Enhancing Semi-Supervised Learning with a Meta-Feature Based Safeguard System

Martin Schumann

Ludwig-Maximilians-University Munich
Geschwister-Scholl-Platz 1, 80539 Munich, Germany
martin.schumann@campus.lmu.de

Abstract. The selection of Meta-Features can strongly influence performance and accuracy of semi-supervised learning methods. In this paper, we analyze the role of different meta-features in enhancing the performance of semi-supervised learning models. We present experiments with benchmark data sets, which meta-features contribute most significantly to model accuracy and robustness.
We propose an enhanced safeguard system for semi-supervised learning that leverages meta-features to predict the potential benefits of pseudo-labeling, with a focus on simultaneously reducing computational resource consumption and improving the overall performance of semi-supervised learning models. By determining when to train a second predictor, our system optimizes computational efficiency, thereby minimizing energy usage and the associated carbon footprint. Thus, this approach emphasizes the importance of developing resource-conscientious machine learning methodologies that contribute to the broader goal of sustainable technology.

Keywords: AutoML, Machine Learning, semi-supervised learning, Meta-Features

1 Introduction

One of the biggest problems in machine learning (ML) and artificial intelligence (AI) research is the scarcity of labeled data, which are required to train many models. However, most useful data models do not have labels, which requires expensive [22,16], time-consuming, labor-intensive, and not easily scalable human labeling, which is prone to errors [30,22].

For ML, a larger amount of correct labeled data almost always leads to higher accuracy results of the model [16]. This is not a problem when ML is done with unsupervised learning, because it does not require labeled data, and instead identifies patterns directly [4,22]. However, it lacks “ground truth” for evaluation and is sensitive to how data is input and used (e.g., introducing biases) [26,7]. Supervised learning, which relies on labeled data, and is used e.g., for prediction, does not have some of the limitations that unsupervised learning has, but has others, including sensitivity to the quality of labels. As a result, ML research often relies on small, well-labeled datasets like the IMDB dataset [21] or MNIST [19]. Canned benchmark datasets however, can contain incorrect [29], biased [6], or incomplete information [29]. Furthermore, older data might not correspond to changes in the real world [9].

To deal with the problems arising from this lack of up-to-date correct labeled training data, the concept of self-training was introduced [17,27]. One approach works by having one supervised learning model create "pseudo-labels" for unlabeled data, which is then used, together with existing "human" labels, to train a second supervised learning model. This reduces the need for massive amounts of labeled data. To streamline this process, we utilize AutoML (Automated Machine Learning)-based libraries, which are able to automate many parts of an ML workflow [13].

However, pseudo-labeling introduces risks, as inaccurate labels can degrade performance. We therefore ask our main research question: can an AutoML-trained ML model achieve better performance by utilizing previously unlabeled data, which a separately (AutoML) trained model has labeled?

In order to address this question, we have developed an approach that uses meta-features to decide if the newly generated pseudo-label is likely to lead to an improvement of performance. Our safeguard system is designed to evaluate the behavior of the previously trained model using our meta-feature approach. It will decide if this model be further used for labeling or not. In the latter case, the original model will be kept as the newly trained one is likely to result in worse performance.

To study our approach in detail, we split up our initial question into two research questions: RQ1) Does our approach perform favorably to AutoGluon [11]'s built in support for semi-supervised learning? and RQ2) Which models and settings influence the performance of the safeguard system?

The remainder of the paper is structured as follows: In Sec. 2, we briefly describe AutoML. In Sec. 3, we describe our approach, including our improved AutoML pipeline and its safeguard system. In Sec. 4, we answer the 2 research questions and present experimental results. Sec. 5 discusses related work and Sec. 6 concludes the paper and outlines potential directions for future work.

2 Background

AutoML (Automated Machine Learning) as a concept describes the process of automating and simplifying ML tasks and workflows. There is a wide field of libraries, which accomplish simplification by automating a lot of the time-consuming and tedious process of setting up an ML pipeline [13].

One such library is AutoGluon, which promises to be very easy to use while having competitive performance. It can achieve this by "ensembling multiple models and stacking them in multiple layers" [11], e.g., by combining the output of multiple models for one prediction. We chose it as it is one of the best performing AutoML libraries on current benchmarks [15], high quality of models, and ease of use.

3 Our Approach

3.1 The Improved AutoML Pipeline

In this section, we introduce our linear-ensembling pipeline (Fig. 1), consisting of the two models M1 and M2 to be trained. Each model is a classifier trained using AutoGluon ("AutoML" in Fig. 1). The models are trained one after the other and the pipeline returns M2 as the final model. The pipeline utilizes three subsets of one dataset.

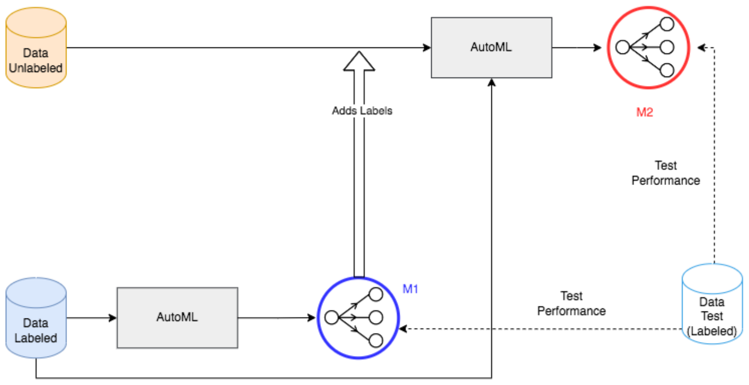


Fig. 1: Graphical representation of the pipeline. M1 is the first model (labeler), M2 is the second model.

M1 is trained on labeled data and then pseudo-labels all the unlabeled data, keeping only the data points above a certain confidence threshold. It predicts a probability distribution for the possible labels for each data point and if no label meets the threshold, the data point is not used for further training. This pseudo-labeling step is the "Adds labels" process in Fig. 1 and is shown in more detail in Fig. 2. All the newly pseudo-labeled data and labeled data are combined into a new dataset used to train the model M2.

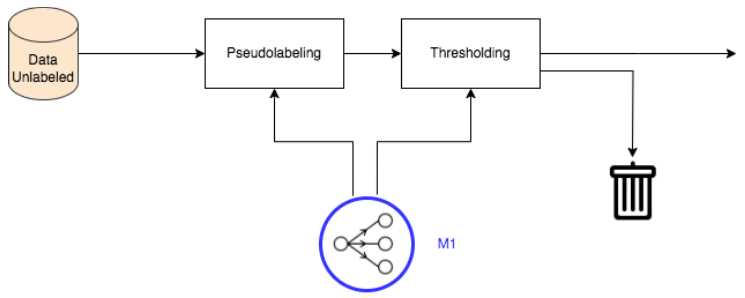


Fig. 2: Graphical representation of the “Adds labels” process from Fig. 1. Data with pseudo-labels that do not meet a certain threshold get “thrown away” (trashcan).

As it is possible that M2 sometimes performs worse than M1, we introduce a safeguard system (Sec. 3.3) to overcome this problem.

3.2 Meta-Features

Meta-features are calculated measures of a dataset that describe aspects of the dataset [1], and can be used to predict the performance of ML algorithms, among other things [18].

Alcobaça et al., the authors of an implementation of meta features for python (pymfe [1]), provide a lot of different meta features to choose from. These meta-features can be split into multiple groups, including, but not limited to: model based (“measures extracted from a [simple] model” [25], landmarking (“measures that use the performance of simple and fast learning algorithms to characterize datasets” [25]), and clustering (characteristics of dataset clusters) [1,25].

In this work we utilize meta-features for our safeguard system, to be able to predict if training with “pseudo-labeled” data will lead to an improvement of performance.

3.3 Safeguard System

Adding “pseudo-labeled” data can hurt model performance, due to confirmation bias [2], where incorrect labels from M1 lead to M2 learning from poor predictions. This means that M2 might perform worse than M1, which was trained only on correctly labeled data. We have decided to alleviate the problem of deteriorated performance and the high cost of training by introducing a safeguard system.

This system predicts whether a given M2 model will perform better or worse than the M1 model. Then, depending on settings, pseudo-labeling of data and the training of M2 will not occur. As can be seen in Fig. 3, prediction is done by extracting meta-features from the initially labeled dataset, and then returning whether M2 would perform better. If M2 is predicted to perform better, the safeguard system returns “plus”, otherwise it returns “minus”.

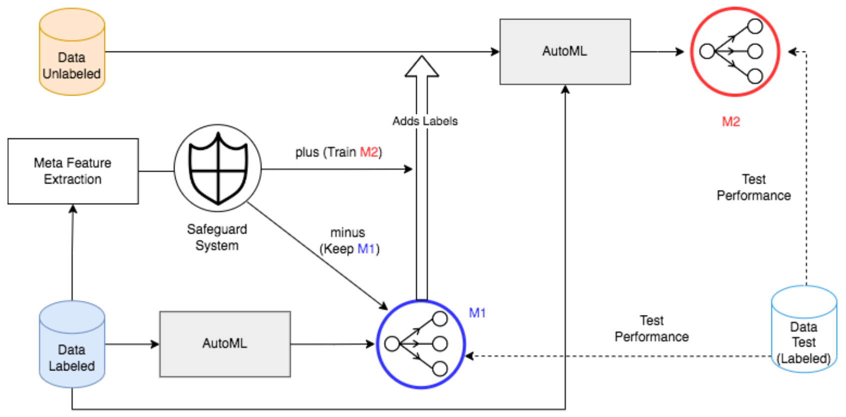


Fig. 3: Pipeline with safeguard system included. Pseudo-labeling and training of M2 only occurs if the safeguard system allows it. Otherwise M1 is kept as the main model.

This system provides two major benefits. First, it is able to give insight into when and how an ML model performs better or worse after being trained with pseudo-labeled data, and for which datasets this occurs. Second, it saves time and compute resources as it can be used to provide a quicker and more efficient insight into whether a dataset will perform worse with M2, and so M2 does not even need to be trained.

The safeguard system is implemented both using scikit-learn [23]'s RandomForestClassifier and AutoGluon. This is to make evaluating various performance differences easier, as is described in Subsec. 4.2. Both systems are trained using the same procedure.

4 Experiments and Results

We run tests to compare whether utilizing pseudo-labeled data can improve the performance of a classifier and we train and benchmark our safeguard system.

We run both of these tasks on the LRZ's interactive and serial Linux clusters [24]. We do not utilize GPUs for our work, as they bring no huge performance benefit for our datasets (tabular data). We utilize python 3.10 from anaconda3/2022.10 (part of spack/23.1.0).

For AutoGluon's fit method we keep the most of the default settings ([12]), except for increasing the time limit to 600 seconds and changing the quality parameter. We primarily use AutoGluon's medium quality setting (which defines the speed, quality and size of models) for its balance of speed and performance, with high quality tests to validate our findings [12]. We decided not to utilize best quality as it consumes a lot of resources. The results of datasets trained using the medium quality setting are used when training the safeguard system. For pymfe, we also add a time limit of 600 seconds. If the time limit is reached, our safeguard system method errors out and linear-ensembling continues with training M2.

For the datasets, we utilize the widely used [5] OpenML100 [5] datasets and its successors, the OpenML-CC18 [5] datasets as a baseline for performance comparison. For more variety and to see how the implementation functions on larger and more diverse datasets, we utilize some datasets which were compiled by Feurer et al. [14]. We put them in 3 dataset groups and named them CD1 [14], CD2 [14], and CD3 [14].

4.1 Linear-Ensembling versus Autogluon's Built In Semi-Supervised Learning Support (RQ1)

In this section, we answer the question: Does our linear-ensembling approach perform favorably to AutoGluon's built in support for semi-supervised learning?

We want to compare our approach to two different variations. The first variation is comparing our approach (linear-ensembling, S1) to AutoGluon's support for semi-supervised learning (S2). The second variation is a hybrid approach (S3), where we use AutoGluon's support to train M1 and then use our approach to train M2. We also compare S1's M1 model to S2's model, as the safeguard system is not in effect, so sometimes the S1's M1 model might have better results than S1's M2 model. We compare the performance using the "accuracy" measurement.

As we can see in Tab. 1, the performance is almost always better after linear-ensembling ("S1 (M2) vs S3 (M2)") and always better than just AutoGluon's semi-supervised support ("S1 (M1) vs S3 (M2)"). However, the performance of S1's M1 model compared to the S2 model is mixed ("S1 (M1) vs S2"). For "S1 (M2) vs S2", S2 is sometimes better than S1's M2, but not always. This means that the purely semi-supervised support is sometimes better than our approach, excluding the safeguard system. What we have learned is that semi-supervised support has worse performance than our linear-ensembling approach, and it provides marginal benefits utilizing it for training M1.

4.2 Safeguard System (RQ2)

In this section, we answer the research question: which models and settings influence the performance of the safeguard system?

We compare the performance along 2 directions. The first is which classifier we use for the safeguard system, scikit-learn's RandomForestClassifier (Sec. 4.2), or AutoGluon's TabularPredictor (Sec. 4.2). For the AutoGluon classifier we also look at different metrics, including a custom metric which weighs false negatives or false positives heavily and is based on [28].

The second direction is looking at meta-feature performance and relevance. To measure the benefit of the safeguard system, we measure the mean accuracy with and without utilizing the safeguard system. If the mean accuracy is better or the same, this means the safeguard system has a positive benefit overall, if not,

Table 1: Comparison results of S1 with S2 and S3 across various datasets and model configurations.

Datasets	Comparison	$\lvert S1\rvert$ better	$\lvert S2\rvert$ better	$\lvert S3\rvert$ better	$\lvert EqualPerf\rvert$
`OpenML100`	S1 (M1) vs S2	15	21	-	43
	S1 (M1) vs S3 (M2)	30	-	24	25
	S1 (M2) vs S3 (M2)	19	-	16	44
	S1 (M2) vs S2	31	27	-	21
`OpenML-CC18`	S1 (M1) vs S2	12	14	-	28
	S1 (M1) vs S3 (M2)	20	-	18	16
	S1 (M2) vs S3 (M2)	9	-	16	29
	S1 (M2) vs S2	25	18	-	11
`CD1`	S1 (M1) vs S2	11	2	-	12
	S1 (M1) vs S3 (M2)	17	-	6	2
	S1 (M2) vs S3 (M2)	11	-	5	9
	S1 (M2) vs S2	9	14	-	2
`CD2`	S1 (M1) vs S2	39	28	-	24
	S1 (M1) vs S3 (M2)	36	-	28	27
	S1 (M2) vs S3 (M2)	35	-	28	28
	S1 (M2) vs S2	33	37	-	21
`CD3`	S1 (M1) vs S2	6	8	-	8
	S1 (M1) vs S3 (M2)	7	-	3	12
	S1 (M2) vs S3 (M2)	6	-	2	14
	S1 (M2) vs S2	7	7	-	8

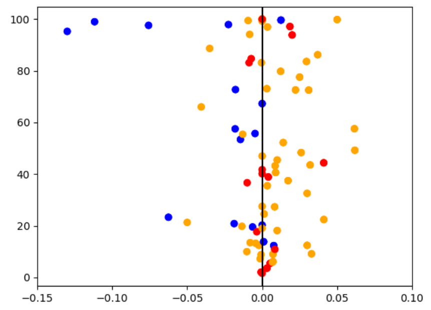
it has a negative benefit. Over a total of 32 runs, 8 runs had worse performance. This includes versions of the safeguard system with less than optimal settings.

When we compare AutoGluon to scikit-learn by looking at Fig. 4 A vs Fig. 4 B, we can see that AutoGluon barely makes any "minus" (blue circle) predictions, i.e., it almost never thinks that the performance will be decrease, which makes it pretty useless as a predictor.

AutoGluon AutoGluon's TabularPredictor showed limited effectiveness as a safeguard system, with 3 out of 7 runs resulting in worse performance. The classifier predominantly predicted "plus" (i.e., a performance improvement) regardless of actual outcomes. We attempted to improve its behavior through various approaches: 1) heavily weighting false positives (Fig. 5 A) 2) using f_β scoring with $\beta = 2$ (Fig. 5 B) and $\beta = 0.5$ (Fig. 5 C) and 3) using F1 scoring (Fig. 5 D). We used the same experimental setup: trained on OpenML100 and OpenML-CC18, run on CD3, utilizing model-based meta-features.

None of these changes significantly improved performance. For both landmarking and clustering meta-features, we can even see that the mean performance is identical with and without the safeguard system, with all four being 0.6675.

In conclusion, the AutoGluon predictor does not seem to be the optimal solution for our safeguard system. We therefore look at scikit-learn's RandomForestClassifier in the next section.



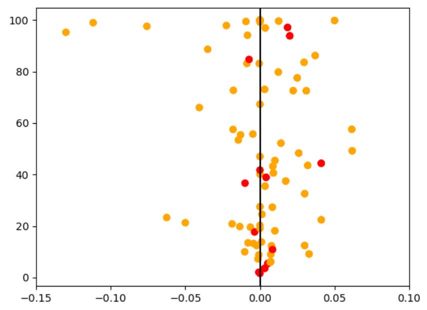


Fig. 4: Percentage of data ignored vs change in accuracy ($accuracy_{M2} - accuracy_{M1}$) for different models: RandomForestClassifier (A) and AutoGluon (B). In all figures: each dot represents a dataset. Orange is classifier predicting a positive change, Blue a negative change. Red is an error with the safeguard system.

RandomForestClassifier In this section, we take a look at the performance of the safeguard system implemented using scikit-learn's RandomForestClassifier [8]. For the 25 runs using the RandomForestClassifier, five had worse performance. This includes versions of the safeguard system with less than optimal settings, four of which had worse performance.

As we can see in the following table (Tab. 2), the scikit-learn based safeguard system almost always performs better with it than without.

Table 2: AutoGluon: Comparison of Mean with safeguard system (SfgS) and Mean. "Is Better?" being "True" means that the mean with safeguard system is greater than the mean without it.

Meta-Features and Datasets	Mean with SfgS	Mean	Is Better?
Clustering	0.7184	0.7103	True
Model-Based + Clustering: Run on OpenML100	0.7831	0.7764	True
Model-Based + Clustering: Run on CD3	0.7121	0.7019	True
Landmarking	0.7160	0.7101	True
Model-Based $\rightarrow$ Bad Performance	0.6953	0.6966	False
Model-Based $\rightarrow$ Good Performance	0.7918	0.7867	True

There are some examples where the safeguard system did not perform better in earlier tests, but by changing our code to use bootstrap=False the performance is better. We can see the benefits of no bootstrapping in the following graphs (Fig. 6 A and Fig. 6 B), where the classifier predicts more datasets correctly.

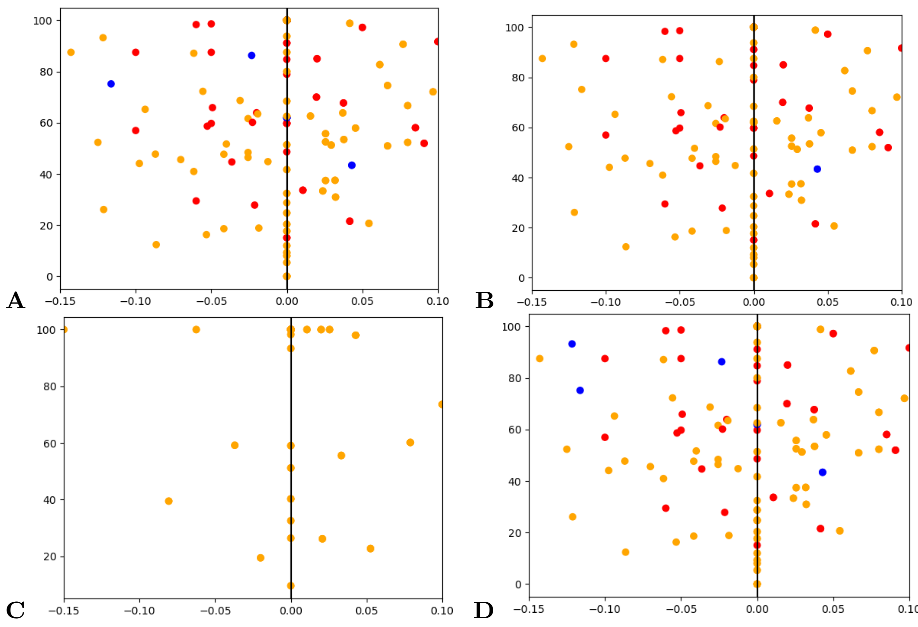


Fig. 5: Percentage of data ignored vs. change in accuracy ($accuracy_{M2} - accuracy_{M1}$) for different metrics. A: Custom Metric, B: f_β metric ($\beta = 2$), C: F1 metric, D: f_β metric ($\beta = 0.5$).

4.3 Conclusion

In conclusion, scikit-learn's RandomForestClassifier with bootstrap=False proved superior to AutoGluon's approach, offering better prediction accuracy and simpler tuning capabilities. Our safeguard system provides a net benefit for most applications, by avoiding the need for computing M2, with minimal overhead from the RandomForestClassifier. Furthermore, we have found that our approach compares favorably to AutoGluon's built-in semi-supervised learning support.

Finally, we have explored how to predict dataset performance using meta-features and identified model-based and clustering meta-features as the best for our approach.

Overall, our results demonstrate that better performance can be achieved by utilizing unlabeled data, which has been labeled by a separately trained model.

5 Related Work

There are two main papers that focus on unifying AutoML and semi-supervised learning. The first, by Van Engelen et al. introduces the term co-ensembling [10], where K classifiers are trained on labeled data, and $K - 1$ of them generate pseudo-labels for unlabeled data. This combined data, selected based on a minimum probability threshold, is then used to train the K-th original classifier further. Although this process can be multistep, their findings conclude that

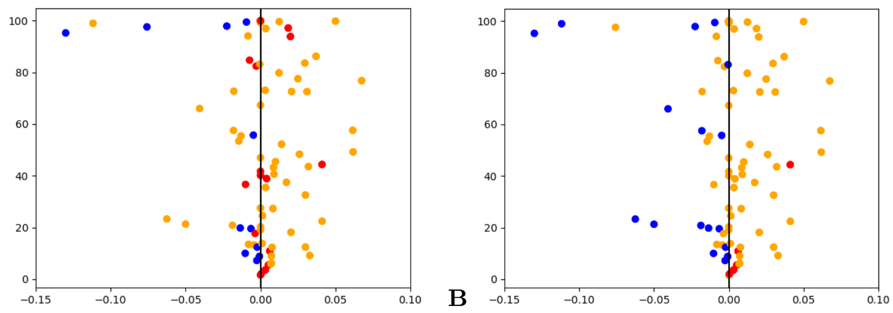


Fig. 6: scikit-learn: Percentage of data ignored vs change in accuracy with (A) and without bootstrapping (B)

single-step is better, as multiple steps can lead to errors at the beginning being multiplied.

The second, by Li et al., introduces the term AutoSSL (for auto semi-supervised learning) to describe unifying the two approaches (AutoML and SSL) from the view of SSL. The authors use a clustering approach to characterize the data, and use meta-features of the data to inform the "automated learning" [20] process. This process utilizes well-known SSL algorithms and combines it with AutoML techniques to tune the hyperparameters of these algorithms. This approach is compared to a normal auto-sklearn [13] approach, which the authors claim has "highly competitive" [20] performance compared to classic SSL techniques. Li et al.'s paper is also referenced by [10] as having a different approach, although both papers focus on how to combine AutoML and semi-supervised learning, with the differences lying in the angle from which the authors tackle the problem.

AutoGluon also has experimental support for "unlabeled_data" via the experimental "FT_TRANSFORMER" [3, #L504] model, which "does not scale well to >100 features." [3, #L504], limiting the usability compared to our approach. There is also some support for "pseudo-labeling" [3, #L1904], which labels data above a certain probability threshold and uses it as extra training data to refit the original model.

6 Conclusions

We have shown how semi-supervised models can have better performance when trained on small amounts of labeled data. Through our novel approach, known as linear-ensembling, we are able to provide better performance overall, compared to both a supervised learning and AutoGluon's semi-supervised approach, despite not achieving 100% accuracy.

We have investigated which factors could lead to poor performance at the end of a learning process, such as the threshold or AutoGluon's quality configurations and the meta-features of the dataset to be trained on. We have identified clustering and model-based meta-features as particularly meaningful in predicting how a dataset will perform. To take advantage of this information, we have introduced our safeguard system. This system is able to leverage the improved accuracy from our linear-ensembling approach, while reducing computational resources and run times by eliminating unnecessary model training. Furthermore, our findings have shown that the issues with performance loss can be eliminated without time-consuming steps needing to be taken.

Finally, we provide valuable guidance for future research, by identifying which settings and meta-features are the most useful and provide the best results. This includes which factors affect the performance of the safeguard system itself, such as the choice between scikit-learn and AutoGluon. These insights offer a clear direction forward for further research with diverse datasets and applications.

There are several directions for future work. A further improvement and generalization of the safeguard system can include training it on more datasets, tuning the various settings of the RandomForestClassifier, or possibly selecting a different classifier model. Another area of research could be further refining the selection of the meta-features and increasing the robustness of the pymfe system to make it more useful for the safeguard system. A third direction concerns a more hybrid approach, where a human would verify/change some pseudo-labeled results, e.g., the lowest confidence ones, to improve the performance of M2. The fourth direction is expanding our work to non-classifier based labeling systems with AutoML.

References

1. Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L.P.F., Oliva, J.T., de Carvalho, A.C.P.L.F.: Mfe: Towards reproducible meta-feature extraction. Journal of Machine Learning Research **21**(111), 1–5 (2020)
2. Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K.: Pseudo-labeling and confirmation bias in deep semi-supervised learning. In: Proc IJCNN, pp. 1–8 (2020)
3. Autogluon authors: predictor.py, `https://github.com/autogluon/autogluon/blob/0e3bc0e54ab4b0edf865c8b99e1418472006d6b7/tabular/src/autogluon/tabular/predictor/predictor.py`, accessed on 26-07-2024
4. Barlow, H.B.: Unsupervised learning. Neural computation **1**(3), 295–311 (1989)
5. Bischl, B., Casalicchio, G., Feurer, M., Gijsbers, P., Hutter, F., Lang, M., Mantovani, R.G., van Rijn, J.N., Vanschoren, J.: OpenML benchmarking suites (2021)
6. Carlisle, M.: A boston housing dataset controversy (2019), `https://medium.com/@docintangible/racist-data-destruction-113e3eff54a8`
7. Chouldechova, A., Roth, A.: The frontiers of fairness in machine learning. arXiv preprint arXiv:1810.08810 (2018)
8. scikit-learn developers: Randomforestclassifier, `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
9. Ding, F., Hardt, M., Miller, J., Schmidt, L.: Retiring adult: New datasets for fair machine learning. Neurips **34**, 6478–6490 (2021)
10. van Engelen, J.E., Hoos, H.H.: Semi-supervised co-ensembling for automl. In: Trustworthy AI - Integrating Learning, Optimization and Reasoning. pp. 229–250. Springer (2021)

11. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., Smola, A.: Autogluon-tabular: Robust and accurate AutoML for structured data. arXiv preprint arXiv:2003.06505 (2020)
12. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., Smola, A.: autogluon.tabular.TabularPredictor.fit (2024), `https://auto.gluon.ai/stable/api/autogluon.tabular.TabularPredictor.fit.html`
13. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Neurips 28 (2015). pp. 2962–2970 (2015)
14. Fusi, N., Sheth, R., Elibol, M.: Probabilistic matrix factorization for automated machine learning. In: Neurips Vol. 31 (2018)
15. Gijsbers, P., Bueno, M.L.P., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., Vanschoren, J.: Amlb: an AutoML benchmark (2023), `https://arxiv.org/abs/2207.12560`
16. Halevy, A., Norvig, P., Pereira, F.: The unreasonable effectiveness of data. IEEE intelligent systems **24**(2), 8–12 (2009)
17. He, J., Gu, J., Shen, J., Ranzato, M.: Revisiting self-training for neural sequence generation. arXiv preprint arXiv:1909.13788 (2019)
18. Kotlar, M., Punt, M., Radivojević, Z., Cvetanović, M., Milutinovic, V.: Novel meta-features for automated machine learning model selection in anomaly detection. IEEE Access **9**, 89675–89687 (2021). `https://doi.org/10.1109/ACCESS.2021.3090936`
19. LeCun, Y.: The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/` (1998)
20. Li, Y.F., Wang, H., Wei, T., Tu, W.W.: Towards automated semi-supervised learning. In: Proceedings of the AAAI Conf. on AI. vol. 33, pp. 4237–4244 (2019)
21. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proc. 49th Annual Meeting of the Association for Computational Linguistics. pp. 142–150. (6 2011)
22. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT press. (2012)
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
24. Leibnitz Rechenzentrum: Available slurm clusters and features, `https://doku.lrz.de/available-slurm-clusters-and-features-11483939.html`
25. Rivolli, A., Garcia, L.P., Soares, C., Vanschoren, J., de Carvalho, A.C.: Towards reproducible empirical research in meta-learning. arXiv preprint arXiv:1808.10406 pp. 32–52 (2018)
26. Schwartz, R., Vassilev, A., Greene, K., Perine, L., Burt, A., Hall, P., et al.: Towards a standard for identifying and managing bias in artificial intelligence. NIST special publication **1270**(10.6028) (2022)
27. Scudder, H.: Probability of error of some adaptive pattern-recognition machines. IEEE Transactions on Information Theory **11**(3), 363–371 (1965)
28. Thai-Nghe, N., Gantner, Z., Schmidt-Thieme, L.: Cost-sensitive learning methods for imbalanced data. In: Proc. IJCNN. pp. 1–8. IEEE (2010)
29. Verma, S., Ernst, M., Just, R.: Removing biased data to improve fairness and accuracy. arXiv preprint arXiv:2102.03054 (2021)
30. Yao, Q., Wang, M., Chen, Y., Dai, W., Li, Y.F., Tu, W.W., Yang, Q., Yu, Y.: Taking human out of learning applications: A survey on automated machine learning. arXiv preprint arXiv:1810.13306 (2018)