Programming in Python Language		
Group members: Jakub Łudzik, Filip Żurek **Reviewed by:** Paweł Michalcewicz, Krystian Ruszczak	**Peer Review** **Topic:** Implementation of Decision Tree and Random Forest algorithms from scratch.	**Date:** 4.12.2025

Review of the "RandomForest" Project

1. Project Description and Goal

The project is a custom implementation of the Random Forest classifier and Decision Tree, based on the NumPy library. The main educational goal was to create a working machine learning algorithm "from scratch" and compare its performance with the reference implementation from the scikit-learn library.

2. Code Structure and Implementation Quality

The project stands out with a very professional directory structure, consistent with Python best practices (utilizing the src/package_name standard).

- **Modularity:** The code is logically divided into two main modules: DecisionTree.py (responsible for single tree logic, entropy/Gini calculation, and splits) and RandomForest.py (responsible for bootstrapping and vote aggregation).
- **Interface:** The authors maintained compatibility with the scikit-learn API by implementing fit() and predict() methods. This makes using their class intuitive for anyone familiar with standard ML libraries.
- **Code Cleanliness:** The code is readable, and variable and method names are self-explanatory (e.g., _best_split, _most_common_label). Attention to implementation details is evident.

3. Tests and Verification

Verification of the algorithm's correctness was conducted in two ways:

- **Unit Tests (tests/):** The project contains a set of tests written in pytest (test_random_forest.py), which check the model's operation on simple data and subsets of real datasets (Wine). The test code coverage is at a very high level.
- **Visual Comparison (example/):** The authors prepared a compare_plots.py script that generates plots comparing the accuracy of their implementation against sklearn depending on tree depth and the number of estimators. The results (saved as comparision_plots.png) confirm the mathematical correctness of the implementation.

4. Documentation

The project documentation is at a very high level, rarely seen in student projects.

- **README:** The README.rst file contains all necessary information: description, installation instructions, usage, and examples.
- **Sphinx:** The docs/ directory contains Sphinx configuration, allowing for the generation of professional technical documentation in HTML/PDF format.
- **Project Management:** Files such as pyproject.toml, setup.cfg, and tox.ini demonstrate an advanced approach to package management and distribution.

5. Summary

The "RandomForest" project is an exemplary executed programming task. The authors not only correctly implemented a complex algorithm but also ensured a complete engineering environment – from file structure through automated tests to professional documentation. The code is ready for further development and easy to maintain.

Final Rating: Excellent (Exemplary engineering quality).