Tunability analysis

Bartłomiej Borycki Jan Cwalina

November 2024

Introduction

In this project, we analyze the tunability of hyperparameters and models using four different datasets for binary classification. We focus on three machine learning models:

- Random Forest
- Decision Tree
- Logistic Regression

For each model, we explore the impact of hyperparameter tuning through three different techniques. The goal is to evaluate how well each model can be optimized and how their performance varies across different datasets and tuning methods.

Data and preprocessing

Below is a summary of the datasets, their column types, and balance percentages.

Dataset	**Column Types**	**Balance (%)**
Diabetes	4 integer columns 3 float columns 2 object columns	92% / 8%
Students	29 integer columns 5 float columns 1 object column	61% / 39%
Telco	16 object columns 2 integer columns 2 float columns	73% / 27%
Pistachio	26 float columns 2 integer columns 1 object column	57% / 43%

Table 1: Overview of Datasets

For preprocessing, a consistent pipeline was applied across all datasets. The numeric features were imputed and scaled using the `SimpleImputer` and `MinMaxScaler`, while categorical features were one-hot encoded using `OneHotEncoder` and imputed with `SimpleImputer`. This was implemented using the following pipeline configuration:

```
pip_num = Pipeline(steps=[('impute', SimpleImputer()),
                          ('scale', MinMaxScaler())])
pip_cat = Pipeline(steps=[('ohe', OneHotEncoder(drop='first')),
                          ('impute', SimpleImputer())])
col_trans = ColumnTransformer([
    ("numeric", pip_num, make_column_selector(dtype_include=np.number)),
    ("cat", pip_cat, make_column_selector(dtype_include=np.object_))
])

example_pipeline = Pipeline([("trans", col_trans), ("model", RandomForestClassifier())])
```

Each dataset was trimmed to 1500 records, stratified by the target variable `y`, and split into training and test sets with a test size of 20%.

Optimization techniques

For model optimization, we used the ROC AUC score as the evaluation metric. The hyperparameter ranges were chosen based on the article referenced in the project. Below are the optimization methods used:

Model	Hyperparameter	Range
Logistic Regression	C	uniform(0.01, 100)
	penalty	['l1', 'l2']
	solver	['liblinear']
Decision Tree	ccp_alpha	uniform(0, 1)
	max_depth	randint(1, 31)
	min_samples_leaf	randint(1, 61)
	min_samples_split	randint(2, 61)
Random Forest	n_estimators	randint(1, 2001)
	max_samples	uniform(0.1, 1)
	max_features	uniform(0, 1)
	min_samples_leaf	randint(1, 31)

Table 2: Hyperparameter Ranges Used for Each Model

1. **Random Search:** We performed a random search over 200 different configurations, drawn uniformly from the hyperparameter grids. The search was conducted using our custom implementation of random search in a loop. For each dataset, the same set of configurations was used to ensure consistency. The hyperparameters searched for each model were:

2. **Bayesian Optimization:** We used `BayesSearchCV` from the `scikit-optimize` library for Bayesian optimization. For each model and dataset, we ran 30 iterations. To analyze the behavior of the models, we initialized the optimization process from three different starting points, setting different random seeds. This allowed us to evaluate the consistency of the results and the stability of the optimization process.

3. **Custom Weighted Random Search:** In addition to the standard random search, we explored a custom distribution for hyperparameter sampling. Specifically, we implemented a weighted random integer sampling method, where the probability of sampling lower values was higher, mimicking an exponentially decaying distribution. The function `weighted_randint` was used to sample from this distribution. We performed 200 iterations with this approach, similar to the random search

For all optimization methods, the hyperparameter ranges remained the same across different methods, ensuring consistency in the search process.

Algorithms tunability

For evaluating the tunability of the models, we chose the default configurations for each model from `sklearn`. The tunability was calculated as the difference between the ROC AUC score on test dataset of the model with default hyperparameters and the model after tuning. Thus, the smaller the difference, the more tunable the model is. Specifically, we calculated tunability for each dataset, model, and tuning method separately.

The tunability for each configuration can be expressed as:

$$\text{Tunability} = \text{AUC}_{\text{default}} - \text{AUC}_{\text{tuned}}$$

Where:

- $\text{AUC}_{\text{default}}$ is the ROC AUC score of the model with its default hyperparameters.
- $\text{AUC}_{\text{tuned}}$ is the ROC AUC score of the model after performing hyperparameter tuning using one of the optimization methods (random search, Bayesian optimization, or custom weighted random search).

The tunability was calculated separately for each model, dataset, and tuning method, allowing us to analyze how each method affected the performance of the models across different datasets. A lower tunability value indicates that the model can benefit more from tuning, suggesting that the hyperparameters significantly impact the model's performance.

Hyperparameters tunability

To evaluate the tunability of hyperparameters, we compared the performance of models with default hyperparameters from `sklearn` to their performance after hyperparameter tuning. Tunability was quantified as the difference in the *ROC AUC* score between the default and the tuned model. This approach allows us to assess how much a model's performance can improve through hyperparameter tuning.

The tunability for each model, dataset, and parameter is calculated as follows:

$$\text{Tunability} = \text{AUC}_{\text{default}} - \text{AUC}_{\text{tuned}} \tag{1}$$

Where:

- $\text{AUC}_{\text{default}}$ represents the *ROC AUC* score of the model with default hyperparameters from `sklearn`.
- $\text{AUC}_{\text{tuned}}$ represents the *ROC AUC* score after tuning a specific hyperparameter using Bayesian optimization (`BayesSearchCV`).

In this study, the tunability score was computed separately for each dataset, model, and hyperparameter. A lower tunability score indicates a greater potential for performance improvement through hyperparameter tuning. Conversely, a low tunability score suggests that the model performs well even with default settings, implying limited benefits from further tuning.

Short Summary

Decision Tree

The Decision Tree model consistently performed as well as or better than the default parameter configuration when optimized. The tunability of individual hyperparameters for this model was also notably effective.

Random Forest

For the Random Forest model, optimizing parameters generally resulted in lower performance compared to the default configuration. Among the individual hyperparameters, tuning n_estimators was the only adjustment that led to better results than the default.

Logistic Regression

Logistic Regression, despite its simplicity, demonstrated surprising tunability in terms of its hyperparameters.

Bias Sampling

There were observable differences in performance based on sampling methods. Overall, Bayesian search yielded the best results. However, the best tunability for the Decision Tree model was achieved using a custom random search.

Different Starting Points for Bayesian Search

In our experiments, varying the starting points in Bayesian search had minimal impact on model performance.

Results and visualizations

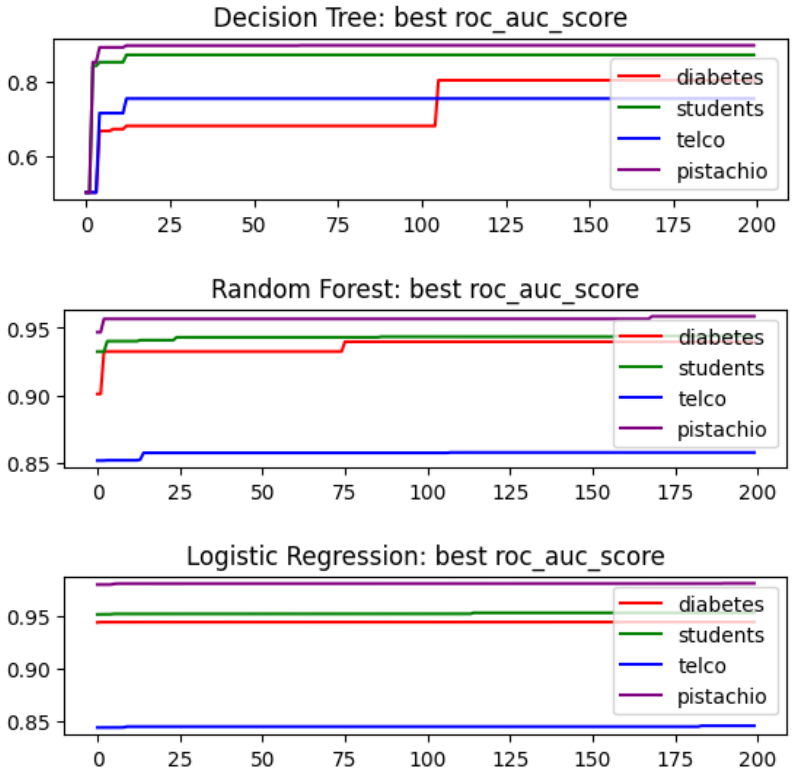

Figure 1: Best score during optimization process for random search

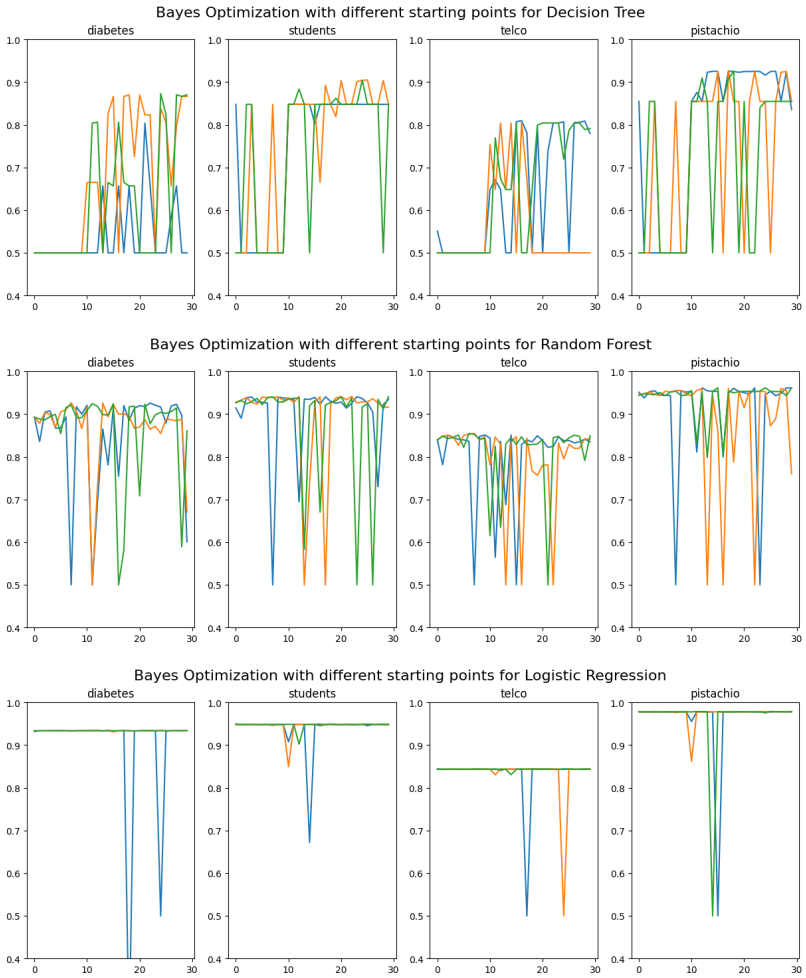

Figure 2: Scores during optimization process for bayes search

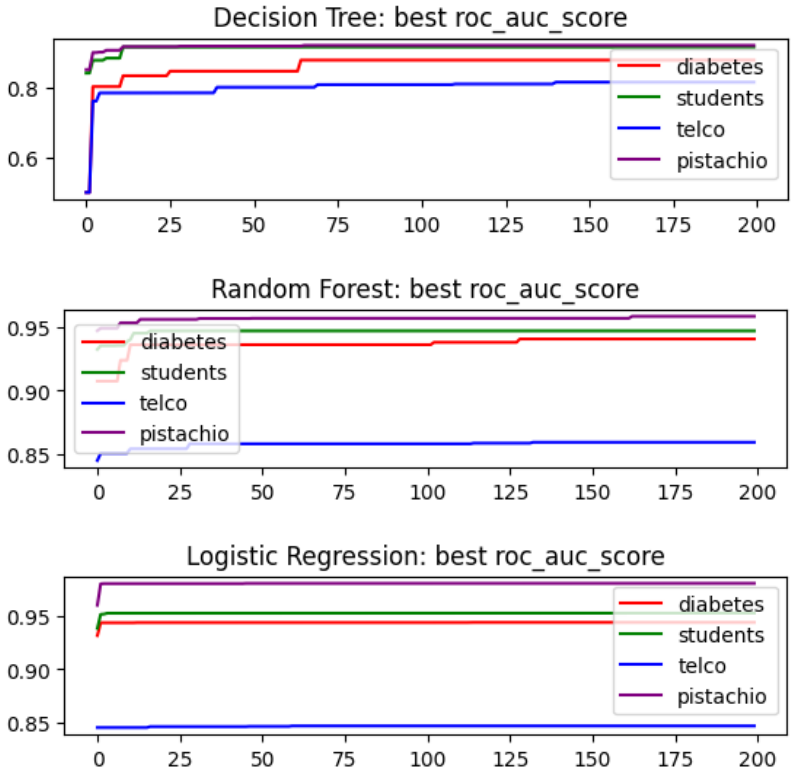

Figure 3: Best score during optimization process for custom random search

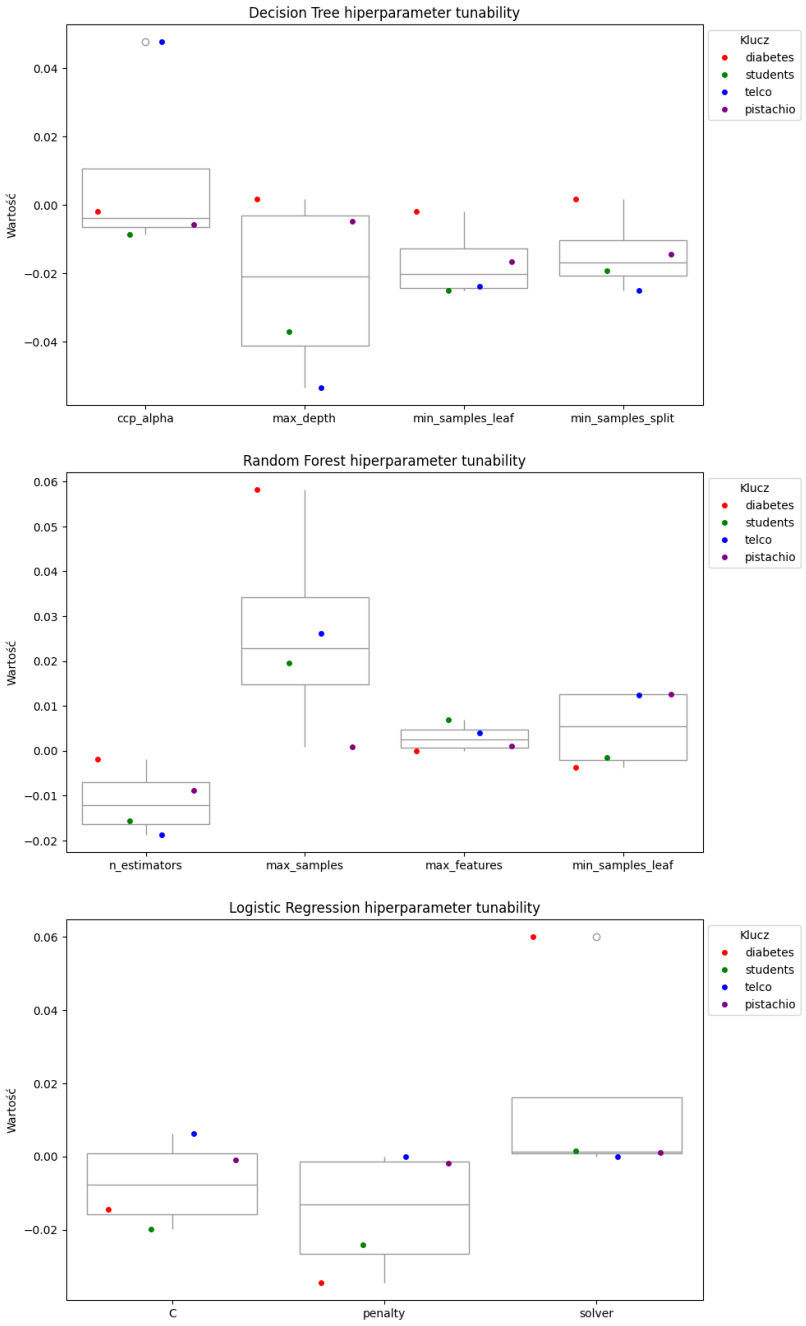

Figure 4: single hyperparameter tunability for random search for each model 1)Decision Tree 2)Random Forest 3) Logistic Regression

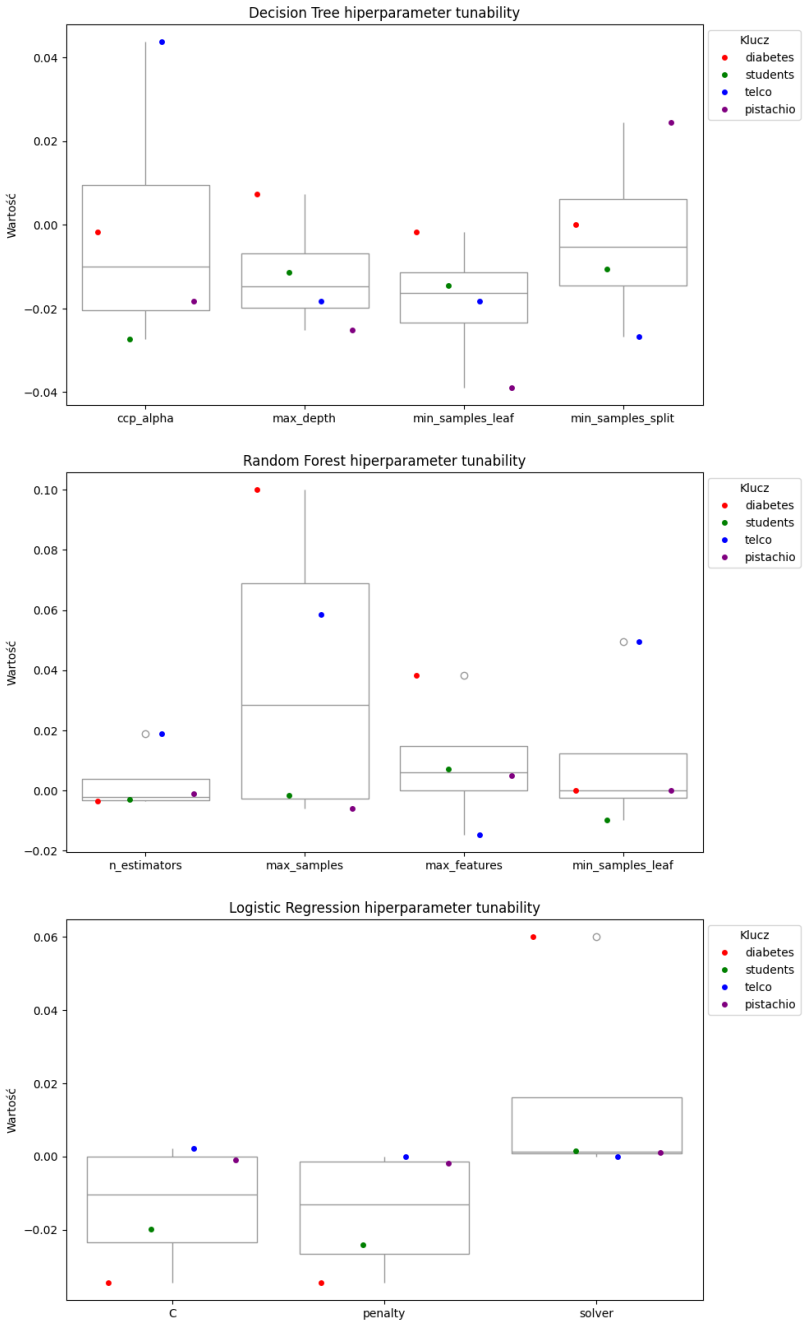

Figure 5: single hyperparameter tunability for bayes search for each model 1)Decision Tree 2)Random Forest 3) Logistic Regression

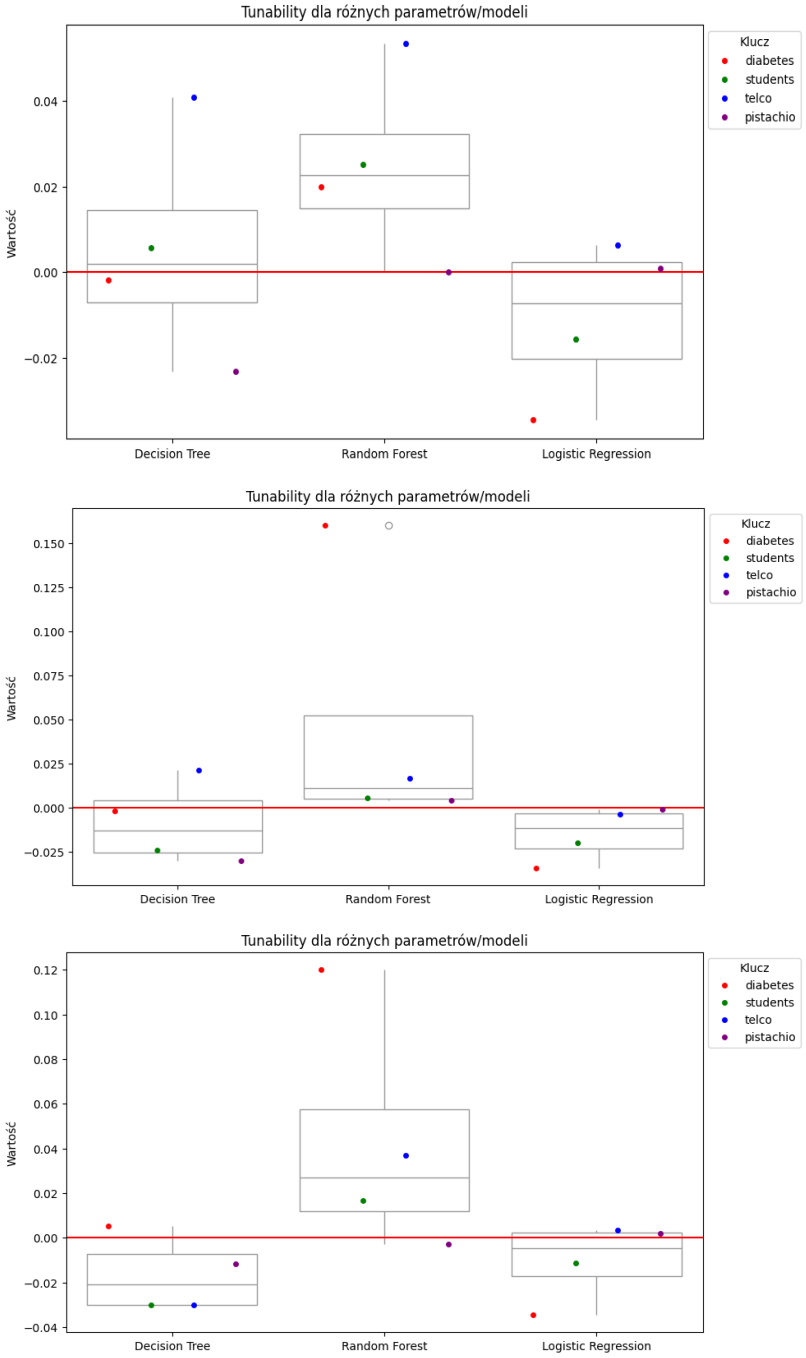

Figure 6: 1) tunability for random search 2) tunability for bayes search 3) tunability for custom random search

Model	Median tunability	Standard Deviation
Decision Tree	0.003	0.023
Random Forest	0.024	0.018
Logistic Regression	-0.008	0.015

Table 3: Tunability results for random search

Model	Median tunability	Standard Deviation
Decision Tree	-0.0125	0.021
Random Forest	0.008	0.066
Logistic Regression	-0.009	0.014

Table 4: Tunability results for bayes search

Model	Median tunability	Standard Deviation
Decision Tree	-0.021	0.025
Random Forest	0.027	0.046
Logistic Regression	-0.006	0.016

Table 5: Tunability results for custom random search