TP Prétraitement et transformation

Resultats + Interpretations

Exercice 1 :

# Ouvrir le dataset
dataset <- airquality
# Connaître le nombre de valeurs différentes dans chaque colonne
sapply(dataset, function(x) length(unique(x)))

```
> dataset <- airquality
> sapply(dataset, function(x) length(unique(x)))
  Ozone Solar.R    Wind    Temp   Month     Day
     68     118      31      40       5      31
> 
```

# Compter le nombre de valeurs manquantes dans chacune des colonnes
sapply(dataset,function(x) sum(is.na(x)))

```
> sapply(dataset,function(x) sum(is.na(x)))
  Ozone Solar.R    Wind    Temp   Month     Day
     37       7       0       0       0       0
> 
```

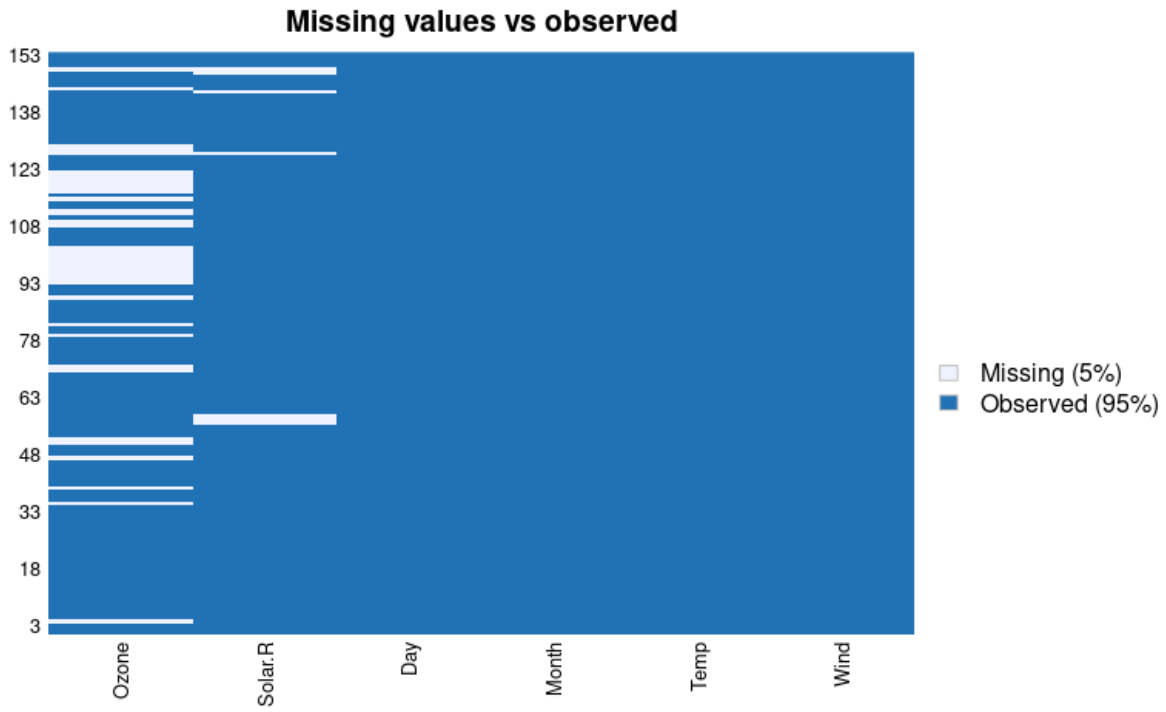
# Compter le nombre de NA dans une colonne en particulier ici Ozone
sum(is.na(dataset$colonne))
sum(is.na(dataset$Ozone))

```
> sum(is.na(dataset$Ozone))
[1] 37
```

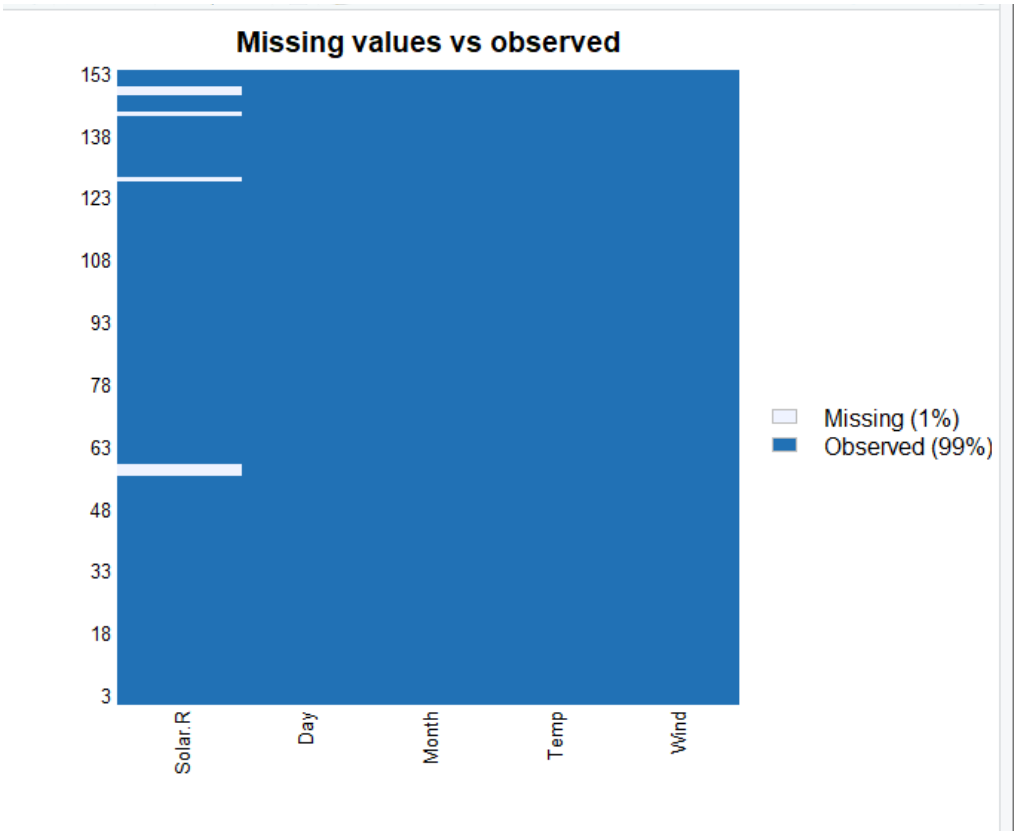
On voit que la colonne Ozone contient 37 valeurs qui sont NA
Afficher les lignes dont la valeur dans la colonne "Ozone" est NA
dataset[is.na(dataset$colonne),]
dataset[is.na(dataset$Ozone),]

```
> dataset[is.na(dataset$Ozone),]
    Ozone Solar.R Wind Temp Month Day
5      NA      NA 14.3   56     5   5
10     NA     194  8.6   69     5  10
25     NA      66 16.6   57     5  25
26     NA     266 14.9   58     5  26
27     NA      NA  8.0   57     5  27
32     NA     286  8.6   78     6   1
33     NA     287  9.7   74     6   2
34     NA     242 16.1   67     6   3
35     NA     186  9.2   84     6   4
36     NA     220  8.6   85     6   5
37     NA     264 14.3   79     6   6
39     NA     273  6.9   87     6   8
42     NA     259 10.9   93     6  11
43     NA     250  9.2   92     6  12
45     NA     332 13.8   80     6  14
46     NA     322 11.5   79     6  15
52     NA     150  6.3   77     6  21
53     NA      59  1.7   76     6  22
54     NA      91  4.6   76     6  23
55     NA     250  6.3   76     6  24
56     NA     135  8.0   75     6  25
57     NA     127  8.0   78     6  26
58     NA      47 10.3   73     6  27
59     NA      98 11.5   80     6  28
60     NA      31 14.9   77     6  29
61     NA     138  8.0   83     6  30
65     NA     101 10.9   84     7   4
72     NA     139  8.6   82     7  11
75     NA     291 14.9   91     7  14
83     NA     258  9.7   81     7  22
84     NA     295 11.5   82     7  23
102    NA     222  8.6   92     8  10
103    NA     137 11.5   86     8  11
107    NA      64 11.5   79     8  15
115    NA     255 12.6   75     8  23
119    NA     153  5.7   88     8  27
150    NA     145 13.2   77     9  27
```

# Afficher un graphe avec les valeurs manquantes vs les valeurs observées
library(Amelia)
missmap(dataset, main = "Missing values vs observed")



# Option 1. Supprimer la colonne de votre dataset par exemple dans notre cas Ozone
dataset_opt1 <- airquality
dataset_opt1$Ozone <- NULL
missmap(dataset_opt1, main = "Missing values vs observed")

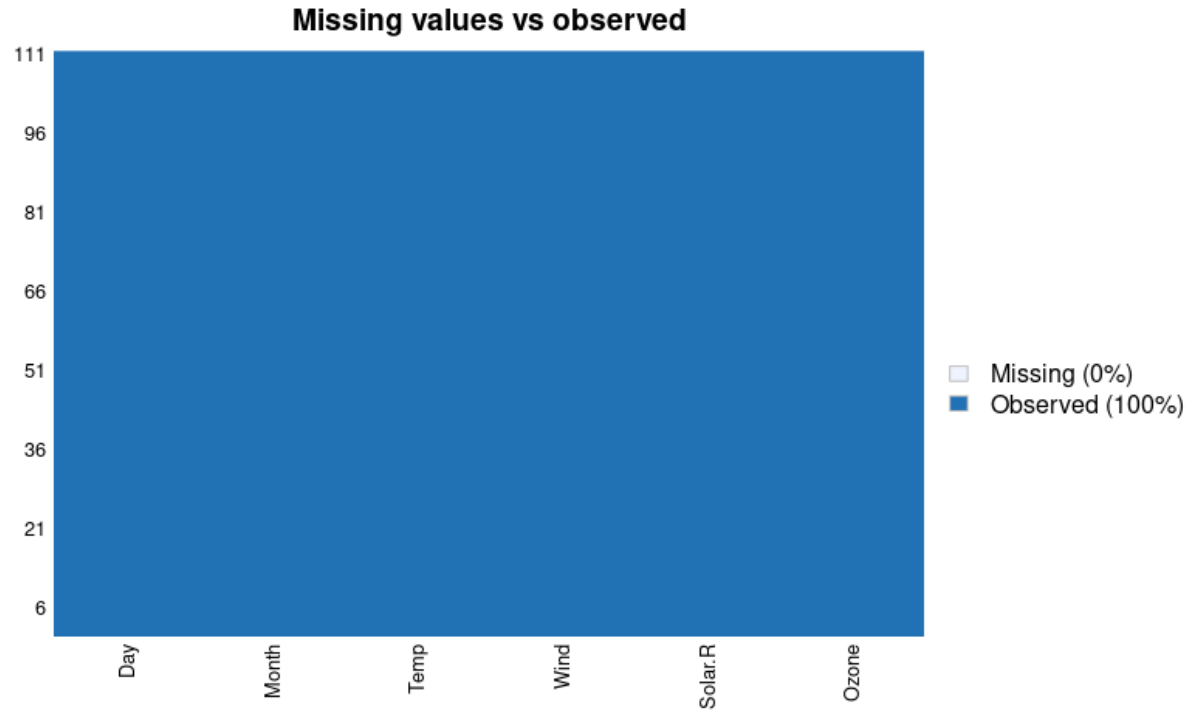


Option 2. Supprimer les lignes dont la valeur de cette colonne est NA ici on a pris les deux Ozone + Solar.R

```
dataset_2 <- subset(dataset, !is.na(dataset$colonne))
dataset_2 <- subset(dataset, !is.na(dataset$Ozone))
dataset_2 <- subset(dataset_2, !is.na(dataset_2$Solar.R))
dataset_2
missmap(dataset_2, main = "Missing values vs observed")
```

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
7	23	299	8.6	65	5	7

Par exemple c'est le cas de la ligne 5.

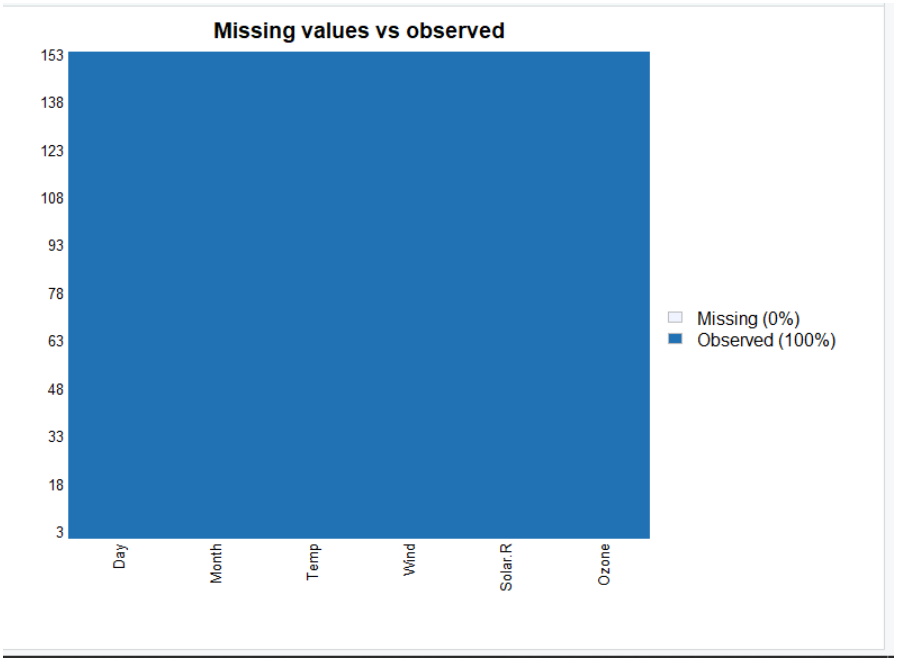


Remplacer les valeurs manquantes par la moyenne de la colonne ici on a pris les deux Ozone + Solar.R

```
dataset_opt3 <- airquality
dataset_opt3$Ozone[is.na(dataset_opt3$Ozone)] <- mean(dataset_opt3$Ozone,na.rm=T)
dataset_opt3$Solar.R[is.na(dataset_opt3$Solar.R)] <- mean(dataset_opt3$Solar.R,na.rm=T)
missmap(dataset_opt3, main = "Missing values vs observed")
```

```
> dataset_opt3 <- airquality
> dataset_opt3$Ozone[is.na(dataset_opt3$Ozone)] <- mean(dataset_opt3$Ozone,na.rm=T)
> dataset_opt3$Solar.R[is.na(dataset_opt3$Solar.R)] <- mean(dataset_opt3$Solar.R,na.rm=T)
> missmap(dataset_opt3, main = "Missing values vs observed")
> dataset_opt3
       Ozone  Solar.R Wind Temp Month Day
1   41.00000 190.0000  7.4   67     5   1
2   36.00000 118.0000  8.0   72     5   2
3   12.00000 149.0000 12.6   74     5   3
4   18.00000 313.0000 11.5   62     5   4
5   42.12931 185.9315 14.3   56     5   5
6   28.00000 185.9315 14.9   66     5   6
7   23.00000 299.0000  8.6   65     5   7
8   19.00000  99.0000 13.8   59     5   8
9    8.00000  19.0000 20.1   61     5   9
10  42.12931 194.0000  8.6   69     5  10
11   7.00000 185.9315  6.9   74     5  11
12  16.00000 256.0000  9.7   69     5  12
13  11.00000 290.0000  9.2   66     5  13
14  14.00000 274.0000 10.9   68     5  14
15  18.00000  65.0000 13.2   58     5  15
```

Et c'est le cas de la NA de la colonne Ozone (ligne 10)



Remplacer les valeurs manquantes par la médiane de la colonne ici on a pris les deux Ozone + Solar.R

```
dataset$Colonne[is.na(dataset$Colonne)] <- median(dataset$Colonne,na.rm=T)
dataset_4 <- airquality
dataset_4$Ozone[is.na(dataset_4$Ozone)] <- median(dataset_4$Ozone,na.rm=T)
dataset_4$Solar.R[is.na(dataset_4$Solar.R)] <- median(dataset_4$Solar.R,na.rm=T)
dataset_4
missmap(dataset_4, main = "Missing values vs observed")
```

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41.0	190	7.4	67	5	1
2	36.0	118	8.0	72	5	2
3	12.0	149	12.6	74	5	3
4	18.0	313	11.5	62	5	4
5	31.5	205	14.3	56	5	5
6	28.0	205	14.9	66	5	6
7	23.0	299	8.6	65	5	7
8	19.0	99	13.8	59	5	8
9	8.0	19	20.1	61	5	9
10	31.5	194	8.6	69	5	10

Utiliser la régression linéaire pour amputer la valeur manquante.

```
reg = lm(Ozone~.,dataset)
summary(reg)
dataset[is.na(dataset$Ozone), "Ozone"] <- predict(reg, newdata =
dataset[is.na(dataset$Ozone), ])
dataset
```

Ici on a creer un model de regression qui va permettre de predire les valeurs manquantes et on les remplace par les valeurs prédites. On l'a fait qu'avec la colonne Ozone.

AVANT(ligne 10)

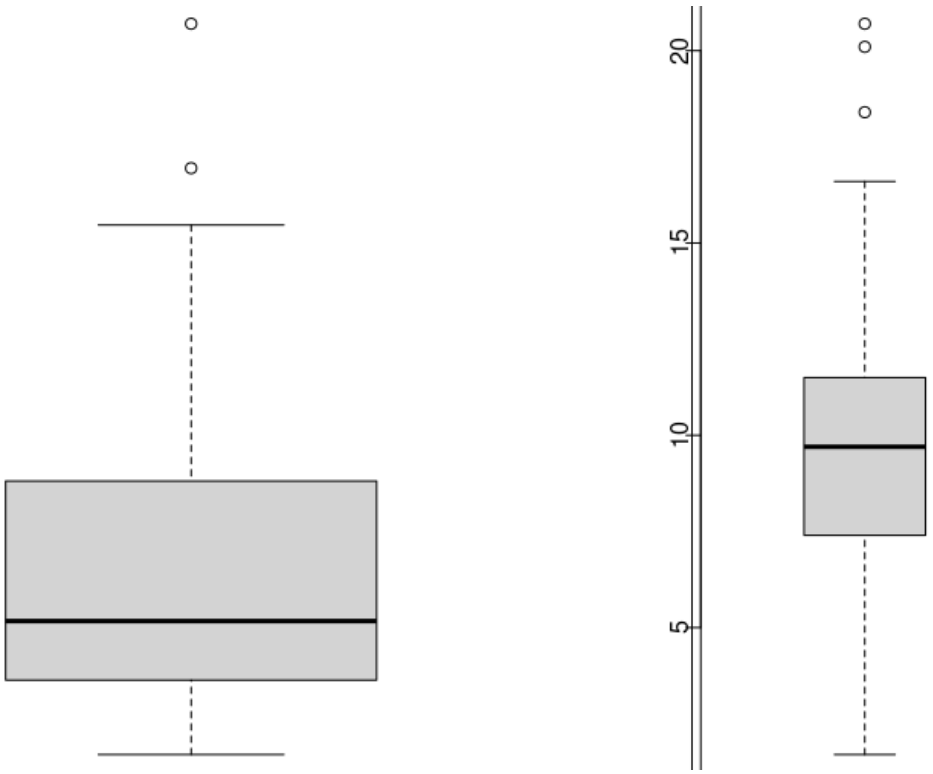
	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10

APRÈS(ligne 10)

	Ozone	Solar.R	Wind	Temp	Month	Day	ozone
1	41.000000	190	7.4	67	5	1	NA
2	36.000000	118	8.0	72	5	2	NA
3	12.000000	149	12.6	74	5	3	NA
4	18.000000	313	11.5	62	5	4	NA
5	NA	NA	14.3	56	5	5	NA
6	28.000000	NA	14.9	66	5	6	NA
7	23.000000	299	8.6	65	5	7	NA
8	19.000000	99	13.8	59	5	8	NA
9	8.000000	19	20.1	61	5	9	NA
10	35.446534	194	8.6	69	5	10	35.446534

Utiliser la boîte à moustache pour visualiser les valeurs aberrantes.

```
boxplot(airquality$Ozone)
boxplot(airquality$Wind)
```



Dans ce cas on voit que la colonne Ozone a deux valeurs aberrantes et Wind 3. (Icompter es cercles)

Exercice 2 :

Vérifier qu'une variable catégorielle est encodée en Factor

is.factor(dataset$Colonne)
is.factor(dataset$Sepal.Length)

```
> is.factor(dataset$Sepal.Length)
[1] FALSE
```

Si la réponse est FALSE, il faut l'encoder vous-même :

dataset$Colonne <- factor(dataset$Colonne, levels = c(0, 1))
dataset$Sepal.Length <- factor(dataset$Sepal.Length, levels = c(0, 1))

Encoder plusieurs variables en factor d'un coup

factor_vars <- c('colonneA','colonneB','colonneC','colonneD')
factor_vars <- c('Sepal.Length','Sepal.Width','Petal.Length','Petal.Width')

dataset[factor_vars] <- lapply(dataset[factor_vars], function(x) as.factor(x))

# On crée un nouvelle colonne pour chaque valeur de la variable catégorielle
for(unique_value in unique(dataset$Gender)){ dataset[paste("Gender", unique_value, sep = ".")] <- ifelse(dataset$Gender == unique_value, 1, 0)}

```
for(unique_value in unique(dataset$Species)){  dataset[paste("Species", unique_value, sep = ".")] <- ifelse(dataset$Species == unique_value, 1, 0)}
```

Supprimer une colonne générée par le One Hot Encoding dans notre cas “Species”
dataset$Gender.Male <- NULL
dataset$Species <- NULL

#Installez et chargez le package varhandle
library(varhandle)
dataset$variable <- unfactor(dataset$variable)

Mettre les colonnes 1 et 2 à la même échelle
dataset[,1:2] = scale(dataset[,1:2])

Pour cette exemple, nous allons normaliser deux colonnes: Sepal.Length et Sepal.Width

	Sepal.Length	Sepal.Width
1	-0.89767388	1.01560199
2	-1.13920048	-0.13153881
3	-1.38072709	0.32731751
4	-1.50149039	0.09788935
5	-1.01843718	1.24503015
6	-0.53538397	1.93331463
7	-1.50149039	0.78617383
8	-1.01843718	0.78617383
9	-1.74301699	-0.36096697
10	-1.13920048	0.09788935
11	-0.53538397	1.47445831
12	-1.25996379	0.78617383
13	-1.25996379	-0.13153881
14	-1.86378030	-0.13153881
15	-0.05233076	2.16274279
16	-0.17309407	3.08045544
17	-0.53538397	1.93331463
18	-0.89767388	1.01560199
19	-0.17309407	1.70388647
20	-0.89767388	1.70388647
21	-0.53538397	0.78617383
22	-0.89767388	1.47445831
23	-1.50149039	1.24503015
24	-0.89767388	0.55674567
25	-1.25996379	0.78617383
26	-1.01843718	-0.13153881
27	-1.01843718	0.78617383
28	-0.77691058	1.01560199
29	-0.77691058	0.78617383
30	-1.38072709	0.32731751

- Utiliser une autre normalisation

  On peut utiliser, dans ce cas, la normalisation Min/Max.
  library(caret) # Pour les fonctions min et max

```
# Implémenter la fonction qui permet de faire la normalisation.
minMax <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# Normalisation des données
normalisedData <- as.data.frame(lapply(dataset, minMax))
head(normalisedData)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species.setosa	Species.versicolor	Species.virginica
1	0.22222222	0.6250000	0.06779661	0.04166667	1	0	0
2	0.16666667	0.4166667	0.06779661	0.04166667	1	0	0
3	0.11111111	0.5000000	0.05084746	0.04166667	1	0	0
4	0.08333333	0.4583333	0.08474576	0.04166667	1	0	0
5	0.19444444	0.6666667	0.06779661	0.04166667	1	0	0
6	0.30555556	0.7916667	0.11864407	0.12500000	1	0	0

Exercice 3 :

1- Approche filtre:

Extraire les attributs pertinents en utilisant le gain d'information sur les datasets Glass, Shuttle, Sonar en suivant le code suivant:.

```
library(FSelector)
```

Exemple sur Iris

```
data=iris
```

Gain d'information:

```
weights <- information.gain(Species~., data)
print(weights)
```

prendre les k meilleurs (k=2)

```
subset <- cutoff.k(weights, 2)
```

ou prendre un % d'attributs

```
subset1 <- cutoff.k.percent(weights,0.5)
```

utiliser cette formule dans le classifieur

```
f <- as.simple.formula(subset, "Species")
print(f)
```

<u>Sans normalisation: (voir le fonctionnement seulement)</u>

```
install.packages("FSelector")
library(FSelector)

# Load the CSV file data
dataset <- read.csv("/home/ludmila/Downloads/archive/glass.csv")
dataset
```

```
         RI    Na   Mg   Al    Si    K    Ca   Ba   Fe Type
1   1.52101 13.64 4.49 1.10 71.78 0.06  8.75 0.00 0.00    1
2   1.51761 13.89 3.60 1.36 72.73 0.48  7.83 0.00 0.00    1
3   1.51618 13.53 3.55 1.54 72.99 0.39  7.78 0.00 0.00    1
4   1.51766 13.21 3.69 1.29 72.61 0.57  8.22 0.00 0.00    1
5   1.51742 13.27 3.62 1.24 73.08 0.55  8.07 0.00 0.00    1
6   1.51596 12.79 3.61 1.62 72.97 0.64  8.07 0.00 0.26    1
7   1.51743 13.30 3.60 1.14 73.09 0.58  8.17 0.00 0.00    1
8   1.51756 13.15 3.61 1.05 73.24 0.57  8.24 0.00 0.00    1
9   1.51918 14.04 3.58 1.37 72.08 0.56  8.30 0.00 0.00    1
10  1.51755 13.00 3.60 1.36 72.99 0.57  8.40 0.00 0.11    1
11  1.51571 12.72 3.46 1.56 73.20 0.67  8.09 0.00 0.24    1
12  1.51763 12.80 3.66 1.27 73.01 0.60  8.56 0.00 0.00    1
```

# On verra le gain d'information par rapport a la colonne Type
weights <- information.gain(Type~., dataset)
print(weights)

```
> print(weights)
   attr_importance
RI       0.0000000
Na       0.1966412
Mg       0.3992271
Al       0.1906301
Si       0.0000000
K        0.2588650
Ca       0.0000000
Ba       0.1989846
Fe       0.0000000
```

subset <- cutoff.k(weights, 2)
subset1 <- cutoff.k.percent(weights,0.5)
f <- as.simple.formula(subset, "Type")
print(f)

```
> print(f)
Type ~ Mg + K
<environment: 0x5601edb57f48>
```

Ici on peut facilement voir que Mg et K ont le gain le plus important par rapport a Type.

- Utiliser une autre évaluation (gain.ratio, symmetrical.uncertainty,.....).
# On utilise le gain.ratio, on aura le code suivant
weights_gr <- gain.ratio(Type ~ ., dataset)
print(weights_gr)
subset_gr <- cutoff.k(weights_gr, 2)
f1 <- as.simple.formula(subset_gr, "Type")
print(f1)

Resultat:

```
> print(f1)
Type ~ Ba + Mg
<environment: 0x5601edff3788>
```

Remarque: d'après quelques recherches, on arrive a la conclusion qu'il est normal de trouver des résultats différents pour les méthodes gain.ratio et information.gain car ce sont des mesures d'évaluation différentes avec des critères différents pour la sélection des caractéristiques. Et donc, le choix de la méthode dépend du problème spécifique et des caractéristiques des données.
Mais dans notre cas, nos données ne sont même pas normaliser et donc les résultats ne seront pas corrects.

Avec normalisation: On utilise les données obtenues de l'exercice 2:

But: Extraire les attributs pertinents en utilisant le gain d'informations par rapport a Petal.Length

```
weights <- information.gain(Petal.Length~., normalisedData)
print(weights)
```

```
> print(weights)
                  attr_importance
Sepal.Length            0.7070120
Sepal.Width             0.1624881
Petal.Width             0.8856912
Species.setosa          0.5092113
Species.versicolor      0.3485508
Species.virginica       0.4758537
```

```
subset <- cutoff.k(weights, 2)
f <- as.simple.formula(subset, "Petal.Length")
print(f)
```

```
> print(f)
Petal.Length ~ Petal.Width + Sepal.Length
<environment: 0x56357555c488>
```

Cela signifie que, selon le calcul du gain d'information, les caractéristiques Petal.Width et Sepal.Length sont les plus informatives pour prédire la variable cible Petal.Length.