SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: mgr inż. Wojciech Kasolik

|  |  |
| --- | --- |
| Laboratorium Nr 2  Data 11.10.2025 Temat: „Implementacja zaawansowanych modeli klasyfikacji danych w Pythonie” Wariant 13 | Marika Daniszewska  Informatyka  II stopień, stacjonarne,  2 semestr, gr. 1a |

1. Polecenie: wariant 13 zadania

W ramach laboratorium należało wykonać następujące zadanie:

Zbudowanie wieloklasowego klasyfikatora, który rozróżnia liczby:

* niepodzielne przez 3 ani 5,
* podzielne przez 3,
* podzielne przez 5,
* podzielne zarówno przez 3, jak i 5.

Dane stanowił zbiór losowych liczb całkowitych z przedziału [1, 1000], a klasy zostały określone na podstawie ich podzielności.

2. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

Do realizacji zadania wykorzystano język **Python** i bibliotekę **scikit-learn**.  
Model: **RandomForestClassifier** – klasyfikator wieloklasowy oparty na zespołach drzew decyzyjnych.

# 1. Import bibliotek

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# 2. Generowanie danych

# Liczby losowe 1–1000

X = np.random.randint(1, 1000, (500, 1))

# Klasy:

# 0 – niepodzielne przez 3 ani 5

# 1 – podzielne przez 3

# 2 – podzielne przez 5

# 3 – podzielne przez 3 i 5

y = (

    (X % 3 == 0).astype(int) + 2 \* (X % 5 == 0).astype(int)

).ravel()

# 3. Przygotowanie danych

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# 4. Budowa modelu

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train\_scaled, y\_train)

# 5. Ocena modelu

y\_pred = model.predict(X\_test\_scaled)

print("=== Raport klasyfikacji ===")

print(classification\_report(y\_test, y\_pred))

# Macierz pomyłek

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(6,4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title("Macierz pomyłek")

plt.xlabel("Predykcja")

plt.ylabel("Rzeczywista klasa")

plt.show()

# 6. Test przykładowych liczb

sample\_numbers = np.array([[3], [5], [15], [7], [30], [11]])

sample\_scaled = scaler.transform(sample\_numbers)

preds = model.predict(sample\_scaled)

print("\n=== Przykładowe predykcje ===")

for n, p in zip(sample\_numbers.ravel(), preds):

    label = ["Niepodzielna", "Podzielna przez 3", "Podzielna przez 5", "Podzielna przez 3 i 5"][p]

    print(f"Liczba {n}: {label}")

=== Raport klasyfikacji ===

precision recall f1-score support

0 0.72 0.70 0.71 60

1 0.30 0.42 0.35 19

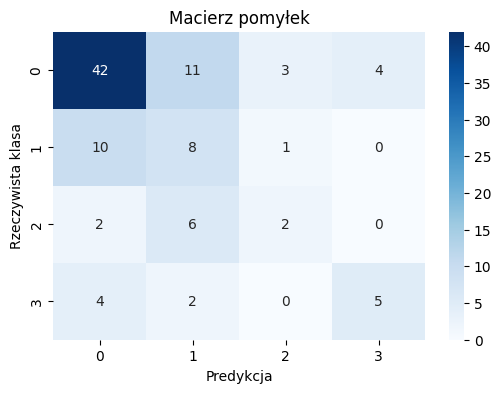
2 0.33 0.20 0.25 10

3 0.56 0.45 0.50 11

accuracy 0.57 100

macro avg 0.48 0.44 0.45 100

weighted avg 0.59 0.57 0.57 100



Github: <https://github.com/MarikaDan/NODII/tree/main/LAB2>

3. Wnioski

 Uzyskana dokładność klasyfikacji (accuracy ≈ 57%) jest umiarkowana, ale przewyższa losowe zgadywanie (25%), co potwierdza, że model nauczył się pewnych wzorców.

 Najlepiej rozpoznawana była klasa 0 (liczby niepodzielne przez 3 i 5), ponieważ stanowiła największą część danych.

 Klasy podzielne przez 3, 5 lub oba jednocześnie były mniej liczne – przez to model miał trudności z ich poprawnym rozpoznaniem.

 Random Forest potrafi częściowo odtworzyć prostą zależność arytmetyczną, mimo że operacja *modulo* nie jest dla niego bezpośrednio intuicyjna.

 Wyniki można poprawić, np. poprzez:

* zrównoważenie liczności klas,
* dodanie dodatkowych cech (X % 3, X % 5),
* użycie prostszego drzewa decyzyjnego, które łatwiej odwzorowuje reguły logiczne.