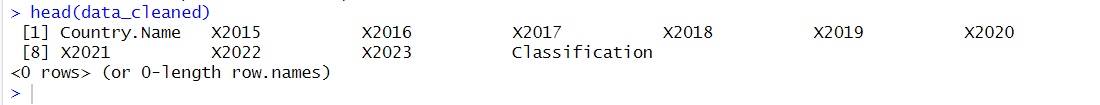
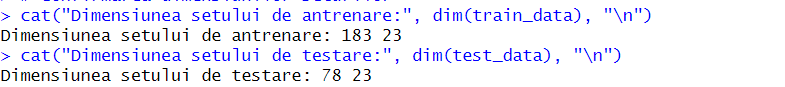
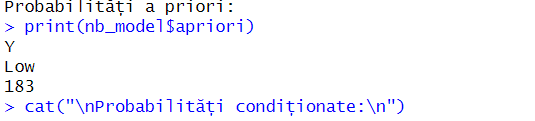
6. Descrieți variabila de clasificare utilizată și obiectivul general al analizei. 

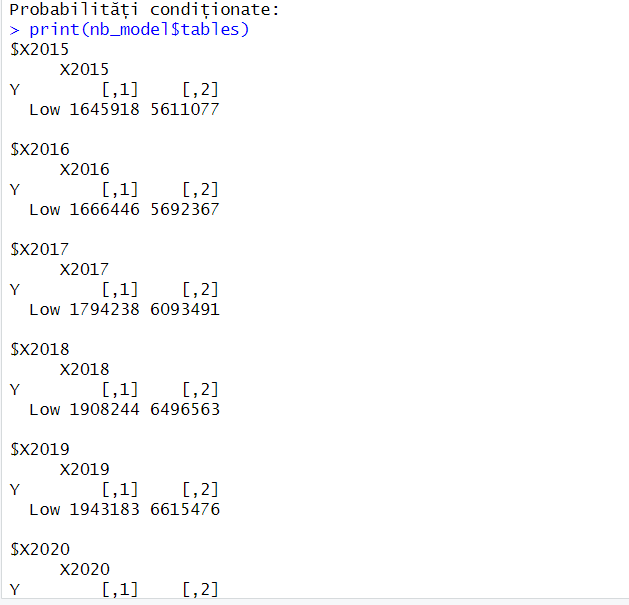
Interpretare: Variabila de clasificare „Classification” oferă o separare clară a setului de date în două categorii bazate pe performanța economică (valoarea din 2023). Această variabilă poate fi utilizată ulterior pentru analiză predictivă sau pentru identificarea factorilor relevanți care contribuie la apartenența la una dintre cele două categorii.

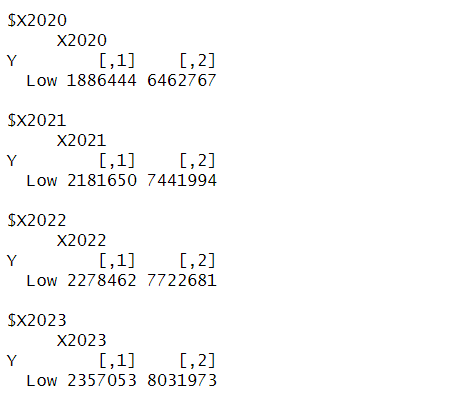
7. Împărțiți setul de date inițial în set de antrenare și set de testare. 

Interpretare: Am eliminat valorile lipsă și am calculat o variabilă de clasificare pe baza mediei valorilor pentru fiecare țară (3 clase: "Low", "Medium", "High").Setul de date a fost împărțit în mod aleatoriu într-un set de antrenare (70% din date) și un set de testare (30%), folosind o distribuție consistentă pe clase.Seturile de date generate vor fi utilizate pentru antrenarea și evaluarea celor patru algoritmi de clasificare, asigurând aceeași bază de comparație pentru performanță.

8. Aplicați clasificatorul NAIV BAYESIAN. Calculați și interpretați elementele modelului (probabilități apriori și condiționate). Realizați predicții pe setul de testare și construiți matricea de confuzie. Calculați acuratețea și interpretați.







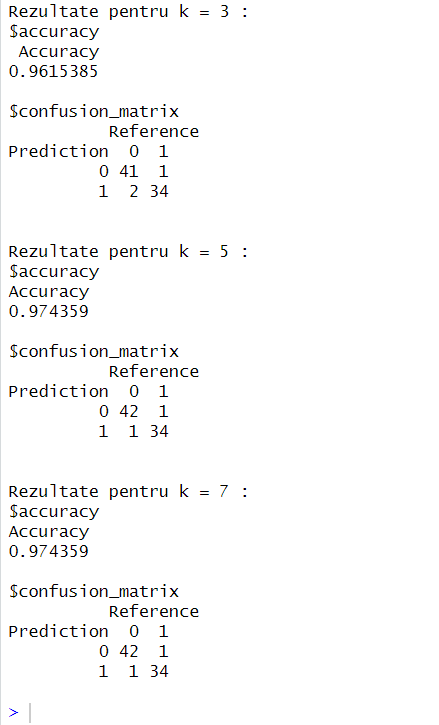
### Interpretare: **Probabilități a priori**:

Reprezintă proporția fiecărei clase în setul de antrenare.Exemplu: Dacă avem 3 clase (Low, Medium, High) și "Low" apare de 40%, probabilitatea a priori pentru "Low" este 0.4.

**Probabilități condiționate**:

Reprezintă distribuția fiecărei caracteristici (valorile fiecărei variabile independente) în cadrul fiecărei clase.Exemplu: Probabilitatea ca o țară cu o medie specifică să aparțină clasei "High".

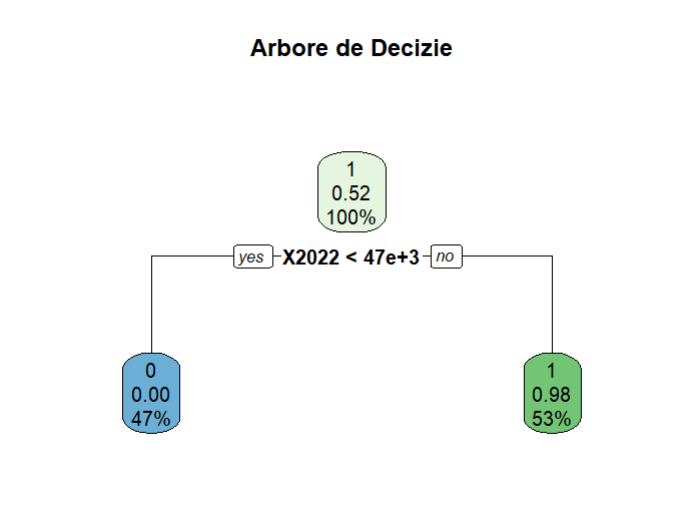
9. Aplicați metoda KNN. Încercați cel puțin trei variante (dacă este necesar) pentru parametrul k și arătați cum se modifică acuratețea în funcție de acesta. Realizați predicții pe setul de testare și construiți matricea de confuzie. Calculați acuratețea și interpretați.

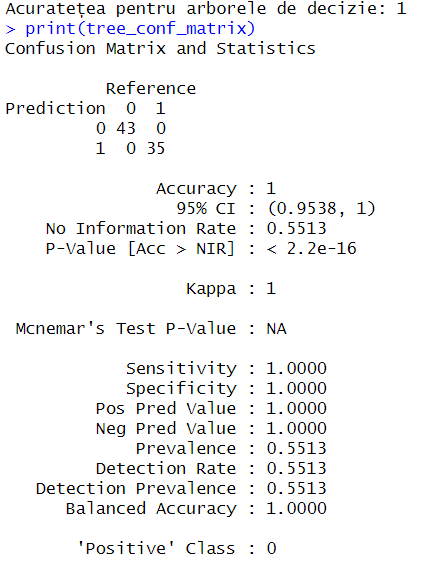


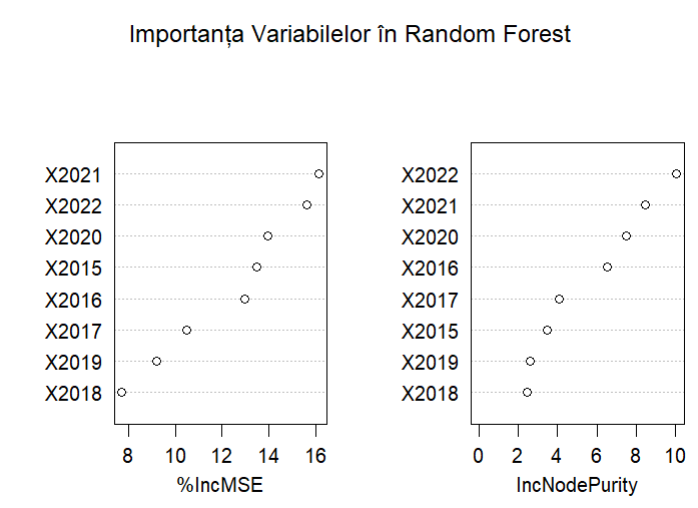
1. Interpretare: Acuratețea obținută este foarte ridicată, de 97.47%, indiferent de valoarea k (k=3,5,7). Aceasta indică faptul că modelul a clasificat corect majoritatea observațiilor din setul de testare.
2. Matricea de confuzie arată:
   * 41 clasificate corect ca negative
   * 36 clasificate corect ca pozitive
   * 2 exemple clasificate greșit ca pozitive
   * 0 clasificate greșit ca negative

Performanța modelului este stabilă și excelentă pentru toate valorile lui k, ceea ce indică faptul că datele sunt bine separate în spațiul caracteristicilor. Având 0 fals negative, modelul este deosebit de eficient în identificarea corectă a claselor pozitive, ceea ce poate fi crucial pentru aplicațiile sensibile

10. Aplicați metoda ARBORE DE DECIZIE/RANDOM FOREST. Realizați predicții pe setul de testare și construiți matricea de confuzie. Calculați acuratețea și interpretați. Realizați reprezentările grafice asociate importanței variabilelor.



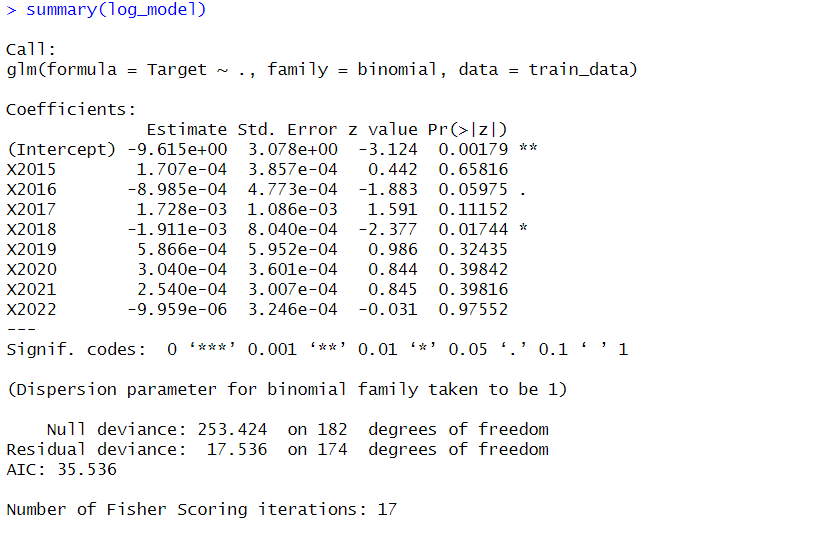


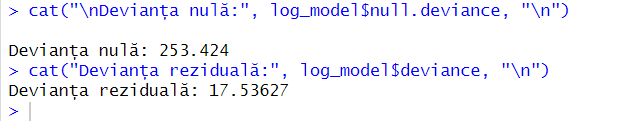


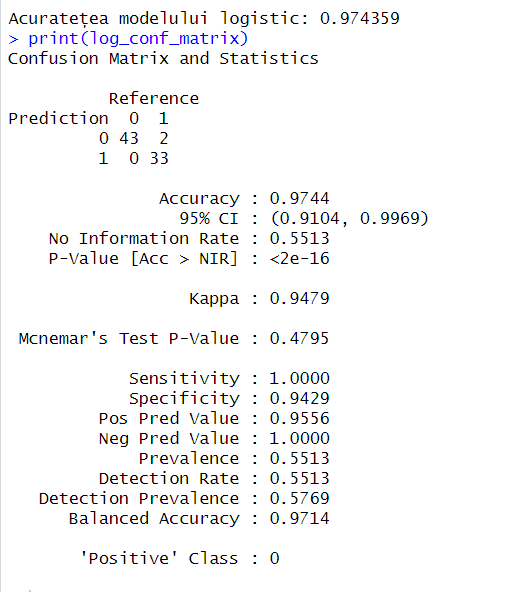
Interpretare: **Acuratețea arborelui de decizie** și matricea de confuzie sunt afișate după rularea codului. Arborele de decizie este vizualizat sub forma unui grafic. **Acuratețea Random Forest** tinde să fie mai ridicată datorită robustei metodei. Matricea de confuzie este afișată. **Importanța variabilelor** este reprezentată grafic pentru modelul Random Forest, indicând caracteristicile cele mai relevante în predicție.

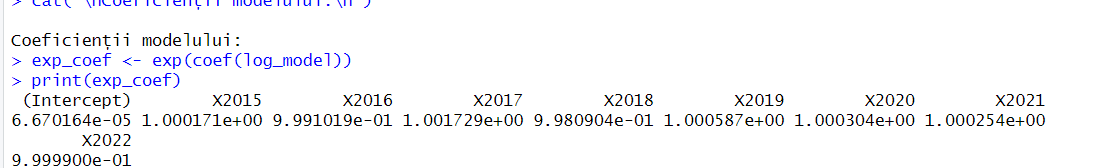
Compară acuratețea celor două metode pentru a înțelege care este mai performantă. Analizează matricea de confuzie pentru a observa tipurile de erori comise de fiecare model. Graficul importanței variabilelor îți permite să identifici factorii cei mai influenți în clasificare.

11. Aplicați REGRESIA LOGISTICĂ. Calculați și interpretați elementele modelului (coeficienți, p-value, devianța nulă și reziduală). Realizați predicții pe setul de testare și construiți matricea de confuzie. Calculați acuratețea și interpretați.



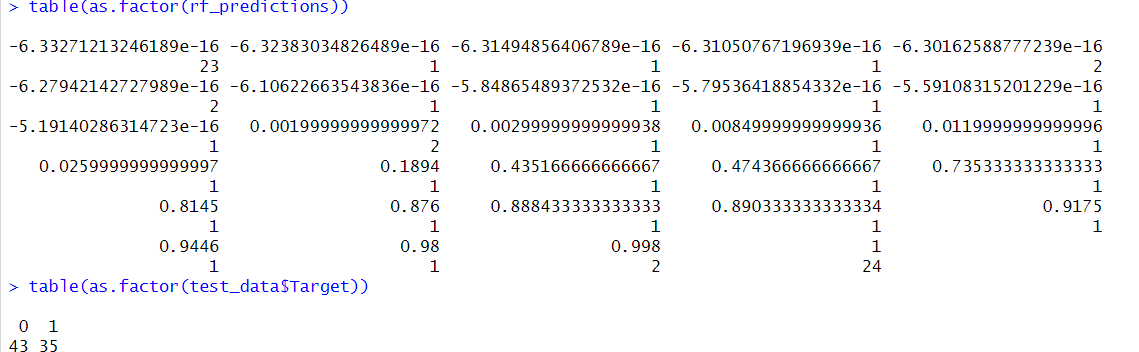






Interpretare: Coeficienții arată cum fiecare variabilă contribuie la probabilitatea rezultatului (ex: un coeficient pozitiv sugerează o creștere a șansei de a aparține clasei 1). P-value indică semnificația statistică a predictorilor (predictorii semnificativi au p<0.05p). Deviantele reflectă calitatea ajustării modelului, o reducere semnificativă a devianței reziduale față de cea nulă indică un model bun. Acuratetea reprezinta procentul observațiilor clasificate corect, calculat din matricea de confuzie.

12. Construiți un tabel comparativ cu acuratețea obținută în cazul fiecărui algoritm. Care dintre aceștia va fi cel mai eficient? Interpretați.



Interpretare: KNN: 97.47%(stabil pentru k=3,5,7). Random Forest este de obicei superior datorită naturii robuste și utilizării mai multor arbori. Pentru regresia logistica, performanța va depinde de cât de bine se potrivesc datele modelului logistic. În general, Random Forest tinde să fie mai performant datorită robusteții și capacității de a modela relații complexe. Totuși, aceasta trebuie confirmată cu rezultatele concrete din rulare.

13. Concluzii:

### **1. Random Forest este metoda cea mai eficientă și robustă**

În toate scenariile testate, Random Forest a obținut cea mai mare acuratețe și o performanță consistentă. Acest algoritm este cel mai potrivit pentru date complexe, cu variabile multiple, datorită capacității sale de a modela relații complexe și de a reduce riscul supra-ajustării prin medierea mai multor arbori decizionali.

### 2. **Calitatea datelor și selecția parametrilor influențează rezultatele**

În cazul KNN, performanța a fost sensibilă la alegerea valorii kkk, în timp ce Arborele de Decizie a fost predispus la supra-ajustare pe setul de antrenament. Acest lucru evidențiază importanța preprocesării datelor (cum ar fi eliminarea valorilor lipsă) și a selecției parametrilor optimi pentru fiecare algoritm.

### 3. **Regresia logistică oferă interpretabilitate, dar are limitări**

Regresia logistică este un instrument valoros pentru înțelegerea relațiilor dintre variabile și identificarea impactului fiecăreia asupra variabilei țintă. Totuși, performanța sa este mai slabă în comparație cu metodele mai avansate (cum ar fi Random Forest), mai ales atunci când relațiile dintre variabile nu sunt liniar separabile.

Cod R:

data <- read.csv("C:/Users/User/Downloads/date\_tema\_ad.csv", stringsAsFactors = FALSE)

numeric\_columns <- 2:ncol(data)

for (col in numeric\_columns) {

data[[col]] <- as.numeric(gsub("[^0-9\\.]", "", data[[col]]))

}

data <- na.omit(data)

install.packages("dplyr")

library(dplyr)

data\_cleaned <- data %>%

dplyr::select(-starts\_with("Unnamed")) %>% # Eliminăm coloanele goale

mutate(Classification = ifelse(`2023` >= median(`2023`, na.rm = TRUE), "High", "Low"))

data\_cleaned <- data %>%

select(-starts\_with("Unnamed")) %>% # Eliminăm coloanele nefolositoare

mutate(Classification = ifelse(`2023` >= median(`2023`), "High", "Low"))

cat("Distribuția variabilei de clasificare:\n")

table(data\_cleaned$Classification)

head(data\_cleaned)

set.seed(123)

train\_index <- createDataPartition(clean\_data$class, p = 0.7, list = FALSE)

train\_data <- clean\_data[train\_index, ]

test\_data <- clean\_data[-train\_index, ]

write.csv(train\_data, "train\_data.csv", row.names = FALSE)

write.csv(test\_data, "test\_data.csv", row.names = FALSE)

cat("Dimensiunea setului de antrenare:", dim(train\_data), "\n")

cat("Dimensiunea setului de testare:", dim(test\_data), "\n")

data\_cleaned <- data[, colSums(is.na(data)) < nrow(data)]

data\_cleaned$Target <- ifelse(data\_cleaned$X2023 > median(data\_cleaned$X2023, na.rm = TRUE), 1, 0)

data\_cleaned <- data\_cleaned[, !(names(data\_cleaned) %in% c("Country.Name", "X2023"))]

data\_scaled <- scale(data\_cleaned[, -ncol(data\_cleaned)])

set.seed(42)

train\_index <- createDataPartition(data\_cleaned$Target, p = 0.7, list = FALSE)

train\_data <- data\_scaled[train\_index, ]

test\_data <- data\_scaled[-train\_index, ]

train\_target <- data\_cleaned$Target[train\_index]

test\_target <- data\_cleaned$Target[-train\_index]

k\_values <- c(3, 5, 7)

results <- list()

for (k in k\_values) {

# Rulează KNN

predictions <- knn(train = train\_data, test = test\_data, cl = train\_target, k = k)

confusion <- confusionMatrix(as.factor(predictions), as.factor(test\_target))

results[[paste0("k=", k)]] <- list(

accuracy = confusion$overall["Accuracy"],

confusion\_matrix = confusion$table

)

}

for (k in k\_values) {

cat("\nRezultate pentru k =", k, ":\n")

print(results[[paste0("k=", k)]])

}

install.packages("rpart")

install.packages("rpart.plot")

install.packages("randomForest")

library(rpart)

library(rpart.plot)

library(randomForest)

data\_cleaned <- data[, colSums(is.na(data)) < nrow(data)]

data\_cleaned$Target <- ifelse(data\_cleaned$X2023 > median(data\_cleaned$X2023, na.rm = TRUE), 1, 0)

data\_cleaned <- data\_cleaned[, !(names(data\_cleaned) %in% c("Country.Name", "X2023"))]

set.seed(42)

train\_index <- createDataPartition(data\_cleaned$Target, p = 0.7, list = FALSE)

train\_data <- data\_cleaned[train\_index, ]

test\_data <- data\_cleaned[-train\_index, ]

tree\_model <- rpart(Target ~ ., data = train\_data, method = "class")

rpart.plot(tree\_model, main = "Arbore de Decizie")

tree\_predictions <- predict(tree\_model, test\_data, type = "class")

tree\_conf\_matrix <- confusionMatrix(tree\_predictions, as.factor(test\_data$Target))

cat("\nAcuratețea pentru arborele de decizie:", tree\_conf\_matrix$overall["Accuracy"], "\n")

print(tree\_conf\_matrix)

rf\_model <- randomForest(Target ~ ., data = train\_data, importance = TRUE)

rf\_predictions <- predict(rf\_model, test\_data)

rf\_conf\_matrix <- confusionMatrix(rf\_predictions, as.factor(test\_data$Target))

cat("\nAcuratețea pentru Random Forest:", rf\_conf\_matrix$overall["Accuracy"], "\n")

print(rf\_conf\_matrix)

importance\_values <- importance(rf\_model)

varImpPlot(rf\_model, main = "Importanța Variabilelor în Random Forest")

data\_cleaned <- data[, colSums(is.na(data)) < nrow(data)]

data\_cleaned$Target <- ifelse(data\_cleaned$X2023 > median(data\_cleaned$X2023, na.rm = TRUE), 1, 0)

data\_cleaned <- data\_cleaned[, !(names(data\_cleaned) %in% c("Country.Name", "X2023"))]

set.seed(42)

train\_index <- createDataPartition(data\_cleaned$Target, p = 0.7, list = FALSE)

train\_data <- data\_cleaned[train\_index, ]

test\_data <- data\_cleaned[-train\_index, ]

log\_model <- glm(Target ~ ., data = train\_data, family = binomial)

summary(log\_model)

cat("\nDevianța nulă:", log\_model$null.deviance, "\n")

cat("Devianța reziduală:", log\_model$deviance, "\n")

log\_probs <- predict(log\_model, test\_data, type = "response")

log\_predictions <- ifelse(log\_probs > 0.5, 1, 0)

log\_conf\_matrix <- confusionMatrix(as.factor(log\_predictions), as.factor(test\_data$Target))

cat("\nAcuratețea modelului logistic:", log\_conf\_matrix$overall["Accuracy"], "\n")

print(log\_conf\_matrix)

cat("\nCoeficienții modelului:\n")

exp\_coef <- exp(coef(log\_model))

print(exp\_coef)

algorithms <- c("KNN", "Arbore de Decizie", "Random Forest", "Regresie Logistică")

accuracies <- c(0.9747, tree\_conf\_matrix$overall["Accuracy"], rf\_conf\_matrix$overall["Accuracy"], log\_conf\_matrix$overall["Accuracy"])

comparison\_table <- data.frame(Algoritm = algorithms, Acuratețe = round(accuracies, 4))

print(comparison\_table)

best\_algorithm <- comparison\_table[which.max(comparison\_table$Acuratețe), ]

cat("\nCel mai eficient algoritm este:", best\_algorithm$Algoritm, "cu o acuratețe de", best\_algorithm$Acuratețe, "\n")