TP mixte 2 : apprentissage statistique

Installation des librairies

```
rm(list=ls())
install.packages('party')
install.packages('caret')
install.packages('rpart')
install.packages('rpart.plot')
install.packages('ggplot2')
install.packages('rattle')
install.packages('e1071')
install.packages("Metrics")
install.packages("isotree")
library(party)
library(caret)
library(rpart)
library(rpart.plot)
library(ggplot2)
library(rattle)
library(e1071)
library(Metrics)
library(pROC)
library(Rarity)
library(isotree)
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
<ipython-input-1-1d04e8b453a3> in <cell line: 1>()
----> 1 rm(list=ls())
      2 install.packages('party')
      3 install.packages('caret')
      4 install.packages('rpart')
      5 install.packages('rpart.plot')

NameError: name 'rm' is not defined
```

Question 1 : analyse statistique

```
Data <- read.csv('Income_Inequality;csv',sep=';')

str(Data)
parametres <- names(Data)[1:22]
print(parametres)
```

Nous avons 870 observations de 22 variables. Celles-ci correspondent à des observations de 87 pays sur la période de 2010 à 2019, de différents paramètres.

Nous avons 19 données explicatives telles que des données :

- économiques(eco 1, 2, 3)
- énergétiques (energie 1,2,3)
- sanitaires (1, 2)
- financières (finance 1,2,3,4,5)
- sur la gouvernance
- sur la pauvreté
- sur l'environnement
- trois autres

Toutes ces données sont des variables continues hormis "energy1" qui est une variable entière.

A cela s'ajoute 2 variables entières :

- Country : identifiant le pays
- Year : identifiant l'année des données

Ainsi ces 21 variables ont pour objectif de déterminer les inégalités de revenus dans ces 87 pays afin de les classifier en deux catégories; les pays "H" avec de fortes innégalités de revenus, et les pays "L" avec des faibles innégalités entre les revenus, qui est la variable dépendante représentant "income inequality".

```
missing_values <- sapply(Data, function(x) sum(is.na(x)))
print(missing_values)
```

Nous observons que certaines valeurs dans la colonne other 2 pour chaque pays entre 2016 et 2019, sont égales à 0. Et nous obtenons ainsi aucune valeur manquantes.

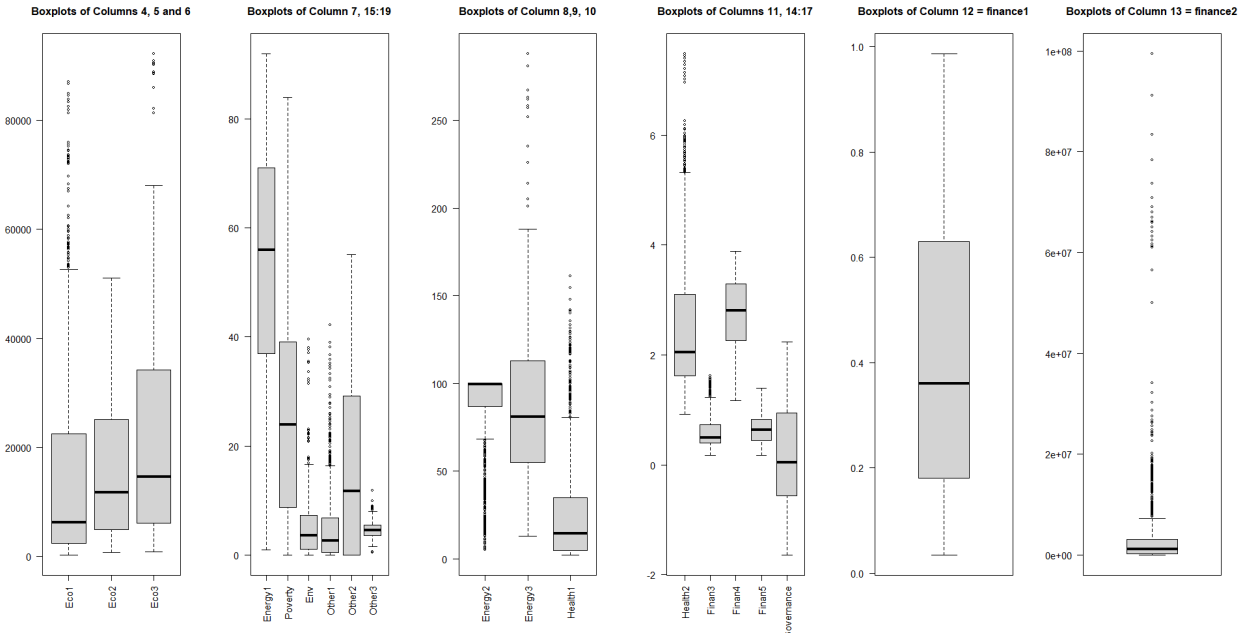
```
summary(Data)
table(Data$Income_Inequality)
```

Nous avons 542 observations qui ont été classés dans la classe "H" et 328 dans la classe "L".

Boxplots

```
max_val <- apply(Data[, 4:22], 2, max)
ordegrandeur<- floor(log10(max_val))
print(ordegrandeur)
par(mfrow=c(1,6))
boxplot(Data[, 4:6], main="Boxplots of Columns 4, 5 and 6", las=2)
boxplot(Data[, c(7,18:22)], main="Boxplots of Column 7, 15:19", las=2)
boxplot(Data[, 8:10], main="Boxplots of Column 8,9, 10", las=2)
boxplot(Data[, c(11,14,15,16,17)], main="Boxplots of Columns 11,
14:17", las=2)
boxplot(Data[, 12], main="Boxplots of Column 12 = finance1", las=2)
```

```
boxplot(Data[, 13], main="Boxplots of Column 13 = finance2", las=2)
par(mfrow=c(1,1))
```

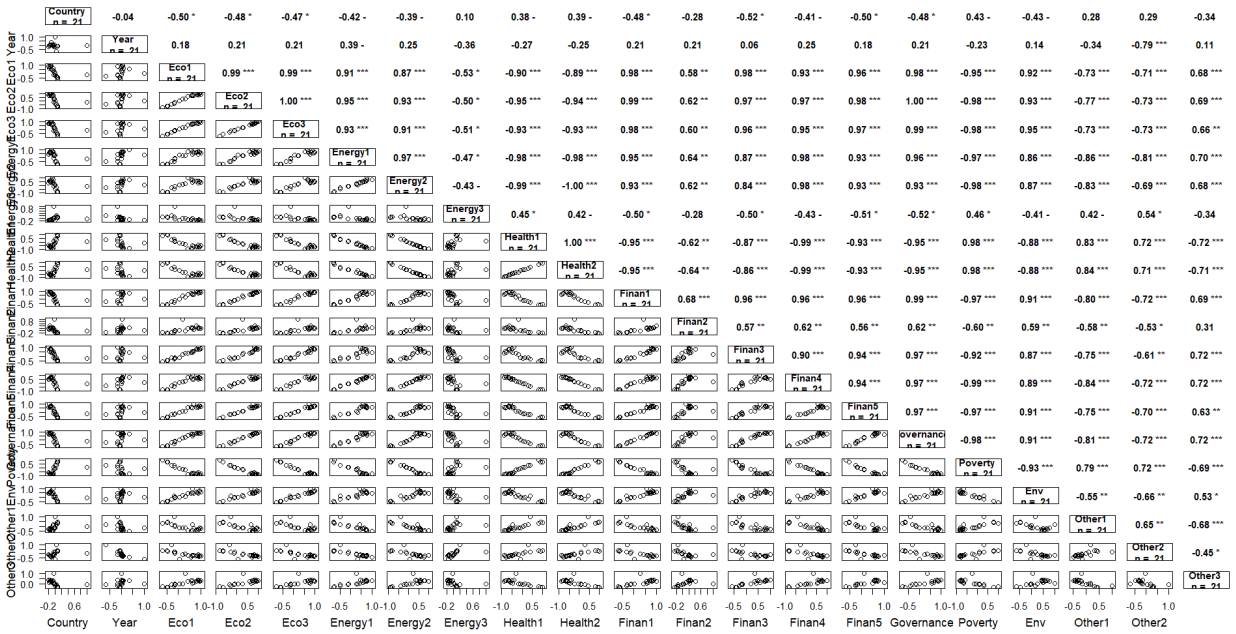


Nous remarquons que pour certains paramètres il existe des valeurs abérrantes :

- supérieures pour les variables suivantes: eco 1, eco 3, environnement, other 1 , energy 3, health 1, health 2,finance 3
- inférieur pour la variable energy 2

Matrice de corrélation

```
cor_matrice <- cor(Data[, sapply(Data, is.numeric)])
print(cor_matrice)
corPlot(cor_matrice, method = 'pearson')
```



Les variables "Eco1" "eco2" "eco3" sont fortement corrélées cela pourrrait signifier qu'elles mesurent des aspects similaires de l'économie.

"Energy1" est très positivement corrélée a "Eco2" et "Eco3", tandis que "Energy2" montre une très forte corrèlation négative avec "Health1" et "Health2".

"Finance1" est corrélé avec "Eco1", "Eco2", "Eco3", ce qui inidque un lien entre les finances et l'économie.``

Enfin, "Poverty" a une forte corrélation négative avec "Eco2", "Energy1", "Finance1" et "Governace". Cela signifie que des améliorations dans ces domaines pourraient être associées à une réduction de pauvreté.

Distribution

```
skewness_vals <- sapply(Data[, sapply(Data, is.numeric)], skewness)
kurtosis_vals <- sapply(Data[, sapply(Data, is.numeric)], kurtosis)
print(skewness_vals[3:21])
print(kurtosis_vals[3:21])
```

Energy2 présente une asymétrie négative marquée (-1.6657157) et une kurtosis positive (1.2735631), suggérant une distribution avec une queue à gauche et des queues plus lourdes que la normale.

Finan2 se démarque par une asymétrie très positive (4.6778757) et une kurtosis extrêmement élevée (25.3042247), ce qui indique une distribution très asymétrique avec une longue queue à droite et des valeurs extrêmes plus fréquentes que dans une distribution normale.

Env et Other1 ont également des valeurs élevées de skewness et kurtosis, suggérant une asymétrie positive et des queues plus lourdes.

Plusieurs autres variables telles que Health1, Health2, Eco3, et Other3 montrent aussi une asymétrie positive avec une kurtosis plus ou moins élevée, suggérant des distributions avec des queues à droite.

Il y a aussi des cas où la kurtosis est négative (comme Eco2, Energy1, Finan4, Governance, Poverty, et Other2), ce qui suggère des distributions plus aplaties que la normale.

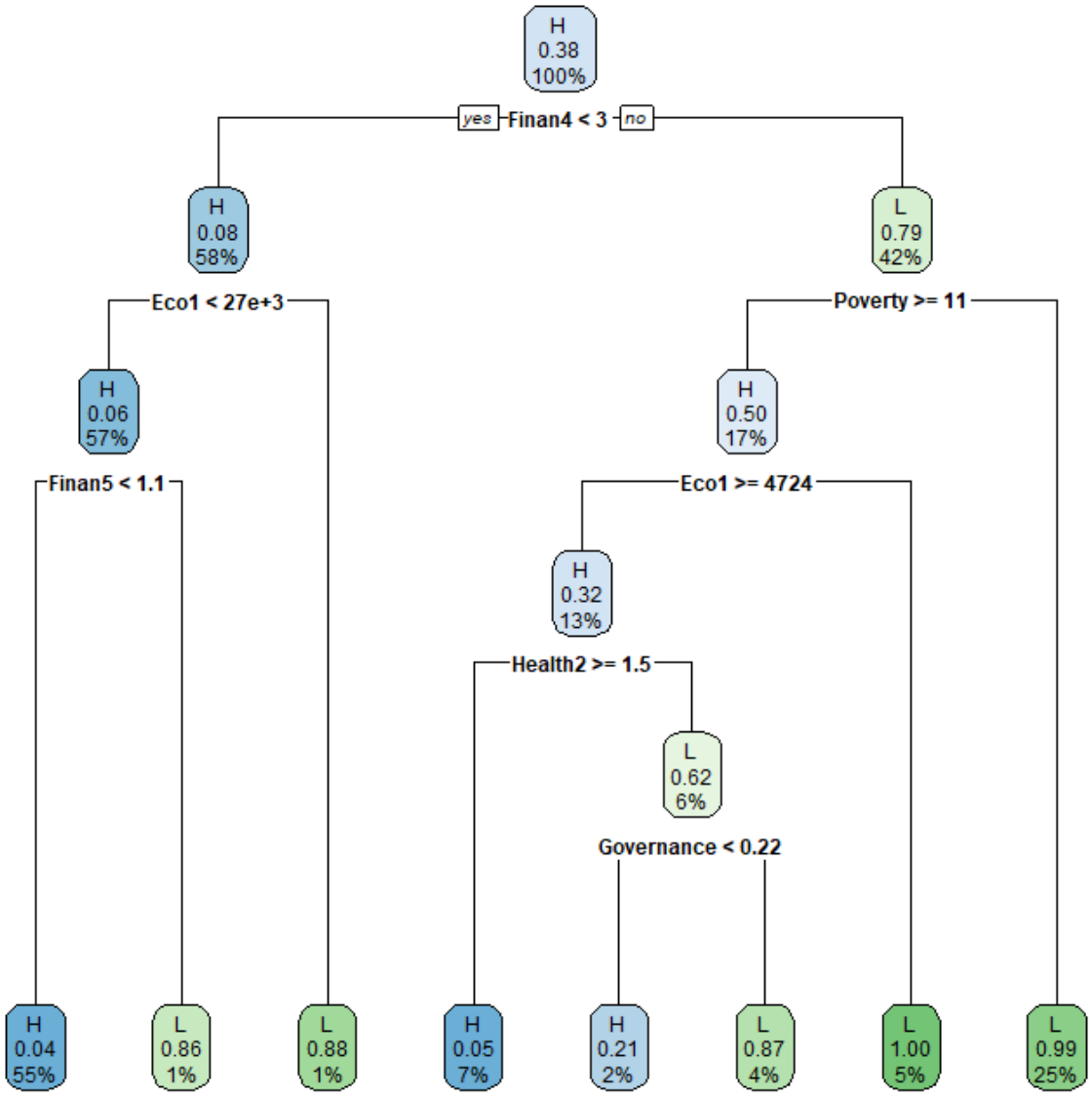
Question 2 : construction de l'arbre de décision

Division des données

```
set.seed(1234)
Data <- Data[, -c(1, 2)]
splitIndex <- createDataPartition(Data$Income_Inequality, p = 0.7,
list = FALSE)
trainData <- Data[splitIndex, ]
testData <- Data[-splitIndex, ]
```

Construction et optimisation de l'arbre

```
fitControl <- trainControl(method = "cv", number = 5)
fit <- train(Income_Inequality ~ ., data=trainData, method="rpart",
trControl=fitControl,
             tuneGrid=data.frame(cp=seq(0.01, 0.5, 0.01)))
best_cp <- fit$bestTune$cp
tree <- rpart(Income_Inequality ~ ., data=trainData, cp=best_cp)

# Visualiser l'arbre élagué
windows(rescale="fit", width=6, height=6)
rpart.plot(tree)
print(tree)

#predictions
predictions <- predict(tree, testData, type="class")
```

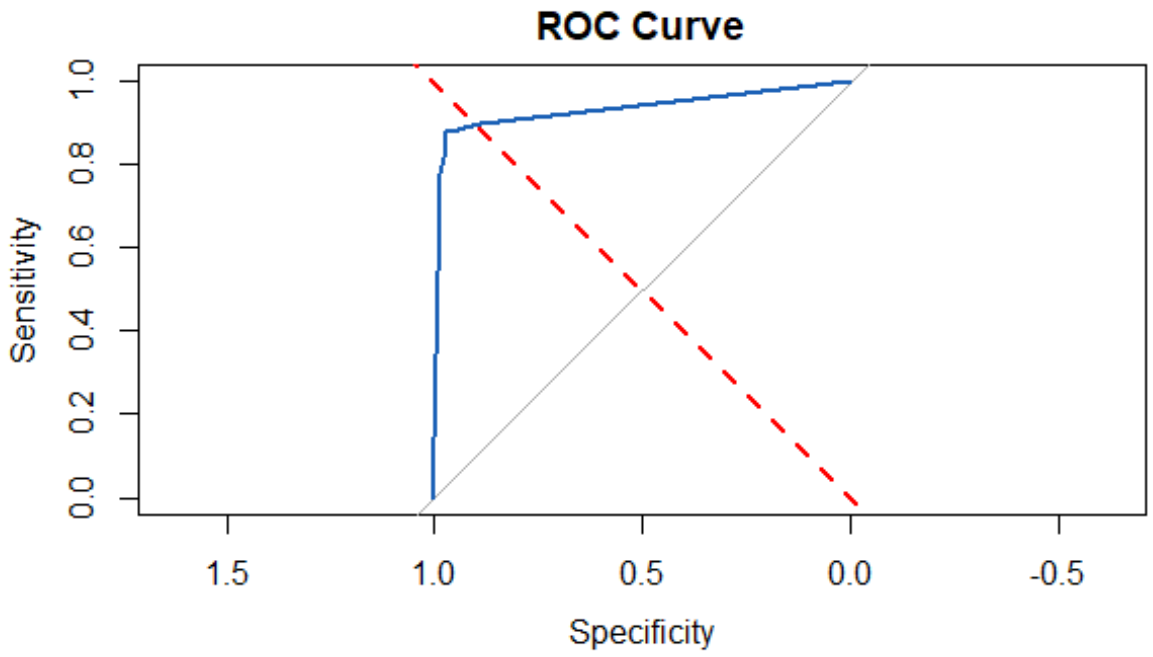


Résultat de notre modele

```
#Matrice de confusion
predictions <- factor(predictions)
testData$Income_Inequality <- factor(testData$Income_Inequality)
cm <- confusionMatrix(predictions, testData$Income_Inequality)
print(cm)

#Detailed Accuracy By Class
cm$byClass
```

Courbe ROC et AUC

```
oc_obj <- roc(testData$Income_Inequality, as.numeric(predictions))
plot.roc(roc_obj, main="Courbe ROC", col="blue")
auc(roc_obj)
```



Interprétation

Detailed Accuracy By Class

- Sensibilité: Notre modèle identifie correctement 97.53% des cas "H".
- Spécificité: Il identifie correctement 87.75% des cas "L".
- Précision: Lorsqu'il prédit la classe "H", il est juste 92.94% du temps.
- Valeur Prédictive Négative: Lorsqu'il prédit la classe "L", il est juste 95.56% du temps.
- Score F1: Le modèle a un bon équilibre entre précision et rappel, avec un score de 95.18%.
- Exactitude Équilibrée: Le modèle performe bien sur les deux classes avec une moyenne de 92.64%.

Globalement, votre modèle est très efficace pour classer les individus dans les catégories "H" et "L".

Matrice de confusion

Sur 260 prédictions, **158** sont des vrais positifs (H correctement identifiés), **86** sont des vrais négatifs (L correctement identifiés), **12** sont des faux positifs (L incorrectement identifiés comme H), **4** sont des faux négatifs (H incorrectement identifiés comme L.

- Exactitude: Le modèle a correctement prédit 93.85% des cas (soit la précision globale).

- Intervalle de confiance: Il y a 95% de chances que l'exactitude réelle du modèle se situe entre 90.2% et 96.44%.
- Taux d'erreur sans information: Si le modèle avait toujours prédit la classe majoritaire, il aurait été correct dans 62.31% des cas.
- Valeur de P: La probabilité que le modèle soit précis par hasard est inférieure à 2e-16, ce qui indique que la précision du modèle est significative.
- Kappa: La statistique Kappa de 0.8668 montre un très bon accord entre les prédictions et les valeurs réelles, au-delà de la chance.
- Test de McNemar: La valeur de p de 0.08012 suggère qu'il n'y a pas de différence significative dans l'erreur de prédiction entre les classes H et L (pas de biais significatif vers l'une ou l'autre classe).

La performance du modèle est forte et significative, avec une très bonne précision et un équilibre entre les capacités de détection des classes H et L.

Courbe ROC et AUC

Le graphique de la courbe ROC montre que notre modèle discrimine bien entre les classes "H" et "L". Avec une AUC de 0.9356, le modèle a une très bonne capacité de classification. Cela confirme nos résultats précédents.

Question 3 : implémentation d'une forêt aléatoire

Construction de la forêt

```
iso_forest <- isolation.forest(Data)
```

Calcul des scores

```
anomaly_scores <- predict(iso_forest, Data)
```

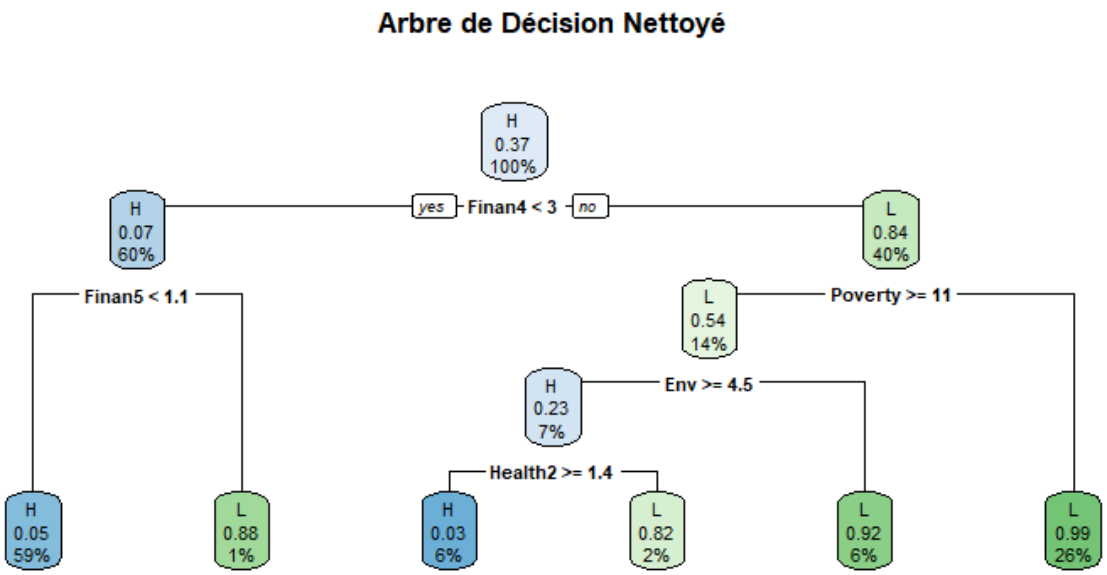
10 observations ayant les scores d'anomalie les plus élevés et 10 observations ayant les scores les plus bas

```
# Trier les scores d'anomalie et obtenir les indices des 10 scores les plus élevés et les plus bas
top_10_anomalies_indices <- order(anomaly_scores, decreasing = TRUE)[1:10]
bottom_10_anomalies_indices <- order(anomaly_scores)[1:10]

# Extraire les observations correspondantes
top_10_anomalies <- Data[top_10_anomalies_indices, ]
bottom_10_anomalies <- Data[bottom_10_anomalies_indices, ]
```

```
# Afficher les observations
print("Top 10 Anomalies:")
print(top_10_anomalies)
print("Bottom 10 Anomalies:")
print(bottom_10_anomalies)
```

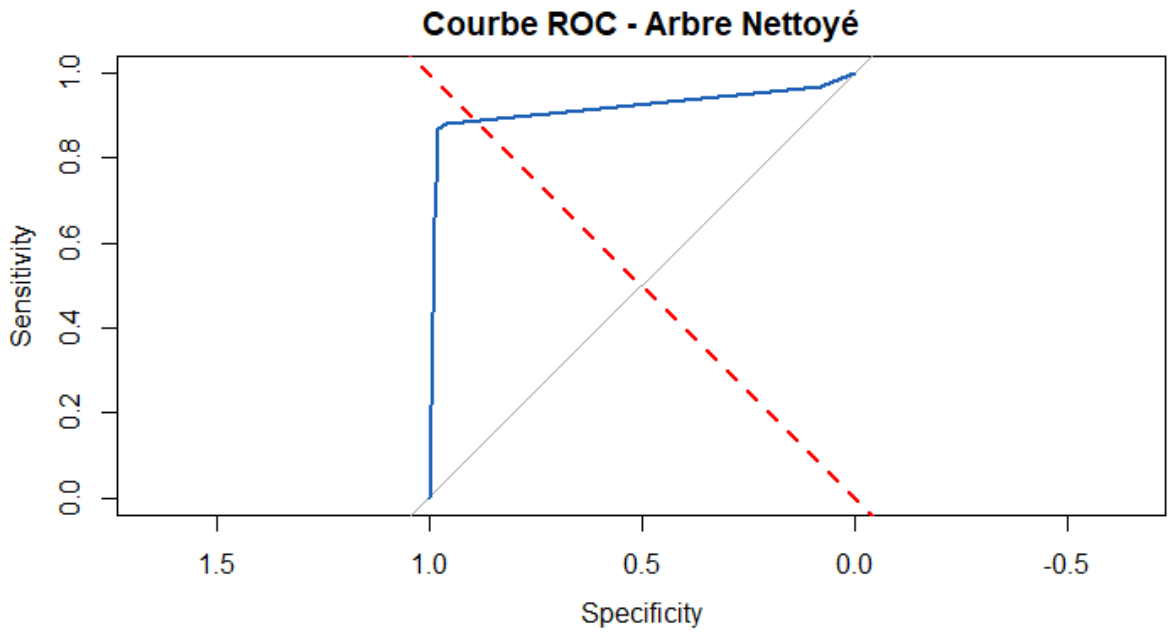
Suprression des 50 observations ayant les scores d'anomalies les plus élevés et construction du nouvel arbre de décision

```
# Supprimer les 50 observations ayant les scores d'anomalie les plus élevés
indices_to_remove <- order(anomaly_scores, decreasing = TRUE)[1:50]
Data_cleaned <- Data[-indices_to_remove, ]

# Diviser les données nettoyées en ensembles d'entraînement et de test
set.seed(1234)
splitIndex_cleaned <-
createDataPartition(Data_cleaned$Income_Inequality, p = 0.7, list =
FALSE)
trainData_cleaned <- Data_cleaned[splitIndex_cleaned, ]
testData_cleaned <- Data_cleaned[-splitIndex_cleaned, ]

# Construire et visualiser un nouvel arbre de décision avec les données nettoyées
fitControl_cleaned <- trainControl(method = "cv", number = 5)
fit_cleaned <- train(Income_Inequality ~ ., data = trainData_cleaned,
method = "rpart", trControl = fitControl_cleaned, tuneGrid =
data.frame(cp = seq(0.01, 0.5, 0.01)))
best_cp_cleaned <- fit_cleaned$bestTune$cp
tree_cleaned <- rpart(Income_Inequality ~ ., data = trainData_cleaned,
cp = best_cp_cleaned)
dev.off()
rpart.plot(tree_cleaned, main="Arbre de Décision Nettoyé")
```




```r
# Prédictions avec l'arbre nettoyé
predictions_cleaned <- predict(tree_cleaned, testData_cleaned, type =
"class")

#Matrice de confusion
# Convertir les prédictions et les vraies valeurs en facteurs
predictions_cleaned <- factor(predictions_cleaned)
testData_cleaned$Income_Inequality <-
factor(testData_cleaned$Income_Inequality)

# Calculer la matrice de confusion
predictions_cleaned <- factor(predictions_cleaned)
testData_cleaned$Income_Inequality <-
factor(testData_cleaned$Income_Inequality)
cm_cleaned <- confusionMatrix(predictions_cleaned,
testData_cleaned$Income_Inequality)
print(cm_cleaned)

#Detailed Accuracy By Class
cm_cleaned$byClass

#Courbe ROC et AUC
# Prédictions de probabilité
prob_predictions_cleaned <- predict(tree_cleaned, testData_cleaned,
type="prob")

# Calcul de la courbe ROC et de l'AUC
roc_obj_cleaned <- roc(response = testData_cleaned$Income_Inequality,
predictor = prob_predictions_cleaned[, "H"])
auc_cleaned <- auc(roc_obj_cleaned)
```

```
auc_cleaned

# Tracer la courbe ROC
plot(roc_obj_cleaned, main="Courbe ROC - Arbre Nettoyé",
col="#1c61b6", lwd=2)
abline(a=0, b=1, lwd=2, lty=2, col="red")
```



Interpretation des performances de notre nouveau modele

Analyse Détaillée par Classe

- Sensibilité (96.10%) : Le modèle est extrêmement efficace pour identifier correctement les cas "H". Avec une sensibilité de près de 96%, cela indique que peu de vrais cas "H" sont manqués par le modèle.
- Spécificité (87.91%) : Le modèle est également efficace pour identifier les cas "L", avec une spécificité d'environ 88%. Cela signifie qu'il y a une probabilité modérée de fausses alarmes (cas "L" incorrectement classés comme "H").
- Précision (93.08%) : Lorsque le modèle prédit la classe "H", il est correct environ 93% du temps. Cela reflète un taux de faux positifs relativement faible.
- Valeur Prédictive Négative (93.03%) : Lorsqu'il prédit la classe "L", le modèle est presque toujours juste, indiquant que peu de vrais cas "L" sont incorrectement classifiés.
- Score F1 (94.57%) : Le modèle montre un bon équilibre entre la précision et le rappel, indiquant une performance robuste à la fois dans la détection des cas "H" et dans la minimisation des fausses alarmes.
- Exactitude Équilibrée (92.01%) : Le modèle affiche des performances presque égales pour les deux classes, reflétant une bonne capacité à gérer les cas "H" et "L" de manière équilibrée.

Matrice de confusion

- Exactitude (93.06%) : Le modèle a une précision globale élevée, indiquant qu'il classifie correctement une grande majorité des observations.
- Kappa (0.8497) : La statistique Kappa montre un accord très bon entre les prédictions et les valeurs réelles, bien au-delà de ce qui serait attendu par hasard.
- Mcnemar's Test (P-Value = 0.332) : La valeur de p de 0.332 suggère qu'il n'y a pas de différence significative dans l'erreur de prédiction entre les classes H et L.

Courbe ROC et AUC

Avec une AUC de 0.9179, le modèle démontre une tres bonne capacité de classification. Cela confirme la robustesse du modèle mise en évidence par d'autres métriques.

Comparaison des performances

Au niveau des métriques

En retirant les 50 observations les plus anormales, le modèle a montré une légère baisse dans la plupart des métriques de performance, notamment en sensibilité, en valeur prédictive négative, en score F1, en exactitude équilibrée, et en AUC. Cela suggère que bien que le nettoyage des données puisse parfois être bénéfique, dans ce cas particulier, il a légèrement diminué la performance globale du modèle, indiquant que les données éliminées pouvaient contenir des informations utiles pour le modèle.

Au niveau des arbres

- Pronfondeur de l'arbre : 6 vs 5
- Noeud : 7 vs 5
- Feuilles : 8 vs 6
- Variables Utilisées : Finan4, Eco1, Finan5, Poverty, Health2, Governance vs Finan 4, Finan5, Poverty, Env, Health

Conclusion

L'Arbre 1, avec une profondeur, un nombre de noeuds et de feuilles plus élevé, ainsi qu'une gamme plus large de variables, est probablement plus détaillé et précis dans des situations spécifiques, mais avec un risque accru de surajustement.

En revanche, l'Arbre 2, bien que potentiellement moins précis dans certains cas, pourrait offrir une meilleure généralisation grâce à sa structure plus simple et à la focalisation sur un ensemble de variables plus restreint.

Finalement, le choix entre les deux dépendra de l'équilibre souhaité entre précision spécifique et généralisabilité dans l'application prévue.

Question 4 : ACP sur Rp

Travail sur les données TRAIN (70%)

L'analyse statistique réalisée au début du TP nous permet de mettre en évidence certaines variables qui dominent les autres par leur variance élevée. Afin d'éviter d'étirer trop le nuage de

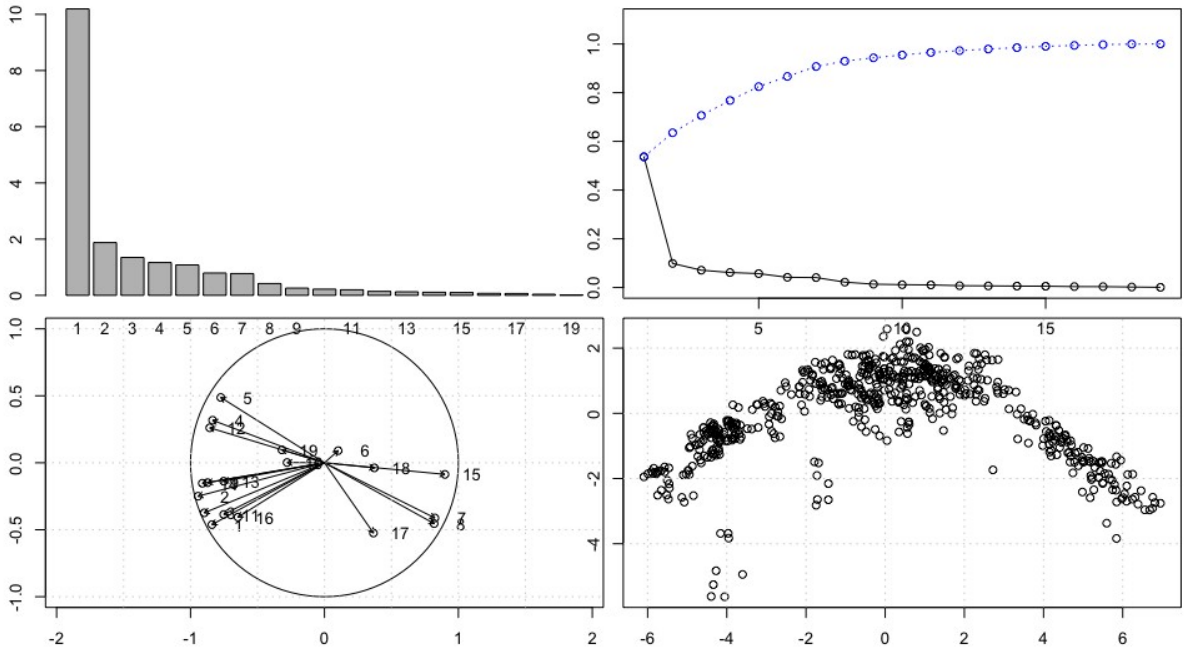
points et de biaiser notre ACP, nous décidons de réduire les données pour travailler sur un nuage de points standardisés.

Récupération des individus de train

```
trainDataACP <- trainData[,2:ncol(trainData)]
```

Test d'ACP pour déterminer le nombre de composantes à retenir

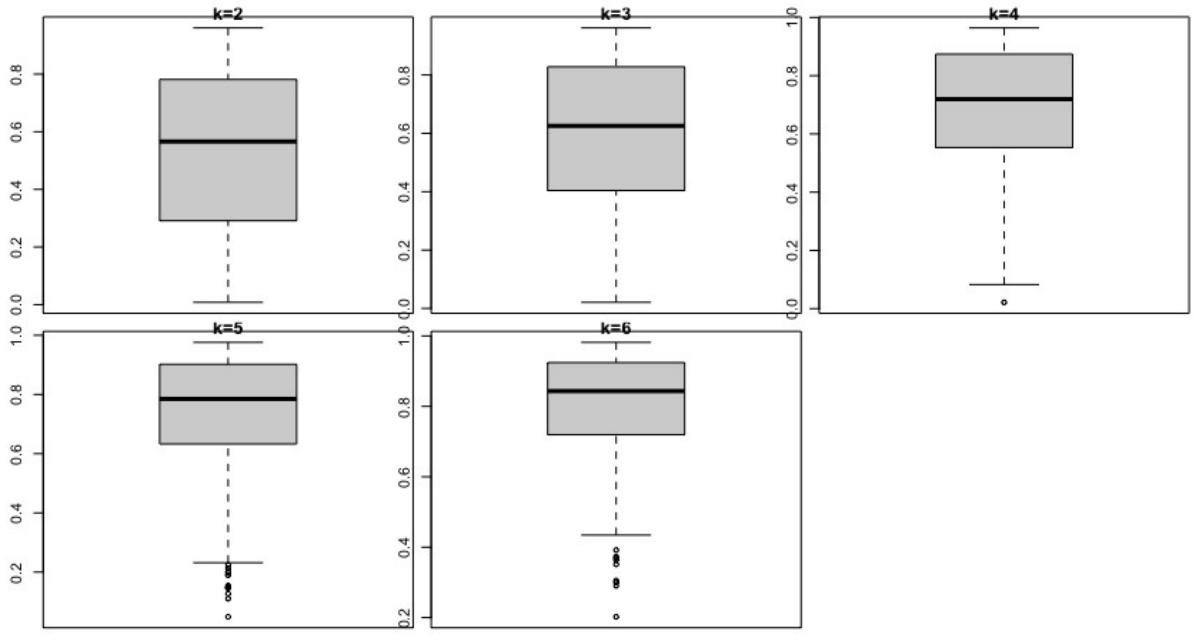
```
acp <- ACP(trainDataACP, norm = TRUE)
q_k <- quality_ACP(acp$Val_p, acp$Comp, acp$Vect_p, norm=TRUE)
```



On remarque très rapidement la prédominance d'une variance, malgré la normalisation de nos données. Plusieurs tests nous permettent de déterminer un nombre k de composantes à conserver :

- Test de Kaiser : conserver les 6 premières valeurs propres, toutes supérieures à 1,
- Sélectionner un nombre de valeur propre nous permettant d'atteindre le seuil des 80% d'inertie totale expliquée.
- Conserver les valeurs propres prédominantes.

On décide de considérer le test de Kaiser comme base, et de tracer un boxplot de la qualité de la projection des individus dans le cas k = {2, 3, 4, 5, 6}.

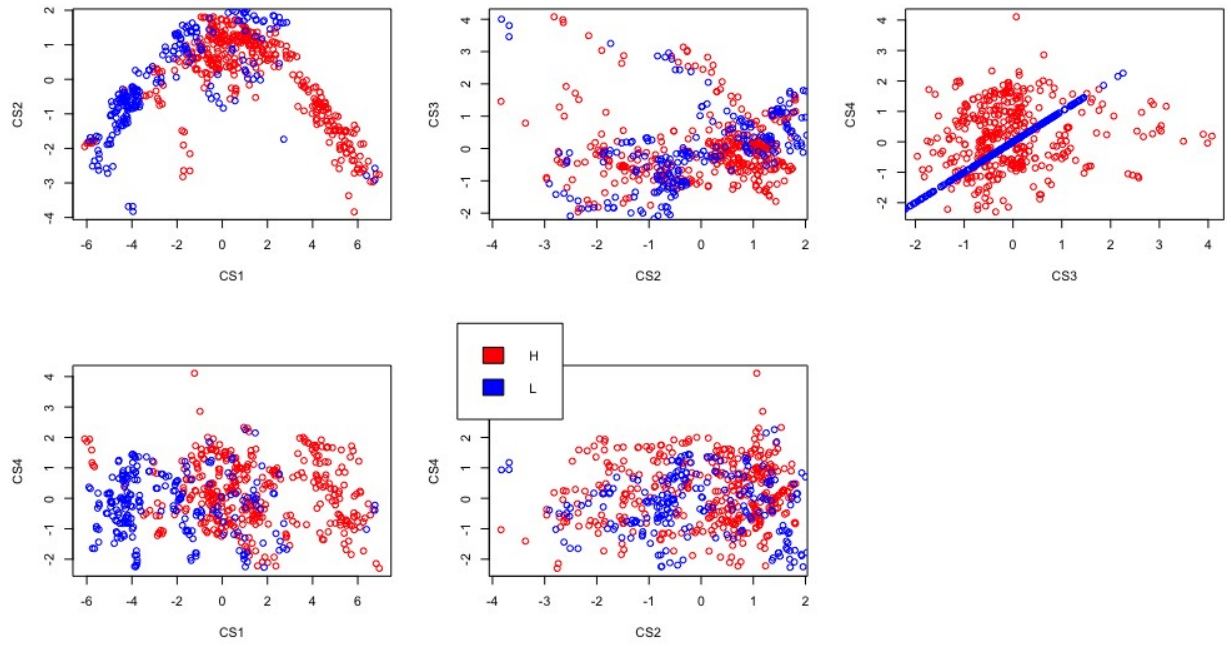


On remarque que la qualité de la projection de nos individus sur notre nouvel espace de dimension k ne semble pas réellement s'améliorer au delà de k = 4, ce qui correspond d'ailleurs au nombre de composantes nécessaires pour atteindre 80% de l'inertie totale expliquée.

Comme la 1ère valeur propre est très prépondérante par rapport aux autres, ne conserver que le premier axe ou le premier plan factoriel ne permet que d'expliquer la moitié de l'inertie du nuage, ce qui est trop peu pour être un choix raisonnable.

Nous décidons donc pour la suite de réduire notre nuage Rp à un espace à k = 4 dimensions.

Tracé des premiers axes factoriels



On remarque que le 1er plan factoriel est celui qui discrimine le mieux les variables selon le niveau d'ingénalité, bien que la qualité de la discrimination ne soit pas optimale. Remarquons également que nos 3ième et 4ième axes factoriels permettent d'expliquer linéairement la classe "L".

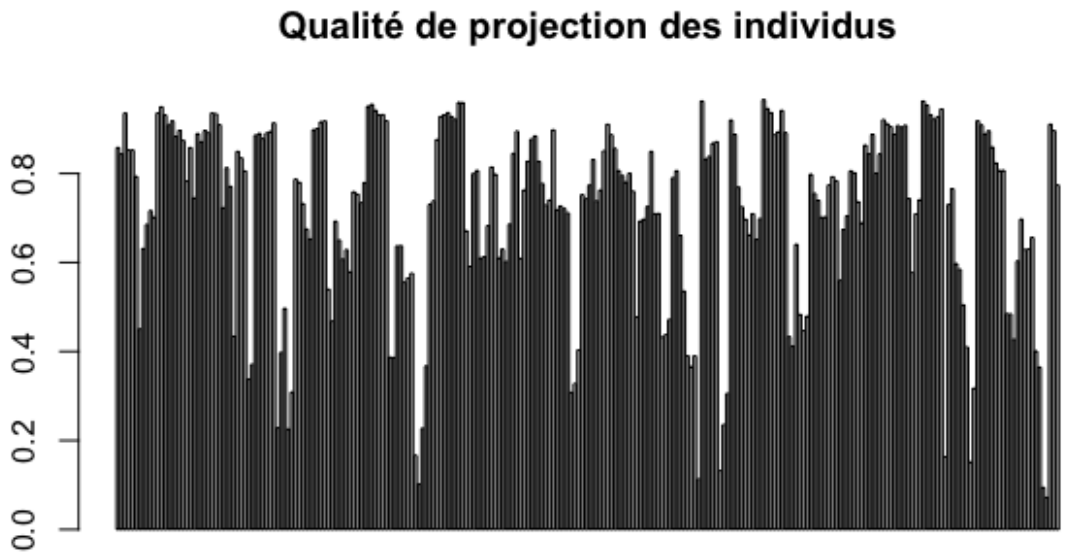
Travail sur les données TEST (30%)

Récupération des données et normalisation

```
testDataACP <- as.matrix(testData[,2:ncol(trainData)])
for (i in 1:ncol(testDataACP)) {
  testDataACP[,i] <- (testDataACP[,i] - mean(testDataACP[,i])) /
sd(testDataACP[,i])
}
```

Détermination des composantes de nos données test en utilisant les axes factoriels déterminés précédemment.

```
comptest <- testDataACP %*% acp$Vect_p
qtest <- quality_ACP(acp$Val_p, comptest, acp$Vect_p, norm=TRUE)
```



Une simple ligne de commande nous permet de récupérer les indices des 10 individus les moins bien projetés :

[211, 58, 57, 175, 210, 53, 54, 243, 25, 260].

On remarque que les individus les moins bien projetés sont ceux qui possèdent des valeurs extrêmes dans leurs coordonnées, soit ceux qui tendent à étirer le nuage de point malgré la normalisation.

Question 5 : AFD sur Rp

Données de train (70%)

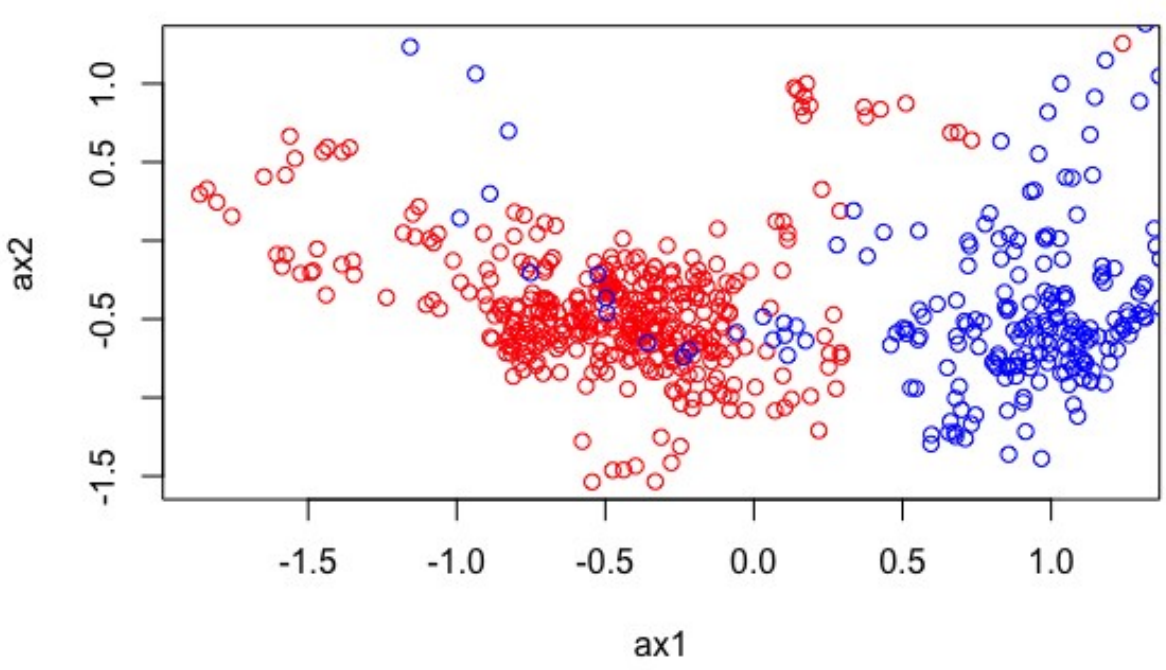
Récupération et ajustement des données train

```
val <- trainData[,1]
trainDataAFD <- trainData[,2:ncol(trainData)]
trainDataAFD <- cbind(trainDataAFD, val)
```

La fonction AFD permet de renvoyer directement les matrices B, W et V.

```
afd <- AFD(trainDataAFD, c("H", "L"), 20, diag_V=TRUE, norm=TRUE)
B <- afd$B
V <- afd$V
W <- afd$W
```

Comme nous n'avons qu'une seule variable qualitative divisée en 2 catégories "L" ou "H", un seul axe suffit pour décrire et séparer le nuage de points. En considérant cela-dit le premier plan factoriel, on obtient alors le graphe suivant :

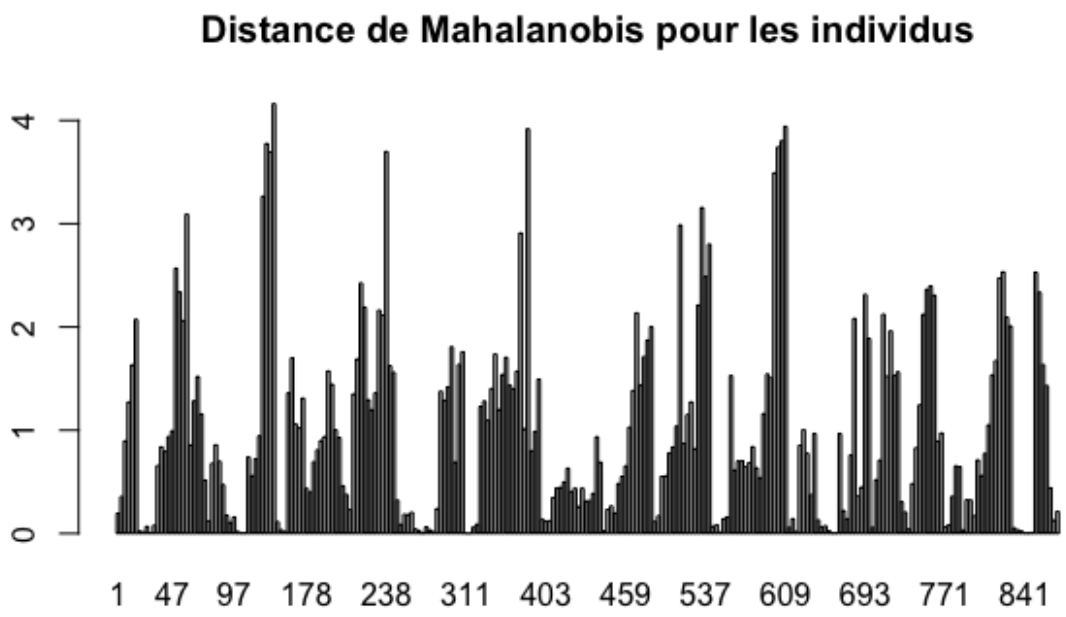


On remarque ici que, même s'il reste toutefois des superpositions de points, l'AFD nous permet de mieux catégoriser les données et de mieux les séparer que l'ACP précédente.

Données de test (30%)

En utilisant la fonction d'AFD lda de la librairie MASS, on obtient les résultats suivants :

```
val_test <- testData[,1]
afd_test_result <- MASS::lda( testData[,2:ncol(testData)], val_test)
print(afd_test_result)

scores <- predict(afd_test_result)$x

mahalanobis_distances <- mahalanobis(scores, colMeans(scores),
cov(scores))
order(mahalanobis_distances)[1:10]
```



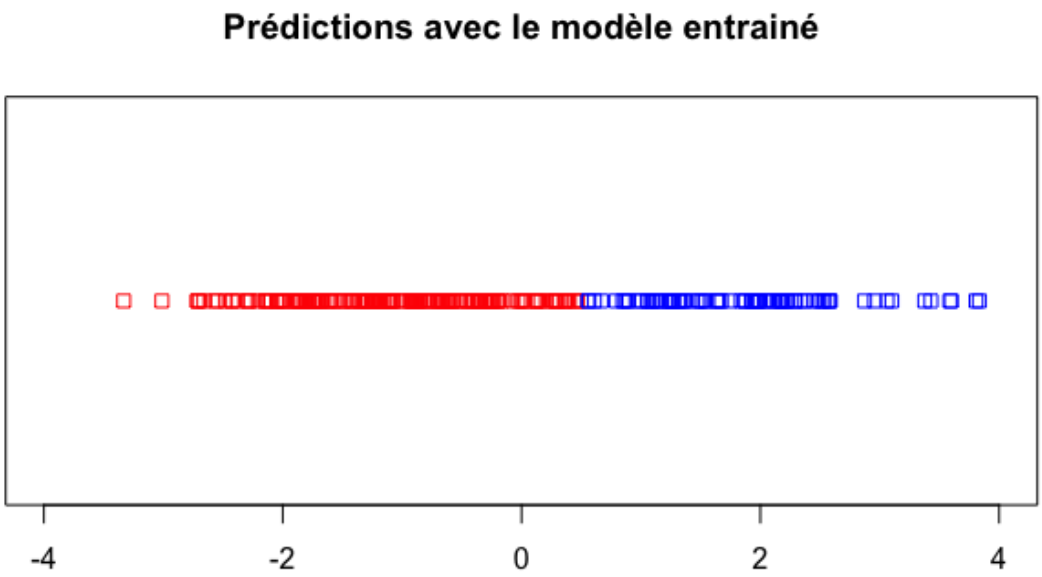
Les vecteurs les moins bien projetés sont donc les vecteurs d'indices : [98 97 198 252 199 85 35 36 251 10]

Question 6 : AFD prédictive

Fonction de discrimination

Puisque le modèle utilisé dans la partie 5 (a) provient de notre code d'ACP pas encore correctement terminé, nous avons décidé de ré-entrainé un autre modèle via la fonction LDA(MASS) sur les données train. C'est ce modèle que nous utilisons ici. On obtient alors la classification suivante, la couleur rouge correspondant à la classe H et la bleue à la classe L :

```
afd_predict <- MASS::lda(trainData[,2:ncol(trainData)], val)
predictions_afd <- predict(afd_predict, testData[,2:ncol(testData)])
print(predictions_afd$class)

H1 <- predictions_afd$x[predictions_afd$class=="H",]
L1 <- predictions_afd$x[predictions_afd$class=="L",]

stripchart(H1, col="red", xlim=c(-4,4), main="Prédictions avec le
modèle entrainé")
stripchart(L1, col="blue", add=TRUE)
```



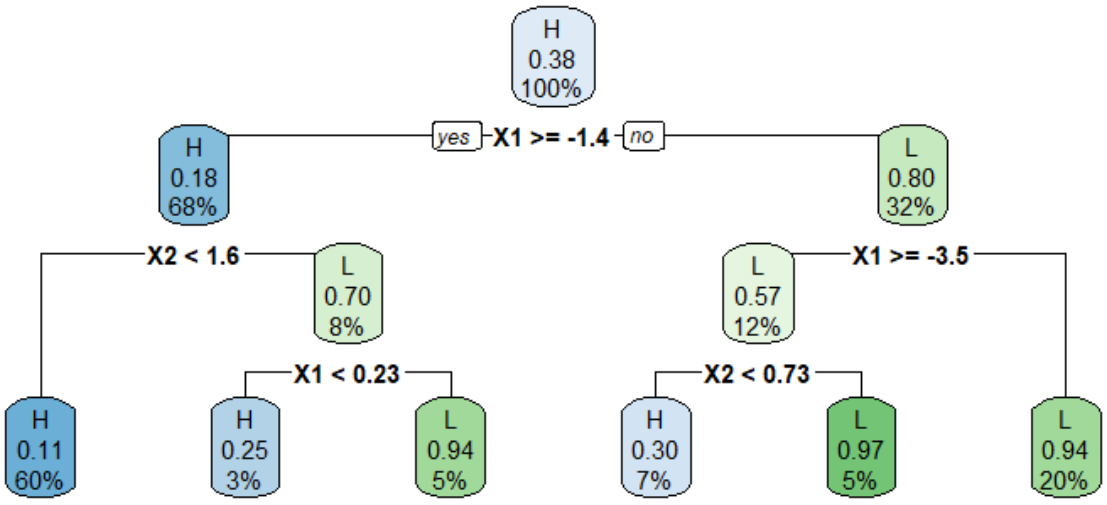

```
real_classes <- testData$Income_Inequality
goods <- 0
for (i in 1:length(real_classes)) {
  if (predictions$class[i] == real_classes[i]) {
    goods <- goods + 1
  }
}
cat("Accuracy :", goods/length(real_classes))

cm_afd <- confusionMatrix(predictions_afd$class,
testData$Income_Inequality)
print("Matrice de confusion - AFD:")
print(cm_afd)
```

Le modèle nous permet d'obtenir une score d'accuracy de 89,6%.

Arbre de décision avec les données de l'ACP

```
# Utilisation directe des scores des composantes principales
trainData_reduced <- acp$Comp[, 1:4]

# Ajout de la variable cible à ces données réduites
trainData_reduced <- data.frame(trainData_reduced)
trainData_reduced$Income_Inequality <- trainData$Income_Inequality

# Construction de l'arbre de décision
fitControl <- trainControl(method = "cv", number = 5)
```

```
fit <- train(Income_Inequality ~ ., data = trainData_reduced, method =
"rpart", trControl = fitControl, tuneGrid = data.frame(cp = seq(0.01,
0.5, 0.01)))
best_cp <- fit$bestTune$cp
tree <- rpart(Income_Inequality ~ ., data = trainData_reduced, cp =
best_cp)

# Visualisation de l'arbre
rpart.plot(tree)

# Prédictions sur l'ensemble de test réduit
# Utilisation directe des scores des composantes principales
testData_reduced <- comptest[,1:4]

# Ajout de la variable cible à ces données réduites
testData_reduced <- data.frame(testData_reduced)
predictions <- predict(tree, testData_reduced, type="class")

# Conversion des prédictions en facteurs
predictions <- factor(predictions,
levels=levels(testData$Income_Inequality))

# Matrice de confusion pour évaluer la performance

cm <- confusionMatrix(predictions, testData$Income_Inequality)
print(cm)
```



AFD prédictive

```
# Ajout de la variable cible à ces données réduites
trainData_reduced$Income_Inequality <- trainData$Income_Inequality
testData_reduced$Income_Inequality <- testData$Income_Inequality
```

```r
# Construction du modèle AFD
afd_model <- MASS::lda(Income_Inequality ~ ., data =
trainData_reduced)

# Prédictions AFD sur l'ensemble de test réduit
predictions_afd_reduced <- predict(afd_model, testData_reduced[, -
ncol(testData_reduced)])$class
predictions_afd_reduced <- factor(predictions_afd_reduced,
levels=levels(testData_reduced$Income_Inequality))

# Matrice de confusion pour l'AFD
cm_afd_reduced <- confusionMatrix(predictions_afd_reduced,
testData_reduced$Income_Inequality)
print("Matrice de confusion - AFD sur données réduites:")
print(cm_afd_reduced)
```

Comparaison abre de décision et AFD prédictive sur les données de l'ACP

- Précision: L'arbre de décision est plus précis que l'AFD prédictive.
- Sensibilité: L'arbre de décision a une meilleure capacité à identifier correctement les cas "H".
- Spécificité: Similaire pour les deux méthodes, mais légèrement meilleure pour l'arbre de décision.
- Valeurs Prédictives: L'arbre de décision a de meilleures valeurs prédictives positives et négatives.
- Kappa: L'arbre de décision montre un meilleur accord que l'AFD prédictive.
- Balanced Accuracy: L'arbre de décision présente une meilleure performance globale en termes d'équilibre entre la sensibilité et la spécificité.

Dans l'ensemble, l'arbre de décision surpasse l'AFD prédictive sur la plupart des métriques clés. Cela suggère que pour vos données spécifiques et le type de réduction de dimensionnalité appliqué (ACP), l'arbre de décision est mieux adapté pour capturer les relations complexes et fournir des prédictions plus précises que l'AFD prédictive.

Comparaison avec l'arbre de décision de la partie 2

La performance globale du modèle initial est supérieure à celle du modèle utilisant les 4 composantes de l'ACP. Cela peut suggérer que la réduction de la dimensionnalité via l'ACP a entraîné une perte d'informations cruciales pour la classification, impactant négativement la performance des modèles basés sur l'ACP.

Seule la profondeur de l'arbre (et les variables utilisées) a changé.

Comparaison avec l'AFD de la question 5

L'AFD prédictive initiale, sans la réduction de dimensionnalité via ACP, surpasse les modèles utilisant les composantes de l'ACP en termes d'exactitude globale, de sensibilité, de spécificité, et des valeurs prédictives. Cela suggère que, pour nos données, l'approche sans réduction de dimensionnalité est plus efficace.

La réduction de dimensionnalité via l'ACP, bien que bénéfique pour simplifier le modèle et réduire la complexité, peut entraîner une perte d'informations cruciales pour la classification, ce qui se reflète dans la performance inférieure des modèles basés sur l'ACP.