Random Forest

Завантаження пакетів

```
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(pacman, ggplot2,
   plotly, corrr, caret, randomForest, corrplot)
```

```
library(pacman)
library(caret)
library(randomForest)
library(ggplot2)
library(corrplot)
library(corrr)
```

Завантаження даних

```
data <- read.csv("bodyPerformance.csv")
```

1. Опис даних

Це дані, які підтверджують оцінку працездатності з віком і деякі дані про результативність фізичної підготовки. Дани зібрані Корейським фондом підтримки спорту. Джерело: https://www.kaggle.com/kukuroo3/body-performance-data.

Розглянемо перші рядки датасету:

```
head(data)
```

```
##   age gender height_cm weight_kg body.fat_. diastolic systolic gripForce
## 1  27      M     172.3     75.24       21.3        80      130      54.9
## 2  25      M     165.0     55.80       15.7        77      126      36.4
## 3  31      M     179.6     78.00       20.1        92      152      44.8
## 4  32      M     174.5     71.10       18.4        76      147      41.4
## 5  28      M     173.8     67.70       17.1        70      127      43.5
## 6  36      F     165.4     55.40       22.0        64      119      23.8
##   sit.and.bend.forward_cm sit.ups.counts broad.jump_cm class
## 1                    18.4             60          217     C
## 2                    16.3             53          229     A
## 3                    12.0             49          181     C
## 4                    15.2             53          219     B
## 5                    27.1             45          217     B
## 6                    21.0             27          153     B
```

Перевірка на наявність NA даних:

```
colSums(is.na(data))
```

```
##                     age                  gender               height_cm
##                       0                       0                       0
##               weight_kg             body.fat_.               diastolic
##                       0                       0                       0
##                systolic              gripForce sit.and.bend.forward_cm
##                       0                       0                       0
##          sit.ups.counts          broad.jump_cm                   class
##                       0                       0                       0
```

Переглянемо тип даних кожної колонки датасету:

```
sapply(data, class)
```

```
##                     age                  gender               height_cm
##               "numeric"             "character"               "numeric"
##               weight_kg             body.fat_.               diastolic
##               "numeric"               "numeric"               "numeric"
##                systolic              gripForce sit.and.bend.forward_cm
##               "numeric"               "numeric"               "numeric"
##          sit.ups.counts          broad.jump_cm                   class
##               "numeric"               "numeric"             "character"
```

Сконвертуємо змінні gender та class на факторні:

```
# Converting 'gender' and 'class' to a factor
data$gender <- factor(data$gender)
data$class <- factor(data$class)
```

Розділимо датасет на тренувальний та тестовий:

```
set.seed(123)

samp <- sample(nrow(data), 0.8 * nrow(data))

train <- data[samp, ]
test <- data[-samp, ]
dim(train)
```

```
## [1] 10714    12
```

```
dim(test)
```

```
## [1] 2679   12
```

```
#drop <- c("class")
#test2 <- test[,!(names(test) %in% drop)]
```

Оскільки алгоритм випадкового лісу не чутливий до різних по масштабу даних, стандартизацію (scaling) не проводимо.

Використовуємо перехресну перевірку для знаходження оптимального значення mtry:

```
set.seed(51)
# Training using 'random forest' algorithm
model <- train(class ~ .,
data = train, # Use the train data frame as the training data
method = 'rf',# Use the 'random forest' algorithm
trControl = trainControl(method = 'cv', # Use cross-validation
number = 5)) # Use 5 folds for cross-validation
model
```

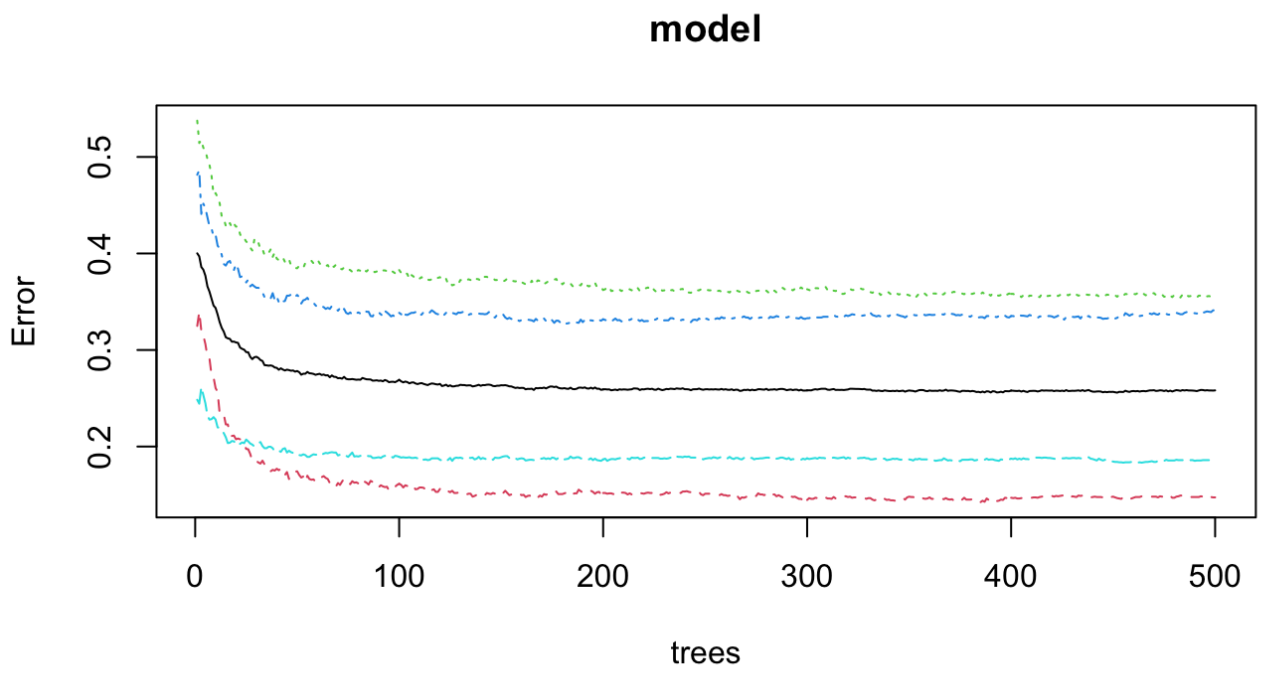
```
## Random Forest
##
## 10714 samples
##    11 predictor
##     4 classes: 'A', 'B', 'C', 'D'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 8571, 8571, 8572, 8571, 8571
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7181264  0.6242318
##    6    0.7369798  0.6493813
##   11    0.7348330  0.6465364
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

Використовуючу значення mtry = 6, побудуємо модель на основі випадкового лісу:

```
set.seed(71)
model <- randomForest(class ~ ., data = train, ntree = 500, mtry= 6)
model
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = train, ntree = 500,      mtry = 6)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 25.82%
## Confusion matrix:
##      A    B    C    D class.error
## A 2240  358   22    7   0.1473163
## B  598 1752  281   83   0.3544584
## C  216  564 1779  141   0.3411111
## D   36  164  296 2177   0.1855593
```

```
plot(model)
```

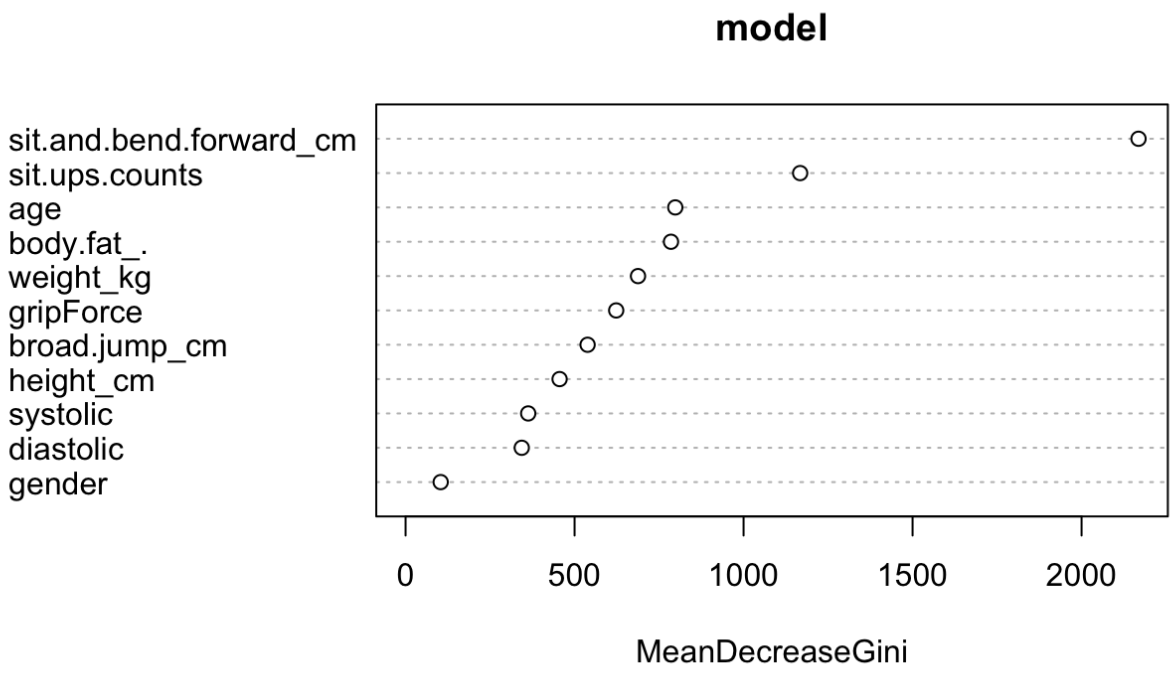


Розглянемо міри важливості регресорів (середнє падіння індекса Джинні):

```
importance(model)
```

```
##                           MeanDecreaseGini
## age                              797.9103
## gender                           103.8296
## height_cm                        455.6810
## weight_kg                        687.6615
## body.fat_.                       785.2394
## diastolic                        343.7255
## systolic                         362.8185
## gripForce                        623.1103
## sit.and.bend.forward_cm         2168.4792
## sit.ups.counts                  1167.3365
## broad.jump_cm                    538.5084
```

```
varImpPlot(model)
```



Перевіримо модель на тестових даних:

```
prediction <- predict(model, test[-12])
```

Оскільки це задача класифікації, побудуємо матрицю невідповіднестей (confusion matrix):

```
table(prediction, test$class)
```

```
##
## prediction   A   B   C   D
##          A 621 125  52   7
##          B  94 404 122  40
##          C   6  78 447  76
##          D   0  26  28 553
```

Обчислимо точність передбачення:

```
sum(prediction==test$class) / nrow(test)
```

```
## [1] 0.7558791
```

Можливі покращенння: grid search, random search, викинути деякі змінні, що не впливають на target value (class).

```
#set.seed(1234)
#tuneGrid <- expand.grid(.mtry = c(1: 10))
#rf_mtry <- train(class~.,
#data = train,
#method = "rf",
#metric = "Accuracy",
```

```
#tuneGrid = tuneGrid,
#trControl = trainControl(),
#importance = TRUE,
#nodesize = 14,
#ntree = 500)
#print(rf_mtry)
```

```
# hyperparameter grid search
#hyper_grid <- expand.grid(
#  mtry       = seq(20, 30, by = 2),
#  node_size  = seq(3, 9, by = 2),
#  sampe_size = c(.55, .632, .70, .80),
#  OOB_RMSE   = 0
#)

# total number of combinations
#nrow(hyper_grid)
```