ModeleRF_11353138

Cristian Samson

2024-11-22

R Markdown

```
# Clear the environment
rm(list = ls())

# Charger les bibliothèques nécessaires
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(rpart)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(stringr) # Pour utiliser str_extract

#== PRETRAITEMENT DES DONNÉES ==#

# Lire les données
repairs <- read.csv("C:/Users/samso/OneDrive/Bureau/Apprentissage statistique/Devoir 1/repairs.csv")
sensors_study <- read.csv("C:/Users/samso/OneDrive/Bureau/Apprentissage statistique/Devoir 1/sensors-stu

# Reshaper les données avec un ID par lignes
sensors_study_reshaped <- sensors_study %>%
```

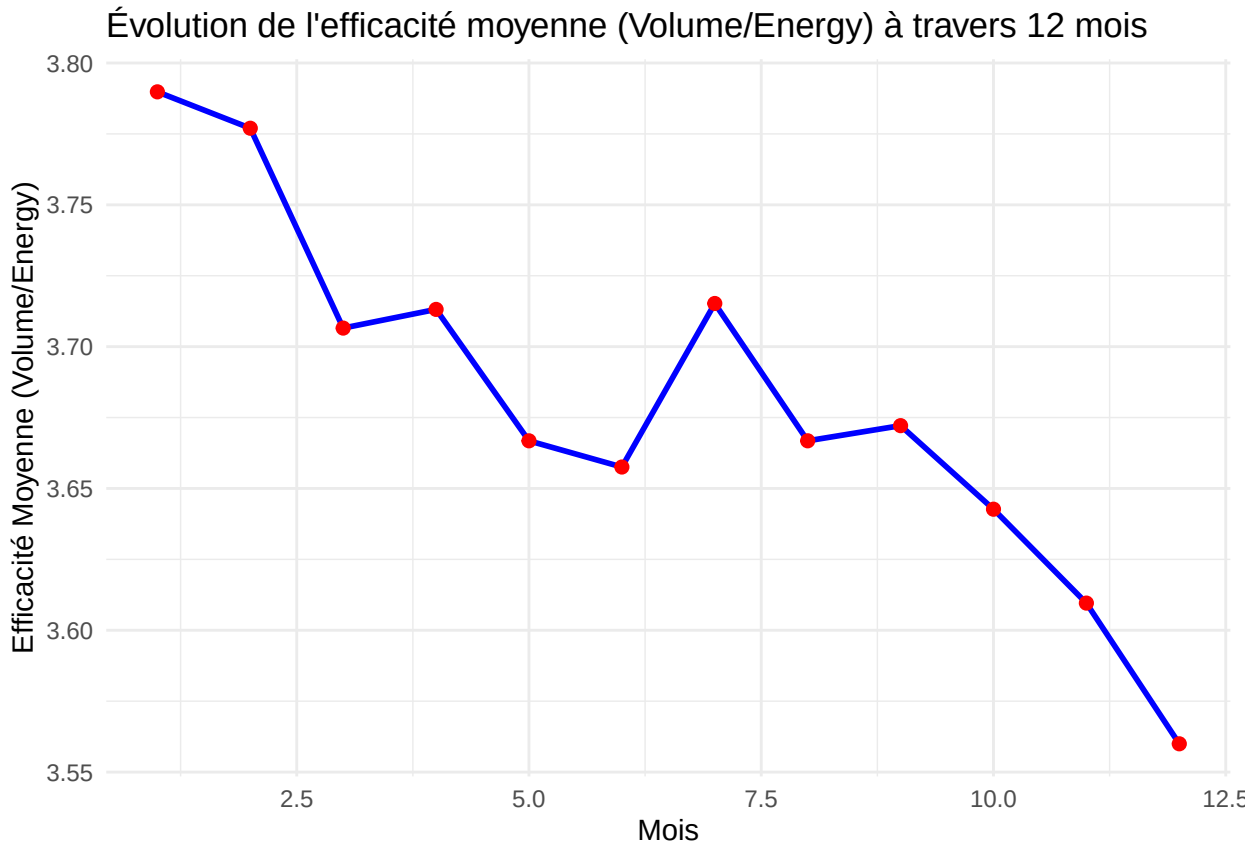
```r
  pivot_wider(names_from = Month, values_from = c(Volume, Energy, starts_with("PSD"))) %>%
  rename_with(~ gsub("_", "", .), starts_with("Volume_")) %>%
  rename_with(~ gsub("_", "", .), starts_with("Energy_")) %>%
  rename_with(~ gsub("_", "", .), starts_with("PSD"))

# Calcul des colonnes d'efficacité pour chaque mois
# Étape 1 : Créer une liste des colonnes Volume et Energy
volume_cols <- grep("^Volume\\d+$", names(sensors_study_reshaped), value = TRUE)
energy_cols <- gsub("Volume", "Energy", volume_cols)

# Étape 2 : Ajouter les colonnes d'efficacité
for (i in seq_along(volume_cols)) {
  sensors_study_reshaped <- sensors_study_reshaped %>%
    mutate(!!paste0("Efficacite", str_extract(volume_cols[i], "\\d+$")) :=
             !!sym(volume_cols[i]) / !!sym(energy_cols[i]))
}

# Nettoyer et fusionner les données
repairs$Cost6[is.na(repairs$Cost6)] <- 0

study_data <- sensors_study_reshaped %>%
  left_join(repairs, by = "ID") %>%
  mutate(Treatment = ifelse(Cost6 > 0, 1, 0))

#== ANALYSE EXPLORATOIRE ==#

#1. Visualiser la courbe de l'efficacité moyenne (Volume/Energy) à travers les 12 mois

# Calculer l'efficacité moyenne pour chaque mois
efficiency_columns <- grep("^Efficacite\\d+$", names(study_data), value = TRUE)
mean_efficiency <- study_data %>%
  summarise(across(all_of(efficiency_columns), mean, na.rm = TRUE))
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(all_of(efficiency_columns), mean, na.rm = TRUE)`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```r
# Transformer les données en format long pour ggplot
mean_efficiency_long <- mean_efficiency %>%
  pivot_longer(cols = everything(),
               names_to = "Mois",
               values_to = "Efficacite") %>%
  mutate(Mois = as.numeric(gsub("Efficacite", "", Mois)))

# Créer le graphique
```

```r
ggplot(mean_efficiency_long, aes(x = Mois, y = Efficacite)) +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "red", size = 2) +
  labs(title = "Évolution de l'efficacité moyenne (Volume/Energy) à travers 12 mois",
       x = "Mois",
       y = "Efficacité Moyenne (Volume/Energy)") +
  theme_minimal()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```




```r
#== Test statistique (ANOVA ou autre) pour évaluer si le pic à 6 mois est significativement différent d
# Identifier les colonnes d'efficacité
efficiency_columns <- grep("^Efficacite\\d+$", names(study_data), value = TRUE)

# Reshaper les données pour ANOVA
anova_data <- study_data %>%
  select(ID, all_of(efficiency_columns)) %>%
  pivot_longer(cols = -ID, names_to = "Mois", values_to = "Efficacite") %>%
  mutate(Mois = as.numeric(gsub("Efficacite", "", Mois)))
```

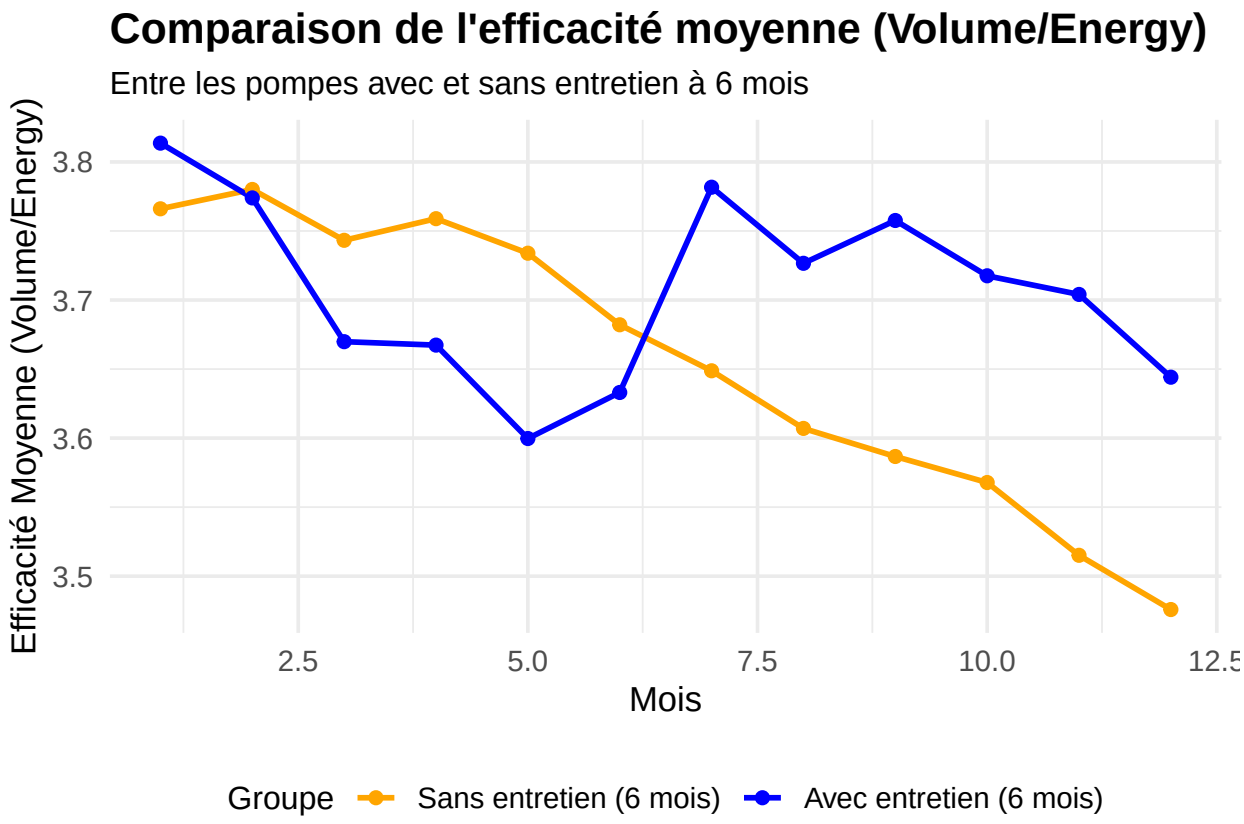
```r
# Effectuer le test ANOVA
anova_result <- aov(Efficacite ~ as.factor(Mois), data = anova_data)
summary(anova_result)
```

```
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## as.factor(Mois)   11   23.1  2.0982   6.106 4.86e-10 ***
## Residuals       5988 2057.8  0.3437
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Optionnel : Afficher la p-value
p_value <- summary(anova_result)[[1]]$`Pr(>F)`[1]
cat("P-value : ", p_value, "\n")
```

```
## P-value :  4.855077e-10
```

```r
#La p-value est bien inférieure à 0.05,
#indiquant que les différences d'efficacité entre les mois sont statistiquement significatives.

#Visualiser cette courbe d'efficacité à travers l'année pour le groupe traitement et le groupe contrôle

# Calculer l'efficacité moyenne pour chaque groupe (avec et sans entretien)
comparison_data <- study_data %>%
  pivot_longer(cols = all_of(efficiency_columns),
               names_to = "Mois",
               values_to = "Efficacite") %>%
  mutate(Mois = as.numeric(gsub("Efficacite", "", Mois))) %>%
  group_by(Treatment, Mois) %>%
  summarise(Efficacite_Moyenne = mean(Efficacite, na.rm = TRUE), .groups = "drop")

# Créer le graphique
ggplot(comparison_data, aes(x = Mois, y = Efficacite_Moyenne, color = as.factor(Treatment))) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  scale_color_manual(values = c("orange", "blue"),
                     labels = c("Sans entretien (6 mois)", "Avec entretien (6 mois)")) +
  labs(
    title = "Comparaison de l'efficacité moyenne (Volume/Energy)",
    subtitle = "Entre les pompes avec et sans entretien à 6 mois",
    x = "Mois",
    y = "Efficacité Moyenne (Volume/Energy)",
    color = "Groupe"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "bottom",
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 11),
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 12),
    axis.title = element_text(size = 13),
    axis.text = element_text(size = 11)
  )
```




```
#== TROUVER LA PROPORTION DES POMPES (AVEC ENTRETIEN) QUI ONT UNE EFFICACITÉ CROISSANTE APRÈS LE 6IEM M

# Identifier les pompes avec entretien
treated_data <- study_data %>%
  filter(Treatment == 1)

# Vérifier les colonnes pour le 6ème et le 7ème mois
eff_month_6 <- "Efficacite6"
eff_month_7 <- "Efficacite7"

# Calculer la proportion de pompes avec une augmentation d'efficacité après le 6ème mois
treated_growth <- treated_data %>%
  filter(!!sym(eff_month_7) > !!sym(eff_month_6))

proportion_growth <- nrow(treated_growth) / nrow(treated_data)

# Afficher le résultat
cat("Proportion des pompes avec une augmentation d'efficacité après le 6ème mois :", proportion_growth,
```

```
## Proportion des pompes avec une augmentation d'efficacité après le 6ème mois : 0.548
```

```
#== TROUVER LA PROPORTION DES POMPES (AVEC ENTRETIENT) AVEC UNE AUGMENTATION D'EFFICACITÉ APRÈS LE 6ÈME
# Identifier les pompes avec entretien
treated_data <- study_data %>%
  filter(Treatment == 1)
```

```r
# Vérifier les colonnes pour le 6ème et le 7ème mois
eff_month_6 <- "Efficacite6"
eff_month_7 <- "Efficacite7"

# Filtrer les pompes avec une augmentation d'efficacité après le 6ème mois
treated_growth <- treated_data %>%
  filter(!!sym(eff_month_7) > !!sym(eff_month_6))

# Filtrer celles ayant un coût d'entretien de 500 au 12ème mois
treated_growth_with_cost <- treated_growth %>%
  filter(Cost12 == 500)

# Calculer la proportion
proportion_growth_with_cost <- nrow(treated_growth_with_cost) / nrow(treated_growth)

# Afficher le résultat
cat("Proportion des pompes avec augmentation d'efficacité et un coût d'entretien de 500 au 12ème mois :
    proportion_growth_with_cost, "\n")
```

```
## Proportion des pompes avec augmentation d'efficacité et un coût d'entretien de 500 au 12ème mois : 0
```

```r
#== CRÉER LA VARIABLE CIBLE ===

# Identifier les colonnes pertinentes pour l'efficacité
# Identifier les colonnes pertinentes pour l'efficacité
efficiency_columns <- grep("^Efficacite", names(study_data), value = TRUE)

# Calculer les seuils basés sur le dataset
cost_threshold <- 750  # 3ème quartile pour Cost12 basé sur l'analyse précédente
efficiency_threshold <- study_data %>%
  select(all_of(efficiency_columns)) %>%
  summarise(mean_efficiency = mean(unlist(across(everything())), na.rm = TRUE)) %>%
  pull(mean_efficiency)

# Vérifier que le seuil est bien calculé
if (is.na(efficiency_threshold)) {
  stop("Le calcul de efficiency_threshold a échoué. Vérifiez les données d'entrée.")
}

# Créer la variable cible "Entretien_Necessaire" avec une gestion explicite des NAs
study_data <- study_data %>%
  mutate(
    Entretien_Necessaire = ifelse(
      # Condition 1: Diminution significative de l'efficacité après le 6ème mois
      (!is.na(Treatment) & Treatment == 0 &
         !is.na(Efficacite7) & !is.na(Efficacite6) & Efficacite7 < Efficacite6) |
        # Condition 2: Coût projeté élevé au 12ème mois
        (!is.na(Cost12) & Cost12 > cost_threshold) |
        # Condition 3: Efficacité moyenne inférieure au seuil global pour les 5 premiers mois
        (rowMeans(select(., all_of(efficiency_columns[1:5])), na.rm = TRUE) < efficiency_threshold),
      1, 0
    )
```

```
  )

# Vérifier la distribution de la variable cible
target_distribution <- study_data %>%
  count(Entretien_Necessaire)

print(target_distribution)
```

```
## # A tibble: 2 x 2
##   Entretien_Necessaire     n
##                  <dbl> <int>
## 1                    0   213
## 2                    1   287
```

```
# Identifier les colonnes pertinentes pour les 5 premiers mois
filtered_columns <- grep("^(Volume[1-5]$|Energy[1-5]$|Efficacite[1-5]$|PSD[0-9]+0[1-5]$)", names(study_

# Ajouter la variable cible "Entretien_Necessaire" aux colonnes sélectionnées
filtered_columns <- c(filtered_columns, "Entretien_Necessaire")

# Créer un dataset filtré
study_data_filtered <- study_data[, filtered_columns]

# Vérifier les colonnes filtrées pour validation
print(names(study_data_filtered))
```

```
##  [1] "Volume1"             "Volume2"             "Volume3"
##  [4] "Volume4"             "Volume5"             "Energy1"
##  [7] "Energy2"             "Energy3"             "Energy4"
## [10] "Energy5"             "PSD5001"             "PSD5002"
## [13] "PSD5003"             "PSD5004"             "PSD5005"
## [16] "PSD7501"             "PSD7502"             "PSD7503"
## [19] "PSD7504"             "PSD7505"             "PSD10001"
## [22] "PSD10002"            "PSD10003"            "PSD10004"
## [25] "PSD10005"            "PSD12501"            "PSD12502"
## [28] "PSD12503"            "PSD12504"            "PSD12505"
## [31] "PSD15001"            "PSD15002"            "PSD15003"
## [34] "PSD15004"            "PSD15005"            "PSD17501"
## [37] "PSD17502"            "PSD17503"            "PSD17504"
## [40] "PSD17505"            "PSD20001"            "PSD20002"
## [43] "PSD20003"            "PSD20004"            "PSD20005"
## [46] "PSD22501"            "PSD22502"            "PSD22503"
## [49] "PSD22504"            "PSD22505"            "PSD25001"
## [52] "PSD25002"            "PSD25003"            "PSD25004"
## [55] "PSD25005"            "PSD27501"            "PSD27502"
## [58] "PSD27503"            "PSD27504"            "PSD27505"
## [61] "PSD30001"            "PSD30002"            "PSD30003"
## [64] "PSD30004"            "PSD30005"            "Efficacite1"
## [67] "Efficacite2"         "Efficacite3"         "Efficacite4"
## [70] "Efficacite5"         "Entretien_Necessaire"
```

```r
#== DETERMINER LES VARIABLES LES PLUS IMPORTANTES EN UTILISANT RANDOMFOREST ===

# Charger la bibliothèque randomForest
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

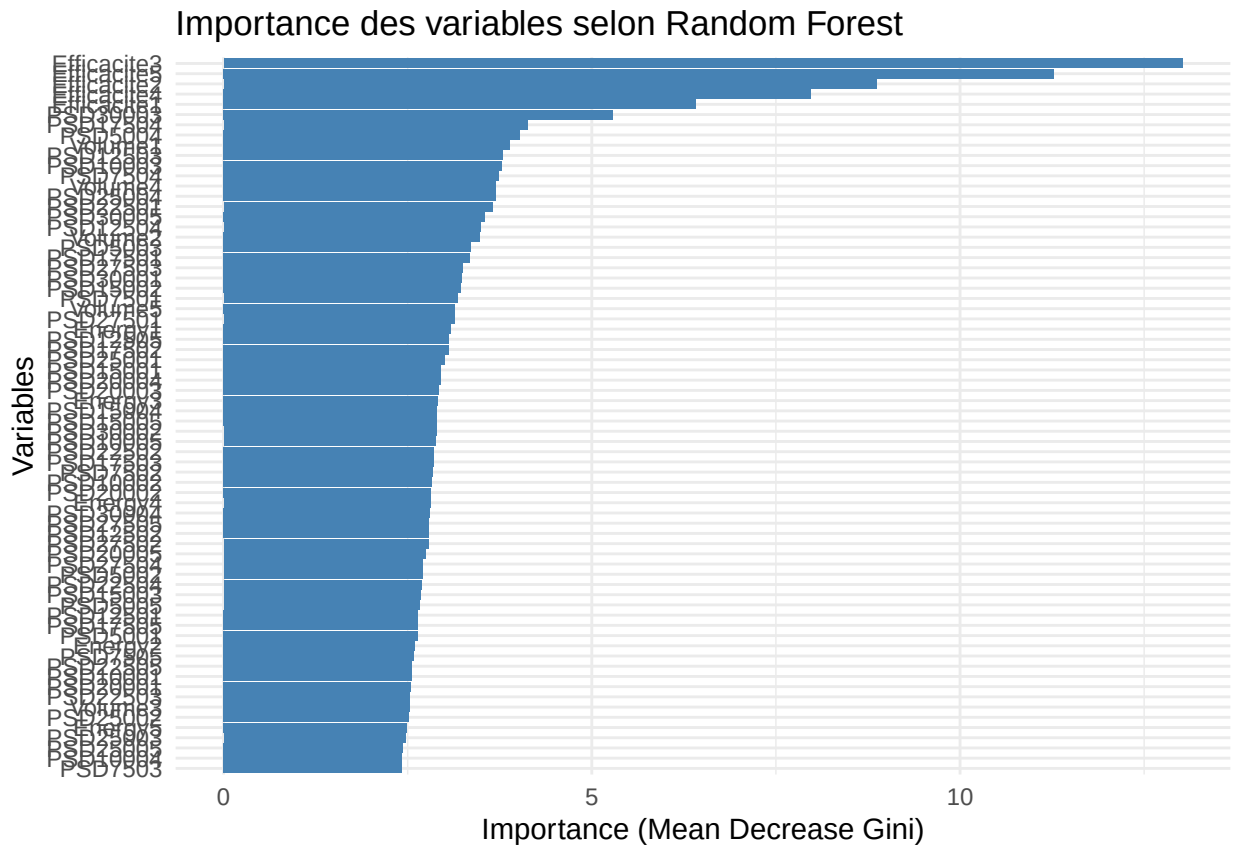
```r
# Assurer que la variable cible est un facteur
study_data_filtered$Entretien_Necessaire <- as.factor(study_data_filtered$Entretien_Necessaire)

# Créer le modèle Random Forest
rf_model <- randomForest(
  Entretien_Necessaire ~ .,  # Modèle basé sur toutes les autres colonnes
  data = study_data_filtered,
  ntree = 500,  # Nombre d'arbres
  importance = TRUE  # Calculer l'importance des variables
)

# Extraire les scores d'importance des variables
variable_importance <- importance(rf_model)

# Convertir les résultats en data.frame pour plus de clarté
importance_df <- data.frame(
  Variable = rownames(variable_importance),
  MeanDecreaseGini = variable_importance[, "MeanDecreaseGini"]
)

# Trier les variables par importance décroissante
importance_df <- importance_df[order(-importance_df$MeanDecreaseGini), ]

# Afficher les variables les plus importantes
print("Variables importantes :")
```

```
## [1] "Variables importantes :"
```

```r
print(head(importance_df, 10))  # Afficher les 10 premières variables
```

```
##                Variable MeanDecreaseGini
## Efficacite3 Efficacite3        13.020171
## Efficacite5 Efficacite5        11.279095
## Efficacite2 Efficacite2         8.866106
## Efficacite4 Efficacite4         7.975253
## Efficacite1 Efficacite1         6.409800
## PSD30003       PSD30003         5.292071
## PSD17504       PSD17504         4.126300
## PSD5004         PSD5004         4.020572
## Volume1         Volume1         3.892298
## PSD12503       PSD12503         3.799161
```

```r
# Optionnel : Visualiser les importances avec ggplot2
library(ggplot2)
ggplot(importance_df, aes(x = reorder(Variable, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Importance des variables selon Random Forest",
    x = "Variables",
    y = "Importance (Mean Decrease Gini)"
  ) +
  theme_minimal()
```



```r
# Définir un seuil pour la sélection des variables importantes
threshold <- 6  # Ajustez ce seuil selon vos données

# Filtrer les variables importantes
important_vars <- importance_df$Variable[importance_df$MeanDecreaseGini > threshold]

# Afficher les variables importantes
print("Variables importantes :")
```

```
## [1] "Variables importantes :"
```

```r
print(important_vars)
```

```
## [1] "Efficacite3" "Efficacite5" "Efficacite2" "Efficacite4" "Efficacite1"
```

```r
#== MODELISATION RandomForest ===

# Filtrer les données pour ne conserver que les colonnes importantes et la variable cible
study_data_filtered_selected <- study_data_filtered[, c(important_vars, "Entretien_Necessaire")]

# Assurer que la variable cible est un facteur
study_data_filtered_selected$Entretien_Necessaire <- as.factor(study_data_filtered_selected$Entretien_Ne

# Diviser les données en ensemble d'entraînement et de test
set.seed(123)  # Pour reproductibilité
train_index <- createDataPartition(study_data_filtered_selected$Entretien_Necessaire, p = 0.8, list = F
train_data <- study_data_filtered_selected[train_index, ]
test_data <- study_data_filtered_selected[-train_index, ]

# Entraîner le modèle Random Forest avec les variables importantes
rf_model <- randomForest(
  Entretien_Necessaire ~ .,  # Utiliser uniquement les colonnes importantes
  data = train_data,
  ntree = 500,  # Nombre d'arbres
  importance = TRUE
)

# Faire des prédictions sur l'ensemble de test
test_pred <- predict(rf_model, newdata = test_data)

# Calculer la matrice de confusion
test_cm <- confusionMatrix(test_pred, test_data$Entretien_Necessaire)

# Afficher la matrice de confusion
print("Matrice de confusion (Test) :")
```

```
## [1] "Matrice de confusion (Test) :"
```

```r
print(test_cm)
```

```
## Confusion Matrix and Statistics
```

```
##
##            Reference
## Prediction  0  1
##          0 33 11
##          1  9 46
##
##                Accuracy : 0.798
##                  95% CI : (0.7054, 0.872)
##     No Information Rate : 0.5758
##     P-Value [Acc > NIR] : 2.613e-06
##
##                   Kappa : 0.589
##
##  Mcnemar's Test P-Value : 0.8231
##
##             Sensitivity : 0.7857
##             Specificity : 0.8070
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.8364
##              Prevalence : 0.4242
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4444
##       Balanced Accuracy : 0.7964
##
##        'Positive' Class : 0
##
```

```r
#=============================================================================================================

####======= PRÉPARATION SUR LES DONNÉES DE TEST ==========================================================
#Lire les données
sensors_score <- read.csv("C:/Users/samso/OneDrive/Bureau/Apprentissage statistique/Devoir 1/sensors-sc

# Reshaper les données avec un ID par lignes
sensors_score_reshaped <- sensors_score %>%
  pivot_wider(names_from = Month, values_from = c(Volume, Energy, starts_with("PSD"))) %>%
  rename_with(~ gsub("_", "", .), starts_with("Volume_")) %>%
  rename_with(~ gsub("_", "", .), starts_with("Energy_")) %>%
  rename_with(~ gsub("_", "", .), starts_with("PSD"))

# Calcul des colonnes d'efficacité pour chaque mois
# Étape 1 : Créer une liste des colonnes Volume et Energy
volume_cols <- grep("^Volume\\d+$", names(sensors_score_reshaped), value = TRUE)
energy_cols <- gsub("Volume", "Energy", volume_cols)

# Étape 2 : Ajouter les colonnes d'efficacité
for (i in seq_along(volume_cols)) {
  sensors_score_reshaped <- sensors_score_reshaped %>%
    mutate(!!paste0("Efficacite", str_extract(volume_cols[i], "\\d+$")) :=
             !!sym(volume_cols[i]) / !!sym(energy_cols[i]))
}

##== PRÉDICTION ==================================
```

```r
# Filtrer uniquement les variables importantes dans sensors_score_reshaped, exclure `ID` des prédicteurs
predictor_columns <- setdiff(important_vars, "ID")  # Exclure ID si présent dans les variables importan
sensors_score_filtered <- sensors_score_reshaped[, c("ID", predictor_columns)]


sensors_score_filtered$Prediction <- predict(rf_model, newdata = sensors_score_filtered[, -1])  # Exclu

# Ajouter les probabilités de prédiction
sensors_score_filtered$Probability <- predict(rf_model, newdata = sensors_score_filtered[, -1], type =

# Trier les pompes par probabilité de nécessiter un entretien
sensors_score_ranked <- sensors_score_filtered %>%
  arrange(desc(Probability))

# Sélectionner les 20 000 premières pompes
selected_pumps <- sensors_score_ranked %>%
  slice_head(n = 20000)

# Afficher ou enregistrer les résultats
print(selected_pumps)
```

```
## # A tibble: 20,000 x 8
##        ID Efficacite3 Efficacite5 Efficacite2 Efficacite4 Efficacite1 Prediction
##     <dbl>       <dbl>       <dbl>       <dbl>       <dbl>       <dbl> <fct>
##  1 500254        1.55        1.68        1.26        1.55        3.18 1
##  2 500352        3.15        3.12        3.02        2.84        3.19 1
##  3 500466        1.65        1.52        2.88        1.26        3.30 1
##  4 500790        2.87        1.27        2.48        2.91        3.07 1
##  5 501002        2.92        3.26        3.09        2.61        3.22 1
##  6 501422        3.05        3.19        3.28        2.98        3.20 1
##  7 501703        1.66        1.52        1.70        1.30        2.88 1
##  8 501762        1.52        1.36        1.20        1.63        3.33 1
##  9 501952        1.34        1.32        1.73        1.66        3.28 1
## 10 501962        1.15        1.55        1.17        1.45        2.85 1
## # i 19,990 more rows
## # i 1 more variable: Probability <dbl>
```

```r
# Sélectionner les 20 000 premières pompes

selected_pumps <- selected_pumps[1:20000, "ID"]
```