Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Project 2 - Advanced Machine Learning

Szymon Matuszewski, Mikołaj Roguski

Version 1.0

31.05.2024

Contents

1 Introduction

1.1 Problem Definition

In this project, undertaken as part of the Advanced Machine Learning course at Warsaw University of Technology during the 2024L semester, we focus on a dataset comprising **500** features and **5000** observations. The dataset is binary-classified with **2496** instances labeled as 1 and **2504** instances labeled as 0. Our primary objective is to develop a machine learning model that maximizes the effectiveness score on a separate test dataset, which also contains **5000** observations.

The effectiveness score, which evaluates the performance of the model, is defined by the following formula:

$$\text{effectiveness\_score} = \text{TP} \times \text{TP}_{\text{reward}} - \text{FC} \times n_{\text{features}} \tag{1}$$

where:

- TP is the count of true positives, calculated as the sum of instances where both y_{true} and y_{pred} are 1.
- $\text{TP}_{\text{reward}}$ is the reward for each true positive, set to 10.
- FC is the cost associated with each feature used in the model, set to 200.
- n_{features} is the number of features used by the model.

Additionally, the model is constrained to output a maximum of **1000** predictions as 1 on the test dataset.

This report outlines the approach taken to build and evaluate the machine learning model under these constraints, aiming to achieve the highest possible effectiveness score.

1.2 Approaches

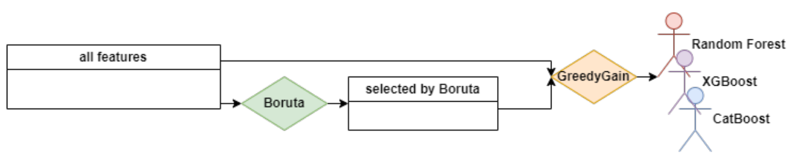


Figure 1: Pipeline simplifying tested approaches.

In our approach, we explore two distinct paths (Figure 1) for feature selection and model evaluation to maximize the effectiveness score on the test dataset.

Path 1: All Features The first path utilizes a comprehensive feature selection strategy. We apply **the Greedy Gain Selector**, which is inspired by the *step* method from *stats* package [1] in *R* and the Akaike Information Criterion (AIC) [3]. Our adaptation of **the Greedy Gain Selector** explores different strategies, including *forward selection, backward elimination, and random improvement/top-1 feature addition*, just like evolutional algorightms do. These strategies help identify unique sets of features for further testing.

Path 2: Boruta's First At the very beggining, we employ **the Boruta selector** [6] with a **Voting Classifier** as the estimator. The Voting Classifier combines three tree-based algorithms: **Random Forest** [4], **XGBoost** [5], and **CatBoost** [7]. This approach ensures robust feature selection by leveraging the strengths of each individual algorithm. Then, as in the Path 1, we apply **the Greedy Gain Selector** in different configurations.

Evaluation and Model Selection For each path, we derive *unique sets of features* and subsequently evaluate the models using **the three tree-based algorithms** (Random Forest, XGBoost, and CatBoost). The evaluation process employs our custom effectiveness score. The final step involves selecting the best model based on a comprehensive evaluation framework, which takes into consideration: *mean cross-validation score, standard deviation of cross-validation score and test score on the effectiveness metric*. By systematically comparing the performance across these metrics, we ensure the selection of **the most effective and reliable model**.

2 Feature Selectors

NOTE: Our data X was split into training set X_train of 4000 observations and X_test of 1000 observations with stratification of target variable. Initially, we also removed the correlated features.

NOTE: In the report we present the features and observations with indices in the Python's order (counting from 0), but final results from files *313435_vars.txt* and *313435_obs.txt* have an offset 1 added so as they match the R's indexing.

2.1 Boruta

The Boruta feature selector is a state-of-the-art algorithm that adheres to the **'all-relevant' principle**, ensuring that all potentially important features are retained. This approach was crucial before applying the Greedy Gain algorithm, as Boruta ensures that no significant columns are prematurely excluded.

Boruta works by creating **shadow features**, which are duplicates of the original features but shuffled to destroy any relationship with the target variable. These shadow features serve as a benchmark for assessing the importance of the actual features. The algorithm iteratively compares the importance of real features to these shadow features, using a **Random Forest Classifier** to evaluate their significance.

A key advantage of the Boruta algorithm is its ability to return a **relatively low number of features without requiring any threshold** settings which seemed to perfectly fit our task. This makes it an efficient and user-friendly choice for feature selection. By analyzing the distribution of feature importance scores and comparing them to the shadow features, Boruta provides a robust mechanism to finalize which features offer enough confidence to be included in the final list for further modeling.

NOTE: In our solution we slightly modify the original Boruta selector and replace classic Random Forest estimator with Voting Classifier consisting of Random Forest, XGBoost and CatBoost. Moreover, all tenant features were removed from extracted sets.

2.2 GreedyGainSelector

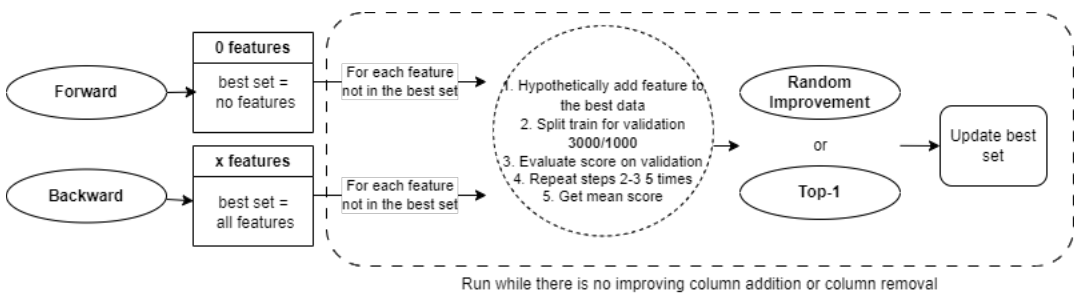


Figure 2: The Greedy Gain Selector algorithm.

As the training and validation datasets have always **1000** observations there was the need for adjustment a scorer metric used in modified Cross-Validation inside the Greedy Gain Selector (Figure 2). Assuming, the maximum number of applying label 1 in the final test dataset $(1000/5000 = 0.2)$ we decided to multiply the cost of features in the Formula 1.1 by the factor **0.2**. Moreover, estimator used inside the selector, which is once again Voting Classifier, during each fold iteration applies label 1 to the most probable **20%** of observations which sums up to **200** possible ones in return. Furthermore, our effectiveness scores have an **upper limit 2000** (not like in the final test dataset: 10000). This is just mathematically adjusted formula for smaller datasets.

2.3 Results

Features selected by the Boruta algorithm with the Voting Classifier: **x101, x100, x102, x105, x9, x103, x104**.

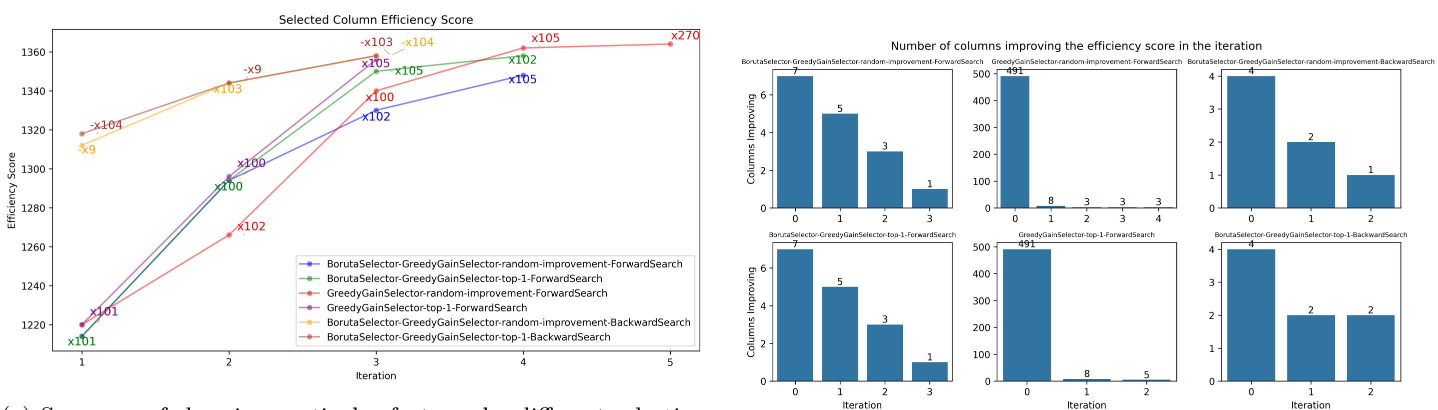
Feature Selector	Forward/Backward	Selection Strategy	x100	x101	x102	x105	x270
Boruta+GreedyGain	Backward	Random Improvement	x	x	x	x	

Boruta+GreedyGain	Forward	Random Improvement	x	x	x	x	
Boruta+GreedyGain	Backward	Top-1	x	x	x	x	
Boruta+GreedyGain	Forward	Top-1	x	x	x	x	
GreedyGain	Forward	Random Improvement	x	x	x	x	x
GreedyGain	Forward	Top-1	x	x		x	

Table 1: Feature selection results. Backward strategy for Greedy Gain only was too time consuming for evaluation.

Interestingly, each configuration of the Boruta and the Greedy Gain feature selectors ended up giving the same **4** features (1). Greedy Gain alone in two configurations gave **2** different sets of features. This sums up to **3 unique** sets to further investigation. Luckily, 3 columns appear in every single trial, these are: **x100**, **x101**, **x105**.

NOTE: We did not try the Greedy Gain and Backward options because of a time-consuming computation and too wide area to be explored.



(a) Sequence of choosing particular features by different selection methods improving mean efficiency score. '-' means that a column was removed.

(b) Number of columns which potentially improved the mean efficiency score by iteration.

Figure 3: The insight into selection order.

Table 3a demonstrates that there is an upper limit around a mean effectiveness score of **approximately 1370**, which all feature selection methods strive to attain.

Surprisingly, the Greedy Gain without Boruta tends to lower the potential features improving the mean efficiency score up to **8** in the 1. (practically second) iteration (Table 3b). When Boruta added potentially improving columns decrease rather linearly.

3 Models

3.1 Overview

We tested three different tree-based models: **Random Forest**, **XGBoost**, and **CatBoost**. All models were modified to predict the top 20% most probable labels as 1, with the remaining labels as 0. For feature optimization, we utilized the **Optuna** [2] package, aiming to maximize the mean cross-validated effectiveness score using **4-fold cross-validation**. There were 100 trials per model per set of features (there were **3 unique**). Optimised hyperparameters are:

Random Forest

- n_estimators = 1000
- max_depth: 3 to 8
- min_samples_split: 2 to 16
- min_samples_leaf: 1 to 16

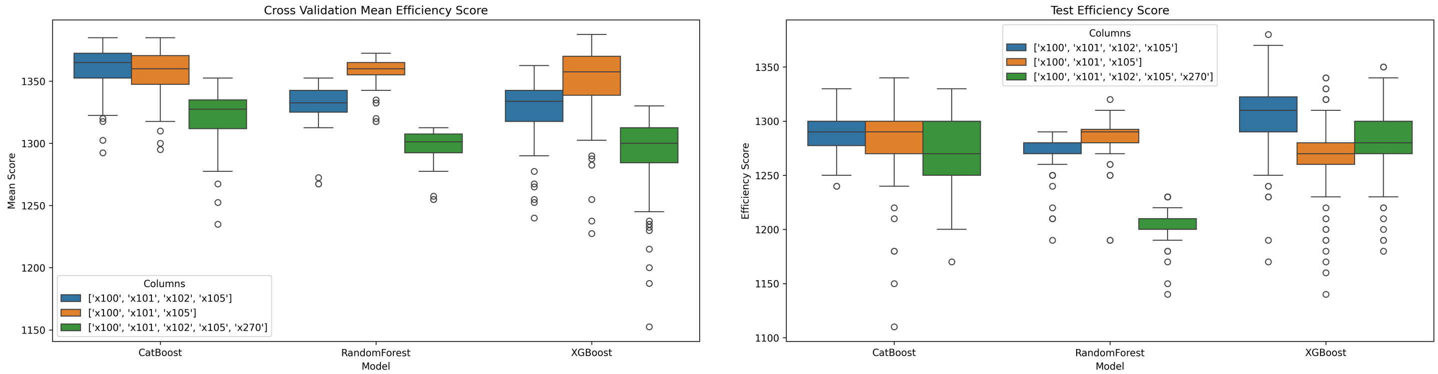
XGBoost

- n_estimators = 1000
- max_depth: 3 to 8
- learning_rate: 0.001 to 0.1 (log scale)
- subsample: 0.05 to 1.0
- colsample_bytree: 0.05 to 1.0
- min_child_weight: 1 to 10

CatBoost

- iterations = 1000
- depth: 3 to 8
- learning_rate: 0.001 to 0.1 (log scale)
- subsample: 0.05 to 1.0
- colsample_bylevel: 0.05 to 1.0
- min_data_in_leaf: 1 to 10

3.2 Results



(a) Cross Validation Mean Efficiency Score per model during hyperparameter optimisation.

(b) Mean Efficiency Score on the test dataset per model during hyperparameter optimisation.

Figure 4: Efficiency Score evaluation during Cross Validation and on test dataset.

The selection of the best model was as follows: select best model for each algorithm (Random Forest, XGBoost, CatBoost) based on the highest cross-validated mean effectiveness score, the smallest standard deviation of the cross-validated mean effectiveness score and the highest score on test dataset. The Figure 4a indicates that there is promising CatBoost model with columns: **x100**, **x101**, **x102**, **x105**. Nevertheless, the scores on the test dataset (Figure 4b) shows struggles each model had with this subset of data. Though, we decided not to rely as much on the test dataset because the validation mean consisted of more balanced data.

model	**x100**	**x101**	**x102**	**x105**	**cv_mean**	**cv_std**	**test_score**
CatBoost	x	x	x	x	1385.0	32.02	1310.0
RandomForest	x	x		x	1372.5	48.15	1290.0
XGBoost	x	x		x	1387.5	50.68	1280.0

Table 2: Best model of each type of the tree based algorithm. Selected based on cross-validated mean effectiveness score, cross-validated standard deviation of the effectiveness score and the effectiveness score on the test dataset.

As our **best model** we choose the **CatBoost** with features: **x100**, **x101**, **x102**, **x105** because of its high cross-validated mean score and relatively low standard deviation.

4 Summary

The project in numbers:

6 different feature selection methods	**3** unique sets of features	**3** different prediction models	**300** trials of hyperparameters optimisation

In our solution, the best model was identified as CatBoost with the features: **x100**, **x101**, **x102**, and **x105**. Overall, our attempts indicated that the optimal number of features for the task is between 3 and 4. While using 3 features resulted in the highest mean accuracy score, the selected combination of features and models achieved a very high score with relatively small standard deviation.

Our experiments also highlighted that the final result is significantly influenced by randomness. For instance, there is little correlation between the plots in Figure 4, and mean efficiency scores vary from the scores on the test dataset. In an ideal scenario where an employee has a maximum of 3 projects (rather than 6-7) simultaneously, it would be possible to test smaller validation datasets, resulting in a larger number of folds, as well as more feature selection algorithms, prediction models, and AutoML solutions.

Nevertheless, we successfully maximised the full potential of the Boruta feature selection combined with the original Greedy Gain Selector.

References

[1] step: Choose a model by aic in a stepwise algorithm. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/step.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[3] Hamparsum Bozdogan. Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. https://doi.org/10.1007/BF02294361, 1987.

[4] L. Breiman. Random forests. machine learning 45, 5–32. https://doi.org/10.1023/A:1010933404324, 2001.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. https://arxiv.org/abs/1603.02754, 2016.

[6] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. https://doi.org/10.18637/jss.v036.i11, 2010.

[7] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. https://arxiv.org/abs/1706.09516, 2017.

List of Tables

List of Figures