PV056 semestral project

XGBoost with random forest base learner

Josef Karas, Filip Chládek, Martin Beňa

May 2024

1 Assignment

This report is for the semester-long project in the PV056 class. The project aimed to compare the XGBoost classifier [1] using a Decision Tree (DT) as the base learner (the default setting) against using a Random Forest (RF) as the base learner. This comparison was performed across four tasks utilizing OpenML datasets for binary classification, tagged OpenML-CC18, each containing 1000 or more instances, see Figure 1.

In this report, we describe our approach to each task and the outcomes of our experiments.

Figure 1: Datasets description.

Dataset	Inst.	Feat.	Clas.	Dataset	Inst.	Feat.	Clas.
dresses-sales	500	13	2	mfeat-fourier	2000	77	10
KC2	522	22	2	mfeat-karhunen	2000	65	10
climate-model-simulation-crashes	540	21	2	mfeat-morphological	2000	7	10
cylinder-bands	540	40	2	mfeat-pixel	2000	241	10
ilpd	583	11	2	mfeat-zernike	2000	48	10
balance-scale	625	5	3	KC1	2109	22	2
credit-rating	690	16	2	segment	2310	20	7
eucalyptus	736	20	5	ozone-level-8hr	2534	73	2
blood-transfusion-service	748	5	2	madelon	2600	501	2
pima-diabetes	768	9	2	dna	3186	181	3
analcatdata-dmft	797	5	6	splice	3190	61	3
vehicle	846	19	4	spambase	4601	58	2
tic-tac-toe	958	10	2	churn	5000	21	2
vowel	990	13	11	phoneme	5404	6	2
credit-g	1000	21	2	wall-robot-navigation	5456	25	4
qsar-biodeg	1055	42	2	texture	5500	41	11
cnae9	1080	857	9	optdigits	5620	65	10
PC1	1109	22	2	first-order-theorem	6118	52	6
pc4	1458	38	2	satimage	6430	37	6
cmc	1473	10	3	data.va3.gesture	9873	33	5
pc3	1563	38	2	JM1	10885	22	2
semeion	1593	257	10	letter	20000	17	26
car	1728	7	4	doushouqi-raw-egtb-2-pieces	44819	7	3
spf3	1941	28	7	bank-marketing-full	45211	17	2
mfeat-factors	2000	217	10	electricity	45312	9	2

The four tasks are as follows:

- Imbalanced data: Each dataset was randomly under-sampled to ratios such as 50:50, 75:25, 80:20, ..., up to 99:1, wherever possible.

- Semi-supervised learning: Labels for 1%, 2%, ..., up to 10% of the data were removed.
- Noisy data: For each dataset column, 1%, 2%, ..., up to 10% of the features were randomly selected and replaced with a value from a uniform distribution.
- Feature inadequacy: No modifications were made to the data.

Josef Karas was responsible for the imbalanced data and feature inadequacy tasks. Filip Chládek handled the semi-supervised learning task, and Martin Beňa took on the noisy datasets. Each member wrote the code and the 'Approach' and 'Results' sections for their respective tasks. We collaborated on the remaining parts of the report and the ideas behind all approaches.

2 XGBoost Hyper-parameter Setting

We configured the base learner for our experiments using the `n_estimators` and `num_parallel_tree` hyper-parameters. Specifically, we set `n_estimators` to 10 and `num_parallel_tree` to 10 for the RF, and `n_estimators` to 100 and `num_parallel_tree` to 1 for the DT, which is the default settings.

To ensure reproducibility, we consistently used `seed = 0` and `seed_per_iteration = True` for all experiments.

We assume that RF as a base learner may allow for more parallelization; we allocated sufficient computational resources to the XGBoost classifier. For evaluation, we utilized the Aura server at Masaryk University, which had 255 threads available[1], and we set the `n_jobs` parameter of the XGBoost classifier to 32.

Our project guidelines included specific recommendations for setting hyper-parameters when using a RF as the base learner for the XGBoost classifier. Following these recommendations, which are detailed in a provided tutorial[2], we adjusted several hyper-parameters as follows:

- `learning_rate = 0.2`
- `max_depth = 20`
- `subsample = 0.63`
- `colsample_bynode =` $\frac{\lfloor \sqrt{\#\texttt{features}} \rfloor}{\#\texttt{features}}$
- `reg_lambda = 0`
- `min_child_weight = 2`

[1]The website for the server can be found at https://www.fi.muni.cz/tech/unix/aura.html.en.

[2]See the tutorial on using RF with XGBoost at https://lorentzen.ch/index.php/2021/05/21/strong-random-forests-with-xgboost/

3 Approach

Handling categorical features with XGBoost is currently an experimental feature; therefore, we decided to use OneHotEncode for all datasets.

3.1 Imbalanced data

To address the data imbalance in our first task, we pre-processed the data using SVMSMOTE, SMOTETomek, SMOTEENN, ADASYN, and BorderlineSMOTE. These methods are sophisticated variations of SMOTE (Synthetic Minority Oversampling Technique) [2] [3] and are implemented in the imbalanced-learn library [4]. Additionally, we used a weighting strategy that assigns a weight to each instance, ensuring the sums of weights in each class are equal.

The selection of the pre-processing method and its hyper-parameters was performed using Bayesian optimization, which explored approximately 100 configurations (± 5 configurations).

The optimization was facilitated by a modified version of Auto-sklearn that supports the resampler interface introduced in [5]. It is important to note that META-learning, ensembling, data pre-processing, and feature pre-processing were disabled in our experiments. The sole classifier option was the XGBoost classifier, a newly added primitive to Auto-sklearn. We deliberately chose not to optimize any XGBoost parameters.

The use of Auto-sklearn was primarily for convenience. Similar results could potentially be achieved using SMAC3 [6], or Optuna [7] directly, which would also not require extensive coding.

3.2 Feature Inadequacy

Our approach to addressing feature inadequacy was similar to how we handled imbalanced data; however, instead of optimizing the pre-processing steps for balancing the data, we focused on optimizing the pre-processor for feature selection. Details of our feature selection techniques are listed in the referenced section Figure 2.

Similarly to our previous task, we utilized Auto-sklearn for this optimization. We deactivated META-learning, ensembling, data pre-processing, and balancing, focusing solely on the XGBoost classifier as the only algorithm option. Importantly, no XGBoost parameters were optimized in this process.

3.3 Noisy data

To address noise in the dataset, we employed an XGBoost regressor to generate continuous labels for the data. Using a threshold of 0.5, we converted these continuous labels into discrete categories (0 and 1). This newly labeled dataset was then utilized to both train and test an XGBoost classifier.

During our experiments utilizing an RF as the base learner, we initially set the hyper-parameters for regression to match those used in classification.

Figure 2: Auto-sklearn feature pre-processor hyper-parameter space, as detailed in [8].

Preprocessing method	#λ	cat (cond)	cont (cond)
Extremely randomized trees preprocessing	5	2 (–)	3 (–)
Fast ICA	4	3 (–)	1 (1)
Feature agglomeration	4	3 ()	1 (–)
Kernel PCA	5	1 (–)	4 (3)
Rand. kitchen sinks	2	–	2 (–)
Linear SVM preprocessing	3	1 (–)	2 (–)
No preprocessing	–	–	–
Nystroem sampler	5	1 (–)	4 (3)
Principal component analysis (PCA)	2	1 (–)	1 (–)
Polynomial	3	2 (–)	1 (–)
Random trees embed.	4	–	4 (–)
Select percentile	2	1 (–)	1 (–)
Select rates	3	2 (–)	1 (–)

However, we observed that the regressor classified all samples as the majority class for 11 of 14 datasets. Consequently, we reverted to using the default settings, which employ a DT as the base learner, for all regression tasks.

3.4 Semi-supervised Learning

To implement our semi-supervised learning strategy, we constructed a bagging ensemble consisting of 11 XGBoost classifiers. Each classifier in the ensemble was trained on a subset comprising half of the instances and half of the features.

Throughout 10 iterations, we continuously retrained the ensemble using all available labeled data. In each iteration, we expanded the training set by incorporating the most confidently predicted unlabeled data, assigning them the labels deemed most probable by the ensemble. We then used this augmented, fully labeled dataset to train a final XGBoost classifier. For the testing phase, only instances that retained their original labels were used to evaluate the final XGBoost classifier.

4 Results

For each task, we evaluated the performance of DT and RF across all datasets by employing the Wilcoxon signed-rank test [9]. This test determined whether there was a statistically significant difference in the performance of the algorithms. Additionally, we utilized ANOVA [10] to assess whether the degree of these dataset challenges or whether the combination of the algorithm choice and degree of the challenge significantly affected algorithm performance.

In instances where the assumptions of ANOVA were not met (normality assessed by the Lilliefors test [11] and homoscedasticity by Bartlett's test [12]), we resorted to the Kruskal-Wallis test [13], followed by Dunn's post hoc test [14] for pairwise comparisons.

Our primary evaluation metrics included AUC ROC, minority precision, minority recall, minority F1, majority F1, and macro F1—the latter being the unweighted average of minority and majority F1 scores. The focus on metrics for the minority class is justified by the inherent imbalance in the original datasets. On imbalanced datasets, the majority class typically achieves high scores across all metrics, rendering them less indicative of overall model performance.

For tasks where preprocessing was optimized using Auto-sklearn, we optimized for AUC ROC. Since this metric is threshold-independent, we selected a threshold in post-processing that maximized the TPR-FPR for threshold-dependent metrics using half of the test data, the second half of the test data was used for evaluation of the threshold-dependent metrics. Threshold independent metrics were evaluated using all of the test data.

Additionally, we analyzed the training and inference time for both algorithms across all tasks. In all cases, for training and inference time, except for Feature Inadequacy—limited by a small sample size—the Wilcoxon signed-rank test indicated a statistically significant difference, with a p-value less than 0.05, favoring DT over RF for training and inference times.

4.1 Imbalanced Datasets

Some datasets could not be under-resampled to all the imbalanced ratios. The number of datasets with specific ratios can be seen in Figure 3 in the labels above AUC ROC.

We observed a statistically significant difference in the performance of DT and RF with respect to the minority F1 score, with a statistical significance of p=0.05, using the Wilcoxon signed-rank test. Our analysis of the plot Figure 3 shows that DT performs slightly better.

Further analysis using ANOVA demonstrated that the differences in performance concerning macro F1, minority precision, minority recall, and minority F1 scores can be partially attributed to the imbalance ratio (IR) at a significance level of p=0.05.

For the majority F1 score, the assumptions required for ANOVA were not met, necessitating the use of the Kruskal-Wallis test. This test identified statistically significant effects, which can be explained by the IR.

Figure 3: Results of imbalanced data

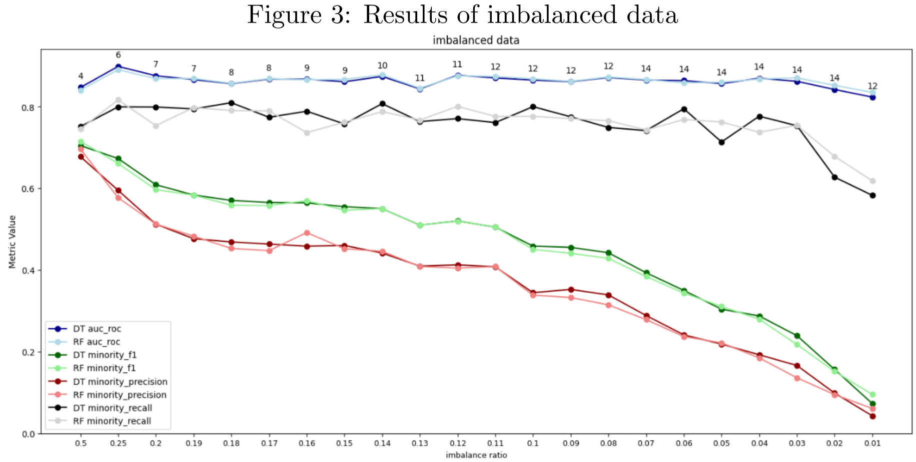


Figure 4: Time comparison of imbalanced data

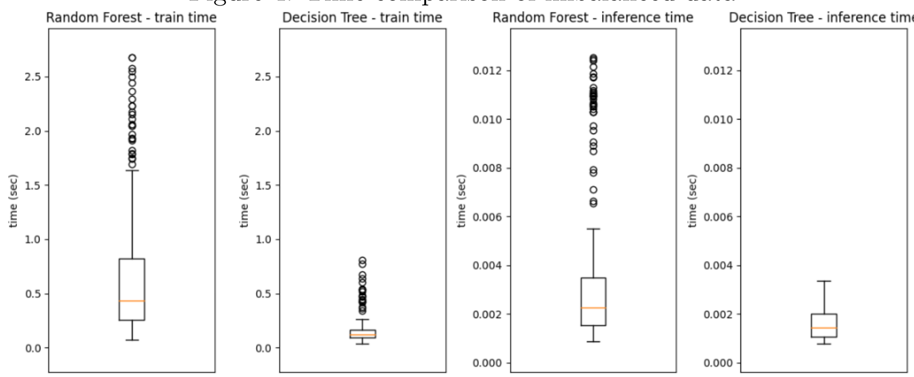
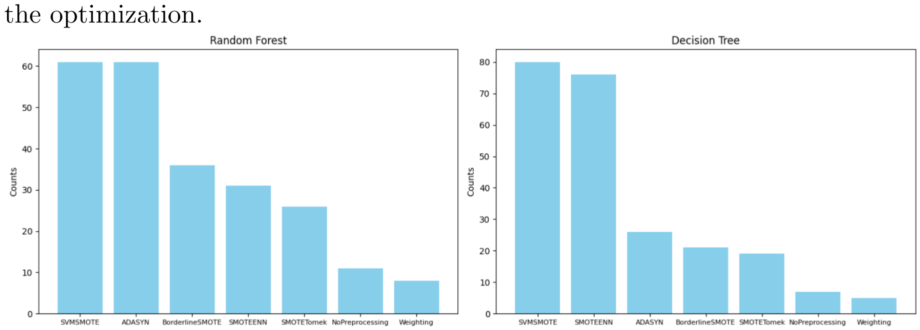


Figure 5: Histogram of pre-processing methods for imbalanced data selected by the optimization.



4.2 Noisy Data

Using the Wilcoxon signed-rank test, differences in macro F1, AUC ROC, minority precision, minority recall, and minority F1 scores are statistically significant. DT consistently outperformed RF in these metrics, as depicted in Figure 6.

Readers refer to the detailed analysis in the supplementary csv document for additional metrics.

We did not find statistically significant results when employing ANOVA to investigate the impact of the noise amount and the interaction between noise amount and algorithm choice.

Figure 6: Results of noisy data

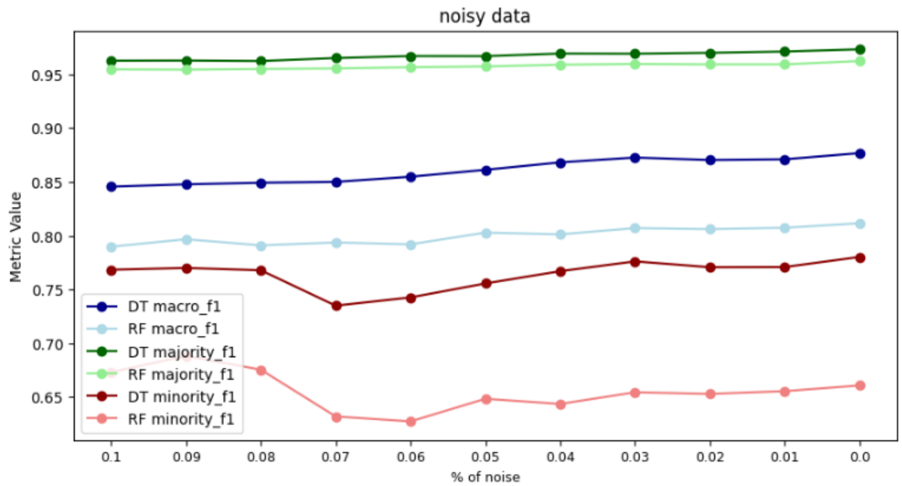
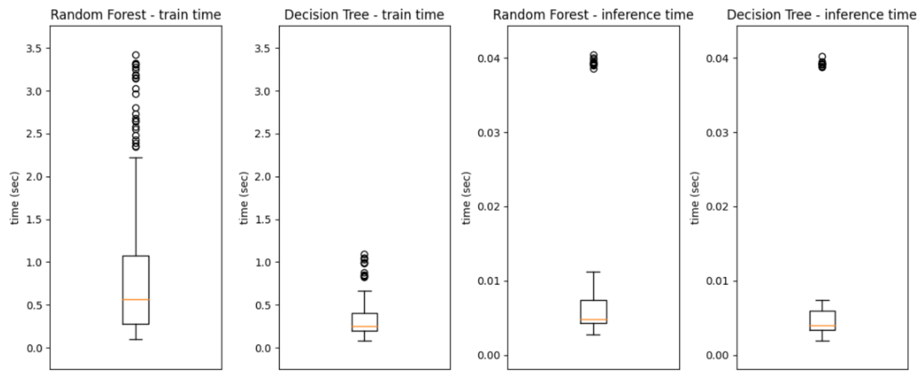


Figure 7: Time comparison of noisy data



4.3 Semi-supervised Learning

Our findings indicate statistically significant differences favoring DT in macro F1, AUC ROC, minority precision, minority recall, and minority F1 scores. These results underscore DT's superior performance in handling partially labeled datasets, as can be viewed in the relevant plots Figure 8.

Figure 8: Results of Semi-supervised Learning

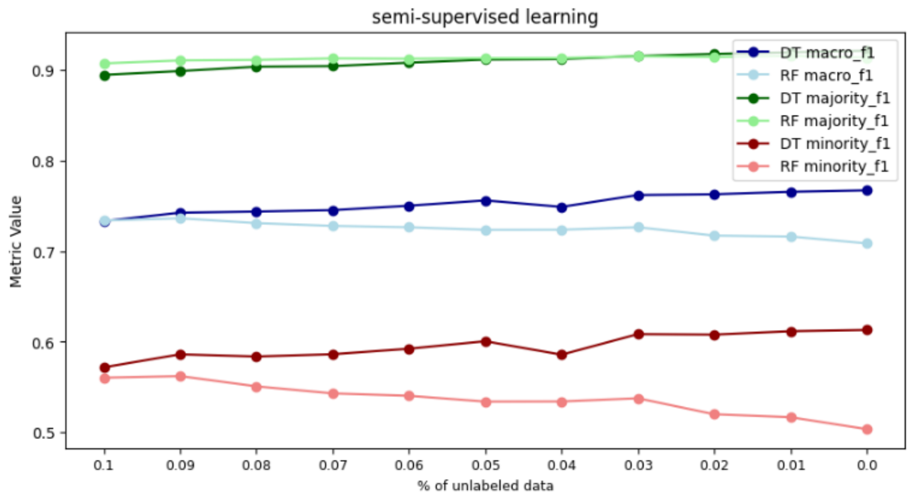
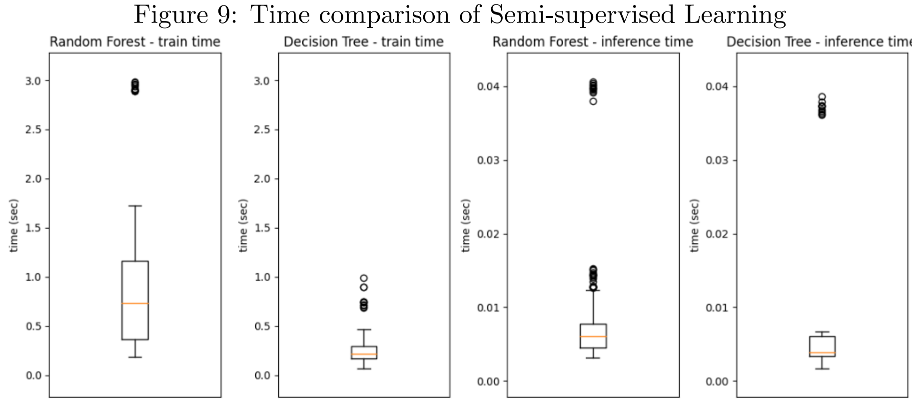


Figure 9: Time comparison of Semi-supervised Learning



Interestingly, for the majority F1 score, DT performed better than RF when the proportion of unlabeled data was less than 3%. This suggests that RF may be better suited to managing larger volumes of unlabeled data in this specific metric.

Despite these findings from the Wilcoxon test, our analysis using ANOVA revealed no significant results, indicating that the differences observed are not statistically significant when considering factors such as the interaction between the type of algorithm used and the level of unlabeled data.

4.4 Feature Inadequacy

We did not find statistically significant results in examining feature inadequacy Figure 10. This suggests that the models' performance was not notably affected by the presence or absence of adequate features under the conditions tested.

Figure 10: Results of Feature Inadequacy

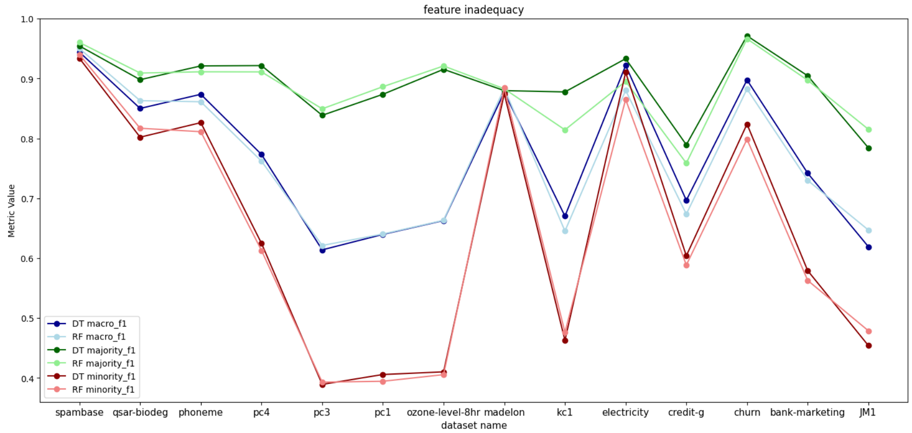
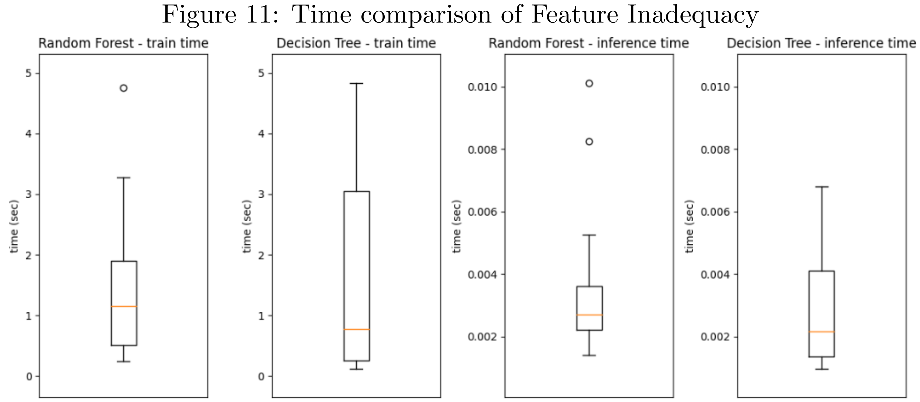


Figure 11: Time comparison of Feature Inadequacy



5 Discussion and Conclusion

We observed that with an increased proportion of unlabeled data, RF tended to perform better, whereas DT showed a decline in performance. This suggests that RF may be more robust to scenarios with high levels of unlabeled data. However, further experiments with an even greater volume of unlabeled data are necessary to confirm these trends and to explore the underlying mechanisms.

Overall, DT exhibited superior performance across various metrics and tasks. However, when pre-processing steps were optimized DT and RF demonstrated more comparable results. This indicates that the choice and optimization of pre-processing strategies can significantly influence the efficacy of these models.

In the noisy data experiments, the RF regressor performed poorly, classifying almost all samples as the majority class. This necessitated a switch to the DT regressor, which yielded better performance under noisy conditions.

Regarding the time performance, the disparity is likely influenced by the configuration of tree-specific parameters such as *max_depth*, *reg_lambda*, and *min_child_weight*. These settings affect the depth and branching of the trees, which can significantly impact both the computational burden and the effectiveness of the learning process.

For future work, it would be beneficial to explore the optimization of hyperparameters for the XGBoost models, including adjustments to the number of parallel trees and estimators. These modifications could potentially enhance model performance and provide deeper insights into the optimal configurations for different types of data challenges.

References

[1] 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, pages 785–794, New York, NY, USA, 2016. ACM.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Radovan Haluška, Jan Brabec, and Tomáš Komárek. Benchmark of data preprocessing methods for imbalanced classification. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2970–2979. IEEE, 2022.

[4] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[5] Josef Karas. Automl and data preprocessing [online], [cit. 2024-05-23]. SUPERVISOR : Lubomír Popelínský.

[6] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[7] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[8] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[9] Robert F Woolson. Wilcoxon signed-rank test. *Encyclopedia of Biostatistics*, 8, 2005.

[10] Lars St, Svante Wold, et al. Analysis of variance (anova). *Chemometrics and intelligent laboratory systems*, 6(4):259–272, 1989.

[11] Mehmet Mendes and Akin Pala. Type i error rate and power of three normality tests. *Pakistan Journal of Information and Technology*, 2(2):135–139, 2003.

[12] Hossein Arsham and Miodrag Lovric. Bartlett's test. *International encyclopedia of statistical science*, 2:20–23, 2011.

[13] Patrick E McKight and Julius Najab. Kruskal-wallis test. *The corsini encyclopedia of psychology*, pages 1–1, 2010.

[14] Alan C Elliott and Linda S Hynan. A sas® macro implementation of a multiple comparison post hoc test for a kruskal–wallis analysis. *Computer methods and programs in biomedicine*, 102(1):75–80, 2011.