Advanced Machine Learning

Project 2

Antoni Zajko, 313553,
Dawid Płudowski, 313452
Paweł Gelar, 313343

1 Introduction

In this report, we present our work during the Advanced Machine Learning course. This project aims to develop the model that obtains the best performance according to the custom metric that promotes the solutions with high accuracy and rejects the solutions using too many variables.

1.1 Data and Evaluation

The data is provided by the lecturer. It contains 500 variables. Most of them seem to be generated from the normal distribution with parameters selected from a closed and relatively small set. This assumption is confirmed by the Figure 1.

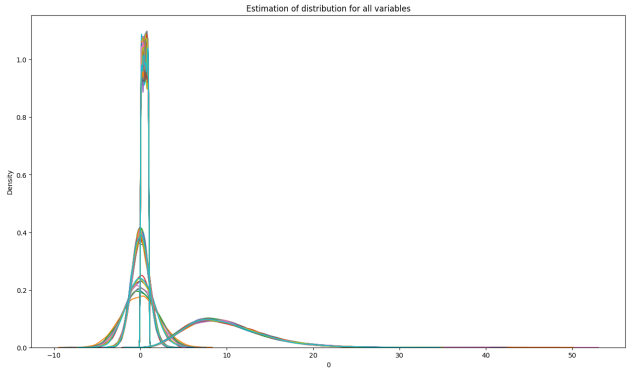


Figure 1: KDE plots for all variables from the training dataset.

The performance metric is the function balancing between precision and the number of features used in the prediction. Formally it can be described as follows:

$$m_{\alpha,\beta}(y, \hat{y}, \theta) = \alpha \sum_{i=1}^{n} \mathbb{1}_{\{y_i = \hat{y}_i\}} \mathbb{1}_{\{y_i = 1\}} - \beta\theta, \tag{1}$$

where α and β are metric parameters and are equal 10 and 200 respectively. Here, θ denotes the number of the features used during the prediction. During our development, as α and β are loosely related to the dataset size, we changed their values to make possible values of the metric bounded.

2 Methodology

The framework of the whole project can be divided into two parts: feature selection method and optimizing the model of choice. To restrict the number of experiments that we made to a considerable size, we propose the workflow presented in Figure 2.

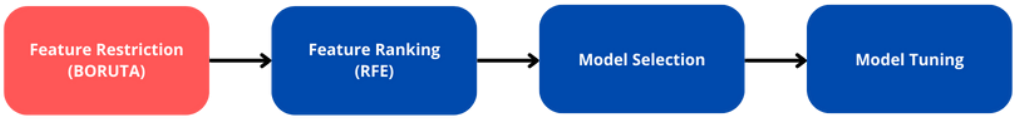


Figure 2: Workflow of our experiments. The blue color denotes the elements that we optimize, treating them as the hyperparameters of the pipeline.

2.1 Feature Selection

To not explore all subsets of the features, we use two techniques: we restrict ourselves to the features selected by the BORUTA algorithm [Kursa and Rudnicki, 2010]. To make the features' subset space even smaller, we create the ranking of the features using the Recursive Feature Elimination (RFE) algorithm, using the elastic net and random forest models as the base of the ranking. In this step, we introduce two dimensions to the experiment optimization – feature selection methods $FS = \{ElasticNet,\ RandomForest\}$ and several features used $p \in [p_B]$ where p_B denotes the number of features selected by the BORUTA algorithm.

2.2 Used Models

Next, we propose the array of models to train: XGBoost [Chen and Guestrin, 2016], ElasticNet, SVM [Evgeniou and Pontil, 2001], RandomForest, and AdaBoost. The full list of their hyperparameters to be tuned is presented in Appendix A. In this step, we introduce the next two dimensions to the optimization pipeline – a portfolio of models $\mathcal{M}$ and their hyperparameters Φ.

2.3 Evaluation Metrics

We evaluated models using the following metrics:

- **Gain** – The final objective is described in the project statement but with the scaled maximum number of selected customers. Used in optimization, visualizations, and final model selection,
- **Precision** – Fraction of positive observations among selected ones (scaled as well). Used in visualizations.

2.4 Pipeline Optimization

Taking all four dimensions that we introduce in previous subsections, we create the optimization pipeline that employs Bayesian Optimization (BO) to find the best configuration of the hyperparameters. In the optimization, we use the Tree Parzen Estimator (TPE) algorithm [Watanabe, 2023].

We ran optimization with 15-minute timeout per algorithm, which resulted in a total of around 6000 trials.

2.5 Obtaining Predictions

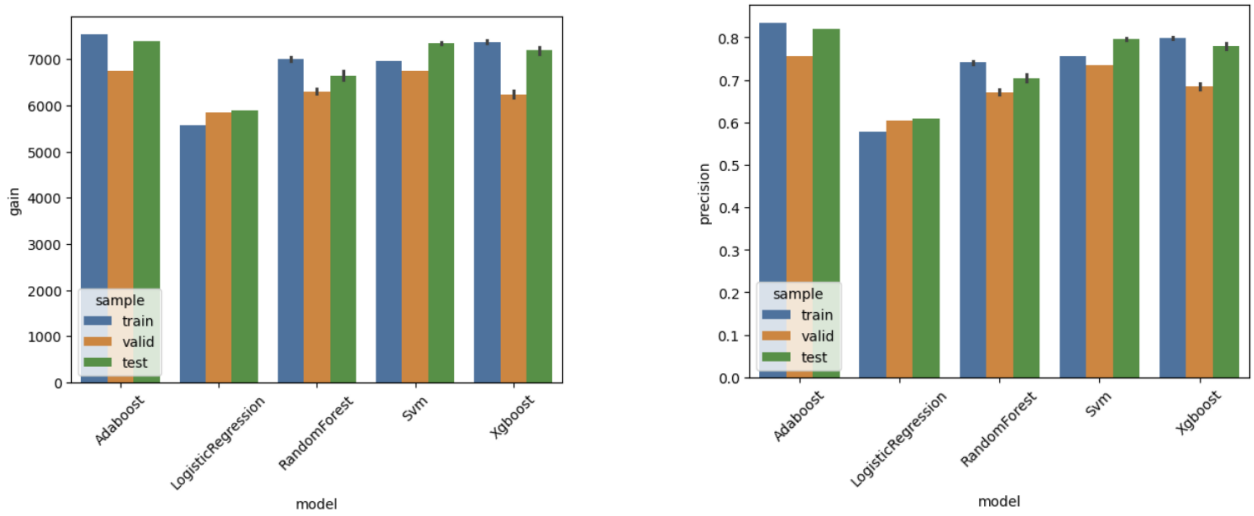
After the optimization, we trained each algorithm using its best hyperparameters 20 times and chose one with the best average gain on the test set (not used in the optimization). We trained it on all labeled data

and selected indices of 1000 observations with the highest probabilities of belonging to the positive class.

3 Results

In this section, we present the results of the experiments. We show the comparison of the overall models' performance during HPO in Figure 3. On average the best model was Adaboost and the worst one was Logistic Regression. However, the best maximal performance was achieved by the XGBoost which can be seen in Table 1. It achieved 7500 gain which is slightly higher than the AdaBoost and is mainly achieved by the lower number of features. In Table 2 we present numbers of features that were used by the best instances of the considered algorithm. The highest number of features was used by the Adaboost. On the other hand, the Logistic Regression used only one feature and still was able to achieve performance significantly better than random sampling.

In Figure 4 we show the dependency of the gain on the number of features on the validation set. RFE with Random Forest was able to provide more meaningful features than one backed with the elastic net. Models with 2, 3, and 4 features were able to achieve highest gains.



(a) Gain of the considered models.

(b) Precision of the considered models.

Figure 3: Performance of the models with their best hyperparameters.

Algorithm	Max gain
XGBoost	7500
Adaboost	7400
SVM	7400
Random Forest	7000
Logistic Regression	5900

Table 1: Maximal gain achieved by considered algorithms.

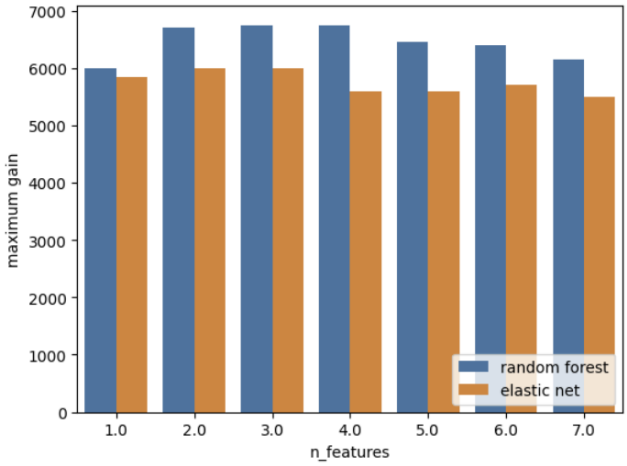


Figure 4: Maximum gain over number of features used during model creation

Algorithm	Ranking Type	Number of features for best model
Adaboost	Random Forest	4
SVM	Random Forest	3
XGBoost	Random Forest	3
Random Forest	Random Forest	2
Logistic Regression	Random Forest	1

Table 2: Number of features used to create best instances of the considered algorithms

4 Summary

During this project, we developed an efficient pipeline that allowed us to select the best model, optimize its hyperparameters, and perform predictions in cases with many insignificant variables and particular objective functions.

The feature ranking with Elastic Net as a model was significantly worse than the one using Random Forest. As a predictor Elastic Net performed the worst, Random Forest was slightly better, while AdaBoost, SVM, and XGBoost similarly achieved the best scores.

References

Miron Kursa and Witold Rudnicki. Feature selection with boruta package. *Journal of Statistical Software*, 36:1–13, 09 2010. doi: 10.18637/jss.v036.i11.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016. doi: 10.1145/2939672.2939785. URL `http://dx.doi.org/10.1145/2939672.2939785`.

Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 09 2001. ISBN 978-3-540-42490-1. doi: 10.1007/3-540-44673-7_12.

Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance, 2023.

A Models' Hyperparameters grids

Hyperparameter	Range	Distribution
max_depth	[1, 5]	Uniform (int)
n_estimators	[10, 100]	Uniform (int)
max_leaves	[1, 64]	Uniform (int)
learning_rate	[0.00001, 10]	Uniform (log domain)
gamma	[0.00001, 10]	Uniform (log domain)
colsample_bytree	[0.5, 1]	Uniform
subsample	[0.5, 1]	Uniform
reg_alpha	[0.00001, 10]	Uniform (log domain)
reg_lambda	[0.00001, 10]	Uniform (log domain)

Table 3: Hyperparameter ranges of the XGBoost

Condition	Hyperparameter	Values' range	Distribution
n/a	tol	[0.0001, 0.001]	loguniform
n/a	C	[0.0001, 10000]	loguniform
n/a	solver	[lbfgs, liblinear, newton-cg, newton-cholesky, sag, saga]	categorical
solver = liblinear	intercept scaling	[0.001, 1]	uniform
	penalty	[l1, l2]	categorical
solver = liblinear and penalty = l2	dual	[true, false]	categorical
solver = saga	penalty	[elasticnet, l1, l2, null]	categorical
	l1 ratio	[0, 1]	uniform
solver $\neq$ saga and solver $\neq$ liblinear	penalty	[l2, null]	categorical

Table 4: Hyperparameters grid used for HPO of elastic net algorithm.

Hyperparameter	Range	Distribution
C	[0.0001, 10000]	Uniform (log domain)
kernel	{linear, poly, rbf, sigmoid}	Uniform (categorical)
degree	[1, 5]	Uniform (int)
gamma	{scale, auto}	Uniform (categorical)

Table 5: Hyperparameters grid used for HPO of SVM algorithm.

Hyperparameter	Range	Distribution
n_estimators	[10, 200]	Uniform (int)
criterion	{gini, entropy, log_loss}	Uniform (categorical)
max_depth	[1, 7]	Uniform (int)
min_samples_split	[2, 128]	Uniform (int)
min_samples_leaf	[2, 128]	Uniform (int)
max_leaf_nodes	[2, 128]	Uniform (int)
min_weight_fraction_leaf	[0.00001, 0.5]	Uniform (log domain)

Table 6: Hyperparameters grid used for HPO of Random forest algorithm.

Hyperparameter	Range	Distribution
max_depth	[1, 5]	Uniform (int)
min_samples_leaf	[10, 100]	Uniform (int)
n_estimators	[10, 200]	Uniform (int)
learning_rate	[0.00001, 10]	Uniform (log domain)

Table 7: Hyperparameters grid used for HPO of Adaboost algorithm.