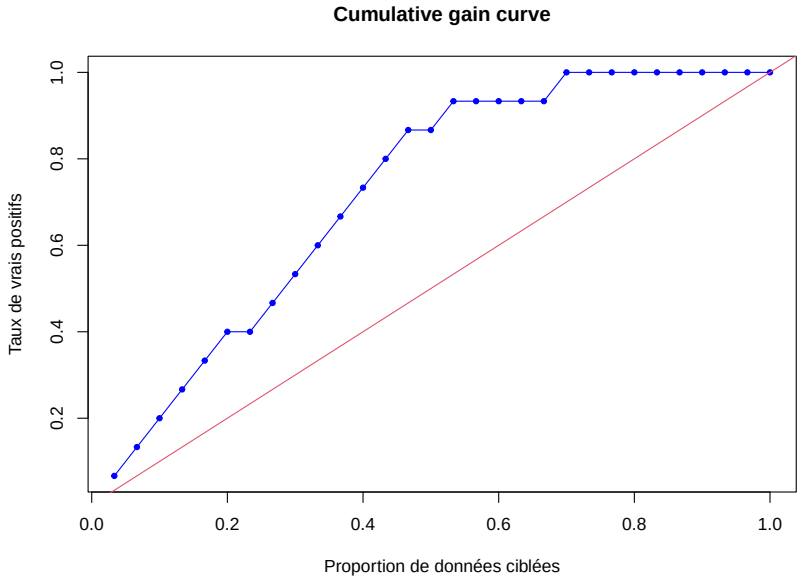


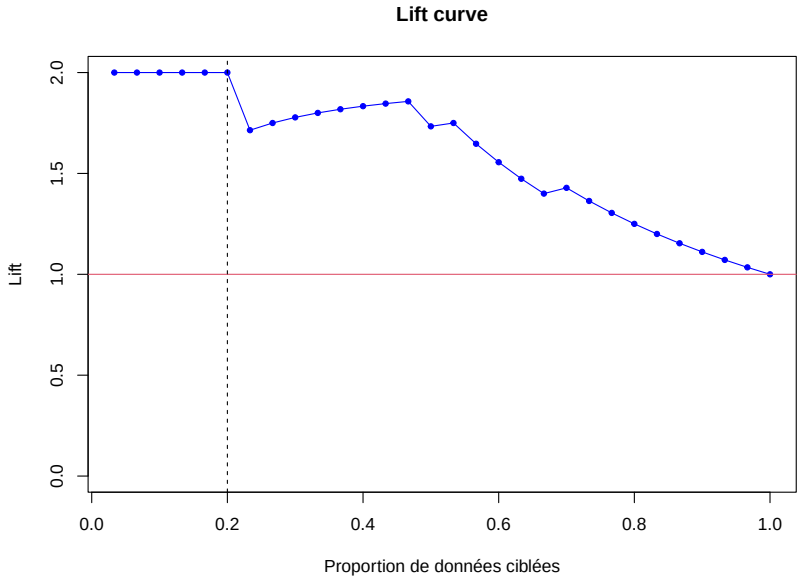
Exercice 4. Les données de l'exemple du cours sur le publipostage sont dans le fichier `ex_publipostage.txt`.

```
tab <- read.table("../data/ex_publipostage.txt", header = TRUE, sep="")
```

1. A partir de ces données reproduire les graphiques du cours représentant la courbe des gains cumulés et la courbe Lift :

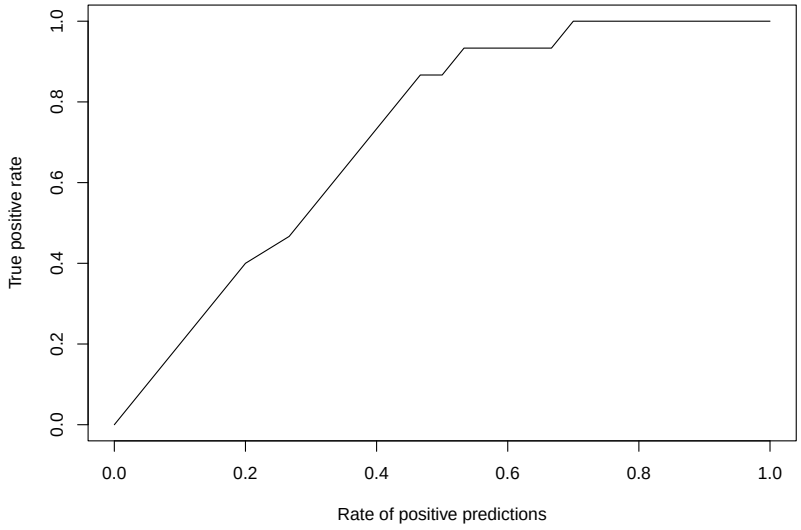
(a) en calculant vous même les coordonnées des points et en utilisant ensuite la fonction `plot`,

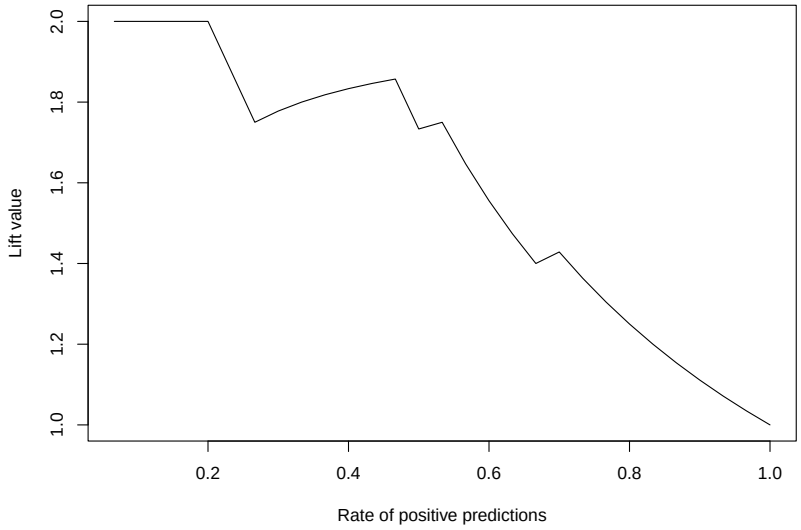




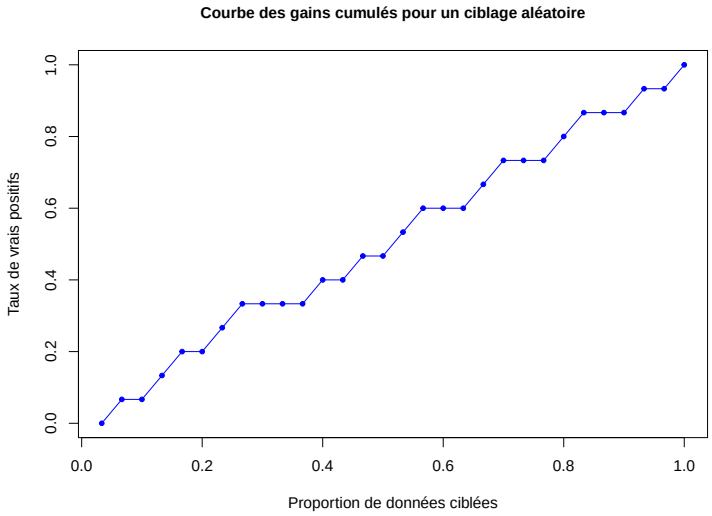
(b) en utilisant le package `ROCR` [2].

2. `https://cran.r-project.org/web/packages/ROCR/ROCR.pdf`





2. Tracer la courbe des gains cumulés correspondant à un ciblage aléatoire (données non triées).



Exercice 5. Les données proviennent de la compétition `CoIL Challenge 2000`[3] dont l'objectif était de repérer parmi les clients d'une compagnie d'assurance, ceux susceptibles de contracter une police d'assurance pour leur caravane.

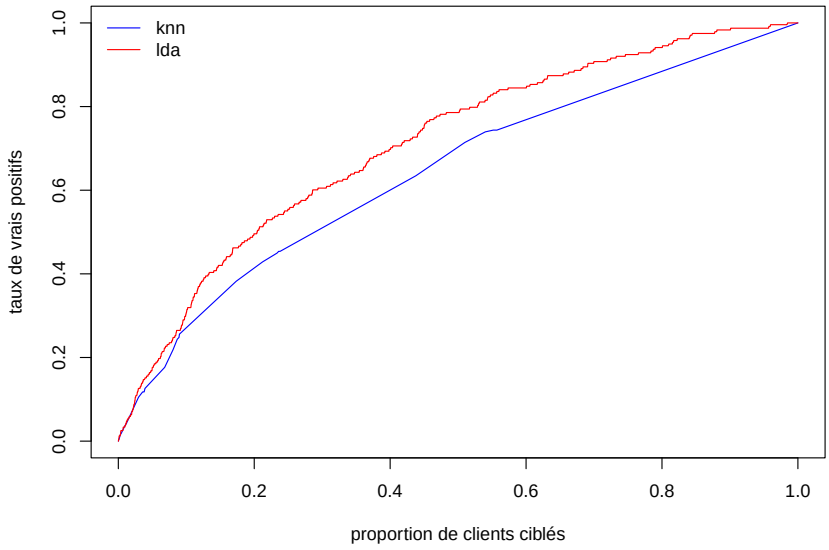
Le fichier ticdata.xls[4] contient 9822 observations (clients) dont 5822 pour l'apprentissage et 4000 pour l'évaluation. La variable STATUS permet de les distinguer. Outre la variable cible (prendre ou pas une police d'assurance pour sa caravane), les clients sont décrits dans ce fichier sur 85 variables dont 43 décrivent son environnement socio-économique et les 42 suivantes décrivent son comportement avec d'autres produits.

```
dat <- read.table("../data/ticdata.csv", header = TRUE, sep=";")
```

```
Xtrain <- dat[which(dat$STATUS=="Learning"),1:85]
Ytrain <- dat[which(dat$STATUS=="Learning"),86]
Xtest <- dat[which(dat$STATUS=="Test"),1:85]
Ytest <- dat[which(dat$STATUS=="Test"),86]
```

1. Calculer le score (probabilités à posteriori) des données test avec :
   — la méthode `knn` avec k choisi par validation croisée LOO dans la grille de valeurs suivantes :
   ```
   kmax <- 20
   grille_k <- seq(from=1, to=kmax, by=2)
   ```
   Que pensez-vous de cette grille ? Quelle est la valeur optimale de k dans cette grille ?
   — la méthode `lda`.
2. Construite la courbe Lift (version gains cumulés) des scores obtenus avec les deux méthodes.

3. `http://www.liacs.nl/~putten/library/cc2000/report2.html`
4. `http://eric.univ-lyon2.fr/~ricco/tanagra/fr/tanagra.html`



Quelle méthode est la plus performante et pourquoi ?

3. Avec cette méthode,
   - si on cible 20% des clients ayant le meilleur score , quelle est la part de client appétents dans la cible (vous pouvez utiliser la fonction `approxfun`) ?
   - Si on cible ainsi 20% des client de la base des données test, quel est le nombre de clients ayant effectovement acheté une assurance qui sont ciblés ?
   - Si on cible maintenant une nouvelle base de 200000 clients avec cette méthode de scoring, quel devrait être le nombre de clients parmi ceux ciblés qui vont acheter une assurance pour leur caravane ?
   - Si à l'inverse, on veut vendre 5000 assurances, quelle doit être la taille de la cible ?