Projekt (Wprowadzenie do uczenia maszynowego)

Piotr Mariusz Kozikowski, Kacper Dulewicz, Anna Kaczmarek

Klasyfikacja danych w oparciu o wybrane modele klasyfikacyjne

Algorytmy:

- J48 (C4.5)
- RandomTree

```
# Wczytanie bibliotek potrzebnych do wykorzystania funkcji z pakietu Weka
# oraz wyznaczania wartości miar jakości klasyfikacji
library(RWeka)
library(caret)
```

```
## Ładowanie wymaganego pakietu: ggplot2
```

```
## Ładowanie wymaganego pakietu: lattice
```

```
# Tworzenie klasyfikatora RandomTree
RandomTree <- make_Weka_classifier("weka.classifiers.trees.RandomTree")
```

```
# Wczytanie danych
pokemon_df <- read.csv("pokemon_clean.csv")
pokemon_df <- subset(pokemon_df, select=-c(X, class))
```

```
# Konwersja kolumny 'num_class' na typ factor (klasa kategoryczna)
pokemon_df$num_class <- as.factor(pokemon_df$num_class)
sapply(pokemon_df, class)
```

```
##         hp     attack    defense  sp_attack sp_defense      speed  num_class
##  "integer"  "integer"  "integer"  "integer"  "integer"  "integer"   "factor"
```

```
# Funkcja do obliczenia miar jakości klasyfikacji
calculate_metrics <- function(actual, predicted) {
  cm <- confusionMatrix(predicted, actual)

  list(
    Accuracy = cm$overall["Accuracy"],
    Sensitivity = mean(cm$byClass[, "Sensitivity"], na.rm = TRUE),
    Specificity = mean(cm$byClass[, "Specificity"], na.rm = TRUE),
    F_Measure = mean(cm$byClass[, "F1"], na.rm = TRUE)
  )
}
```

```
# Inicjalizacja pustej ramki wyników
results <- data.frame(
  `Typ metody klasyfikacji` = character(),
  `Typ metody ewaluacji` = character(),
  `Jakość klasyfikacji` = numeric(),
  `Czułość` = numeric(),
```

```
  `Swoistość` = numeric(),
  `F-measure` = numeric(),
  stringsAsFactors = FALSE
)
```

```
# 1. Metoda resubstytucji
for (method in c("RandomTree", "J48")) {
  model <- eval(parse(text = paste0(method, "(num_class ~ ., data = pokemon_df)")))
  predictions <- predict(model, pokemon_df)

  metrics <- calculate_metrics(pokemon_df$num_class, predictions)
  results <- rbind(results, data.frame(
    `Typ metody klasyfikacji` = method,
    `Typ metody ewaluacji` = "metoda resubstytucji",
    `Jakość klasyfikacji` = metrics$Accuracy,
    `Czułość` = metrics$Sensitivity,
    `Swoistość` = metrics$Specificity,
    `F-measure` = metrics$F_Measure,
     row.names = NULL
  ))
}
```

```
# 2. Podział na część uczącą i testującą (2/3 vs 1/3)
set.seed(123)
trainIndex <- sample(1:nrow(pokemon_df), size = 0.67 * nrow(pokemon_df))
trainData <- pokemon_df[trainIndex, ]
testData <- pokemon_df[-trainIndex, ]

for (method in c("RandomTree", "J48")) {
  model <- eval(parse(text = paste0(method, "(num_class ~ ., data = trainData)")))
  predictions <- predict(model, testData)

  metrics <- calculate_metrics(testData$num_class, predictions)
  results <- rbind(results, data.frame(
    `Typ metody klasyfikacji` = method,
    `Typ metody ewaluacji` = "podział zbioru",
    `Jakość klasyfikacji` = metrics$Accuracy,
    `Czułość` = metrics$Sensitivity,
    `Swoistość` = metrics$Specificity,
    `F-measure` = metrics$F_Measure,
     row.names = NULL
  ))
}
```

```
# 3. Krosswalidacja (k = 13)
folds <- createFolds(pokemon_df$num_class, k = 13)

for (method in c("RandomTree", "J48")) {
  accuracy <- c()
  sensitivity <- c()
  specificity <- c()
  f_measure <- c()

  for (fold in folds) {
```

```
    trainData <- pokemon_df[-fold, ]
    testData <- pokemon_df[fold, ]

    model <- eval(parse(text = paste0(method, "(num_class ~ ., data = trainData)")))
    predictions <- predict(model, testData)

    metrics <- calculate_metrics(testData$num_class, predictions)
    accuracy <- c(accuracy, metrics$Accuracy)
    sensitivity <- c(sensitivity, metrics$Sensitivity)
    specificity <- c(specificity, metrics$Specificity)
    f_measure <- c(f_measure, metrics$F_Measure)
  }

  results <- rbind(results, data.frame(
    `Typ metody klasyfikacji` = method,
    `Typ metody ewaluacji` = "walidacja krzyżowa",
    `Jakość klasyfikacji` = mean(accuracy),
    `Czułość` = mean(sensitivity),
    `Swoistość` = mean(specificity),
    `F-measure` = mean(f_measure),
     row.names = NULL
  ))
}
```

```
# 4. Leave-One-Out Cross-Validation (LOOCV)
for (method in c("RandomTree", "J48")) {
  # Przygotowanie zmiennych do zapisu wyników
  y_true <- c()
  y_pred <- c()
  y_probs <- c()

  # Pętla Leave-One-Out Cross-Validation
  for (i in 1:nrow(pokemon_df)) {
    # Podział danych
    trainData <- pokemon_df[-i, ]
    testData <- pokemon_df[i, , drop = FALSE]

    # Trenowanie modelu
    model <- eval(parse(text = paste0(method, "(num_class ~ ., data = trainData)")))

    # Predykcje
    predictions <- predict(model, testData)

    # Zbieranie wyników
    y_true <- c(y_true, testData$num_class)
    y_pred <- c(y_pred, predictions)
  }

  cm <- confusionMatrix(factor(y_pred), factor(y_true))

  # Obliczanie miar jakości
  accuracy <- cm$overall["Accuracy"]
  sensitivity <- mean(cm$byClass[, "Sensitivity"], na.rm = TRUE)  # Czułość
  specificity <- mean(cm$byClass[, "Specificity"], na.rm = TRUE)  # Swoistość
```

```
  f1 <- mean(cm$byClass[, "F1"], na.rm = TRUE)  # F1-measure

  # Dodawanie wyników do ramki danych
  results <- rbind(results, data.frame(
    `Typ metody klasyfikacji` = method,
    `Typ metody ewaluacji` = "leave-one-out",
    `Jakość klasyfikacji` = accuracy,
    `Czułość` = sensitivity,
    `Swoistość` = specificity,
    `F-measure` = f1,
    row.names = NULL
  ))
}
```

```
# Wyświetlenie wyników
results
```

```
##   Typ.metody.klasyfikacji Typ.metody.ewaluacji Jakość.klasyfikacji   Czułość
## 1              RandomTree metoda resubstytucji           0.9766791 0.9186532
## 2                     J48 metoda resubstytucji           0.7145522 0.6441498
## 3              RandomTree       podział zbioru           0.1977401 0.2058695
## 4                     J48       podział zbioru           0.1723164 0.1557349
## 5              RandomTree   walidacja krzyżowa           0.2200518 0.2242498
## 6                     J48   walidacja krzyżowa           0.1967407 0.1783844
## 7              RandomTree        leave-one-out           0.2126866 0.1957460
## 8                     J48        leave-one-out           0.1809701 0.1524915
##   Swoistość F.measure
## 1 0.9986663 0.9751676
## 2 0.9837249 0.6974732
## 3 0.9545267 0.2068154
## 4 0.9528999 0.1723088
## 5 0.9557606 0.3208484
## 6 0.9543221 0.3036876
## 7 0.9554589 0.2154251
## 8 0.9533244 0.1712587
```

```
# Eksport wyników do pliku CSV
write.csv(results, "projekt_WdML_RT_J48_results.csv", row.names = FALSE, fileEncoding = "Windows-1250")
```