Statistical learning project

MASSIMBO D N Damien _ TRABELSI Sarra

17/09/2020

Contents

Chargement des librairies

```
library(caret)
library(pROC)
library(ggplot2)
```

Exercice 1 :Algorithme K-nn et validation croisée

1. Simulation d'un échantillon de taille n = 200 de notre modèle stocké dans dans un dataframe df

```
set.seed(12345)
n=200
x=runif(n, min=-1,max=1)
y=sin(pi*x)+rnorm(n,sd=0.5)
df=data.frame(x=x,y=y)
head(df)
```

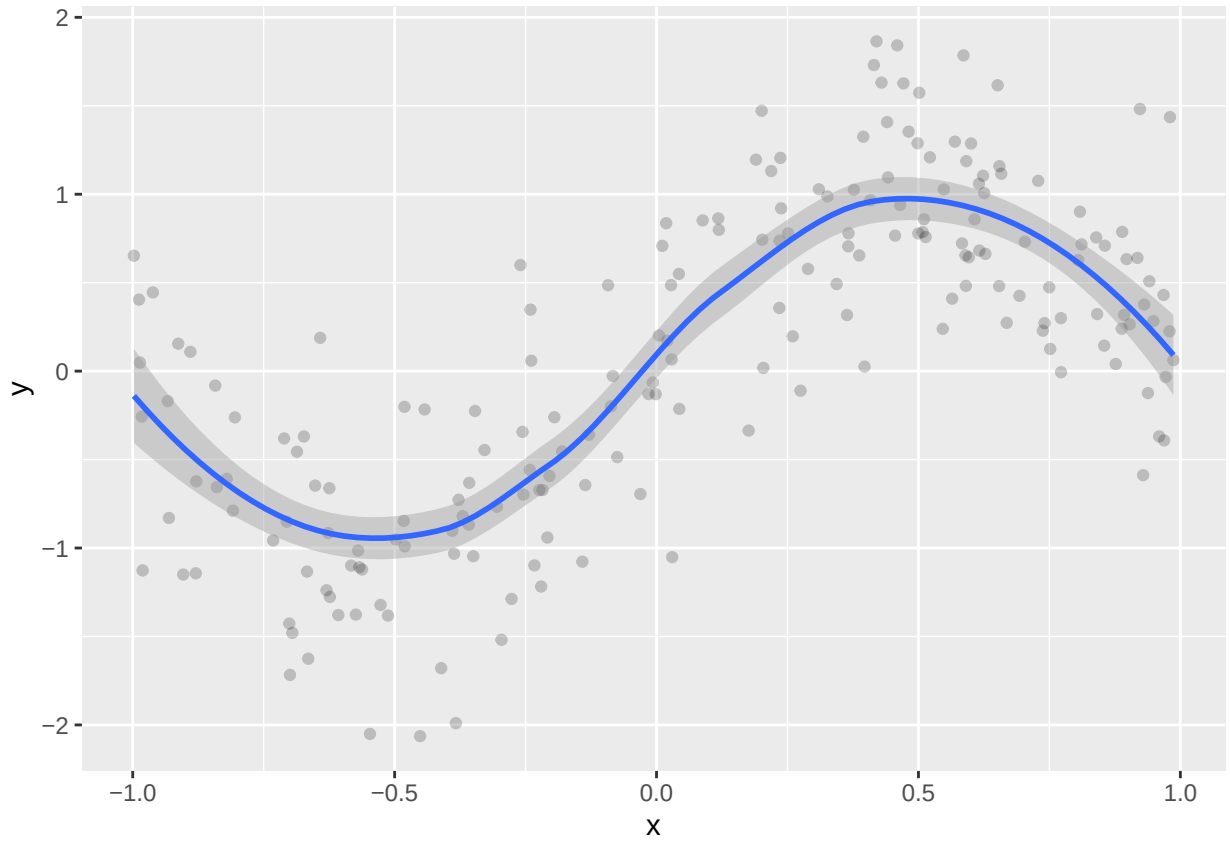
```
##             x            y
## 1  0.44180779  1.095298311
## 2  0.75154639  0.125551579
## 3  0.52196466  1.208829432
## 4  0.77224913 -0.006382405
## 5 -0.08703808 -0.199501326
## 6 -0.66725643 -1.133121519
```

Nous fixons la graine pour obtenir les mêmes résultats à chaque nouvelle compilation du programme R. Ensuite, on implémente la loi de X et de Y en simulant un échantillon de taille n = 200 que l'on stock dans un dataframe appélé df.

2. Représentation du nuage de points et la fonction de régression et 3. Creation de la fonction eta_est (x, k, data) calculant la valeur en x de l'estimateur calculé avec k sur les données data

L'erreur de Bayes est la variance de epsilon egale à 0.25.

```
x_vec=seq(-1,1,0.01)
eta=sin(pi*x_vec)
ggplot(df, aes(x , y )) +
  geom_point(alpha = 0.2)+
        geom_smooth()
```



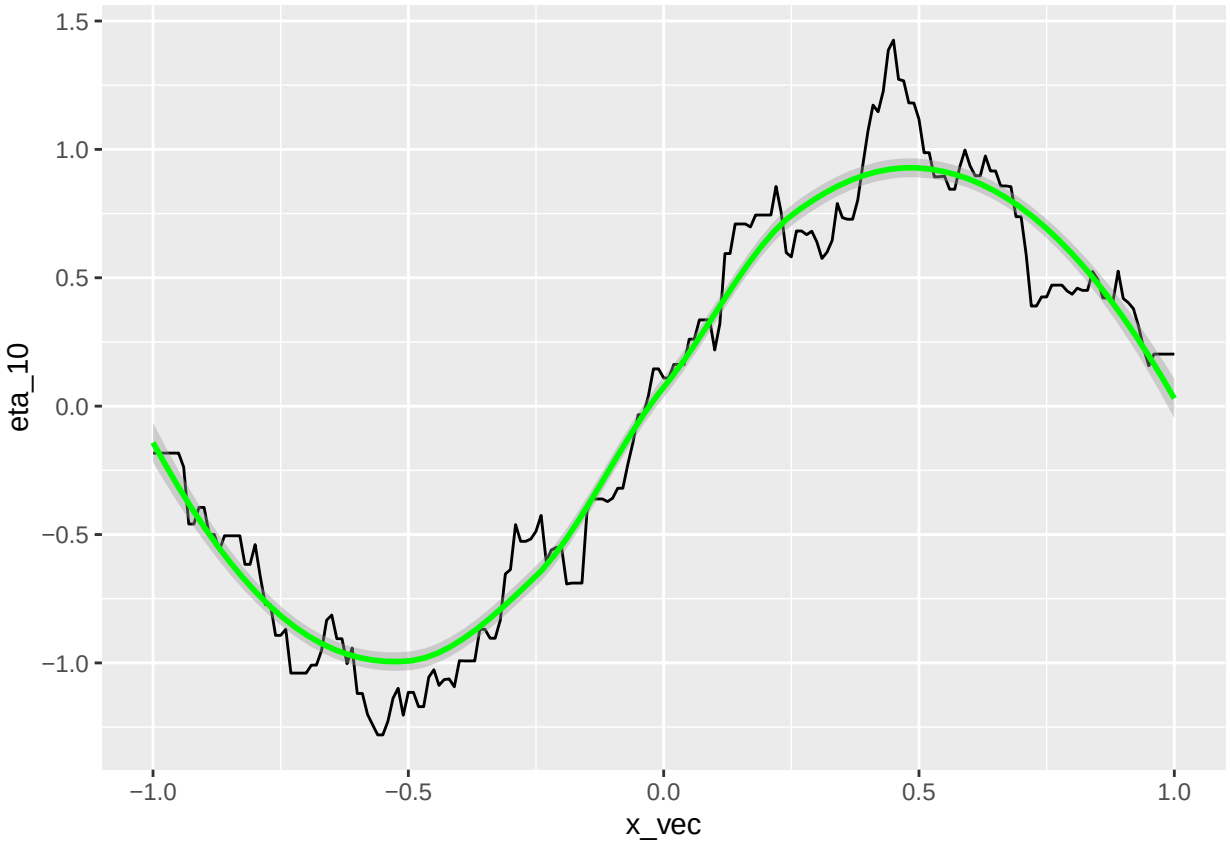

```
##creation de la fonction eta_est(x, k, data)
eta_est=function(x, k, data){

  X=data$x
  dist=abs(X-x)
  index=sort(dist, index.return=T)$ix
  knn=index[1:k]
  Y=data$y
  return(mean(Y[knn]))

}

#test
eta_est(0,k=10,data=df)
```

```
## [1] 0.1094922
```

Nous calculons la valeur de cette fonction au point x = 0, k = 10 en utilisant notre base de donnée initiale (df). Nous obtenons la valeur obtenue ci_dessus.

4. Donnons la répresentation graphique de la fonction et 5. Créons une fonction err_train(k,data) calculant l'erreur quadratique sur l'echantillon d'apprentissage

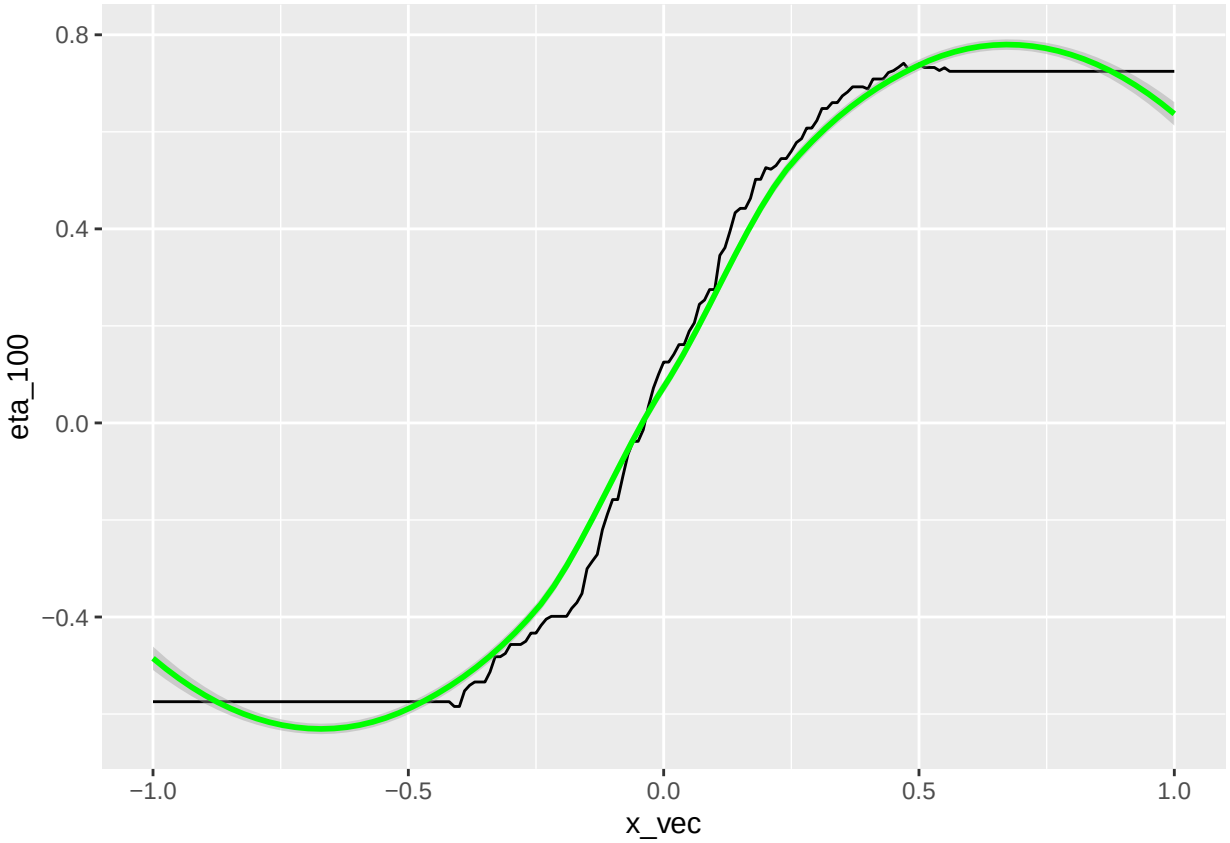
```
eta_10=sapply(x_vec,eta_est,k=10, data=df)
eta_10 <- data.frame(eta_10)
#colnames(eta_est)=c("x_vec")
#colnames(eta_est)=c("eta_10")

# pour k = 10
ggplot(eta_10, aes(x_vec, eta_10)) +
  geom_line(aes(x_vec, eta_10 )) +
        geom_smooth(colour="green")
```



On constate que le k joue un role d'un paramètre de lissage ou de regularisation, k= 10, j'ai beaucoup peu de bruit. Nous sommes dans le sous apprentissage lorsque k est grand versus sur apprentissage lorsque k est petit.

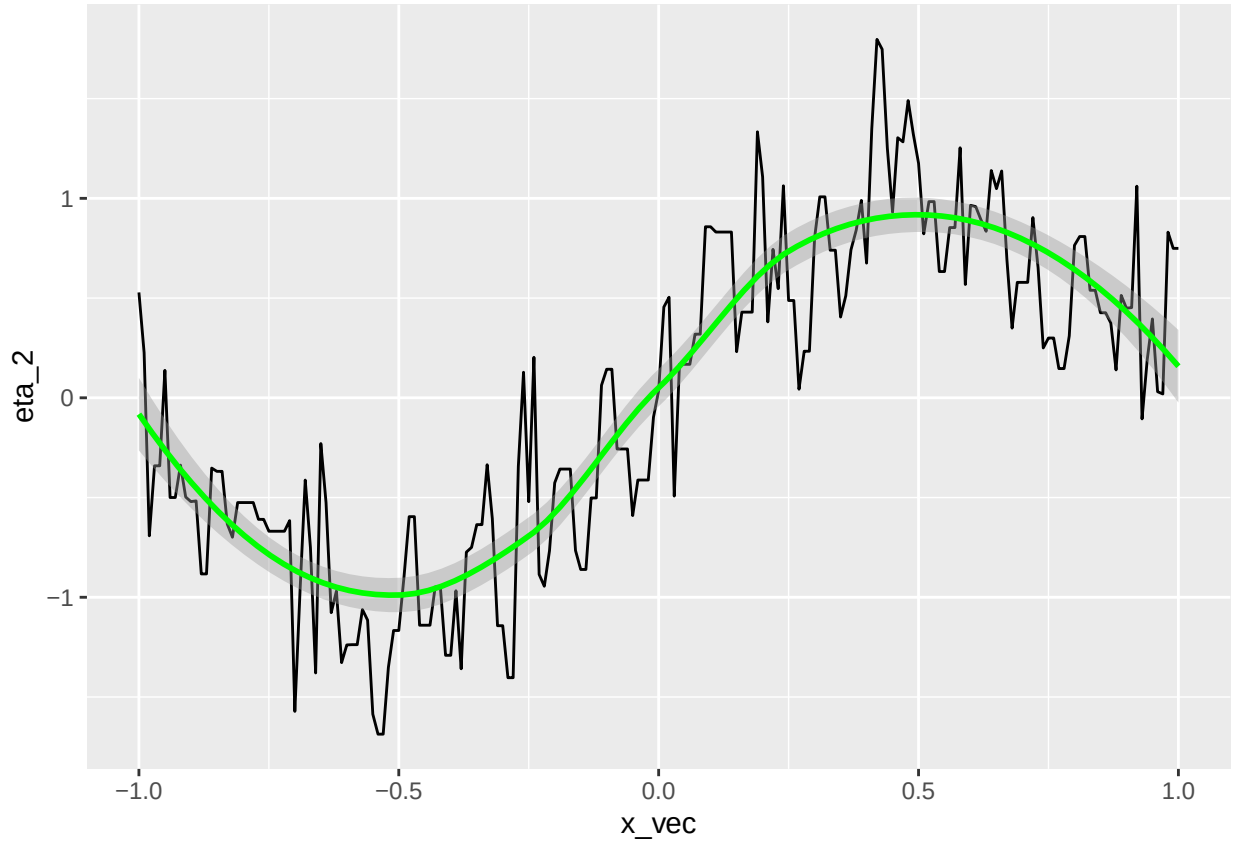
Représentation graphique pour k =100

```
eta_100=sapply(x_vec,eta_est,k=100, data=df)
eta_100 <- data.frame(eta_100)
ggplot(eta_100, aes(x_vec, eta_100)) +
  geom_line(aes(x_vec, eta_100 )) +
        geom_smooth(colour="green")
```



Pour K = 100, on constate qu'on a peu de bruit. **Représentation graphique pour k = 2**

```
eta_2 =sapply(x_vec,eta_est,k=2, data=df)
eta_2 <- data.frame(eta_2)
ggplot(eta_2, aes(x_vec, eta_2)) +
  geom_line(aes(x_vec, eta_2)) +
        geom_smooth(colour="green")
```



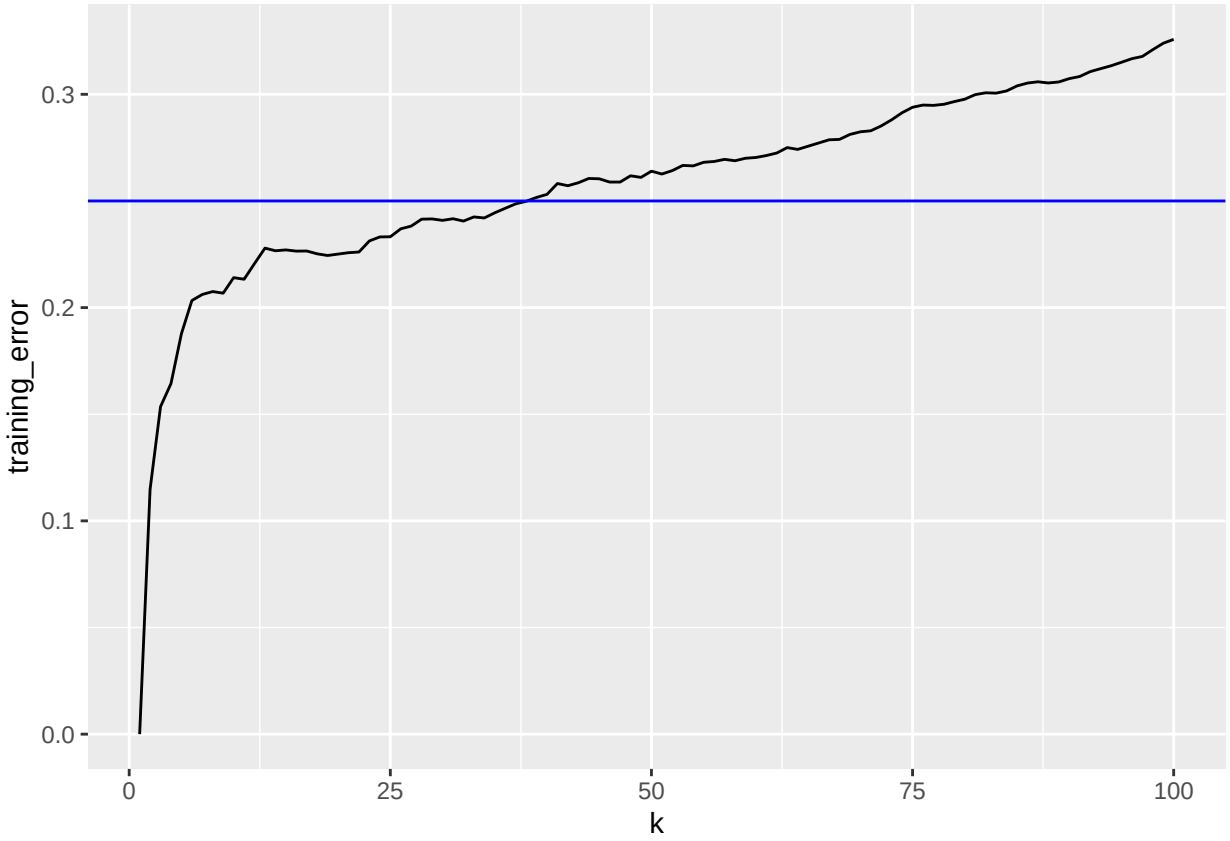
On constate que le k joue un role d'un paramètre de lissage ou de regularisation, k= 2, dans ce cas, j'ai beaucoup de bruit.

```
err_train =function(k, data){
  y_pred=sapply(df$x, eta_est, k,data=df)
  err =mean((y_pred-df$y)^2)
  return(err)

}
#test
err_train(k=10, data=df)
```

```
## [1] 0.2140397
```

Nous calculons la valeur de la fonction de l'erreur quadratique sur l'échantillon d'apprentissage au point k = 10 en utilisant notre base de donnée initiale (df). Nous obtenons la valeur obtenue ci_dessus. **Représentatiopn graphique de la fonction pour k variant de 1 à 50**

```
err_train_plot<- sapply(1:100,err_train, data=df)
err_train_plot <- data.frame(err_train_plot)
ggplot(err_train_plot,aes(x, err_train_plot))+
geom_line(aes(1:100, err_train_plot))+
geom_hline(yintercept= 0.25,col = "blue") +
  xlab("k") +
  ylab("training_error")
```



On remarque qu'une erreur plus petite que l'erreur de bayes suggere un sur_apprentissage.

6. Création d'un vecteur group correspondant à une division en K = 5 groupes de l'échantillon et 7. Superposition des courbes err_train, err_test pour 1,..50.

```
set.seed(12345)
group <- sample(rep(1:5,each=40), 200)
group[1:10]
```

```
##  [1] 4 2 4 2 3 2 3 1 3 5
```

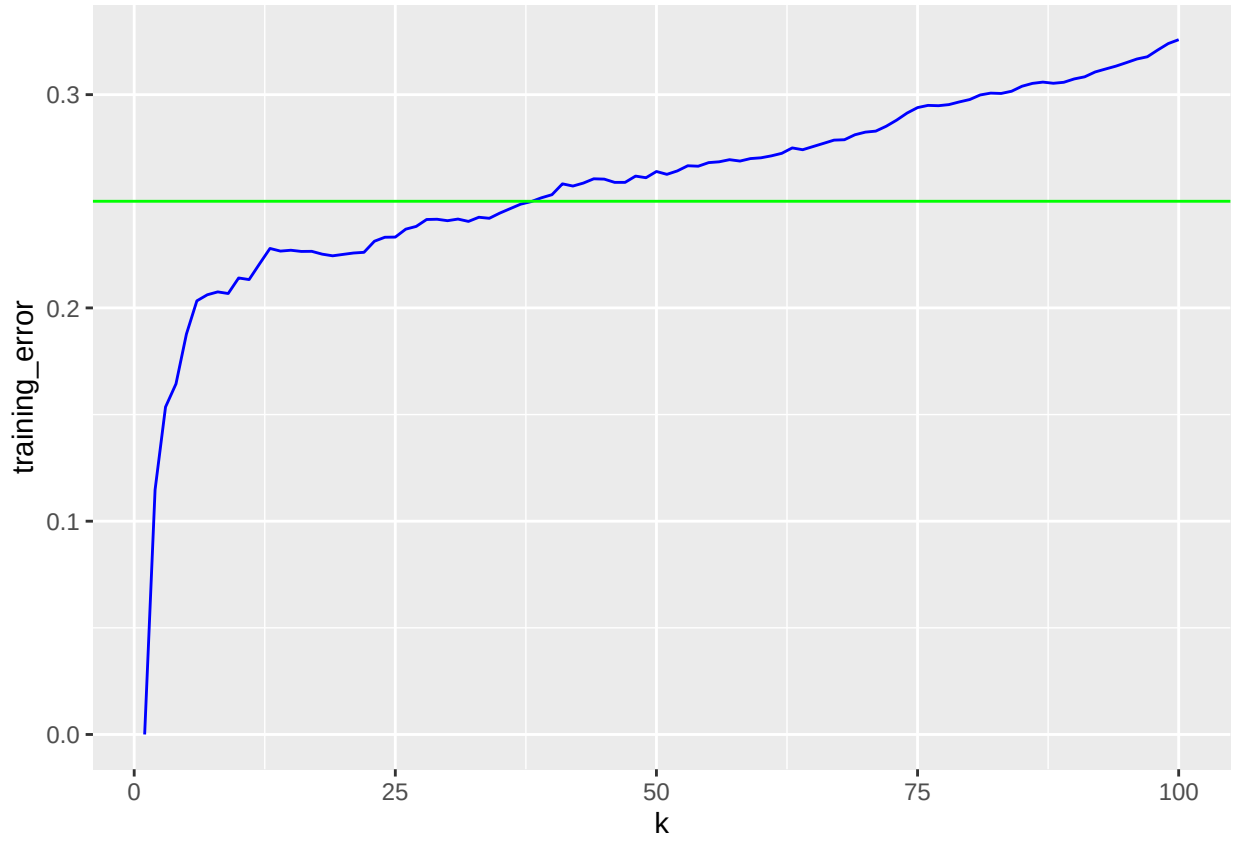
Erreur test err_test(k, data, group) calculée par la validation croisée On initialise aussi les predictions à 0, et y_pred predit sur la couche j, data frame qui exclut la couche j. Ci joint le programme R ci_dessous :

```
err_test <- function(k){
y_pred=rep(0,n)
for (j in 1:5) {

x_test <- df$x[group==j]
y_pred_test <- sapply(x_test, eta_est, k, data=df[group!=j,])
y_pred[group==j] <- y_pred_test
}
return(mean((df$y-y_pred)^2))
```

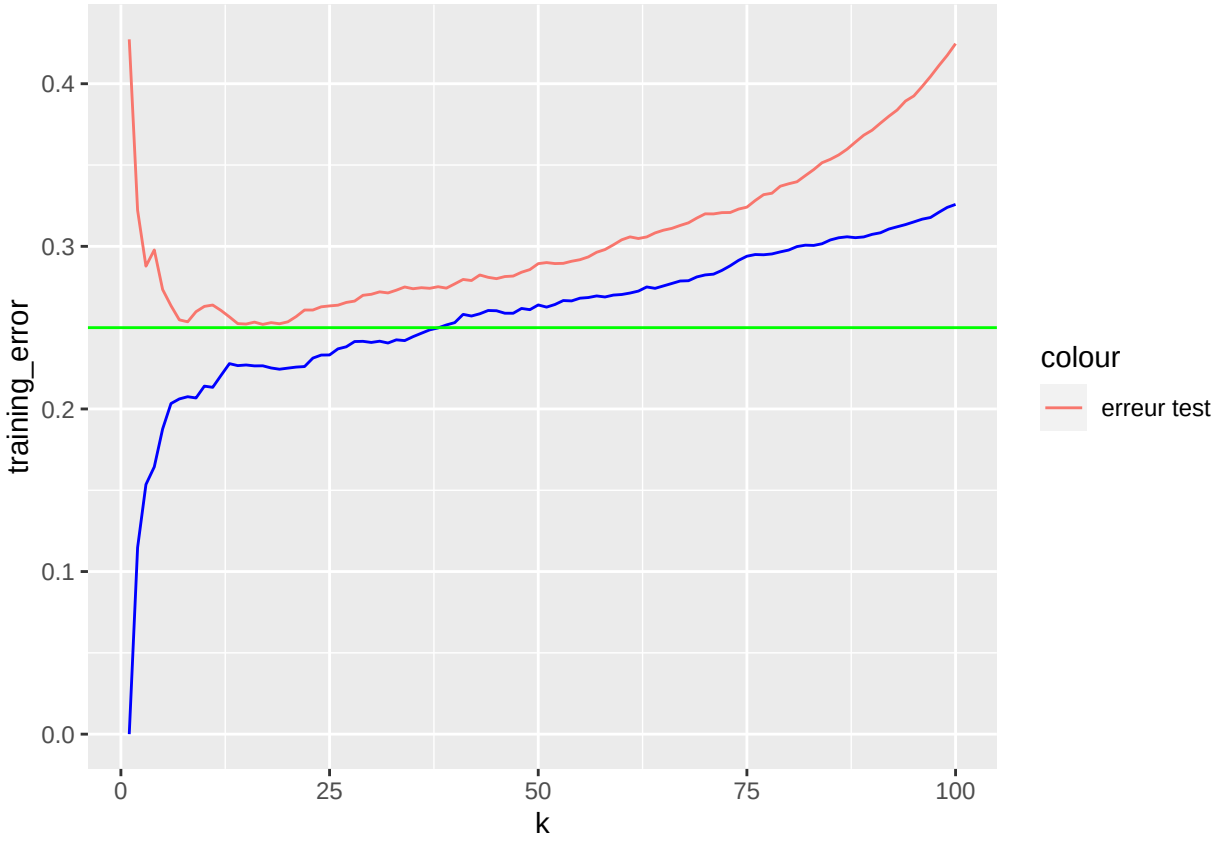
```
}
err_test(10)
```

```
## [1] 0.2630709
```

```
err_train_plot<- sapply(1:100,err_train, data=df)
err_train_plot <- data.frame(err_train_plot)
gg=ggplot(err_train_plot,aes(x, err_train_plot))+
geom_line(aes(1:100, err_train_plot),color="blue")+
geom_hline(yintercept= 0.25,col = "green") +
  xlab("k") +
  ylab("training_error")
gg
```




```
err_test_plot<- sapply(1:100, err_test)
err_test_plot<- data.frame(err_test_plot)
gg+geom_line(aes(x=1:100,y=err_test_plot$err_test_plot,color="erreur test"))
```



8. Estimation de l'erreur de prediction associée.

```
kopt=17
eta_opt <- sapply(x_vec, eta_est, k=kopt, data=df)

err_test_opt <- min(err_test_plot)
#kopt <- which.min(err_test_plot)
c(err_test_opt=err_test_opt,kopt=kopt)
```

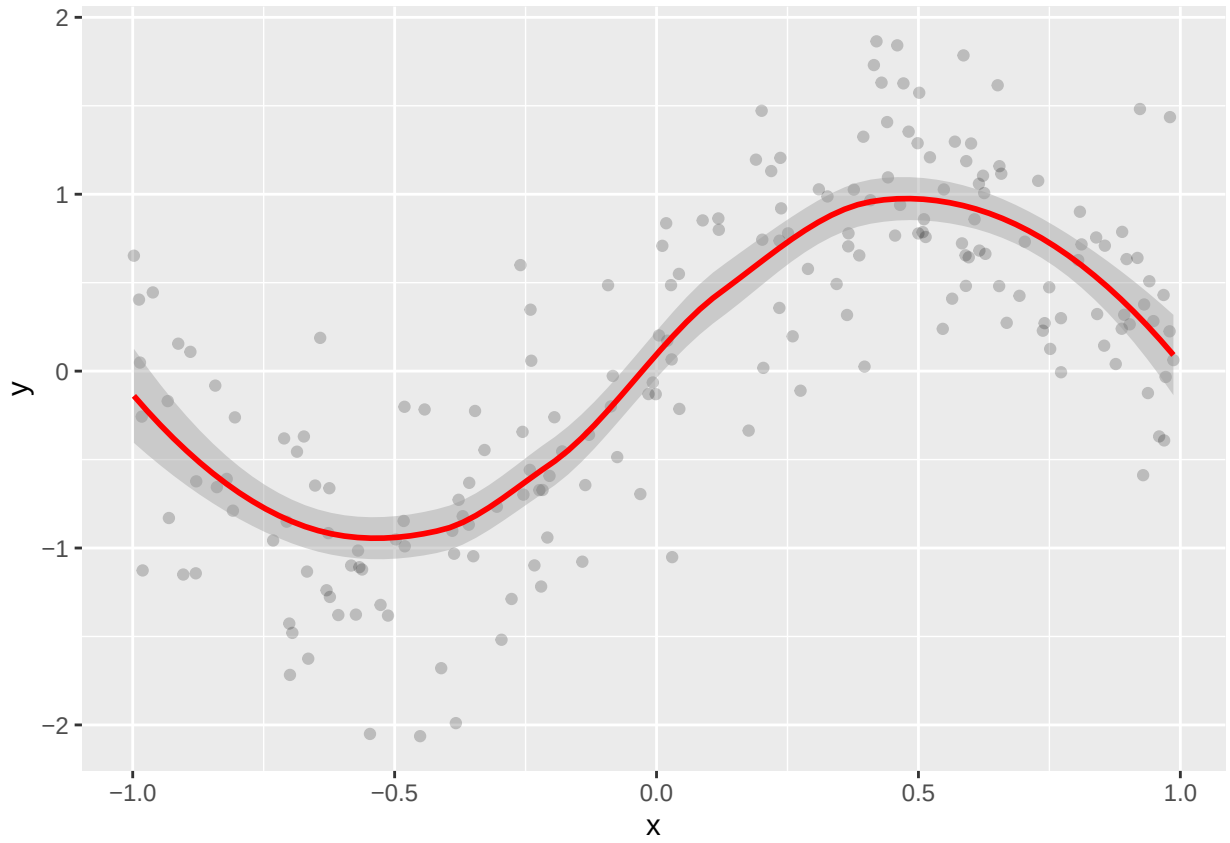
```
## err_test_opt         kopt
##    0.2520036   17.0000000
```

```
eta_opt <- sapply(x_vec, eta_est, k=kopt, data=df)
```

Le K optimale obtenue est de 17 et l'erreur test optimale est de 0.250036.

Représentation du predicteur correspondant évalué sur l'échantillon complet

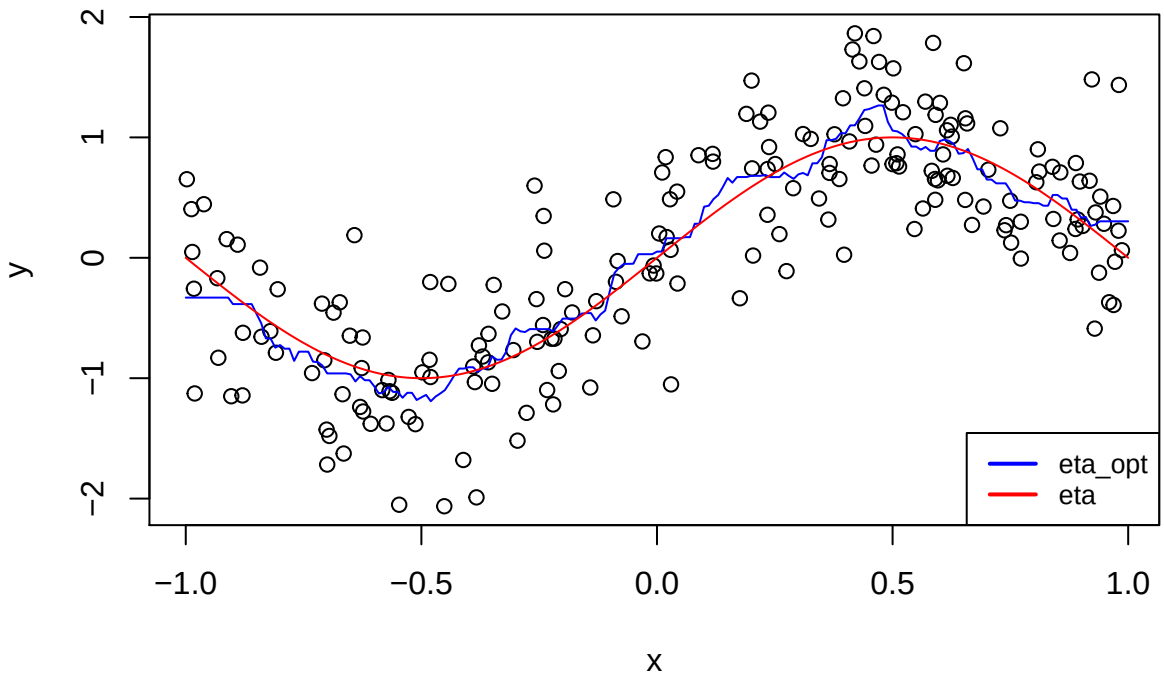
```
x_vec=seq(-1,1,0.01)
eta=sin(pi*x_vec)
tt=ggplot(df, aes(x, y )) +
geom_point(alpha = 0.2)+
geom_smooth(color="red")
tt
```




```
eta_opt<-sapply(x_vec, eta_est, k=kopt, data=df)
eta_opt<-data.frame(eta_opt)
eta_opt=df


err_test_opt <- min(err_test_plot)
#kopt <- which.min(err_test_plot)
c(err_test_opt=err_test_opt,k_opt=kopt)
```

```
## err_test_opt        k_opt
##    0.2520036   17.0000000
```

```
eta_opt <- sapply(x_vec,eta_est,k=kopt,data=df)
plot(x,y)
lines(x_vec,eta_opt,type='l',col='blue')
lines(x_vec,eta,col='red')

legend("bottomright", legend = c("eta_opt","eta"), col=c("blue","red"),
lty=1, cex=0.8, text.font=1,lwd=2) # petite légende
```




```
#tt+geom_line(aes(x=x_vec,y=df$eta_opt),color="blue")
```

La courbe en rouge correspond au prédicteur de Bayes et celle en bleue correspond au predicteur des plus proches voisins.

9. Nous reprenons les deux questions précédentes grâce au package caret

```
ctrl<-trainControl(method = "repeatedcv", number = 5, repeats = 5)
param=expand.grid(k=seq(1,100))
fit_knn <- train(x~y, data=df, method='knn', trControl=ctrl, tuneLength=50, tuneGrid=param)
fit_knn
```

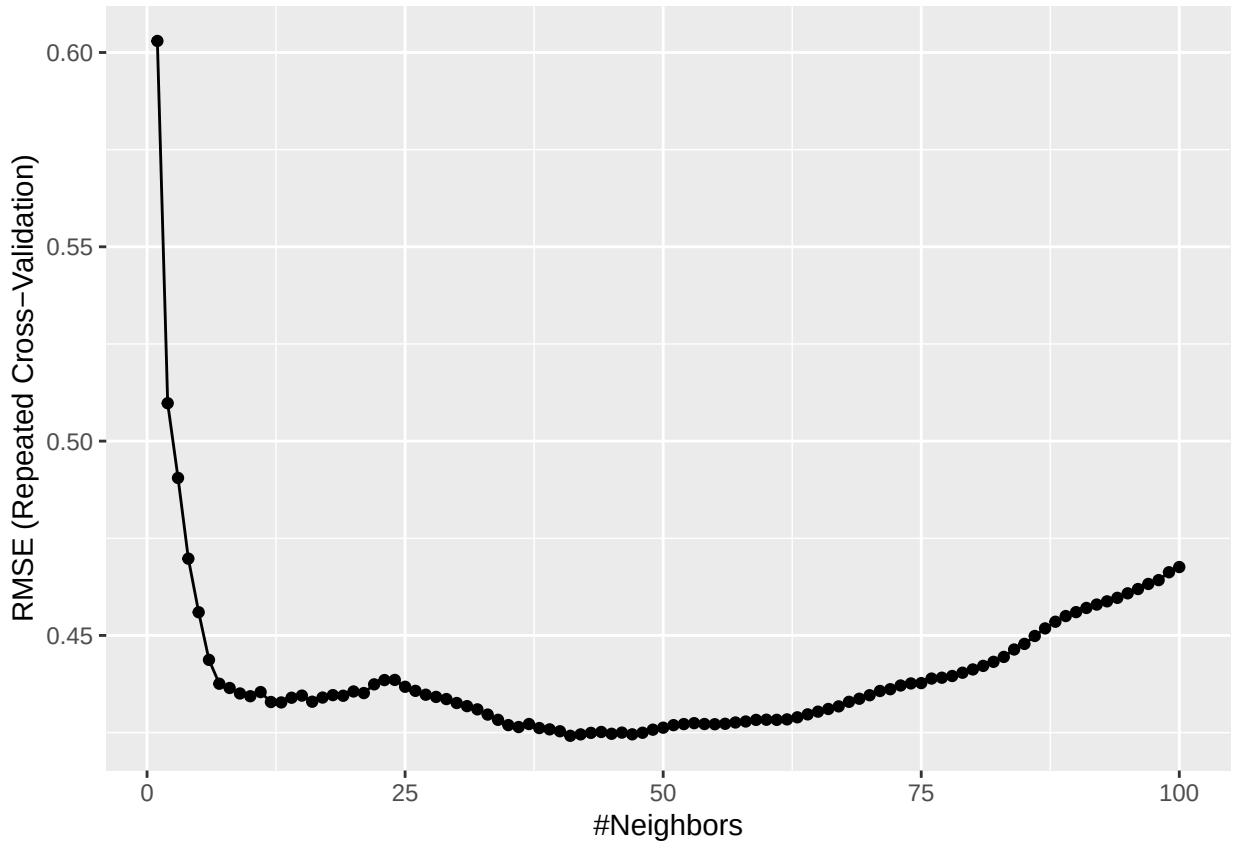
```
## k-Nearest Neighbors
##
## 200 samples
##   1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 160, 160, 160, 160, 160, 160, ...
## Resampling results across tuning parameters:
##
##   k    RMSE       Rsquared   MAE
```

```
##      1  0.6029628  0.2398038  0.4311217
##      2  0.5097537  0.3518056  0.3768070
##      3  0.4905297  0.3703536  0.3530397
##      4  0.4697593  0.4032208  0.3357603
##      5  0.4559685  0.4272063  0.3261784
##      6  0.4437141  0.4520675  0.3207859
##      7  0.4375716  0.4629510  0.3185216
##      8  0.4364770  0.4649844  0.3182409
##      9  0.4350667  0.4666792  0.3185451
##     10  0.4343594  0.4678105  0.3185853
##     11  0.4354214  0.4650270  0.3185296
##     12  0.4328680  0.4700326  0.3169020
##     13  0.4327613  0.4699858  0.3156392
##     14  0.4339658  0.4672220  0.3164665
##     15  0.4345183  0.4660877  0.3160735
##     16  0.4329450  0.4697795  0.3148516
##     17  0.4340161  0.4681107  0.3145112
##     18  0.4346417  0.4667623  0.3145304
##     19  0.4344765  0.4672907  0.3142883
##     20  0.4355812  0.4651494  0.3143286
##     21  0.4351697  0.4665122  0.3132479
##     22  0.4374155  0.4609761  0.3148836
##     23  0.4385115  0.4581687  0.3155296
##     24  0.4385549  0.4579461  0.3160077
##     25  0.4367900  0.4619140  0.3149646
##     26  0.4357413  0.4642497  0.3142655
##     27  0.4347458  0.4668142  0.3136357
##     28  0.4341785  0.4677228  0.3129618
##     29  0.4336228  0.4692700  0.3125803
##     30  0.4326079  0.4712234  0.3119905
##     31  0.4318089  0.4729237  0.3119880
##     32  0.4309749  0.4745446  0.3116752
##     33  0.4296296  0.4778888  0.3109221
##     34  0.4282861  0.4810394  0.3098985
##     35  0.4269290  0.4845093  0.3091993
##     36  0.4264284  0.4858774  0.3086760
##     37  0.4272027  0.4839104  0.3099059
##     38  0.4261648  0.4867378  0.3095259
##     39  0.4258320  0.4877796  0.3092041
##     40  0.4253430  0.4892447  0.3090338
##     41  0.4241797  0.4924123  0.3082501
##     42  0.4245404  0.4917490  0.3088836
##     43  0.4249282  0.4913059  0.3090923
##     44  0.4251568  0.4910894  0.3098676
##     45  0.4246942  0.4928091  0.3095239
##     46  0.4249959  0.4924005  0.3097156
##     47  0.4245572  0.4938867  0.3098552
##     48  0.4249570  0.4939636  0.3106235
##     49  0.4257485  0.4925106  0.3113408
##     50  0.4262626  0.4914912  0.3117316
##     51  0.4269140  0.4904046  0.3120921
##     52  0.4271883  0.4902426  0.3125013
##     53  0.4274291  0.4908649  0.3129653
##     54  0.4272009  0.4916540  0.3128338
```

```
##     55  0.4271526  0.4922675  0.3127651
##     56  0.4272646  0.4925993  0.3127798
##     57  0.4275993  0.4916531  0.3127635
##     58  0.4278531  0.4917999  0.3127116
##     59  0.4282663  0.4908372  0.3131140
##     60  0.4283079  0.4913341  0.3134280
##     61  0.4282810  0.4921527  0.3131985
##     62  0.4283962  0.4928386  0.3133533
##     63  0.4289347  0.4923823  0.3145475
##     64  0.4296849  0.4919468  0.3154196
##     65  0.4303841  0.4918777  0.3164780
##     66  0.4310753  0.4916102  0.3176159
##     67  0.4317456  0.4913067  0.3186265
##     68  0.4329162  0.4903866  0.3199071
##     69  0.4337072  0.4908102  0.3212696
##     70  0.4346161  0.4906861  0.3225863
##     71  0.4357080  0.4900066  0.3240296
##     72  0.4361655  0.4898718  0.3249456
##     73  0.4371184  0.4891050  0.3260700
##     74  0.4376466  0.4889683  0.3270572
##     75  0.4377332  0.4895856  0.3275233
##     76  0.4389006  0.4888665  0.3288272
##     77  0.4391253  0.4893762  0.3290317
##     78  0.4395608  0.4892535  0.3294806
##     79  0.4404036  0.4889707  0.3306289
##     80  0.4412468  0.4895798  0.3320688
##     81  0.4421626  0.4893016  0.3331282
##     82  0.4432082  0.4905582  0.3348804
##     83  0.4444720  0.4916030  0.3371381
##     84  0.4463830  0.4916418  0.3398090
##     85  0.4478205  0.4917994  0.3416982
##     86  0.4498581  0.4914596  0.3443266
##     87  0.4518118  0.4907678  0.3468185
##     88  0.4535268  0.4907016  0.3490978
##     89  0.4549862  0.4915774  0.3510143
##     90  0.4559942  0.4926759  0.3526718
##     91  0.4570742  0.4934645  0.3541769
##     92  0.4579463  0.4938258  0.3556442
##     93  0.4587724  0.4953839  0.3569130
##     94  0.4596742  0.4960511  0.3583545
##     95  0.4608353  0.4964126  0.3599964
##     96  0.4619348  0.4968096  0.3616131
##     97  0.4632483  0.4959130  0.3636116
##     98  0.4642364  0.4970269  0.3652389
##     99  0.4662540  0.4959257  0.3679861
##    100  0.4675990  0.4965273  0.3700936
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 41.
```

Graphique Obtenue

```
ggplot(fit_knn, color ="blue")
```



Exercice 2

Préparation des données

Le jeu de données wine contient 12 variables portant sur la qualité de 4898 vins blancs. Nous avons 11 variables quantitatives:

- fixed.acidity
- volatile.acidity
- citric.acid
- residual.sugar
- chlorides
- free.sulfur.dioxide
- total.sulfur.dioxide
- density
- pH
- sulphates
- alcohol

et une variable qualitative quality.

```
library(caret)
library(pROC)
setwd('D:/TP APPRENTISSAGE')
wine=read.csv('D:/TP APPRENTISSAGE/winequality-white.csv',sep=";",header=T)
```

```
wine$quality <- wine$quality
summary(wine)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol         quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
```

```
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

Pour commencer, on va recoder la variable **quality** en facteurs à 2 classes

```
wine$quality <- as.factor(ifelse(wine$quality>=6,'Good','Bad'))
summary(wine$quality)
```

```
##  Bad Good
## 1640 3258
```

```
summary(wine)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol      quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Bad :1640
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   Good:3258
##  Median :3.180   Median :0.4700   Median :10.40
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20
```

Régression Logistique

Nous proposons ici d'ajuster un modèle de régression logistique dans le but de prédire la variable quality en fonction des autres covariables. En effet, la régression logistique a pour objectif d'expliquer et de prédire les valeurs d'une variable qualitative, le plus souvent binaire, à partir de variables explicatives qualitatives et quantitatives.

Question 2)a

```
set.seed(12345)
ctrl <- trainControl(method = "cv", number = 5)
model_glm <- train(quality~., data=wine, method='glm',trControl=ctrl)
print(model_glm)
```

```
## Generalized Linear Model
##
## 4898 samples
##   11 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
## Resampling results:
##
##   Accuracy   Kappa
##   0.7488787  0.3963273
```

Question 2)b

L'option summaryFunction=twoClassSummary calculera les mesures spécifiques aux problèmes à deux classes, telles que l'aire sous la courbe ROC, la sensibilité et la spécificité.

```
set.seed(12345)
ctrl <- trainControl(method='cv', number = 5,
classProbs=T,savePredictions = T,
summaryFunction=twoClassSummary)
model_glm <- train(quality~., data=wine, method='glm', trControl=ctrl)
print(model_glm)
```

```
## Generalized Linear Model
##
## 4898 samples
##   11 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8000755  0.4920732  0.8781469
```

Question 2)c

```
head(model_glm$pred)
```

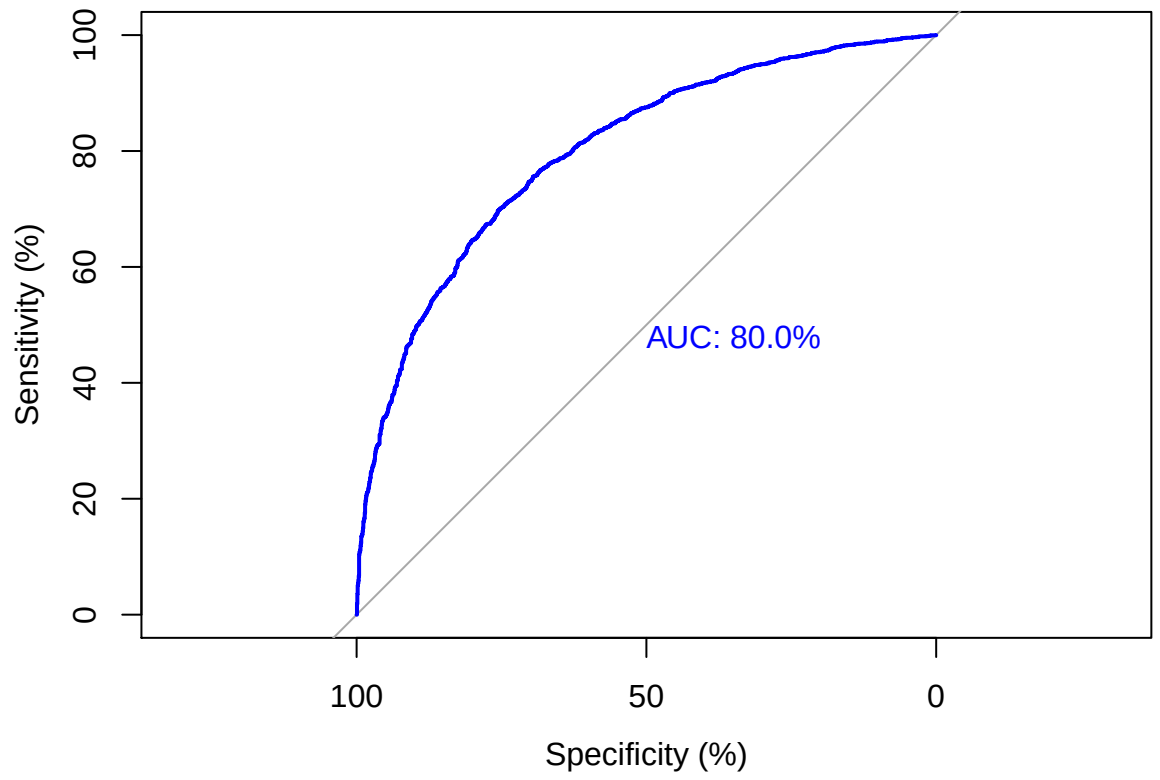
```
##   pred  obs        Bad      Good rowIndex parameter Resample
## 1 Good Good 0.34679684 0.6532032        5      none    Fold1
## 2 Good Good 0.47174470 0.5282553        8      none    Fold1
## 3 Good  Bad 0.13542864 0.8645714       11      none    Fold1
## 4 Good Good 0.02709463 0.9729054       14      none    Fold1
## 5 Good Good 0.34874982 0.6512502       19      none    Fold1
## 6  Bad  Bad 0.56810476 0.4318952       20      none    Fold1
```

Le data frame contient les prédictions des données d'entraînement et de test et les probabilités de classe.

Question 2)d

La courbe ROC ci dessus ou encore courbe de sensibilité/spécificité nous offre une visualisation du taux de vrais positifs (proportion des positifs "Good") en fonction du taux de faux positifs (proportion des négatifs "Bad" qui sont mal détectés). L'analyse de cette courbe révèle une assez bonne sensibilité dans la capacité prédictive de notre modèle en validation croisée à 5 couches. Autrement dit, on prédit un peu près correctement la qualité du vin.On observe une faible probabilité de la mauvaise qualité du vin.Notre AUC est assez bon donc nos prédictions sont correctes.

```
roc1=plot.roc(model_glm$pred$obs,model_glm$pred$Good, percent=T, print.auc=T,col="blue")
```



Question 3)a

Maintenant, effectuons par validation croisée à 5 couches l'analyse discriminante linéaire quadratique.

```
set.seed(12345)
ctrl <- trainControl(method = "cv", number = 5)
model_glm <- train(quality~., data=wine, method='lda',trControl=ctrl)
print(model_glm)
```

```
## Linear Discriminant Analysis
##
## 4898 samples
##   11 predictor
```

```
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
## Resampling results:
##
##   Accuracy   Kappa
##   0.7496956  0.3962131
```

Question 3)b

```
set.seed(12345)
ctrl <- trainControl(method='cv', number = 5,
classProbs=T, savePredictions = T,
summaryFunction=twoClassSummary)
model_glm <- train(quality~., data=wine, method='lda', trControl=ctrl)
print(model_glm)
```

```
## Linear Discriminant Analysis
##
## 4898 samples
##   11 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8002229  0.4865854  0.8821374
```

L'option summaryFunction=twoClassSummary calculera les mesures spécifiques aux problèmes à deux classes, telles que l'aire sous la courbe ROC, la sensibilité et la spécificité.

Question 3)c

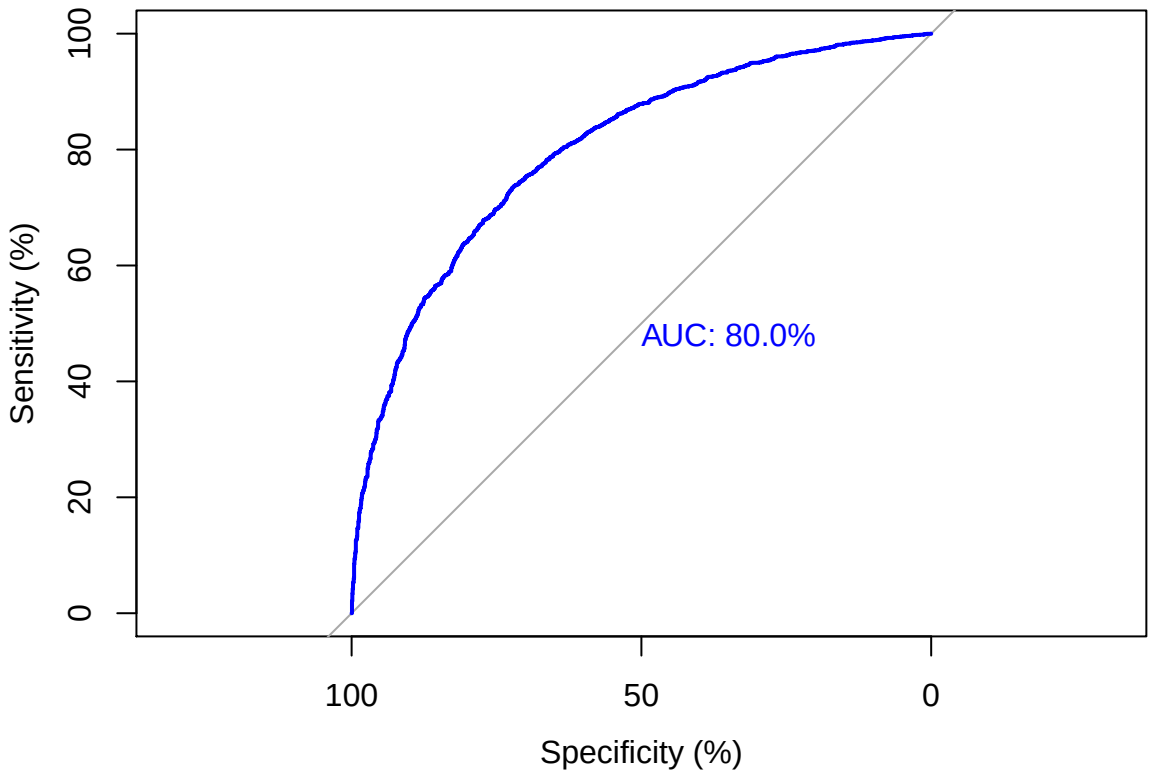
```
head(model_glm$pred)
```

```
##   pred  obs        Bad      Good rowIndex parameter Resample
## 1 Good Good 0.33742667 0.6625733        5      none    Fold1
## 2 Good Good 0.45780291 0.5421971        8      none    Fold1
## 3 Good  Bad 0.13658857 0.8634114       11      none    Fold1
## 4 Good Good 0.02656538 0.9734346       14      none    Fold1
## 5 Good Good 0.37124762 0.6287524       19      none    Fold1
## 6  Bad  Bad 0.55482375 0.4451763       20      none    Fold1
```

Le data frame contient les prédictions des données d'entraînement et de test et les probabilités de classe.

Question 3)d

```
roc2=plot.roc(model_glm$pred$obs,model_glm$pred$Good, percent=T, print.auc=T,col="blue")
```



La courbe ROC notifie qu'on prédit toujours correctement la qualité de vin.

Question 4)a

Maintenant, effectuons par validation croisée à 5 couches l'analyse discriminante linéaire.

```
set.seed(12345)
ctrl <- trainControl(method = "cv", number = 5)
model_glm <- train(quality~., data=wine, method='qda',trControl=ctrl)
print(model_glm)
```

```
## Quadratic Discriminant Analysis
##
## 4898 samples
##   11 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
```

```
## Resampling results:
##
##   Accuracy   Kappa
##   0.7431607  0.3956302
```

Question 4)b

```
set.seed(12345)
ctrl <- trainControl(method='cv', number = 5,
classProbs=T, savePredictions = T,
summaryFunction=twoClassSummary)
model_glm <- train(quality~., data=wine, method='qda', trControl=ctrl)
print(model_glm)
```

```
## Quadratic Discriminant Analysis
##
## 4898 samples
##   11 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
## Resampling results:
##
##   ROC       Sens       Spec
##   0.795627  0.5243902  0.8532852
```

L'option summaryFunction=twoClassSummary calculera les mesures spécifiques aux problèmes à deux classes, telles que l'aire sous la courbe ROC, la sensibilité et la spécificité.

Question 4)c

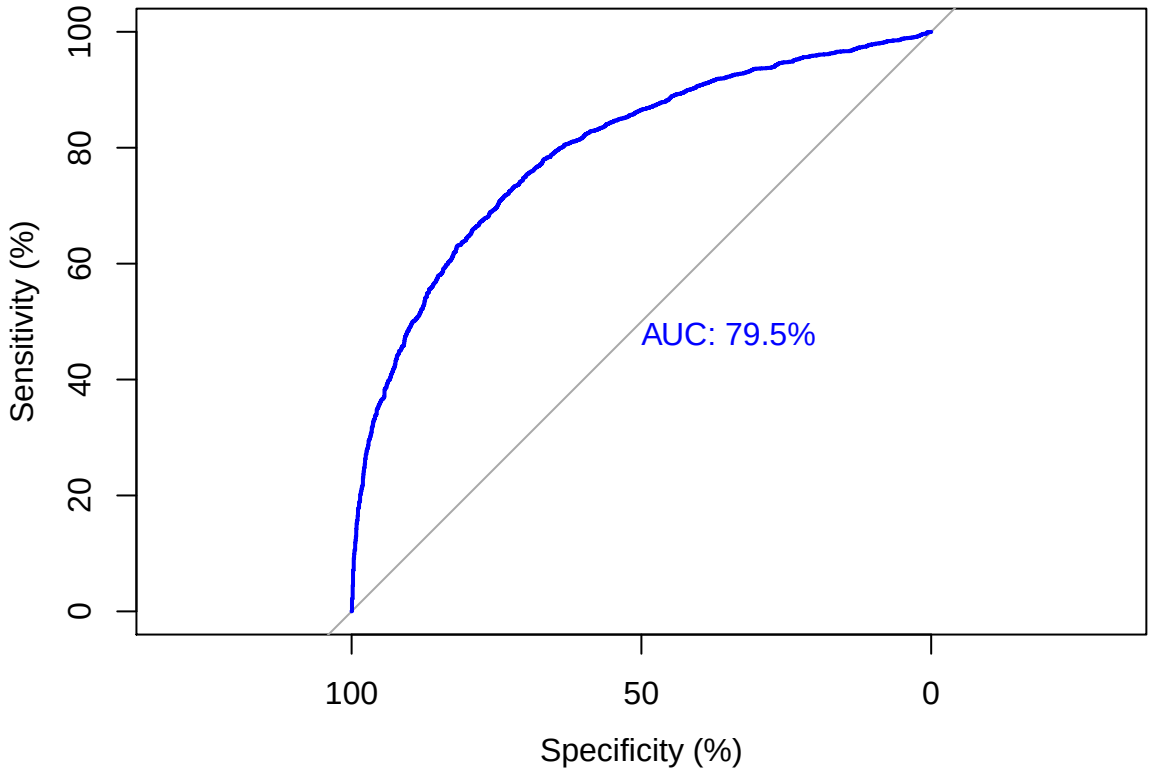
Le data frame contient les prédictions des données d'entraînement et de test et les probabilités de classe.

```
head(model_glm$pred)
```

```
##   pred  obs         Bad      Good rowIndex parameter Resample
## 1 Good Good 0.217619496 0.7823805        5      none    Fold1
## 2 Good Good 0.157319126 0.8426809        8      none    Fold1
## 3 Good  Bad 0.069007536 0.9309925       11      none    Fold1
## 4 Good Good 0.007106977 0.9928930       14      none    Fold1
## 5 Good Good 0.318235142 0.6817649       19      none    Fold1
## 6 Good  Bad 0.493108667 0.5068913       20      none    Fold1
```

Question 4)d

```
roc3=plot.roc(model_glm$pred$obs,model_glm$pred$Good, percent=T, print.auc=T,col="blue")
```



La courbe ROC notifie qu'on prédit toujours correctement la qualité de vin.

Question 5)a

```
set.seed(12345)
ctrl <- trainControl(method = "cv", number = 5)
k=seq(5,100,by=5)
K=data.frame(k)
model_knn <- train(quality~., data=wine, method="knn",
trControl=ctrl,
preProcess=c("center","scale"),
tuneGrid=K)
print(model_knn)
```

```
## k-Nearest Neighbors
##
## 4898 samples
##   11 predictor
##    2 classes: 'Bad', 'Good'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
## Resampling results across tuning parameters:
##
```

```
##   k    Accuracy   Kappa
##     5  0.7584768  0.4424203
##    10  0.7643960  0.4497044
##    15  0.7641923  0.4467835
##    20  0.7678677  0.4518645
##    25  0.7625588  0.4356366
##    30  0.7645999  0.4396458
##    35  0.7637819  0.4361381
##    40  0.7631690  0.4354942
##    45  0.7621490  0.4325138
##    50  0.7617398  0.4302505
##    55  0.7623560  0.4295301
##    60  0.7613333  0.4266110
##    65  0.7603131  0.4251555
##    70  0.7599041  0.4223056
##    75  0.7568421  0.4145115
##    80  0.7554129  0.4106186
##    85  0.7539847  0.4062665
##    90  0.7521455  0.4020882
##    95  0.7527573  0.4037119
##   100  0.7527562  0.4021412
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 20.
```

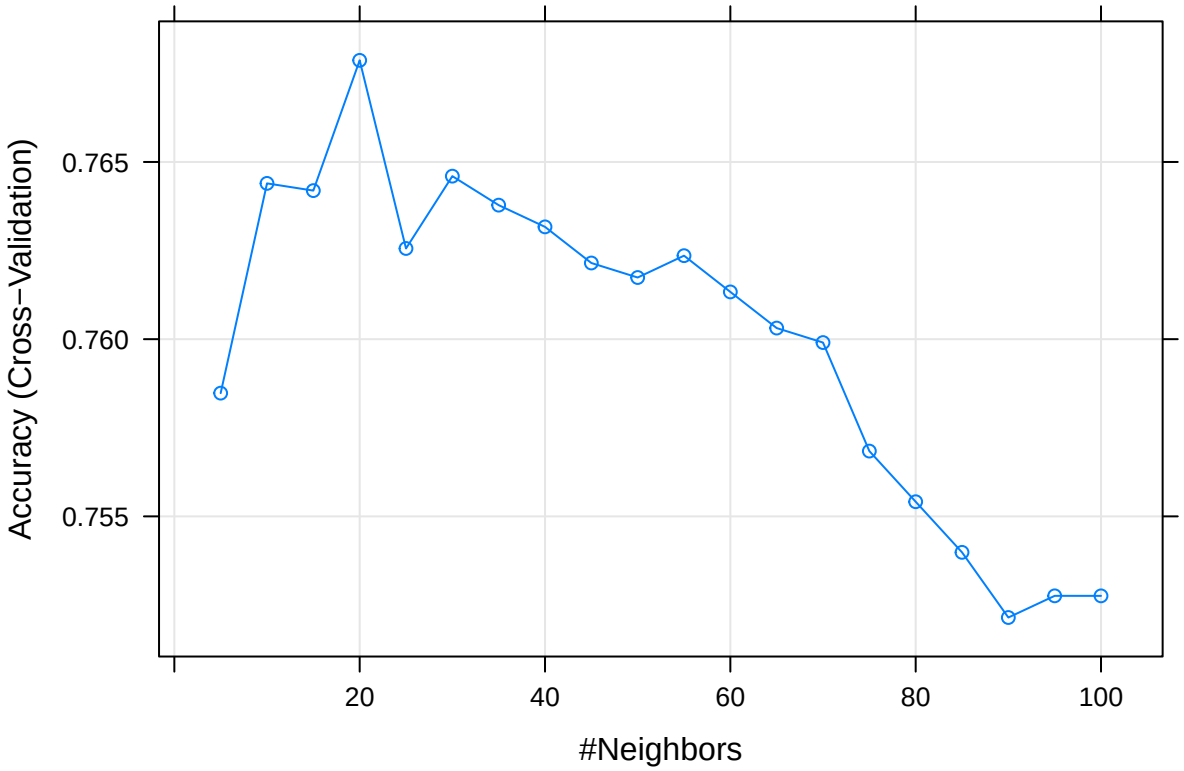
Pour chaque valeur de k dans tuneGrid le taux de bien classés (Accuracy) et le Kappa sont calculés.L'accucary est maximale pour k=20,avec cette technique nous choisirons donc la règle des 20 plus proches voisins,comme indiquée à la fin de la sortie.On peut récupérer cette valeur:

```
model_knn$bestTune
```

```
##    k
## 4 20
```

Un plot appliqué à l'objet construit permet de visualiser la précision (Accuracy) en fonction de k.

```
plot(model_knn)
```



Question 5)b

```
set.seed(12345)
k=seq(5,100,by=5)
K=data.frame(k)
ctrl <- trainControl(method='cv', number = 5,
classProbs=T, savePredictions = T,
summaryFunction=twoClassSummary)
model_knn <- train(quality~., data=wine, method="knn",trControl=ctrl,preProcess=c("center","scale"),tun
print(model_knn)
```

```
## k-Nearest Neighbors
##
## 4898 samples
##   11 predictor
##    2 classes: 'Bad', 'Good'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3919, 3918, 3918, 3919, 3918
## Resampling results across tuning parameters:
##
##   k    ROC        Sens       Spec
##     5  0.8112807  0.5841463  0.8462295
```

```
##    10  0.8188020  0.5701220  0.8621875
##    15  0.8185817  0.5609756  0.8664876
##    20  0.8231352  0.5542683  0.8753880
##    25  0.8231309  0.5335366  0.8778448
##    30  0.8206448  0.5335366  0.8809133
##    35  0.8210911  0.5274390  0.8827523
##    40  0.8188084  0.5292683  0.8809095
##    45  0.8172502  0.5262195  0.8809104
##    50  0.8179557  0.5213415  0.8827509
##    55  0.8180067  0.5146341  0.8870558
##    60  0.8176399  0.5115854  0.8870539
##    65  0.8171195  0.5134146  0.8845990
##    70  0.8153333  0.5067073  0.8873597
##    75  0.8141488  0.5006098  0.8858269
##    80  0.8132648  0.4969512  0.8855202
##    85  0.8127595  0.4920732  0.8858269
##    90  0.8116808  0.4902439  0.8839831
##    95  0.8111079  0.4914634  0.8842899
##   100  0.8106655  0.4865854  0.8867434
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 20.
```

L'option summaryFunction=twoClassSummary calculera les mesures spécifiques aux problèmes à deux classes, telles que l'aire sous la courbe ROC, la sensibilité et la spécificité.

Question 5)c

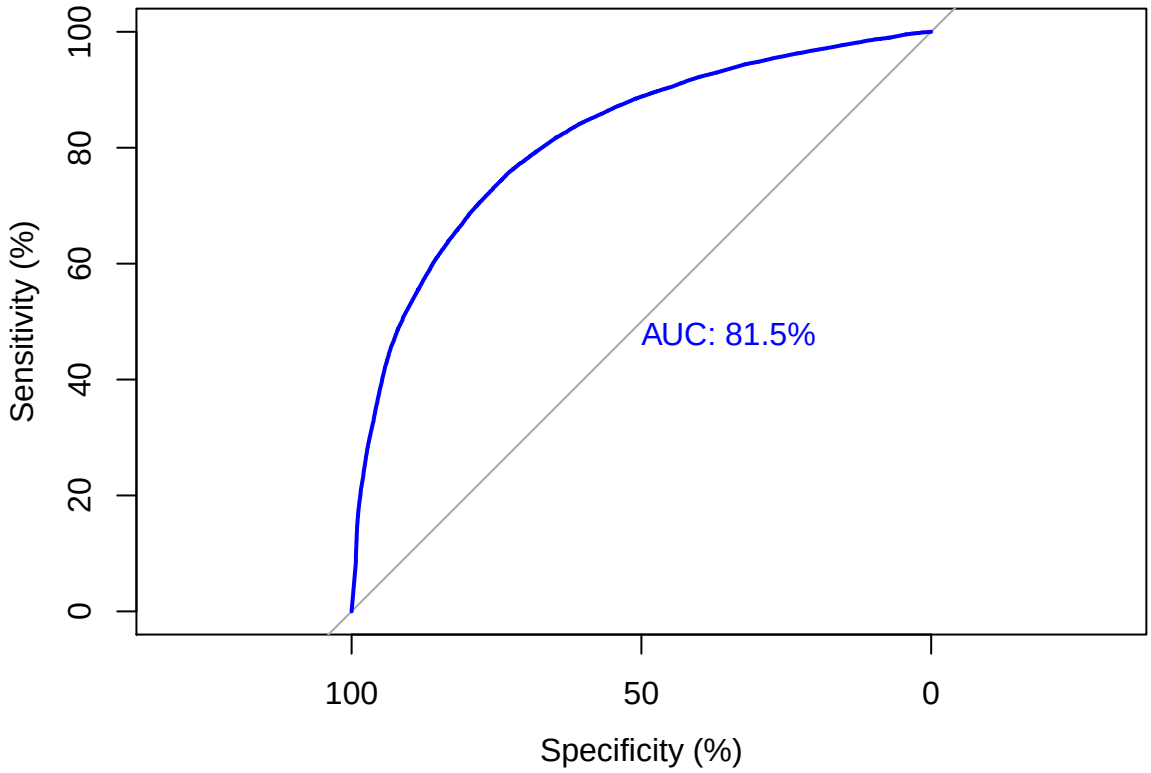
Le data frame contient les prédictions des données d'entraînement et de test et les probabilités de classe.

```
head(model_knn$pred)
```

```
##   pred  obs       Bad      Good rowIndex k Resample
## 1  Bad Good 0.6000000 0.4000000        5 5    Fold1
## 2  Bad Good 0.5000000 0.5000000        8 5    Fold1
## 3 Good  Bad 0.2000000 0.8000000       11 5    Fold1
## 4 Good Good 0.3333333 0.6666667       14 5    Fold1
## 5 Good Good 0.2000000 0.8000000       19 5    Fold1
## 6  Bad  Bad 0.8333333 0.1666667       20 5    Fold1
```

Question 5)d

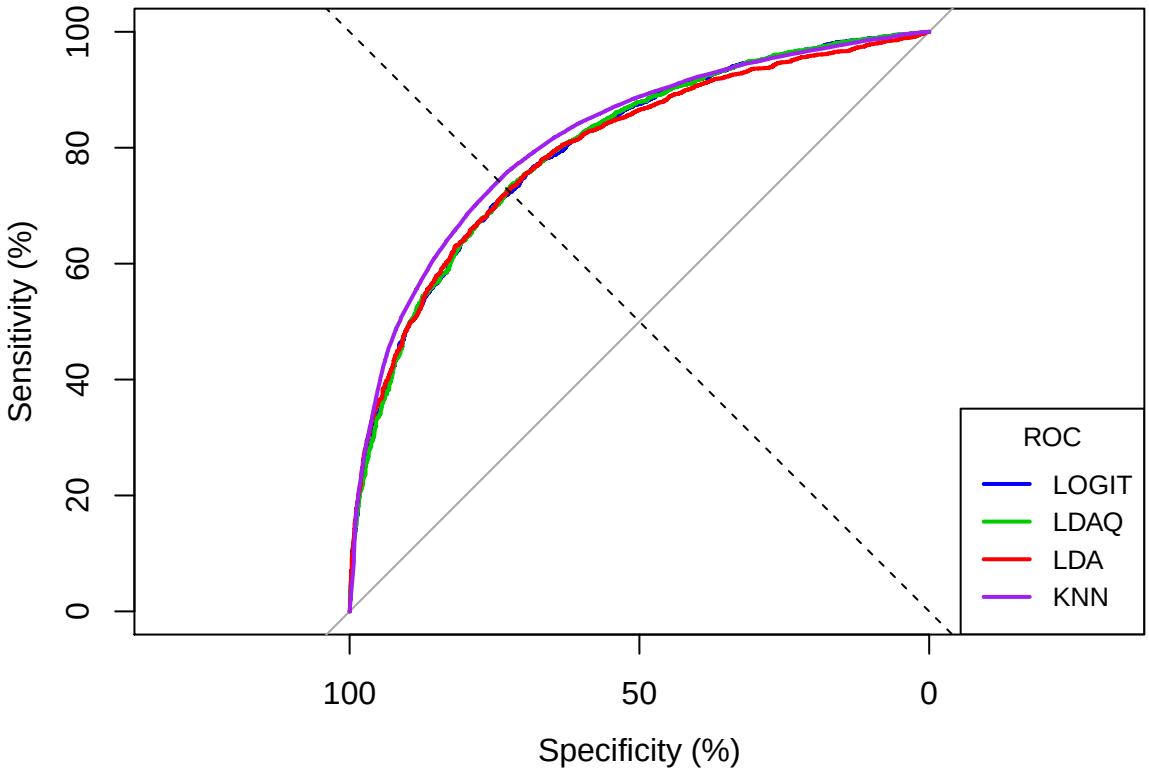
```
roc4=plot.roc(model_knn$pred$obs,model_knn$pred$Good, percent=T, print.auc=T,col="blue")
```



La courbe ROC notifie qu'on prédit toujours correctement la qualité de vin.

Comparaison des courbes ROC

```
plot(roc1, col="blue",lwd=2)
plot(roc2, add=TRUE, col="green3",lwd=2)
plot(roc3, add=TRUE, col="red",lwd=2)
plot(roc4, add=TRUE, col="purple",lwd=2)
abline(a=0,b=1, lty = 2)
legend("bottomright", legend = c("LOGIT","LDAQ","LDA","KNN"), col=c("blue","green3","red","purple"),
lty=1, title="ROC", cex=0.8, text.font=1,lwd=2) # petite légende
```



Par visualisation, il apparaît que pour ces données, KNN est le meilleur modèle qui fournit les meilleurs résultats parmis les méthodes que nous avons examinées jusqu'à présent par validation croisée à 5 couches.