Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Advanced Machine Learning - Project 2

Wojciech Kosiuk, Adam Majczyk, Damian Skowroński

Version 1.0

31.05.2024

Contents

1 Introduction

NOTE: In the report the variables are indexed from 1 to 500 and rows from 1 to 5000.

1.1 General approach

The summary of the approach discussed in this report is presented in Figure 1.

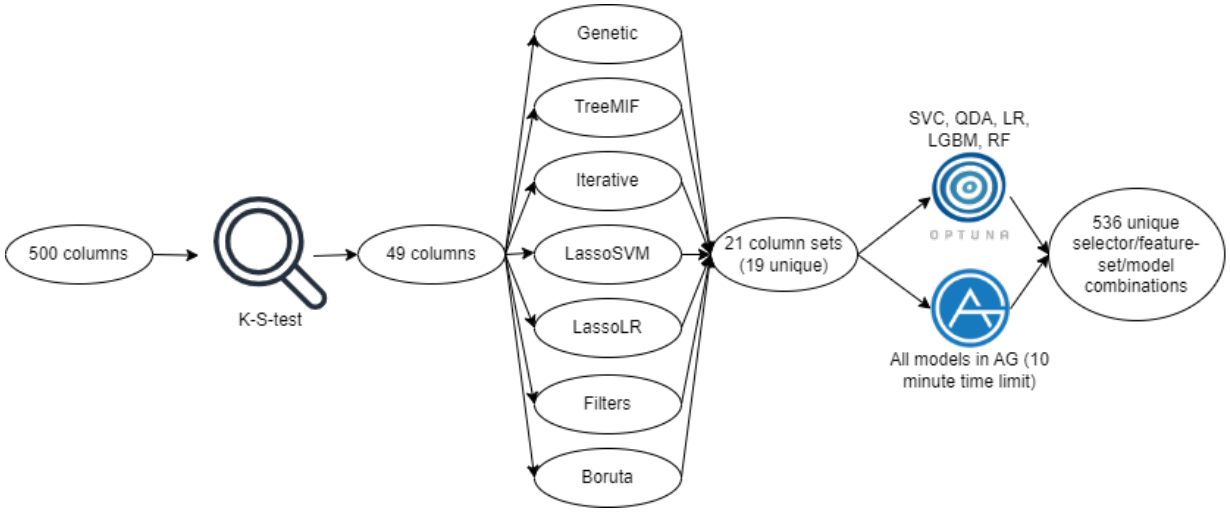

Figure 1: General methodology diagram

1.2 SCORE function

Throughout the project, a custom metric specifically designed for this task was used. It evaluates a model based on the number of true positives (precision) among the top 20% of predictions based on probability and penalizes based on the number of features used. $N_{20\%}$ denotes the set of the 20% of observations with the highest probability, n_{test} denotes the size of the testing sample. n_{test} denotes the number of features used. $\frac{5000}{n_{test}}$ captures the scaling factor (since the actual testing sample will have 5000 observations, and our validation samples have n_{test} it is necessary to scale the reward appropriately).

$$\text{SCORE} = \frac{5000}{n_{test}} \times 10 \times [\sum_{i \in \{k: x_k \in N_{20\%}\}} \mathbb{1}(y_{\text{pred},i} = y_{\text{true},i} = 1)] - 200 \times n_{\text{features}}$$

2 Feature selection methods

2.1 (0) - Initial Filtering

Since there were 500 columns to begin with, it was decided to filter out ones, which are unimportant. Two methods were applied:

1. T-test - check if mean of variable is same in both classes. If so, discard it ($\alpha = 0.05$).
2. K-S-test - check if the distribution of the variable is same in both classes. If so, discard it ($\alpha = 0.05$).

Later the distributions of the variables chosen by the methods were examined. It was determined, that the variables not discarded by the T-test had very similar distributions. Therefore, the variables not discarded by the K-S-test were later used as the initial variables for the next 7 methods. K-S-test has not discarded **49 variables**. Plots of variables chosen by the methods are Figures A.1 and A.2.

2.2 (1) - Genetic Algorithm

An Evolutionary Algorithm for Feature Selection is a bio-inspired optimization technique used to identify the most relevant features from a dataset. It mimics natural selection processes to iteratively improve feature subsets. Applied to feature selection, the evolutionary algorithm selects the best features by the pre-defined metric - in our case we used the SCORE function. Each member of the population is in fact the set of used columns by that member. For each member, we trained simple, logistic regression model and then measured its performance. We repeated the experiments with different algorithm parameters, and finally we chose a few (7) column subsets with the highest SCORE.

2.3 (2) - Tree MIF

This method is based on model feature importances [1][2] and can be applied to any tree-based model that computes feature importances. We chose to use the RandomForestClassifier due to its low computation time. In each experiment we trained random forest 25 times with different random seeds, and the mean importance of each feature was calculated. This experiment was repeated several times with different hyperparameters, and the 7 most important features from each run were selected for further analysis. Despite testing various configurations, certain features consistently appeared among the top selections. Consequently, we created a comprehensive set of all features that were among the top 7 in at least one experiment. From this set, we tested every combination of features with lengths greater than 2 by fitting a Random Forest model with default hyperparameters and evaluating its performance on the test set using the SCORE function. Finally, the 7 best sets of columns (by SCORE) were chosen for further steps.

2.4 (3) - Iterative approach

This approach was inspired by the classical stepwise method utilizing BIC/AIC metrics. Instead of these metrics, ITERATIVE_SCORE, precision, and accuracy were used for evaluation, with precision and accuracy serving as tie breakers.

$$\text{ITERATIVE\_SCORE} = \text{precision} \times \left(\frac{5000}{n_{test}} \times 10 \times \sum_{i=1}^{n_{test}} \mathbb{1}(y_{\text{pred},i} = y_{\text{true},i} = 1) - 200 \times n_{\text{features}} \right)$$

NOTE: The ITERATIVE_SCORE was defined early in the project and, as the task became better understood, evolved into the SCORE function. The outcomes from this initial selector were satisfactory, so it was not rerun with the newly defined SCORE function.

At each step, two competing models were trained for each feature removed from the current subset of features: an SVM with a hyperparameter of C=0.8 and a RandomForest (default settings). In each iteration, the column with the lowest ITERATIVE_SCORE overall was discarded. This experiment was time-consuming as it required training many models and consequently was performed only once. The algorithm did not stop when there were no improvements.

As a result, **7 subsets** of columns were selected. The best ITERATIVE_SCORE was achieved with a subset of six variables. Additionally, the best subset of five variables was chosen, as well as all subsets of four variables.

2.5 (4) - Lasso SVM

Out of the 49 variables not rejected by the K-S Test in Section 2.1, we employed the classical approach of feature selection using L1 regularization for Linear SVM. Despite numerous attempts and modifications of hyperparameters, the resulting subset of selected variables was large and did not show promising performance. In the subsequent step, only **one of the feature subsets** was utilized.

2.6 (5) - Lasso Logistic Regression

Similar to Section 2.5, but this time applied to logistic regression. The obtained subsets of features again showed poor performance. In the next step, only **one subset** was utilized.

2.7 (6) - Filter Based Approach

3 filters were applied. That is:

1. 50% of variables with highest correlation to the target were kept (24 out of 49).
2. 50% of variables with highest mututal information to the target were kept (24 out of 49).
3. variables with a variance of at least 1 were kept (28 out of 49).

The intersection of the 3 list given by the filtering methods was then used as the final result. Those were **4 variables** (x106, x176, x413, x459).

2.8 (7) - Boruta

On the 49 variables, Boruta [3] was run 100 times with a different seed. Each time the selected columns were to a list. Later the columns, which were chosen in each of the 100 trials were kept as the final column set. Those were **7 variables** (x101, x102, x103, x104, x105, x106, x9).

2.9 Notice

It is important to notice, that two of the subsets have been chosen twice. The subsets (x101, x102, x103, x105, x106) and (x101, x102, x103, x106) were chosen by both the *Iterative* and *TreeMIF* methods. They are however considered as different, since they come from different feature selection methods. Thus, in total 19 unique feature-sets were identified, 21 selector/feature-set combinations.

3 Models

3.1 Optuna

On the 21 column sets chosen by the 7 methods, the hyperparameters were optimized for **SVC**, **LGMB**, **Random Forest**, **Logistic Regression** and **QDA** using Optuna [4].

For SVC, **gamma** (10^{-3} to 10) and **C** (10^{-3} to 10) were optimized, for LGBM and RF **max_depth** (2 to 10) and **num_estimators** (50 to 500), **penalty** (*l2* or *l1*) and **C** (10^{-3} to 10) for Logistic Regression and for QDA the **regularisation parameter** (0 to 1) was optimized.

5-fold cross validation was applied in the objective function. The metric used for evaluation during CV was **precision of the 20% most probable rows** - note that the number of variables in not considered, so it is not equal to the SCORE function. This resulted in a set of hyperparameters for each model/column set/selector combination ($5 \times 21 =$ **105 feature/model combinations**).

The 105 selector/feature/model-hyperparameter combinations, were later scored (using precision, accuracy, f1, recall and the SCORE metric), each on 5 train-test splits, 80-20 size (seeds were used for reproducibility).

NOTE: the selector is mentioned as part of the combination, as TreeMIF and Iterative chose 2 of the same column subsets. See more in Subsection 2.9.

3.2 AutoGluon

Similarly, on the 21 column sets AutoGluon-TabularPredictor[5] was run 5 times (5 train-test splits, 80-20 size, seeds were used for reproducibility), with the evaluation metric set to precision. For each feature/seed combination the operation time was limited to 10 minutes (therefore not every model was evaluated 5 times). In total, it resulted in **431 selector/feature-set/model combinations**, evaluated 1 or more (up to 5) times.

Note: Only the top 10 (by precision) models from each seed/feature-set were later evaluated.

Each selector/feature-set/model/seed combination was evaluated using the SCORE metric and precision, accuracy, f1, recall.

4 Final results

4.1 Best strategies

In the Table 1 the top 20 features/model strategies are presented. They are sorted by the minimum value of SCORE function descending and standard deviation of SCORE function ascending.

index	features	model_name	method	mean_score	min_score	max_score	std_score	count
1	x101_x102_x103_x104	NeuralNetTorch_r14_BAG_L1	TreeMIF	6900.00	6800.0	7000.0	141.42	2
2	x101_x102_x103_x106	NeuralNetTorch_r197_BAG_L1	Iterative	6775.00	6750.0	6800.0	35.36	2
3	x101_x103_x105_x106	NeuralNetTorch_r30_BAG_L1	Iterative	6766.67	6700.0	6800.0	57.74	3
4	x101_x102_x105_x106	NeuralNetTorch_r197_BAG_L1	Iterative	6800.00	6700.0	6900.0	100.00	3
5	x101_x102_x103_x105_x106	NeuralNetTorch_r14_BAG_L1	Iterative	6700.00	6650.0	6750.0	70.71	2
6	x102_x104_x106	NeuralNetFastAI_r191_BAG_L2	TreeMIF	6733.33	6650.0	6800.0	76.38	3
7	x101_x102_x103_x104	NeuralNetFastAI_r11_BAG_L1	TreeMIF	6780.00	6650.0	6900.0	90.83	5
8	x101_x103_x105_x106	NeuralNetTorch_r197_BAG_L1	Iterative	6750.00	6650.0	6850.0	91.29	4
9	x101_x103_x105_x106	NeuralNetTorch_r71_BAG_L1	Iterative	6725.00	6650.0	6800.0	106.07	2
10	x101_x102_x103_x104	NeuralNetTorch_BAG_L1	TreeMIF	6725.00	6650.0	6800.0	106.07	2
11	x101_x102_x103_x104	WeightedEnsemble_L3	TreeMIF	6775.00	6650.0	6900.0	119.02	4
12	x101_x102_x103_x105_x106	NeuralNetTorch_r197_BAG_L1	Iterative	6750.00	6650.0	6850.0	141.42	2
13	x101_x102_x103_x104	SVM	TreeMIF	6850.00	6650.0	7000.0	145.77	5
14	x101_x102_x103_x104	NeuralNetFastAI_r191_BAG_L2	TreeMIF	6800.00	6650.0	7000.0	158.11	4
15	x101_x102_x103_x104	NeuralNetFastAI_r191_BAG_L1	TreeMIF	6775.00	6650.0	6900.0	176.78	2
16	x102_x104_x106	NeuralNetFastAI_r191_BAG_L1	TreeMIF	6600.00	6600.0	6600.0	0.00	2
17	x101_x103_x105_x106	NeuralNetTorch_r158_BAG_L1	Iterative	6650.00	6600.0	6750.0	70.71	4
18	x101_x102_x103_x104	NeuralNetFastAI_r143_BAG_L1	TreeMIF	6675.00	6600.0	6750.0	106.07	2
19	x101_x102_x103_x104	NeuralNetFastAI_r103_BAG_L1	TreeMIF	6760.00	6600.0	6950.0	138.74	5
20	x101_x102_x103_x105	NeuralNetTorch_r197_BAG_L1	Iterative	6716.67	6600.0	6900.0	160.73	3

Table 1: Top 20 strategies

NOTE: only strategies with at least 2 iterations are taken into consideration, as ones with only 1 iteration are considered untrustworthy. It is clear that the two feature selection methods which yielded the best results are the Iterative and TreeMIF methods. They dominate the top 20 list. Also it is interesting to see, that the best results are achieved for 4 columns most often. Aside from that only one model not based on AutoGluon appears in the top 20 (row 13).

4.2 Final model

As the final model we used **NeuralNetTorch_r14_BAG_L1** with columns (**x101, x102, x103, x104**) refitted on the entire train dataset.

4.3 Summary of strategies

In the Figure 1 the selection methods are summarised.

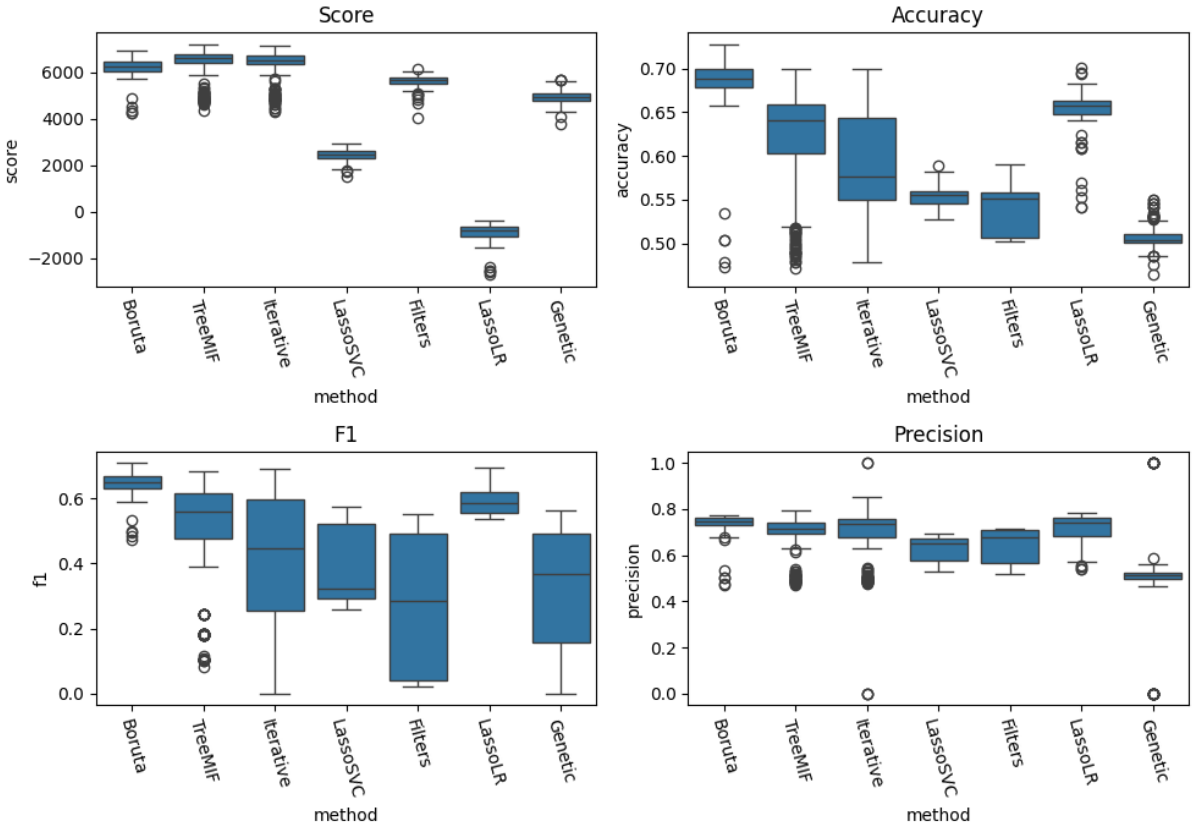

Figure 2: Summary of selection methods

It is clear that the 2 best selection methods were TreeMIF and Iterative in terms of the SCORE function. It is however not a surprise, as they were evaluated (TreeMIF) or inherently optimized (Iterative) with the SCORE function in mind. However Boruta is comparable - yielded results roughly 200-300 SCORE points lower than TreeMIF and Iterative. Both Filter and Genetic methods consistently resulted in SCOREs significantly lower than the 3 before-mentioned methods. Lasso based approaches (SVC and especially LR) are deemed as useless, as they gave the worst results (too many variables, sometimes **negative SCORE**).

In term of other metrics, Boruta was often the favourite, followed by TreeMIF, Iterative and Lasso Logistic Regression. The other methods were significantly worse.

In the Appendix, plots visualising the top models were also provided in Figure A.3

Appendix

Feature selection methods - plots

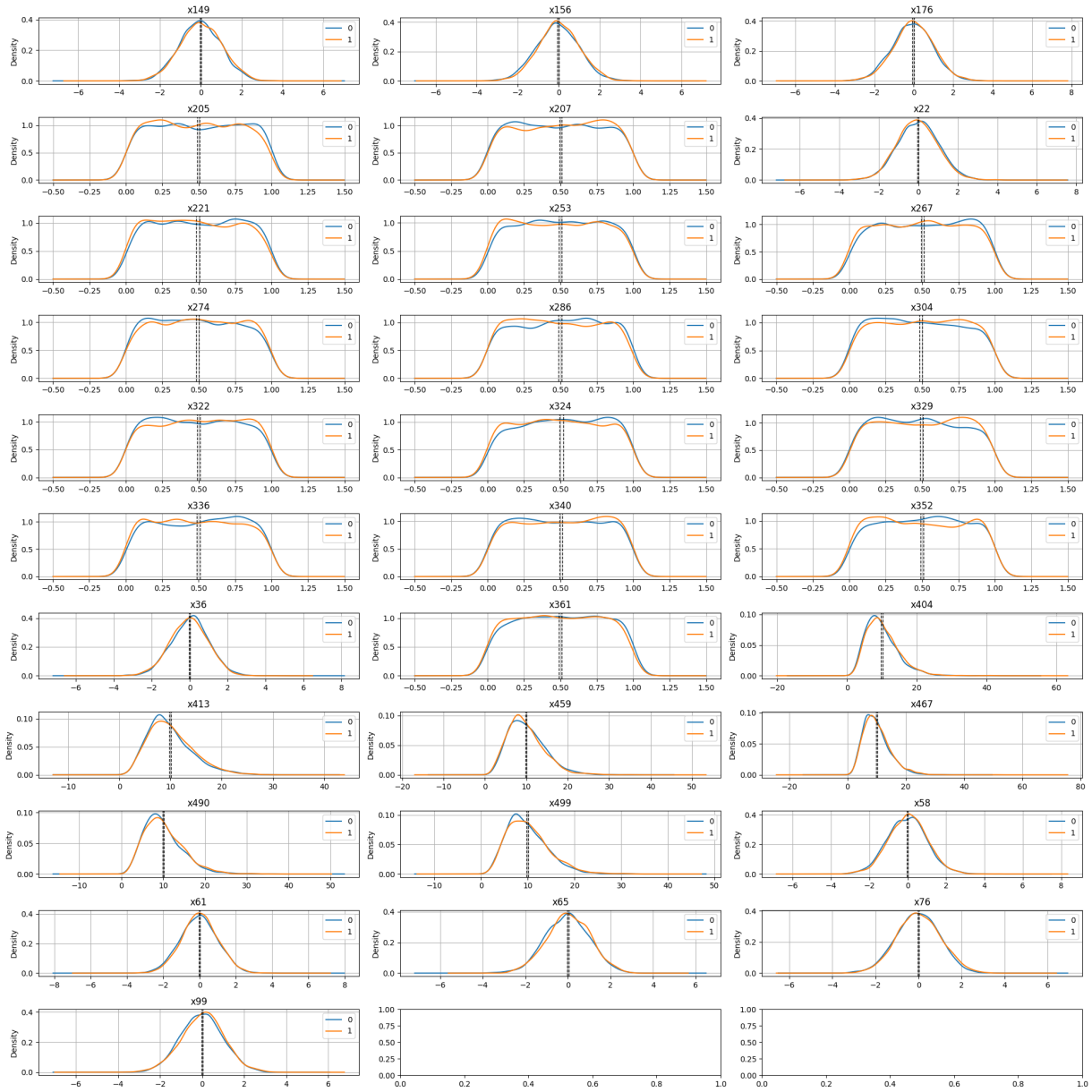

Figure A.1: Variables chosen by T-test

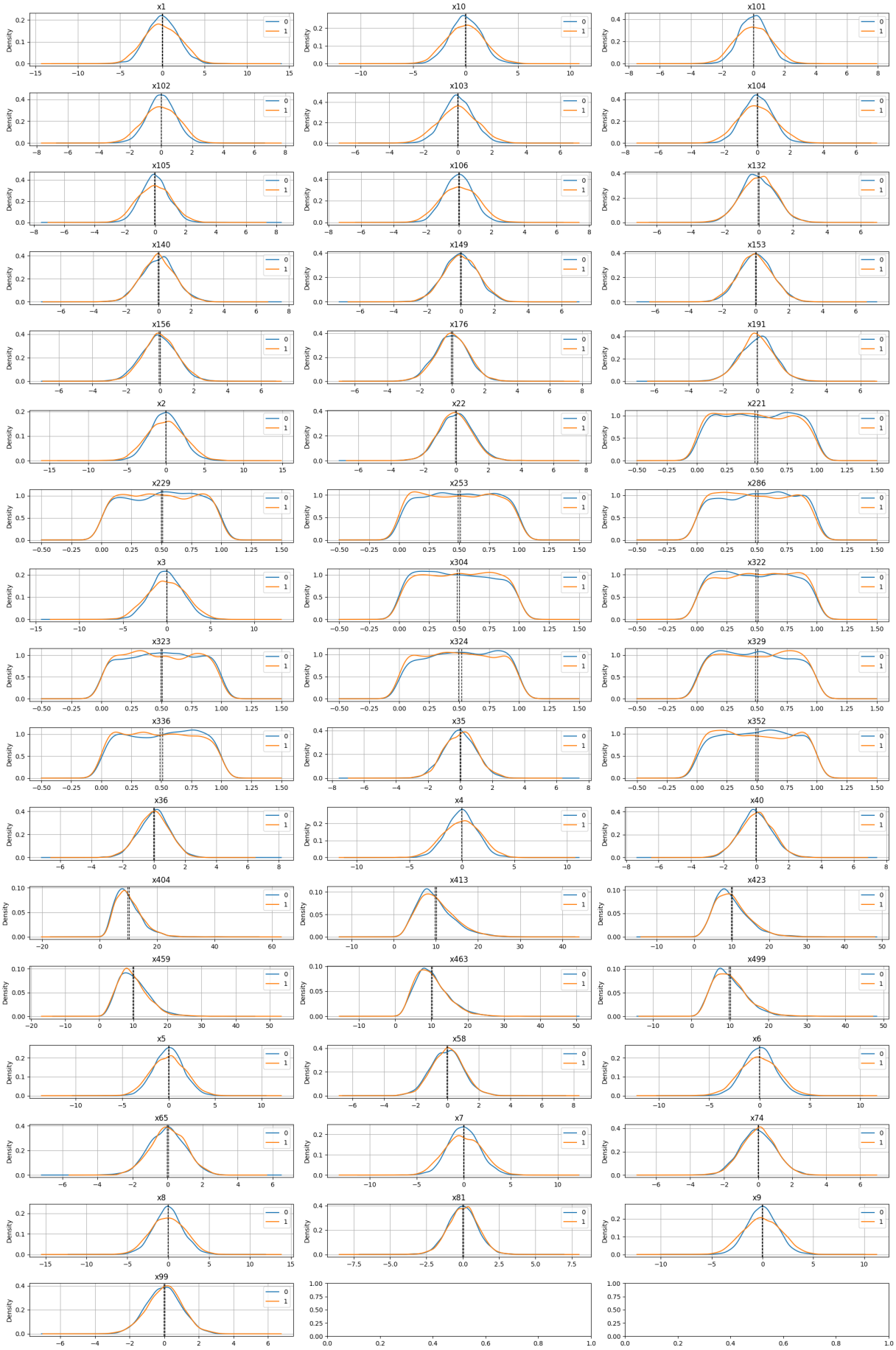

Figure A.2: Variables chosen by KS-test

Models - plots

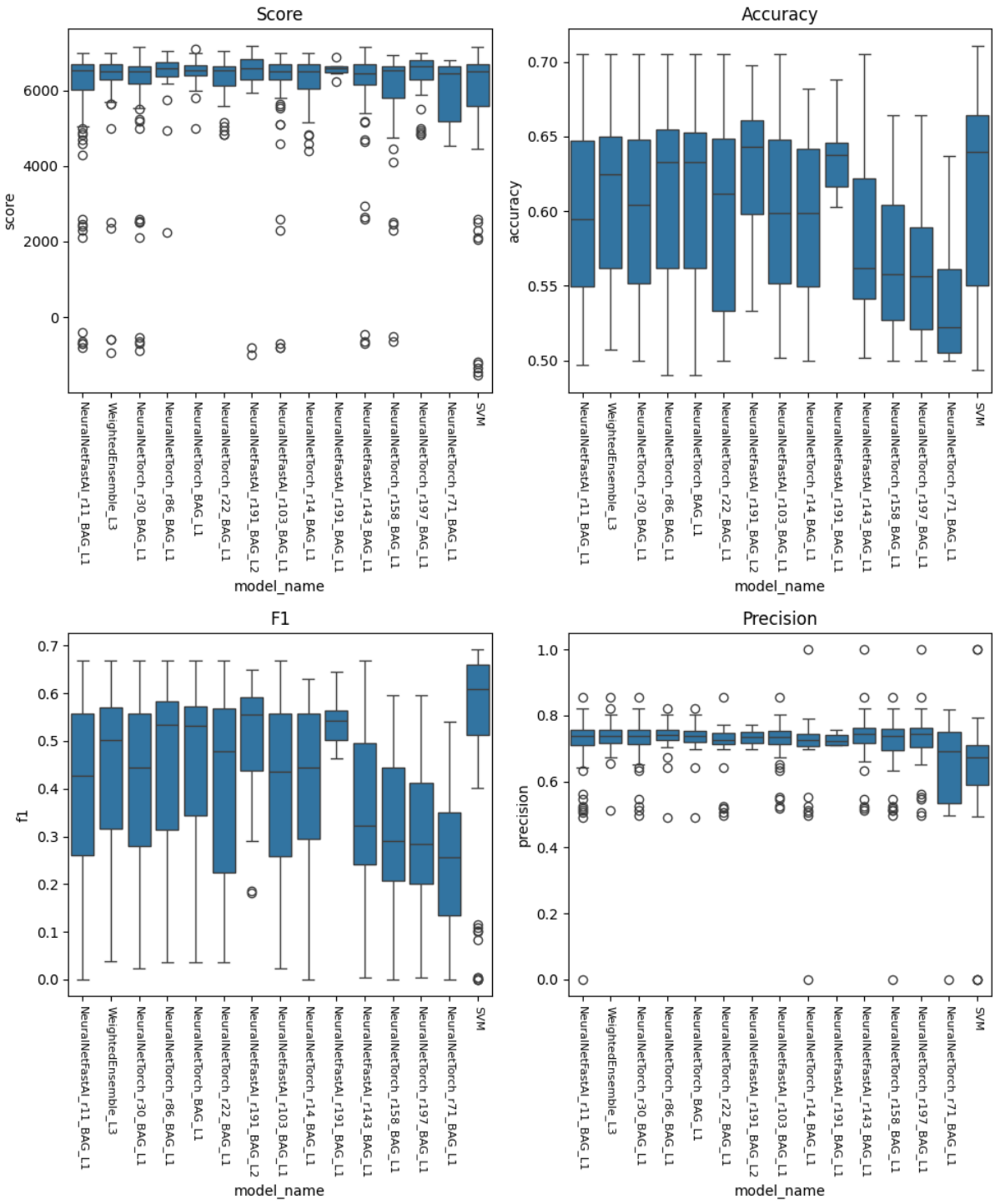

Figure A.3: Best Models (by SCORE)

References

[1] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 04 2010.

[2] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[3] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.

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

[5] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data, 2020.